A Flexible Optimization Framework for Regularized Matrix-Tensor Factorizations with Linear Couplings

Carla Schenker, Jeremy E. Cohen, *Member, IEEE* and Evrim Acar, *Member, IEEE*

***Abstract*—Coupled matrix and tensor factorizations (CMTF) are frequently used to jointly analyze data from multiple sources, a task also called data fusion. However, different characteristics of datasets stemming from multiple sources pose many challenges in data fusion and require to employ various regularizations, constraints, loss functions and different types of coupling structures between datasets. In this paper, we propose a flexible algorithmic framework for coupled matrix and tensor factorizations which utilizes Alternating Optimization (AO) and the Alternating Direction Method of Multipliers (ADMM). The framework facilitates the use of a variety of constraints, loss functions and couplings with linear transformations in a seamless way. Numerical experiments on simulated and real datasets demonstrate that the proposed approach is accurate, and computationally efficient with comparable or better performance than available CMTF methods for Frobenius norm loss, while being more flexible. Using Kullback-Leibler divergence on count data, we demonstrate that the algorithm yields accurate results also for other loss functions.**

***Index Terms*—tensor factorizations, coupled tensor factorizations, linear couplings, AO-ADMM.**

I. Introduction

In many areas of science, various sensing technologies are used to obtain information about a single system of interest. Often, none of the datasets alone contains a complete view of the system, but the data measured from different modalities can complement each other. For instance, brain activity patterns can be captured using both electroencephalography (EEG) and functional magnetic resonance imaging (fMRI) signals, which have complementary temporal and spatial resolutions. Similarly, in metabolomics, multiple analytical techniques such as LC-MS (Liquid Chromatography - Mass Spectrometry) and NMR (Nuclear Magnetic Resonance) spectroscopy are used to measure chemical compounds in biological samples, providing a more complete picture of underlying biological processes. Joint analysis of datasets from multiple sources, also referred to as data fusion (or multi-modal data mining), exploits these complementary measurements, and allows for better interpretability and, potentially, more accurate recovery of patterns characterizing the underlying phenomena. Nevertheless, data fusion poses many challenges, and there is an emerging need for data fusion methods that can take into account different characteristics of data from multiple sources in many disciplines [1]–[4].

C. Schenker is with the Simula Metropolitan Center for Digital Engineering and Oslo Metropolitan University, Oslo, Norway (e-mail: carla@simula.no).
J. E. Cohen is with CNRS, Université de Rennes, Inria, CNRS, IRISA.
E. Acar is with the Simula Metropolitan Center for Digital Engineering, Oslo, Norway.

Data from multiple sources can often be represented in the form of matrices and higher-order tensors. Coupled matrix and tensor factorizations (CMTF) are an effective approach for joint analysis of such datasets in many domains including social network analysis [5]–[8], neuroscience [9]–[13], and chemometrics [2], [14]. In such coupled factorizations, each dataset is modelled by a low-rank approximation (Fig. 1). One of the most popular tensor factorization methods, Canonical Polyadic Decomposition (CPD) (also known as CANDECOMP/PARAFAC (CP)) [15]–[17] models tensor $\boldsymbol{\mathcal{T}}$ of order D as the sum of R (usually a small number) rank-one components,

$$\boldsymbol{\mathcal{T}} \approx \sum_{r=1}^{R} \mathbf{C}_1(:,r) \circ \mathbf{C}_2(:,r) \circ \ldots \circ \mathbf{C}_D(:,r) =: [\![\mathbf{C}_d]\!]_{d=1}^{D}, \quad (1)$$

where $\mathbf{C}_d(:,r)$ is the rth column of factor matrix in the dth mode, and $\circ$ denotes the vector outer product. Those components may reveal patterns of interest in the data. Coupled factorization problems are often formulated by extracting the same latent factors from the coupled mode [6]–[9], [18], as in the following formulation, where a third-order tensor $\boldsymbol{\mathcal{T}}_1$ is coupled with a matrix $\mathbf{T}_2$ in the first mode:

$$\begin{aligned} \min_{\{\mathbf{C}_{i,d}\}_{\substack{i=1,2 \\ d \le D_i}}} \quad & \| \boldsymbol{\mathcal{T}}_1 - [\![\mathbf{C}_{1,1}, \mathbf{C}_{1,2}, \mathbf{C}_{1,3}]\!] \|_F^2 + \| \mathbf{T}_2 - \mathbf{C}_{2,1}\mathbf{C}_{2,2}^T \|_F^2 \\ \text{s.t.} \quad & \mathbf{C}_{1,1} = \mathbf{C}_{2,1} \end{aligned} \quad (2)$$

where $\mathbf{C}_{i,d}$ denotes the factor matrix in the dth mode for dataset i, and D_i is the order of dataset i.

However, factors corresponding to the coupled mode are not necessarily equal in datasets from different modalities. Therefore, in this paper, we focus on more general linear coupling relationships. In particular, we cover the cases, where not all factors are shared between tensors [2], [11], [19]–[23]. For the time being, we assume that the number of shared components is known or estimated beforehand. Moreover, linear transformations are also used to couple factors that are sampled from a continuous phenomenon with different sampling rates [24] or aggregation intervals [25]. Using such linear couplings, different resolutions of EEG and fMRI signals have been previously incorporated while jointly analyzing data from these two modalities [10], [11]. Linear couplings for spatial and spectral transformations also appear in the context of hyperspectral super-resolution [26].

In order to address the challenges in data fusion applications, in addition to more general couplings, the formulation in (2) needs to be further extended to incorporate various constraints and loss functions. In many applications, it is

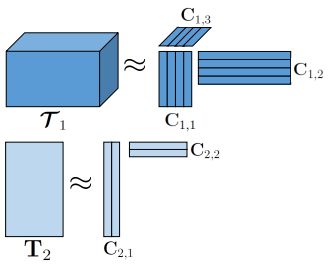


Fig. 1. Illustration of data tensors and their decompositions.

necessary to impose certain constraints or regularizations on the latent factors to obtain physically meaningful and identifiable patterns. Popular constraints include non-negativity [7], [10], [19], [27], sparsity [20], [28] and smoothness [29]. Furthermore, while Frobenius norm-based loss functions have so far been common in coupled factorizations, this loss function implicitly assumes that entries in the data tensor $\mathcal{T}_i$ are normally distributed around their mean parameterized by the CPD model. However, for nonnegative, discrete or binary data, this assumption is usually not valid and other loss functions yield more accurate results [5], [7], [30], [31].

Existing algorithmic approaches for coupled matrix and tensor factorizations can incorporate constraints, linear couplings and different loss functions but none of them has shown to be flexible enough to incorporate all. For instance, Alternating least squares (ALS)-based approaches, solving for one factor matrix at a time by fixing the others, are one of the most commonly used techniques for solving (2) [9], [32], with linear couplings as described in [24], [26], [33] or with various constraints [20], [34]. However, they are limited to Frobenius norm-based loss functions. Other alternating methods resorting to block coordinate descent methods have also been used to fit CMTF models with linear couplings [25], but without any constraints and using a Frobenius norm-based loss function. The CMTF Toolbox [18] which uses all-at-once gradient-based optimization with quasi-Newton methods can only handle box-constraints, and sparsity [28]. Some other constraints and regularizations are possible but require the algorithm to be redesigned. The same holds in principle for Gauss-Newton (GN) type algorithms used in Tensorlab [3], [35], where nevertheless a variety of constraints are available via a transformation of variables, and also linear couplings can be incorporated as in [10], [11]. Within the all-at-once optimization framework, in order to have a flexible modelling framework that can incorporate various constraints, a general purpose optimization solver has been used to solve constrained CMTF problems [27]; however, that has not considered linear couplings and different loss functions. Gradient-based all-at-once methods can be extended to other differentiable loss functions as proposed in [30] for a single CP decomposition. However, such extensions to coupled factorizations are not yet available. Another framework for handling general loss functions, derived from the exponential family of distributions, in a coupled setting has been proposed [36]. There, factor matrices are updated alternatingly using a Gauss-Newton method, which results in efficient multiplicative and additive update rules in special cases for non-negative and real data, respectively. This algorithm, however, is limited in terms of constraints and efficient only for special instances, e.g., for nonnegativity constraints [7]. More recently, Huang et al. [37] proposed a flexible and efficient framework for constrained matrix and tensor factorizations that seamlessly incorporates a wide variety of constraints, regularizations and loss functions. The framework uses Alternating Optimization (AO), where each subproblem is solved inexactly using the Alternating Direction Method of Multipliers (ADMM) [38]. However, coupled factorizations are not explicitly considered.

In order to address these limitations, in this paper we introduce a general algorithmic framework to solve a large class of constrained linearly coupled matrix-tensor factorization problems, building onto the AO-ADMM framework [37]. Using numerical experiments on simulated and real data, we demonstrate that the proposed framework is:

- **flexible:** A large variety of constraints, regularizations, coupling structures and loss functions can be handled in a plug-and-play fashion.
- **accurate:** In the case of Frobenius loss, the accuracy achieved by the proposed framework is comparable to state-of-the-art methods. For other loss functions, our proposed framework achieves more accurate results than the extension of the approach introduced in [37] to coupled factorizations.
- **efficient:** We provide efficient closed form updates for a number of commonly used linear coupling relations in the case of Frobenius loss. We show in extensive numerical experiments on simulated data that our approach achieves competitive performance compared to existing methods for problems with exact coupling, and comparable or even superior performance on problems with linear couplings.

In our preliminary results, we have previously demonstrated the promise of AO-ADMM approach for Frobenius norm loss and exact linear couplings [39]. Here, we provide a more detailed derivation and discussion of the algorithm, and the extension to other loss functions. We also include extensive experiments on synthetic data with different levels of difficulty and using different loss functions, as well as a demonstration on real datasets from two different domains.

After establishing some background and notations, in Section II we state the general coupled factorization problems we aim to solve. We briefly explain the basics of ADMM and alternating optimization in Section III, before we derive the AO-ADMM framework for regularized linearly coupled matrix and tensor factorizations in Section IV. Details of the algorithm are given in Section V. Numerical experiments on simulated and real data are presented in Section VI. Finally, limitations of the proposed framework and possible extensions are discussed in Section VII. The supplementary material contains a discussion on the coupling model, experimental details and an extension to flexible couplings.

a) Background and Notation: Here, we define some important tensor notations and concepts, for a full review we refer to [40]. We denote tensors by boldface uppercase calligraphic letters $\mathcal{T}$, matrices by boldface uppercase letters $\mathbf{M}$, vectors by boldface lowercase letters $\mathbf{v}$ and scalars by lowercase letters a. A tensor is a multidimensional array, each

dimension is called a *mode*, and the number of modes is called the *order*. We denote the rank-R_i CP decomposition of a tensor $\boldsymbol{\mathcal{T}}_i$ of order D_i and size $n_{1_i} \times n_{2_i} \times ... \times n_{D_i}$ as $[\![\mathbf{C}_{i,d}]\!]_{d=1}^{D_i}$, as defined in (1), where $\mathbf{C}_{i,d} \in \mathbb{R}^{n_{d_i} \times R_i}$ are called the factor matrices. We use the same notation for the matrix case $D_i = 2$, where $[\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}]\!] = \mathbf{C}_{i,1}\mathbf{C}_{i,2}^T$ corresponds to a generic matrix decomposition. Furthermore, $*$ denotes the (element-wise) Hadamard product between two equally sized matrices, $\otimes$ denotes the Kronecker product of two matrices $\mathbf{A} \in \mathbb{R}^{M \times N}$, $\mathbf{B} \in \mathbb{R}^{K \times J}$,

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} \mathbf{A}(1,1)\mathbf{B} & \cdots & \mathbf{A}(1,N)\mathbf{B} \\ \vdots & \ddots & \vdots \\ \mathbf{A}(M,1)\mathbf{B} & \cdots & \mathbf{A}(M,N)\mathbf{B} \end{bmatrix} \in \mathbb{R}^{MK \times NJ}$$

and $\odot$ denotes the Khatri-Rao product

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{A}(:,1) \otimes \mathbf{B}(:,1) & \cdots & \mathbf{A}(:,N) \otimes \mathbf{B}(:,N) \end{bmatrix}$$

between two matrices with the same number of columns $\mathbf{A} \in \mathbb{R}^{M \times N}$, $\mathbf{B} \in \mathbb{R}^{K \times N}$. Operator $\text{vec}(\mathbf{C})$ refers to a column-wise vectorization of matrix $\mathbf{C}$, and $\boldsymbol{\mathcal{T}}_{[d]}$ denotes the mode-d unfolding of tensor $\boldsymbol{\mathcal{T}}$ into a matrix as defined in [40]. The mode-d unfolding of the CPD model tensor $\boldsymbol{\mathcal{X}}_i = [\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!]$ has the form $\boldsymbol{\mathcal{X}}_{i[d]} = \mathbf{C}_{i,d}\mathbf{M}_{i,d}^T$ [40], where

$$\mathbf{M}_{i,d} := \mathbf{C}_{i,D_i} \odot \ldots \odot \mathbf{C}_{i,d+1} \odot \mathbf{C}_{i,d-1} \odot \ldots \odot \mathbf{C}_{i,1}. \quad (3)$$

Finally, by $\mathbf{I}_R$ we denote the $R \times R$ unit matrix.

II. Regularized CMTF with Linear Couplings

We start by introducing the formalism needed for the general regularized linearly coupled factorization problems we aim to solve. We consider N tensors $\{\boldsymbol{\mathcal{T}}_i\}_{i=1,...,N}$, of not necessarily equal order $D_i \geq 2$ and size $n_{1_i} \times n_{2_i} \times ... \times n_{D_i}$. Thus, matrices are also included. We suppose that each tensor $\boldsymbol{\mathcal{T}}_i$ follows approximately a CPD with rank R_i,

$$\boldsymbol{\mathcal{T}}_i \approx [\![\mathbf{C}_{i,d}]\!]_{d=1}^{D_i},$$

where $\mathbf{C}_{i,d} \in \mathbb{R}^{n_{d_i} \times R_i}$ denotes the factor matrix of mode d in tensor $\boldsymbol{\mathcal{T}}_i$.

Moreover, we suppose that some of the factors $\mathbf{C}_{i,d}$ are regularized using proper lower semi-continuous convex functions $g_{i,d}(\mathbf{C}_{i,d})$. This covers the important case of constrained factors: suppose factor $\mathbf{C}_{i,d}$ should belong to a convex set $\mathcal{C}_{i,d}$, then we may set $g_{i,d} = \iota_{\mathcal{C}_{i,d}}$, where $\iota_{\mathcal{C}_{i,d}}$ is the characteristic function which is null on $\mathcal{C}_{i,d}$ and infinity elsewhere. Also, $g_{i,d}$ can, for instance, be a sparsity inducing norm such as the ℓ_1 norm. As we discuss later on, we only require that the proximity operator of $g_{i,d}$ is computable.

Finally, we suppose that some factors are shared across tensors. We consider the case of exact linearly coupled factors, where two or more tensors $\boldsymbol{\mathcal{T}}_i$ are coupled in any mode d via some underlying matrix $\boldsymbol{\Delta}_d \in \mathbb{R}^{m_{1,d} \times m_{2,d}}$, as follows:

$$\mathbf{H}_{i,d} \,\text{vec}(\mathbf{C}_{i,d}) = \mathbf{H}_{i,d}^{\Delta} \,\text{vec}(\boldsymbol{\Delta}_d), \qquad i = 1, ..., N, \quad (4)$$

for given transformation matrices $\mathbf{H}_{i,d} \in \mathbb{R}^{h_{i,d} \times R_i n_{d_i}}$ and $\mathbf{H}_{i,d}^{\Delta} \in \mathbb{R}^{h_{i,d} \times m_{1,d} m_{2,d}}$. This type of coupling includes many useful cases such as the coupling of modes that have different sampling rates or the partial coupling of tensors that only share a subset of components. Illustrations of such coupling instances are given in Section IV-A. The proposed coupling formulation, which introduces consensus variables $\boldsymbol{\Delta}_d$, is a generalization of pairwise couplings (i.e. couplings such as $\mathbf{H}_{i,d} \,\text{vec}(\mathbf{C}_{i,d}) = \mathbf{H}_{j,d} \,\text{vec}(\mathbf{C}_{j,d})$) and allows for paralellization, see the Supplementary Material for more details. While we focus on exact, *i.e., hard*, couplings in this paper, we propose an extension of our algorithmic framework for approximate, *flexible/soft*, couplings [11], [13], [24], in the Supplementary Material. For ease of notation, for the remainder of the paper we assume that couplings between different tensors are only in the same mode and there does not exist more than one coupling in each mode.

Using general (convex) loss functions $\mathcal{L}_i(\cdot, \cdot)$, we aim at solving optimization problems of the form

$$\begin{aligned} \underset{\{\mathbf{C}_{i,d}, \boldsymbol{\Delta}_d\}_{d \leq D_i, i \leq N}}{\text{argmin}} \quad & \sum_{i=1}^{N} \left[w_i \mathcal{L}_i \left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{C}_{i,d}]\!]_{d=1}^{D_i} \right) + \sum_{d=1}^{D_i} g_{i,d} \left(\mathbf{C}_{i,d} \right) \right], \\ \text{s.t.} \quad & \mathbf{H}_{i,d} \,\text{vec}(\mathbf{C}_{i,d}) = \mathbf{H}_{i,d}^{\Delta} \,\text{vec}(\boldsymbol{\Delta}_d) \end{aligned} \quad (5)$$

where w_i are weighting parameters.

Some reductions. While problem (5) may seem intricate, the following particular settings help reducing its complexity.

- **Loss functions**: Usually, loss functions are chosen to be the squared Frobenius norm. However, in some situations other loss functions may be preferable, see Section IV-B.
- **Regularization-free**: Note that setting $g_{i,d}$ to zero gives the regularization-free, *i.e.,* unconstrained, case.
- **Coupling-free**: Similarly, setting $\mathbf{H}_{i,d}, \mathbf{H}_{j,d}$ and $\mathbf{H}_{i,d}^{\Delta}, \mathbf{H}_{j,d}^{\Delta}$ to contain only zeros, means there is no coupling between tensors i and j in mode d. Note that for any two coupled matrices $\mathbf{C}_{i,d}$ and $\mathbf{C}_{j,d}$ the constraints, if any, should be compatible with each other and with the transformations.
- **Coupled Matrix and Tensor Factorization (CMTF)**: By setting $g_{i,d} = 0$ for all (i,d) and $\mathbf{H}_{i,1} = \mathbf{H}_{i,1}^{\Delta} = \mathbf{I}_{n_1 \cdot R}$ for $i = 1, 2$, the coupled factorization in (2), where shared factors in mode 1 are exactly identical, is obtained [18].

Problem (5) is non-convex, and typically difficult to solve for all i and d simultaneously, in particular, when various constraints or regularizations are imposed on the factors. However, for many choices of $\mathcal{L}$ and g, the cost function is convex w.r.t. $\{\mathbf{C}_{i,d}\}_{i=1}^{N}$ with fixed d. Therefore, we propose an Alternating Optimization (AO) algorithm, where each subproblem is solved inexactly using ADMM.

III. AO-ADMM preliminaries

A. ADMM

Alternating Direction Method of Multipliers (ADMM) is a primal-dual algorithm that aims at solving convex constrained optimization problems of the form

$$\begin{aligned} \underset{\mathbf{x}, \mathbf{z}}{\text{argmin}} \quad & f(\mathbf{x}) + g(\mathbf{z}), \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c} \end{aligned} \quad (6)$$

Algorithm 1 Skeleton of scaled-form ADMM [38]

while convergence criterion is not met **do**

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\operatorname{argmin}}\ f(\mathbf{x}) + \frac{\rho}{2} \| \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}^{(k)} - \mathbf{c} + \boldsymbol{\mu}^{(k)} \|_2^2$$

$$\mathbf{z}^{(k+1)} = \underset{\mathbf{z}}{\operatorname{argmin}}\ g(\mathbf{z}) + \frac{\rho}{2} \| \mathbf{A}\mathbf{x}^{(k+1)} + \mathbf{B}\mathbf{z} - \mathbf{c} + \boldsymbol{\mu}^{(k)} \|_2^2$$

$$\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\mu}^{(k)} + \mathbf{A}\mathbf{x}^{(k+1)} + \mathbf{B}\mathbf{z}^{(k+1)} - \mathbf{c}$$

$k = k + 1$

end while

where f and g are (extended-real-valued) convex functions and f is smooth [38]. ADMM makes use of Dual Ascent on the augmented Lagrangian to find a solution to (6). The augmented Lagrangian is a cost function L obtained by transforming the linear constraints $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}$ into a penalty term involving a dual variable $\boldsymbol{\mu}$, here in scaled form [38]:

$$L(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu}) = f(\mathbf{x}) + g(\mathbf{z}) + \frac{\rho}{2} \| \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c} + \boldsymbol{\mu} \|_2^2$$

ADMM is based on the duality theory for convex optimization: the augmented Lagrangian should be minimized for constant $\boldsymbol{\mu}$, but the dual function $G(\boldsymbol{\mu}) = \min_{\mathbf{x},\mathbf{z}} L(\mathbf{x}, \mathbf{z}, \boldsymbol{\mu})$ should be maximized w.r.t. $\boldsymbol{\mu}$. Therefore, ADMM simply alternates between the minimization of L along variables $\mathbf{x}, \mathbf{z}$ and a gradient ascent step to maximize $G(\boldsymbol{\mu})$ as described in Algorithm 1.

Following [38], based on primal and dual feasibility conditions, a reasonable termination criterion is that primal ($\mathbf{r}^{(k)}$) and dual residuals ($\mathbf{s}^{(k)}$) must be small, *i.e.,*

$$\begin{aligned} \| \mathbf{r}^{(k)} \|_2 &= \| \mathbf{A}\mathbf{x}^{(k)} + \mathbf{B}\mathbf{z}^{(k)} - \mathbf{c} \|_2 \leq \epsilon^{\text{pri}}, \\ \| \mathbf{s}^{(k)} \|_2 &= \| \rho \mathbf{A}^T \mathbf{B} \left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)} \right) \|_2 \leq \epsilon^{\text{dual}}, \end{aligned} \tag{7}$$

where $\epsilon^{\text{pri}} > 0$ and $\epsilon^{\text{dual}} > 0$ are feasibility tolerances, which can be set as follows:

$$\epsilon^{\text{pri}} = \sqrt{\text{length}(\mathbf{c})}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \max \left\{ \| \mathbf{A}\mathbf{x}^{(k)} \|_2, \| \mathbf{B}\mathbf{z}^{(k)} \|_2, \| \mathbf{c} \|_2 \right\},$$
$$\epsilon^{\text{dual}} = \sqrt{\text{length}(\mathbf{x})}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \rho \| \mathbf{A}^T \boldsymbol{\mu}^{(k)} \|_2,$$

with $\epsilon^{\text{abs}} > 0$ and $\epsilon^{\text{rel}} > 0$ denoting the absolute and relative tolerance, respectively.

ADMM has become popular in recent years because of mainly two features. First, given mild hypotheses on the convex functions f and g [38], it is guaranteed to converge to the optimal solution of (6). Second, in many particular instances, ADMM can be implemented using parallel computing, which leverages modern computer architectures. Moreover, ADMM is built upon the theoretical development of proximal operators. For any $\lambda > 0$, a proximal operator of a function g is the following function [41], [42]

$$\text{prox}_{\lambda g}(\mathbf{x}) = \underset{\mathbf{u}}{\operatorname{argmin}}\ g(\mathbf{u}) + \frac{1}{2\lambda} \| \mathbf{x} - \mathbf{u} \|_2^2, \tag{8}$$

which is single-valued for convex g and well-defined for proper lower semi-continuous g. For many functions g such as characteristic functions of convex sets, a closed-form expression is available, see Table II and lists here [42], [43]. One may notice that ADMM updates for $\mathbf{x}$ and $\mathbf{z}$ are proximal operators if respectively $\mathbf{A}$ or $\mathbf{B}$ are orthogonal matrices, although since f is differentiable the $\mathbf{x}^{(k+1)}$ update can be carried out by smooth convex descent algorithms.

B. AO-ADMM: ADMM within block-coordinate descent.

Now, suppose the following optimization problem is given:

$$\underset{\mathbf{x},\mathbf{y}}{\operatorname{argmin}}\ f(\mathbf{x}, \mathbf{y}) + g_x(\mathbf{x}) + g_y(\mathbf{y}) \tag{9}$$

where f is non-convex, but $f_y + g_y : \mathbf{y} \mapsto f(\mathbf{x}, \mathbf{y}) + g(\mathbf{y})$ is convex as well as $f_x + g_x$. A simple yet powerful idea is to partially solve (9) w.r.t. $\mathbf{x}$ for fixed $\mathbf{y}$, then for $\mathbf{y}$ with fixed $\mathbf{x}$, and to iterate this process until convergence. Namely, the cost function is (approximatively) minimized over blocks $\mathbf{x}$ and $\mathbf{y}$ alternatively, an optimization method commonly known as block-coordinate descent. To solve each convex constrained subproblem, it is possible to use ADMM described above in a straightforward manner. The resulting optimization algorithm, that solves alternatively partial problems using ADMM, has been named as AO-ADMM [37]. The convergence of AO-ADMM, to the best of our knowledge, is not proven, see also [37] for an overview of convergence results. It is only known, that if the minimum of each subproblem is uniquely attained and the cost is non-increasing on the update path, every limit point of the algorithm is guaranteed to be a stationary point [44] (3rd ed., proposition 3.7.1 pp324). To ensure there is a unique solution to the subproblems, a majorized version can be solved which is strongly convex, the so called block succesive upper-bound minimization (BSUM) [45]. This can, for example, be done by adding a proximal regularization term, with $\alpha_x^{(k)} > 0$, such that the update for $\mathbf{x}$ becomes [37]

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\operatorname{argmin}}\ f(\mathbf{x}, \mathbf{y}^{(k)}) + g_x(x) + \frac{\alpha_x^{(k)}}{2} \left\| \mathbf{x} - \mathbf{x}^{(k)} \right\|_2^2.$$

C. Related work: ADMM for constrained tensor factorizations

Besides the already mentioned AO-ADMM framework [37], many examples of ADMM-based algorithms for constrained matrix or tensor factorizations can be found in the literature. For instance, the AO-ADMM framework [37] has been extended to deal with robust tensor factorization, where some slabs are grossly corrupted [46]. A parallelization strategy and high performance implementation of the AO-ADMM framework [37] has been presented [47]. Furthermore, ADMM has been used for non-negative matrix factorizations with the beta-divergence [48], [49], and for fitting CP models with various constraints [50], [51]. Recently, Afshar et al. have also used ADMM to fit a constrained PARAFAC2 model [52]. Direct ADMM has also been applied to coupled matrix factorizations with ℓ_1 loss in the Robust JIVE algorithm [53].

IV. AO-ADMM for regularized CMTF with linear couplings

Using the algorithms described in the previous section, we can now derive a flexible algorithmic framework to solve the general optimization problem for regularized linearly coupled matrix and tensor factorizations (5), which is repeated here for convenience:

$$\begin{aligned} \underset{\{\mathbf{C}_{i,d}, \boldsymbol{\Delta}_d\}_{d \leq D_i, i \leq N}}{\operatorname{argmin}} \quad & \sum_{i=1}^{N} \left[w_i \mathcal{L}_i \left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{C}_{i,d}]\!]_{d=1}^{D_i} \right) + \sum_{d=1}^{D_i} g_{i,d} \left(\mathbf{C}_{i,d} \right) \right] \\ \text{s.t.} \quad & \mathbf{H}_{i,d} \operatorname{vec}(\mathbf{C}_{i,d}) = \mathbf{H}_{i,d}^{\Delta} \operatorname{vec}(\boldsymbol{\Delta}_d) \end{aligned}$$

This problem is non-convex with respect to all arguments. However, the problem w.r.t. the block of parameters $\{\{\mathbf{C}_{i,d}\}_{i=1}^{N}, \boldsymbol{\Delta}_d\}$ with fixed mode d, *e.g.,* for mode 1,

$$\begin{aligned} \underset{\{\mathbf{C}_{i,1}\}_{i\leq N}, \boldsymbol{\Delta}_1}{\text{argmin}} \quad & \sum_{i=1}^{N} \left[w_i \mathcal{L}_i \left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{C}_{i,d}]\!]_{d=1}^{D_i} \right) + g_{i,1} \left(\mathbf{C}_{i,1} \right) \right] \\ \text{s.t.} \quad & \mathbf{H}_{i,1} \operatorname{vec}(\mathbf{C}_{i,1}) = \mathbf{H}_{i,1}^{\Delta} \operatorname{vec}(\boldsymbol{\Delta}_1) \end{aligned}$$

is convex for convex functions $\mathcal{L}_i$ and $g_{i,1}$. For each factor matrix $\mathbf{C}_{i,1}$ we define a "split" matrix variable $\mathbf{Z}_{i,1}$, similar to $\mathbf{z}$ in (6), which separates the regularization from the factorization, but introduces the additional equality constraint $\mathbf{C}_{i,1} = \mathbf{Z}_{i,1}$. Variable $\boldsymbol{\Delta}_1$ can also be seen as a split variable since it decouples the coupled factor matrices. This leads to a convex optimization problem similar to (6),

$$\begin{aligned} \underset{\{\mathbf{C}_{i,1}, \mathbf{Z}_{i,1}\}_{i\leq N}, \boldsymbol{\Delta}_1}{\text{argmin}} \quad & \sum_{i=1}^{N} \left[w_i \mathcal{L}_i \left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{C}_{i,d}]\!]_{d=1}^{D_i} \right) + g_{i,1} \left(\mathbf{Z}_{i,1} \right) \right] \\ \text{s.t.} \quad & \mathbf{H}_{i,1} \operatorname{vec}(\mathbf{C}_{i,1}) = \mathbf{H}_{i,1}^{\Delta} \operatorname{vec}(\boldsymbol{\Delta}_1) \\ & \mathbf{C}_{i,1} = \mathbf{Z}_{i,1} \end{aligned} \tag{10}$$

which can be solved with ADMM. We introduce two sets of dual variables, $\{\boldsymbol{\mu}_{i,1_{(z)}}\}_{i\leq N}$ (matrix-valued) for the regularization constraint and $\{\boldsymbol{\mu}_{i,1_{(\delta)}}\}_{i\leq N}$ (vector-valued) for the coupling constraint. The augmented Lagrangian can then be written explicitly as follows, with $\boldsymbol{\delta}$ denoting $\operatorname{vec}(\boldsymbol{\Delta})$:

$$\begin{aligned} L(\mathbf{C}_{i,1}, \mathbf{Z}_{i,1}, \boldsymbol{\delta}_1, \boldsymbol{\mu}_{i,1_{(z)}}, \boldsymbol{\mu}_{i,1_{(\delta)}}) = \sum_{i=1}^{N} \Bigg[& w_i \mathcal{L}_i \left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{C}_{i,d}]\!]_{d=1}^{D_i} \right) \\ & + g_{i,1}(\mathbf{Z}_{i,1}) + \frac{\rho}{2} \left\| \mathbf{C}_{i,1} - \mathbf{Z}_{i,1} + \boldsymbol{\mu}_{i,1_{(z)}} \right\|_F^2 \\ & + \frac{\rho}{2} \left\| \mathbf{H}_{i,1} \operatorname{vec}(\mathbf{C}_{i,1}) - \mathbf{H}_{i,1}^{\Delta} \boldsymbol{\delta}_1 + \boldsymbol{\mu}_{i,1_{(\boldsymbol{\delta})}} \right\|_2^2 \Bigg] \end{aligned}$$

Analogously to Algorithm 1, alternating minimization of this augmented Lagrangian with respect to $\{\mathbf{C}_{i,1}\}_{i\leq N}$, $\{\mathbf{Z}_{i,1}\}_{i\leq N}$ and $\boldsymbol{\Delta}_1$ and gradient ascent for the dual variables $\{\boldsymbol{\mu}_{i,1_{(z)}}\}_{i\leq N}$ and $\{\boldsymbol{\mu}_{i,1_{(\delta)}}\}_{i\leq N}$ results in the ADMM Algorithm 2.

Note that both *for* loops can be computed using parallel programming. For unconstrained factor matrices $\mathbf{C}_{i,1}$, all terms involving $\mathbf{Z}_{i,1}$ become unnecessary and are omitted. Similarly, uncoupled, but constrained, factor matrices $\mathbf{C}_{i,1}$ are updated in an independent ADMM loop, where all terms involving $\boldsymbol{\delta}_1$ are omitted. Finally, for uncoupled and unconstrained factor matrices $\mathbf{C}_{i,1}$, ADMM iterations are not necessary, since the exact solution can be obtained by a least-squares update. Note that Algorithm 2 is presented for mode 1 only, for ease of notation, but can easily be adapted to any other mode. Using alternating optimization (AO) as explained in Section III-B, Algorithm 2 is then repeatedly looped over all modes d to estimate all the variables $\mathbf{C}_{i,d}$. This alternating optimization constitutes the frame of our algorithm and is illustrated in Algorithm 3. Note that factors of different modes are updated sequentially, but one mode for all tensors at the same time as shown in Algorithm 2. This results in independent ADMM runs for uncoupled factors and one AO-ADMM run for all coupled factors of that mode.

Algorithm 2 ADMM for subproblem w.r.t. mode 1 of regularized linearly coupled CPD

1: **while** convergence criterion is not met **do**
2: **for** $i = 1, ..., N$ **do**
3: $\mathbf{C}_{i,1}^{(k+1)} = \underset{\mathbf{X}}{\text{argmin}}\ w_i \mathcal{L}_i \left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{X}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!] \right) + \frac{\rho}{2} \left\| \mathbf{X} - \mathbf{Z}_{i,1}^{(k)} + \boldsymbol{\mu}_{i,1_{(z)}}^{(k)} \right\|_F^2 + \frac{\rho}{2} \left\| \mathbf{H}_{i,1} \operatorname{vec}(\mathbf{X}) - \mathbf{H}_{i,1}^{\Delta} \boldsymbol{\delta}_1^{(k)} + \boldsymbol{\mu}_{i,1_{(\delta)}}^{(k)} \right\|_2^2$
4: **end for**
5: $\boldsymbol{\delta}_1^{(k+1)} = \underset{\mathbf{z}}{\text{argmin}} \sum_{i=1}^{N} \left\| \mathbf{H}_{i,1} \operatorname{vec}(\mathbf{C}_{i,1}^{(k+1)}) - \mathbf{H}_{i,1}^{\Delta} \mathbf{z} + \boldsymbol{\mu}_{i,1_{(\delta)}}^{(k)} \right\|_2^2$
6: **for** $i = 1, ..., N$ **do**
7: $\mathbf{Z}_{i,1}^{(k+1)} = \underset{\mathbf{Z}}{\text{argmin}}\ g_{i,1}(\mathbf{Z}) + \frac{\rho}{2} \left\| \mathbf{C}_{i,1}^{(k+1)} - \mathbf{Z} + \boldsymbol{\mu}_{i,1_{(z)}}^{(k)} \right\|_F^2 = \operatorname{prox}_{\frac{1}{\rho} g_{i,1}} \left(\mathbf{C}_{i,1}^{(k+1)} + \boldsymbol{\mu}_{i,1_{(z)}}^{(k)} \right)$
8: $\boldsymbol{\mu}_{i,1_{(z)}}^{(k+1)} = \boldsymbol{\mu}_{i,1_{(z)}}^{(k)} + \mathbf{C}_{i,1}^{(k+1)} - \mathbf{Z}_{i,1}^{(k+1)}$
9: $\boldsymbol{\mu}_{i,1_{(\delta)}}^{(k+1)} = \boldsymbol{\mu}_{i,1_{(\delta)}}^{(k)} + \mathbf{H}_{i,1} \operatorname{vec}(\mathbf{C}_{i,1}^{(k+1)}) - \mathbf{H}_{i,1}^{\Delta} \boldsymbol{\delta}_1^{(k+1)}$
10: **end for**
11: $k = k + 1$
12: **end while**

Algorithm 3 AO-ADMM algorithm for regularized coupled CPD

initialize $\{\mathbf{C}_{i,d}\}_{d\leq D_i, i\leq N}$, $\{\boldsymbol{\Delta}_d\}_{d\leq \max(D_i)}$, $\{\mathbf{Z}_{i,d}\}_{d\leq D_i, i\leq N}$, $\{\boldsymbol{\mu}_{i,d_{(\boldsymbol{\delta})}}\}_{d\leq D_i, i\leq N}$, $\{\boldsymbol{\mu}_{i,d_{(z)}}\}_{d\leq D_i, i\leq N}$

while convergence criterion is not met **do**
  for $d = 1, ..., \max_i(D_i)$ **do**
    update $\{\mathbf{C}_{i,d}\}_{i\leq N}$, $\boldsymbol{\Delta}_d$, $\{\mathbf{Z}_{i,d}\}_{i\leq N}$, $\{\boldsymbol{\mu}_{i,d_{(\boldsymbol{\delta})}}\}_{i\leq N}$, $\{\boldsymbol{\mu}_{i,d_{(z)}}\}_{i\leq N}$, using few iterations of Algorithm 2
  end for
end while
return factor matrices $\{\mathbf{C}_{i,d}\}_{d\leq D_i, i\leq N}$

The main advantage of ADMM is that it splits the problem (10) into easier individual problems for $\mathbf{C}_{i,1}$, $\mathbf{Z}_{i,1}$ and $\boldsymbol{\delta}_1$, respectively. For the commonly used squared Frobenius norm loss, the update of factor matrices $\mathbf{C}_{i,1}$ (line 3 of Algorithm 2) results in the solution of a linear least-squares problem, which can be solved efficiently, provided $\mathbf{H}_{i,1}$ and $\mathbf{H}_{i,1}^{\Delta}$ have a specific structure. In this case, also the $\boldsymbol{\delta}_1$ update (line 5 of Algorithm 2) can be computed efficiently. This is discussed in detail in Section IV-A. In case of other loss functions than the squared Frobenius norm, we resolve to numerical optimization to update $\mathbf{C}_{i,1}$, as discussed in Section IV-B.Furthermore, handling the regularization reduces to the computation of the proximal operator $\operatorname{prox}_{\frac{1}{\rho} g_{i,1}} \left(\mathbf{C}_{i,1}^{(k+1)} + \boldsymbol{\mu}_{i(z)}^{(k)} \right)$ to update $\mathbf{Z}_{i,1}^{(k+1)}$ (line 7 of Algorithm 2). For many commonly used regularization functions g, the corresponding proximal operator can easily be computed, see Table II. The implementation of the above algorithm can be found in [54].

A. Different types of linear coupling

In the case of squared Frobenius norm loss, the updates for

factor matrices $\mathbf{C}_{i,1}$,

$$\begin{aligned}\mathbf{C}_{i,1}^{(k+1)} = \underset{\mathbf{X}}{\operatorname{argmin}}\ & w_i \left\| \boldsymbol{\mathcal{T}}_{i[1]} - \mathbf{X}\mathbf{M}_{i,1}^{(k)^T} \right\|_F^2 \\ & + \frac{\rho}{2} \left\| \mathbf{X} - \mathbf{Z}_{i,1}^{(k)} + \boldsymbol{\mu}_{i,1}{}_{(z)}^{(k)} \right\|_F^2 \\ & + \frac{\rho}{2} \left\| \mathbf{H}_{i,1} \operatorname{vec}(\mathbf{X}) - \mathbf{H}_{i,1}^{\Delta} \boldsymbol{\delta}_1^{(k)} + \boldsymbol{\mu}_{i,1}{}_{(\delta)}^{(k)} \right\|_2^2,\end{aligned}$$

where $\mathbf{M}_{i,1}$ is defined as in (3), are given by a large linear least-squares problem, which can hardly be computed efficiently. However, for the following five special forms of linear couplings $\mathbf{H}_{i,1} \operatorname{vec}(\mathbf{C}_{i,1}) = \mathbf{H}_{i,1}^{\Delta} \operatorname{vec}(\boldsymbol{\Delta}_1)$, efficient updates are available.

1) Case 1*: Hard coupling (no transformation):* One way of coupling tensors i and j is simply requiring equality of the factor matrices $\mathbf{C}_{i,1} = \mathbf{C}_{j,1}$. This case has been used in formulations of coupled matrix-tensor factorizations and coupled tensor factorizations [18], [32], [36], [55], [56] as well as their applications in data mining [7], [9], [57]. Obviously, coupled factor matrices have to be of the same size and obey the same constraints. The coupling constraint can then simply be written in matrix form as $\mathbf{C}_{i,1} = \boldsymbol{\Delta}_1$. The update of $\mathbf{C}_{i,1}$ is obtained by solving the following linear system, where $\boldsymbol{\mu}_{i(\Delta)}$ denotes the matricized version of $\boldsymbol{\mu}_{i(\delta)}$,

$$\begin{aligned}&\mathbf{C}_{i,1}^{(k+1)} \left[w_i \mathbf{M}_{i,1}^{(k)^T} \mathbf{M}_{i,1}^{(k)} + \frac{\rho}{2} (\mathbf{I}_R + \mathbf{I}_R) \right] = \\ &\quad \left[w_i \boldsymbol{\mathcal{T}}_{i[1]} \mathbf{M}_{i,1}^{(k)} + \frac{\rho}{2} \left(\mathbf{Z}_{i,1}^{(k)} - \boldsymbol{\mu}_{i,1}{}_{(z)}^{(k)} + \boldsymbol{\Delta}_1^{(k)} - \boldsymbol{\mu}_{i,1}{}_{(\Delta)}^{(k)} \right) \right].\end{aligned}$$

The update of $\boldsymbol{\Delta}_1$ is given by the average

$$\boldsymbol{\Delta}_1^{(k+1)} = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{C}_{i,1}^{(k+1)} + \boldsymbol{\mu}_{i,1}{}_{(\Delta)}^{(k)} \right). \tag{11}$$

2) Case 2*: Transformations in mode dimension:* Often, measurements obtained from different instruments will correspond to different temporal or spatial sampling grids or aggregation intervals. For instance, assume tensors i and j have different dimensions n_{1_i}, n_{1_j} in mode 1, due to different sampling rates, but sample the same underlying function. Although direct coupling would not make any sense, it may still be possible to approximate the common underlying function via interpolations. Those interpolations can then be compared on a common sampling grid of size n_{Δ_1} [33], where the consensus variable $\boldsymbol{\Delta}_1 \in \mathbb{R}^{n_{\Delta_1} \times R}$ represents the function on the common grid. There are two possibilities for such couplings. Illustrations of these are shown in Fig. 2. We discuss rank and size restrictions on the particular transformation matrices in the Supplementary Material.

a) Case 2*a:* The factor matrices can be coupled via known transformation matrices $\tilde{\mathbf{H}}_{i,1} \in \mathbb{R}^{n_{\Delta_1} \times n_{1_i}}$, $n_{\Delta_1} \leq \min_i n_{1_i}$, such that

$$\tilde{\mathbf{H}}_{i,1} \mathbf{C}_{i,1} = \boldsymbol{\Delta}_1.$$

This results from setting $\mathbf{H}_{i,1}^{\Delta} = \mathbf{I}_{n_{\Delta_1} \cdot R}$ and $\mathbf{H}_{i,1} = \mathbf{I}_R \otimes \tilde{\mathbf{H}}_{i,1}$ in (4). The update for $\mathbf{C}_{i,1}$ is then given by the solution of the following Sylvester equation

$$\begin{aligned}&\frac{\rho}{2} \left(\mathbf{I}_R + \tilde{\mathbf{H}}_{i,1}^T \tilde{\mathbf{H}}_{i,1} \right) \mathbf{C}_{i,1}^{(k+1)} + \mathbf{C}_{i,1}^{(k+1)} w_i \mathbf{M}_{i,1}^{(k)^T} \mathbf{M}_{i,1}^{(k)} = \\ &w_i \boldsymbol{\mathcal{T}}_{i[1]} \mathbf{M}_{i,1}^{(k)} + \frac{\rho}{2} \left[\mathbf{Z}_{i,1}^{(k)} - \boldsymbol{\mu}_{i,1}{}_{(z)}^{(k)} + \tilde{\mathbf{H}}_{i,1}^T \left(\boldsymbol{\Delta}_1^{(k)} - \boldsymbol{\mu}_{i,1}{}_{(\Delta)}^{(k)} \right) \right].\end{aligned}$$

Examples of this type of coupling can be found in [10], where a model of neurovascular coupling and downsampling is exploited to deal with the different temporal resolutions of EEG and fMRI. In [11] also the spatial mode of EEG and fMRI is matched using the lead field matrix. In [13], the transformation matrices correspond to first or second order (numerical) derivative matrices. The update of $\boldsymbol{\Delta}_1$ is again given by an average,

$$\boldsymbol{\Delta}_1^{(k+1)} = \frac{1}{N} \sum_{i=1}^{N} \left(\tilde{\mathbf{H}}_{i,1} \mathbf{C}_{i,1}^{(k+1)} + \boldsymbol{\mu}_{i,1}{}_{(\Delta)}^{(k)} \right).$$

Note that in [10] the matrix $\tilde{\mathbf{H}}_{i,1}$ is actually a parameterized model of a hemodynamic response function and the parameters are fitted in the optimization process. We leave learning $\tilde{\mathbf{H}}_{i,1}$ as future work.

b) Case 2*b:* Transformations in Case 2a do not allow the common sampling grid to have more points n_{Δ_1} than the smallest dimension n_{1_i} of all coupled factor matrices. This restriction can be avoided by transformations of the form

$$\mathbf{C}_{i,1} = \tilde{\mathbf{H}}_{i,1}^{\Delta} \boldsymbol{\Delta}_1,$$

where $\tilde{\mathbf{H}}_{i,1}^{\Delta}$ is of size $n_{1_i} \times n_{\Delta_1}$. This is obtained by setting $\mathbf{H}_{i,1} = \mathbf{I}_{n_{1_i} \cdot R}$ and $\mathbf{H}_{i,1}^{\Delta} = \mathbf{I}_R \otimes \tilde{\mathbf{H}}_{i,1}^{\Delta}$. The updates of the factor matrices result in solving the following linear systems

$$\begin{aligned}&\mathbf{C}_{i,1}^{(k+1)} \left[w_i \mathbf{M}_{i,1}^{(k)^T} \mathbf{M}_{i,1}^{(k)} + \frac{\rho}{2} (\mathbf{I}_R + \mathbf{I}_R) \right] = \\ &\quad \left[w_i \boldsymbol{\mathcal{T}}_{i[1]} \mathbf{M}_{i,1}^{(k)} + \frac{\rho}{2} \left(\mathbf{Z}_{i,1}^{(k)} - \boldsymbol{\mu}_{i,1}{}_{(z)}^{(k)} + \tilde{\mathbf{H}}_{i,1}^{\Delta} \boldsymbol{\Delta}_1^{(k)} - \boldsymbol{\mu}_{i,1}{}_{(\Delta)}^{(k)} \right) \right].\end{aligned}$$

and the update of $\boldsymbol{\Delta}_1$ is given by

$$\boldsymbol{\Delta}_1^{(k+1)} = \left[\sum_{i=1}^{N} \left(\tilde{\mathbf{H}}_{i,1}^{\Delta^T} \tilde{\mathbf{H}}_{i,1}^{\Delta} \right) \right]^{-1} \left[\sum_{i=1}^{N} \tilde{\mathbf{H}}_{i,1}^{\Delta^T} \left(\mathbf{C}_{i,1}^{(k+1)} + \boldsymbol{\mu}_{i,1}{}_{(\Delta)}^{(k)} \right) \right].$$

An example of such a formulation can be found in the context of tensor disaggregation [25], where the transformation corresponds to an aggregation matrix. It is also used in a tensor fusion model for hyperspectral super-resolution, where the first transformation corresponds to blurring and downsampling to model the spatial degragation in the hyperspectral image and the second transformation corresponds to band-selection and averaging for spectral degragation in the multispectral image [26].

3) Case 3*: Transformations in component dimension:* Transformations in the component dimension can be used in cases where coupled tensors do not have the same rank and/or share only a part of their components. We again differentiate between two subtypes.

a) Case 3*a:* Setting $\mathbf{H}_{i,1}^{\Delta} = \mathbf{I}_{n_1 \cdot R_{\Delta}}$ and $\mathbf{H}_{i,1} = \hat{\mathbf{H}}_{i,1}^T \otimes \mathbf{I}_{n_1}$ results in linear couplings of the form

$$\mathbf{C}_{i,1} \hat{\mathbf{H}}_{i,1} = \boldsymbol{\Delta}_1, \tag{12}$$

with known transformation matrices $\hat{\mathbf{H}}_{i,1} \in \mathbb{R}^{R_i \times R_{\Delta}}$. This is equivalent to matching linear combinations of the factor vectors of different components via the consensus variable $\boldsymbol{\Delta}_1 \in \mathbb{R}^{n_1 \times R_{\Delta}}$. It allows tensors with the same size $n_{1_i} = n_{1_j} = n_1$ in mode 1, but different number of components $R_i \neq R_j$, to be coupled. In particular, the special case of rectangular "identity" matrices $\hat{\mathbf{H}}_{i,1} \in \mathbb{R}^{R_i \times R_{\Delta}}$, where the

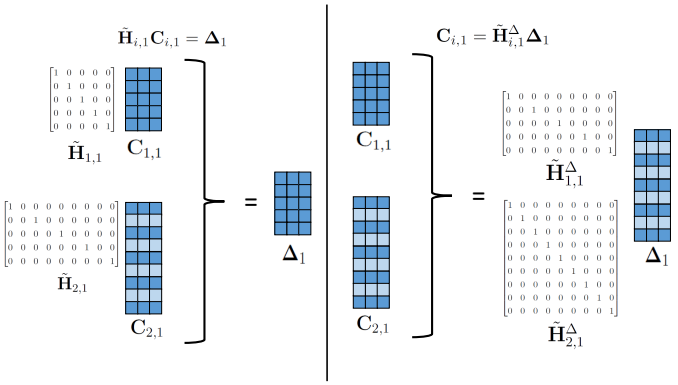


Fig. 2. Illustration of possible couplings in Case 2 where tensors 1 and 2 have different dimensions in mode 1 and the factor matrix $\mathbf{C}_{1,1}$ contains only every second row of matrix $\mathbf{C}_{2,1}$. Left: Case 2a. Right: Case 2b.

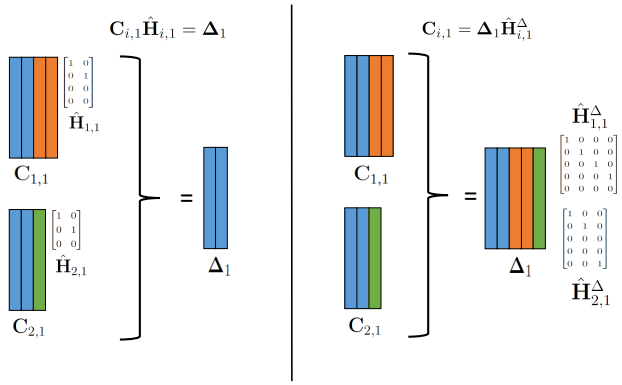


Fig. 3. Illustration of possible coupling where two components are shared between tensor 1 and 2, tensor 1 has two additional components, tensor 2 has one additional component. Left: Case 3a. Right: Case 3b.

upper block consists of the $R_\Delta \times R_\Delta$ unit matrix and the lower block contains only zeros has a meaningful application. In that case, the consensus variable $\boldsymbol{\Delta}_1$ contains the R_Δ shared vectors of the possible larger factor matrices $\{\mathbf{C}_{i,1}\}_{i\leq N}$. The number of those shared components has to be known beforehand. An illustration of this can be seen in Fig. 3 on the left side. The update of $\mathbf{C}_{i,1}$ is then given by

$$\mathbf{C}_{i,1}^{(k+1)}\left[w_i{\mathbf{M}_{i,1}^{(k)}}^T\mathbf{M}_{i,1}^{(k)} + \frac{\rho}{2}\left(\mathbf{I}_{R_i} + \hat{\mathbf{H}}_{i,1}\hat{\mathbf{H}}_{i,1}^T\right)\right] = \left[w_i\boldsymbol{\mathcal{T}}_{i[1]}\mathbf{M}_{i,1}^{(k)} + \frac{\rho}{2}\left(\mathbf{Z}_{i,1}^{(k)} - {\boldsymbol{\mu}_{i,1}}_{(z)}^{(k)} + \left(\boldsymbol{\Delta}_1^{(k)} - {\boldsymbol{\mu}_{i,1}}_{(\Delta)}^{(k)}\right)\hat{\mathbf{H}}_{i,1}^T\right)\right].$$

The update of the consensus variable $\boldsymbol{\Delta}_1$ is again given by an average,

$$\boldsymbol{\Delta}_1^{(k+1)} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{C}_{i,1}^{(k+1)}\hat{\mathbf{H}}_{i,1} + {\boldsymbol{\mu}_{i,1}}_{(\Delta)}^{(k)}\right).$$

Such coupled tensor factorizations with known number of shared and unshared factors have been previously studied [22] and also used to jointly analyze EEG and fMRI signals [11].

b) Case 3b: On the other hand, in couplings of the form

$$\mathbf{C}_{i,1} = \boldsymbol{\Delta}_1\hat{\mathbf{H}}^{\Delta}_{i,1},$$

where $\mathbf{H}_{i,1} = \mathbf{I}_{n_1\cdot R_i}$ and $\mathbf{H}^{\Delta}_{i,1} = {\hat{\mathbf{H}}^{\Delta}_{i,1}}^T \otimes \mathbf{I}_{n_1}$, the consensus variable $\boldsymbol{\Delta}_1 \in \mathbb{R}^{n_1\times R_\Delta}$ can be thought of as a dictionary from which factor matrices are built via known linear combinations of its elements. Especially, when some components are only shared between some but not all coupled tensors, $\boldsymbol{\Delta}_1$ may contain all columns of all coupled factor matrices $\{\mathbf{C}_{i,1}\}$, no matter if they belong to shared or unshared components. This information can be encoded in "identity" matrices $\hat{\mathbf{H}}^{\Delta}_{i,1} \in \mathbb{R}^{R_\Delta\times R_i}$, which contain exactly one 1 per column and at most one 1 per row. An illustration of this can be seen in Fig. 3 on the right side. The update of the factor matrices is given by

$$\mathbf{C}_{i,1}^{(k+1)}\left[w_i{\mathbf{M}_{i,1}^{(k)}}^T\mathbf{M}_{i,1}^{(k)} + \frac{\rho}{2}\left(\mathbf{I}_{R_i} + \mathbf{I}_{R_i}\right)\right] = \left[w_i\boldsymbol{\mathcal{T}}_{i[1]}\mathbf{M}_{i,1}^{(k)} + \frac{\rho}{2}\left(\mathbf{Z}_{i,1}^{(k)} - {\boldsymbol{\mu}_{i,1}}_{(z)}^{(k)} + \boldsymbol{\Delta}_1^{(k)}\hat{\mathbf{H}}^{\Delta}_{i,1} - {\boldsymbol{\mu}_{i,1}}_{(\Delta)}^{(k)}\right)\right].$$

The update of $\boldsymbol{\Delta}_1$ results in a linear system of the form

$$\boldsymbol{\Delta}_1^{(k+1)} = \left[\sum_{i=1}^{N}\left(\mathbf{C}_{i,1}^{(k+1)} + {\boldsymbol{\mu}_{i,1}}_{(\Delta)}^{(k)}\right){\hat{\mathbf{H}}^{\Delta}_{i,1}}^T\right]\left[\sum_{i=1}^{N}\left(\hat{\mathbf{H}}^{\Delta}_{i,1}{\hat{\mathbf{H}}^{\Delta}_{i,1}}^T\right)\right]^{-1}.$$

An example of this type of coupling has been used in audio source separation [23], where a binary transformation matrix is used to make sure that only a predefined number of columns of a spectral template are used in one of the decompositions.

Additionally, for the combination of coupling Cases 2a with 3b and 2b with 3a, efficient updates with small linear systems or Sylvester equations, respectively, can be derived in a similar way.

B. Different loss functions

Various data mining applications may require cost functions other than the Frobenius norm, which relies on the underlying assumption that the data follows a Gaussian distribution. Social network analysis often relies on count data, which does not follow the Gaussian assumption; therefore, other loss functions such as the Kullback-Leibler (KL) divergence have shown to be more effective for this kind of data [7]. Similarly, if $\boldsymbol{\mathcal{T}}$ contains music pieces, an Itakuro-Saito (IS) divergence-based cost function has shown to reveal more interpretable results [48]. From a statistical perspective, the cost function is a by-product of the distribution of the entries of $\boldsymbol{\mathcal{T}}_i$ around their mean parameterized by the CPD model, recently studied in detail in [30]. Therefore, it is possible to imagine a situation where various $\boldsymbol{\mathcal{T}}_i$ have various data fitting functions. For a general loss function $\mathcal{L}_i(\cdot,\cdot)$, the minimization problem in line 3 of Algorithm 2 may not have a closed form solution. Therefore, we resolve to numerical optimization. We use the limited-memory BFGS with bound constraints (L-BFGS-B) [58] in the implementation [59]. Using L-BFGS-B has the advantage that bound constraints, like non-negativity which is necessary for several loss functions, can be ensured in every iteration of the algorithm. The disadvantage of using a gradient based optimization method is that our framework will be limited to loss functions $\mathcal{L}_i(\cdot,\cdot)$ which are differentiable in the second argument. Details about the gradient computation can be found in the Supplementary Material. Nevertheless, many loss functions are differentiable and can be handled by this approach, some important examples are given in Table I together with references to some of their applications in matrix

and tensor factorizations. There, ℓ denotes the element-wise loss function, such that

$$\mathcal{L}(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{X}}) = \sum_{j \in \mathcal{J}} \ell(t_j, x_j), \tag{13}$$

where j is a multiindex $j = (j_1, j_2, ..., j_D)$ and $\mathcal{J}$ the set of all possible indices. Note that some loss functions introduce non-negativity constraint on the tensor entries in $\boldsymbol{\mathcal{X}} = [\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!]$, which we enforce via non-negativity constraints on all factor matrices. As described before, these bound constraints can be handled by L-BFGS-B, without the need to introduce additional variables $\mathbf{Z}_{i,1}$. β- and α-divergences are generalized families of divergences and include the squared Frobenius norm, KL and IS divergence as special cases. The Huber loss can be thought as a smooth approximation to the ℓ_1-loss [30], which is often used for robust fitting when the data is sparsely corrupted by outliers [60].

Another approach to solving the problem in line 3 of Algorithm 2 (without the coupling) in an AO-ADMM setting is proposed in [37]. There, another auxiliary tensor variable $\tilde{\boldsymbol{\mathcal{T}}}_i$ is introduced which is fitted to the data tensor $\boldsymbol{\mathcal{T}}_i$ via the loss function $\mathcal{L}_i(\cdot, \cdot)$, while $\tilde{\boldsymbol{\mathcal{T}}}_i = [\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!]$ is treated as another constraint using ADMM. This results in closed form updates for a number of possible loss functions and is thus more efficient than numerical optimization. However, our experiments show that it typically takes a large number of outer iterations before this approach converges. We also observe that it is less accurate than our approach in the case of KL divergence, see Section VI-B5. We suspect that the "double approximation" first in Frobenius norm and then in the loss function can have negative effects on accuracy. In the following, we refer to this approach as *split AO-ADMM*.

C. Some useful proximal operators

When a factor matrix $\mathbf{C}_{i,1}$ is regularized, the update of the corresponding auxiliary variable $\mathbf{Z}_{i,1}$ in Algorithm 2 is given by the proximal operator (8)

$$\mathbf{Z}_{i,1}^{(k+1)} = \operatorname{prox}_{\frac{1}{\rho} g_{i,1}} \left(\mathbf{C}_{i,1}^{(k+1)} + \boldsymbol{\mu}_{i,1}{}_{(z)}^{(k)} \right)$$

where the function $g_{i,1}$ can be any proper, lower semi-continuous and convex function [41], [42], [64]. Thus, besides hard constraints given by a characteristic function $\iota_{\mathcal{C}}$ of a convex set $\mathcal{C}$, soft constraints are also included in this framework via convex regularization functions. In many cases, closed form solutions and/or efficient algorithms exist, see [42], [64] and the references therein, and we resort to the implementations collected in *The Proximity Operator Repository* [43]. In Table II we recall only a few constraints and regularizations, which are of practical interest, while many others are also possible [42], [64]. Note that the regularization can be defined as an element-wise function $g(x)$ or act on columns $\mathbf{x}$ of factor matrices. As a result of the scaling ambiguity in the CPD model, normalization of the factors may be needed in coupled factorizations. The usual way to tackle norm constraints on the factors is to consider a ℓ_2 constraint, *e.g.,* $\| \mathbf{C}_{i,1}(:, j) \|_2 = 1$, where j is any column index [13], [65]. Here, we can also tackle the normalization of the factors in the same way as other constraints by applying the proximal operator of the ℓ_2 unit ball on the columns of the split variables $\mathbf{Z}_{i,1}$. In some cases, it is possible to combine different constraints with one proximal operator, see [42], [66]. In practice it is possible to handle also non-convex regularization functions g, as long as *a* solution of the corresponding proximal operator can be computed, since its solution may not be unique.

V. Algorithm Details

In this section, we present some algorithmic aspects in more detail and discuss complexity and efficient implementations.

A. Stopping conditions

We adapt the stopping criterion (7) of the inner ADMM iterations to Algorithm 2, by splitting it into coupling and constraint related conditions, as follows:

$$\sum_i \left(\|\mathbf{C}_{i,1}^{(k)} - \mathbf{Z}_{i,1}^{(k)}\|_F / \|\mathbf{C}_{i,1}^{(k)}\|_F \right) \leq \epsilon^{\text{p,constr}}$$

$$\sum_i \left(\|\mathbf{H}_{i,1} \operatorname{vec}(\mathbf{C}_{i,1}^{(k)}) - \mathbf{H}_{i,1}^{\Delta} \boldsymbol{\delta}_1^{(k)}\|_2 / \|\mathbf{H}_{i,1} \operatorname{vec}(\mathbf{C}_{i,1}^{(k)})\|_2 \right) \leq \epsilon^{\text{p,coupl}}$$

$$\sum_i \left(\|\mathbf{Z}_{i,1}^{(k+1)} - \mathbf{Z}_{i,1}^{(k)}\|_F / \|\boldsymbol{\mu}_{i,1}{}_{(z)}^{(k)}\|_F \right) \leq \epsilon^{\text{d,constr}}$$

$$\sum_i \left(\|\mathbf{H}_{i,1}^{\Delta} \left(\boldsymbol{\delta}_1^{(k+1)} - \boldsymbol{\delta}_1^{(k)} \right) \|_2 / \|\boldsymbol{\mu}_{i,1}{}_{(\delta)}^{(k)}\|_2 \right) \leq \epsilon^{\text{d,coupl}}$$

All of the above conditions should be satisfied to stop the algorithm, except when a predefined maximum number of inner iterations is reached. We usually set this to a reasonably small number between 5 and 10, since we do not want to solve the subproblems exactly in the beginning of the algorithm, when the initializations are probably far away from the optimal solution. As the algorithm proceeds, the inner ADMM algorithm typically terminates due to the relative tolerances and the number of iterations goes down. For a study of the maximum number of inner iterations, see also the experiment in Section VI-B2. The whole algorithm is terminated, when each of the following residuals $f_\star^{(k)}$,

$$\begin{aligned} f_{\text{tensors}}^{(k)} &= \sum_i w_i \mathcal{L}_i \left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!] \right) \\ f_{\text{couplings}}^{(k)} &= \sum_{i,d} \left(\|\mathbf{H}_i \operatorname{vec}(\mathbf{C}_{i,d}^{(k)}) - \mathbf{H}_i^{\Delta} \boldsymbol{\delta}_d^{(k)}\|_2 / \|\mathbf{H}_i \operatorname{vec}(\mathbf{C}_{i,d}^{(k)})\|_2 \right) \\ f_{\text{constraints}}^{(k)} &= \sum_{i,d} \left(\|\mathbf{C}_{i,d}^{(k)} - \mathbf{Z}_{i,d}^{(k)}\|_F / \|\mathbf{C}_{i,d}^{(k)}\|_F \right). \end{aligned} \tag{14}$$

has either reached a small absolute tolerance $\epsilon^{\text{abs,outer}}$, or has not changed more than some small relative tolerance $\epsilon^{\text{rel,outer}}$ compared to the previous iteration,

$$f_\star^{(k)} < \epsilon^{\text{abs,outer}}, \qquad | f_\star^{(k)} - f_\star^{(k-1)} | \,/\, | f_\star^{(k)} | < \epsilon^{\text{rel,outer}}$$

or a predefined number of maximal outer iterations is reached.

B. Choice of ρ

To the best of our knowledge, an optimal step-size ρ has been derived for ADMM applied to quadratic programming

TABLE I
OVERVIEW OF DIFFERENT LOSS FUNCTIONS

loss	underlying distribution	elementwise loss fucntion	references
Kullback-Leibler (KL) divergence	Poisson	$\ell(t,x) = x - t\log x + t\log t - t,\ t \in \mathbb{N}, x \geq 0$	[5], [61]
Itakura-Saito (IS) divergence	Gamma	$\ell(t,x) = \frac{t}{x} + \log x - \log t - 1,\ t, x > 0$	[48]
β-divergence		$\ell(t,x) = \frac{t^\beta}{\beta(\beta-1)} + \frac{x^\beta}{\beta} - \frac{t x^{\beta-1}}{\beta-1},\ \beta \in \mathbb{R} \setminus \{0,1\}$	[7], [49], [62]
α-divergence		$\ell(t,x) = \frac{1}{\alpha(\alpha-1)}\left(t^\alpha x^{1-\alpha} - \alpha t + (\alpha-1)x\right),\ \alpha \in \mathbb{R} \setminus \{0,1\}$	[62], [63]
Huber-loss		$\ell(t,x;d) = \begin{cases} (t-x)^2 & \text{if } \lvert t-x \rvert \leq d \\ 2d\lvert t-x \rvert - d^2 & \text{otherwise} \end{cases}$	[53]

TABLE II
SOME PROXIMAL OPERATORS

structure	function $g(x)$	proximal operator $\operatorname{prox}_{\frac{1}{\rho}g}(x)$
non-negativity	$\iota_{\mathbb{R}_+}(x)$	$\max\{0, x\}$
box-constraints	$\iota_{\text{Box}[\ell,u]}(x)$	$\min\{\max\{x,\ell\}, u\}$
simplex constraints	$\iota_{\Delta_N}(\mathbf{x})$	$[\max\{0, x_n - \lambda\}]_{1\leq n\leq N}$, with $\lambda \in \mathbb{R}$ s.t. $\sum_{n=1}^N \max\{0, x_n - \lambda\} = 1$
monotonicity	$\iota_{\{x_1 \leq x_2 \leq \ldots \leq x_n\}}(\mathbf{x})$	$\left[\max_{1\leq j\leq n} \min_{n\leq k\leq N} \frac{1}{k-j+1}\sum_{\ell=j}^k x_\ell\right]_{1\leq n\leq N}$
hard sparsity constraints	$\iota_{\{\mathbf{x}:\lVert\mathbf{x}\rVert_1 \leq r\}}(\mathbf{x})$	$\begin{cases} \mathbf{x}, & \lVert\mathbf{x}\rVert_1 \leq r \\ \mathcal{T}_{\lambda^\star}(\mathbf{x}), & \lVert\mathbf{x}\rVert_1 > r, \end{cases}$ with soft-thresholding operator $\mathcal{T}_\lambda(x) = \max\{0, \lvert x\rvert - \lambda\}\operatorname{sgn}(x)$, and $\lambda^\star$ is any positive solution of $\lVert\mathcal{T}_\lambda(\mathbf{x})\rVert_1 = r$
normalization	$\iota_{\{\mathbf{x}:\lVert\mathbf{x}\rVert_2 \leq 1\}}(\mathbf{x})$	$1/\left(\max\left\{\lVert\mathbf{x}\rVert_2, 1\right\}\right)\mathbf{x}$
Lasso regularization	$\gamma\lVert\mathbf{x}\rVert_1$	soft-thresholding operator $\mathcal{T}_{\frac{\gamma}{\rho}}(\mathbf{x})$
ℓ_2-Regularization	$\gamma\lVert\mathbf{x}\rVert_2$	$\left(1 - \frac{\gamma}{\rho\max\{\lVert\mathbf{x}\rVert_2, \gamma/\rho\}}\right)\mathbf{x}$
Smoothness	$\gamma\lVert\mathbf{D}\mathbf{x}\rVert_2^2$	$\left(\frac{2\gamma}{\rho}\mathbf{D}^T\mathbf{D} + \mathbf{I}\right)^{-1}\mathbf{x}$
Normalized sparsity	$\iota_{\{\mathbf{x}:\lVert\mathbf{x}\rVert_2=1\}\cap\{\mathbf{x}:\lVert\mathbf{x}\rVert_0\leq k\}}$	$H_k(\mathbf{x})/\lVert H_k(x)\rVert_2$ where H_k is the hard-thresholding operator [66]

only [67]. In choosing ρ we follow [37] and choose a different step-length $\rho_{i,d}^{(k)}$ for the update of each factor matrix $\mathbf{C}_{i,d}^{(k)}$ as

$$\rho_{i,d}^{(k)} = \|\mathbf{M}_{i,d}^{(k)}\|_F^2/R_i = \operatorname{trace}({\mathbf{M}_{i,d}^{(k)}}^T\mathbf{M}_{i,d}^{(k)})/R_i, \qquad (15)$$

which can be efficiently computed from (16).

C. Efficient implementations

In all types of linear couplings described in Section IV-A, with the exception of Case 2a, the update of the factor matrix $\mathbf{C}_{i,1}$ reduces to the solution of a linear system where the matrix inverse is only of size $R_i \times R_i$. Note that this matrix inverse is never explicitly computed. Instead, a Cholesky decomposition which costs $O(R_i^3)$ is precomputed outside the ADMM loop. Thus, solving the linear systems at each ADMM iteration reduces to one forward- and one backward-substitution with complexity $O(R_i^2 n_{1_i})$. Also for the update of $\boldsymbol{\Delta}_1$, a Cholesky decomposition of the matrices $\sum_i^N \left(\tilde{\mathbf{H}}_{i,1}^{\Delta T}\tilde{\mathbf{H}}_{i,1}^{\Delta}\right)$ and $\left[\sum_{i=1}^N \left(\hat{\mathbf{H}}_{i,1}^{\Delta}\hat{\mathbf{H}}_{i,1}^{\Delta T}\right)\right]$ in Cases 2b and 3b respectively, can be precomputed at the beginning of the AO-ADMM algorithm. In Case 2b, this can be expensive, with the Cholesky decomposition having a complexity of $O(n_{\Delta_1}^3)$ and each forward and backward-substitution $O(n_{\Delta_1}^2 R_i)$, in contrast to $O(R_\Delta^3)$ and $O(R_\Delta^2 n_1)$ for Case 3b. In Cases 1, 2a and 3a, the update of $\boldsymbol{\Delta}_1$ is computed as an average with complexity $O(Nn_1R)$, $O(Nn_{\Delta_1}n_{1_i}R)$ and $O(Nn_1R_iR_\Delta)$, respectively.

Furthermore, the matricized tensor times Khatri-Rao product $\boldsymbol{\mathcal{T}}_{i[1]}\mathbf{M}_{i,1}$, can be computed efficiently [40], for which we use the `mttkrp` function from the Tensor Toolbox [68]. It can be precomputed outside the ADMM loop. Also the product of Khatri-Rao products $\mathbf{M}_{i,1}^T\mathbf{M}_{i,1}$ can be computed efficiently using the relation [40]

$$\mathbf{M}_{i,1}^T\mathbf{M}_{i,1} = \mathbf{C}_{i,2}^T\mathbf{C}_{i,2} * \ldots * \mathbf{C}_{i,D_i}^T\mathbf{C}_{i,D_i} \qquad (16)$$

with precomputed products $\{\mathbf{C}_{i,d}^T\mathbf{C}_{i,d}\}_{i\leq N, d\leq D_i}$. These products can be stored throughout the whole AO-ADMM algorithm and need only to be updated for each mode after the corresponding outer AO iteration.

The evaluation of the residual $f_{\text{tensors}}^{(k)}$ for the stopping condition can be computationally expensive. It is, therefore, desirable to reuse as many previous computations as possible. For tensors $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{\mathcal{M}}$, the difference in squared Frobenius norm can be calculated as

$$\|\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{M}}\|_F^2 = \|\boldsymbol{\mathcal{T}}\|_F^2 + \|\boldsymbol{\mathcal{M}}\|_F^2 - 2\langle\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{M}}\rangle$$

Here, $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{T}}_i$ is the given tensor, which does not change, and it is enough to compute its norm only once. The same holds for other loss functions, *e.g.*, for KL-divergence, it is sufficient to compute the constant term $\sum_{j\in\mathcal{J}}[t_j\log t_j - t_j]$ only once. Furthermore, in the case of Frobenius norm, the following holds:

$$\begin{aligned}\|\boldsymbol{\mathcal{M}}\|_F^2 &= \|[\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, \ldots, \mathbf{C}_{i,D_i}]\!]\|_F^2 \\ &= \mathbf{e}^T\left[\mathbf{C}_{i,1}^T\mathbf{C}_{i,1} * \mathbf{C}_{i,2}^T\mathbf{C}_{i,2} * \ldots * \mathbf{C}_{i,D_i}^T\mathbf{C}_{i,D_i}\right]\mathbf{e},\end{aligned}$$

where $\mathbf{e}$ is a column vector of 1s with matching length. This is useful since the products $\mathbf{C}_{i,d}^T\mathbf{C}_{i,d}$ can be reused. The tensor inner product $\langle \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{M}} \rangle$ can also be computed efficiently via

$$\langle \boldsymbol{\mathcal{T}}_i, [|\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}|] \rangle = \mathbf{e}^T \left[\boldsymbol{\mathcal{T}}_{i[d]} \mathbf{M}_{i,d} * \mathbf{C}_{i,d} \right] \mathbf{e},$$

where again the matricized tensor times Khatri-Rao product $\boldsymbol{\mathcal{T}}_{i[d]}\mathbf{M}_{i,d}$ from the last updated mode d of tensor i can be reused.

VI. Experiments

In this section, we assess the performance of the proposed AO-ADMM approach in terms of accuracy and computational efficiency on synthetic datasets, and demonstrate its use on real datasets in two applications. In the case of Frobenius loss, a number of methods is available for comparison. However, their ability to handle specific constraints and coupling structures varies. So whenever applicable, we compare the performance with other commonly used methods, namely Alternating Least Squares (ALS) and all-at-once optimization using quasi-Newton and Gauss-Newton methods. The results show that the AO-ADMM approach achieves comparable performance to state-of-the art methods on problems with hard coupling (Case 1) in a number of settings with varying collinearity, tensor sizes and constraints. Furthermore, our experiments indicate that AO-ADMM has advantages over other methods for problems with linear couplings. Moreover, in the case of KL loss, we show that our AO-ADMM approach is more accurate than the split AO-ADMM approach.

A. Experimental Set-up

For ALS (referred to as CMTF-ALS), we use our own implementation for CMTF based on cp_als from the Tensor Toolbox [68]. It can so far only handle hard couplings. For nonnegativity constraints, we solve the alternating nonnegative least squares problems using Hierarchical ALS (HALS) [69]. For all-at-once optimization with quasi-Newton methods, we use cmtf_opt from the CMTF Toolbox [18] (referred to as CMTF opt) using Nonlinear Conjugate Gradient (NCG) for unconstrained cases and Limited Memory BFGS with bounds (LBFGS-B) for nonnegativity constraints. The implementation can only handle box-constraints. Linear couplings are theoretically possible, but are not implemented. Finally, for Gauss-Newton, we use the Tensorlab implementation sdf_nls [35] (referred to as Tensorlab GN), which can handle a variety of constraints and coupling structures. Our implementation of the AO-ADMM algorithm can be found in [54].

We monitor the convergence of different algorithms through the function value f_{tensors} and the factor match score (FMS). Given the true factor matrices $\mathbf{C}_{i,d}^{\text{true}}$, the FMS for $\mathbf{C}_{i,d}$ is computed as

$$\text{FMS} = \prod_{i=1}^{N} \frac{1}{R_i} \sum_{r=1}^{R_i} \left(\prod_{d=1}^{D_i} \frac{\left\langle \mathbf{C}_{i,d}(:,r), \mathbf{C}_{i,d}^{\text{true}}(:,r) \right\rangle}{\| \mathbf{C}_{i,d}(:,r) \|_2 \, \| \mathbf{C}_{i,d}^{\text{true}}(:,r) \|_2} \right),$$

after finding the best permutation of factors. We run each algorithm until the relative change in function value is less than 10^{-10} or a maximum number of 3000 outer iterations has been reached[1]. For AO-ADMM, we set the absolute tolerance for the residuals in Eq. (14) to 10^{-4}. For each dataset, five initializations (for the more difficult experiment 4, ten initializations) are used, and the best run, i.e., the one with the lowest final function value, is reported. We also report the number of failed runs in Tables III and IV, where a run is considered a failed run if it reaches the maximum number of iterations or gives an FMS below the threshold $0.99^{\sum_{i=1}^{N} D_i}$. When no constraints are imposed and Frobenius loss is used, in the first run, factor matrices $\mathbf{C}_{i,d}$ are initialized using the first R left singular vectors of the corresponding concatenated (if coupled) and unfolded tensors in mode d. Otherwise, factor matrices are initialized at random, drawing from the standard normal or, in the case of nonnegativity, uniform distribution, and the columns are normalized. All dual variables, as well as coupling variables $\boldsymbol{\Delta}_d$, are initialized using the standard normal distribution. The split variables $\mathbf{Z}_{i,d}$ are initialized by $\text{prox}_{1,g_{i,d}}(\mathbf{C}_{i,d})$. The experiments were performed on a standard 16GB Windows 10 laptop using MATLAB R2019a, without the use of parallel for-loops in Algorithm 2. Computing times are measured using the `tic-toc` function in MATLAB.

B. Simulated Data

For each experiment, we generate 50 random datasets. Following the CP model, tensors $\boldsymbol{\mathcal{X}}_i$ are constructed from known factor matrices. In the case of Frobenius loss, *i.e.,* experiments $1-4$, tensors $\boldsymbol{\mathcal{T}}_i$ are then generated by adding noise tensors $\boldsymbol{\mathcal{N}}_i$ with entries drawn from a standard normal distribution as follows:

$$\boldsymbol{\mathcal{T}}_i = \boldsymbol{\mathcal{X}}_i + 0.2(\| \boldsymbol{\mathcal{X}}_i \|_F / \| \boldsymbol{\mathcal{N}}_i \|_F)\boldsymbol{\mathcal{N}}_i.$$

This corresponds to a signal-to-noise ratio of around 14dB. We normalize each tensor to have Frobenius norm 1 and set the weights to $w_i = 1/2$. In the case of other loss functions, tensors $\boldsymbol{\mathcal{T}}_i$ are generated by drawing each entry from the underlying distribution, *e.g.,* Poisson distribution for KL-divergence, with the mean given by the corresponding entry in $\boldsymbol{\mathcal{X}}_i$.

1) Experiment 1: Frobenius loss, coupling Case 1 *and unconstrained:*

a) Mildly Collinear Factors: In this example, a third-order tensor of size $200 \times 100 \times 300$ is coupled in the first mode with a matrix of size 200×1000 based on hard coupling, *i.e.,* Case 1. These datasets are jointly analyzed using Frobenius loss and no constraints are imposed. The ground-truth factor matrices with rank $R = 3$ are generated with entries drawn from a standard normal distribution and then transformed as in [70] such that the factors have a congruence of 0.5. A summary of the convergence behaviour of different algorithms for 50 random datasets is shown in Fig. 4, which depicts the median curves and quantiles of function value and factor match score over iterations and time. Here, only the

[1]Other parameters are set as follows: In cmtf_opt (NCG/LBFSG-B): `MaxFuncEvals/maxTotalIts`=10^5, `StopTol/pgtol`=10^{-32}, in Tensorlab: `CGMaxIter`=15, `TolX`=10^{-32}, `TolAbs`=0, in HALS: `maxiter`=500.

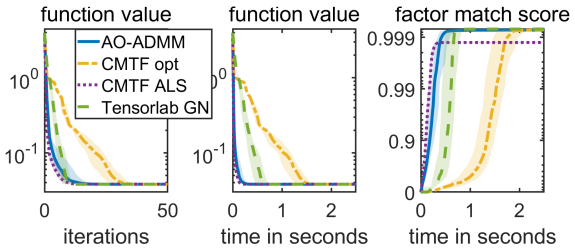


Fig. 4. Experiment1a: Median and quartiles of function values and FMS for different algorithms.

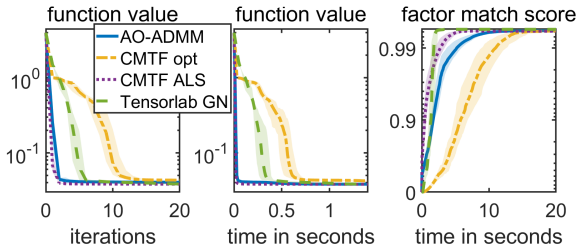


Fig. 5. Experiment1b: Median and quartiles of function values and FMS for different algorithms.

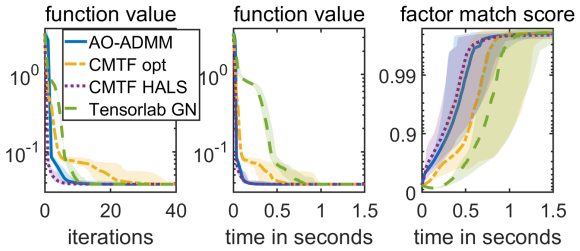


Fig. 6. Experiment2: Median and minimum/maximum function values and FMS for different algorithms.

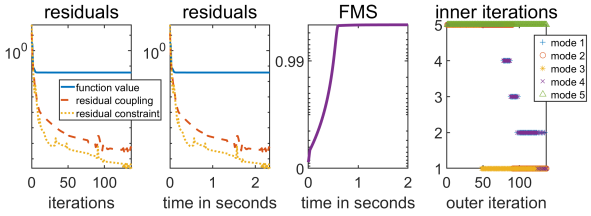


Fig. 7. Experiment 2: Detailed convergence of one AO-ADMM run.

best run (among five initializations), i.e., the one giving the lowest function value, is reported for each dataset. Boxplots showing the complete distribution of computing times can be found in the Supplementary Material for this and the following experiments. We observe that alternating methods CMTF-ALS and AO-ADMM behave similarly, but AO-ADMM is more accurate, achieving an FMS comparable with that achieved by all-at-once methods. In the presence of collinearity, singular vectors seem to be a bad choice for initialization for AO-ADMM. Table III shows a high number of failed runs for AO-ADMM which are all due to initializations based on singular vectors. On the other hand, when we use factor matrices with random collinearity, this type of initialization is not an issue (results not shown here). It should also be mentioned here that the timing comparisons do not include the time needed for computing the singular value decomposition for the first initialization. Since all-at-once methods usually perform best with this initialization (in this experiment) their total computation time is actually higher than shown in the figures. Results for this setting with unbalanced data sizes and more noise can be found in the Supplementary Material.

b) Highly Collinear Factors: Here, collinear factors are generated using a congruence of 0.9. Fig. 5 shows that AO-ADMM again behaves similar to ALS and is also accurate except for the failing cases shown in Table III for the best 50 runs. In this case, Tensorlab GN is faster, which is expected. In the presence of strong collinearity, ALS type algorithms struggle due to ill-conditioned linear systems. GN type methods, on the other hand, can find an accurate solution fast thanks to the use of approximate second-order information [71].

2) Experiment 2: Frobenius loss, coupling Case 1 *and nonnegative:* The setting of the second experiment is similar to the first one. In addition, nonnegativity constraints are imposed on factor matrices in all modes. The ground-truth factor matrices are drawn from a uniform distribution and thus the collinearity of the components is not controlled. The convergence behaviour of different methods in Fig. 6 demonstrates that AO-ADMM is computationally efficient, with an average performance similar to CMTF-HALS. All methods can accurately recover the true factors used to generate the data. The convergence of the different residuals, see equation (14), in AO-ADMM can be seen in Fig. 7 for a single run of AO-ADMM. It can be observed that even though the CP fitting term does not seem to improve after a number of iterations due to the noise, the residuals for couplings and constraints keep improving, as well as the factor match score. In the right most subfigure, the number of performed inner iterations is plotted against the outer iterations for each mode. All modes start with the maximum of 5 inner iterations. The iterations go down fast and stay at 1 for modes 2 and 3, which are the uncoupled tensor modes. The coupled modes 1 and 4 always have the same number of inner iterations since they are updated together in one ADMM run. Also, inner iterations of those coupled modes decrease stepwise after some iterations. The only mode that stays at 5 inner iterations is mode 5, which is in the presence of noise probably the most difficult one, since it is the uncoupled matrix mode. Results for unbalanced datasets, datasets with different number of components as well as a study of different numbers of inner iterations in the AO-ADMM algorithm can be found in the Supplementary Material.

3) Experiment 3: Frobenius loss, coupling Case 2*a and unconstrained:* In this experiment, we test the linear coupling with transformation on a simple example. Linear couplings of type 2a allow tensors with different dimensions to be coupled. Here, a tensor $\boldsymbol{\mathcal{T}}_1$ of size $400 \times 100 \times 300$ is coupled in the first mode with a matrix $\mathbf{T}_2$ of size 200×1000 via a transformation $\tilde{\mathbf{H}}_{1,1}\mathbf{C}_{1,1} = \Delta_1$, that discards every other row in $\mathbf{C}_{1,1}$. The rank of both is again $R = 3$ and the true factor matrices have entries drawn from the standard normal

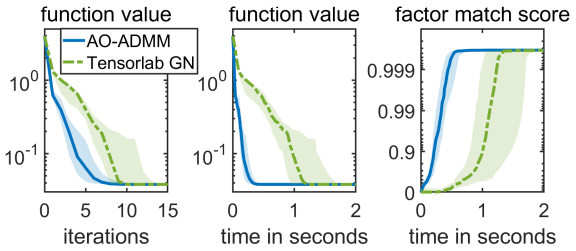


Fig. 8. Experiment3: Median and quartiles of function values and FMS for AO-ADMM and Tensorlab GN.

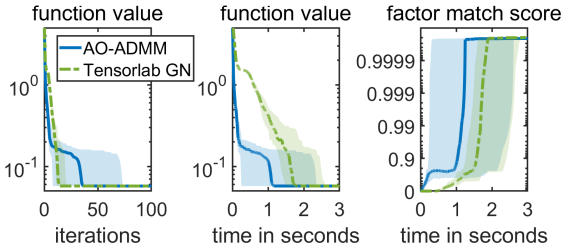


Fig. 9. Experiment4: Median and quartiles of function values and FMS for AO-ADMM and Tensorlab GN.

TABLE III
NUMBER OF FAILED RUNS IN EXPERIMENTS 1-4

	AO-ADMM		CMTF opt		CMTF ALS		TL GN	
Exp.	all init.	best runs	all init.	best runs	all init.	best runs	all init.	best runs
1a	38	0	0	0	7	0	2	0
1b	82	17	0	0	0	0	3	0
2	0	0	0	0	0	0	0	0
3	7	0	-	-	-	-	44	0
4	296	1	-	-	-	-	323	0

distribution. We compare the AO-ADMM-based approach only with Tensorlab GN since among the available methods that can handle linear couplings, it is the most flexible in terms of incorporating constraints. Fig. 8 shows that AO-ADMM finds the true factors faster (on average) than Tensorlab GN. Both are accurate while Tensorlab is slightly more sensitive to initializations, see Table III.

4) Experiment 4: Frobenius loss, coupling Case 3b and unconstrained: Linear couplings of type 3b allow tensors with different number of components to be coupled. We demonstrate this type of coupling with an example, where three tensors of size $200 \times 100 \times 300$, $200 \times 300 \times 100$, $200 \times 100 \times 100$ and number of components $R = 2, 3, 4$, respectively, are coupled in the first mode. Two components are shared by all tensors while the additional component in the second tensor is also present in the third tensor. The third tensor has an additional unshared component. This *ordering* of shared and unshared components makes the problem more difficult and many random initializations will not converge to this order, as shown in Table III. Therefore, we use 10 random initializations for each dataset. However, when the best runs out of multiple initializations are considered, both AO-ADMM and Tensorlab GN are accurate while AO-ADMM finds the true factors faster, see Fig. 9.

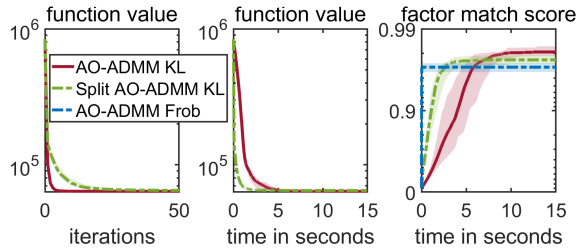


Fig. 10. Experiment5: Median and quartiles of function values and FMS for different algorithms.

TABLE IV
NUMBER OF FAILED RUNS IN EXPERIMENT 5

AO-ADMM KL		split AO-ADMM KL		AO-ADMM Frobenius	
all init.	best runs	all init.	best runs	all init.	best runs
0	0	19	3	0	0

5) Experiment 5: KL-divergence and coupling Case 1: In this experiment, instead of adding Gaussian noise, we construct tensors containing count data $0, 1, 2, \ldots$ by drawing each entry from a Poisson distribution with mean given by the CPD model of the ground-truth factor matrices. The factor matrices are drawn from a gamma distribution with shape and scale parameters equal to 1. This ensures that factor matrices and, thus, the tensors constructed from them are non-negative. A tensor of size $40 \times 50 \times 60$ is coupled in the first mode with a matrix of size 40×100 using coupling Case 1. The rank of both is $R = 3$. We compare the performance of our AO-ADMM implementation using KL divergence as described in Section IV-B with the variant proposed in [37], which is using another split tensor variable to deal with the KL divergence. For fairer comparison and because the code is not presently accessible online, we use our own implementation. Results based on 50 random datasets can be seen in Fig. 10. For each dataset, 5 random initializations are used. Initial factor matrices are drawn from a standard uniform distribution. While in our implementation non-negativity constraints are handled by using LBFGS-B [2] with lower bound 0, in the split AO-ADMM variant, non-negativity constraints are explicitly imposed and handled via ADMM. The parameters used for the split AO-ADMM are the same as for AO-ADMM. The split tensor variable is initialized equal to the data tensor, while its dual tensor is initialized with all zeros. The split AO-ADMM approach is computationally more efficient and thus converges faster than our approach, although it needs considerably more outer iterations. Fig. 10 shows that both algorithms achieve FMS that are above the FMS achieved by using the squared Frobenius norm loss. However, our approach is even more accurate than the split AO-ADMM and achieves higher factor match scores after convergence. It has no failed runs, while the split AO-ADMM failed for 3 out of 50 random datasets, see Table IV.

[2]with parameters $\texttt{m} = 5$, $\texttt{maxIts} = 100$, $\texttt{maxTotalIts} = 5000$, $\texttt{factr} = 10^{-10}/\texttt{eps}$, $\texttt{pgtol} = 10^{-10}$

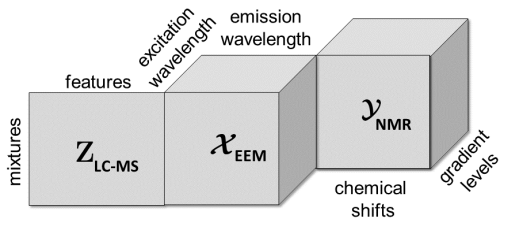


Fig. 11. Datasets from different platforms, coupled in the *mixtures* mode.

C. Real Data

Finally, we showcase the proposed framework on two different applications from chemometrics and remote sensing.

1) Chemometrics: We first demonstrate an application of the proposed framework by jointly analyzing measurements of mixtures from multiple analytical platforms, and show that underlying patterns of individual chemicals in the mixtures can be accurately revealed. Datasets consist of measurements of 28 mixtures using fluorescence spectroscopy (excitation-emission matrices (EEM)), LC-MS and diffusion NMR spectroscopy (DOSY-NMR). Mixtures were prepared using five chemicals, *i.e.,* Valine-Tyrosine-Valine (Val-Tyr-Val), Tryptophan-Glycine (Trp-Gly), Phenylalanine (Phe), Maltoheptaose (Malto) and Propanol, based on a predetermined design. The fluorescence data is represented as a third-order tensor, $\boldsymbol{\mathcal{X}}_{EEM}$, with modes: *mixtures*, *excitation* and *emission wavelengths*; DOSY-NMR as a third-order tensor, $\boldsymbol{\mathcal{Y}}_{NMR}$, with modes: *mixtures*, *chemical shifts*, and *gradient levels*, and LC-MS data as a *mixtures* by *features* matrix, $\mathbf{Z}_{LCMS}$. Datasets are coupled in the *mixtures* mode (Fig. 11). Three chemicals are visible in all platforms while another chemical can be detected using both NMR and LC-MS. The fifth chemical, *i.e.,* Proponal, is only visible by NMR, indicating that $\boldsymbol{\mathcal{X}}_{EEM}$, $\boldsymbol{\mathcal{Y}}_{NMR}$, and $\mathbf{Z}_{LCMS}$ have three, five and four components, respectively, if each component models one chemical. In total, we use a 6-component model since, in addition to the five chemicals, there is an additional component modelling the structured noise in LC-MS. See [65], [72] for a detailed discussion.

We jointly analyze these datasets by coupling the factor matrices in the *mixtures* mode using the linear coupling given in Case 3b, as follows:

$$\begin{aligned}\min_{\substack{\{\mathbf{C}_{i,d}\},\boldsymbol{\Delta}_1\\ i\le 3\\ d\le D_i}} & \left\| \boldsymbol{\mathcal{X}}_{EEM} - [\![\mathbf{C}_{1,d}]\!]_{d=1}^{3} \right\|_F^2 + \left\| \boldsymbol{\mathcal{Y}}_{NMR} - [\![\mathbf{C}_{2,d}]\!]_{d=1}^{3} \right\|_F^2 \\ & + \left\| \mathbf{Z}_{LCMS} - [\![\mathbf{C}_{3,d}]\!]_{d=1}^{2} \right\|_F^2 \\ \text{s.t.} \quad & \mathbf{C}_{i,d} \ge 0, i \le 3, d \le D_i \\ & \mathbf{C}_{1,1} = \boldsymbol{\Delta}_1 \hat{\mathbf{H}}_{1,1}^{\Delta}, \mathbf{C}_{2,1} = \boldsymbol{\Delta}_1 \hat{\mathbf{H}}_{2,1}^{\Delta}, \mathbf{C}_{3,1} = \boldsymbol{\Delta}_1 \hat{\mathbf{H}}_{3,1}^{\Delta}, \end{aligned} \tag{17}$$

where $\hat{\mathbf{H}}_{1,1}^{\Delta} = [e_1\ e_2\ e_3]$, $\hat{\mathbf{H}}_{2,1}^{\Delta} = [e_1\ e_2\ e_3\ e_4\ e_5]$ and $\hat{\mathbf{H}}_{3,1}^{\Delta} = [e_1\ e_2\ e_3\ e_4\ e_6]$, with e_i denoting the i-th unit vector in $\mathbb{R}^6$. Nonnegativity constraints are imposed in all modes since all factor matrices are expected to be nonnegative (*e.g.,* $\mathbf{C}_{1,1}$ models the relative concentrations of chemicals in the mixtures while $\mathbf{C}_{2,2}$ corresponds to emission spectra). By using Algorithm 3 to solve the optimization problem in (17), we can successfully extract components modelling the individual chemicals in the mixtures. Fig. 12 shows the factor matrices $\mathbf{C}_{1,1}, \mathbf{C}_{2,1}, \mathbf{C}_{3,1}$ coupled through the given transformation matrices in the *mixtures* mode, demonstrating that individual chemicals can be separated and the underlying true design of the experiment can be recovered. While there are six components in total, each dataset captures components modelling some of the chemicals, as indicated in the transformation matrices. Similarly, factor matrices in other modes have also been extracted accurately and can be used to identify the chemical being modelled by each component (not shown). Here, we have used multiple initializations and the one reaching the minimum function value has been reported after checking the uniqueness of the model.

2) Hyperspectral super-resolution: In this second experiment, we address the problem of reconstructing a full hyperspectral image $\mathbf{X} \in \mathbb{R}_+^{n_1 \times n_2}$ (which is a collection of n_1 vectorized color images with n_2 pixels) from two smaller images. These HyperSpectral and MultiSpectral Images (HSI-MSI) $\mathbf{X}_h \in \mathbb{R}_+^{n_1 \times n_h}$ and $\mathbf{X}_m \in \mathbb{R}_+^{n_m \times n_2}$ have respectively fewer pixels and fewer spectral bands. In this example, we will assume that the transformations from the large image to the two smaller ones, which are respectively a blurring/downsampling (by a factor 9) matrix $\mathbf{H}$ and a spectral degradation function $\mathbf{S}$ (we used the one from the Landsat 7 multispectral sensor), are known.

There is a rather large body of literature on how to recover the super-resolution image $\mathbf{X}$. Here, our purpose is to show that we can use the proposed algorithmic framework to infer good parameters for a popular model based on joint constrained nonnegative matrix factorization [73]. We aim at solving

$$\begin{aligned}\underset{\mathbf{E}\in\mathbb{R}_+^{n_1\times R},\mathbf{A}\in\mathbb{R}_+^{n_2\times R}}{\operatorname{argmin}} & \|\mathbf{X}_h - \mathbf{E}(\mathbf{H}\mathbf{A})^T\|_F^2 + \|\mathbf{X}_m - (\mathbf{S}\mathbf{E})\mathbf{A}^T\|_F^2 \\ & + \eta_{\mathcal{S}_{n_2}}(\mathbf{A}) + \eta_{\mathcal{S}_{n_h}}(\mathbf{H}\mathbf{A}) \end{aligned} \tag{18}$$

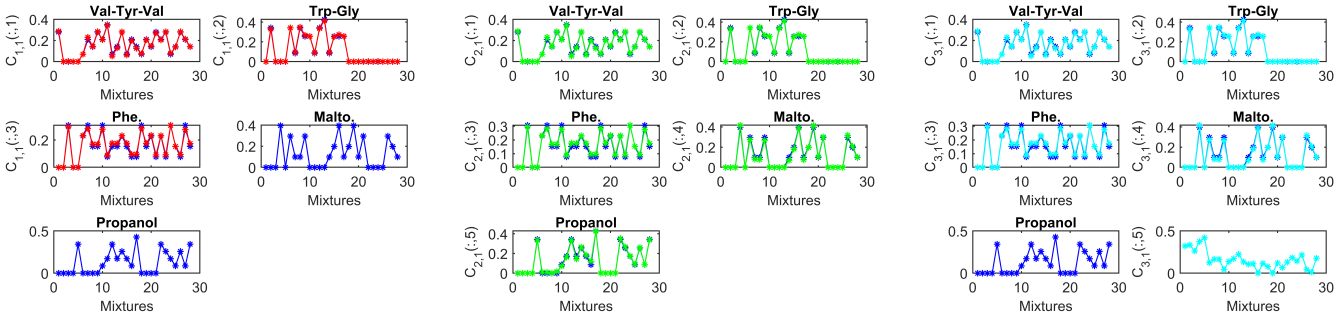
for a fixed number of components R. Here $\eta_{\mathcal{S}_n}$ is the (column-wise) characteristic function of the simplex of dimension n. This regularization enforces that all abundances are nonnegative and sum to one, which is a common assumption when processing hyperspectral and multispectral images. It can be noted that if every column of $\mathbf{A}$ is on the unit simplex, then so are the columns of $\mathbf{HA}$ due to the construction of $\mathbf{H}$. However enforcing both constraints is only redundant at the optimum.

We set up our algorithm by splitting $\mathbf{A}$, $\mathbf{HA}$, $\mathbf{E}$ and $\mathbf{SE}$ respectively into four different variables $\mathbf{A}_m, \mathbf{A}_h, \mathbf{E}_h, \mathbf{E}_m$ constrained by coupling-type 2b, namely

$$\mathbf{A}_m = \boldsymbol{\Delta}_A,\ \mathbf{A}_h = \mathbf{H}\boldsymbol{\Delta}_A,\ \mathbf{E}_m = \mathbf{S}\boldsymbol{\Delta}_E,\ \mathbf{E}_h = \boldsymbol{\Delta}_E\ . \tag{19}$$

Our baseline is the implementation of proximal alternating gradient descent from Lanaras *et. al.* (PALM), which has a lower per-iteration complexity than the proposed AO-ADMM [73], [74]. We will compare both methods in terms of relative reconstruction error RMSE $= \frac{\|\mathbf{X}-\hat{\mathbf{X}}\|_F}{\sqrt{n_1 n_2}}$ given an approximate full-resolution matrix $\hat{\mathbf{X}}$.

Because the problem is highly non-convex, we also need a good initialization. Similarly to our baseline [73], we used



(a) Columns of $\mathbf{C}_{1,1}$ extracted from EEM. (b) Columns of $\mathbf{C}_{2,1}$ extracted from NMR. (c) Columns of $\mathbf{C}_{3,1}$ extracted from LC-MS.

Fig. 12. Joint analysis of measurements from multiple platforms by solving the optimization problem in (17). Blue lines indicate the true design of the experiment while other colors are used to plot the columns of the factor matrices extracted by the model.

SISAL [75] to estimate $\mathbf{E}_h$, while $\mathbf{A}_h$ is obtained by SUNSAL [76]. These methods do not constitue a deterministic initialization, so we picked the best run out of three in terms of RMSE for the baseline (but all runs reached similar results).

Finally, we used the well-known Urban dataset[3], which we degrade artifically to create the two smaller HSI and MSI. It has been reported in the literature than a good rank for decomposing Urban is between four and six, so we set $R = 6$ [77].

Figure 13 shows the relative reconstruction error of the true HSI with respect to time, while Figure 14 shows the final residual images. Both the proposed AO-ADMM algorithm and the baseline have similar outputs, given their very similar residual maps at fixed RMSE (estimated abundances and endmembers are also very similar, see Supplementary Materials). This means that our general framework can successfully perform hyperspectral super-resolution using the model of Lanaras *et. al.* [73]. As expected, the proposed method is faster per-iteration. However, the baseline reaches a given RMSE threshold on average roughly six times faster than our method. This is arguably not a bad performance, given that the baseline is specifically written for hyperspectral super-resolution, with particular code optimization and compiled components. On the other hand, we used our AO-ADMM naively, without paying particular attention to the structured nature of the coupling matrices (except sparsity), as an average user would have done. Our framework however allows for customization: we may change the cost function, the constraints or the coupling types without efforts. On the contrary, the PALM algorithm would not straightfowardly handle a complexification of the constraints as, for instance, projection on the intersection of linear subspaces and the simplex is not trivial. Furthermore, changes to the model require tinkering with the baseline implementation.

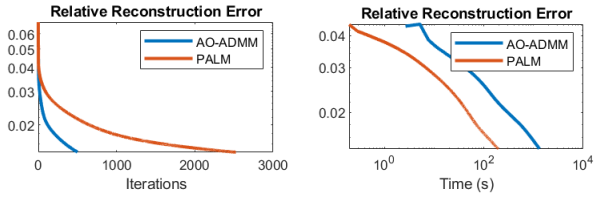


Fig. 13. RMSE per iterations and per time for alternating proximal gradient descent [73] and the proposed AO-ADMM.

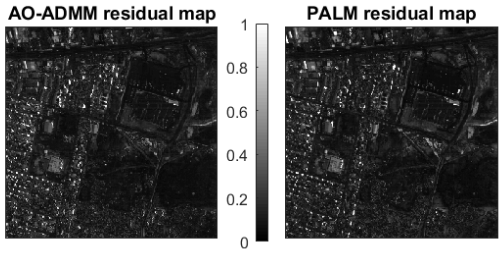


Fig. 14. Pixel-wise residual images, normalized in the $[0, 1]$ range. The final RMSE value is the same for both methods and is set to 0.0144.

VII. Discussions

In this paper, we have presented a flexible algorithmic framework for regularized and linearly coupled tensor factorizations based on ADMM which is able to handle many important loss functions, coupling structures and regularizations in a plug-and-play fashion. Numerical experiments show that our approach is computationally at least comparable, and in linearly coupled cases superior, to other state-of-the-art methods while being more flexible.

Nevertheless, there exist limitations of the current algorithmic framework. Firstly, while the proposed algorithm can incorporate many commonly used non-parametric constraints via the proximal operator, parametric constraints can not be handled straightforwardly. Moreover, using gradient-based numerical optimization to update the estimates of factor matrices, requires, on the one hand, the loss function $\mathcal{L}_i(\cdot, \cdot)$ to be differentiable in the second argument. This excludes, amongst others, the ℓ_1-norm loss. On the other hand, even for differentiable loss functions, the constraints on the entries of the reconstructed tensor that arise from the assumptions on the mean of the data distribution, may be difficult to handle, for example, for binary data following a Bernoulli distribution. Note that in [30], also non-linear link functions, which map

[3]Available at http://www.erdc.usace.army.mil/

the mean of the distribution to the CPD tensor entries, are discussed which have the advantage that certain constraints can be avoided. However, this non-linear transformation changes the factor matrices irreversibly and one should be cautious when interpreting the resulting factors, especially in a coupled setting and when additional constraints are imposed on the factor matrices. For this reason, we refrain from covering non-linear link functions in this paper and in the code.

Furthermore, when coupled datasets have different statistical properties, it is desirable to use different loss functions $\mathcal{L}_i$ to fit each tensor $\boldsymbol{\mathcal{T}}_i$. However, how to choose the weights w_i in equation (5), that compensate for different scales and noise levels in the datasets, becomes a critical issue. The problem of optimal weights is still more or less unresolved. In [23], the authors propose a probabilistic approach for estimating the dispersion parameters while estimating the factor matrices for coupled tensor factorization models with mixed divergences. Another approach is to estimate the variances in the datasets beforehand [78], [79].

A major limitation of the current framework is that linear coupling transformations have to be known. Often, however, they are not known exactly, but might be well approximated by some parameterized function as in [10]. We are therefore currently investigating the possibilities of learning the (parameterized) transformation matrices within the framework. This is also especially important for the automatic identification of shared and unshared components. Another possibility to reveal shared components is the ACMTF model proposed in [65], which supposes that all factors are identical, but only a few components have non-zero strength in each tensor $\boldsymbol{\mathcal{T}}_i$. This model can theoretically be obtained within the AO-ADMM framework by setting $g_{i,d}$ as the sparsity-inducing ℓ_1 norm on the ℓ_2 norms of columns of coupled factors, supposing other factors are normalized.

Future work includes also the extension of this framework to other tensor factorization models than the CPD. Furthermore, it can be worthwhile to consider Nesterov-type acceleration to increase efficiency of the AO-ADMM framework. Accelerations of this type have previously been applied to ALS for fitting CP decompositions [80], [81].

REFERENCES

[1] I. Van Mechelen and A. K. Smilde, "A generic linked-mode decomposition model for data fusion," *Chemometrics and Intelligent Laboratory Systems*, vol. 104, no. 1, pp. 83–94, 2010.
[2] E. Acar, R. Bro, and A. K. Smilde, "Data fusion in metabolomics using coupled matrix and tensor factorizations," *Proceedings of the IEEE*, vol. 103, no. 9, pp. 1602–1620, 2015.
[3] L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion," *IEEE J. Sel. Top. Signal Process.*, vol. 9, no. 4, pp. 586–600, 2015.
[4] D. Lahat, T. Adali, and C. Jutten, "Multimodal data fusion: An overview of methods, challenges, and prospects," *Proceedings of the IEEE*, vol. 103, pp. 1449–1477, 2015.
[5] Y.-R. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, and A. Kelliher, "Metafac: community discovery via relational hypergraph factorization," in *KDD'09*, 2009, pp. 527–536.
[6] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang, "Collaborative filtering meets mobile recommendation: A user-centered approach," in *AAAI'10: Proc. 24th Conf. on Artificial Intelligence*, 2010, pp. 236–241.
[7] B. Ermis, E. Acar, and A. T. Cemgil, "Link prediction in heterogeneous data via generalized coupled tensor factorization," *Data Mining and Knowledge Discovery*, vol. 29, pp. 203–236, 2015.
[8] M. Araujo, P. Ribeiro, and C. Faloutsos, "Tensorcast: Forecasting with context using coupled tensors," in *ICDM'17*, 2017, pp. 71–80.
[9] E. E. Papalexakis, T. M. Mitchell, N. D. Sidiropoulos, C. Faloutsos, P. P. Talukdar, and B. Murphy, "Turbo-SMT: Accelerating coupled sparse matrix-tensor factorizations by 200x," in *SDM'14*, 2014.
[10] S. Van Eyndhoven, B. Hunyadi, L. De Lathauwer, and S. Van Huffel, "Flexible data fusion of EEG-fMRI: Revealing neural-hemodynamic coupling through structured matrix-tensor factorization," in *EUSIPCO'17*, 2017, pp. 26–30.
[11] C. Chatzichristos, M. Davies, J. Escudero, E. Kofidis, and S. Theodoridis, "Fusion of EEG and fMRI via soft coupled tensor decompositions," *EUSIPCO'18*, pp. 56–60, 2018.
[12] E. Acar, C. Schenker, Y. Levin-Schwartz, V. Calhoun, and T. Adali, "Unraveling diagnostic biomarkers of schizophrenia through structure-revealing fusion of multi-modal neuroimaging data," *Frontiers in Neuroscience*, vol. 13, p. 416, 2019.
[13] B. Rivet, M. Duda, A. Guérin-Dugué, C. Jutten, and P. Comon, "Multimodal approach to estimate the ocular movements during EEG recordings: a coupled tensor factorization method," in *EMBC'15*, 2015, pp. 6983–6986.
[14] U. Wunsch, E. Acar, B. P. Koch, K. R. Murphy, P. Schmitt-Kopplin, and C. A. Stedmon, "The molecular fingerprint of fluorescent natural organic matter offers insight into its diagenetic state," *Analytical Chemistry*, vol. 90, no. 24, pp. 14 188–14 197, 2018.
[15] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *J. Math. Phys.*, vol. 6, no. 1, pp. 164–189, 1927.
[16] 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.
[17] 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, pp. 283–319, 1970.
[18] E. Acar, T. G. Kolda, and D. M. Dunlavy, "All-at-once optimization for coupled matrix and tensor factorizations," in *KDD Workshop on Mining and Learning with Graphs*, 2011.
[19] J. Yoo, M. Kim, K. Kang, and S. Choi, "Nonnegative matrix partial co-factorization for drum source separation," in *ICASSP'10*, 2010, pp. 1942–1945.
[20] K. Van Deun, T. F. Wilderjans, R. A. Van Den Berg, A. Antoniadis, and I. Van Mechelen, "A flexible framework for sparse simultaneous component based data integration," *BMC Bioinformatics*, vol. 12, p. 448, 2011.
[21] S. A. Khan, E. Leppaaho, and S. Kaski, "Bayesian multi-tensor factorization," *Machine Learning Research*, vol. 105, p. 233–253, 2016.
[22] T. Yokota, A. Cichocki, and Y. Yamashita, "Linked PARAFAC/CP tensor decomposition and its fast implementation for multi-block tensor analysis," in *ICONIP'12*, 2012, pp. 84–91.
[23] U. Simsekli, B. Ermis, A. T. Cemgil, and E. Acar, "Optimal weight learning for coupled tensor factorization with mixed divergences," in *EUSIPCO*, 2013, pp. 1–5.
[24] R. Cabral Farias, J. E. Cohen, and P. Comon, "Exploring multimodal data fusion through joint decompositions with flexible couplings," *IEEE Trans. Sig. Proc.*, vol. 64, no. 18, pp. 4830–4844, Sep. 2016.
[25] F. M. Almutairi, C. I. Kanatsoulis, and N. D. Sidiropoulos, "Tendi: Tensor disaggregation from multiple coarse views," in *PAKDD*, 2020, pp. 867–880.
[26] C. Kanatsoulis, X. Fu, N. Sidiropoulos, and W.-K. Ma, "Hyperspectral super-resolution: A coupled tensor factorization approach," *IEEE Trans. Sig. Proc.*, vol. 66, pp. 6503 – 6517, 2018.
[27] E. Acar, M. Nilsson, and M. Saunders, "A flexible modeling framework for coupled matrix and tensor factorizations," in *EUSIPCO*, 2014, pp. 111–115.
[28] E. Acar, G. Gurdeniz, M. A. Rasmussen, D. Rago, L. O. Dragsted, and R. Bro, "Coupled matrix factorization with sparse factors to identify potential biomarkers in metabolomics," in *ICDM Workshop on Biological Data Mining and its Applications in Healthcare*, 2012, pp. 1–8.
[29] M. Timmerman and H. Kiers, "Three-way component analysis with smoothness constraints," *Computational Statistics and Data Analysis*, vol. 40, no. 3, pp. 447–470, 9 2002.
[30] D. Hong, T. G. Kolda, and J. A. Duersch, "Generalized canonical polyadic tensor decomposition," *SIAM Review*, vol. 62, no. 1, pp. 133–163, 2020.
[31] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *KDD'08*, 2008, pp. 650–658.
[32] T. Wilderjans, E. Ceulemans, and I. Van Mechelen, "Simultaneous analysis of coupled data blocks differing in size: A comparison of two

weighting schemes," *Comput. Stat. Data Anal.*, vol. 53, pp. 1086–1098, 2009.

[33] R. C. Farias, J. E. Cohen, C. Jutten, and P. Comon, "Joint decompositions with flexible couplings," in *LVA/ICA'15*, 2015, pp. 119–126.

[34] S. Bahargam and E. E. Papalexakis, "Constrained coupled matrix-tensor factorization and its application in pattern and topic detection," in *ASONAM'18: IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2018, pp. 91–94.

[35] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab v3. 0," *available online, URL: www. tensorlab. net*, 2016.

[36] Y. K. Yilmaz, A. T. Cemgil, and U. Simsekli, "Generalised coupled tensor factorisation," in *NIPS'11*, 2011, pp. 2151–2159.

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

[38] 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.

[39] C. Schenker, J. E. Cohen, and E. Acar, "An optimization framework for regularized linearly coupled matrix-tensor factorization," in *EUSIPCO*, 2020.

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

[41] J.-J. Moreau, "Proximité et dualité dans un espace hilbertien," *Bull. Soc. Math. France*, vol. 93, no. 2, pp. 273–299, 1965.

[42] A. Beck, *First-Order Methods in Optimization*. SIAM, 2017.

[43] http://proximity-operator.net/proximityoperator.html.

[44] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.

[45] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, 09 2012.

[46] X. Fu, K. Huang, W. Ma, N. Sidiropoulos, and R. Bro, "Joint tensor factorization and outlying slab suppression with applications," *IEEE Transactions on Signal Processing*, vol. 63, no. 23, pp. 6315–6328, 2015.

[47] S. Smith, A. Beri, and G. Karypis, "Constrained tensor factorization with accelerated AO-ADMM," in *ICP 2017*, 2017, pp. 111–120.

[48] C. Fevotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural Computation*, vol. 21, pp. 793–830, 2009.

[49] D. L. Sun and C. Fevotte, "Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence," in *ICASSP'14*, 2014, pp. 6201–6205.

[50] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers," *IEEE Trans. Sig. Proc.*, vol. 63, no. 20, pp. 5450–5463, 2015.

[51] Y. Wang, R. Chen, J. Ghosh, J. Denny, A. Kho, Y. Chen, B. Malin, and J. Sun, "Rubik: Knowledge guided tensor factorization and completion for health data analytics," in *KDD'15*, 2015, pp. 1265–1274.

[52] A. Afshar, I. Perros, E. E. Papalexakis, E. Searles, J. Ho, and J. Sun, "Copa: Constrained parafac2 for sparse & large datasets," in *CIKM'18: Proc. 27th ACM Int. Conf. Inf. and Knowl. Management*, 2018, pp. 793–802.

[53] C. Sagonas, Y. Panagakis, A. Leidinger, and S. Zafeiriou, "Robust joint and individual variance explained," in *CVPR'17*, 2017, pp. 5739–5748.

[54] C. Schenker, "AO-ADMM code for coupled matrix and tensor factorizations," 2020. [Online]. Available: https://github.com/AOADMM-DataFusionFramework/Matlab-Code

[55] A. Beutel, A. Kumar, E. E. Papalexakis, P. P. Talukdar, C. Faloutsos, and E. P. Xing, "FLEXIFACT: Scalable flexible factorization of coupled tensors on hadoop," in *SDM'14: Proc. SIAM Conf. Data Mining*, 2014, pp. 109–117.

[56] M. Sorensen, I. Domanov, and L. D. Lathauwer, "Coupled canonical polyadic decompositions and (coupled) decompositions in multilinear rank-$(l_{r,n}, l_{r,n}, 1)$ terms — part ii: Algorithms," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 3, pp. 1015–1045, 2015.

[57] B. Jeon, I. Jeon, S. Lee, and U. Kang, "Scout: Scalable coupled matrix-tensor factorization - algorithm and discoveries," in *ICDE'16*, 2016, pp. 811–822.

[58] R. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, 1995.

[59] S. Becker, "L-BFGS-B C code with Matlab wrapper," 2019. [Online]. Available: https://github.com/stephenbeckr/L-BFGS-B-C

[60] E. C. Chi and T. G. Kolda, "Making tensor factorizations robust to non-Gaussian noise," in *NIPS Workshop on Tensors, Kernels, and Machine Learning*, October 2010.

[61] ——, "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.

[62] A. Cichocki, R. Zdunek, S. Choi, R. Plemmons, and S.-i. Amari, "Non-negative tensor factorization using alpha and beta divergences," in *ICASSP'07*, 2007.

[63] A. Cichocki and A.-H. Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE Transactions*, vol. 92-A, pp. 708–721, 03 2009.

[64] G. Chierchia, E. Chouzenoux, P. L. Combettes, and J.-C. Pesquet, *The Proximity Operator Repository. User's guide*.

[65] E. Acar, E. E. Papalexakis, G. Gurdeniz, M. A. Rasmussen, A. J. Lawaetz, M. Nilsson, and R. Bro, "Structure-revealing data fusion," *BMC Bioinformatics*, vol. 15, p. 239, 2014.

[66] L. L. Magoarou and R. Gribonval, "Flexible multilayer sparse approximations of matrices and applications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 688–700, June 2016.

[67] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 644–658, 2014.

[68] B. W. Bader, T. G. Kolda *et al.*, "Matlab tensor toolbox version 3.1," Jun. 2019. [Online]. Available: https://www.tensortoolbox.org

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

[70] G. Tomasi and R. Bro, "A comparison of algorithms for fitting the parafac model," *Comput. Stat. Data Anal.*, vol. 50, no. 7, pp. 1700 – 1734, 2006.

[71] N. Vervliet and L. De Lathauwer, "Numerical optimization based algorithms for data fusion," *Elsevier*, 2018.

[72] G. Tomasi, E. Acar, and R. Bro, "Multilinear models, iterative methods," in *Comprehensive Chemometrics (2nd Edition)*. Elsevier, 2020, pp. 267–304.

[73] C. Lanaras, E. Baltsavias, and K. Schindler, "Hyperspectral super-resolution by coupled spectral unmixing," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3586–3594.

[74] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2014.

[75] J. M. Bioucas-Dias, "A variable splitting augmented lagrangian approach to linear spectral unmixing," in *2009 First workshop on hyperspectral image and signal processing: Evolution in remote sensing*. IEEE, pp. 1–4.

[76] J. M. Bioucas-Dias and M. A. Figueiredo, "Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing," in *2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*. IEEE, pp. 1–4.

[77] F. Zhu, "Hyperspectral unmixing: ground truth labeling, datasets, benchmark performances and survey," *arXiv preprint arXiv:1708.05125*, 2017.

[78] T. Wilderjans, E. Ceulemans, I. Van Mechelen, and R. A. van den Berg, "Simultaneous analysis of coupled data matrices subject to different amounts of noise," *British Journal of Math. and Stat. Psychology*, vol. 64, pp. 277–290, 2011.

[79] Y. Song, J. A. Westerhuis, and A. K. Smilde, "Separating common (global and local) and distinct variation in multiple mixed types data sets," *Journal of Chemometrics*, vol. 34, no. 1, p. e3197, 2020.

[80] D. Mitchell, N. Ye, and H. De Sterck, "Nesterov acceleration of alternating least squares for canonical tensor decomposition: Momentum step size selection and restart mechanisms," arXiv:1810.05846, Tech. Rep., 2019.

[81] A. M. Shun Ang, J. E. Cohen, L. T. Khanh Hien, and N. Gillis, "Extrapolated alternating algorithms for approximate canonical polyadic decomposition," in *ICASSP'20*, 2020, pp. 3147–3151.