Non-parametric B-spline decoupling of multivariate functions

Joppe De Jonghe
Dept. of Computer Science
NUMA, KU Leuven
Geel, Belgium
joppe.dejonghe@kuleuven.be

Mariya Ishteva
Dept. of Computer Science
NUMA, KU Leuven
Geel, Belgium
mariya.ishteva@kuleuven.be

***Abstract*—Many scientific fields and applications require compact representations of multivariate functions. For this problem, decoupling methods are powerful techniques for representing the multivariate functions as a composition of linear transformations and nonlinear univariate functions.**

This work introduces an efficient decoupling algorithm that leverages the use of B-splines to allow a non-parametric estimation of the decoupling's internal functions. The use of B-splines alleviates the problem of choosing an appropriate basis, as in parametric methods, but still allows an intuitive way to adjust the flexibility of the estimated functions. Besides the non-parametric property, the use of B-spline representations allows for easy integration of nonnegativity or monotonicity constraints on the function shapes, which is not trivial for the currently available (non-)parametric decoupling methods.

The proposed algorithm is illustrated on synthetic examples that highlight the flexibility of the B-spline representation and the ease with which a monotonicity constraint can be added. The examples also show that if monotonic functions are required, enforcing a constraint is necessary.

***Index Terms*—Tensor, Decomposition, Decoupling, B-spline**

I. Introduction

Finding compact representations of multivariate functions forms an essential part of many scientific fields, such as block-oriented system identification [1] and deep neural network compression [4]. The decoupling methodology [2] is a powerful technique that aims at representing a function by a composition of linear transformations, with elementwise nonlinear functions sandwiched between them (see Fig. 1). The advantage of the decoupling formulation is that it allows to leverage a tensor decomposition, which makes it attractive in many applications [4], [6]. The decoupled representations can be viewed as neural networks with trainable activation functions, per neuron [4].

Dreesen et al. [2] introduced a tensor-based method for solving the decoupling problem based on first-order information of the given function and the computation of the Canonical Polyadic Decomposition (CPD) of a third-order tensor. In the noiseless case it is guaranteed to solve the decoupling problem thanks to the uniqueness properties of the CPD. To deal with more practical noisy, or non-unique CPD scenarios, several approaches have been proposed, including [3]–[6]. One of the key issues is the estimation of the nonlinear functions in Fig. 1.

To this end, Hollander et al. [6] propose parameterizing the internal functions g_i as polynomials of a certain degree.

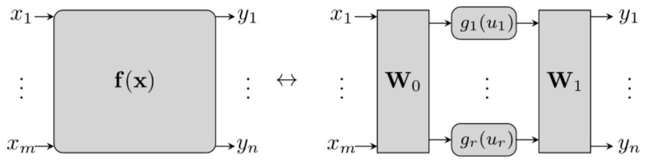

Fig. 1: Decoupling of a multivariate function $\mathbf{f}(\mathbf{x})$ (left) into the model $\mathbf{f}(\mathbf{x}) = \mathbf{W}_1\mathbf{g}(\mathbf{W}_0\mathbf{x})$ (right) constitutes a linear transformation of the input by $\mathbf{W}_0$, followed by branches of univariate functions and a final linear transformation by $\mathbf{W}_1$.

Zniyed et al. [4] provide a more general basis function representation but choose to parameterize the functions g_i as piecewise linear functions for their application of neural network compression. In contrast, Decuyper et al. [3] present an algorithm with a fully non-parametric representation of the decoupling's internal functions.

The drawback of existing approaches is that they mostly assume a simple parametric form of the internal functions (polynomial and piecewise linear) [4], [6]. The exception is the work of Decuyper et al. [3], which presents a non-parametric 'filtering' approach to estimating internal functions, but this approach is mostly heuristic and comes at a higher computational cost.

In this paper, we propose a principled alternative that uses B-splines for the internal functions and develop an algorithm for it. Such a choice allows us to develop a coupled matrix-tensor factorization algorithm that is not only efficient, but also provides a non-parametric estimation of the internal functions. Unlike polynomials, B-splines result in better behavior of the algorithm steps.

II. Notations and tensor background

A. Notations

Matrices and vectors are denoted with bold capital and lowercase letters, respectively. Tensors are denoted by calligraphic capital letters. For a third order tensor $\mathcal{X}$ of size $I \times J \times K$, the i-th horizontal, j-th lateral and k-th frontal slice are denoted by $\mathcal{X}_{i,:,:}$, $\mathcal{X}_{:,j,:}$ and $\mathcal{X}_{:,:,k}$ respectively. The operation $\text{unfold}_k(\mathcal{X})$ unfolds the tensor $\mathcal{X}$ over its k-th mode as described in [8]. The i-th row and j-th column of a matrix $\mathbf{A}$ are denoted as $\mathbf{A}^{i,:}$ and $\mathbf{A}^{:,j}$ respectively. The symbol $\odot$ denotes the Khatri-Rao product. Finally, the $\text{diag}(.)$ operation forms a diagonal matrix where the main diagonal is the vector that is provided as parameter and $\|.\|$ denotes the Frobenius norm of a tensor, defined as the square root of the sum of the squares of its elements.

B. Canonical polyadic decomposition

The *canonical polyadic decomposition* (CPD) [8] of a third-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ expresses the tensor as a sum of rank-one tensors, or alternatively, the tensor $\mathcal{X}$ admits a CPD if its slices can be represented as

$$\mathcal{X}_{:,:,k} = \mathbf{A} \cdot \text{diag}(\mathbf{C}^{k,:}) \cdot \mathbf{B}^\top, \text{ for } k = 1, 2, \ldots, K, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{I\times r}$, $\mathbf{B} \in \mathbb{R}^{J\times r}$ and $\mathbf{C} \in \mathbb{R}^{K\times r}$ are the factor matrices. We use the notation $\mathcal{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$. The canonical rank is the smallest value r for which equation (1) holds.

The CPD described in equation (1) is unique under mild conditions. Several sufficient uniqueness conditions exist (for example Kruskal's condition), see [9] for an overview. Uniqueness here means that the CPD is unique up to the following scaling and permutation ambiguities [8]:

$$\mathcal{X} = [\![\mathbf{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}_\mathbf{A}, \mathbf{B}\boldsymbol{\Pi}\boldsymbol{\Lambda}_\mathbf{B}, \mathbf{C}\boldsymbol{\Pi}\boldsymbol{\Lambda}_\mathbf{C}]\!],$$

with permutation matrix $\boldsymbol{\Pi} \in \mathbb{R}^{r\times r}$ and diagonal matrices $\boldsymbol{\Lambda}_\mathbf{A}$, $\boldsymbol{\Lambda}_\mathbf{B}$, $\boldsymbol{\Lambda}_\mathbf{C}$ for which $\boldsymbol{\Lambda}_\mathbf{A}\boldsymbol{\Lambda}_\mathbf{B}\boldsymbol{\Lambda}_\mathbf{C} = \mathbf{I}$.

III. Decoupling

A. The decoupling problem

Dreesen et al. [2] formulate the decoupling problem as follows: given a multivariate vector function $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^n$, find a decoupled representation of $\mathbf{f}(\mathbf{x})$ such that

$$\mathbf{f}(\mathbf{x}) = \mathbf{W}_1\mathbf{g}(\mathbf{W}_0\mathbf{x}) \tag{2}$$

with linear transformation matrices $\mathbf{W}_1 \in \mathbb{R}^{n\times r}, \mathbf{W}_0 \in \mathbb{R}^{r\times m}$ and $\mathbf{g}(\mathbf{u}) = \begin{bmatrix} g_1(u_1) & g_2(u_2) & \cdots & g_r(u_r)\end{bmatrix}^\top \in \mathbb{R}^r$ consists of univariate functions $g_i : \mathbb{R} \to \mathbb{R}$. Dreesen et al. [2] propose a tensor-based solution strategy that uses the first-order information of $\mathbf{f}(\mathbf{x})$. This first-order information is encapsulated in the Jacobian $\mathbf{J}_\mathbf{f}(\mathbf{x})$

$$\mathbf{J}_\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial f_1(\mathbf{x})}{\partial x_m} \\ \vdots & & \vdots \\ \dfrac{\partial f_n(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial f_n(\mathbf{x})}{\partial x_m} \end{bmatrix} \in \mathbb{R}^{n\times m},$$

where, under the assumption that $\mathbf{f}(\mathbf{x})$ follows the model in equation (2), it holds that

$$\mathbf{J}_\mathbf{f}(\mathbf{x}) = \mathbf{W}_1 \text{ diag}(\mathbf{g}'(\mathbf{W}_0\mathbf{x}))\ \mathbf{W}_0.$$

Next, $\mathbf{J}_\mathbf{f}(\mathbf{x})$ can be evaluated in S sample points $\mathbf{x}^{(s)} \in \mathbb{R}^m$, for $s = 1, 2, \ldots, S$,

$$\mathbf{J}_\mathbf{f}(\mathbf{x}^{(s)}) = \mathbf{W}_1 \text{ diag}(\mathbf{g}'(\mathbf{W}_0\mathbf{x}^{(s)}))\ \mathbf{W}_0 = \mathbf{W}_1\ \mathbf{D}_\mathbf{g}^{(s)}\ \mathbf{W}_0.$$

Note here that 1) $\mathbf{D}_\mathbf{g}^{(s)} \in \mathbb{R}^{r\times r}$ is diagonal and 2) $\mathbf{W}_0$ and $\mathbf{W}_1$ are independent of the sample point $\mathbf{x}^{(s)}$ [2].

As a result, stacking the Jacobian matrices $\mathbf{J}_\mathbf{f}(\mathbf{x}^{(s)})$, for $s = 1, 2, \ldots, S$, as frontal slices of a third-order tensor yields a Jacobian tensor $\mathcal{J} \in \mathbb{R}^{n\times m\times S}$, for which

$$\mathcal{J}_{:,:,s} = \mathbf{J}_\mathbf{f}(\mathbf{x}^{(s)}) = \mathbf{W}_1\ \mathbf{D}_\mathbf{g}^{(s)}\ \mathbf{W}_0, \tag{3}$$

for $s = 1, 2, \ldots, S$. This shows that by construction, the tensor $\mathcal{J}$ admits a CPD, $\mathcal{J} = [\![\mathbf{W}_1, \mathbf{W}_0^\top, \mathbf{G}]\!]$, where for $\mathbf{W}_0\mathbf{x}^{(s)} = \mathbf{u}^{(s)} \in \mathbb{R}^r$, for $s = 1, 2, \ldots, S$, the factor matrix $\mathbf{G}$ is

$$\mathbf{G} = \begin{bmatrix} g_1'(u_1^{(1)}) & \ldots & g_r'(u_r^{(1)}) \\ \vdots & & \vdots \\ g_1'(u_1^{(S)}) & \ldots & g_r'(u_r^{(S)}) \end{bmatrix}. \tag{4}$$

Equation (3) indicates that computing the CPD of $\mathcal{J}$ yields the factor matrices $\mathbf{W}_1$ and $\mathbf{W}_0$ of the decoupled model (2) as well as the matrix $\mathbf{G} \in \mathbb{R}^{S\times r}$ which contains first-order information of the internal functions g_i, for $i = 1, 2, \ldots, r$.

The tensor-based solution strategy is summarized as:

1) Evaluate the Jacobian of $\mathbf{f}(\mathbf{x})$ in S sample points $\mathbf{x}^{(s)}$, yielding $\mathbf{J}_\mathbf{f}(\mathbf{x}^{(1)}), \mathbf{J}_\mathbf{f}(\mathbf{x}^{(2)}), \ldots, \mathbf{J}_\mathbf{f}(\mathbf{x}^{(S)})$.
2) Stack the Jacobian matrices $\mathbf{J}_\mathbf{f}(\mathbf{x}^{(s)})$, for $s = 1, 2, \ldots, S$, into the tensor $\mathcal{J} \in \mathbb{R}^{n\times m\times S}$.
3) Compute the CPD of $\mathcal{J}$, yielding $\mathbf{W}_1, \mathbf{W}_0$ and $\mathbf{G}$ up to scaling and permutation ambiguities.
4) Use the first-order information in $\mathbf{G}$ to determine the representation of the decoupling's internal functions g_i.

B. Representation of the internal functions

Ideally, a representation is chosen such that the decoupling's internal functions can be represented in a non-parametric way without adding substantial algorithmic complexity. To this end, this work introduces the use of B-spline functions to represent the internal functions g_i (or derivatives g_i'), for $i = 1, 2, \ldots, r$,

$$g_i(u_i) = c_{i,0} + \sum_{j=1}^{df} c_{i,j} B_{j,d}^{\boldsymbol{\Delta}_i}(u_i), \text{ differentiate to get } g_i', \tag{5}$$

or

$$g_i'(u_i) = \sum_{j=1}^{df} c_{i,j} B_{j,d-1}^{\boldsymbol{\Delta}_i}(u_i), \text{ integrate to get } g_i, \tag{6}$$

where df, d and $\boldsymbol{\Delta}_i$ are the degrees of freedom (DoF), order and knot vector of the spline, respectively. The coefficients $c_{i,j}$ are to be learned as part of the decoupling problem.

The proposed B-spline representation allows to easily incorporate constraints on the internal functions g_i, such as nonnegativity or monotonicity, by constraining the coefficients of the spline to be nonnegative. This is not trivial for a polynomial parameterization or the non-parametric method of Decuyper et al. [3].

C. Optimization problem

A critical part of the tensor-based solution strategy is the computation of the CPD of $\mathcal{J}$. The following optimization problem formulates the unstructured CPD of $\mathcal{J}$

$$\min_{\mathbf{W}_1, \mathbf{W}_0, \mathbf{G}} \|\mathcal{J} - [\![\mathbf{W}_1, \mathbf{W}_0^\top, \mathbf{G}]\!]\|^2. \tag{7}$$

However, optimization problem (7) only uses first-order information of $\mathbf{f}(\mathbf{x})$. Because of this, it is unable to approximate the constant terms of the internal functions g_i and the resulting system $\widehat{\mathbf{f}}(\mathbf{x})$ will show a bias relative to the actual system $\mathbf{f}(\mathbf{x})$. The two main strategies to solve this problem are 1) correcting the bias of the computed decoupling $\widehat{\mathbf{f}}(\mathbf{x})$ as a whole in a second step, as done by Decuyper et al. [3], and 2) adding zeroth-order information into the optimization problem to directly estimate the constant terms of the internal functions g_i, as done by Zniyed et al. [4].

This work adopts the strategy of [4] and integrates a zeroth-order information matrix $\mathbf{F} \in \mathbb{R}^{n\times S}$ into optimization problem (7). The structure of $\mathbf{F}$ is given by

$$\begin{aligned}\mathbf{F} &= \begin{bmatrix}\mathbf{f}(\mathbf{x}^{(1)}) & \mathbf{f}(\mathbf{x}^{(2)}) & \cdots & \mathbf{f}(\mathbf{x}^{(S)})\end{bmatrix} \\ &= \mathbf{W}_1 \begin{bmatrix}\mathbf{g}(\mathbf{u}^{(1)}) & \mathbf{g}(\mathbf{u}^{(2)}) & \cdots & \mathbf{g}(\mathbf{u}^{(S)})\end{bmatrix} \\ &= \mathbf{W}_1 \cdot \mathbf{R}^\top,\end{aligned} \tag{8}$$

where $\mathbf{u}^{(s)} = \mathbf{W}_0\mathbf{x}^{(s)} \in \mathbb{R}^r$ and $\mathbf{R} \in \mathbb{R}^{S\times r}$ contains zeroth-order information of the internal functions g_i.

Incorporating $\mathbf{F}$ into optimization problem (7) results in a coupled matrix-tensor factorization (CMTF) [4], [11]

$$\min_{\mathbf{W}_1,\mathbf{W}_0,\mathbf{G},\mathbf{R}} \|\mathcal{J} - [\![\mathbf{W}_1, \mathbf{W}_0^\top, \mathbf{G}]\!]\|^2 + \lambda\|\mathbf{F} - \mathbf{W}_1\mathbf{R}^\top\|^2. \tag{9}$$

This problem still computes an unstructured CPD of $\mathcal{J}$. However if the reconstruction error of the CPD is non-zero, or the computed CPD is not unique then there is no guarantee that the computed $\mathbf{G}$ and $\mathbf{R}$ have the required structure.

We take this problem into account by forcing the required structure onto $\mathbf{G}$ and $\mathbf{R}$ through the B-spline representation of the internal functions given in equations (5) and (6). Equations (4) and (8) show the structure of $\mathbf{G}$ and $\mathbf{R}$ respectively. This structure is enforced by adding the following constraints on the columns of $\mathbf{G}$ and $\mathbf{R}$

$$\mathbf{G}^{:,j} = \mathbf{B}_j \cdot \mathbf{c}_j \quad \text{for } j = 1,2,\dots,r, \tag{10}$$

$$\mathbf{R}^{:,j} = \widetilde{\mathbf{B}}_j \cdot \mathbf{c}_j \quad \text{for } j = 1,2,\dots,r, \tag{11}$$

where $\mathbf{c}_j = \begin{bmatrix}c_{j,0} & c_{j,1} & \cdots & c_{j,df}\end{bmatrix}^\top \in \mathbb{R}^{df+1}$. The matrices $\mathbf{B}_j \in \mathbb{R}^{S\times df+1}$ and $\widetilde{\mathbf{B}}_j \in \mathbb{R}^{S\times df+1}$ are the B-spline design matrices, evaluated at the values $u_j^{(s)} = (\mathbf{W}_0\mathbf{x}^{(s)})_j$, for $s = 1,2,\dots,S$, with an extra first column such that $\mathbf{B}^{:,1}$ is a column of zeros and $\widetilde{\mathbf{B}}^{:,1}$ is a column of ones. The extra column is to take into account the constant term $c_{j,0}$ in $\mathbf{c}_j$.

Incorporating (10) and (11) as constraints in optimization problem (9) yields the final optimization problem

$$\begin{aligned}\min_{\substack{\mathbf{W}_1,\mathbf{W}_0,\\ \{\mathbf{c}_j\}_{j=1}^r}} \quad & \|\mathcal{J} - [\![\mathbf{W}_1, \mathbf{W}_0^\top, \mathbf{G}]\!]\|^2 + \lambda\|\mathbf{F} - \mathbf{W}_1\mathbf{R}^\top\|^2 \\ \text{s. t.} \quad & \mathbf{G}^{:,j} = \mathbf{B}_j \cdot \mathbf{c}_j \quad \text{for } j = 1,2,\dots,r, \\ & \mathbf{R}^{:,j} = \widetilde{\mathbf{B}}_j \cdot \mathbf{c}_j \quad \text{for } j = 1,2,\dots,r.\end{aligned} \tag{12}$$

The λ parameter is typically set to a fixed low value, 0.01 or 0.1, and stays fixed during execution or is increased over time.

IV. Algorithm

This work uses the efficient projection strategy algorithm introduced by Zniyed et al. [4] to solve optimization problem (12). However, the projection step is adapted to facilitate the B-spline representations (5) and (6) and allow nonnegativity or monotonicity constraints on the internal functions. The use of B-splines combines the efficiency of the projection algorithm with the non-parametric property of Decuyper's algorithm [3].

Algorithm 1 shows the full algorithm, called CMTF-BSD (**CMTF B-S**pline **D**ecoupling). The CMTF-BSD algorithm normalizes the columns of $\mathbf{W}_0^\top$ for improved conditioning, which is not part of the algorithm proposed by Zniyed [4]. The normalization procedure is given in Algorithm 2.

Algorithm 1 CMTF-BSD algorithm

```
Input: $\mathcal{J} \in \mathbb{R}^{n\times m\times S}, \mathbf{F} \in \mathbb{R}^{n\times S}, df, d, r,$ samples $\in \mathbb{R}^{m\times S}$
 1: $\mathbf{W}_0, \mathbf{G}, \mathbf{R} \leftarrow$ Random initialization
 2: while stop criteria not met do
 3:     $\mathbf{W}_1 \leftarrow \arg\min_{\mathbf{W}_1} \|\text{unfold}_1(\mathcal{J}) - \mathbf{W}_1 \cdot (\mathbf{G} \odot \mathbf{W}_0^\top)^\top\|^2$
 4:                 $+\lambda\|\mathbf{F} - \mathbf{W}_1\mathbf{R}^\top\|^2$
 5:     $\mathbf{W}_0 \leftarrow \arg\min_{\mathbf{W}_0} \|\text{unfold}_2(\mathcal{J}) - \mathbf{W}_0^\top \cdot (\mathbf{G} \odot \mathbf{W}_1)^\top\|^2$
 6:     $\mathbf{W}_0, \mathbf{W}_1 \leftarrow$ Normalize_columns_$\mathbf{W}_0^\top(\mathbf{W}_0, \mathbf{W}_1)$
 7:     $\mathbf{G} \leftarrow \arg\min_{\mathbf{G}} \|\text{unfold}_3(\mathcal{J}) - \mathbf{G} \cdot (\mathbf{W}_0^\top \odot \mathbf{W}_1)^\top\|^2$
 8:     $\mathbf{R} \leftarrow \arg\min_{\mathbf{R}} \|\mathbf{F} - \mathbf{W}_1 \cdot \mathbf{R}^\top\|^2$
 9:     xSamples $\leftarrow \mathbf{W}_0 \cdot$ samples
10:     $\mathbf{G}, \mathbf{R} \leftarrow$ Bspline_projection $(\mathbf{G}, \mathbf{R}, df, d,$ xSamples$)$
11: end while
Output: $\mathbf{W}_1, \mathbf{W}_0, \mathbf{G}, \mathbf{R}$
```

Algorithm 2 Normalize_columns_$\mathbf{W}_0^\top$

```
Input: $\mathbf{W}_0, \mathbf{W}_1$
 1: for $i = 1, 2, \dots, r$ do
 2:     $\beta \leftarrow \|(\mathbf{W}_0^\top)^{:,i}\|$
 3:     $(\mathbf{W}_0^\top)^{:,i} \leftarrow (\mathbf{W}_0^\top)^{:,i}/\beta$
 4:     $\mathbf{W}_1^{:,i} \leftarrow \beta\, \mathbf{W}_1^{:,i}$
 5: end for
Output: $\mathbf{W}_0, \mathbf{W}_1$
```

Algorithm 3 shows the B-spline projection step (line 10 in Algorithm 1) in more detail. The Determine_knots function retrieves the knots from the input values $\mathbf{x}_j$ based on quantiles, which depend on the degrees of freedom df and the order d of the spline [7]. The Design_matrix(.) function constructs the B-spline design matrix for the given parameters $\mathbf{\Delta}_j, df$ and d, evaluated at the points $\mathbf{x}_j$. The Differentiate(.) and Integrate(.) functions construct the B-spline design matrix for the derivatives and indefinite integrals respectively, of the B-spline basis that was used to construct the design matrix that is given as a parameter, evaluated in the same points.

Enforcing a monotonicity or nonnegativity constraint on the internal functions g_i can be done by using nonnegative least squares to solve for $\mathbf{c}_j$ on line 11 in algorithm 3.

Algorithm 3 Bspline_projection

Input: $\mathbf{G}, \mathbf{R}, df, d, \text{xSamples} \in \mathbb{R}^{r \times S}$

```
 1: for $j = 1, 2, \dots, r$ do
 2:     $\mathbf{x}_j \leftarrow \text{xSamples}^{j,:}$
 3:     $\mathbf{\Delta}_j \leftarrow \text{Determine\_knots}(\mathbf{x}_j, df, d)$
 4:     if Representation (5) used then
 5:         $\widetilde{\mathbf{B}} \leftarrow \text{Design\_matrix}(\mathbf{x}_j, \mathbf{\Delta}_j, df, d)$
 6:         $\mathbf{B} \leftarrow \text{Differentiate}(\widetilde{\mathbf{B}})$
 7:     else if Representation (6) used then
 8:         $\mathbf{B} \leftarrow \text{Design\_matrix}(\mathbf{x}_j, \mathbf{\Delta}_j, df, d)$
 9:         $\widetilde{\mathbf{B}} \leftarrow \text{Integrate}(\mathbf{B})$
10:     end if
11:     $\mathbf{c}_j \leftarrow \arg\min_{\mathbf{c}_j} \|\mathbf{G}^{:,j} - \mathbf{B}\cdot\mathbf{c}_j\|^2 + \lambda\|\mathbf{R}^{:,j} - \widetilde{\mathbf{B}}\cdot\mathbf{c}_j\|^2$
12:     $\mathbf{G}^{:,j} \leftarrow \mathbf{B}\cdot\mathbf{c}_j, \quad \mathbf{R}^{:,j} \leftarrow \widetilde{\mathbf{B}}\cdot\mathbf{c}_j$
13: end for
```

Output: $\mathbf{G}, \mathbf{R}$

V. Experiments and Results

A. Metrics

The following metrics are used in the experiments

$$\text{Error}(\mathcal{J}) = \frac{\|\mathcal{J} - \widehat{\mathcal{J}}\|^2}{\|\mathcal{J}\|^2},$$

$$e_i = \frac{\sqrt{\frac{1}{S}\sum_{s=1}^{S}\left(f_i(\mathbf{x}^{(s)}) - \widehat{f}_i(\mathbf{x}^{(s)})\right)^2}}{\sqrt{\frac{1}{S}\sum_{s=1}^{S}\left(f_i(\mathbf{x}^{(s)}) - \mathbb{E}(f_i)\right)^2}} \times 100,$$

where $\widehat{\mathcal{J}}$ is the approximation of $\mathcal{J}$ resulting from the CMTF-BSD algorithm. Note here that e_i is a relative error on the ith output of $\mathbf{f}(\mathbf{x})$, as a percentage and $\mathbb{E}(f_i)$ is the average value of the ith output over the S sampling points.

B. Effect of DoF and degree of spline representation

In this section, the CMTF-BSD algorithm is executed on a system with trigonometric internal functions

$$\mathbf{f}_{trig}(\mathbf{x}) = \mathbf{W}_1\mathbf{g}(\mathbf{W}_0\mathbf{x}), \tag{13}$$

$$\mathbf{W}_1 = \begin{bmatrix} -1.7 & -2.3 & 2.5 \\ 0.5 & -0.5 & 0.2 \end{bmatrix}, \mathbf{W}_0 = \begin{bmatrix} 2.1 & -1 \\ 0.4 & -1.8 \\ -1.6 & -0.2 \end{bmatrix},$$

$$\mathbf{g}(\mathbf{u}) = \begin{bmatrix} sin(u_1) + 2 \\ cos(2\cdot u_2) - 1.5 \\ sin(2\cdot u_3) + \dfrac{u_3}{2} \end{bmatrix}$$

The goal of the CMTF-BSD(.) algorithm is to retrieve the decoupled representation (13) from the Jacobian tensor $\mathcal{J}$ and zeroth-order information matrix $\mathbf{F}$. The decoupled representation contains 3 internal functions and only 2 inputs and outputs, leading to a non-unique CPD of $\mathcal{J}$. Note that the experiments in this work use the known, correct, canonical rank r. In a practical setting, different ranks r will have to be tested to determine the optimal one.

For the experiment, the algorithm is executed 30 times for different spline degrees ($d = 1, 2, 3$) and DoF ($4, 6, \dots, 28$), using representation (5). After executing the CMTF-BSD algorithm, the discovered internal function shapes are interpolated by polynomials of degree 10. The Jacobian is constructed with $S = 100$ sample points drawn uniformly from $[-1.5, 1.5]^2$, yielding the tensor $\mathcal{J} \in \mathbb{R}^{2\times2\times100}$ and matrix $\mathbf{F} \in \mathbb{R}^{2\times100}$. Each execution draws a new set of 100 sample points.

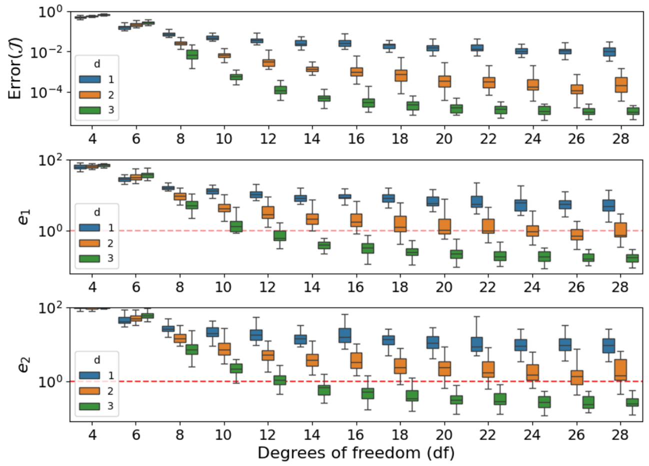

Fig. 2: Results for applying the CMTF-BSD(.) Algorithm 1 to the sytem $\mathbf{f}_{trig}$ of equation (13), for 30 executions per (d, df) pair where $d \in \{1, 2, 3\}$ and $df \in \{4, 6, \dots, 28\}$. Top figure: the relative reconstruction error of the Jacobian tensor $\mathcal{J}$; middle figure: the relative error of the computed system for the first output; bottom figure: the relative error for the second output. The red dotted line indicates an error of 1%.

Fig. 2 shows the results for $\mathbf{f}_{trig}$ (outliers are not plotted for readability). The cubic spline representation, i.e., $d = 3$, performs the best, yielding output errors below 1% when $df \geq 12$. The approximation of $\mathbf{f}_{trig}$ is worse for $d = 2$, as less flexibility requires a higher df value to reach output errors below 1%. The case for which $d = 1$ gives the worst results. The executions for which the output errors drop below 1% are able (or close) to recover the system $\mathbf{f}_{trig}$, even though the CPD itself is not unique.

The problem of recovering $\mathbf{f}_{trig}$ can be seen as a regression task for which the case of $d = 1$ is not optimal since this results in piecewise linear functions g_i that are interpolated by polynomials in a second step. For the piecewise linear case of $d = 1$, a neural network or decision boundary problem such as in [4] is more relevant. Do note here that compared to [4], achieving piecewise linear functions does not require a different basis as we just set the degree of the B-spline representation equal to 1.

C. Monotonically increasing internal functions

This section explores the incorporation of a monotonicity constraint on the decoupling's internal functions g_i. The constraint is enforced through the use of the B-spline representation (6) where the coefficients $\mathbf{c}_j$ on line 11 of the Bspline_projection Algorithm 3 are computed using non-negative least squares. Since B-spline basis functions are nonnegative, this results in nonnegative derivatives g_i' and monotonically increasing internal functions g_i.

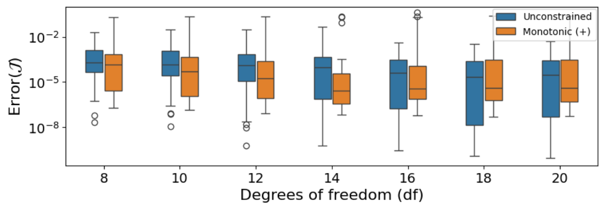

Fig. 3: Reconstruction errors for the Jacobian tensor $\mathcal{J}$, for 30 executions of the CMTF-BSD(.) Algorithm 1 and degrees of freedom $df \in \{8, 10, 12, \ldots, 20\}$, with and without monotonicity constraint. The monotonic (+) results compute the coefficients on line 11 of Algorithm 3 with nonnegative least squares to retrieve monotonically increasing functions.

However, it is possible when using nonnegative least squares that during intermediate steps of the algorithm the coefficients $\mathbf{c}_j$ are computed to be all zero, which crashes the algorithm if not taken into account. To take this into account, an extra check is added after the computation of the coefficients $\mathbf{c}_j$ where, if the computed coefficients are all zero, the function g_i' is replaced by a monotonically increasing LeakyReLU activation function [10] with a slope of 0.5 for the negative part.

The system used for the experiment has 3 inputs, 3 outputs and 3 monotonically increasing internal functions
$\mathbf{g}(\mathbf{u}) = \begin{bmatrix} g_1(u_1) & g_2(u_2) & g_3(u_3) \end{bmatrix}^\top$,

$$g_1(u_1) = \frac{u_1^3}{3} + u_1, \quad g_2(u_2) = e^{u_2}, \quad g_3(u_3) = \frac{1}{1+e^{-u_3}}.$$

The entries of the factor matrices $\mathbf{W}_0 \in \mathbb{R}^{3\times 3}$ and $\mathbf{W}_1 \in \mathbb{R}^{3\times 3}$ are randomly drawn from $\mathcal{U}(-2, 2)$, for each execution of the algorithm.

Fig. 3 shows the results for 30 executions of the CMTF-BSD algorithm for the described system with $d = 4$ and $df \in \{8, 10, 12, \ldots, 20\}$. The algorithm is executed both with and without the monotonicity constraint, denoted as 'Monotonic (+)' and 'Unconstrained' respectively. For the 'Monotonic (+)' results, an unconstrained decoupling is used as initialization.

Fig. 3 shows that the median reconstruction errors for $\mathcal{J}$ are improved by adding the monotonicity constraint. More importantly, Table I shows that for the unconstrained case, even if the underlying functions of the system are monotonically increasing or decreasing, there is no guarantee that the resulting functions are monotonic. On the other hand, by enforcing the monotonicity constraint the resulting decouplings from all 30 executions have certified monotonically increasing internal functions.

TABLE I: Number of executions of the CMTF-BSD algorithm (out of 30) for $df \in \{8, 10, 12, \ldots, 20\}$, where the resulting decoupling has certified monotonic internal functions.

	Degrees of freedom (df)						
	8	**10**	**12**	**14**	**16**	**18**	**20**
Unconstrained	9	8	8	12	14	16	14
Monotonic (+)	**30**	**30**	**30**	**30**	**30**	**30**	**30**

VI. Conclusion

This work introduced the CMTF-BSD decoupling algorithm, which combines the efficiency of the decoupling algorithm of Zniyed [4] and the non-parametric property of that of Decuyper [3]. This is done by 1) using the projection algorithm of [4] and 2) incorporating a B-spline representation of the internal functions to allow non-parametric estimation. In addition, as shown in the experiments, the use of B-splines allows to easily enforce a nonnegativity or monotonicity constraint, which is not trivial for a polynomial basis or the non-parametric method of Decuyper [3]. The constrained example in section V-C shows good results for the incorporation of a monotonicity constraint and the fact that, even when the underlying functions are monotonic, adding the constraint is necessary for guaranteed monotonicity. Furthermore, the example in section V-B shows that the CMTF-BSD algorithm performs well on a system with trigonometric functions, which would require a high degree polynomial to approximate. The CMTF-BSD algorithm achieves stable output errors below 1% for different configurations of d and df and is able to recover the underlying system despite a non-unique unstructured CPD.

Future work includes analyzing the use of other spline types such as smoothing splines and natural cubic splines [7], improving the incorporation of constraints, providing a formal complexity analysis of the algorithm to compare with Decuyper [3], applying the CMTF-BSD algorithm to different applications in system identification and neural network compression and formalizing the theoretical basis of the proposed algorithm through existing B-spline approximation theory.

Acknowledgment

This work was supported by the FWO (FWO Vlaanderen) fundamental research fellowship 11A2H25N and KU Leuven C2 project 3E230536-SPECTRAI.

References

[1] Dreesen, P., Esfahani, A. F., Stoev, J., Tiels, K., & Schoukens, J. (2016, January). Decoupling nonlinear state-space models: case studies. In Proceedings of the International Conference on Noise and Vibration Engineering (ISMA) (pp. 2639-2646).
[2] Dreesen, P., Ishteva, M., & Schoukens, J. (2015). Decoupling multivariate polynomials using first-order information and tensor decompositions. SIAM J. Matrix Anal. Appl., 36(2), 864-879.
[3] Decuyper, J., Tiels, K., Weiland, S., Runacres, M. C., & Schoukens, J. (2022). Decoupling multivariate functions using a nonparametric filtered tensor decomposition. Mech. Syst. Signal Process., 179, 109328.
[4] Zniyed, Y., Usevich, K., Miron, S., & Brie, D. (2021, October). A tensor-based approach for training flexible neural networks. In 2021 55th Asilomar Conf. Signals Syst. Comput. (pp. 1673-1677). IEEE.
[5] Zniyed, Y., Usevich, K., Miron, S., & Brie, D. (2021). Learning nonlinearities in the decoupling problem with structured CPD. IFAC-PapersOnLine, 54(7), 685-690.
[6] Hollander, G. (2017). Multivariate polynomial decoupling in nonlinear system identification.
[7] de Boor, C. (1978). A Practical Guide to Splines, Springer.
[8] Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. SIAM review, 51(3), 455-500.
[9] Sidiropoulos, N. D., et al. (2017). Tensor decomposition for signal processing and machine learning. IEEE Trans. Signal Process., 65(13), 3551-3582.
[10] Lederer, J. (2021). Activation functions in artificial neural networks: A systematic overview. arXiv preprint arXiv:2101.09957.
[11] Liu, Y. (Ed.). (2021). Tensors for data processing: theory, methods, and applications. Academic Press.