Nonnegative Matrix Factorization over Continuous Signals using Parametrizable Functions☆

Cécile Hautecoeur*, François Glineur

Université catholique de Louvain - CORE and ICTEAM Institute B-1348 Louvain-la-Neuve - Belgium

Abstract

Nonnegative matrix factorization is a popular data analysis tool able to extract significant features from nonnegative data. We consider an extension of this problem to handle functional data, using parametrizable nonnegative functions such as polynomials or splines. Factorizing continuous signals using these parametrizable functions improves both the accuracy of the factorization and its smoothness. We introduce a new approach based on a generalization of the Hierarchical Alternating Least Squares algorithm. Our method obtains solutions whose accuracy is similar to that of existing approaches using polynomials or splines, while its computational cost increases moderately with the size of the input, making it attractive for large-scale datasets.

Keywords: Nonnegative matrix factorization, hierarchical alternating least squares (HALS), functional nonnegative matrix factorization, (projection on) nonnegative polynomials, (projection on) nonnegative splines.

1. Introduction

1.1. Nonnegative matrix factorization

Nonnegative matrix factorization (NMF) is a commonly used linear dimensionality reduction technique for nonnegative data. This method compresses data and is able to filter noise. It expresses data vectors using a part-based representation [1, 2] and extracts nonnegative characteristic features from the dataset. To do so, NMF takes a collection of nonnegative vectors, concatenated as the columns of input matrix $\boldsymbol{Y} \in \mathbb{R}^{m\times n}_+$, and tries to decompose each of them as a nonnegative linear combination (with coefficients in $\boldsymbol{X} \in \mathbb{R}^{r\times n}_+$) of a few nonnegative basis vectors, contained in the columns of $\boldsymbol{A} \in \mathbb{R}^{m\times r}_+$. Since an exact decomposition is in general not achievable, several cost functions can be considered to measure the accuracy of the approximation, such as the commonly used Frobenius distance:

$$\min_{\boldsymbol{A}\in\mathbb{R}^{m\times r}_+,\ \boldsymbol{X}\in\mathbb{R}^{r\times n}_+} ||\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{X}||_F^2 \qquad \text{(NMF)}$$

The factorization performed in NMF problems is in general not unique [3]. Therefore, a penalty term on the objective function is often added to take into account a priori knowledge on the data, such as sparsity [4], smoothness [1], orthogonality [5], etc.

NMF is a non-convex problem, and is NP-Hard [6]. However, it is convex with respect to $\boldsymbol{A}$ when $\boldsymbol{X}$ is fixed, and vice-versa. Moreover, as $||\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{X}||_F^2 = ||\boldsymbol{Y}^\top - \boldsymbol{X}^\top\boldsymbol{A}^\top||_F^2$, the problem is symmetric in $\boldsymbol{A}$ and $\boldsymbol{X}$. Hence, many NMF algorithms proceed by (approximately) minimizing the problem alternatively on both matrices [7], including the popular hierarchical alternating least squares method which we describe next.

1.2. Hierarchical Alternating Least Squares

The hierarchical alternating least squares method [8] (HALS) is frequently used to obtain state-of-the art results for NMF [9, 10], and corresponds to Algorithm 1 below.

Algorithm 1 HALS

Require: data $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$, rank $r \ll m, n$, initial $\boldsymbol{A} \in \mathbb{R}^{m\times r}$ and initial $\boldsymbol{X} \in \mathbb{R}^{r\times n}$
  while Stop Condition not encountered **do**
    $\boldsymbol{A} \leftarrow$ `updateHALS`($\boldsymbol{Y}$,$\boldsymbol{A}$,$\boldsymbol{X}$)
    $\boldsymbol{X}^\top \leftarrow$ `updateHALS` ($\boldsymbol{Y}^\top$,$\boldsymbol{X}^\top$, $\boldsymbol{A}^\top$)
  end while

  function UPDATEHALS($\boldsymbol{Y}$, $\boldsymbol{A}$, $\boldsymbol{X}$)
    $\boldsymbol{P} = \boldsymbol{Y}\boldsymbol{X}^\top$, $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{X}^\top$ ▷ $(2n-1)r(m+r)$ flops
    for $\boldsymbol{a_{:j}}$ in $\boldsymbol{A}$ **do**
      $\boldsymbol{t} = (\boldsymbol{p_{:j}} - \sum_{k\neq j} \boldsymbol{a_{:k}} q_{kj})/q_{jj}$ ▷ unconstrained update
      $\boldsymbol{a_{:j}} \leftarrow \max(0, \boldsymbol{t})$ ▷ projection
    end for ▷ $2mr^2$ flops in loop
    return $\boldsymbol{A}$
  end function ▷ $\mathcal{O}(nrm)$

HALS updates the matrix $\boldsymbol{A}$ through its columns $\boldsymbol{a_{:j}}$.

☆This work was supported by the Fonds de la Recherche Scientifique - FNRS and the Fonds Wetenschappelijk Onderzoek - Vlaanderen under EOS Project no 30468160.

*Corresponding author

Email addresses: `cecile.hautecoeur@uclouvain.be` (Cécile Hautecoeur), `francois.glineur@uclouvain.be` (François Glineur)

Columns are successively updated by optimizing the problem considering all the other variables as fixed. The optimal update for a column is obtained by computing the optimal solution of the unconstrained problem and projecting it over the set of nonnegative vectors [11]. Note that each component in the solution of the unconstrained problem is obtained in closed-form as the minimum of a univariate convex quadratic function. Performing an update of all columns leads to a better approximation of the optimal matrix $\boldsymbol{A}$ for $\boldsymbol{X}$ fixed.

1.3. NMF over continuous signals

NMF is often used to analyze continuous signals, such as spectral data [12, 13, 14]. Many of these signals can be well approximated using parametrizable functions like polynomials or splines. Hence, in this work, we tackle the situation where input data consists of continuous signals (possibly discretized), and seek to obtain an NMF-like factorization where the columns of matrix $\boldsymbol{A}$ will be replaced by continuous functions. Indeed, decomposing input data as linear combinations of functions instead of vectors will allow considering the input signals continuously, and not only at some discretization points, and a suitable choice of parametrizable functions will enforce some intrinsic features on the recovered data, such as smoothness in the case of polynomials or splines.

Section 2 introduces the functional NMF problem, and proposes a generalization of HALS to compute factorizations over parametrizable functions. Section 3 focuses on a crucial component of the new HALS algorithm, namely the projection operator over the set of nonnegative parametrizable functions. Section 4 describes existing approaches for the functional NMF problem. These are then compared to our approach in Section 5, where several numerical experiments, using both synthetic and real-world signals, allow us to assess the accuracy and speed of all algorithms.

2. HALS using parametrizable functions (F-HALS)

2.1. NMF over parametrizable functions (F-NMF)

Consider a set $\mathcal{Y} = \{y_1(t), \cdots, y_n(t)\}$ of n univariate functions defined over a common fixed interval $[a, b]$. The functional NMF problem (F-NMF) aims at recovering a dictionary $\mathcal{A} = \{a_1(t), \cdots a_r(t)\}$ of r nonnegative functions over interval $[a, b]$, and a nonnegative mixing matrix $\boldsymbol{X} \in \mathbb{R}_+^{r\times n}$, which can provide an approximate linear description of the original data as follows:

$$y_i(t) \simeq \sum_{j=1}^{r} a_j(t)x_{ji} \qquad \forall t \in [a, b],$$
$$\text{such that } a_j(t) \geq 0,\ x_{ji} \geq 0 \qquad 1 \leq j \leq r,\ 1 \leq i \leq n.$$

Factorization rank $r \ll n$ is provided, as well as a set $\mathcal{F}$ of parametrizable functions containing the dictionary ($\mathcal{A} \subset \mathcal{F}$). In this work, we consider functions that are linearly parametrizable through a finite number of parameters, i.e. that can be described as a linear combination of d fixed basis functions $\{\pi_1(t), \pi_2(t), \cdots \pi_d(t)\}$:

$$a_j(t) = \sum_{k=1}^{d} \pi_k(t)b_{kj} \quad \forall 1 \leq j \leq r.$$

Coefficients b_{kj} expressing function $a_j(t) \in \mathcal{A}$ in that basis are stored in a coefficient matrix $\boldsymbol{B} \in \mathbb{R}^{d\times r}$ (each of its columns $\boldsymbol{b_{:j}} \in \mathbb{R}^d$ describing a function $a_j(t) \in \mathcal{A}$). We also introduce the set $\mathcal{F}_+([a, b]) \subset \mathbb{R}^d$, which is the set of coefficients describing functions in $\mathcal{F}$ that are nonnegative over interval $[a, b]$, i.e.

$$\boldsymbol{b_{:j}} \in \mathcal{F}_+([a, b]) \Leftrightarrow a_j(t) = \sum_{k=1}^{d} \pi_k(t)b_{kj} \geq 0 \text{ for all } t \in [a, b].$$

To solve the (F-NMF) problem, we need to introduce some assumptions on the input functions in $\mathcal{Y}$. We consider two cases:

1. Functions are square-integrable over interval $[a, b]$, and the integral of their product with any basis function is computable, i.e. we know $\int_a^b \pi_j(t)y_i(t)\ \mathrm{d}t$.
2. Functions are known for m fixed discretization points, denoted $\{\tau_l\}_{l=1}^m \subset [a, b]$ and are represented using column vectors $\{\boldsymbol{y_{:i}}\}_{i=1}^n \subset \mathbb{R}^m$ with $y_{li} = y_i(\tau_l)$.

This leads to the following two finite-dimensional formulations of (F-NMF), where $\boldsymbol{x_{:i}}$ denotes the i^{th} column of matrix $\boldsymbol{X}$ and basis functions are grouped in row vector $\boldsymbol{\pi(t)} = (\pi_1(t)\ \cdots\ \pi_d(t))$:

$$\min_{\boldsymbol{B},\boldsymbol{X}} \sum_{i=1}^{n} \int_a^b \underbrace{\left(y_i(t) - \boldsymbol{\pi(t)}\boldsymbol{B}\boldsymbol{x_{:i}}\right)^2}_{\text{integrable functions}} \mathrm{d}t \quad \text{(integral cost)}$$
$$\text{or} \quad \min_{\boldsymbol{B},\boldsymbol{X}} \sum_{i=1}^{n}\sum_{l=1}^{m} \underbrace{\left(y_{li} - \boldsymbol{\pi(\tau_l)}\boldsymbol{B}\boldsymbol{x_{:i}}\right)^2}_{\text{only observations}} \quad \text{(sum cost)}$$
$$\text{s.t.} \quad \boldsymbol{b_{:j}} \in \mathcal{F}_+([a, b]),\ x_{ji} \geq 0 \qquad \forall\ i, j \qquad (1)$$

These formulations differ in the way they assess the accuracy of the reconstructed functions: in the first case, named *integral cost*, the difference between an input signal $y_i(t)$ and its approximation is measured using the (squared) functional L_2 norm, while the second *sum cost* formulation only sums the (squared) differences at the $\{\tau_l\}_{l=1}^m$ observation abscissas (discretization points).

Interestingly, these two costs can be analyzed at once using the following equivalent formulation, with matrices $\boldsymbol{Z}$ and $\boldsymbol{M}$ defined in Table 1 below:

$$\min_{\boldsymbol{B},\boldsymbol{X}} \sum_{i=1}^{n} -2\boldsymbol{z_{:i}}^\top\boldsymbol{B}\boldsymbol{x_{:i}} + \boldsymbol{x_{:i}}^\top\boldsymbol{B}^\top\boldsymbol{M}\boldsymbol{B}\boldsymbol{x_{:i}} \quad \text{s.t. (1) (F-NMF)}$$

Note that the sum case can be seen as an approximation of the integral one, as one can approximate integral $\int_a^b f(t)g(t)$ with the Riemann sum $\frac{b-a}{m}\sum_{\tau_l} f(\tau_l)g(\tau_l)$ using only values at discretization points.

	integral cost	sum cost
$Z \in \mathbb{R}^{d\times n}$	$z_{ji} = \int_a^b \pi_j(t) y_i(t)\ \mathrm{d}t$	$\sum_{\tau_l} \pi_j(\tau_l) y_{li}$
$M \in \mathbb{R}^{d\times d}$	$m_{ji} = \int_a^b \pi_j(t)\pi_i(t)\ \mathrm{d}t$	$\sum_{\tau_l} \pi_j(\tau_l)\pi_i(\tau_l)$

Table 1: Matrices $\boldsymbol{Z}$ and $\boldsymbol{M}$ for unified formulation of (F-NMF)

2.2. A generalized HALS for F-NMF

We propose to solve the (F-NMF) problem with an adapted version of HALS (Algorithm 1) that will update alternatively the columns of $\boldsymbol{A}$ (through their coefficients contained in matrix $\boldsymbol{B}$) and the rows of mixing matrix $\boldsymbol{X}$. As the considered problem is no longer symmetric, the updates of $\boldsymbol{A}$ and $\boldsymbol{X}$ are now different, and we described them in turn below.

Update of columns in $\boldsymbol{B}$. To update the columns in $\boldsymbol{B}$, one must minimize the cost

$$C(\boldsymbol{B}, \boldsymbol{X}) = \sum_i -2\boldsymbol{z}_{:i}^\top \boldsymbol{B}\boldsymbol{x}_{:i} + \boldsymbol{x}_{:i}^\top \boldsymbol{B}^\top \boldsymbol{M}\boldsymbol{B}\boldsymbol{x}_{:i}$$

with respect to one column $\boldsymbol{b}_{:k}$ at a time, while keeping all the other columns of $\boldsymbol{B}$ and matrix X fixed. The gradient of the cost with respect to $\boldsymbol{b}_{:k}$ is

$$\frac{\partial C}{\partial \boldsymbol{b}_{:k}}(\boldsymbol{B}, \boldsymbol{X}) = 2(-\boldsymbol{Z}\boldsymbol{x}_{k:}^\top + \boldsymbol{M}\textstyle\sum_{j=1}^r \boldsymbol{b}_{:j}\boldsymbol{x}_{j:}\boldsymbol{x}_{k:}^\top)$$

Ignoring the nonnegativity constraint, the solution of this problem can be obtained by canceling the gradient:

$$\frac{\partial C}{\partial \boldsymbol{b}_{:k}}(\boldsymbol{B}, \boldsymbol{X}) = 0 \Rightarrow \boldsymbol{M}\boldsymbol{b}_{:k} = \frac{\boldsymbol{Z}\boldsymbol{x}_{k:}^\top - \boldsymbol{M}\sum_{j\neq k} \boldsymbol{b}_{:j}\boldsymbol{x}_{j:}\boldsymbol{x}_{k:}^\top}{\boldsymbol{x}_{k:}\boldsymbol{x}_{k:}^\top}$$

To take the nonnegativity constraint $\boldsymbol{b}_{:j} \in \mathcal{F}_+([a,b])$ into account, one simply projects the unconstrained solution on set $\mathcal{F}_+([a,b])$ using a suitable metric (this is due to the fact that cost C is quadratic in $\boldsymbol{b}_{:k}$), and obtain the following update for the columns in $\boldsymbol{B}$: (we used that matrix $\boldsymbol{M}$ is invertible)

$$\boldsymbol{b}_{:k} \leftarrow \left[\frac{(\boldsymbol{M})^{-1}\boldsymbol{Z}\boldsymbol{x}_{k:}^\top - \sum_{j\neq k} \boldsymbol{b}_{:j}\boldsymbol{x}_{j:}\boldsymbol{x}_{k:}^\top}{\boldsymbol{x}_{k:}\boldsymbol{x}_{k:}^\top}\right]_{\mathcal{F}_+([a,b])}$$

The projection operator onto $\mathcal{F}_+([a,b])$, denoted as $[\cdot]_{\mathcal{F}_+([a,b])}$, is not as straightforward to compute as in the case of standard HALS (for which projection over nonnegative vectors is a simple thresholding operation, see Algorithm 1) and, in particular, nonnegativity of the coefficients is in general not equivalent to nonnegativity of the function. Nevertheless, computing this projection is possible, and is presented in Section 3 in the case of polynomials and splines.

Update of rows in $\boldsymbol{X}$. To update rows in $\boldsymbol{X}$, we compute the gradient of the cost with respect to $\boldsymbol{x}_{k:}$

$$\frac{\partial C}{\partial \boldsymbol{x}_{k:}}(\boldsymbol{B}, \boldsymbol{X}) = 2\boldsymbol{b}_{:k}^\top(-\boldsymbol{Z} + \boldsymbol{M}\textstyle\sum_{j=1}^r \boldsymbol{b}_{:j}\boldsymbol{x}_{j:})$$

Cancellation of the gradient followed by projection onto the feasible set ($\boldsymbol{x}_{k:} \geq 0$) provides the update:

$$\boldsymbol{x}_{k:} \leftarrow \left[\frac{\boldsymbol{b}_{:k}^\top\boldsymbol{Z} - \sum_{j\neq k} \boldsymbol{b}_{:k}^\top\boldsymbol{M}\boldsymbol{b}_{:j}\boldsymbol{x}_{j:}}{\boldsymbol{b}_{:k}^\top\boldsymbol{M}\boldsymbol{b}_{:k}}\right]_+$$

(where $[\cdot]_+$ the straightforward projection over nonnegative reals: $[\xi]_+ = \max\{\xi, 0\}$).

The pseudo-code for F-HALS, our adapted version of HALS, can be found in Algorithm 2, including the number of floating point operations for each statement. Besides the two updates described above, we normalize (or scale) all rows of $\boldsymbol{X}$ after each full update according to $\boldsymbol{x}_{k:} \leftarrow \frac{\boldsymbol{x}_{k:}}{(\boldsymbol{x}_{k:}\boldsymbol{x}_{k:}^\top)^{1/2}}$ (scaling the columns of the $\boldsymbol{B}$ matrix accordingly), and similarly normalize the columns of $\boldsymbol{B}$ after each full update with $\boldsymbol{b}_{:k} \leftarrow \frac{\boldsymbol{b}_{:k}}{(\boldsymbol{b}_{:k}^\top\boldsymbol{M}\boldsymbol{b}_{:k})^{1/2}}$ (while scaling the rows of the $\boldsymbol{X}$ matrix accordingly).

Algorithm 2 F-HALS

Require: matrices $\boldsymbol{Z}, \boldsymbol{M}$, rank r, initial $\boldsymbol{B} \in \mathbb{R}^{d\times r}$ and initial $\boldsymbol{X} \in \mathbb{R}^{r\times n}$
  $\boldsymbol{M_1} = \boldsymbol{M}^{-1}\boldsymbol{Z}$
  while Stop Condition not encountered **do**
    $\boldsymbol{B} \leftarrow$ `updateB`($\boldsymbol{M_1}$, $\boldsymbol{M}$, $\boldsymbol{B}$, $\boldsymbol{X}$)
    $\boldsymbol{X} \leftarrow$ `updateX` ($\boldsymbol{Z}$, $\boldsymbol{M}$, $\boldsymbol{B}$, $\boldsymbol{X}$)
  end while

  function UPDATEB($\boldsymbol{M_1}$, $\boldsymbol{M}$, $\boldsymbol{B}$, $\boldsymbol{X}$)
    $\boldsymbol{P} = \boldsymbol{M_1}\boldsymbol{X}^\top$, $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{X}^\top$ ▷ $(2n-1)r(d+r)$ flops
    for $\boldsymbol{b}_{:k}$ in $\boldsymbol{B}$ **do**
      $\boldsymbol{t} = \boldsymbol{p}_{:k} - \sum_{j\neq k} \boldsymbol{b}_{:j}q_{jk}$ ▷ $2d(r-1)$ flops
      $\boldsymbol{b}_{:k} \leftarrow$ `Projection`$(\boldsymbol{t}/q_{kk})$ ▷ P flops
    end for
    for $\boldsymbol{b}_{:k}$ in $\boldsymbol{B}$ **do** ▷ Normalization
      $nb \leftarrow (\boldsymbol{b}_{:k}^\top\boldsymbol{M}\boldsymbol{b}_{:k})^{1/2}$ ▷ $2d^2 + 2d - 1$ flops
      $\boldsymbol{b}_{:k} \leftarrow \boldsymbol{b}_{:k}/nb$, $\boldsymbol{x}_{k:} \leftarrow \boldsymbol{x}_{k:} * nb$ ▷ $d + n$ flops
    end for
    return $\boldsymbol{B}$ ▷ $\mathcal{O}(rP + nrd)$
  end function

  function UPDATEX($\boldsymbol{Z}$, $\boldsymbol{M}$, $\boldsymbol{B}$, $\boldsymbol{X}$)
    $\boldsymbol{P} = \boldsymbol{B}^\top\boldsymbol{Z}$, $\boldsymbol{Q} = \boldsymbol{B}^\top\boldsymbol{M}\boldsymbol{B}$ ▷ $(2d-1)r(n+d+r)$ flops
    repeat $\min(1 + \rho_{\boldsymbol{X}}/2, 10)$ times
      for $\boldsymbol{x}_{k:}$ in $\boldsymbol{X}$ **do**
        $\boldsymbol{t} = \boldsymbol{p}_{k:} - \sum_{j\neq k} q_{kj}\boldsymbol{x}_{j:}$ ▷ $2n(r-1)$ flops
        $\boldsymbol{x}_{k:} \leftarrow \max(0, \boldsymbol{t}/q_{kk})$ ▷ $2n$ flops
      end for
    until no more progress
    for $\boldsymbol{x}_{k:}$ in $\boldsymbol{X}$ **do** ▷ Normalization
      $nx \leftarrow (\boldsymbol{x}_{k:}\boldsymbol{x}_{k:}^\top)^{1/2}$ ▷ $3n - 1$ flops
      $\boldsymbol{x}_{k:} \leftarrow \boldsymbol{x}_{k:}/nx$, $\boldsymbol{b}_{:k} \leftarrow \boldsymbol{b}_{:k} * nx$ ▷ $n + d$ flops
    end for
    return $\boldsymbol{X}$
  end function ▷ $\mathcal{O}(nrd)$

We also use the acceleration technique introduced in [15], which consists in performing the first for loop of the

updates several times. Note that the matrices $\boldsymbol{Z}$, $\boldsymbol{M}$ and $\boldsymbol{M_1} = \boldsymbol{M^{-1}Z}$ can be pre-computed. Looking at the update of $\boldsymbol{B}$, the projection into the set of nonnegative functions can be very costly, so that no gain is likely to be achieved when repeating the corresponding for loop. Considering now the update of $\boldsymbol{X}$, we see that the ratio between the number of flops needed for one complete iteration and the number of flops for iterations doing only the first for loop is equal to:

$$\rho_{\boldsymbol{X}} = 1 + \frac{(2d-1)(d+r+n) + 4n + d - 1}{2nr}$$

Using the results obtained for HALS updates in [15], update of matrix $\boldsymbol{X}$ is performed $\max\{1 + \frac{\rho_{\boldsymbol{X}}}{2}, 10\}$ times before alternating (updates are also stopped when the improvement is no longer significant, see [15]).

As this acceleration scheme does not modify the order of complexity of the algorithm, we can say that updating both matrices $\boldsymbol{B}$ and $\boldsymbol{X}$ can be done with a total complexity of $\mathcal{O}(rP + nrd)$ where P the complexity of the projection (this estimate is based on $r < d < n$).

3. Projection onto the set of nonnegative functions

To perform the $\boldsymbol{B}$ update in F-HALS we need to project its columns onto the set $\mathcal{F}_+([a,b])$, so that the functions used in the factorization remain nonnegative. This projection, which is performed on a vector of coefficients $\boldsymbol{f} \in \mathbb{R}^d$, can be obtained as the solution of the following minimization problem: $[\boldsymbol{f}]_{\mathcal{F}_+([a,b])}$ is the minimizer of

$$\begin{aligned} &\min_{\boldsymbol{g}} \|\boldsymbol{f} - \boldsymbol{g}\|_{\boldsymbol{M}}^2 && \text{s.t. } \boldsymbol{g} \in \mathcal{F}_+([a,b]) \qquad (2)\\ &\min_{\boldsymbol{g}} (\boldsymbol{f} - \boldsymbol{g})^\top \boldsymbol{M} (\boldsymbol{f} - \boldsymbol{g}) && \text{s.t. } \boldsymbol{g} \in \mathcal{F}_+([a,b])\\ \Leftrightarrow &\min_{\boldsymbol{g}} ||\boldsymbol{L}(\boldsymbol{f} - \boldsymbol{g})||_2^2 && \text{s.t. } \boldsymbol{g} \in \mathcal{F}_+([a,b]). \end{aligned}$$

Note that matrix $\boldsymbol{M}$ (defined in Table 1) appearing in the objective defines the metric used for the projection (it actually comes from the Hessian of the cost function with respect to a column of $\boldsymbol{B}$). It is symmetric and positive definite, so that it can be expressed as $\boldsymbol{M} = \boldsymbol{L}^\top \boldsymbol{L}$ with $L \in \mathbb{R}^{d \times d}$.

The objective function is easy to handle, since it is a convex quadratic in $\boldsymbol{g}$, and the main difficulty is the constraint $\boldsymbol{g} \in \mathcal{F}_+([a,b])$. We now explain how this minimization problem can be solved when the set $\mathcal{F}$ contains polynomials or splines.

3.1. Projection onto nonnegative polynomials

When $\mathcal{F}$ is the set of univariate polynomials with some fixed maximum degree, one can use any basis of polynomials such as the monomial basis or the Chebyshev basis. These basis allow to write a degree d polynomial as $f(t) = \boldsymbol{\pi(t)}\boldsymbol{f}$ where $\boldsymbol{f}$ is a vector in $\mathbb{R}^{d+1}$. In this work, we use Chebyshev basis as in [16] in order to attempt avoiding issues with ill-conditioning.

To express that a univariate polynomial is nonnegative we use the sum-of-squares technique (SOS). Indeed, it is known (see e.g. [17]) that a univariate polynomial $a(t)$ is nonnegative if and only if it is SOS, meaning that it can be written as $a(t) = \sum_i (h_i(t))^2$ for some polynomials $h_i(t)$ (actually, two such polynomials are sufficient).

To express nonnegativity on the considered interval $[a,b]$ we use the Markov-Lukács theorem. It states in the case of an even degree d that nonnegativity over interval $[a,b] = [-1,1]$ is equivalent to

$$\boldsymbol{g} \in \mathcal{F}_+([-1,1]) \;\Leftrightarrow g(t) = a(t) + (1-t^2)b(t)$$

where $a(t)$ and $b(t)$ are two nonnegative polynomials (with respective degrees d and $d-2$). The case of odd degree polynomials is given by (4) in the next section, and an appropriate change of variable allows to adapt the condition for any interval $[a,b]$.

Moreover, sum of squares polynomials can be expressed using positive semidefinite matrices [18], which can be optimized very efficiently as semidefinite programs using interior-point algorithms (see e.g. [19]). Indeed, a degree d polynomial $a(t)$ is SOS if and only if

$$a(t) = \textstyle\sum_i (\boldsymbol{\pi(t)}\boldsymbol{h_i})^2 = \boldsymbol{\pi(t)} \sum_i \boldsymbol{h_i}\boldsymbol{h_i}^\top \boldsymbol{\pi}^\top\boldsymbol{(t)} = \boldsymbol{\pi(t)}\boldsymbol{Q}\boldsymbol{\pi}^\top\boldsymbol{(t)}$$

where $\boldsymbol{Q}$ is a positive semidefinite matrix in $\mathbb{R}^{(\frac{d}{2}+1)\times(\frac{d}{2}+1)}$ (since it is a sum of positive semidefinite rank-one terms $\boldsymbol{h_i}\boldsymbol{h_i}^\top$), called the Gram matrix associated to $a(t)$. Coefficients $\boldsymbol{a}$ can be easily recovered from matrix $\boldsymbol{Q}$, in way that depends on the chosen basis. For example, in the monomial basis, we have $a_k = \sum_{i+j=k} q_{ij}$ and in the Chebyshev basis $a_k = \sum_{i+j=k} \frac{q_{ij}}{2} + \sum_{|i-j|=k} \frac{q_{ij}}{2}$. Therefore, using an appropriate matrix $\boldsymbol{G}$, we have $\boldsymbol{a} = \boldsymbol{G}\ \mathrm{vec}(\boldsymbol{Q})$.

Using this equivalence it is possible to obtain the projection as the solution of the following semidefinite program:

$$\begin{aligned} &\min t\\ &\text{s.t. } (\boldsymbol{u}, t) \in \mathbb{L}^{d+1} && \text{Lorentz cone, } ||\boldsymbol{u}|| \le t\\ &\boldsymbol{A} \in \mathbb{S}^{d/2+1}, \boldsymbol{B} \in \mathbb{S}^{d/2} && \text{Semidefinite cones}\\ &\boldsymbol{u} = \boldsymbol{L}\left(\boldsymbol{f} - \begin{bmatrix}\boldsymbol{G} & \boldsymbol{G'}\end{bmatrix}\begin{bmatrix}\mathrm{vec}(\boldsymbol{A})\\ \mathrm{vec}(\boldsymbol{B})\end{bmatrix}\right) && \boldsymbol{u} = \boldsymbol{L}(\boldsymbol{f} - \boldsymbol{g}) \end{aligned}$$

$\boldsymbol{G'}$ is such that $\boldsymbol{G'}\ \mathrm{vec}(\boldsymbol{B})$ is the vector of coefficients of $(1-t^2)b(t)$. Figure 1 illustrates this result of such a projection.

3.2. Projection onton nonnegative splines

As splines are piecewise polynomials, it is expected that an approach similar to the one presented above allows to project splines onto the nonnegative set. We consider in this work splines of degree 3, represented using the

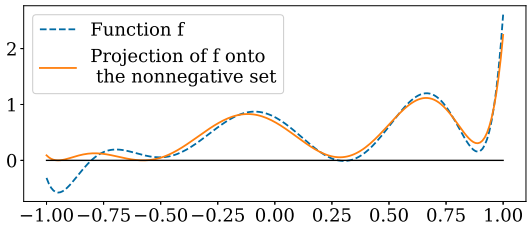

Figure 1: Projection of a degree 8 polynomial onto the set of nonnegative polynomials.

B-Spline basis, which can be computed using the Cox-de Boor recursion formula:

$$B_{i,0}(t) = \begin{cases} 1 & \text{if } t \in [t_i, t_{i+1}] \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

$$B_{i,k}(x) = \frac{t - t_i}{t_{i+k} - t_i} B_{i,k-1}(t) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i-1,k-1}(t)$$

The basis consists of all the $B_{i,3}(t)$ polynomials. The set of $\{t_i\}$ abscissas are the interior knots of the splines. To consider a closed interval, the first and the last knot are repeated 4 times. Each basis element is nonzero over 4 intervals (including length zero intervals) and is a cubic polynomial over these intervals.

Traditionally, splines are analyzed over interval $[0, 1]$. A cubic polynomial is nonnegative on this interval if and only if it can be expressed as (Markov-Lukacs):

$$f(t) = f_1(t)t + f_2(t)(1 - t) \tag{4}$$

with f_1, f_2 two quadratic nonnegative polynomials. Moreover, a quadratic polynomial $f_1(t) = a_1x^2 + b_1x + c_1$ is nonnegative if and only if $a_1, c_1 \geq 0$ and $b_1^2 \leq 4a_1c_1$. Using a (convex) rotated quadratic cone

$$\mathcal{Q}_r = \{(x_1, x_2, x_3) \text{ st. } 2x_1x_2 \geq x_3^2, x_1, x_2 \geq 0\}$$

this nonnegativity condition on $f(t) = at^2 + bt + c$ becomes exactly $(a, c, b/\sqrt{2}) \in \mathcal{Q}_r$. Hence a cubic polynomial is nonnegative over $[0, 1]$ if and only if it can be written as

$$f(t) = (a_1 - a_2)t^3 + (b_1 - b_2 + a_2)t^2 + (c_1 + b_2 - c_2)t + c_2$$

with $\boldsymbol{q_1} = (a_1, c_1, b_1/\sqrt{2}) \in \mathcal{Q}_r$ and $\boldsymbol{q_2} = (a_2, c_2, b_2/\sqrt{2}) \in \mathcal{Q}_r$. Hence we can write the vector of coefficients of f as a linear transformation of $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$, i.e. $\boldsymbol{f} = \boldsymbol{Q}(\boldsymbol{q_1}, \boldsymbol{q_2})^\top$.

For every non-empty interval $[t_i, t_i + 1]$, coefficients of the corresponding cubic polynomial can be expressed as a linear mapping of the coefficients of the four B-Splines that are nonzero over this interval, $\boldsymbol{g}_i$. This linear map $\boldsymbol{N_i}$ can be computed from the definition in (3) and is invertible. Therefore, B-Splines coefficients can be expressed with a linear map applied to elements of a rotated quadratic cone:

$$\boldsymbol{g_i} = \boldsymbol{N_i}^{-1}\boldsymbol{f} = \boldsymbol{N_i}^{-1}\boldsymbol{Q}(\boldsymbol{q_1}, \boldsymbol{q_2})^\top \quad \text{with } \boldsymbol{q_1}, \boldsymbol{q_2} \in \mathcal{Q}_r. \tag{5}$$

Imposing constraint (5) over all intervals ensures the nonnegativity of the spline created by the coefficients $\boldsymbol{g}$. If k interior knots are used, $k + 2$ basis splines are necessary over the $k - 1$ nonzero intervals and the problem is:

$$\begin{aligned} & \min t \\ \text{s.t} \quad & (\boldsymbol{u}, t) \in \mathbb{L}^{k+2} \\ & \boldsymbol{q_{1i}} \in \mathcal{Q}_r, \boldsymbol{q_{2i}} \in \mathcal{Q}_r && \forall i = 0, \cdots k-2 \\ & \boldsymbol{g_i} = \boldsymbol{N_i}^{-1}\boldsymbol{Q}(\boldsymbol{q_{1i}}, \boldsymbol{q_{2i}})^\top && \forall i = 0, \cdots k-2 \\ & \boldsymbol{u} = \boldsymbol{L}(\boldsymbol{g} - \boldsymbol{f}) \end{aligned}$$

Note that each $\boldsymbol{g_i}$ is a sub-vector of $\boldsymbol{g}$ ($\boldsymbol{g_i} = \boldsymbol{g}[i : i + 4]$) and that each coefficient in $\boldsymbol{g}$ appears in several $\boldsymbol{g_i}$.

An example of projection using this method is presented in Figure 2, together with another approach that imposes the B-spline coefficients to be nonnegative, but not the spline itself (as proposed in [20] and [21], see also the next Section). Splines with nonnegative B-Splines coefficients are always nonnegative, but there exist nonnegative splines that cannot be described in such a way [22]. Therefore, this second approach is less accurate that projection described above.

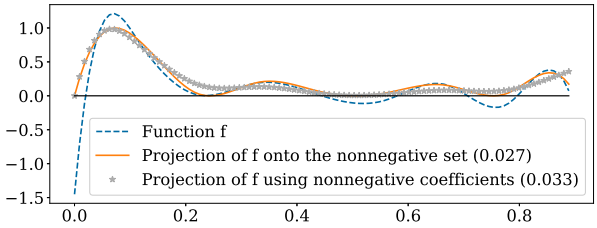

Figure 2: Example of projection onto the set of nonnegative splines, using a spline with 9 equally spaced interior knots. The number in parenthesis is the value of the cost function (i.e. the squared distance).

4. Prior work on functional NMF

Before testing our proposed F-HALS algorithm we summarize existing work on the functional NMF problem. Several authors recently considered the F-NMF problem where input functions $y_i(t)$ are known at some observed discretization points (this data is collected in matrix $\boldsymbol{Y}$), corresponding to what we called the *sum cost*. To tackle the problem, they work with an NMF-like factorization $\boldsymbol{Y} \approx \boldsymbol{AX}$ where columns in matrix $\boldsymbol{A}$ are forced to correspond to the discretization of parametrizable functions (meaning that each element in a given column is obtained by evaluating a parametrizable function on an observation point). Three main approaches have been developed, and all of them optimize a matrix of parameters describing the functions that in turn define the columns of matrix $\boldsymbol{A}$.

Using unconstrained parameters: nonlinear least-squares. Debals et al. propose in [16] a nonlinear parametrization

of the functions defining matrix $\boldsymbol{A}$. They rely on the previously described SOS technique to express nonnegativity, and use the coefficients of the polynomials appearing in those sums of squares as parameters. The weight matrix $\boldsymbol{X}$ is also represented with $x_{ij} = z_{ij}^2$. Note that all parameters in this formulation (for both $\boldsymbol{A}$ and $\boldsymbol{X}$) are now free (no constraints needs to be enforced), so that the NMF problem becomes an unconstrained non-linear least-squares problem, featuring a non-convex objective function, to be solved using a standard solver.

If d is the degree of the polynomials in $\boldsymbol{A}$ and m the number of discretization points at which the polynomials are observed, one iteration in the least squares solver can be computed in $\mathcal{O}(dnr)$ flops. As $d < m$, the asymptotic complexity per iteration is lower than for HALS updates. Moreover, the authors observed experimentally that this method needed fewer iterations than HALS to converge, and that the features obtained after convergence are smooth.

However, this method appears to be very slow compared to HALS for problems with less than 10^3 discretization points. Moreover, its extension to other parametrizable functions would involve finding an unconstrained parametrization for them, which may not be straightforward.

Using nonnegative parameters: nonnegative basis functions. Closer to our approach, one can consider that matrix $\boldsymbol{A}$ can be described as $\boldsymbol{A} = \boldsymbol{\Pi B}$, where $\boldsymbol{\Pi}$ contains the discretization over m points of d *nonnegative basis functions* ($\pi_{ij} = \pi_j(\tau_i) \geq 0 \ \forall i, j$). Imposing coefficients in $\boldsymbol{B}$ to be nonnegative is then sufficient to ensure the nonnegativity of $\boldsymbol{A}$ (but may not be necessary).

This approach has been considered when matrix $\boldsymbol{\Pi}$ contains B-Splines by Backenroth [20] and Zdunek et al. [21]. This problem, closer to the original NMF problem, can be solved in an efficient way and provides smooth features. However, constraining $\boldsymbol{B}$ to be nonnegative is a stronger constraint than strictly needed. Indeed, B-Splines combined using some negative coefficients can still be nonnegative, and it has been proven than estimating nonnegative functions using B-Splines with nonnegative coefficients may leads to a poorer reconstruction [22].

Using constrained parameters: arbitrary basis functions. Suppose again that matrix $\boldsymbol{A}$ can be described as $\boldsymbol{A} = \boldsymbol{\Pi B}$, where $\boldsymbol{\Pi}$ is the discretization over m points of d basis functions, which can now be arbitrary (i.e. not necessarily nonnegative). It is then possible to ensure the nonnegativity of $\boldsymbol{A}$ by constraining parameter matrix $\boldsymbol{B}$. If the mixing matrix $\boldsymbol{X}$ is fixed, this constrained NMF problem is convex and can be described as a quadratic problem. A method using active sets is suggested in [23] for Gaussian Radial Basis Functions, while a method using Alternating Direction Method of Multipliers (ADMM [24]) is suggested in [25] for B-Splines. The active-set method leads to a relatively large computational complexity for large-scale problems [23], while the ADMM method is more suitable. Using indicator function $\Phi(A) = \sum_{i,j} \phi(a_{ij})$, with $\phi(a) = \infty$ if $a < 0$ and 0 otherwise, this method solves the following problem to update $\boldsymbol{A}$:

$$\min_{\boldsymbol{A},\boldsymbol{B}} \frac{1}{2}||\boldsymbol{Y} - \boldsymbol{\Pi B X}||_F^2 + \Phi(\boldsymbol{A}) \quad \text{st.} \quad \boldsymbol{A} = \boldsymbol{\Pi B} \tag{6}$$

via iterations, using projection $[\xi]_+ = \max\{0, \xi\}$, $\tau > 0$ and $\boldsymbol{\Pi}^\dagger = (\boldsymbol{\Pi}^\top \boldsymbol{\Pi})^{-1} \boldsymbol{\Pi}^\top$:

$$\begin{aligned}
\boldsymbol{B}_{k+1} &= \boldsymbol{\Pi}^\dagger \left[\boldsymbol{Y X}^\top + \boldsymbol{\Lambda}_k + \tau \boldsymbol{A}_k\right] \left(\boldsymbol{X X}^\top + \tau \boldsymbol{I}_r\right)^{-1} \\
\boldsymbol{A}_{k+1} &= \left[\boldsymbol{\Pi B}_{k+1} - \tau^{-1} \boldsymbol{\Lambda}_k\right]_+ \\
\boldsymbol{\Lambda}_{k+1} &= \boldsymbol{\Lambda}_k + \tau(\boldsymbol{A}_{k+1} - \boldsymbol{\Pi B}_{k+1}).
\end{aligned}$$

Note that in this approach, nonnegativity of the functions in the dictionary is only ensured at the discretization points considered in $\boldsymbol{A}$. Also, the presented scheme updates matrix $\boldsymbol{A}$, and not the matrix of coefficients $\boldsymbol{B}$. Coefficients are thus not directly accessible. Nevertheless, when the algorithm converges it leads to $\boldsymbol{A} = \boldsymbol{\Pi B}$.

The F-HALS algorithm proposed in Section 2 also corresponds to this third approach, using $\boldsymbol{A} = \boldsymbol{\Pi B}$, where $\boldsymbol{\Pi}$ is arbitrary while the coefficients $\boldsymbol{B}$ are constrained to ensure nonnegativity. However it differs from previous work in several ways: first, we do not explicitly compute matrix $\boldsymbol{A}$, and the coefficients $\boldsymbol{B}$ are used instead. Note that in principle this allows working with an infinite-dimensional matrix $\boldsymbol{A}$ (i.e. infinitely many observation points), which is in essence what we do when using the *integral cost* formulation. Then, we ensure the nonnegativity of the functions in $\boldsymbol{A}$ over the entire considered interval, and not only at discretization points, using the same parametrization as in [16]. This is ensured using projections (see Section 3, for both polynomials and splines), which is the main new ingredient in our approach.

5. Experimental results

In this section, we assess the performance of the F-HALS algorithm (Algorithm 2) through experiments both over synthetic and real signals. Our method is compared with standard HALS and some of the methods described in the previous section.

Unless stated otherwise, the input signals we use are created as $\boldsymbol{Y} = \boldsymbol{AX} + \boldsymbol{N}$ where $\boldsymbol{N} \in \mathbb{R}^{m \times n}$ is an additive Gaussian noise with a chosen signal-to-noise ratio (SNR). Matrix $\boldsymbol{X} \in \mathbb{R}^{r \times n}$ is randomly generated using a normal distribution $\mathcal{N}(0, 1)$ with negative values replaced by zero. Matrix $\boldsymbol{A}$ contains the discretization over m points of r functions, which can be either random nonnegative polynomials, random nonnegative splines or real reflectance signals (coming from the U.S. Geological Survey (USGS) database [26][1]). Discretization points are equally spaced

[1] https://www.usgs.gov/labs/spec-lab/capabilities/spectral-library

over $[-1, 1]$, including for the reflectance signals as they are closer to polynomials and splines in this configuration (it improves the best possible recovery).

Using such an artificially created synthetic dataset lets us assess the error compared to the ground truth for input signals $\boldsymbol{Y}^* = \boldsymbol{AX}$. If $\bar{\boldsymbol{B}}$ and $\bar{\boldsymbol{X}}$ are the matrices recovered by the algorithm (with $\bar{\boldsymbol{A}} = \boldsymbol{\Pi}\bar{\boldsymbol{B}}$), we compute the (relative) residue error as res $= ||\boldsymbol{Y}^* - \bar{\boldsymbol{A}}\bar{\boldsymbol{X}}||/||\boldsymbol{Y}^*||$. We can also evaluate the quality of the recovered signals ($\bar{\boldsymbol{A}}$) with respect to the original signals ($\boldsymbol{A}$). To do so, it is important to remark that a factorization is defined up to permutations and scaling of the columns of $\bar{\boldsymbol{A}}$ and rows of $\bar{\boldsymbol{X}}$. Therefore we must first find the optimal permutation and scaling before evaluation the quality of recovery. Once the best $\bar{\boldsymbol{A}}$ is found, each of its column $\bar{\boldsymbol{a}}_{\boldsymbol{i}}$ is compared to the columns $\boldsymbol{a}_i$ of $\boldsymbol{A}$, using the signal-to-interference ratio (SIR) proposed in [1], that is $\text{SIR}(\bar{\boldsymbol{a}}, \boldsymbol{a}) = 10 \log\left(\frac{||\gamma\bar{\boldsymbol{a}}||_2^2}{||\boldsymbol{a}-\gamma\bar{\boldsymbol{a}}||_2^2}\right)$, where γ minimize $||\boldsymbol{a}-\gamma\bar{\boldsymbol{a}}||_2^2$. The same is done for the rows of $\bar{\boldsymbol{X}}$.

In our experiments, we use a Chebyshev basis to represent polynomials and B-Splines to represent splines. Our algorithms are compared to several other methods constraining the columns of matrix $\boldsymbol{A}$ in NMF problem to be the discretization of continuous functions. We implemented these methods based on the cited papers, and list them below:

- **HALS:** standard HALS algorithm applied on NMF.
- **P-LS:** $\boldsymbol{A}$ contains nonnegative polynomials, represented using **unconstrained** parameters. Problem is solved using a nonlinear Least Squares solver [16].
- **PS-HALS and PI-HALS:** our F-HALS algorithm using polynomials, respectively with *sum cost* and with *integral cost*.
- **S-MU:** $\boldsymbol{A}$ contains splines with **nonnegative** B-Splines coefficients. Problem is solved using Multiplicative Updates [21].
- **S-ADMM:** $\boldsymbol{A}$ contains nonnegative splines, represented using **constrained** parameters. Problem is solved using the Alternative Direction Method of Multipliers [25]
- **SS-HALS and SI-HALS:** our F-HALS algorithm using splines, respectively with *sum cost* and with *integral cost*.

Projections in F-HALS are computed using conic programs solved by MOSEK [27]. Algorithms are stopped when cost function $||\boldsymbol{Y}-\bar{\boldsymbol{A}}\bar{\boldsymbol{X}}||$ no longer improves, namely when $|\text{cost} - \text{previous cost}|/\text{cost} < 10^{-7}$.

5.1. Quality of recovered signals

We first compare the signals recovered by our algorithm to the vectors recovered by HALS. Each test uses $n = 50$ observations and $r = 3$ basis elements. We used in F-HALS functions parametrized with 21 coefficients: polynomials of degree 20 and splines with 19 interior knots regularly distributed in $[-1, 1]$. Basis signals in $\boldsymbol{A}$ are either polynomials or splines with 21 coefficients (Figure 3) or real reflectance signals of olivine, spessartine and hypersthene (Figure 4). When $\boldsymbol{A}$ contains polynomials or splines and no noise is added, and the integral case is considered, the inputs of our algorithms are the functions in $\boldsymbol{Y}$. Otherwise the signals are discretized over $m = 100$ points for polynomials or splines and 414 points for the real reflectance signals. Integrals needed to compute $\boldsymbol{Z}$ in the integral case are approximated using a piecewise interpolation of order 1 of the data, while matrix $\boldsymbol{M}$ is still computed with integrals. When noise is added its SNR is equal to 20 dB.

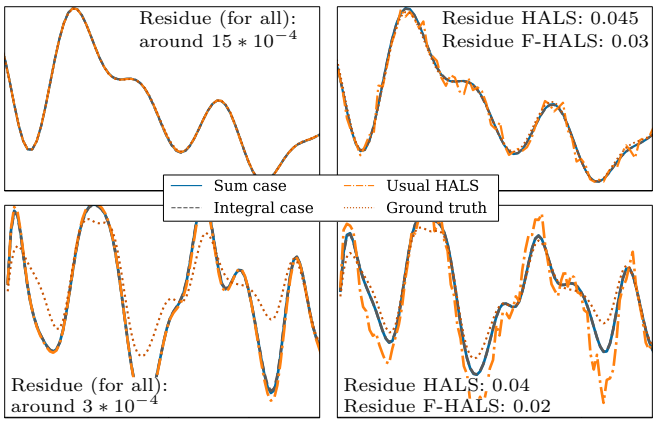

Figure 3: Example of recovered signals. Up: polynomials, Down: splines, Left: noise-free case, Right: noisy case. Our algorithm obtains very similar performances in integral and sum case.

In Figure 3 we tested the performance of our algorithms using the appropriate functional set $\mathcal{F}$ (polynomials if the basis signals are polynomials and splines otherwise). We observe that the signals recovered in the noiseless case are similar for the three tested methods. However, when noise is added to the signals, signals recovered by HALS are less smooth than the signals recovered by our algorithms. It is interesting to observe that the recovery of splines is worse than for polynomials, even if the residues are similar. A deeper inspection of the recovered signals shows that they have an higher representation power than the original signals, that can be recovered as a nonnegative linear combination of the found signals. This phenomenon occurs because the basis defined by the original splines has a non-unique representation. We observed this kind of behavior mostly on low-degree polynomials and on dense splines (with many nonzero coefficients).

If we observe the recovered signals when $\boldsymbol{A}$ contains real reflectance signals in Figure 4, we observe again that the signals recovered by HALS are non-smooth in cases with noise. Its residue is around 0.04 instead of 0.02 for the others. However, HALS is much better in the noiseless case with a residue around $4 * 10^{-4}$ instead of 0.015, as it

is able to describe the "less smooth" parts of the signals, unlike low-degree polynomials or splines. It is interesting to notice that the residues of our methods does not change much between the noise-free and the noisy case while it is very different between these two cases for HALS. This suggests that our methods are less sensitive to noise than HALS.

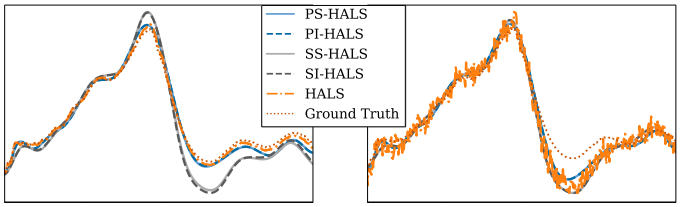

Figure 4: Example of recovered signals of spessartine. Left: noise-free case, Right: noisy case. PS-HALS and PI-HALS obtains very similar performance, like SS-HALS and SI-HALS.

5.2. Comparison between sum and integral cases

Using polynomials of degree 20 and splines with 19 interior knots, we compared the performance of our algorithms using the sum or the integral costs. Figure 5 present the results for $r = 3$ polynomials of degree 12 with $n = 100$ observations and $m = [25, 50, 100, 150, 250, 500]$ equally spaced discretization points in sum case. Residue is computed over 15000 points equally spaced in $[-1, 1]$.

Using polynomial signals, PI-HALS performs a bit better when fewer discretization points are available, and the same is observed for splines. This is expected as the integrals contain the perfect information about the data, unlike the discretization. Increasing the number of discretization points improve of course the performance of the algorithms using the sum cost, but increases also the initialization time.

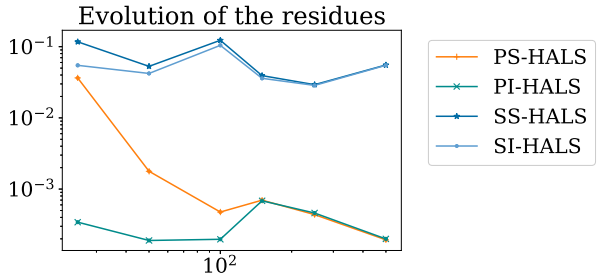

Figure 5: Performance of our algorithms over polynomial signals. No noise is added. Average over 10 problems.

In Figure 6, we observe the performance of our algorithms using the same data as for the previous test except that the basis elements are splines with 19 interior knots and a noise of 20dB is added to the data. In this case, the sum case seems slightly better than the integral one, except when very few discretization points are available. However, during our experimentation we observed that the necessarily approximate computation of the integrals required for the integral case have a large influence on the final result. Hence, the performance of the integral case may be improved with a more accurate evaluation of the integrals. Nevertheless, using the sum cost is already quite robust. Therefore, based on the results obtained in these tests, we recommend using the integral cost when the input functions are known and the sum case if they are provided as vectors.

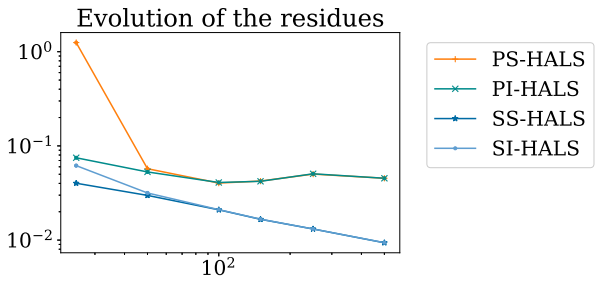

Figure 6: Performance of our algorithms over spline signals with noise of 20 dB. Average over 10 problems.

5.3. Comparison with other approaches

Unless stated otherwise, the following tests are made over polynomials of degree 12 and splines with 11 interior knots, with $n = 100$ observations as well as a noise level of 20 dB. Tests are made over $r = 5$ real reflectance signals when possible, otherwise $r = 3$ polynomial signals of degree 12 are used. Each result is the average over 10 tests.

5.3.1. Performance on large datasets

To study their computational performance we run each algorithm during 20 iterations for an increasing number of observations and discretization points ($n = m$), over polynomial signals. Figure 7 illustrates that one iteration of

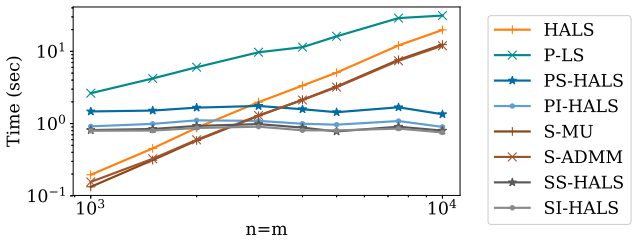

Figure 7: Time for 20 iterations without initialization for increasing $n = m$. S-MU and S-ADMM display very similar performances, as SS-HALS and SI-HALS.

HALS, S-MU and S-ADMM has $\mathcal{O}(mn)$ complexity, while for LS it is only $\mathcal{O}(n)$ [16]. In contrast, time spent in computations increases very moderately with $n = m$ for our methods, because they spend a large fraction of their computational time on the projection step (more than 95% for $n = m = 10^4$), which does not depend on n neither m.

Note that the initialization time is not presented in this graph. For P-LS and our methods, this initialization can

be quite costly, in $\mathcal{O}(mn)$, but must be computed only once. All the presented algorithms are influenced by the initial values of matrices $\boldsymbol{A}$ and $\boldsymbol{X}$, and a way to find the best solution is to run the algorithm with different initial values. In this case, the initialization of our algorithms and P-LS must only be computed once for all runs.

Among our methods, PS-HALS has a higher computational time. A look at the projections in this case shows that matrix $\boldsymbol{M}$ is not as sparse as for the other approaches. The lower sparsity of $\boldsymbol{M}$ slows down the projections (performed by an interior-point method) and thus the algorithm in general.

Increasing only the number of observations n, we observed that the accuracy of HALS is significantly improved while the improvement is less important for the other methods, especially when using polynomials. However, the opposite is observed when increasing the number of discretization points m: better improvement for functional-based methods compared to HALS.

5.3.2. Performance when number of coefficients varies

We analyze the influence of the size of the chosen parametrizable set $\mathcal{F}$, that is the degree for polynomials or the number of interior knots for splines. In Figure 8, we observe that our algorithms and P-LS spend more time in computations when the degree d of polynomials increases, and that this slowdown is more consequent for F-HALS. For P-LS method, increasing d increases the size of the least-square problem to solve, while for our methods it increases the size of the projection problem. Moreover, we observe than using higher-degree polynomials is beneficial only until a certain point. The choice of the degree of the polynomials is thus very important. The same observations can be made for splines if we increase the number of interior knots, even though the increase in time is less pronounced than for polynomials. Time performance of S-MU and S-ADMM is not much influenced by the number of interior knots of the splines.

We observe that S-ADMM obtains an slightly higher residue than our methods, unlike S-MU that obtains similar residues once the number of interior knots is larger than 20. S-MU is also much faster than our methods when we observe 100 signals. When we look at the best recovery of the used reflectance signal using splines, we observe that the spline coefficients are nonnegative. As S-MU uses splines with nonnegative coefficients, it is not surprising that it obtains good results in this configuration. However, when we compared the performances of our methods to S-MU on nonnegative splines without imposing nonnegative coefficients, we observe that S-MU is not always able to recover accurate signals (see for example Table 2). Nevertheless, it is interesting to notice that using splines with nonnegative coefficients can be accurate in some situations. Note that this assumption could also be used in our F-HALS approach, and would accelerate its projection step ; we leave the exploration of this idea for future work.

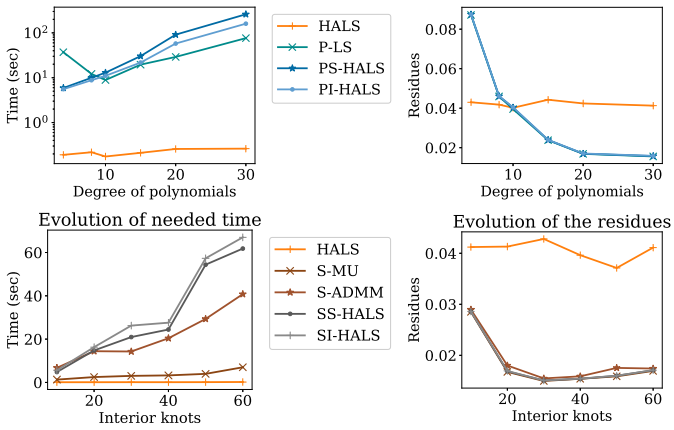

Figure 8: performance for increasing degree (Top) or increasing number of interior knots (Down). Left: needed time, Right: residues. S-MU has similar time performances than HALS and obtain a similar residue than our methods when the number of interior knots is higher than 30.

Method	Time	Its	SIR_A	SIR_X	Res
S-MU	0.26	115.25	13.90	36.70	18
SS-HALS	1.89	31.50	48.27	42.27	1

Table 2: performance of S-MU and SS-HALS over spline signals with 20 dB noise, $r = 3$, $n = m = 500$, number of interior knots = 20. Test over 10 problems and 10 initializations. Time is expressed in seconds and Its stands for the number of iterations. Res are the residuals multiplied by 10^3.

5.3.3. Performance over noisy data

We now pay attention to the performance of the algorithms over various levels of noise on the data, as displayed on Figure 9. We observe that when using polynomials or splines as basis signals solutions appear to be almost insensitive to the noise, unlike the vector-based HALS. The computational effort required by all methods appears to be independent of the level of noise.

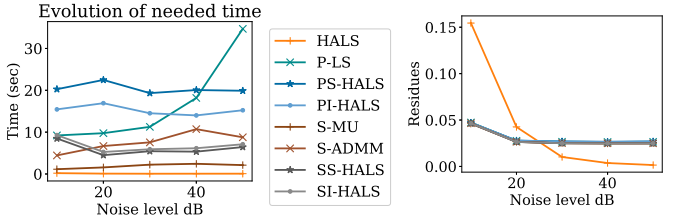

Figure 9: performance for different noise levels. Rightmost value is the noise-free situation (replacing the 50 dB mark). All algorithms obtain similar residues except HALS.

5.3.4. Detailed analysis of tests with reflectance signals

Table 3 contains the average results of the considered algorithms over $n = 250$ observations of reflectance signals using 10 random input datasets, each with 10 different starting values (100 tests in total). We chose the number of observation not too high but not too low either, in order

to have accurate results for polynomials and spline-based methods. Noise level is 20dB on which functional NMF provides good results (see Figure 9). Moreover, based on results from Subsection 5.3.2, we used polynomials of degree 20 and splines with 30 interior knots, with a noise of 20dB. We observe that splines obtain better residues than polynomials, leading to lower residues than HALS. Moreover, methods using splines are generally faster than methods with polynomials.

We see also that S-ADMM obtains a significantly higher residue than the other methods using splines. We could observe during our tests that this behavior occurs for datasets with many observations. For $n = 100$ for example, the results of S-ADMM are comparable to the other methods using splines. Our method using splines in the sum case leads to the best residue.

The method leading to the closest signals on average is P-LS. In general, HALS-based methods seem to obtain matrices $\boldsymbol{A}$ and $\boldsymbol{X}$ with worse SIR than the other methods if we assess them by looking only at permutation and rescaling (PS) of the obtained signals. However, the representation power of the recovered basis is similar to that of the other methods (and sometimes even better) when using the best nonnegative linear combinations (LC) of obtained matrices $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{X}}$ to compute the SIR.

We can observe in Figure 10 the evolution of the residues with respect to elapsed time (including time for initialization). Our stopping criterion appears to be in general well-adapted to the tested methods, as they all seem to have converged. In this example, S-ADMM appears to stop a bit too early while P-LS stops a bit too late, while the other methods stop at the right time. The S-MU and HALS methods are significantly faster than the others here.

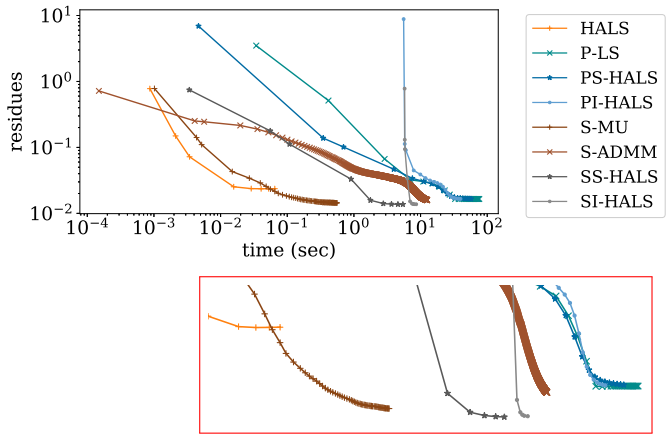

Figure 10: Evolution and zoom on this evolution of the residues for each method with respect to time. A point is plotted every 10 iterations. The y-scale of the zoom is no longer in log-scale to improve readability. The three polynomial methods converge in a similar way, while ADMM is the only spline-based method stopping with a residue similar to the polynomial ones.

6. Discussion and conclusion

Extending the NMF framework to handle polynomial or spline signals leads to functional NMF, which enables the recovery of smoother features and is less sensitive to noise when compared to standard NMF applied to discretized signals.

In this work we adapt the HALS algorithm to this extension. Our new F-HALS algorithm requires the ability to project over sets of nonnegative parametrizable functions, which is computationally feasible for polynomials and splines. Two cost functions can be considered, both competitive with existing approaches. Integral cost is recommended when integrals involving the input signals are computable, while the sum cost can be used if only discretized values are available.

Our algorithms are naturally well-suited to deal with data originating from nonnegative polynomials or nonnegative splines. However, they also lead to good results for (relatively) smooth real-world reflectance signals. The choice of the degree of parametrization is crucial to obtain as accurate results as possible. The degree of parametrization is the degree of the used polynomials or the number of interior knots of the splines. This degree should be not too large when using our methods as it heavily impacts their computational time. In contrast, the computational effort spent by our method does not increase much with the problem size compared to existing approaches, which makes it possible to handle large-scale problems (i.e. with large numbers of observations or discretization points).

For now, our algorithms apply only to polynomials and splines, but it is straightforward to extend them to any linearly parametrizable function provided that one can compute the projection over the corresponding nonnegative set.

References

[1] A. Cichocki, R. Zdunek, A. H. Phan, S.-i. Amari, Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation, John Wiley & Sons, 2009 (2009).

[2] D. D. Lee, H. S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788 (1999).

[3] D. Donoho, V. Stodden, When does non-negative matrix factorization give a correct decomposition into parts?, in: Advances in Neural Information Processing Systems 16, MIT Press, 2004, pp. 1141–1148 (2004).

[4] P. O. Hoyer, Non-negative matrix factorization with sparseness constraints, Journal of machine learning research 5 (Nov) (2004) 1457–1469 (2004).

[5] S. Choi, Algorithms for orthogonal nonnegative matrix factorization, Neural Networks IJCNN (2008) 1828–1832 (2008).

[6] S. A. Vavasis, On the complexity of nonnegative matrix factorization, SIAM Journal on Optimization 20 (3) (2009) 1364–1377 (2009).

[7] H. Kim, H. Park, Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method, SIAM journal on matrix analysis and applications 30 (2) (2008) 713–730 (2008).

Method	Time	Its	$\text{SIR}_{\text{A}}^{PS}$	$\text{SIR}_{\text{X}}^{PS}$	$\text{SIR}_{\text{A}}^{LC}$	$\text{SIR}_{\text{X}}^{LC}$	Res
HALS	0.11	61.33	7.85	3.62	29.23	21.94	24.27
P-LS	101.74	497.00	11.41	4.27	35.27	23.24	16.63
PS-HALS	85.29	230.22	7.20	3.78	34.56	22.60	16.73
PI-HALS	58.05	235.22	7.19	3.72	34.42	22.49	16.81
S-MU	2.05	542.22	9.67	4.02	37.24	22.78	13.93
S-ADMM	46.99	10317.56	12.05	4.44	31.53	18.33	16.85
SS-HALS	17.52	131.00	9.79	4.26	36.85	22.83	13.87
SI-HALS	19.55	131.67	9.79	4.27	36.81	22.81	13.92

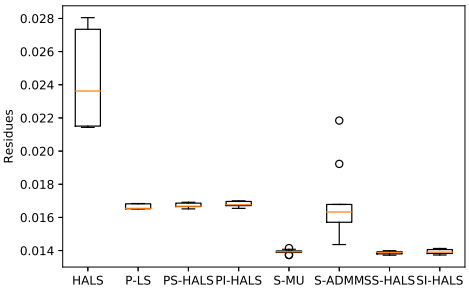

Table 3: Performance of the algorithms over real reflectance signals of Adularia, Clinochlore, Hypersthene, Olivine and Spessartine. Time expressed in seconds, Its stands for the number of iterations. Res are the residues multiplied by 10^3. PS stands for Permutations and Scaling while LC designs nonnegative Linear Combinations. The image of the right contains the boxplot of the residues for each method.

[8] A. Cichocki, R. Zdunek, S.-i. Amari, Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization, in: International Conference on Independent Component Analysis and Signal Separation, Springer, 2007, pp. 169–176 (2007).

[9] J. Feng, X. Huo, L. Song, X. Yang, W. Zhang, Evaluation of different algorithms of nonnegative matrix factorization in temporal psychovisual modulation, IEEE Transactions on Circuits and Systems for Video Technology 24 (4) (2013) 553–565 (2013).

[10] E. Karahan, P. A. Rojas-Lopez, M. L. Bringas-Vega, P. A. Valdés-Hernández, P. A. Valdes-Sosa, Tensor analysis and fusion of multimodal brain images, Proceedings of the IEEE 103 (9) (2015) 1531–1559 (2015).

[11] J. Kim, Y. He, H. Park, Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework, Journal of Global Optimization 58 (2) (2014) 285–319 (2014).

[12] A. M. Darsono, C. C. Toh, S. Saat, A. A. M. Isa, N. A. Manap, M. M. Ibrahim, β-divergence nonnegative matrix factorization on biomedical blind source separation, Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 9 (2) (2017) 1–4 (2017).

[13] H. Kameoka, M. Nakano, K. Ochiai, Y. Imoto, K. Kashino, S. Sagayama, Constrained and regularized variants of nonnegative matrix factorization incorporating music-specific constraints, in: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, IEEE, 2012, pp. 5365–5368 (2012).

[14] P. Sajda, S. Du, T. R. Brown, R. Stoyanova, D. C. Shungu, X. Mao, L. C. Parra, Nonnegative matrix factorization for rapid recovery of constituent spectra in magnetic resonance chemical shift imaging of the brain, IEEE transactions on medical imaging 23 (12) (2004) 1453–1465 (2004).

[15] N. Gillis, F. Glineur, Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization, Neural computation 24 (4) (2012) 1085–1105 (2012).

[16] O. Debals, M. Van Barel, L. De Lathauwer, Nonnegative matrix factorization using nonnegative polynomial approximations, IEEE Signal Processing Letters 24 (7) (2017) 948–952 (2017).

[17] B. Reznick, Some concrete aspects of Hilbert's 17th problem, Contemporary mathematics 253 (2000) 251–272 (2000).

[18] G. Blekherman, P. A. Parrilo, R. R. Thomas, Semidefinite optimization and convex algebraic geometry, SIAM, 2012 (2012).

[19] L. Vandenberghe, S. Boyd, Semidefinite programming, SIAM review 38 (1) (1996) 49–95 (1996).

[20] D. Backenroth, Methods in functional data analysis and functional genomics, Columbia University, 2018 (2018).

[21] R. Zdunek, A. Cichocki, T. Yokota, B-spline smoothing of feature vectors in nonnegative matrix factorization, in: International Conference on Artificial Intelligence and Soft Computing, Springer, 2014, pp. 72–81 (2014).

[22] C. De Boor, J. W. Daniel, Splines with nonnegative B-spline coefficients, Mathematics of computation 28 (126) (1974) 565–568 (1974).

[23] R. Zdunek, Approximation of feature vectors in nonnegative matrix factorization with gaussian radial basis functions, in: International Conference on Neural Information Processing, Springer, 2012, pp. 616–623 (2012).

[24] T. Goldstein, B. O'Donoghue, S. Setzer, R. Baraniuk, Fast alternating direction optimization methods, SIAM Journal on Imaging Sciences 7 (3) (2014) 1588–1623 (2014).

[25] R. Zdunek, Alternating direction method for approximating smooth feature vectors in nonnegative matrix factorization, in: 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, 2014, pp. 1–6 (2014).

[26] R. Kokaly, al., USGS spectral library version 7 (2017).

[27] M. ApS, The MOSEK optimization toolbox for MATLAB manual. Version 9.0. (2019).
URL `http://docs.mosek.com/9.0/toolbox/index.html`