Least-squares methods for nonnegative matrix factorization over rational functions

Cécile Hautecoeur, Lieven De Lathauwer, Nicolas Gillis, François Glineur

***Abstract*—Nonnegative matrix factorization (NMF) models are widely used to analyze linearly mixed nonnegative data. When the data is made of samplings of continuous signals, the factors in NMF can be constrained to be samples of nonnegative rational functions. This leads to a fairly general model referred to as NMF using rational functions (R-NMF). We first show that, unlike NMF, R-NMF possesses an essentially unique factorization under mild assumptions, which is crucial in applications where the ground-truth factors need to be recovered, as in blind source separation problems. Then we present different approaches to solve R-NMF: the R-HANLS, R-ANLS and R-NLS methods. In our tests, no method significantly outperforms the others in all cases, and all three methods offer a different trade-off between solution accuracy and computational requirements. Indeed, while R-HANLS is fast and accurate for large problems, R-ANLS is more accurate, but also more resources demanding, both in time and memory and R-NLS is even more accurate but only for small problems. Then, crucially we show that R-NMF models outperforms NMF in various tasks including the recovery of semi-synthetic continuous signals, and a classification problem of real hyperspectral signals.**

***Index Terms*—nonnegative matrix factorization, block-coordinate-descent, sampled signals, nonlinear least squares, nonnegative rational functions, projection**

I. Introduction

Linear dimension reduction techniques are simple but powerful methods to reduce the size of a dataset while extracting meaningful information and filtering noise. When the data is nonnegative, it is common to use nonnegative matrix factorization (NMF). In NMF, the nonnegative input data matrix Y is approximated by the product of two nonnegative matrices, A and X, such that $Y \simeq AX^\top$. The number r of columns of these two matrices is typically much smaller than the dimensions of the input matrix, leading to a compressed representation. This allows the description of each column of Y as a nonnegative weighted sum of r characteristic nonnegative elements, the columns of A [27].

Nonnegativity constraints occur naturally in many situations, e.g., when recording intensities, occurrences, frequencies, proportions, and probabilities. Imposing nonnegativity in the factorization leads to more meaningful decompositions: the basis formed by the column of A can be interpreted in the same way as the data, while the input matrix Y is reconstructed using only additive combinations of these basis elements, which leads to a part-based representation [27]. This explains the popularity of NMF in various fields such as image processing, text mining, blind source separation, and microarray data analysis; see [8], [14] and the references therein.

To further improve the quality of the factorization and be even less sensitive to noise, other constraints can be considered on the factors A and X. For example, when the data is smooth, one can consider that the columns of A are discretizations of continuous nonnegative functions like polynomials [11], splines [2], [39], [40], or mixture(s) of Gaussian radial basis functions [38]. NMF can then be solved in several ways, but an efficient approach is to generalize the Hierarchical Alternating Least Squares (HALS) algorithm [7] and solve the problem using block-coordinate descent (BCD) with $2r$ blocks: the columns of A and X. HALS requires to repeatedly project each block on the considered set of nonnegative functions; for example on nonnegative polynomials and splines [18].

When the columns of the input matrix Y are samples of nonnegative continuous signals, mostly smooth with possibly some peaks, it makes sense to consider that they are samples of nonnegtive rational functions. Indeed, when the denominator of a rational function is close to zero, it results in a peak in the signal. In fact, rational functions are able to represent a large range of shapes and curves [22]. NMF over rational functions, R-NMF, has been introduced in [20], and is recalled in Section II. In Section III, we prove that unlike standard NMF, R-NMF is essentially unique under mild conditions, which is very important when the objective is to recover the sources that generated the data.

In [20] it is shown that R-NMF leads to better factorization and reconstruction than standard NMF on noisy data. However, the set of nonnegative rational functions of fixed degree is not convex, and the projection on it is not easy to compute. Therefore, the problem is solved using an HALS-like approach, named R-HANLS, that uses an approximate projection method. We explore in Section V other methods to approximately project on nonnegative rational functions, with the goal to determine whether some methods lead to better projections and/or if some are more adapted for R-NMF, that is, lead to better

This work was supported by the Fonds de la Recherche Scientifique - FNRS and the Fonds Wetenschappelijk Onderzoek - Vlaanderen under EOS Project no 30468160, by the Leuven Institute for Artificial Intelligence (Leuven.ai), by KU Leuven Internal Funds: iBOF/23/064, C16/15/059, IDN/19/014, and by the Francqui Foundation.

factorizations.

One of the main advantages of HALS for standard NMF is the simplicity of its iterations. However, when using rational functions, each iteration is difficult due to the projection. It is thus questionable whether this approach is suitable, so we consider other block decompositions in Section IV, the R-ANLS and R-NLS methods. These methods are then analyzed and compared in Section VI, where we find that R-HANLS is suited for large-scale data, while R-ANLS obtains more accurate factorizations but is slower. R-NLS is effective only for very small data. Moreover, R-NMF is more accurate than NMF using polynomials, splines or vectors on various datasets, like semi-synthetic datasets containing mixture of real reflectance signals, and on a real problem, the Indian Pines classification problem.

II. NMF using rational functions

Consider an input data matrix $Y \in \mathbb{R}^{m\times n}$, containing in each of its columns the samples of a continuous signal taken in m known discretization points $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^m \subset \mathbb{R}$; the sampling points need not to be equidistant. Let T be the interval on which $\boldsymbol{\tau}$ is defined: $T = [\tau_{\min}, \tau_{\max}]$, and $\mathcal{F}^{\boldsymbol{d},T}$ be the set of rational functions of degree $\boldsymbol{d}$ nonnegative on T. The goal of R-NMF is to approximate the columns of Y as a nonnegative linear combination of r functions in $\mathcal{F}^{\boldsymbol{d},T}$. However, as the input signals are known only at points $\boldsymbol{\tau}$, to evaluate the quality of the factorization, we focus on the discretization of $\mathcal{F}^{\boldsymbol{d},T}$ on $\boldsymbol{\tau}$: $\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T} = \{f(\boldsymbol{\tau}) | f \in \mathcal{F}^{\boldsymbol{d},T}\} \subset \mathbb{R}_+^m$, and use the Frobenius norm $\|\cdot\|_F$ of the reconstruction error of Y as objective.

Definition 1 (R-NMF). *Given an input matrix $Y \in \mathbb{R}^{m\times n}$, discretization points $\boldsymbol{\tau} \in \mathbb{R}^m$, the set $\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ of rational functions of degree $\boldsymbol{d}$ nonnegative on T and evaluated on $\boldsymbol{\tau}$, and a factorization rank $r \geq 1$, R-NMF aims to compute a nonnegative matrix $A \in \mathbb{R}_+^{m\times r}$ containing elements of $\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ in each of its columns, that is, $A_{:j} \in \mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ $\forall j$, and a nonnegative matrix $X \in \mathbb{R}_+^{n\times r}$ solving*

$$\min_{A_{:j}\in\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}, X\in\mathbb{R}_+^{n\times r}} \sum_{i=1}^{n} \left\| Y_{:i} - \sum_{j=1}^{r} A_{:j}X_{ij} \right\|_F^2. \tag{1}$$

The choice of rational functions is motivated by their ability to represent a large range of shapes and their utility in applications; they generalize polynomials or splines [35], and they represent the natural way of modeling linear dynamical systems in the frequency domain [22]. A rational function is defined as the ratio of two polynomials: $f(t) = \frac{h(t)}{g(t)}$. Throughout this work, we consider univariate rational functions with fixed degree $\boldsymbol{d} = (d_1, d_2)$, so that h is of degree d_1 and g of degree d_2. As the degree is fixed, the set of rational functions is not a vector space (e.g., it is easy to check that $\frac{1}{x} + \frac{1}{x+1}$ is of degree $(1, 2)$ and not $(1, 1)$). Nevertheless, this set can be parametrized. Indeed, a rational function nonnegative on a fixed interval can be described as a ratio of two polynomials nonnegative on the same interval [23], and nonnegative polynomials can be parametrized using sums of squares [31]. Moreover, as it is often undesirable for factors to tend to infinity, the denominator is imposed to be nonzero in the considered interval. More details are presented in [20]. For example a rational function of degree $\boldsymbol{d} = (2d_1', 2d_2')$ nonnegative on $[-1, 1]$ can be written as:

$$f(t) = \frac{h_1^2(t) + (1-t^2)h_2^2(t)}{g_1^2(t) + (1-t^2)g_2^2(t) + \epsilon} \tag{2}$$

with h_1, h_2, g_1, g_2 polynomials of degree $d_1', d_1'-1, d_2', d_2'-1$ respectively, and ϵ prevents the denominator from going to 0. To evaluate f on points $\boldsymbol{\tau}$, we use the Vandermonde-like matrix for the chosen basis of polynomials, $V_{\boldsymbol{\tau}}^d$ (in our case, the Chebyshev basis). Using the coefficients $(\boldsymbol{h_1}, \boldsymbol{h_2}, \boldsymbol{g_1}, \boldsymbol{g_2}) \in \mathbb{R}^{d_1'+1} \times \mathbb{R}^{d_1'} \times \mathbb{R}^{d_2'+1} \times \mathbb{R}^{d_2'}$ we have

$$f_{\boldsymbol{\tau}}(\boldsymbol{h_1}, \boldsymbol{h_2}, \boldsymbol{g_1}, \boldsymbol{g_2}) = \frac{(V_{\boldsymbol{\tau}}^{d_1'}\boldsymbol{h_1})^2 + (1-\boldsymbol{\tau}^2)\cdot(V_{\boldsymbol{\tau}}^{d_1'-1}\boldsymbol{h_2})^2}{(V_{\boldsymbol{\tau}}^{d_2'}\boldsymbol{g_1})^2 + (1-\boldsymbol{\tau}^2)\cdot(V_{\boldsymbol{\tau}}^{g_2'-1}\boldsymbol{g_2})^2 + \epsilon}. \tag{3}$$

However, this representation is redundant, as multiplying the numerator and the denominator by the same constant leads to the same rational function. Therefore, we impose the denominator g to be monic. It can be proven that this condition is equivalent to imposing $\boldsymbol{g_1}[d_2'+1] = \frac{\sqrt{8+\boldsymbol{g_2}[d_2']^2}}{2}$.

III. Uniqueness

In this section, we focus on exact factorizations $Y = AX^\top$. In such a factorization, if the column $A_{:i}$ is scaled by a factor α_i while the column $X_{:i}$ is scaled by factor $\frac{1}{\alpha_i}$, $AX^\top$ remains unchanged. Moreover, applying the same permutation to the columns of A and X also keeps $AX^\top$ unchanged. This leads to the notion of essentially unique factorizations.

Definition 2. *The factorization $Y = AX^\top$ is said to be **essentially unique** if all the factorizations of Y can be obtained only using consistent permutations and scalings/counterscaling of the columns of A and X.*

As shown in Lemma 1, a matrix Y with unconstrained factorization $Y = AX^\top$ admits an infinite number of other unconstrained factorizations not resulting from permutations and scalings of A and X.

Lemma 1. *Let $Y = AX^\top$, with $A \in \mathbb{R}^{m\times r}$, $X \in \mathbb{R}^{n\times r}$, and $\operatorname{rank}(Y) = r$. Matrices $A' \in \mathbb{R}^{m\times r}$ and $X' \in \mathbb{R}^{n\times r}$ factorize Y if and only if $A' = AQ$ and $X'^\top = Q^{-1}X^\top$ where $Q \in \mathbb{R}^{r\times r}$ is an invertible matrix.*

Proof. We omit the proof, which is quite straightforward. □

To have an essentially unique factorization, we must add constraints on A and/or X. In NMF, the factors A and X are nonnegative. This constraint allows us, under certain conditions, for an essentially unique factorization. However, these conditions are quite restrictive, and are not met in general, see [13] and [14, Chap. 4] and the references therein.

If we consider that the columns of matrix A are samples of rational functions, it is possible to prove that the factorization $Y = AX^\top$ is essentially unique under certain conditions on the rational functions contained in the columns of A [10]: at most one column can contain a polynomial and the poles of all rational functions must be distincts. The number of discretization points must also be larger than twice the sum of the degrees of the rational functions in A, for example $m > 2r(d_1 + d_2)$ in R-NMF.

In R-NMF, the considered rational functions must be nonnegative and of the same degrees. The exact R-NMF problem described below is thus a special case of [10]. Theorem 1 shows that it is possible to ensure that exact R-NMF is essentially unique with milder conditions on A.

Definition 3 (Exact R-NMF)**.** *Given $Y \in \mathbb{R}_+^{m\times n}$, $\boldsymbol{\tau}$, $\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ and r as in R-NMF, compute, if possible, $A \in \mathbb{R}_+^{m\times r}$ with $A_{:j} \in \mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ for all j and $X \in \mathbb{R}_+^{n\times r}$ such that $Y = AX^\top$.*

Let us introduce some useful lemmas. A rational function $f(t)$ of degree $\boldsymbol{d} = (d_1, d_2)$ can be written as:

$$f(t) = \frac{\alpha \prod_{i=1}^{d_1}(t - z_i)}{\prod_{j=1}^{d_2}(t - p_j)} \quad z_i, p_j \in \mathbb{C}, z_i \neq p_j \ \forall i, j, \alpha \neq 0, \ (4)$$

with $\mathcal{Z}=\{z_i\}_{i=1}^{d_1}$ the zeros of $f(t)$, and $\mathcal{P}=\{p_j\}_{j=1}^{d_2}$ its poles, including the complex zeros/poles. In case of multiple poles, the poles are considered as distinct. For example, let $f_1,\ f_2$ be two rational functions with poles $\mathcal{P}_1=\{p_1, p_2, p_3\}$ with $p_1=p_2=p_3$ and $\mathcal{P}_2=\{p_1, p_2\}$ respectively. The set of all poles is $\{p_1, p_2, p_3\}$ and the set of unique poles, that is, poles appearing in exactly one function, is $\{p_3\}$.

Lemma 2. *Let $\{f_l\}_{l=1}^r$ be a collection of rational functions in the form (4), with $\mathcal{P}_l = \{p_{lj}\}_{j=1}^{d_2}$ holding the poles of f_l and $\mathcal{Z}_l = \{z_{li}\}_{i=1}^{d_1}$ holding the zeros of f_l. Let $\mathcal{S} = \{s_k\}_{k=1}^m$ be the set of unique poles, that is, the poles appearing in exactly one function.*
Then any function $f = \sum_l \beta_l f_l$ with $\beta_l \neq 0$ has a denominator with degree at least equal to the cardinality of $\mathcal{S}$ (=m).

Proof. The function f can be written as:

$$f(t) = \frac{\sum_l \beta_l \alpha_l \prod_{i=1}^{d_1}(t - z_{li}) \prod_{q\in\mathcal{U}\setminus\mathcal{P}_l}(t - q)}{\prod_{q\in\mathcal{U}}(t - q)}.$$

Let $\mathcal{U}$ be the set of all poles in $\{f_l\}_{l=1}^r$. We have $\mathcal{S} \subseteq \mathcal{U}$, and all s_k are therefore potential poles of f. Let us check if they can be simplified by the numerator or not. If $s_k \in \mathcal{S}$ is a pole appearing only in $\mathcal{P}_l$, we have $s_k \in \mathcal{U} \setminus \mathcal{P}_i \ \forall i \neq l$. Therefore, when $t = s_k$, only the lth term is non-zero in the numerator. Moreover, $s_k \notin \mathcal{U}\setminus\mathcal{P}_l$ and $s_k \neq z_{li} \ \forall i$ as s_k is a pole of f_l. The numerator is therefore nonzero when $t = s_k$ and s_k is a pole of f. As this is valid for all $s_k \in \mathcal{S}$, rational function f has denominator degree at least equal to the cardinality of $\mathcal{S} = m$. □

Lemma 3. *Let $\{f_l\}_{l=1}^r$ be a collection of r rational functions in form (4), of degree $\boldsymbol{d} = (d_1, d_2)$, and $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^m$ be a set of distinct discretization points with $m > d_1 + rd_2$, so that the denominators of functions f_l do not cancel at these points. If there exist a rational function f^* of degree $\boldsymbol{d}$ so that $f^*(\boldsymbol{\tau}) = \sum_{l=1}^r \beta_l f_l(\boldsymbol{\tau})$, then $f^* = \sum_{l=1}^r \beta_l f_l$.*

Proof. Let $\mathcal{Z}_l = \{z_{li}\}_{i=1}^{d_1}$ and $\mathcal{P}_l = \{p_{lj}\}_{j=1}^{d_2}$ be the zeros and the poles of f_l and $\tilde{\mathcal{Z}} = \{\tilde{z}_i\}_{i=1}^{d_1}$ and $\tilde{\mathcal{P}} = \{\tilde{p}_j\}_{j=1}^{d_2}$ be the zeros and poles of f^*. We have

$$f^*(\boldsymbol{\tau}) = \sum_{l=1}^r \beta_l f_l(\boldsymbol{\tau})$$

$$\Leftrightarrow \frac{\tilde{\alpha}\prod_{i=1}^{d_1}(\boldsymbol{\tau} - \tilde{z}_i)}{\prod_{j=1}^{d_2}(\boldsymbol{\tau} - \tilde{p}_j)} = \frac{\sum_l \beta_l\alpha_l \prod_i(\boldsymbol{\tau} - z_{li}) \prod_{k\neq l,j}(\boldsymbol{\tau} - p_{kj})}{\prod_{l=1}^r \prod_{j=1}^{d_2}(\boldsymbol{\tau} - p_{lj})}$$

$$\Leftrightarrow \Big(\tilde{\alpha}\prod_{i=1}^{d_1}(\boldsymbol{\tau} - \tilde{z}_i)\Big)\Big(\prod_{l=1}^{r}\prod_{j=1}^{d_2}(\boldsymbol{\tau} - p_{lj})\Big) = \quad (5)$$

$$\Big(\prod_{j=1}^{d_2}(\boldsymbol{\tau} - \tilde{p}_j)\Big)\Big(\sum_{l=1}^{r}\beta_l\alpha_l\prod_{i=1}^{d_1}(\boldsymbol{\tau} - z_{li})\prod_{k\neq l}\prod_{j=1}^{d_2}(\boldsymbol{\tau} - p_{kj})\Big) \quad (6)$$

Elements (5) and (6) are polynomials of degree at most d_1+rd_2, evaluated at discretization points $\boldsymbol{\tau}$. As $\boldsymbol{\tau}$ contains m distinct points with $m > d_1+rd_2$, these two polynomials must be equal everywhere. Therefore, $f^* = \sum_{l=1}^r \beta_l f_l$. □

We now present conditions on matrices A and X that imply that the exact R-NMF $AX^\top$ is essentially unique.

Theorem 1. *Let $A \in \mathbb{R}^{m\times r}$ and $X \in \mathbb{R}^{n\times r}$ be of rank r. Suppose all columns of A are the discretizations of rational functions A_j for $j = 1, 2, \ldots, r$, of degree (d_1, d_2) on m distinct points $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^m$, with $m > d_1 + rd_2$ and $\boldsymbol{\tau}$ not containing poles of the functions A_j. Suppose that for all sets containing 2 functions or more, there are at least $d_2 + 1$ unique poles, that is, poles appearing in exactly one function. Then the exact R-NMF $AX^\top$ is essentially unique.*

Proof. Let A', X' be such that $A'X'^\top = AX^\top$. As A, X are of rank r, we know by Lemma 1 that each column $A'_{:j}$ can be written as a linear combination of the columns of A: $A'_{:j} = \sum_{l=1}^r \beta_l A_{:l} = \sum_{l=1}^r \beta_l A_l(\boldsymbol{\tau})$. To be valid, $A'_{:j}$ must be the discretization of a rational function of degree (d_1, d_2), we name this function A'_j. As $m > d_1 + rd_2$, by Lemma (3), A'_j must be the linear combination of the rational functions in A: $A'_j = \sum_l \beta_l A_l$.

To avoid the trivial case of permutation and scaling, there must be at least one $A'_{:j}$ that is the combination of two or more columns of A. As all sets $\{A_i\}$ containing two functions or more have at least $d_2 + 1$ unique poles, using Lemma 2 we know that A'_j has denominator degree at least $d_2 + 1$. This is in contradiction with the fact that A'_j is a rational function with degree (d_1, d_2). It is therefore not possible to find a valid and not trivial A' so that $A'X'^\top = AX^\top$ and the factorization $AX^\top$ is essentially unique. □

Corollary 1. *Let $A \in \mathbb{R}^{m\times r}$ and $X \in \mathbb{R}^{n\times r}$ be of rank r, with the columns of A obtained through evaluation*

of rational functions of degree (d_1, d_2) on m distinct points $\boldsymbol{\tau}$, with $m > d_1 + r d_2$, and $\boldsymbol{\tau}$ not containing poles of functions in A. If each function has at least $\left\lceil \frac{d_2+1}{2} \right\rceil$ poles distinct from all other functions, the exact R-NMF $Y = AX^\top$ is essentially unique.

Note that the nonnegativity constraint is not necessary for Theorem 1 and Corollary 1 to hold.

Nevertheless, when using representation like (3), functions A_j do not have real poles on interval $T = [\boldsymbol{\tau}_{\min}, \boldsymbol{\tau}_{\max}]$, because of the ϵ added to the denominator. This means that in this case condition "$\boldsymbol{\tau}$ not containing poles of functions in A" is always met.

The conditions of Corollary 1 are generically satisfied, that is, these conditions will be satisfied with probability one for generic rational functions (whose coefficients are generated randomly following a continuous distribution, e.g., Gaussian). In fact, generically, not even two poles are the same.

Moreover, it is straightforward to check *a posteriori* whether the conditions of Corollary 1 are satisfied for a given computed factorization.

IV. Algorithms for R-NMF

In this section, we present three different block decompositions of R-NMF leading to different algorithms.

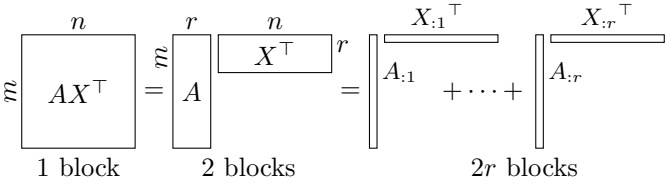

Fig. 1: Illustration of the three block decompositions.

A. General Nonlinear Least Squares approach (R-NLS)

We substitute in (1) $A_{:j}$ by $f_{\boldsymbol{\tau}_j}$ from equation (3), and X_{ij} by C_{ij}^2 to express R-NMF in an unconstrained way:

$$\min_{\substack{\boldsymbol{h_{1j}}, \boldsymbol{h_{2j}}, \\ \boldsymbol{g_{1j}}, \boldsymbol{g_{2j}}, C}} \sum_{i=1}^{n} \left\| Y_{:i} - \sum_{j=1}^{r} f_{\boldsymbol{\tau}_j}(\boldsymbol{h_{1j}}, \boldsymbol{h_{2j}}, \boldsymbol{g_{1j}}, \boldsymbol{g_{2j}}) C_{ij}^2 \right\|^2. \quad (7)$$

This problem can be solved using a standard nonlinear least squares solver. The same approach for polynomials has been proposed in [11]. Note however that in the cited work a compression method is suggested to pre-process the data and reduce the complexity of the problem, but this is not possible in our case because rational function are not linearly parametrizable, that is, they cannot be described using a linear combination of some basis elements.

B. Using Alternating Nonlinear Least Squares (R-ANLS)

Using all-at-once algorithms as R-NLS to solve NMF problems may be computationally costly, especially for large problems. Therefore, most NMF algorithms consider instead alternating schemes [7],[25],[27],[28]. The problem is then solved by alternating on A and X considering the other matrix as fixed, as sketched in Algorithm 1. As $f_{\boldsymbol{\tau}_j}$ is a nonlinear function, each sub-problem is nonlinear, and this method is called alternating nonlinear least squares.

Algorithm 1 Alternating Nonlinear Least Squares

function R-ANLS(Y, A, X)
  while stopping criterion not satisfied **do**:

$$A \leftarrow \underset{\substack{\boldsymbol{h_{1j}}, \boldsymbol{h_{2j}}, \\ \boldsymbol{g_{1j}}, \boldsymbol{g_{2j}}}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| Y_{:i} - \sum_{j=1}^{r} f_{\boldsymbol{\tau}_j}(\boldsymbol{h_{1j}}, \boldsymbol{h_{2j}}, \boldsymbol{g_{1j}}, \boldsymbol{g_{2j}}) X_{ij} \right\|^2 \quad (8)$$

$$X \leftarrow \left(\underset{C \in \mathbb{R}^{n \times r}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| Y_{:i} - \sum_{j=1}^{r} A_{:j} C_{ij}^2 \right\|^2 \right)^2 \quad (9)$$

  return A, X

Problems (8) and (9) are unconstrained and can be solved using a standard nonlinear least squares solver. Note that problem (9) is separable in n independent sub-problems, as the rows of X, are independent (which is not the case for the rows of A): for all $i \in \{1, \cdots, n\}$,

$$X_{i:} \leftarrow \left(\underset{C_{i:} \in \mathbb{R}^r}{\operatorname{argmin}} \left\| Y_{:i} - \sum_{j=1}^{r} A_{:j} C_{ij}^2 \right\|^2 \right)^2.$$

C. Using Hierarchical Alternating Nonlinear Least Squares (R-HANLS)

A popular and effective approach for NMF is the Hierarchical Alternating Least Squares method (HALS). This method further decomposes the problem in smaller blocks: the columns of A/X are updated successively, considering all the other elements as fixed [7]; see also [15]. Because of the quadratic and separable structure of the objective function (that is, the Hessian is diagonal), minimizing (1) when all variables are fixed except a column of A or X can be done by projecting the unconstrained minimizer on the corresponding feasible region. This region is the set $\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ of nonnegative rational functions with fixed degrees for A, or the set $\mathbb{R}_+^n$ of nonnegative vectors for X.

The unconstrained minimizer can easily be found for columns of A and X by cancelling the gradient. Algorithm 2 sketches this approach, using $[\cdot]_S$ for the projection on set S. The projection on $\mathbb{R}_+^n$ is a simple thresholding operation, setting all negative values to 0, while the projection on $\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ is not trivial and discussed in the next section. Moreover, equation (11) is separable: the value of X_{is} can be computed independently from X_{js}, but this is not the case for $A_{:s}$ in equation (10), as the projection is not separable unlike the thresholding operation.

V. Projection on nonnegative rational functions

As mentioned in Section II, rational functions nonnegative on a fixed interval T can be described as the ratio of two polynomials nonnegative on T, with denominator further imposed to be nonzero on T. Let $\mathcal{P}^d$ be the set of polynomials of degree d, $\mathcal{P}_+^{d,T}$ be the set of polynomials

Algorithm 2 R-HANLS

function R-HANLS($Y,\ A,\ X$)
  while stopping criterion not satisfied **do**
    for $A_{:s} \in A$ **do**

$$A_{:s} \leftarrow \left[\frac{YX_{:s} - \sum_{j\neq s} A_{:j}(X_{:j})^\top X_{:s}}{\|X_{:s}\|^2}\right]_{\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}} \quad (10)$$

    for $X_{:s} \in X$ **do**

$$X_{:s} \leftarrow \left[\frac{Y^\top A_{:s} - \sum_{j\neq s} X_{:j}(A_{:j})^\top A_{:s}}{\|A_{:s}\|^2}\right]_{\mathbb{R}_+^n} \quad (11)$$

  return $A,\ X$

of degree d nonnegative on interval T, $\mathcal{P}_{++}^{d,T}$ be the set of polynomials of degree d positive on interval T, and $\boldsymbol{z}$ be the result of evaluating a function $z(t)$ on discretization points $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^m$, $\boldsymbol{z} = z(\boldsymbol{\tau})$. Projecting $\boldsymbol{z}$ on rational functions nonnegative on T is therefore equivalent to solving

$$\min_{h\in\mathcal{P}_+^{d_1,T},\ g\in\mathcal{P}_{++}^{d_2,T}} \left\|\boldsymbol{z} - h(\boldsymbol{\tau})/g(\boldsymbol{\tau})\right\|_2^2. \quad (12)$$

A. Existing approaches to approximate (nonnegative) rational functions

Solving problem (12) is not trivial, even when removing the nonnegativity constraints. Though many works exist in the unconstrained case, most of them consider the infinity norm in (12) [36], and there are very few works imposing nonnegativity: to the best of our knowledge this problem is only addressed in [32], [34], for the infinity norm.

In the unconstrained case, many works are based on another representation of rational functions, namely the Barycentric representation which is as follows

$$f(t) = \sum_{i=1}^{d} \frac{\omega_i z_i}{t-\alpha_i} \Bigg/ \sum_{i=1}^{d} \frac{\omega_i}{t-\alpha_i}. \quad (13)$$

The advantage of this representation is that the basis used, that is, the sets of $\{\alpha_i\}_{i=1}^d$, can be adapted as the algorithm proceeds to avoid numerical problems at nonsmooth points [12], or Froissart doublets [30]. Moreover, when $t \to \alpha_i$, then $f(t) \to z_i$, which allows one to optimize only the ω_i's. The most common method using this representation is the adaptive Antoulas–Anderson (AAA) algorithm [30]. This method gradually increases the size of the basis by judiciously choosing the α_i points to be added. It does not seek to optimise a particular norm, but is a good initialization for future optimisation [9], [12], [21], [24]. On the other hand, even if it is not presented as such, one can see Vector Fitting [16] as using the same representation. In this method, the whole basis is chosen at once. Then one optimises iteratively, using at each iteration the poles of the denominator found at previous iteration as new basis.

In both methods, once the basis is chosen, the numerator h and denominator g of f are found by optimizing $\|\boldsymbol{z}g(\boldsymbol{\tau}) - h(\boldsymbol{\tau})\|$ rather than $\|\boldsymbol{z} - h(\boldsymbol{\tau})/g(\boldsymbol{\tau})\|$. These methods give good results, but are difficult to use in the context of nonnegative rational functions, because nonnegativity is difficult to express in Barycentric form.

Many methods try to get rid of the denominator which is difficult to optimise. Thus, [33] but also [29] and [37] have proposed to solve the problem iteratively, using a guess of the denominator, g^{k-1}, improved throughout iterations, by solving

$$(g^k, h^k) = \operatorname*{argmin}_{g\in\mathcal{P}^{d_2},h\in\mathcal{P}^{d_1}} \left\|\frac{\boldsymbol{z}g(\boldsymbol{\tau}) - h(\boldsymbol{\tau})}{g^{k-1}(\boldsymbol{\tau})}\right\|. \quad (14)$$

In the same idea, a special case of the RKFIT (Rational Krylov fitting) algorithm from [4] focuses on finding a good denominator by solving the following problem iteratively:

$$\min_{g^k\in\mathcal{P}^{d_2}} \left\|\frac{\boldsymbol{z}g^k - h'(g^k;\boldsymbol{z},g^{k-1})}{g^{k-1}}\right\|, \quad (15)$$

where $h'(g^k;\boldsymbol{z},g^{k-1}) = \operatorname*{argmin}_h \left\|\frac{\boldsymbol{z}g^k - h}{g^{k-1}}\right\|$. The problem in h when g^k is fixed has an analytic solution (the solution of a similar problem is presented in Appendix A, in the explanation of **RKFIT+**). This reformulation allows for fewer parameters to be optimised at each iteration.

When using the infinity norm in (12), if $g(\tau_i)$ is positive for all i, the problem can be rewritten as:

$$\min_{h\in\mathcal{P}^{d_1},g(\tau_i)>0,u} u \quad \text{s.t.} \quad \begin{cases} z_i g(\tau_i) - h(\tau_i) \leq u g(\tau_i) \\ h(\tau_i) - z_i g(\tau_i) \leq u g(\tau_i) \end{cases}. \quad (16)$$

If we fix u, then the problem is a feasibility problem, and therefore it is possible to perform a bisection search on u to find the solution. This is the method used in [32], [34] to solve the problem on nonnegative rational functions. The numerator and the denominator of the rational functions are modeled using sum of squares, which makes problem (16) a semidefinite programming feasibility problem for u fixed.

Finally, using equation (3), it is possible to see problem (12) as a nonlinear least squares problem and to solve it using standard methods [36].

B. Proposed projection methods

Let us present five approaches to solve the projection problem onto the set of nonnegative rational functions. Some details of implementation are omitted and presented in Appendix A to lighten the text. All these approaches are approximations, that is, they are not guaranteed to converge to a globally optimal solution. Therefore, we will analyze in Section V-C their average behavior in several situations. The reason is that solving this problem is, unfortunately, believed to be NP-hard in general. Indeed, minimizing rational functions is NP-hard in general [23], and to the best of our knowledge there does not exist a polynomial-time algorithm to perform the projection

on rational functions (with or without the nonnegativity constraint).

Least Squares: Using equation (3), the projection problem can be rewritten in an unconstrained way and solved using a standard nonlinear least squares solver, as in R-NLS or R-ANLS. This is the approach used in [20].

Alternating Least Squares: The projection problem can also be divided in two blocks, and solved using a BCD approach. Finding the best possible numerator when the denominator g is fixed is a convex problem on polynomials:

$$\underset{h\in\mathcal{P}_+^{d_1,T}}{\operatorname{argmin}}\left\|\boldsymbol{z}-\frac{h(\boldsymbol{\tau})}{g(\boldsymbol{\tau})}\right\|^2. \tag{17}$$

This problem is described in more details in Appendix A.

When the numerator h is fixed, finding the best denominator is a challenge as the problem is not convex. Actually this problem is a special case of the projection on rational functions, when the degree of the numerator is equal to 0. So it can also be solved using nonlinear least squares solvers via equation (3). As this problem has fewer variables than the original one, it is reasonable to assume that it will be solved faster.

Algorithm 3 Alternating Least Squares

Input: $\boldsymbol{z}$: signal to approximate, d_1, d_2: degree of the numerator/denominator, $\boldsymbol{\tau}$: discretization points, g: initial guess of the denominator, tol: tolerance of the algorithm

1: **function** ALTERNATING LS($\boldsymbol{z}$, d_1, d_2, $\boldsymbol{\tau}$, g, tol)
2: **while** $\frac{\text{err}_{\text{prev}}-\text{err}}{\text{err}}$ >tol **do**
3: compute h as in problem (17)
4: $g=\underset{g\in\mathcal{P}_{++}^{d_2,T}}{\operatorname{argmin}}\|\boldsymbol{z}-h(\boldsymbol{\tau})/g(\boldsymbol{\tau})\|^2$.
5: $f(\boldsymbol{\tau})=h(\boldsymbol{\tau})/g(\boldsymbol{\tau})$
6: $\text{err}_{\text{prev}}=\text{err}$, $\text{err}=\|\boldsymbol{z}-f(\boldsymbol{\tau})\|^2$
7: **return** $f(\boldsymbol{\tau})$

Conic: This method is inspired by equation (14). From a given estimate of the denominator $\tilde{g}$, we aim to recover the rational function by optimizing a problem without variables at the denominator. The problem we aim to solve is not the same as in (14), and is motivated in Appendix A. Indeed, we aim to approximate $\boldsymbol{z}$ by $f(\boldsymbol{\tau})=\frac{h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})+\delta(\boldsymbol{\tau})}$, with $\tilde{g}\in\mathcal{P}_{++}^{d_2,T}$ fixed, by solving

$$\underset{h\in\mathcal{P}_+^{d_1,T},\delta\in\mathcal{P}_+^{d_2,T}}{\operatorname{argmin}}\left\|\frac{\boldsymbol{z}\tilde{g}(\boldsymbol{\tau})+\boldsymbol{z}\delta(\boldsymbol{\tau})-h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}\right\|^2. \tag{18}$$

Note that the parametrization $f(\boldsymbol{\tau})=\frac{h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})+\delta(\boldsymbol{\tau})}$ allows us to represent any rational function nonnegative on a fixed interval, and that the cost function of problem (18) is an upper bound of the cost function of problem (12). Moreover, if $\boldsymbol{z}$ is a nonnegative rational function of appropriate degrees, for any $\tilde{g}$ it is possible to find h and δ such that the cost function (18) is equal to zero and $\boldsymbol{z}=f(\boldsymbol{\tau})$.

The choice of $\tilde{g}$ is crucial for this algorithm: the smaller is δ, and therefore the closer is $\tilde{g}$ from the denominator of the rational function, the closer are (18) and (12). Thus problem (18) is solved iteratively, updating $\tilde{g}$ as $\tilde{g}+\delta$. Note that to avoid to increase $\tilde{g}$ indefinitely, it is normalized so that $\tilde{g}(\tau_m)=1$ before a new iteration, without loss of generality. This method is sketched in Algorithm 4.

Algorithm 4 Conic

Input: $\boldsymbol{z}$: signal to approximate, d_1, d_2: degree of the numerator/denominator, $\boldsymbol{\tau}$: discretization points, $\tilde{g}$: initial guess of the denominator, tol: tolerance of the algorithm

1: **function** CONIC($\boldsymbol{z}$, d_1, d_2, $\boldsymbol{\tau}$, $\tilde{g}$, tol)
2: **while** nb >tol and $\frac{\text{err}_{\text{prev}}-\text{err}}{\text{err}}$ >tol **do**
3: compute h and δ as in problem (18)
4: $g=\tilde{g}+\delta$
5: $f(\boldsymbol{\tau})=\frac{h(\boldsymbol{\tau})}{g(\boldsymbol{\tau})}$
6: $nb=\|\tilde{g}-g/g(\tau_m)\|^2$; $\tilde{g}=g/g(\tau_m)$
7: $\text{err}_{\text{prev}}=\text{err}$; $\text{err}=\|\boldsymbol{z}-f(\boldsymbol{\tau})\|^2$
8: **return** $f(\boldsymbol{\tau})$

RKFIT+: This approach is inspired from the RKFIT method presented in [4]. To find a good denominator, we consider (18) and replace $h(\boldsymbol{\tau})$ by its best value when δ and $\tilde{g}$ are considered as fixed, without taking into account the nonnegativity constraint. This means that we consider $h'(\tilde{g},\delta,\boldsymbol{z},\boldsymbol{\tau})=\operatorname{argmin}_{h\in\mathcal{P}^{d_1}}\left\|\boldsymbol{z}+\frac{\boldsymbol{z}\delta(\boldsymbol{\tau})-h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}\right\|$ instead of h. As the nonnegativity constraint is omitted, this problem can be solved analytically using matrix operations (see Appendix A). This leads us to the following problem:

$$\underset{\delta\in\mathcal{P}_+^{d_2,T}}{\operatorname{argmin}}\left\|\boldsymbol{z}+\frac{\boldsymbol{z}\delta(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}-\frac{h'(\tilde{g},\delta,\boldsymbol{z},\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}\right\|^2. \tag{19}$$

To find a good projection on the set of nonnegative rational functions we iterate over instances of problem (19). An iterative scheme is useful because problem (19) relies on the fixed parameter $\tilde{g}$. The pseudo-code of RKFIT+ is quite similar to the one of Conic (Algorithm 4). Lines 5 and 7 are deleted, and lines 3 is replaced by

3: compute δ as in problem (19)

Moreover, problem (17) is solved after the while loop to recover the numerator.

LinProj: This approach has been inspired from [32], [34]. In this case we consider the infinity norm instead of the squared norm, to express the problem as a bisection search over feasibility problems on polynomials as in (16). These feasibility problems can even have linear constraints if we impose h and g to be nonnegative on points $\tau_i\in\boldsymbol{\tau}$ instead of being nonnegative on the interval T (this is different from what is done in [32], [34]). The feasibility problem is then:

$$\min_{h(\tau_i)\geq 0, g(\tau_i)\geq 1} 0 \quad \text{s. t.} \quad \begin{cases}\boldsymbol{z}_i g(\tau_i)-h(\tau_i)\leq u g(\tau_i)\\ h(\tau_i)-\boldsymbol{z}_i g(\tau_i)\leq u g(\tau_i)\end{cases} \forall i, \tag{20}$$

and a bisection algorithm is sketched in Algorithm 5. Note that g is prevented from containing values smaller

than 1 at points τ_i without loss of generality, to simplify the feasibility problem, preventing $[-ug(\tau_i), ug(\tau_i)]$ from being too small.

Algorithm 5 LinProj

Input: $\boldsymbol{z}$: signal to approximate, d_1, d_2: degree of the numerator/denominator, $\boldsymbol{\tau}$: discretization points, tol: tolerance of the algorithm

function LinProj($\boldsymbol{z}$, d_1, d_2, $\boldsymbol{\tau}$, tol)
  $u_{\max} = \max_i \left\{ z_i - \sum_{s=1}^{m} z_s/m \right\}$; $u_{\min} = 0$
  while $u_{\max} - u_{\min} >$ tol **do**
    $u_{\text{med}} = (u_{\max} + u_{\min})/2$
    if problem (20) on u_{med} is feasible **then**
      $u_{\max} = u_{\text{med}}$
    else
      $u_{\min} = u_{\text{med}}$
  Find h, g a feasible solution of (20) using $u_{\max}$
  return $\frac{h(\boldsymbol{\tau})}{g(\boldsymbol{\tau})}$

C. Comparison of the projection methods

We now compare these five proposed projection approaches. Algorithms have a tolerance `tol` of 10^{-8}. We consider two sets of inputs:

- The signals to project are the discretization of non-negative rational functions, whose numerator and denominator degrees are d_1 and d_2, respectively. An exact recovery is thus possible (exact).
- The signals to project are the same as in previous case except that we add a Gaussian noise with noise level 20dB (noisy).

Unless specified otherwise, the rational functions have degree $(16, 16)$, with 250 discretization points equally spaced on $[-1, 1]$. Fig. 2 displays the results. The quality of the final projection is computed as the squared norm of the difference between the signal to project and the computed projection, divided by the squared norm of the signal to project. The first observation from this figure is that no method outperforms all others. Indeed, even though RKFIT+ seems quite appropriate for "exact" data, as it obtains the lowest relative error and is among the fastest, it is quite inaccurate for noisy data. On the contrary, Least Squares and Alternating Least Squares provide the best projections on noisy data, but they obtain high errors when there is no noise. When comparing these two approaches, the Least Squares appears to be the best as it is significantly faster and obtains more accurate results. Therefore, we do not consider Alternating Least Squares in what follows. Linproj generally obtains low relative errors, but sometimes it is unable to find a good candidate when there is noise. Finally, the Conic approach is not very accurate compared to the others, but it is the fastest.

We conclude from these experiments that Least Squares and RKFIT+ are the more promising projection methods, but they are not always better than the others, and do not outperform them significantly.

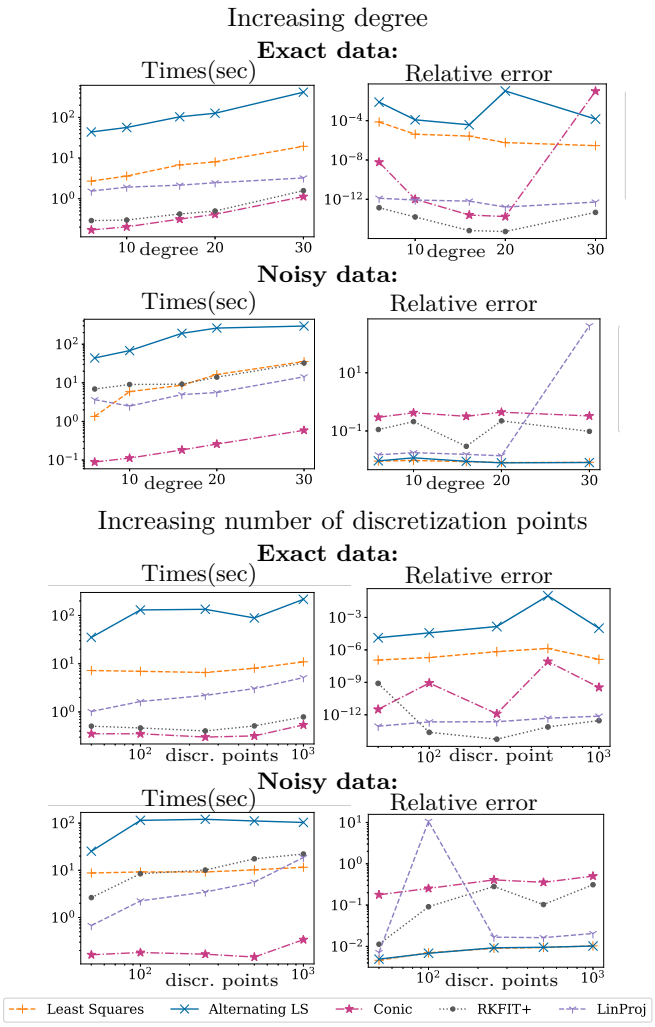

Fig. 2: Comparison of the projections. The results are averaged over 10 trials. The plots represent the time needed for computations (left) and the relative error (right).

VI. Performance and Comparison of R-NMF algorithms

In this section, we first briefly discuss the computational complexity of the proposed algorithms. Then, we compare the R-NMF algorithms presented in Section IV on purely synthetic datasets to analyze their reconstruction ability and their efficiency. After that, the most promising methods are compared to standard HALS and HALS using polynomials or splines [18] on semi-synthetic datasets. We chose to use HALS because it is fast and obtains comparable results in terms of accuracy as other approaches [18]. The methods are also compared on a classification task on a real dataset: the Indian Pines dataset[1].

The least squares solver used for the experimentation is the Python/SciPy function `least_squares`[2] with default

[1] http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes#Indian_Pines

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html

parameters, which solves the least squares problems using a trust region reflective algorithm [6].

The code is available from https://codeocean.com/capsule/2977752/tree, along with other codes developed during the PhD thesis of the first author.

A. Algorithmic complexity of the methods

Let the following reasonable assumption apply: $r < d < n, m$, where d is the number of degrees of freedom of the used function, e.g., $d_1 + d_2 + 1$ for rational functions, the degree plus one for polynomials, and the number of interior knots plus two for splines. The number r is the rank of factorization, n is the number of observations, and m is the number of discretization points. Let the complexity of the least squares solver be $ls(k)$ where k is the size of the Jacobian, and $p(k)$ be the complexity of the projections for polynomials and splines, where k is the number of variables to optimize by the algorithm.

We know that an update of HALS for X has complexity $\mathcal{O}(rmnI)$, where I is the number of iterations. The complexities of HALS using polynomials or splines from [18], and of R-HANLS using least-squares projection, R-ANLS and R-NLS can also be computed. Their value is summarized in Table I. Among HALS methods, R-HANLS is the slowest. Indeed, rational functions are not linearly parametrizable and m appears in the complexity, unlike for polynomials or spline, where m is replaced by d which is significantly lower. Nevertheless, R-HANLS is much faster than R-ANLS or R-NLS for large datasets.

Method	Complexity
HALS	$\mathcal{O}(rmnI)$
Poly/splines	$\mathcal{O}(rdnI + rp(d^2)I)$
R-HANLS	$\mathcal{O}(rmnI + r\ ls(md)I)$
R-ALS	$\mathcal{O}(rmnI + ls(rdmn)I)$
R-LS	$\mathcal{O}(ls(rmn(n+d)))$

TABLE I: Computational complexity of the various NMF and R-NMF methods.

B. Datasets

We use synthetic datasets generated as follows. We generate matrix $X \in \mathbb{R}_+^{n\times r}$ randomly, following a Dirichet distribution whose parameters are equal to $\alpha = 1/r$. The data provided to the algorithms is $Y = AX^\top + N$ where N is additive Gaussian noise with known Signal to Noise Ratio (SNR). The matrix A is generated in two ways:

- a "purely synthetic" A which is the discretization of r nonnegative rational functions. The functions are generated as follows. We first create a nonnegative polynomial of degree d_1 that is perturbed using a rational function of degree $(1, 2)$. This creates a smooth signal with some peaks. The signal is then projected on the set of nonnegative rational functions of degree (d_1, d_2). In this situation, it is therefore possible to find the exact solution of the problem.
- a "semi-synthetic" A whose columns are the real reflectance signals of Adulania, Clinochlore, Hypersthene, Olivine, Spessatine, Andesine, Celestine and Kaolinite evaluated on 414 nonequally spaced points. These signals are showed on Fig. 3 (left) and come from the U.S. Geological Survey (USGS) database [26]. These signals are not particularly close to rational functions, but they are generally smooth even though they present some peaks. If r is smaller than 8, we only consider the first r signals in the list.

In all our experiments we impose methods to have the same number of degrees of freedom (except standard HALS which operates over unstructured nonnegative vectors). This means that if we use rational functions with degree (d_1,d_2), we use polynomials of degree $d_1 + d_2$, and splines of degree 3 with $d_1 + d_2 - 1$ interior knots. Let A^k, X^k denote the factors obtained at iteration k. Accuracy is evaluated trough the relative residue computed as

$$\frac{\|AX^\top - A^k {X^k}^\top\|}{\|AX^\top\|}. \tag{21}$$

Note that this evaluation is performed on $AX^\top$, that is, the data before adding the noise, and therefore the quality is evaluated on data not provided to the algorithm. The stopping criterion of the algorithms is the following:

$$sc^k = \frac{\|Y - A^{k-1}{X^{k-1}}^\top\| - \|Y - A^k {X^k}^\top\|}{\|Y - A^k {X^k}^\top\|} < 10^{-12}. \tag{22}$$

We also impose algorithms to have a maximum running time. Methods based on HALS are limited to 200 seconds, while R-ANLS and R-NLS are limited to 1000 seconds. These times have been inspired from Table I, and selected to be not too important, while allowing the algorithm to converge in most cases, as we will see in the experiments.

We also report the quality of factorizations by computing the Signal to Interference Ratio (SIR) between the computed A' and the original A, proposed in [8]. If A contains n columns, the SIR is:

$$SIR(A', A) = \frac{10}{n}\sum_{i=1}^{n} \log\Big(\frac{\|A'_{:i}\|_2^2}{\|A_{:i} - A'_{:i}\|_2^2}\Big).$$

The larger the SIR, the closer A' is to A. As the factors can be permuted without loss of generality, we first compute the best permutation of the columns of of A' before computing the SIR.

In what follows, each test is performed 10 times, using different initializations. To summarize the performance, in Figures 4 and 8, we compute the minimal and the maximal value obtained for each criterion, and put a marker at the mean value of the criterion. If the graph shows the evolution of two criteria with respect to a parameter (like n, m, d or r), only the mean value is presented to improve readability. We consider that an algorithm converged at iteration k if $\frac{sc^k - sc^o}{sc^k} < 10^{-3}$ for all $o \geq k$. This is used to evaluate the time needed by each algorithm to converge.

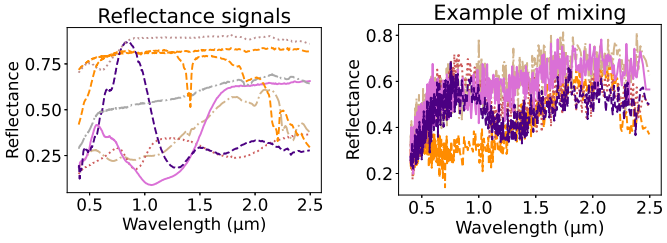

Fig. 3: Left: Considered real reflectance signals. Right: Example of mixing of those signals with noise level 20dB. Each of the five signals is a column of Y.

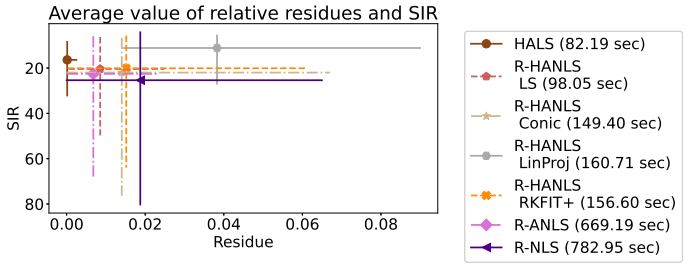

Fig. 4: Summary of performance for varying $n = [20, 100]$, $d = [6, 10]$ and $r = [5, 10]$, in the presence of noise.

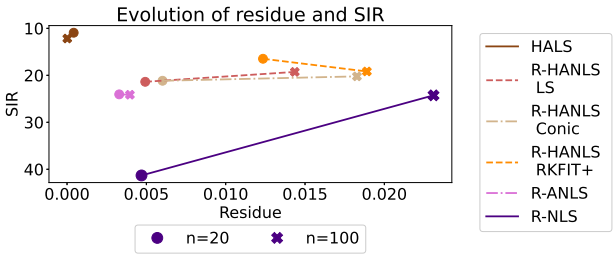

Fig. 5: Performance for varying n. Data is not noisy. Average for $d = [6, 10]$, $r = [5, 10]$.

C. Initialization of the projections in R-HANLS

To get the best out of R-HANLS with the different projections, we use the fact that the last iterates of R-HANLS tend to become close to each other. Therefore, we exploited knowledge from previous iterations, as suggested in [20]:

- Least Squares: use the previous projection as a starting point of the least squares solver.
- Conic and RKFIT+: use the previously obtained denominator as first guess.
- LinProj: use a potentially better $u_{\max} = \max_i\{|\boldsymbol{z}_i - f_{\text{prev}}(\tau_i)|\}$.

Moreover, the tolerance of the projection methods is decreased progressively from 10^{-2} to 10^{-8}, and Conic and RKFIT+ are limited to one iteration. This leads to accurate results in a reasonable time. Nevertheless, we noted during experiments that using knowledge from previous iterations is particularly beneficial for Least Squares.

D. Purely synthetic datasets

Let us present the result with and without noise.

Case without noise: Although there is no noise to filter, it is useful to analyze the data and find the factors behind them. By the uniqueness property of R-NMF in Section III, we can hope that the methods based on rational functions are able to recover the original signals. We observe on Fig. 4 that even though the SIR of methods using rational functions are on average better than the SIR recovered by HALS (which uses any nonnegative vector to represent each column of A), this is not always the case, and there is much more variability on the results when using rational functions than when using HALS. Nevertheless, the best SIR obtained by methods using rational functions are much better than the best SIR obtained when using HALS (except for R-HANLS using LinProj projection).

Moreover, HALS obtains the best residue, which is expected as it has more degrees of freedom. It is therefore difficult to beat HALS in terms of pure data approximation when data is noiseless. Among methods using rational functions, we can see that the LinProj projection is not appropriate; this method is therefore not explored further in what follows. The other R-NMF methods have similar performances, except in terms of computation time. Nevertheless, it seems that R-ANLS is the most accurate method in terms of obtained residue, while R-HANLS-based methods are faster.

We observe on Fig. 5 that when the number of observations n is small ($n = 20$), R-NLS is able to recover the original signals, as this method obtains a low residue and a high SIR. However, it is unable to do so when the number of observations increases. We may wonder if this bad result is due to a too tight time constraint, which prevents the algorithm from converging, but even by running the algorithm for 1h (that is, three times longer), the performance did not improve significantly. R-ANLS is the most robust method among methods using rational functions when n changes as its residue is not impacted by this change, unlike other R-NMF methods.

Case with noise: When noise is added to the dataset, NMF is also useful to filter noise in the data, which can be evaluated through the relative residue (21): a low relative residue means a good ability to filter the noise. Fig. 6 shows the average results for low and high noise levels. We observe that the performance of all algorithms deteriorates when the level of noise increases, as expected. Using the Conic or RKFIT+ projections in R-HANLS does not work well when the noise level is high. The noise level has a high impact on the residue of HALS, which means that it does not perform well at filtering the noise on the data. However, the similarity of the recovered factors to the original ones is not much impacted by the noise level and stays around 15 dB, which is not a very good SIR. R-HANLS LS and R-ANLS obtain the best performances when the noise level is high, both in terms of SIR and residue. Although their SIR is also around 15dB, meaning

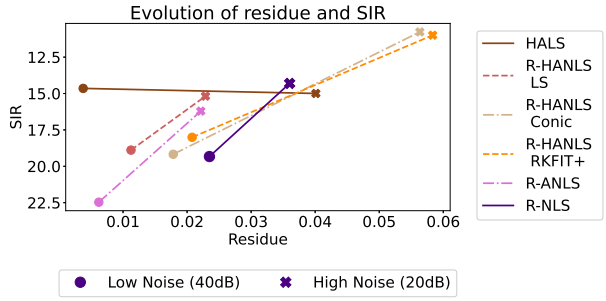

Fig. 6: Performance for different noise levels. Average for $n = [20, 100]$, $d = [6, 10]$ and $r = [5, 10]$.

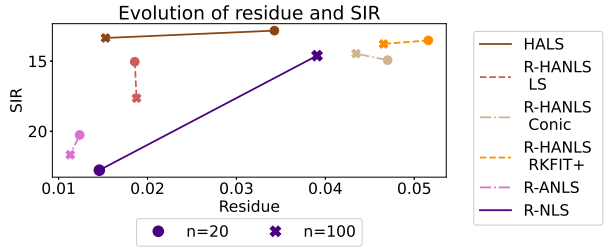

Fig. 7: Performance for varying n, in the presence of noise. Average for $d = [6, 10]$ and $r = [5, 10]$.

that the recovered factors are not so similar to the original ones, the obtained factors are able to model the original data with a small residual, around 0.2. We see on Fig. 7 that increasing n, the number of observations, has a very different impact depending on the used methods: it makes R-NLS perform worse, but it helps the other methods, especially HALS.

E. Semi-synthetic datasets

We saw in previous sections that using rational functions in NMF when data is composed of rational functions can help significantly the algorithm, but is very sensitive to initialization. The use of rational functions is especially relevant for difficult problems, that is, for high noise levels and when only a few observations are available.

Let us analyze the performance of the algorithms in the semi-synthetic case, when the noise level is high (20dB) and the number of observations is low ($n = 20$). This will allow us to validate whether using rational function is beneficial in such situations. We compared the methods to HALS as before, but also to HALS using polynomials or splines presented in [18]. We also considered combining the R-ANLS and the R-HANLS LS methods, to try to obtain a method obtaining the same quality as R-ANLS with speed comparable to R-HANLS LS, and to have thus the best of the two algorithms. When combining these two approaches, we run one of them until the relative residue was below 10^{-2}, and we use the result of this first method as initialization of the second method.

Fig. 8 displays the results. We observe that the R-NMF methods obtain the smallest residues, and are thus best to filter the noise. Among these methods, R-NLS obtains the best SIR, but it is also quite slow despite the small number of observations. R-ANLS and the combination R-ANLS/R-HANLS obtain also good SIR values. Note that the combination is able to obtain accuracy close to the one obtained by R-ANLS but much faster. The objective of combining methods is therefore met in this case. HALS using polynomials or splines also filters well the noise while HALS has more difficulties. However, all methods have difficulties to recover the original signals, as the SIR are low on average for all methods. Fig. 9 shows that when a small number of signals are mixed, $r = 3$, some methods based on rational functions manage to recover a good approximation of the original signals, but when the number of original signal increases, to $r = 5$ or 8, the recovered signals do not really resemble the original ones, as illustrated on Fig. 10. We also observe in this figure that the signals recovered by HALS are highly nonsmooth.

On another hand, changing the degree does not influence the SIR. However, Fig. 11 shows that choosing a too low number of degrees of freedom ($d = 12$) penalizes the algorithms in terms of relative residue, especially when using polynomials or splines. The fact that rational functions already obtain good results for $d = 12$ can be explained by the fact that rational functions are able to express a larger variety of shapes than polynomials or splines for the same number of degrees of freedom. However, this advantage turns into a drawback when the number of degrees of freedom is too high. Indeed, the performances of the methods using rational functions are slightly degraded for larger degrees, because the algorithm starts to model the noise. This is the case in particular for R-HANLS LS and R-ANLS/R-HANLS. Nevertheless, the variability seems to be reduced in this case (the worst case is better than when using a lower number of degrees of freedom).

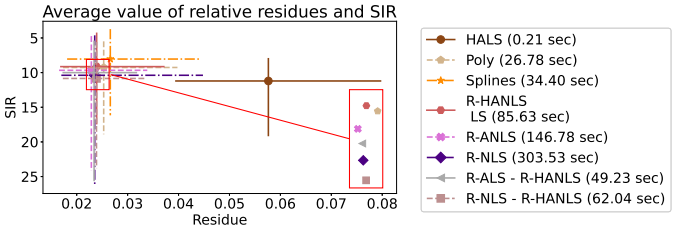

Fig. 8: Summary of performance on semi-synthetic datasets.

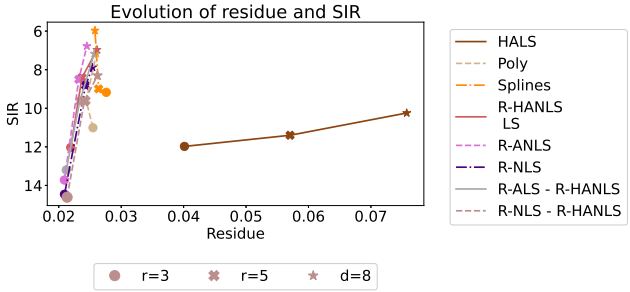

Fig. 9: Performance for varying rank.

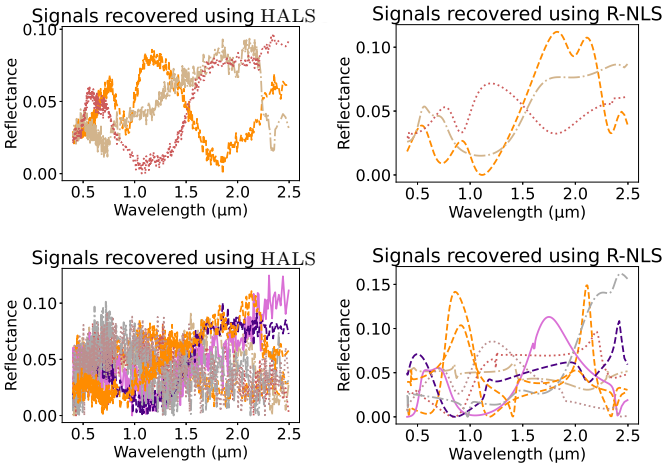

Fig. 10: Example of recovered factor A for $r = 3$ (up) or 8 (down), for HALS (left) or R-NLS (right).

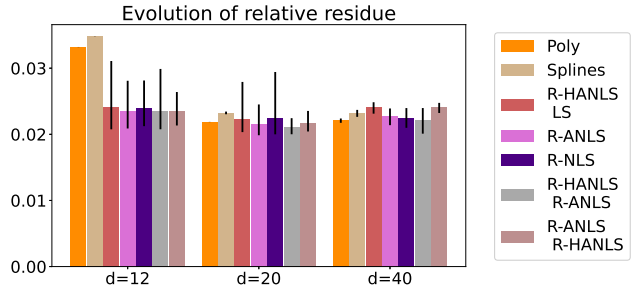

Fig. 11: Performance for varying degree of freedom (d).

F. Using (R-)NMF for classification

We explore the use of R-NMF in the real problem of Indian Pines classification. Classification is performed using the k-nearest-neighbours (KNN) algorithm with $k = 5$. A portion of 70% of the data is used for training.

The data is pre-processed by NMF as follows. Let $Y \in \mathbb{R}^{200\times 21025}$ be the dataset, with 21025 observations of which 6307 should be classified. As the signals are spectra, it can be assumed that they are close to polynomials, splines or rational functions. We approximate Y as $AX^\top$ using NMF, where A contains in its columns sampled functions (note that there is no knowledge of labels at this stage). The columns of X are normalized so that $\|X_{:i}\| = 1$. Then the classification is performed on $X^\top$ instead of Y. The hope is that NMF filters noise in the data, while limiting the number of factors. We use the R-HANLS methods for rational functions due to the high number of observations.

We also considered PCA to do the preprocessing (PCA does not have a nonnegativity constraint). We also tested the method of Debals et al. [10] but the results were not convincing (the accuracy was always below 68%). Perhaps the size of the dataset is too large, or imposing the degrees to be always equal is not appropriate for this approach. Nevertheless, we tested the factorization with rational functions without nonnegativity constraints, using our R-HANLS algorithm, with projection onto rational functions using a least squares solver (Rational). This projection may not be ideal in this case without nonnegativity, but it gives an idea of performance. It also shows that our approach can easily be extended to other sets than the set of nonnegative rational functions. Methods are tested 10 times over different initializations. The number of degrees of freedom is 20, and all methods are limited to 100 seconds. The best factorization for each rank is selected using a K-fold with 5 folds on the 70% of data constituting the training set. As a base line, we use the result of the classification on the whole dataset without preprocessing. It is thus independent of the rank, and corresponds to rank $r = 200$.

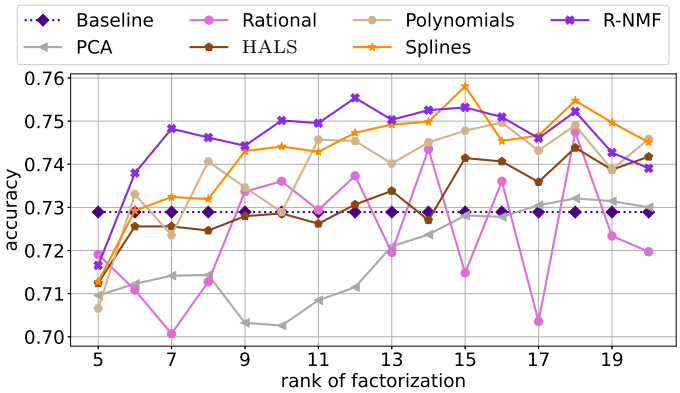

Fig. 12: Accuracy of classification using NMF as preprocessing with various factorization ranks.

Fig. 12 shows the accuracy obtained according to the factorisation rank considered during pre-processing. It confirms the usefulness of R-NMF since this method obtains the best results when $r < 15$. For higher rank values, NMF using splines also obtains very good results, while R-NMF starts to slightly overfit. We also see that imposing nonnegativity makes sense, since PCA and Rational which do not have this constraint obtain the worst results.

Also, using standard NMF improves the baseline only for ranks higher than 15, while using polynomials or splines improves accuracy compared to standard NMF, but to a lesser extent than when using rational functions.

G. Discussion

We observed that R-NMF performs better than other NMF approaches on semi-synthetic data or real-life data. A likely explanation is that, as polynomials and splines, rational functions have less parameters than data points, and hence some form of noise averaging takes place unlike for HALS using vectors. Moreover, they generalize polynomials and splines, and are thus able to express a wider range of shapes, which allows R-NMF to recover more representative signals. On the other hand, the presented methods to compute R-NMF do not obtain very satisfactory results when the data are actually rational functions. Indeed, even when there is no noise, these methods are not always able to recover the original signals and this despite

the fact that the factorization to be recovered is unique, see Section III.

To explain this phenomenon, note that each update of R-ANLS and R-HANLS is not guaranteed to be optimal (as in HALS), and these two methods do thus inexact BCD. But performing inexact BCD was not a problem for polynomials or splines [17], so this might not be the only explanation. Another explanation is that the set of rational functions is not convex, and is not even closed for addition, so there may be many local minima in which the algorithms can get stuck, which also explain why R-NMF is very sensitive to the initialization.

Moreover, R-NMF approaches and especially R-ANLS and R-NLS are more complex and more resources demanding than HALS or NMF using splines or polynomials. One way of investigation to reduce the complexity of the algorithms is to consider other representations of rational functions than fractions of polynomials that could be more accurate, but for which the nonnegativity condition is not trivial, like barycentric representation [12], [30] or sum of fractions [16], which is left for future works.

Furthermore, the methods presented in this paper can be extended to a wider range of rational functions where the numerator and the denominator are not imposed to be nonnegative polynomials, but can be any nonnegative function. To use least-squares based methods, a parametrization of the nonnegativity of the used functions is necessary. If an R-HANLS approach is chosen, the only necessity is that the projection exists. This means, for the Least Squares or the Alternating Least Squares projection, that a parametrization of the nonnegativity of the used functions exists. For Conic projection, a description of the nonnegativity constraint of the used functions must exist (without caring if it is the numerator or the denominator). RKFIT+ requires an operator h' computing the best numerator when the denominator is fixed (possibly neglecting the nonnegativity). Finally, the LinProj projection requires the functions that are used to be linearly parametrizable, in order to keep the problem linear. This highlights the many existing possibilities when performing R-NMF.

VII. Conclusion

We introduced R-NMF, a factorization model using nonnegative rational functions to unmix sampled signals, and presented three approaches to solve the problem. When comparing with standard NMF or with NMF over polynomials or splines, we found that the use of rational functions can outperform existing methods, for synthetic datasets and also for a real life dataset, at the cost of an increase in computational time for large-scale data and greater sensitivity to initialization. This better reconstruction is probably due to the wider range of representation of rational functions. Comparing R-NMF and standard NMF on other applications is a topic of further research.

From our results, it appears that R-HANLS obtains on average worse results than R-ANLS. On an other hand, R-NLS is able to obtain good results on very small problems, but slows down significantly when the problem size increases and has difficulties to converge. Moreover, R-NLS is resource demanding, as is R-ANLS but to a lesser extent. Therefore, we recommend to use R-NLS only for very small problems, when $n < 50$ for example. For medium scale problems, R-ANLS is accurate and not too slow (when $n < 1000$). Finally, for even larger problems, R-HANLS is more appropriate as it is much less computationally demanding. When possible, it should be initialized by a few iterations of R-ANLS to improve performances.

Further works on R-NMF could include in particular developing and improving algorithms, for example by combining the presented methods (see [19]), or by using other representations of rational functions. Note that the proposed methods can in principle be used for rational functions in the broadest sense, that is, for the ratio of two non polynomials functions, which opens the way for more flexible NMF models.

Appendix

We describe the projection methods in more details.

Least Squares: we use the `least_squares` method of python, provided with the jacobian of the cost function, with default parameters. It therefore solves the problem using trust region reflective algorithm. The algorithm is stopped when either the cost function is not enough improved anymore, or the iterates are too close from each others, or the norm of the gradient is very small.

Alternating Least Squares: problem (17) is as a conic problem. Indeed, using Markov-Lukacs theorem, nonnegative polynomials can be expressed using sum of squares of polynomials which can be expressed using positive semidefinite matrices [5]. Therefore, problem (17) can be rewritten using appropriate matrices $V_{\boldsymbol{\tau}}(g)$ a Vandermonde-like matrix taking into account the known denominator, and R the matrix recovering the coefficients of the polynomial from the positive semi-definite matrices. R is built using Gram matrices ([18]). Let $\mathcal{S}_+^d$ be the set of positive semidefinite matrices in $\mathbb{R}^{d\times d}$. We have

$$\min_{(S_1,S_2)\in\mathcal{S}_+^{\frac{d_1}{2}+1}\times\mathcal{S}_+^{\frac{d_1}{2}}} \left|\left| \boldsymbol{z} - V_{\boldsymbol{\tau}}(g) R \begin{bmatrix} vec(S_1) \\ vec(S_2) \end{bmatrix} \right|\right|^2. \tag{23}$$

Problem (23) can be compressed using the singular value decomposition of $V_{\boldsymbol{\tau}}(g) = U\Sigma W^\top$. It can be proved that using $\tilde{V} = \Sigma W^\top$ and $\tilde{\boldsymbol{z}} = U^\top \boldsymbol{z}$ leads to the same minimization problem, to one constant. It is solved using Mosek 9.2 [1]. The problem of finding the best denominator is solved using the same solver as for Least Squares.

Conic: A way to bypass the division difficulty is to consider the modification suggested in [3] on which we add nonnegativity constraints:

$$\min_{h\in\mathcal{P}_+^{d_1,T},\ g\in\mathcal{P}_+^{d_2,T},\ g(\tau_m)=1} \left|\left|\frac{\boldsymbol{z}g(\boldsymbol{\tau})-h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}\right|\right|^2 \tag{24}$$

where $\tilde{g}\in\mathcal{P}_+^{d_2,T}$ is fixed so that $\tilde{g}(\tau_m)=1$. This equation is equivalent to (12) when $g(\boldsymbol{\tau})=\tilde{g}(\boldsymbol{\tau})>0$. It transforms the problem into a simpler problem on polynomials.

The normalisation of g is important to avoid the trivial solution $g=h=0$, and can be done without loss of generality as using αh and αg leads to the same rational function $f=h/g$. Unfortunately, even with normalization, this approach leads to poor reconstruction results, even when input $\boldsymbol{z}$ is exactly a discretization of a nonnegative rational function. We observed that the error is often much smaller on (24) than on (12). For example, suppose that $g(\tau_i)$ and $h(\tau_i)$ are very small and $\tilde{g}(\tau_i)=1$. In this case, $\frac{\boldsymbol{z}_i g(\tau_i)-h(\tau_i)}{\tilde{g}(\tau_i)}$ can be much smaller than $\boldsymbol{z}_i-\frac{h(\tau_i)}{g(\tau_i)}$. Adding a regularization term on the cost function $\lambda\|h(\boldsymbol{\tau})-\tilde{g}(\boldsymbol{\tau})\|^2$ with various $\lambda\geq 0$ allows to reduce the problem but not in a sufficient way. We therefore slightly modify the approach and approximate $\boldsymbol{z}$ by $f(\boldsymbol{\tau})=\frac{h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})+\delta(\boldsymbol{\tau})}$, where $g\in\mathcal{P}_+^{d_1,T}$, $\tilde{g},\delta\in\mathcal{P}_+^{d_2,T}$ and $\tilde{g}$ is fixed. So $\|\boldsymbol{z}-f(\boldsymbol{\tau})\|^2=$

$$\left\|\frac{\boldsymbol{z}\tilde{g}(\boldsymbol{\tau})+\boldsymbol{z}\delta(\boldsymbol{\tau})-h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}\cdot\frac{\tilde{g}(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})+\delta(\boldsymbol{\tau})}\right\|^2. \tag{25}$$

As δ and $\tilde{g}$ are nonnegative, $0<\frac{\tilde{g}(\boldsymbol{\tau})}{\delta(\boldsymbol{\tau})+\tilde{g}(\boldsymbol{\tau})}\leq 1$. The cost function of (26) is thus an upper bound of the cost function of problem (12):

$$\min_{h\in\mathcal{P}_+^{d_1,T},\delta\in\mathcal{P}_+^{d_2,T}} \left|\left|\boldsymbol{z}+\frac{\boldsymbol{z}\delta(\boldsymbol{\tau})-h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}\right|\right|^2. \tag{26}$$

Solving problem (26) ensures to have a rational function that leads also to a low cost in problem (12), which was not the case when solving (24). It can be solved in a similar way as (23). Using appropriate matrices $V_{\boldsymbol{\tau}}(\tilde{g},\boldsymbol{z})$ and Rwe have:

$$\min_{\substack{(S_1,S_2,D_1,D_2)\in\\ \mathcal{S}_+^{\frac{d_1}{2}+1}\times\mathcal{S}_+^{\frac{d_1}{2}}\times\mathcal{S}_+^{\frac{d_2}{2}+1}\times\mathcal{S}_+^{\frac{d_2}{2}}}} \left\|\boldsymbol{z}+V_{\boldsymbol{\tau}}(\tilde{g},\boldsymbol{z})R\begin{bmatrix}vec(S_1)\\vec(S_2)\\vec(D_1)\\vec(D_2)\end{bmatrix}\right\|^2. \tag{27}$$

Problem (27) can be compressed, using the singular value decomposition of $V_{\boldsymbol{\tau}}(\tilde{g},\boldsymbol{z})=U\Sigma W^\top$, with $\tilde{V}=\Sigma W^\top$ and $\tilde{\boldsymbol{z}}=U^\top\boldsymbol{z}$. This problem is solved using Mosek 9.2 solver.

RKFIT+: operator h' from (19) can be solved analytically using matrix V_1 such that $\frac{h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}=V_1\boldsymbol{h}$, where $\boldsymbol{h}$ is the coefficient vector of h. Problem becomes:

$$\frac{h'(\tilde{g},\delta,\boldsymbol{z},\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}=V_1\ \text{argmin}_{\boldsymbol{h}}\left\|\boldsymbol{z}+\frac{\boldsymbol{z}\delta(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}-V_1\boldsymbol{h}\right\|^2. \tag{28}$$

The solution of (28) can be expressed using $V_1^\dagger$ the pseudo-inverse of V_1 as: $\frac{h'(\tilde{g},\delta,\boldsymbol{z},\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}=V_1V_1^\dagger\left(\boldsymbol{z}+\frac{\boldsymbol{z}\delta(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}\right)$.

Similarly, we can define V_2 so that $\frac{\boldsymbol{z}\delta(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})}=V_2\boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is the coefficient vector of δ. Problem (19) is then $\min_{\delta\in\mathcal{P}_+^{d_2,T}}\|(I-V_1V_1^\dagger)(\boldsymbol{z}+V_2\boldsymbol{\delta})\|^2$. This problem can be compressed, using SVD decomposition of $(I-V_1V_1^\dagger)V_2$: $U\Sigma W^\top$. The cost becomes $\|U^\top(I-V_1V_1^\dagger)\boldsymbol{z}+\Sigma W^\top\boldsymbol{\delta}\|^2$. The problem can then be solved using Mosek 9.2.

LinProj: this problem is solved using Mosek 9.2. This solver sometimes consider a problem as feasible when the constraint is violated by a value smaller than 10^{-6}. To avoid this small violation to lead to a huge value of $\max_i\left(\left|\boldsymbol{z}_i-\frac{h(\tau_i)}{g(\tau_i)}\right|\right)$, $g(\boldsymbol{\tau})$ is imposed to be greater than 1.

References

[1] M. ApS. *The MOSEK optimization toolbox for Python manual. Version 9.3.*, 2021.

[2] D. Backenroth. *Methods in functional data analysis and functional genomics.* Columbia University, 2018.

[3] I. Barrodale and J. Mason. Two simple algorithms for discrete rational approximation. *MATHEMATICS of computation*, 24(112):877–891, 1970.

[4] M. Berljafa and S. Güttel. The RKFIT algorithm for nonlinear rational approximation. *SIAM Journal on Scientific Computing*, 39(5):A2049–A2071, 2017.

[5] G. Blekherman, P. A. Parrilo, and R. R. Thomas. *Semidefinite optimization and convex algebraic geometry.* SIAM, 2012.

[6] M. A. Branch, T. F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

[7] A. Cichocki, R. Zdunek, and S.-i. Amari. Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. In *International Conference on Independent Component Analysis and Signal Separation*, pages 169–176. Springer, 2007.

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

[9] S. Costa and L. N. Trefethen. AAA-least squares rational approximation and solution of laplace problems. *arXiv preprint arXiv:2107.01574*, 2021.

[10] O. Debals, M. Van Barel, and L. De Lathauwer. Löwner-based blind signal separation of rational functions with applications. *IEEE Transactions on Signal Processing*, 64(8):1909–1918, 2015.

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

[12] S.-I. Filip, Y. Nakatsukasa, L. N. Trefethen, and B. Beckermann. Rational minimax approximation via adaptive barycentric representations. *SIAM Journal on Scientific Computing*, 40(4):A2427–A2455, 2018.

[13] X. Fu, K. Huang, N. D. Sidiropoulos, and W.-K. Ma. Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications. *IEEE Signal Processing Magazine*, 36(2):59–80, 2019.

[14] N. Gillis. *Nonnegative Matrix Factorization.* SIAM, Philadelphia, 2020.

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

[16] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Transactions on power delivery*, 14(3):1052–1061, 1999.

[17] C. Hautecoeur and F. Glineur. Accelerating nonnegative matrix factorization over polynomial signals with faster projections. In *2019 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019.

[18] C. Hautecoeur and F. Glineur. Nonnegative matrix factorization over continuous signals using parametrizable functions. *Neurocomputing*, 2020.

[19] C. Hautecoeur and F. Glineur. Factorisation nonnégative avec des fonctions rationnelles: partitions efficaces et méthodes hybrides. *Groupe de Recherche et d'Etudes du Traitement du Signal et des Images (GRETSI 2022), pages*, pages 929–932, 2022.

[20] C. Hautecoeur, F. Glineur, and L. De Lathauwer. Hierarchical alternating nonlinear least squares for nonnegative matrix factorization using rational functions. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 1045–1049. IEEE, 2021.

[21] J. M. Hokanson and C. C. Magruder. Least squares rational approximation. *arXiv preprint arXiv:1811.12590*, 2018.

[22] A. Ionita. *Lagrange rational interpolation and its applications to approximation of large-scale dynamical systems*. PhD thesis, Rice University, 2013.

[23] D. Jibetean and E. de Klerk. Global optimization of rational functions: a semidefinite programming approach. *Mathematical Programming*, 106(1):93, 2006.

[24] U. Khristenko and B. Wohlmuth. Solving time-fractional differential equation via rational approximation. *arXiv preprint arXiv:2102.05139*, 2021.

[25] H. Kim and 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):713–730, 2008.

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

[27] D. D. Lee and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401(6755):788, 1999.

[28] C.-J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

[29] H. L. Loeb. *On rational fraction approximations at discrete points*. PhD thesis, Columbia University, 1959.

[30] Y. Nakatsukasa, O. Sète, and L. N. Trefethen. The AAA algorithm for rational approximation. *SIAM Journal on Scientific Computing*, 40(3):A1494–A1522, 2018.

[31] V. Powers and B. Reznick. Polynomials that are positive on an interval. *Transactions of the American Mathematical Society*, 352(10):4677–4692, 2000.

[32] T. Roh and L. Vandenberghe. Discrete transforms, semidefinite programming, and sum-of-squares representations of nonnegative polynomials. *SIAM Journal on Optimization*, 16(4):939–964, 2006.

[33] C. Sanathanan and J. Koerner. Transfer function synthesis as a ratio of two complex polynomials. *IEEE transactions on automatic control*, 8(1):56–58, 1963.

[34] A. Siem, E. de Klerk, and D. den Hertog. Discrete least-norm approximation by nonnegative (trigonometric) polynomials and rational functions. *Structural and Multidisciplinary Optimization*, 35(4):327–339, 2008.

[35] L. N. Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

[36] L. N. Trefethen, Y. Nakatsukasa, and J. Weideman. Exponential node clustering at singularities for rational approximation, quadrature, and pdes. *Numerische Mathematik*, 147(1):227–254, 2021.

[37] L. Wittmeyer. Rational approximation of empirical functions. *BIT Numerical Mathematics*, 2(1):53–60, 1962.

[38] R. Zdunek. Approximation of feature vectors in nonnegative matrix factorization with gaussian radial basis functions. In *International Conference on Neural Information Processing*, pages 616–623. Springer, 2012.

[39] 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)*, pages 1–6. IEEE, 2014.

[40] R. Zdunek, A. Cichocki, and T. Yokota. B-spline smoothing of feature vectors in nonnegative matrix factorization. In *International Conference on Artificial Intelligence and Soft Computing*, pages 72–81. Springer, 2014.