UCLouvain
Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Nonnegative Matrix Factorization using Parametrizable Functions

Cécile Hautecoeur

Thesis submitted in partial fulfillment
of the requirements for the degree of
Doctorat en sciences de l'ingénieur et technologie

Dissertation committee:

Prof. François Glineur (UCLouvain, advisor)
Prof. Pierre-Antoine Absil (UCLouvain, Belgium)
Prof. Lieven De Lathauwer (KULeuven, Belgium)
Prof. Nicolas Gillis (UMons, Belgium)
Dr. Le Thi Khanh Hien (Huawei, Belgium)
Prof. Rafał Zdunek (Politechnika Wrocławska, Poland)
Prof. Philippe Lefèvre (UCLouvain, chair)

Version of September 23, 2022.

Abstract

Nonnegative Matrix Factorization (NMF) is a popular data analysis tool for nonnegative data, able to extract meaningful features from a dataset, compress it and filter its noise. To do so, this method factorizes an input matrix $\boldsymbol{Y}$ as the product of two factors, the low-rank nonnegative matrices $\boldsymbol{A}$ and $\boldsymbol{X}$. Recently, several interesting works have analyzed the possibility of imposing some additional structure to the NMF model, by requiring factors $\boldsymbol{A}$ and $\boldsymbol{X}$ to be based on nonnegative parametrizable functions, such as nonnegative splines or polynomials. When the input data is structured, for example for smooth continuous signals, this leads to factorizations that are less sensitive to noise and better able to find the characteristic elements of the dataset.

This thesis is a continuation of that work. We first study NMF using polynomials and splines. We focus on the Hierarchical Alternating Least Squares (HALS) algorithm, which has the advantage of quickly finding a good solution, and converging to a stationary point under mild conditions. We generalize this algorithm and observe that it obtains good results and is competitive with existing approaches.

We also seek to generalize the NMF model in a more formal way, so that any parameterizable function could be used in the factors. This leads to the definition of the H-NMF (NMF on Hilbert spaces), allowing to factorize a two-variable function (such as a matrix) as the sum of a small number of products of one-variable functions (the characteristic elements). This generalization is proved to have characteristics and theoretical guarantees similar to those of the standard NMF problem.

Our generalization of HALS uses projections onto the functions under con-

sideration (e.g. nonnegative polynomial or splines). As these projections can be costly and difficult to compute, we consider to approximate them. We are able to identify several cases where the algorithm keeps good convergence properties, and use these results to describe heuristic projections speeding up the algorithm without affecting its accuracy.

Finally, we consider the case of rational functions which has no theoretical guarantees because rational functions of fixed degree do not form a convex set. In practice, we observe that our algorithms are very sensitive to their initialization in this case. Nevertheless, using rational functions in NMF can lead to very good results since rational functions are able to represent a wider range of shapes than polynomials and splines.

Acknowledgements

Achieving a thesis is a rich learning experience, which has allowed me to develop both from a professional/scientific and personal point of view. I would like to thank my supervisor François Glineur without whom none of this would have been possible. François always had the art of pointing out the right changes to be made, requiring little work but significantly improving the quality. He also always found the right ways to help me when I was having difficulties. I always felt very supported and encouraged.

I would also like to thank Stephen Vavasis, who mentored me for 3 months during my stay in Waterloo (Canada), it was a very enriching experience. I also thank the members of my thesis committee and jury for their careful reading and valuable comments which undoubtedly improved the quality of this work.

I would also like to thank my colleagues at the Euler, for the superb working atmosphere. In particular, my office-mate Loïc, for all our discussions, about our respective theses or not, and Hazan, for his great kindness, as well as my AsCII co-president, Charles, with whom we did our best to contribute to (and improve?) the life of the institute. There is also Emilie who was my guide during my first years of thesis. Not forgetting all the Œulerians (Adrien, Benjamin, Estelle, Guillaume O., Guillaume B., Nicolas and Sébastien), the super secretaries Marie-Christine and Pascale, the members of the AsCII, and in general all the people of the Euler and of the ICTEAM institute.

I would also like to thank my friends who have supported me for many years (15 years for most of them). Of course, I warmly thank my family who always supports me in everything I do.

Finally, I would like to thank Martin, who has always been there during these four years, enthusiastic about my successes and very present and motivating during my moments of doubt. Thank you for all that you have given me, and will give me for (I hope) many years to come.

Contents

Useful notations

Matrices

$\boldsymbol{A}$	Matrices are represented with bold capital letters
$\boldsymbol{A}_{i,j}$	Element at i^{th} row and j^{th} column of $\boldsymbol{A}$. Indexing starts at 1.
$\boldsymbol{A}_{i:}$	i^{th} row of $\boldsymbol{A}$
$\boldsymbol{A}_{:j}$	j^{th} column of $\boldsymbol{A}$
$\boldsymbol{A}^\top$	Transpose of matrix $\boldsymbol{A}$
$\|\cdot\|_F$	Frobenius norm of a matrix
$\|\cdot\|_2$	Euclidean norm of a vector
$\mathbb{R}^{m\times n}$	Set of real matrices of size $m \times n$
$\mathbb{R}^r$	Set of real vectors of size r
I^r	Identity matrix of size $r \times r$
$0^{m\times n}$	Zero matrix of size $m \times n$
$\mathbb{1}^r$	Vector of ones in $\mathbb{R}^r$
$\boldsymbol{V}_{\boldsymbol{\tau}}^d$	Pseudo-Vandermonde matrix in Chebyshev basis, of degree d, on discretization points $\boldsymbol{\tau}$
$\text{vec}(\boldsymbol{A})$	Vectorization of matrix $\boldsymbol{A}$

Hilbert spaces

$\mathcal{H}$	Hilbert spaces are represented with calligraphic letters
$\langle\cdot,\cdot\rangle_{\mathcal{H}}$	Inner product of $\mathcal{H}$
$\|\cdot\|_{\mathcal{H}}$	Norm of $\mathcal{H}$

Sets and operators

$\lvert \cdot \rvert$	Absolute value
$\lceil \cdot \rceil$	Ceiling function
$\#_S$	Cardinality of set S, i.e. the number of elements in S
$\mathcal{A} \setminus \mathcal{B}$	Exclusion operator, i.e. $\mathcal{A}$ after removing the elements from $\mathcal{B}$
$[\cdot]_S$	Projection on set S
$[\cdot]_+$	Projection on nonnegative vectors (equivalent to a thresholding operation, setting all negative values to 0)
$\mathcal{P}^d$	Set of polynomials of degree d
$\mathcal{P}^d_+$	Set of nonnegative polynomials of degree d
$\mathcal{P}^d_+(I)$	Set of polynomials of degree d nonnegative on interval I
$\mathbb{R}$	Real line
$\mathbb{R}_+$	Nonnegative real line
$\mathbb{S}_+$	Set of positive semidefinite matrices
$\mathbb{L}$	Lorentz cone
$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and variance σ

Acronyms

BCD	Block Coordinate Descent
GRBF	Gaussian Radial Basis Functions
HALS	Hierarchical Alternating Least Squares
H-HALS	Hierarchical Alternating Least Squares in Hilbert spaces
H-NMF	Nonnegative Matrix Factorization in Hilbert spaces
NMF	Nonnegative Matrix Factorization
PCA	Principal Component Analysis
RKHS	Reproducing Kernel Hilbert Space
SDP	SemiDefinite Program
SIR	Signal to Interference Ratio
SNR	Signal to Noise Ratio
SOS	Sum Of Squares
SVD	Singular Value Decomposition

List of algorithms

1

Introduction

Humans have always tried to understand the world around them by drawing conclusions from observations, with varying degrees of success. However, when observations are composed of many features, they are difficult to represent and visualize, and therefore difficult to understand and analyze. Moreover, all features may not be necessary for the aimed task, either because they are irrelevant or because the information they provide is already provided by other features.

Nowadays, with the development of computer sciences, large data with many features can be handled by computers using machine learning algorithms, that are able to process much larger amounts of information than humans. Nevertheless, having too many features also disturbs machine learning algorithms, and decreases their accuracy to perform classification or clustering tasks, for example. Moreover, increasing the number of features slows down the algorithms and is more resource-demanding in terms of needed memory and/or computational power. Consequently, reducing the number of features in a dataset of observations, or being able to describe a complex dataset with a low number of features, without reducing the provided information, is an important challenge in engineering.

To reduce the number of features, and thus the dimension of a dataset, it is natural to identify and eliminate the less useful features for the task we aim to perform. This is called **feature selection**. Feature selection methods

aim at identifying a set of features that provides a large quantity of useful information while being of reduced size. The two main challenges of feature selection are thus to correctly evaluate the quantity and the quality of information provided by a set of features, and to smartly select the subsets of features to evaluate. Indeed, evaluating all possible associations of features is most of the time way too costly.

Two main approaches have been considered to evaluate the quantity and quality of information provided by a set of features. On the one hand, **filters** use statistical tools to determine how the features are related to the objective and/or between each other. On the other hand, **wrappers** apply a learning algorithm to the selected set of features and use the obtained accuracy to determine if this set of features provides useful information or not. A last approach, that can be linked to wrappers, consists of using embedded methods that select the features during a learning process. Regarding the selection of the sets of features to be evaluated, one can do local search, starting from a set of features and browsing the most promising nearby set of features, as in Branch and Bound [95]. Alternatively, the sets of features can be browsed randomly, which is referred to as the Las Vegas algorithm. Interested readers can read the survey on feature selection of Kumar and Minz [78] and the references therein for more details about feature selection methods.

Many feature selection methods are specific to the target task, for example they use some labeled data to select features for a classification task. We say that they are in this case **supervised** methods, in contradiction with **unsupervised** methods that are independent from the target task. These last years, many works have explored the possibility to do unsupervised feature selection, mainly using filters, as wrappers have an intrinsic bias from the machine learning method they use to evaluate features; see for example [115] and the references therein for more information about unsupervised feature selection. This can be hard, as some features can be primordial for certain tasks and useless for others. In general, being limited to existing features restricts the possibilities to express accurately a dataset in a compact way [129]. To avoid this problem, one can be interested in **feature extraction**.

In feature extraction methods, the goal is to describe the original dataset using a small set of new features that are different from the original ones. Feature extraction methods can be supervised, like the Linear Discriminant

Analysis (LDA). But in most cases feature extraction is unsupervised, i.e. independent from the target task. Therefore, the same feature extraction can be used for different tasks. Those methods can either be linear or not. Examples of nonlinear feature extraction methods, able to model complex relationship in data, are Kernel Principal Component Analysis (KPCA) and AutoEncoders. Linear feature extraction methods, also called linear dimension reduction techniques, linearly map the original data to a lower dimensional feature space. The results of those methods are in general the easiest to interpret for human users thanks to linearity. Examples of linear feature extraction methods are Principal Component Analysis (PCA), Singular Value Decomposition (SVD) or Wavelet Transforms (WT). Other examples can be found in [5].

When data is by nature nonnegative, for example when it contains occurrences (e.g. word counts), intensities, reflectance, ratings, etc., it is worth wondering if it can be well described as nonnegative linear combinations of nonnegative features. This means that the data can be expressed as nonnegative weighted sums of a few nonnegative elements. This is quite often true in practice. This property can be explicit. For example, when data consists of hyperspectral images, each pixel of the image contains a spectrum that is the weighted sum of the spectra of the endmembers present in the pixel [100], or when data consists of chemical spectral signatures, they are the weighted sum of the spectral signatures of the pure species composing them [18], or when data consists of musics, they are the sum of the instruments composing them [81]. That property of nonnegative representation can also be implicit. For example, in recommendation systems, that suggest films or other items to their users based on their previous choices, the behavior of a user can often be explained as the nonnegative combination of the behaviors of some fictive typical users [90]. Other examples can be found in [13, 45, 93] and the references therein.

When data can be described as nonnegative linear combinations of nonnegative factors, it is recommended to use Nonnegative Matrix Factorization (NMF). NMF aims at describing an input data matrix $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$ as the product of two nonnegative factors $\boldsymbol{A} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{X} \in \mathbb{R}^{n \times r}$: $\boldsymbol{Y} \simeq \boldsymbol{A}\boldsymbol{X}^\top$. The rank r of the factorization is chosen to be much smaller than m and n, i.e. $r \ll m, n$. Suppose without loss of generality that $\boldsymbol{Y}$ contains n observations of m features. The second factor $\boldsymbol{X}$, named the mixing matrix, describes in each of its n rows the n original data using only r features, reducing thus the dimension of the original data $\boldsymbol{Y}$. Moreover, the first factor

$\boldsymbol{A}$ contains the r features extracted by the NMF in each of its columns.

Most of the time it is not possible to recover factors $\boldsymbol{A}$ and $\boldsymbol{X}$ that describe $\boldsymbol{Y}$ exactly, due to noise. If data does not contain noise, we can do Exact-NMF [122] to recover $\boldsymbol{A}$ and $\boldsymbol{X}$ so that $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}^\top$. Otherwise, we aim at finding $\boldsymbol{A}$ and $\boldsymbol{X}$ so that the difference between $\boldsymbol{Y}$ and $\boldsymbol{A}\boldsymbol{X}^\top$ corresponds to the noise. This means that ideally the low rank product $\boldsymbol{A}\boldsymbol{X}^\top$ is expected to be a better representation of the data than $\boldsymbol{Y}$, as it filtered the noise out.

Imposing $\boldsymbol{A}$ and $\boldsymbol{X}$ to be nonnegative is actually a way of being less sensitive to noise and more interpretable than PCA for example. Indeed, adding a constraint limits the risks of modeling the noise. Moreover, the nonnegativity of $\boldsymbol{X}$ imposes an additive description of the data, which leads to a part-based description of the data and is therefore easier to interpret for users [80]. NMF has proven its efficiency in many areas of data analysis and is now a commonly used method when data is nonnegative (see for example [46] and the references therein).

As imposing nonnegativity has a good impact on the factorization, it is natural to wonder if imposing a certain additional structure to the factorization can improve even more the performance. For this purpose, we can regularize the factorization. This means adding a term to the objective we aim to optimize. This new term in the objective penalizes the factors $\boldsymbol{A}$, $\boldsymbol{X}$ and/or the products $\boldsymbol{A}\boldsymbol{X}^\top$ that do not satisfy the wanted structure (for example sparsity or smoothness). We can also decide to force the factors $\boldsymbol{A}$ and/or $\boldsymbol{X}$ to have a certain structure, which is stricter than the previous approach.

In this work, we consider the second approach, and we impose the factors $\boldsymbol{A}$ and/or $\boldsymbol{X}$ to come from parametrizable functions. This is a particularly good choice when data contains (observations of) continuous functions, for example the evolution of a parameter with respect to time, or with respect to the wavelength (like in hyperspectral images). Most of the time, data collected this way contains samples of signals, and the use of parametrizable functions in the factors allows taking into consideration the fact that the signals still evolve outside the sampling points. On the other hand, even when data does not explicitly come from functions, it can sometimes be well approximated by functions. For example, smooth data can most of the time be well described by splines. Nevertheless, this still implies to have some sort of implied topology in the collected features, as it is neces-

sary to have some notion of neighborhood to talk about smoothness.

We analyze and implement algorithms for NMF using parametrizable functions in $\boldsymbol{A}$ and/or $\boldsymbol{X}$ in three specific cases; namely when the considered functions are polynomials, splines, or rational functions. Moreover, we formalize the necessary theory and conditions to use other parametrizable functions and to solve the problem directly on functions, without requiring sampling. We therefore extend the NMF to two-dimensional functions instead of matrices. Hereafter is a list of the main objectives of this thesis, followed by a quick introduction to the various chapters making up this thesis, as well as the publications linked to each chapter.

1.1 Objectives of the thesis

The three main objectives tackled in this thesis are the following:

- Analyze the possibility of using parametrizable functions in NMF from a theoretical point of view, and determine the needed conditions to use a given class of parametrizable functions.
- Develop efficient algorithms to solve NMF using parametrizable functions. It is hoped that these algorithms can improve the accuracy of factorizations, while keeping the execution time under control.
- Implement and use the developed algorithms in real-world applications to verify their utility in practice.

1.2 Content of the thesis and publications

Chapter 2: Preliminaries and background

Why? Before going any further, it is good to review the basics and make sure we are on the same page. It is also interesting to take a quick look at the literature to see the context of our work.

How? This chapter presents formally the standard NMF and provides a quick overview of its existing improvements. It also provides useful information about the three nonnegative parametrizable functions that we

are interested in, namely nonnegative polynomials, splines, and rational functions.

Chapter 3: NMF using Polynomials and Splines

Why? To find out whether the use of parameterizable functions in NMF can be beneficial, we first analyze the case of two rather simple functions, namely polynomials and splines. These functions have the advantage of being linearly parametrizable (they can be expressed as a linear combination of a small number of basis elements), and have already been studied briefly in the literature in the context of NMF.

How? We study the possibility of extending the use of Hierarchical Alternating Least Squares (HALS) to polynomials or splines. We observe that this extension can be done easily but requires to perform a projection on the set of nonnegative polynomials or splines. Fortunately, projection on these two sets is possible and described in the chapter. We observe that NMF using polynomials or splines can be very accurate, and that using an extension of HALS leads to faster algorithms than those presented in previous works.

Associated publications

[54] C. Hautecoeur and F. Glineur. **Nonnegative matrix factorization with polynomial signals via hierarchical alternating least squares**. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 125–130, 2019.

[56] C. Hautecoeur and F. Glineur. **Nonnegative matrix factorization over continuous signals using parametrizable functions**. *Neurocomputing*, 416: 256–265, 2020.

Chapter 4: Extending NMF to Hilbert spaces (H-NMF)

Why? Being able to extend NMF to polynomials and splines does not mean being able to extend it to any parametrizable function. This chapter presents the requirements for using certain types of function, as well as algorithms for solving the extended NMF problem.

How? We analyze the problem theoretically, assuming that functions belong to Hilbert spaces. Using Hilbert spaces ensures the existence of a

norm, and therefore a distance, to quantify the quality of the factorization. We also present how to extend the classical algorithms used in NMF to Hilbert spaces.

Associated publication

[58] C. Hautecoeur and F. Glineur. **H-NMF: nonnegative and constrained matrix factorization on Hilbert spaces; A unifying framework for NMF on signals.** Submitted to SIAM Journal on Optimization (SIOPT)

Chapter 5: Convergence of H-NMF

Why? To analyze under which conditions some of the presented algorithms are able to find a good solution of the H-NMF problem, with a focus on the generalization of the HALS algorithm. This algorithm requires a projection step that can be difficult. We therefore are also interested in its behavior when the projection step is not done exactly.

How? We analyze the ability of the algorithms to converge to a stationary point. As the NMF problem is not convex, even in its standard form, it is difficult to rely on a better criterion for the quality of the solution than stationarity. We observe that the needed conditions to guarantee convergence are similar as for standard NMF.

Associated publication

[58] C. Hautecoeur and F. Glineur. **H-NMF: nonnegative and constrained matrix factorization on Hilbert spaces; A unifying framework for NMF on signals.** Submitted to SIAM Journal on Optimization (SIOPT)

Chapter 6: Accelerating NMF using polynomials via inexact projections

Why? As it is possible to prove that our generalization of the HALS algorithm converges under certain conditions when the projection is not performed exactly, we wonder if it is possible to speed up the algorithm by using inexact projections.

How? We consider the case of NMF using polynomials, and define several inexact projections more or less heuristically, keeping in mind the con-

vergence results obtained in previous chapter. This allows us to observe that it is indeed possible to define heuristic projections with theoretical foundations that speed up the algorithm using polynomials without altering its accuracy.

Associated publication

[53] C. Hautecoeur and F. Glineur. **Accelerating nonnegative matrix factorization over polynomial signals with faster projections**. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019.

This chapter also contains the findings from a research visit to the University of Waterloo, in the team of Professor Stephen Vavasis, from October to December 2021.

Chapter 7: Application: Using NMF with splines for image completion

Why? Results presented in previous chapters were obtained on synthetic data. This chapter checks the effectiveness of our methods on a real-world problem.

How? We use NMF with splines for image completion and observe the efficiency of this approach that uses the smoothness of splines to obtain more accurate images.

Associated publication

[55] C. Hautecoeur and F. Glineur. **Image completion via nonnegative matrix factorization using HALS and B-splines**. In *28th European Symposium on Artificial Neural Networks-Computational Intelligence and Machine Learning (ESANN)*, pages 73-78, 2020.

Chapter 8: NMF using rational functions

Why? The theory and applications presented in the previous chapters only apply when the chosen set of parametrizable functions is convex. In this chapter we observe what happens when this is no longer the case.

How? We focus on the case of nonnegative rational functions of fixed degree. We observe that the loss of convexity makes the results more variable

and more sensitive to initialization, even though they are good on average. Therefore, it is possible to take advantage of the use of rational functions, which are able to model a large number of shapes.

Associated publications

[59] 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.

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

[52] C. Hautecoeur, L. De Lathauwer, N. Gillis, and F. Glineur **Least-squares methods for nonnegative matrix factorization over rational functions**. Submitted to IEEE Transactions on Signal Processing.

Figure 1.1 here after contains a summary of the implications between the different chapters.

For all chapters, some details about implementation can be found in Appendix C, and code is available on Code Ocean: `https://codeocean.com/capsule/5065386/tree`.

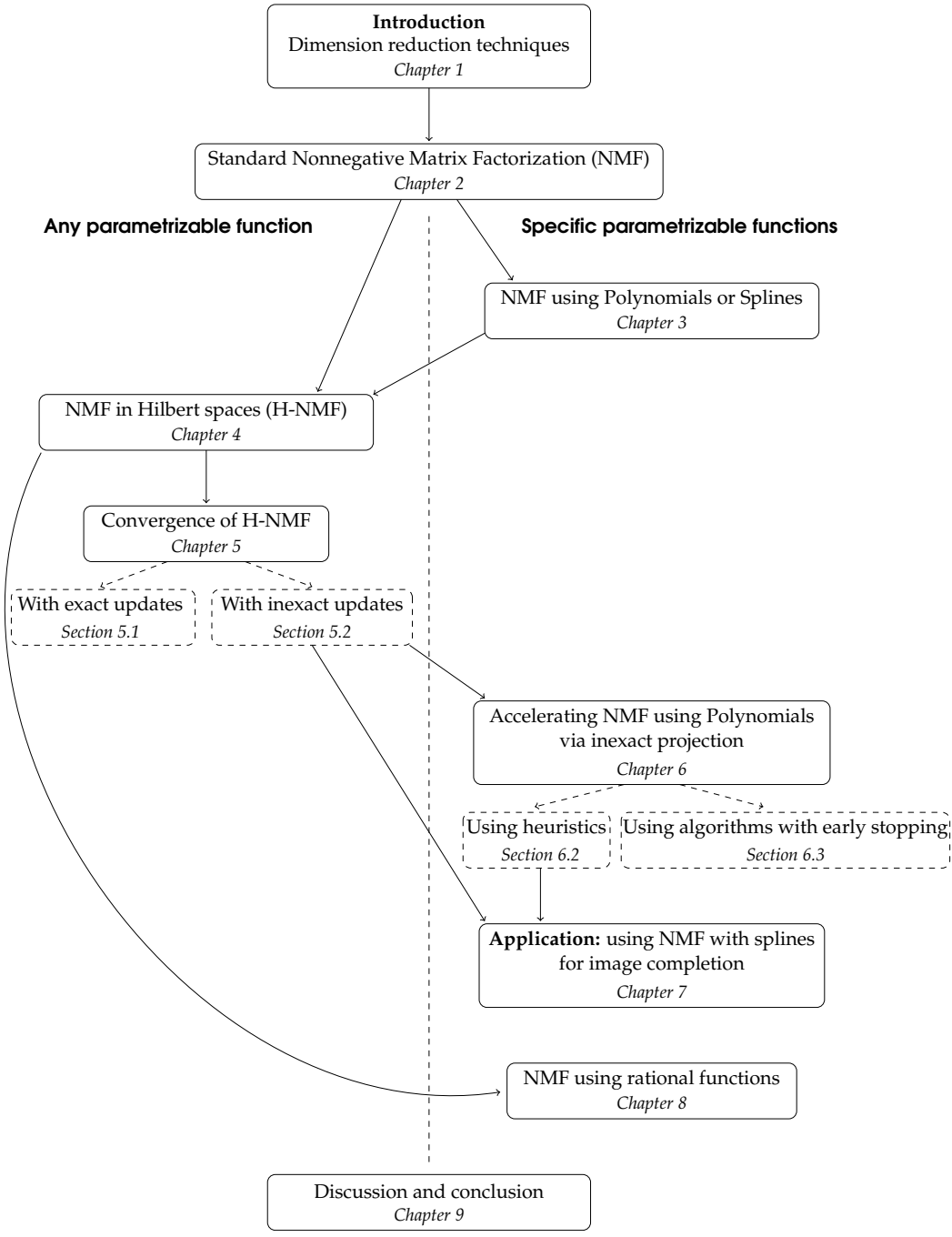


Fig. 1.1 Structure of the thesis

2

Preliminaries and background

2.1 Nonnegative Matrix Factorization (NMF)

Nonnegative matrix factorization (NMF) is a powerful tool, commonly used to analyze nonnegative data. This method is able to compress data, filter noise, and extract meaningful features and their coefficients all at once. NMF, introduced in [97], became popular thanks to [80]. It expresses an input data matrix $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$ as the product of two nonnegative matrices, the factors $\boldsymbol{A} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{X} \in \mathbb{R}^{n \times r}$: $\boldsymbol{Y} \simeq \boldsymbol{A}\boldsymbol{X}^\top$. The rank r is supposed to be small, and in particular $r < \min(m, n)$. Therefore, NMF is a linear dimension reduction technique, that compresses input data. Another way to interpret NMF is to say that each column of $\boldsymbol{Y}$, denoted $\boldsymbol{Y}_{:j}$, is expressed as a nonnegative linear combination of a few nonnegative features, the columns of $\boldsymbol{A}$: $\boldsymbol{Y}_{:j} = \sum_{k=1}^{r} \boldsymbol{A}_{:k}\boldsymbol{X}_{jk}$. The coefficients of the combination are contained in the rows of $\boldsymbol{X}$. This problem is illustrated in Figure 2.1.

Trying to factorize $\boldsymbol{Y}$ exactly is named "Exact NMF", but most of the time this is not possible, as data contains noise. Therefore, several cost functions can be considered to evaluate the quality of the factorization. Some

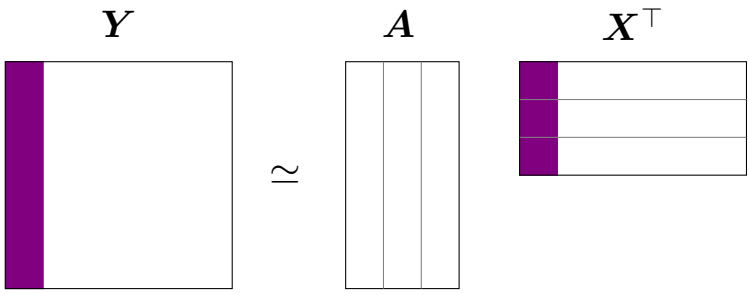


Fig. 2.1 Illustration of the NMF problem.

of them are described in [28]. Here we focus on the Frobenius norm of the reconstruction error. This leads to the definition of the NMF problem.

Definition 2.1. Nonnegative Matrix Factorization Given an input matrix $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$ and a factorization rank r, find nonnegative matrices $\boldsymbol{A} \in \mathbb{R}_+^{m \times r}$ and $\boldsymbol{X} \in \mathbb{R}_+^{n \times r}$ minimizing

$$\underset{\boldsymbol{A} \in \mathbb{R}_+^{m \times r},\, \boldsymbol{X} \in \mathbb{R}_+^{n \times r}}{\operatorname{argmin}} \left\| \boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}^\top \right\|_F^2 . \tag{2.1}$$

NMF is a non-convex problem and is NP-Hard [124]. Nevertheless, several algorithms have been developed to solve it. Some of them are briefly presented in Section 4.3. Let us present here the commonly used Hierarchical Alternating Least Squares algorithm (HALS) [27, 39, 71]. The idea of HALS is to decompose the problem in $2r$ blocks: the columns of $\boldsymbol{A}$ and $\boldsymbol{X}$. Then the problem is iteratively optimized on each block successively while considering the other blocks as fixed. HALS is therefore a block coordinate descent (BCD) algorithm. This algorithm is sketched in Algorithm 2.1, with $[\cdot]_+$ an operator projecting onto the set of nonnegative vectors, i.e. thresholding negative values to 0. An advantage of this algorithm is that equation (2.1) can be solved analytically when all blocks but one are fixed using equation (2.2), which makes the iterations very fast.

2.1.1 Evaluating the quality of a factorization

The Frobenius norm of the reconstruction error in (2.1) enforces the factorization to not deviate much from the original data. This is a good cost function to optimize when only $\boldsymbol{Y}$ is known, but it is not really a good way

Algorithm 2.1 Hierarchical Alternating Least Squares (HALS)

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

  function UPDATEHALS($\boldsymbol{Y}$, $\boldsymbol{A}$, $\boldsymbol{X}$)
    for $\boldsymbol{A}_{:k}$ in $\boldsymbol{A}$ **do**

$$\boldsymbol{A}_{:k} \leftarrow \left[\frac{\boldsymbol{Y}\boldsymbol{X}_{:k} - \sum_{s \neq k} \boldsymbol{A}_{:s} (\boldsymbol{X}_{:s})^\top \boldsymbol{X}_{:k}}{\|\boldsymbol{X}_{:k}\|^2} \right]_+ \tag{2.2}$$

    return $\boldsymbol{A}$ ▷ $\mathcal{O}(mnr)$ flops

to evaluate the quality of a factorization. Indeed, it does not highlight if the noise has been filtered or not, and neither evaluates if the recovered factors are close or not from the original ones.

Let us suppose that the ground truth is known, i.e. the matrices $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{X}}$ from which data $\boldsymbol{Y}$ has been created: $\boldsymbol{Y} = \bar{\boldsymbol{A}}\bar{\boldsymbol{X}}^\top + \boldsymbol{N}$, with $\boldsymbol{N}$ the noise. It is then possible to better evaluate the quality of a factorization, in the following different ways.

- **Ability to filter the noise:** the ability to filter the noise of found factors $\boldsymbol{A}$,$\boldsymbol{X}$ can be evaluated through the relative error compared to $\bar{\boldsymbol{A}}\bar{\boldsymbol{X}}^\top$, that is not known during the factorization process. We denote this relative error as the relative residual, to distinguish it from the error with respect to $\boldsymbol{Y}$ computed in the cost function. The lower the relative residual is, the better the noise is filtered.

$$\text{Relative residual} = \frac{\|\boldsymbol{A}\boldsymbol{X}^\top - \bar{\boldsymbol{A}}\bar{\boldsymbol{X}}^\top\|_F}{\|\bar{\boldsymbol{A}}\bar{\boldsymbol{X}}^\top\|_F} \tag{2.3}$$

- **Quality of factors:** to evaluate the quality of found factors $\boldsymbol{A}$,$\boldsymbol{X}$, it is important to take into consideration the fact that a factorization $\boldsymbol{A}\boldsymbol{X}^\top$ is insensitive to permutation and rescaling of the columns of $\boldsymbol{A}$,$\boldsymbol{X}$. Indeed, if the columns of $\boldsymbol{A}$ and $\boldsymbol{X}$ are permuted the same way, and column i of $\boldsymbol{A}$ is multiplied by $d_i > 0$ while column i of $\boldsymbol{X}$ is multiplied by $\frac{1}{d_i}$, the product $\boldsymbol{A}\boldsymbol{X}^\top$ stays unchanged. Let $\boldsymbol{P}$ be a permutation matrix, i.e. a matrix with exactly one element equal to 1

in each of its rows and columns, and all other elements equals to 0, and $\boldsymbol{D}$ be a diagonal matrix with positive entries. If $\boldsymbol{Q} = \boldsymbol{PD}$, then $\boldsymbol{Q}^{-\top} = \boldsymbol{PD}^{-1}$, and we have $\boldsymbol{APD} = \tilde{\boldsymbol{A}} \in \mathbb{R}_+^{m\times r}$ and $\boldsymbol{XPD}^{-1} = \tilde{\boldsymbol{X}} \in \mathbb{R}_+^{n\times r}$ with $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{X}}^\top = \boldsymbol{AQ}(\boldsymbol{XQ}^{-\top})^\top = \boldsymbol{AX}^\top$. And so $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{X}}$ are nonnegative matrices leading to the same factorization as $\boldsymbol{A}$, $\boldsymbol{X}$.

Therefore, to analyze the quality of factors, we first compute $\boldsymbol{Q}^* \in \mathbb{R}_+^{r\times r}$ the best permutation and rescaling of $\boldsymbol{A}$ with respect to $\bar{\boldsymbol{A}}$, i.e. $\boldsymbol{Q}^* = \underset{\boldsymbol{Q}=\boldsymbol{PD}}{\operatorname{argmin}} \|\boldsymbol{AQ} - \bar{\boldsymbol{A}}\|$. Then, we compute the Signal to Interference Ratio (SIR) [28] between $\bar{\boldsymbol{A}}$ and $\boldsymbol{AQ}^*$ expressed in dB. The higher the SIR is, the closer $\boldsymbol{AQ}^*$ and $\bar{\boldsymbol{A}}$ are:

$$\text{SIR (dB)} = \frac{1}{r}\sum_{k=1}^{r} 10\log_{10}\left(\frac{\|\boldsymbol{AQ}^*_{:k}\|_2^2}{\|\boldsymbol{AQ}^*_{:k} - \bar{\boldsymbol{A}}_{:k}\|_2^2}\right). \tag{2.4}$$

- **Ability to describe original data:** this criterion analyzes if the original factor $\bar{\boldsymbol{A}}$ can be recovered from the found factor $\boldsymbol{A}$ using nonnegative linear combinations. This is interesting because if we can recover $\bar{\boldsymbol{A}}$ from $\boldsymbol{A}$ using nonnegative linear combinations, then any valid $\boldsymbol{Y}_{:j}$ can be described using nonnegative linear combinations of $\boldsymbol{A}$. Indeed, if $\boldsymbol{Y}_{:j} = \bar{\boldsymbol{A}}(\bar{\boldsymbol{X}}^\top)_{:j}$ with $\bar{\boldsymbol{X}} \in \mathbb{R}_+^{n\times r}$, and $\bar{\boldsymbol{A}} = \boldsymbol{AQ}$ with $\boldsymbol{Q} \in \mathbb{R}_+^{r\times r}$, then $\boldsymbol{Y}_{:j} = \boldsymbol{A}(\boldsymbol{Q}\bar{\boldsymbol{X}}^\top)_{:j}$ where $\boldsymbol{Q}\bar{\boldsymbol{X}}^\top \in \mathbb{R}_+^{n\times r}$.

  To measure the ability to describe original data, we first compute $\boldsymbol{Q}^*$ such that $\boldsymbol{Q}^* = \underset{\boldsymbol{Q}\in\mathbb{R}_+^{r\times r}}{\operatorname{argmin}} \|\boldsymbol{AQ} - \bar{\boldsymbol{A}}\|$. Then we compute the SIR between $\bar{\boldsymbol{A}}$ and $\boldsymbol{AQ}^*$. This criterion is called SIR LC. In contrast to the SIR criterion, the SIR LC does not analyze how close to the original signals the found signals are, but rather how well the found factors fit the original data. The information given by the SIR LC is therefore closer to that given by the relative residuals than by the SIR.

In general, the ground truth of a real dataset is unknown. Therefore, in order to analyze the accuracy of an algorithm, we will regularly work on synthetic data. After having constructed two factor matrices $\boldsymbol{A}$ and $\boldsymbol{X}$, the data matrix $\boldsymbol{Y}$ is built as

$$\boldsymbol{Y} = \boldsymbol{AX}^\top + \boldsymbol{N}, \tag{2.5}$$

where $\boldsymbol{N}$ is a Gaussian noise, with a chosen Signal to Noise Ratio (SNR).

The noise as the same level for each column. Let the chosen SNR be s dB. Each element of $\boldsymbol{N}$ is computed using a normal distribution of mean 0 and variance as follows:

$$\boldsymbol{N}_{ij} \sim \mathcal{N}\left(0, \frac{1}{m}\Big(\sum_{i=1}^{m} \boldsymbol{Y}_{ij}^2\Big) 10^{-s/10}\right). \qquad \forall i = 1, \ldots, m \; ; \; j = 1, \ldots, n. \quad (2.6)$$

2.2 Existing improvements of NMF

Improving NMF can be done through two main axes, either by accelerating the resolution of the problem or by improving the quality of the found factorization. To accelerate the resolution of NMF, the focus is mainly on the algorithms solving the problem, and researchers either try to improve existing algorithms or to develop new algorithms better suited to the problem. As an example, HALS in Algorithm 2.1 can be accelerated by pre-computing matrices $\boldsymbol{Y}\boldsymbol{X}$ and $\boldsymbol{X}^\top\boldsymbol{X}$ and performing the for-loop in `updateHALS` several times [47]. Some recent and interesting works to perform fast NMF are [4, 49, 128].

Nevertheless, accelerating NMF algorithms is not our main objective in this work, and we focus on improving the factorization quality instead. Improving the quality of the factorization means being less sensitive to noise and recovering more meaningful factors $\boldsymbol{A}$ and $\boldsymbol{X}$. Actually, the nonnegativity constraints on $\boldsymbol{A}$ and $\boldsymbol{X}$ exist in this goal. Indeed, we could neglect the nonnegative constraints to solve problem (2.1), and the problem would be easy to solve using truncated SVD and lead to a lower error with respect to $\boldsymbol{Y}$. But the factorization would then be very sensitive to noise and the recovered factors would be more difficult to analyze for humans. Indeed, early works on NMF showed that NMF leads to a part-based representation of the data which is easier to handle [80]. Moreover, in addition to trivial permutations and rescaling presented in previous section, there are infinitely many different matrices $\boldsymbol{A}'$, $\boldsymbol{X}'$ that have the same product as $\boldsymbol{A}, \boldsymbol{X}$: $\boldsymbol{A}'\boldsymbol{X}'^\top = \boldsymbol{A}\boldsymbol{X}^\top$. Imposing the nonnegativity restricts the set of solutions. Nevertheless, most of the time the factorization $\boldsymbol{A}\boldsymbol{X}^\top$ remains non-unique [35].

Improving the quality of the factorization often requires to pay attention to the characteristics of the problem being solved: are data in $\boldsymbol{Y}$ sparse,

smooth, with a special shape, etc. and/or do we want the factors $\boldsymbol{A}$, $\boldsymbol{X}$ to have such characteristics? Once these questions have been answered, the problem can either be regularized so that $\boldsymbol{A}\boldsymbol{X}^\top$, $\boldsymbol{A}$ and $\boldsymbol{X}$ come close to the desired characteristics, or these characteristics can be imposed to them, changing thus the structure of the problem.

Regularizing the NMF problem means adding a regularization term on the cost function that penalizes the elements not satisfying the desired shape constraint. Such a regularization term can for example encourage sparsity [40, 63, 108, 137], smoothness [22, 40, 66, 72], orthogonality [24], or encourage factors to highlight meaningful similarities between recovered signals [136]. Most of the time, the objective function of NMF then looks like:

$$\left\|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}^\top\right\|_F^2 + R_{\boldsymbol{A}\boldsymbol{X}^\top}\left(\boldsymbol{A}\boldsymbol{X}^\top\right) + R_{\boldsymbol{A}}(\boldsymbol{A}) + R_{\boldsymbol{X}}(\boldsymbol{X})\,. \tag{2.7}$$

More generally, fitting the cost function to the problem is a wise choice. Indeed, the Frobenius norm is adapted to solve problems where the noise is assumed to be Gaussian, independent and identically distributed. However, it is more appropriate to turn to the Kullback-Leiber divergence when the observed data comes from a Poisson distribution [61, 131] or to the Itakura-Saito divergence when the data analyzed contains audio spectra [41, 85]. One can also consider using more general divergences that generalize the three presented divergences, like the α-divergence [26, 130], the β-divergence [81, 116] or the Bregman divergence [3, 83].

Another approach is to impose a certain structure to factors $\boldsymbol{A}$ and $\boldsymbol{X}$. Actually, this is the main idea behind NMF as $\boldsymbol{A}$ and $\boldsymbol{X}$ are imposed to be nonnegative. A common constraint is to impose normalization of $\boldsymbol{A}$, so that $\boldsymbol{A}$ is row stochastic: $\boldsymbol{A}\mathbb{1}^r = \mathbb{1}^r$ [72, 112]. Sparsity can be enforced to $\boldsymbol{A}$ and $\boldsymbol{X}$ by optimizing on $\boldsymbol{A}\boldsymbol{S}\boldsymbol{X}^\top$ instead of $\boldsymbol{A}\boldsymbol{X}^\top$, where $\boldsymbol{S} = (1-\theta)I^r + \frac{\theta}{r}\mathbb{1}^r\mathbb{1}^{r\top}$, with $\theta \in [0,1]$ determining the level of sparsity of $\boldsymbol{A}$ and $\boldsymbol{X}$ [66, 98, 132]. In Deep-NMF, matrix $\boldsymbol{A}$ is factorized as $\boldsymbol{A} = \boldsymbol{A}_1\boldsymbol{A}_2\cdots\boldsymbol{A}_L$ to learn hierarchical features [40, 82, 135], while in projective NMF we require $\boldsymbol{X}$ to be $\boldsymbol{X}^\top = \boldsymbol{A}^\top\boldsymbol{Y}$ in order to approximate data using projection on its nonnegative subspace [88, 89, 137].

More recently, requiring each column of $\boldsymbol{A}$ to be the sampling of a continuous function has been considered. Indeed, NMF is often used to analyze data originating from continuous signals, such as spectral data [30, 70, 107]. Those signals can be well approximated using parametrizable functions.

For example, polynomials can be used for globally smooth signals [33], splines for piece-wise smooth signals [8, 141], or mixtures of Gaussian radial basis functions (GRBF) for signals with a few modes (described with a few unimodal signals) [133, 139]. Moreover, using samples of parametrizable functions in the columns of $\boldsymbol{A}$, named $\boldsymbol{A}_{:k}$, allows to describe the data on the whole domain and not only at the sampling points. Indeed, each column $\boldsymbol{Y}_{:j}$ is then described by a linear combination of continuous functions $\sum_k \boldsymbol{A}_{:k}\boldsymbol{X}_{jk}$.

This thesis extends those works. We have developed and analyzed algorithms to work on polynomials (Chapters 3, 6), splines (Chapters 3, 7) and rational functions (Chapter 8). In addition, we have formalized the framework necessary to perform NMF using parametrizable functions, called NMF in Hilbert spaces (H-NMF) because data must lie in a Hilbert space and the used functions must belong to Reproducing Kernel Hilbert Spaces, RKHS (Chapters 4, 5).

This work may therefore remind of existing works [20, 138, 142, 145] that use RKHS in NMF. However, kernel functions are used there to map data to a RKHS of higher dimension, in order to extract nonlinear relationships between elements. This can be useful to extract meaningful features from a dataset, to perform clustering for example. However, it does not describe input data, but its mapping to a higher dimensional space, and therefore prevents data compression or data completion. Those approaches use knowledge about the chosen kernel function to optimize their problem, while in our work the kernel functions behind the RKHS do not need to be known, the two approaches are therefore fundamentally different.

On the other hand, H-NMF has some similarities with functional Matrix Factorization (fMF) for recommended systems. Indeed, in fMF there is no nonnegativity constraint, but each column of factor $\boldsymbol{A}$ is expressed thanks to a function (like a decision tree) [23, 144]. The use of functions in $\boldsymbol{A}$ is motivated by the intent to use contextual information to refine the results. The parameters of the functions are therefore provided to the algorithm, and the goal of the algorithm is to find the most accurate function for these parameters, while in our case the structure of the function is provided to the algorithm and we aim at recovering its parameters. The two approaches are therefore very distinct.

Let us now describe in a few words the existing approaches to solve the NMF problem using parametrizable functions. Three main approaches

have been developed, using unconstrained parameters, nonnegative parameters, or constrained parameters.

Using unconstrained parameters: nonlinear least squares

In [33], Debals et al. consider the case when factor matrix $\boldsymbol{A}$ contains discretization of nonnegative polynomials in each of its columns. The polynomials are imposed to be nonnegative on an interval and not only at discretization points. The authors propose a nonlinear parametrization of the nonnegative polynomials defining matrix $\boldsymbol{A}$. This nonlinear parametrization is presented in more details in Section 2.3.1, equation (2.13). The mixing matrix $\boldsymbol{X}$ is also parametrized as $\boldsymbol{X}_{ij} = \boldsymbol{C}_{ij}^2$. This allows having all parameters in the new formulation free of constraints, and therefore the NMF problem becomes an unconstrained nonlinear least squares problem, featuring a non-convex objective function. This problem is then solved using a standard nonlinear least squares 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 (asymptotic complexity, i.e. complexity for very large-scale datasets). 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, in practice, this method appears to be very slow compared to HALS for small or moderate problems. Moreover, this non alternating approach is not known to perform very well on the standard NMF problem. Furthermore, 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

In this case, matrix $\boldsymbol{A}$ can be described as $\boldsymbol{A} = \boldsymbol{\Pi}\boldsymbol{B}$, where $\boldsymbol{\Pi} \in \mathbb{R}^{m \times d}$ contains the discretization over m points $\{\boldsymbol{\tau}_i\}_{i=1}^m$ of d **nonnegative basis functions** $\{\pi_j(\cdot)\}_{j=1}^d$: $\boldsymbol{\Pi}_{ij} = \pi_j(\boldsymbol{\tau}_i) \geq 0 \ \forall i, j$. 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 [8] and Zdunek et al. [141]. 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, some B-Splines combined using some negative coefficients may still be nonnegative, and it has been proven that estimating nonnegative functions using B-Splines with nonnegative coefficients may lead to a poorer reconstruction than using nonnegative splines [31]. This approach has also been considered for GRBF with nonnegative coefficients in [133].

Using constrained parameters: arbitrary basis functions

This last approach considers that matrix $\boldsymbol{A}$ can be described as $\boldsymbol{A} = \boldsymbol{\Pi}\boldsymbol{B}$, where $\boldsymbol{\Pi}$ is the discretization over m points of d basis functions, which are now arbitrary, i.e. not necessarily nonnegative. The nonnegativity of $\boldsymbol{A}$ is then ensured using constraints on the coefficient matrix $\boldsymbol{B}$. If the mixing matrix $\boldsymbol{X}$ is fixed, this constrained problem is convex and can be described as a problem with quadratic objective. A method using active sets is suggested in [139] for Gaussian Radial Basis Functions, while a method using the Alternating Direction Method of Multipliers (ADMM [48]) is suggested in [140] for B-Splines. The active-set method leads to a relatively large computational complexity for large-scale problems [139], while the ADMM is more suitable.

Using indicator function $\Phi(\boldsymbol{A}) = \sum_{i,j} \phi(\boldsymbol{A}_{ij})$, with $\phi(a) = \infty$ if $a < 0$ and 0 otherwise, ADMM solves the following problem to update $\boldsymbol{A}$:

$$\min_{\boldsymbol{A}\in\mathbb{R}^{m\times r},\boldsymbol{B}\in\mathbb{R}^{d\times r}} \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{\Pi}\boldsymbol{B}\boldsymbol{X}\|_F^2 + \Phi(\boldsymbol{A}) \quad \text{s.t.} \quad \boldsymbol{A} = \boldsymbol{\Pi}\boldsymbol{B}. \tag{2.8}$$

Using projection $[\xi]_+ = \max\{0, \xi\}$, $\rho > 0$ and $\boldsymbol{\Pi}^\dagger = (\boldsymbol{\Pi}^\top\boldsymbol{\Pi})^{-1}\boldsymbol{\Pi}^\top$, the iterations are (see [140] for more details):

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

Note that in this approach, nonnegativity of the functions in $\boldsymbol{A}$ is only ensured at the discretization points. Moreover, the presented scheme updates matrix $\boldsymbol{A}$ and matrix of coefficients $\boldsymbol{B}$ separately, each of these two matrices satisfying only one of the constraints (nonnegativity for $\boldsymbol{A}$, being a spline for $\boldsymbol{B}$). Nevertheless, when the algorithm converges it leads asymptotically to $\boldsymbol{A} = \boldsymbol{\Pi}\boldsymbol{B}$.

2.3 Nonnegative parametrizable functions

We now present existing knowledge about the nonnegative functions we use.

2.3.1 Nonnegative Polynomials

Polynomials of fixed degree d can be uniquely described using $d+1$ coefficients and a chosen basis $\{\pi_i\}_{i=1}^{d+1}$, such as the monomial basis or the Chebyshev basis. Then, each polynomial $p(x)$ is described by its coefficients $\boldsymbol{p} \in \mathbb{R}^{d+1}$ as

$$p(x) = \sum_{i=1}^{d+1} \boldsymbol{p}_i \pi_i(x). \tag{2.9}$$

Let $\boldsymbol{\tau} \in \mathbb{R}^m$ be a vector containing m discretization points, and $\boldsymbol{V}_{\boldsymbol{\tau}}^d \in \mathbb{R}^{m\times(d+1)}$ be the Vandermonde matrix for the chosen basis, i.e. $\boldsymbol{V}_{\boldsymbol{\tau}}^d{}_{ij} = \pi_j(\boldsymbol{\tau}_i)$. It is well known that this Vandermonde matrix is ill-conditioned when the chosen basis is the monomial basis. To avoid this issue, it is therefore better to use another basis, such as the Chebyshev basis, commonly used in numerical analysis. We have

$$p(\boldsymbol{\tau}) = \boldsymbol{V}_{\boldsymbol{\tau}}^d \boldsymbol{p} \tag{2.10}$$

A univariate polynomial is nonnegative on $\mathbb{R}$ if and only if it is a Sum Of Squares (SOS) polynomial, which means that it can be expressed as the finite sum of squared polynomials: $p(x) = \sum_k \left(q_k(x)\right)^2$ for some polynomials q_k [104]. It is also possible to characterize polynomials nonnegative on a fixed interval thanks to the Markov-Lukács theorem. Let $\mathcal{P}^d$ be the set of polynomials of degree d and $\mathcal{P}_+^d([-1,1])$ be the set of polynomials of degree d nonnegative on $[-1,1]$. We have for even degree d:

$$p \in \mathcal{P}_+^d([-1,1]) \Leftrightarrow p(x) = a(x) + (1-x^2)b(x) \qquad \forall x, \tag{2.11}$$

with $a \in \mathcal{P}_+^d, b \in \mathcal{P}_+^{d-2}$ both SOS.

For odd degree d, this equation becomes:

$$p \in \mathcal{P}_+^d([-1,1]) \;\Leftrightarrow p(x) = (1-x)a(x) + (1+x)b(x) \qquad \forall x, \tag{2.12}$$

with $a, b \in \mathcal{P}_+^{d-1}$ both SOS.

Both equations (2.11) and (2.12) can be extended to any interval I using an appropriate change of variables. Moreover, from [101], polynomials a and b can be restricted to squared polynomials [33]. Equation (2.11) becomes then

$$p \in \mathcal{P}_+^d([-1,1]) \;\Leftrightarrow p(x) = a^2(x) + (1-x^2)b^2(x) \qquad \forall x, \tag{2.13}$$

with $a \in \mathcal{P}^{d/2}, b \in \mathcal{P}^{d/2-1}$, and similarly for equation (2.12).

On the other hand, SOS polynomials can be expressed using positive semidefinite matrices [15], which can be optimized very efficiently as semidefinite programs using interior-point algorithms (see e.g. [123]). Indeed, a degree d polynomial a is SOS if and only if

$$a(x) = \sum_k \left(q_k(x)\right)^2 = \sum_k \left(\boldsymbol{V}_x^d \boldsymbol{q}_k\right)^2 = \boldsymbol{V}_x^d \sum_k \boldsymbol{q}_k \boldsymbol{q}_k^\top \boldsymbol{V}_x^d = \boldsymbol{V}_x^d \boldsymbol{Q} \boldsymbol{V}_x^d \qquad \forall x, \tag{2.14}$$

where $\boldsymbol{Q}$ is a positive semidefinite matrix in $\mathbb{R}^{\left(\frac{d}{2}+1\right)\times\left(\frac{d}{2}+1\right)}$, $\boldsymbol{Q} \in \mathbb{S}_+^{\frac{d}{2}+1}$, since it is a sum of positive semidefinite rank-one terms $\boldsymbol{q}_k \boldsymbol{q}_k^\top$. $\boldsymbol{V}_x^d \boldsymbol{Q} \boldsymbol{V}_x^d$ is called the Gram matrix representation of a. Coefficients $\boldsymbol{a}$ can be easily recovered from matrix $\boldsymbol{Q}$, in a way that depends on the chosen basis. For example, in the monomial basis, we have $\boldsymbol{a}_k = \sum_{i,j:i+j=k} \boldsymbol{Q}_{ij}$, and in the Chebyshev basis $\boldsymbol{a}_k = \sum_{i+j=k} \frac{\boldsymbol{Q}_{ij}}{2} + \sum_{|i-j|=k} \frac{\boldsymbol{Q}_{ij}}{2}$. Therefore, using an appropriate matrix $\boldsymbol{G}^d \in \mathbb{R}^{(d+1)\times\left(\frac{d}{2}+1\right)^2}$, we have

$$\boldsymbol{a} = \boldsymbol{G}^d \operatorname{vec}(\boldsymbol{Q}) \qquad \text{with } \boldsymbol{Q} \in \mathbb{S}_+^{\frac{d}{2}+1}. \tag{2.15}$$

Using an appropriate matrix $\boldsymbol{R}^d \in \mathbb{R}^{(d+1)\times(d_a^2+d_b^2)}$, it is therefore possible

to express any polynomial $p \in \mathcal{P}_+^d(I)$ as

$$\boldsymbol{p} = \boldsymbol{R}^d \begin{bmatrix} \text{vec}(\boldsymbol{S}_a) \\ \text{vec}(\boldsymbol{S}_b) \end{bmatrix} \quad \text{, with } \boldsymbol{S}_a \in \mathbb{S}_+^{d_a}, \boldsymbol{S}_b \in \mathbb{S}_+^{d_b}. \tag{2.16}$$

For example, for an even degree d polynomial nonnegative on $[-1, 1]$, we have $d_a = d/2 + 1$, $d_b = d/2$, and $\boldsymbol{R}^d = [\boldsymbol{G}^d \ \tilde{\boldsymbol{G}}^d]$, where Gram matrix $\tilde{\boldsymbol{G}}^d$ takes in consideration the multiplication by $(1 - x^2)$.

2.3.2 Nonnegative Splines

Univariate splines with fixed interior knots $\{t_i\}_{i=1}^k$ can be uniquely described using the B-Spline basis. This basis can be computed using the Cox-de Boor recursion formula:

$$\begin{aligned} B_{i,0}(x) &= \begin{cases} 1 & \text{if } x \in [t_i, t_{i+1}] \\ 0 & \text{otherwise.} \end{cases} \\ B_{i,k}(x) &= \frac{x - t_i}{t_{i+k} - t_i} B_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i-1,k-1}(x). \end{aligned} \tag{2.17}$$

The B-Spline basis consists then of all $B_{i,d}$, where d is the considered degree for the splines. In this work, $d = 3$ which is a common choice. To consider a closed interval, the first and the last knot are repeated 4 times. They are therefore $k + 2$ basis elements, with k the number of interior knots, neglecting the repetitions of the first and the last knot. Each basis element is nonzero over 4 intervals (including length zero intervals) and is a cubic polynomial over these intervals. Splines are therefore piecewise cubic polynomials. The B-Spline basis is illustrated in Figure 2.2, for $k = 6$ interior knots.

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):

$$f(x) = f_1(x)x + f_2(x)(1 - x) \tag{2.18}$$

with f_1, f_2 two quadratic nonnegative polynomials. Moreover, a quadratic polynomial $f_1(x) = a_1x^2 + b_1x + c_1$ 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

$$\mathcal{Q}_r = \{(x_1, x_2, x_3) \text{ st. } 2x_1x_2 \geq x_3^2, \ x_1, x_2 \geq 0\} \tag{2.19}$$

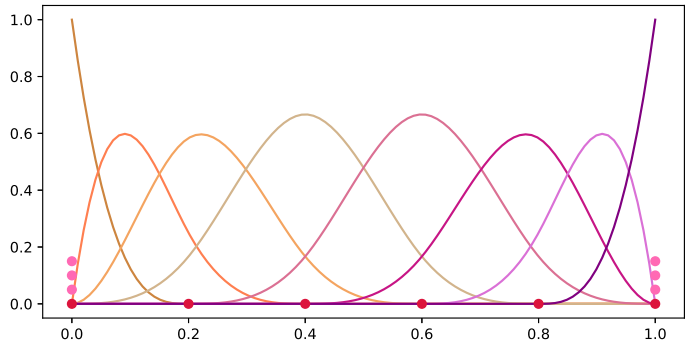


Fig. 2.2 Illustration of the B-Spline basis for $k = 6$ interior knots, in red. The first and last knot are repeated 4 times, as shown by the pink points.

this nonnegativity condition on $f_1(x)$ becomes exactly $(a_1, c_1, b_1/\sqrt{2}) \in \mathcal{Q}_r$. Hence a cubic polynomial is nonnegative over $[0, 1]$ if and only if it can be written as

$$f(x) = (a_1 - a_2)x^3 + (b_1 - b_2 + a_2)x^2 + (c_1 + b_2 - c_2)x + c_2 \qquad (2.20)$$

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. $\boldsymbol{f} = \boldsymbol{R}_S(\boldsymbol{q}_1\ \boldsymbol{q}_2)^\top$.

Let us go back to splines. Splines consist of cubic polynomials on each non-empty interval $[t_i, t_i + 1]$. The coefficients of each polynomial $\boldsymbol{f}_i$ can be expressed as a linear mapping of the coefficients of the four B-Splines that are nonzero over this interval, $(\boldsymbol{g}_i, \boldsymbol{g}_{i+1}, \boldsymbol{g}_{i+2}, \boldsymbol{g}_{i+3})$. This linear map $\boldsymbol{N}_i$ can be computed from the definition in (2.17) and is invertible. Therefore, B-Splines coefficients can be expressed as a linear map applied to elements of a rotated quadratic cone:

$$\begin{bmatrix} \boldsymbol{g}_i \\ \boldsymbol{g}_{i+1} \\ \boldsymbol{g}_{i+2} \\ \boldsymbol{g}_{i+3} \end{bmatrix} = \boldsymbol{N}_i^{-1}\boldsymbol{f}_i = \boldsymbol{N}_i^{-1}\boldsymbol{R}_S \begin{bmatrix} \boldsymbol{q}_{1i} \\ \boldsymbol{q}_{2i} \end{bmatrix} \quad \text{with } \boldsymbol{q}_{1i}, \boldsymbol{q}_{2i} \in \mathcal{Q}_r. \qquad (2.21)$$

Imposing constraint (2.21) over all intervals $i = 1, \ldots, k-1$ ensures the nonnegativity of the spline created by the coefficients $\boldsymbol{g} \in \mathbb{R}^{k+2}$.

2.3.3 Nonnegative Rational Functions

A rational functions is by definition a ratio of two polynomials:

$$f(x) = \frac{h(x)}{g(x)}. \tag{2.22}$$

The degree of a rational function is $\boldsymbol{d} = (d_1, d_2)$, where d_1 is the degree of the numerator h and d_2 is the degree of the denominator g. We focus in this work on univariate rational functions.

Rational functions nonnegative on a fixed interval I can be described without loss of generality as a ratio of two polynomials nonnegative on I [68]. Moreover, as it is often undesirable to tend to infinity, the denominator is also imposed to be nonzero in the considered interval. From Section 2.3.1 we know that nonnegative polynomials can be expressed as a combination of SOS polynomials or a combination of squared polynomials. Therefore, using (2.10) and (2.16) we have that a rational function f nonnegative on $[-1, 1]$ with even degrees $\boldsymbol{d} = (d_1, d_2)$ can be expressed on discretization points $\boldsymbol{\tau}$ as:

$$f(\boldsymbol{\tau}) = \left[\boldsymbol{V}_{\boldsymbol{\tau}}^{d_1}\boldsymbol{R}^{d_1}\begin{bmatrix}\boldsymbol{S}_{h_a}\\ \boldsymbol{S}_{h_b}\end{bmatrix}\right] \Bigg/ \left[\boldsymbol{V}_{\boldsymbol{\tau}}^{d_2}\boldsymbol{R}^{d_2}\begin{bmatrix}\boldsymbol{S}_{g_a}\\ \boldsymbol{S}_{g_b}\end{bmatrix} + \epsilon\right], \tag{2.23}$$

where the division is performed element-wise, $\boldsymbol{S}_{h_a} \in \mathbb{S}_+^{\frac{d_1}{2}+1}$, $\boldsymbol{S}_{h_b} \in \mathbb{S}_+^{\frac{d_1}{2}}$, $\boldsymbol{S}_{g_a} \in \mathbb{S}_+^{\frac{d_2}{2}+1}$, $\boldsymbol{S}_{g_b} \in \mathbb{S}_+^{\frac{d_2}{2}}$ and ϵ is a small fixed positive number preventing the denominator to go to 0.

From (2.13) we know that f can also be expressed as:

$$f_{\boldsymbol{\tau}}(\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{g}_1, \boldsymbol{g}_2) = \frac{(\boldsymbol{V}_{\boldsymbol{\tau}}^{\frac{d_1}{2}}\boldsymbol{h}_1)^2 + (1-\boldsymbol{\tau}^2)\cdot(\boldsymbol{V}_{\boldsymbol{\tau}}^{\frac{d_1}{2}-1}\boldsymbol{h}_2)^2}{(\boldsymbol{V}_{\boldsymbol{\tau}}^{\frac{d_2}{2}}\boldsymbol{g}_1)^2 + (1-\boldsymbol{\tau}^2)\cdot(\boldsymbol{V}_{\boldsymbol{\tau}}^{\frac{d_2}{2}-1}\boldsymbol{g}_2)^2 + \epsilon}; \tag{2.24}$$

where $\boldsymbol{h}_1 \in \mathbb{R}^{\frac{d_1}{2}+1}, \boldsymbol{h}_2 \in \mathbb{R}^{\frac{d_1}{2}}, \boldsymbol{g}_1 \in \mathbb{R}^{\frac{d_2}{2}+1}$, $\boldsymbol{g}_2 \in \mathbb{R}^{\frac{d_2}{2}}$. The division is performed element-wise and '$\cdot$' stands for the element-wise multiplication. Similar expressions to (2.23) and (2.24) can be found to describe rational functions with other degrees and nonnegative on other intervals.

Representation (2.24) is redundant. Indeed, when multiplying $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, $\boldsymbol{g}_1$, $\boldsymbol{g}_2$, and $\boldsymbol{\epsilon}$ by the same constant α, the represented function is the same, although the coefficients are different. To overcome this scaling issue, we can impose a condition on the coefficients, for example that the denominator is monic. As we are working in Chebyshev basis, denoted as $\{T_i\}_{i=0}^{d}$, we have $x^2 = \frac{T_2+T_0}{2}$ and $T_m T_n = \frac{1}{2}\left(T_{m+n} + T_{|m-n|}\right)$. Therefore, the highest coefficient of the denominator is (if $d_2 \geq 4$):

$$\frac{1}{2}(g_1)^2_{\frac{d_2}{2}+1} T_{d_2} - \frac{1}{8}(g_2)^2_{\frac{d_2}{2}} T_{d_2}. \tag{2.25}$$

The determinant is monic if

$$\frac{1}{2}(g_1)^2_{\frac{d_2}{2}+1} - \frac{1}{8}(g_2)^2_{\frac{d_2}{2}} = 1 \Leftrightarrow (\boldsymbol{g}_1)_{\frac{d_2}{2}+1} = \frac{\sqrt{8 + (\boldsymbol{g}_2)^2_{\frac{d_2}{2}}}}{2}. \tag{2.26}$$

Doing something similar for representation (2.23) is possible, and requests to add a constraint:

$$\left(\boldsymbol{R}^{d_2} \begin{bmatrix} \boldsymbol{S}_{g_a} \\ \boldsymbol{S}_{g_b} \end{bmatrix}\right)_{d_2+1} = 1. \tag{2.27}$$

All the analyzed parametrizable functions can thus be constrained to be nonnegative on a fixed interval, and have an explicit representation in this case, using linear or nonlinear (for rational functions) mappings of elements in positive semidefinite cones or rotated quadratic cones.

We now have all the needed tools for the following chapters.

3

NMF using Polynomials and Splines

NMF is regularly used to analyze data consisting of samples of continuous functions. Such data can often be well approximated by polynomials or splines. Therefore, it has been considered to impose the factor $\boldsymbol{A}$ to contain in each of its columns samples of nonnegative polynomials [33], or splines [9, 140, 141]. This allows one to recover a factor $\boldsymbol{A}$ with smooth columns, by nature of polynomials and splines, and by this way may permit to be less sensitive to noise than when using simple nonnegative vectors. Moreover, decomposing input data as linear combinations of functions instead of vectors allows one to express the input signals continuously, and not only at some discretization points.

Polynomials and splines have the advantage of being able to describe many real-life signals using a few parameters. Polynomials are suitable for globally smooth signals, while splines are suitable for piece-wise smooth continuous signals. Moreover, both sets of functions can be uniquely described using coefficients in a certain basis, like the monomial or the Chebyshev basis for polynomials, or the B-spline basis for splines. We say that they are linearly parametrizable functions.

In this chapter, we analyze the NMF problem when factor $\boldsymbol{A}$ contains in

each of its columns the sampling of a nonnegative linearly parametrizable function. We go even further since we also consider the case where data contains functions and no longer samples of functions. In this case, $\boldsymbol{A}$ contains nonnegative linearly parametrizable functions. Section 3.1 introduces this problem, called NMF on Linearly Parametrizable functions (LP-NMF), and proposes a generalization of the Hierarchical Alternating Least Squares (HALS) algorithm to solve it. Section 3.2 focuses on a crucial component of the proposed HALS algorithm, namely the projection operator over the set of nonnegative functions, in the case of polynomials or splines. In Section 3.3, we compare our approach to existing approaches using polynomials or splines in NMF. Several numerical experiments, using both synthetic and real-world signals, allow us to assess the accuracy and speed of algorithms.

3.1 HALS for NMF using linearly parametrizable functions

In this section, we formally define the NMF using linearly parametrizable functions problem, and then extend the HALS algorithm, commonly used for classical NMF, to deal this case.

3.1.1 NMF using Linearly Parametrizable functions (LP-NMF)

Let us first define formally what we mean by "linearly parametrizable functions".

Definition 3.1. Linearly parametrizable functions A set of functions $\mathcal{F}$ contains linearly parametrizable functions if every function $f \in \mathcal{F}$ can be described as a linear combination of a fixed finite basis of functions $\{\pi_1(a), \pi_2(a), \ldots, \pi_d(a)\}$:

$$f(a) = \sum_{l=1}^{d} \pi_l(a) \boldsymbol{f}_l.$$

Vector $\boldsymbol{f} \in \mathbb{R}^d$ is called the vector of coefficients of f.

Consider now a set $\mathcal{Y} = \{Y_1(a), \ldots, Y_n(a)\}$ of n univariate functions defined over a common fixed interval I. The NMF over Linearly Parametrizable functions problem (LP-NMF) aims at recovering a dictionary $\mathcal{A} =$

$\{A_1(a), \ldots, A_r(a)\}$ of r nonnegative linearly parametrizable functions A_k observed on interval I, and a nonnegative mixing matrix $\boldsymbol{X} \in \mathbb{R}_+^{n \times r}$, which provide an approximate linear description of the original data as follows:

$$\begin{aligned} Y_j(a) &\simeq \sum_{k=1}^{r} A_k(a) \boldsymbol{X}_{jk} && \forall a \in I, \\ \text{such that } A_k(a) &\geq 0,\ \boldsymbol{X}_{jk} \geq 0 && 1 \leq k \leq r,\ 1 \leq j \leq n. \end{aligned} \tag{3.1}$$

Factorization rank $r \ll n$ is provided, as well as a the set $\mathcal{F}$ of linearly parametrizable functions containing the dictionary ($\mathcal{A} \subset \mathcal{F}$). As $\mathcal{F}$ contains linearly parametrizable functions, every function $A_k(a) \in \mathcal{A}$ can be described using a vector of coefficients $\boldsymbol{B}_{:k} \in \mathbb{R}^d$. Dictionary $\mathcal{A}$ is therefore described by coefficients matrix $\boldsymbol{B} \in \mathbb{R}^{d \times r}$.

We also introduce the set $\mathcal{F}_+(I) \subset \mathbb{R}^d$, which is the set of coefficients describing functions in $\mathcal{F}$ that are nonnegative over interval I, i.e.

$$\boldsymbol{B}_{:k} \in \mathcal{F}_+(I) \Leftrightarrow A_k(a) = \sum_{l=1}^{d} \pi_l(a) \boldsymbol{B}_{lk} \geq 0 \text{ for all } a \in I. \tag{3.2}$$

To solve the LP-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 I, and the integral of their product with any basis function is computable, i.e. we know $\int_I \pi_l(a) Y_j(a)\, \mathrm{d}a$ $\forall l = 1, \ldots, d\ ;\ j = 1, \ldots n$.

2. Functions are known for m fixed discretization points, denoted $\boldsymbol{\tau} = \{\boldsymbol{\tau}_i\}_{i=1}^m \subset I$ and are represented using column vectors $\{\boldsymbol{Y}_{:j}\}_{j=1}^n \subset \mathbb{R}^m$ with $\boldsymbol{Y}_{ij} = Y_j(\boldsymbol{\tau}_i)$.

This leads to the two following finite-dimensional formulations of LP-NMF,

where basis functions are grouped in row vector $\boldsymbol{\pi}(a) = (\pi_1(a) \ \cdots \ \pi_d(a))$:

$$\min_{\boldsymbol{B},\boldsymbol{X}} \sum_{j=1}^{n} \int_{I} \big(Y_j(a) - \boldsymbol{\pi}(a)\boldsymbol{B}\boldsymbol{X}_{j:}{}^{\top}\big)^2 \,\mathrm{d}a \qquad \text{(integral cost)}$$
$$\text{integrable functions}$$

$$\text{or} \quad \min_{\boldsymbol{B},\boldsymbol{X}} \sum_{j=1}^{n} \sum_{i=1}^{m} \big(\boldsymbol{Y}_{ij} - \boldsymbol{\pi}(\boldsymbol{\tau}_i)\boldsymbol{B}\boldsymbol{X}_{j:}{}^{\top}\big)^2 \qquad \text{(sum cost)}$$
$$\text{only observations}$$

$$\text{s.t.} \ \ \boldsymbol{B}_{:k} \in \mathcal{F}_+(I), \ \boldsymbol{X}_{jk} \geq 0 \qquad \forall\, j = 1,\ldots,n \,;\, k = 1,\ldots,r.$$

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 $Y_i(a)$ 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 $\boldsymbol{\tau}$ observation abscissas (discretization points).

Interestingly, these two costs can be analyzed at once. Indeed, consider the sum cost. It can be rewritten as:

$$\min_{\boldsymbol{B},\boldsymbol{X}} \sum_{j=1}^{n}\sum_{i=1}^{m} \boldsymbol{Y}_{ij}^2 - 2\sum_{j=1}^{n}\Big(\sum_{i=1}^{m} \boldsymbol{Y}_{ij}\boldsymbol{\pi}(\boldsymbol{\tau}_i)\Big)\boldsymbol{B}\boldsymbol{X}_{j:}{}^{\top} + \sum_{j=1}^{n} \boldsymbol{X}_{j:}\boldsymbol{B}^{\top}\Big(\sum_{i=1}^{m} \boldsymbol{\pi}(\boldsymbol{\tau}_i)^{\top}\boldsymbol{\pi}(\boldsymbol{\tau}_i)\Big)\boldsymbol{B}\boldsymbol{X}_{j:}{}^{\top}.$$

The first term can be neglected, while the elements in parenthesis can be pre-computed. Using a similar reasoning for the integral case, we obtain the following equivalent formulation, with matrices $\boldsymbol{Z}$ and $\boldsymbol{M}$ defined in Table 3.1:

$$\min_{\boldsymbol{B},\boldsymbol{X}} \sum_{j=1}^{n} -2\boldsymbol{Z}_{:j}{}^{\top}\boldsymbol{B}\boldsymbol{X}_{j:}{}^{\top} + \boldsymbol{X}_{j:}\boldsymbol{B}^{\top}\boldsymbol{M}\boldsymbol{B}\boldsymbol{X}_{j:}{}^{\top} \qquad \text{(LP-NMF)}$$

$$\text{s.t.} \ \ \boldsymbol{B}_{:k} \in \mathcal{F}_+(I), \ \boldsymbol{X}_{jk} \geq 0 \qquad \forall\, j = 1,\ldots,n \,;\, k = 1,\ldots,r.$$

	integral cost	sum cost
$\boldsymbol{Z} \in \mathbb{R}^{d\times n}$	$\boldsymbol{Z}_{lj} = \int_I \pi_l(a)Y_j(a)\,\mathrm{d}a$	$\sum_{i=1}^{m} \pi_l(\boldsymbol{\tau}_i)\boldsymbol{Y}_{ij}$
$\boldsymbol{M} \in \mathbb{R}^{d\times d}$	$\boldsymbol{M}_{lj} = \int_I \pi_l(a)\pi_j(a)\,\mathrm{d}a$	$\sum_{i=1}^{m} \pi_l(\boldsymbol{\tau}_i)\pi_j(\boldsymbol{\tau}_i)$

Table 3.1 Matrices $\boldsymbol{Z}$ and $\boldsymbol{M}$ for unified formulation of LP-NMF

Note that the sum case can be seen as an approximation of the integral one. Indeed, if the discretization points are equally spaced, one can approximate integral $\int_I f(a)g(a)\,\mathrm{d}a$ with the Riemann sum, using $|I|$ to denote the length of interval I: $\int_I f(a)g(a)\,\mathrm{d}a \simeq \frac{|I|}{m}\sum_{i=1}^{m} f(\tau_i)g(\tau_i)$. The Riemann sum uses only values at discretization points.

3.1.2 A generalized HALS for LP-NMF

We propose to solve the LP-NMF problem with an adapted version of HALS (Algorithm 2.1) that updates alternatively the columns of $\boldsymbol{A}$ (through their coefficients contained in matrix $\boldsymbol{B}$) and the ones 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 describe them in turn below.

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

$$C(\boldsymbol{B},\boldsymbol{X}) = \sum_{j=1}^{n} -2\boldsymbol{Z}_{:j}{}^{\top}\boldsymbol{B}\boldsymbol{X}_{j:}{}^{\top} + \boldsymbol{X}_{j:}\boldsymbol{B}^{\top}\boldsymbol{M}\boldsymbol{B}\boldsymbol{X}_{j:}{}^{\top} \tag{3.3}$$

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

$$\frac{\partial C}{\partial \boldsymbol{B}_{:k}}(\boldsymbol{B},\boldsymbol{X}) = 2\big(-\boldsymbol{Z}\boldsymbol{X}_{:k} + \boldsymbol{M}\sum_{s=1}^{r}\boldsymbol{B}_{:s}\boldsymbol{X}_{:s}{}^{\top}\boldsymbol{X}_{:k}\big). \tag{3.4}$$

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

$$\frac{\partial C}{\partial \boldsymbol{B}_{:k}}(\boldsymbol{B},\boldsymbol{X}) = 0 \Rightarrow \boldsymbol{M}\boldsymbol{B}_{:k} = \frac{\boldsymbol{Z}\boldsymbol{X}_{:k} - \boldsymbol{M}\sum_{s\neq k}\boldsymbol{B}_{:s}\boldsymbol{X}_{:s}{}^{\top}\boldsymbol{X}_{:k}}{\boldsymbol{X}_{:k}{}^{\top}\boldsymbol{X}_{:k}}. \tag{3.5}$$

To take the nonnegativity constraint $\boldsymbol{B}_{:k} \in \mathcal{F}_+(I)$ into account, one simply projects the unconstrained solution on set $\mathcal{F}_+(I)$, using Equation (3.9) presented in next section. This is optimal thanks to the fact that cost C is convex quadratic in $\boldsymbol{B}_{:k}$. A formal proof of the optimality of this approach can be found later in Chapter 5 (Lemma 5.2). This approach leads to the following update for the columns in $\boldsymbol{B}$, using the fact that matrix $\boldsymbol{M}$ is

invertible:

$$\boldsymbol{B}_{:k} \leftarrow \left[\frac{(\boldsymbol{M})^{-1}\boldsymbol{Z}\boldsymbol{X}_{:k} - \sum_{s\neq k}\boldsymbol{B}_{:s}\boldsymbol{X}_{:s}{}^{\top}\boldsymbol{X}_{:k}}{\boldsymbol{X}_{:k}{}^{\top}\boldsymbol{X}_{:k}}\right]_{\mathcal{F}_+(I)} \tag{3.6}$$

The projection operator onto $\mathcal{F}_+(I)$, denoted as $[\cdot]_{\mathcal{F}_+(I)}$, 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 2.1. 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 3.2 in the case of polynomials and splines.

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

$$\frac{\partial C}{\partial \boldsymbol{X}_{:k}}(\boldsymbol{B},\boldsymbol{X}) = 2\big(-\boldsymbol{Z}^{\top}\boldsymbol{B}_{:k} + \sum_{s=1}^{r}\boldsymbol{X}_{:s}\boldsymbol{B}_{:s}{}^{\top}\boldsymbol{M}\boldsymbol{B}_{:k}\big) \tag{3.7}$$

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

$$\boldsymbol{X}_{:k} \leftarrow \left[\frac{\boldsymbol{Z}^{\top}\boldsymbol{B}_{:k} - \sum_{s\neq k}\boldsymbol{X}_{:s}\boldsymbol{B}_{:s}{}^{\top}\boldsymbol{M}\boldsymbol{B}_{:k}}{\boldsymbol{B}_{:k}{}^{\top}\boldsymbol{M}\boldsymbol{B}_{:k}}\right]_{+} \tag{3.8}$$

where $[\cdot]_+$ is the straightforward projection over nonnegative reals: $[\xi]_+ = \max\{\xi, 0\}$. Again this update is optimal (Lemma 5.2).

The pseudo-code for LP-HALS, our adapted version of HALS, can be found in Algorithm 3.1, including the number of floating point operations for each statement. Besides the two updates described above, we normalize (or scale) all columns of $\boldsymbol{B}$ after each full update according to $\boldsymbol{B}_{:k} \leftarrow \frac{\boldsymbol{B}_{:k}}{\left(\boldsymbol{B}_{:k}{}^{\top}\boldsymbol{M}\boldsymbol{B}_{:k}\right)^{1/2}}$, scaling the columns of $\boldsymbol{X}$ accordingly. And similarly, we normalize the columns of $\boldsymbol{X}$ after each full update with $\boldsymbol{X}_{:k} \leftarrow \frac{\boldsymbol{X}_{:k}}{\left(\boldsymbol{X}_{:k}{}^{\top}\boldsymbol{X}_{:k}\right)^{1/2}}$, and scale the columns of $\boldsymbol{B}$ accordingly.

We also use the acceleration technique introduced in [47], which consists in performing the first for loop of the updates several times. Note that the matrices $\boldsymbol{Z}$, $\boldsymbol{M}$ and $\boldsymbol{M}_1 = \boldsymbol{M}^{-1}\boldsymbol{Z}$ can be pre-computed. Looking at the

Algorithm 3.1 LP-HALS

Require: matrices $\boldsymbol{Z}, \boldsymbol{M}$, rank r, initial $\boldsymbol{B} \in \mathbb{R}^{d\times r}$ and initial $\boldsymbol{X} \in \mathbb{R}^{n\times r}$

$\boldsymbol{M_1} = \boldsymbol{M}^{-1}\boldsymbol{Z}$

while Stop Condition not encountered **do**

$\boldsymbol{B} \leftarrow$ `updateB`$(\boldsymbol{M_1}, \boldsymbol{M}, \boldsymbol{B}, \boldsymbol{X})$

$\boldsymbol{X} \leftarrow$ `updateX` $(\boldsymbol{Z}, \boldsymbol{M}, \boldsymbol{B}, \boldsymbol{X})$

function UPDATEB$(\boldsymbol{M_1}, \boldsymbol{M}, \boldsymbol{B}, \boldsymbol{X})$

$\boldsymbol{P} = \boldsymbol{M_1}\boldsymbol{X}, \boldsymbol{Q} = \boldsymbol{X}^\top\boldsymbol{X}$ ▷ $(2n-1)r(d+r)$ flops

for $\boldsymbol{B}_{:k}$ in $\boldsymbol{B}$ **do**

$\boldsymbol{t} = \boldsymbol{P}_{:k} - \sum_{s\neq k}\boldsymbol{B}_{:s}\boldsymbol{Q}_{sk}$ ▷ $2d(r-1)$ flops

$\boldsymbol{B}_{:k} \leftarrow$ `Projection`$(\boldsymbol{t}/\boldsymbol{Q}_{kk})$ ▷ P $+d$ flops

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

$\boldsymbol{B}_{:k} \leftarrow \boldsymbol{B}_{:k}/nb$, $\boldsymbol{X}_{:k} \leftarrow \boldsymbol{X}_{:k}\cdot nb$ ▷ $d+n$ flops

return $\boldsymbol{B}$ ▷ $\mathcal{O}(rP+nrd)$

function UPDATEX$(\boldsymbol{Z}, \boldsymbol{M}, \boldsymbol{B}, \boldsymbol{X})$

$\boldsymbol{P} = \boldsymbol{Z}^\top\boldsymbol{B}, \boldsymbol{Q} = \boldsymbol{B}^\top\boldsymbol{M}\boldsymbol{B}$ ▷ $(2d-1)r(n+d+r)$ flops

repeat $\min(1+\rho_{\boldsymbol{X}}/2, 10)$ times

for $\boldsymbol{X}_{:k}$ in $\boldsymbol{X}$ **do**

$\boldsymbol{t} = \boldsymbol{P}_{:k} - \sum_{s\neq k}\boldsymbol{X}_{:s}\boldsymbol{Q}_{sk}$ ▷ $2n(r-1)$ flops

$\boldsymbol{X}_{:k} \leftarrow \max(0, \boldsymbol{t}/\boldsymbol{Q}_{kk})$ ▷ $2n$ flops

until no more progress

for $\boldsymbol{X}_{:k}$ in $\boldsymbol{X}$ **do** ▷ Normalization

$nx \leftarrow (\boldsymbol{X}_{:k}{}^\top\boldsymbol{X}_{:k})^{1/2}$ ▷ $3n-1$ flops

$\boldsymbol{X}_{:k} \leftarrow \boldsymbol{X}_{:k}/nx$, $\boldsymbol{B}_{:k} \leftarrow \boldsymbol{B}_{:k}\cdot nx$ ▷ $n+d$ flops

return $\boldsymbol{X}$ ▷ $\mathcal{O}(nrd)$

update of $\boldsymbol{B}$, it is not straightforward to compute the complexity of the projection onto the set of nonnegative functions, and this depends on the chosen functions. Moreover, it is expected that this projection can be quite costly and no significative 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 [47], 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 [47]).

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 is the complexity of the projection (this estimate is based on $r < d < n$). The complexity of the projection depends only on the degree d, as we will see in the next section. Since $d < n$, the complexity is therefore dominated by $\mathcal{O}(nrd)$ for large datasets (when n and m are very large). Asymptotically, the complexity of LP-HALS is therefore lower than the complexity of standard HALS which is $\mathcal{O}(nrm)$.

Let us now say a word about how our algorithm LP-HALS fits in with existing work on NMF using polynomials or splines. Existing works are described briefly in Section 2.2. We can see that our method with the sum cost uses an arbitrary basis function $\boldsymbol{\Pi} \in \mathbb{R}^{m \times d}$; with $\boldsymbol{\Pi}_{il} = \pi_l(\boldsymbol{\tau}_i)$, to describe the functions via constrained parameters $\boldsymbol{B}$, and is thus close to [140].

Nevertheless, our approach differs from previous work in several ways. First it does not explicitly compute matrix $\boldsymbol{A}$ but uses the coefficients $\boldsymbol{B}$ instead. 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, our approach ensures the nonnegativity of the functions in $\boldsymbol{A}$ over the entire considered interval, and not only at discretization points. This is ensured using projections (see Section 3.2, for both polynomials and splines), which is the main new ingredient in our approach. We use sum of squares (SOS) in the projection step to represent nonnegative polynomials as in [33], but we solve the problem as a semidefinite program, instead of using a least squares solver. The good accuracy and good convergence properties of the suggested method are assessed trough experiments in Section 3.3.

3.2 Projection onto nonnegative polynomials or splines

To perform the update of $\boldsymbol{B}$ in LP-HALS we need to project its columns onto the set $\mathcal{F}_+(I)$, 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, using the Gram matrix $\boldsymbol{M}$ defined in Table 3.1:

$$\begin{aligned}[\boldsymbol{f}]_{\mathcal{F}_+(I)} &= \operatorname{argmin}_{\boldsymbol{g}} \|\boldsymbol{f}-\boldsymbol{g}\|_{\boldsymbol{M}}^2 && \text{s.t. } \boldsymbol{g} \in \mathcal{F}_+(I) \\ &= \operatorname{argmin}_{\boldsymbol{g}} (\boldsymbol{f}-\boldsymbol{g})^\top \boldsymbol{M} (\boldsymbol{f}-\boldsymbol{g}) && \text{s.t. } \boldsymbol{g} \in \mathcal{F}_+(I)\end{aligned} \tag{3.9}$$

$\boldsymbol{M}$ is a symmetric and positive definite matrix and it can thus be expressed as $\boldsymbol{M} = \boldsymbol{L}^\top \boldsymbol{L}$ with $\boldsymbol{L} \in \mathbb{R}^{d \times d}$ (Cholesky decomposition of $\boldsymbol{M}$). Therefore

$$[\boldsymbol{f}]_{\mathcal{F}_+(I)} = \operatorname{argmin}_{\boldsymbol{g}} \|\boldsymbol{L}(\boldsymbol{f}-\boldsymbol{g})\|_2^2 \qquad \text{s.t. } \boldsymbol{g} \in \mathcal{F}_+(I).$$

Note that matrix $\boldsymbol{M}$ defines the metric used for the projection. It actually comes from the Hessian of the cost function with respect to a column of $\boldsymbol{B}$.

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

3.2.1 Projection onto nonnegative polynomials

In Section 2.3.1 we showed several ways to represent polynomials nonnegative on a given interval. Using in particular equation (2.16), it is possible to express the projection on polynomials of degree d, nonnegative on a given interval, as the solution of the following semidefinite program (SDP):

$$\begin{aligned}&\min t \\ &\text{s.t. } (\boldsymbol{u}, t) \in \mathbb{L}^{d+1} && \text{Lorentz cone: } \|\boldsymbol{u}\| \leq t \\ &\boldsymbol{S}_a \in \mathbb{S}_+^{d_a}, \boldsymbol{S}_b \in \mathbb{S}_+^{d_b} && \text{Semidefinite cones} \\ &\boldsymbol{u} = \boldsymbol{L}\left(\boldsymbol{f} - \boldsymbol{R}^d \begin{bmatrix}\operatorname{vec}(\boldsymbol{S}_a) \\ \operatorname{vec}(\boldsymbol{S}_b)\end{bmatrix}\right) && \boldsymbol{u} = \boldsymbol{L}(\boldsymbol{f}-\boldsymbol{g}).\end{aligned}$$

Figure 3.1 illustrates the result of such a projection.

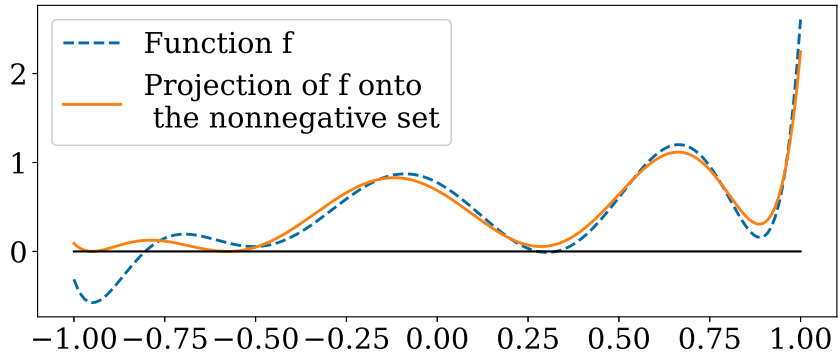


Fig. 3.1 Projection of a degree 8 polynomial onto the set of polynomials nonnegative on $[-1, 1]$.

3.2.2 Projection onto nonnegative splines

Splines are characterized by their degree and their number of interior knots. In this work we focus on splines of degree 3. In Section 2.3.2 we showed that using B-Spline basis, it necessary and sufficient to impose constraint (2.21) over all intervals to ensure the nonnegativity of the spline created by the coefficients $\boldsymbol{g}$. If k interior knots are used, the basis contains $k+2$ elements ($\boldsymbol{g} \in \mathbb{R}^{k+2}$) over the $k-1$ nonzero intervals and the problem is:

$$
\begin{aligned}
&\min t \\
\text{s.t}\ \ &(\boldsymbol{u}, t) \in \mathbb{L}^{k+2} \\
&\boldsymbol{q}_{1i} \in \mathcal{Q}_r, \boldsymbol{q}_{2i} \in \mathcal{Q}_r && \forall i = 1, \cdots k-1 \\
&\boldsymbol{g} \in \mathbb{R}^{k+2} \\
&\begin{bmatrix} \boldsymbol{g}_i \\ \boldsymbol{g}_{i+1} \\ \boldsymbol{g}_{i+2} \\ \boldsymbol{g}_{i+3} \end{bmatrix} = \boldsymbol{N}_i^{-1} \boldsymbol{R}_S \begin{bmatrix} \boldsymbol{q}_{1i} \\ \boldsymbol{q}_{2i} \end{bmatrix} && \forall i = 1, \cdots k-1 \\
&\boldsymbol{u} = \boldsymbol{L}(\boldsymbol{g} - \boldsymbol{f}).
\end{aligned}
$$

An example of projection using this method is presented in Figure 3.2, together with another approach that imposes the B-Spline coefficients to be

nonnegative, but not the spline itself (as proposed in [8] and [141], 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 [31]. Therefore, this second approach is less accurate than the projection described above, which can be observed in the figure.

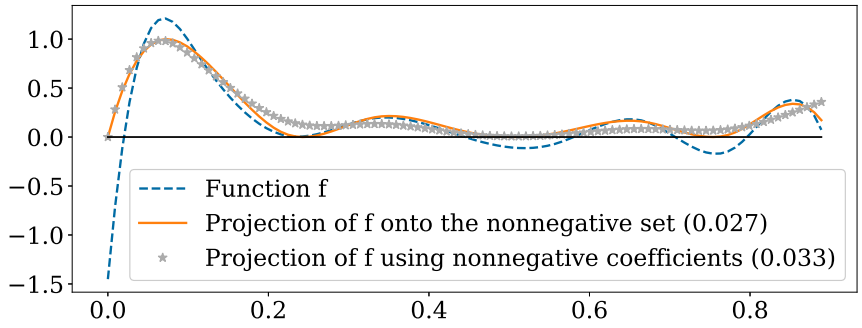


Fig. 3.2 Example of projection onto the set of splines nonnegative on $[0,1]$, using a spline with 9 equally spaced interior knots. The number in parenthesis is the value of the cost function (3.9).

3.3 Experimental results

In this section, we assess the performance of the LP-HALS algorithm (Algorithm 3.1) through experiments over both synthetic and real signals. Our method is compared with standard HALS method and approaches from previous works using also polynomials or splines in the factors.

We work on a synthetic dataset created as follows. Matrix $\bar{\boldsymbol{X}} \in \mathbb{R}^{n \times r}$ is randomly generated using a normal distribution $\mathcal{N}(0,1)$ with negative values replaced by zero. Matrix $\bar{\boldsymbol{A}}$ 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 [77][1]). Discretization points are equally spaced 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). The input data to factorize is then $\boldsymbol{Y} = \bar{\boldsymbol{A}}\bar{\boldsymbol{X}}^\top + \boldsymbol{N}$ where

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

$\boldsymbol{N} \in \mathbb{R}^{m \times n}$ is an additive Gaussian noise with a chosen signal-to-noise ratio (SNR) (see Section 2.1.1 for more details). Using such an artificially created synthetic dataset lets us assess the error compared to the ground truth of input signals: $\boldsymbol{Y}^* = \bar{\boldsymbol{A}}\bar{\boldsymbol{X}}^\top$, denoted as the relative residual. This was described in more details in Section 2.1.1.

In our experiments, we use the Chebyshev basis to represent polynomials and B-Splines to represent splines. We work with the Chebyshev basis instead of the monomial basis to have better conditioned matrices and thus avoid some numerical issues. Our algorithms are compared to several other methods constraining the columns of matrix $\boldsymbol{A}$ in problem (2.1) to be the discretization of continuous functions. These approaches are thus comparable to the LP-NMF problem using the sum cost. We implemented these methods based on the cited papers, and list them below:

- **HALS:** standard HALS algorithm applied on (2.1) (Algorithm 2.1).
- **P-LS:** $\boldsymbol{A}$ contains nonnegative polynomials, represented using **unconstrained** parameters, using equation (2.13). $\boldsymbol{X}$ is also factorized using unconstrained parameters as $\boldsymbol{X}_{ij} = \boldsymbol{C}_{ij}^2$. Problem is then solved using a nonlinear Least Squares solver [33]. We use in this work the function `least_squares` from python [2] with default parameters. The problem is thus solved using a trust region reflective algorithm.
- **PS-HALS and PI-HALS:** our LP-HALS algorithm using polynomials, respectively with **sum cost** and with **integral cost**.
- **S-MU:** $\boldsymbol{A}$ contains splines with **nonnegative** B-Splines **coefficients**. $\boldsymbol{A}$ contains thus nonnegative splines but can not contain all existing nonnegative splines. Problem is solved using Multiplicative Updates [141].
- **S-ADMM:** $\boldsymbol{A}$ contains splines **constrained** to be nonnegative at the considered sampling points. Problem is solved using the Alternative Direction Method of Multipliers [140], see Section 2.2 for more details.
- **SS-HALS and SI-HALS:** our LP-HALS algorithm using splines, respectively with **sum cost** and with **integral cost**.

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

Projections in LP-HALS are computed using conic programs solved by MOSEK fusion version 9 [6]. Algorithms are stopped when cost function $\|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}^\top\|$ no longer improves, namely when $|\text{previous cost} - \text{cost}|/\text{cost} < 10^{-7}$.

3.3.1 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 use in LP-HALS functions parametrized with 21 coefficients: polynomials of degree 20 and splines with 19 interior knots regularly distributed in $[-1, 1]$. Basis signals in original factor $\boldsymbol{A}$ are either polynomials or splines with 21 coefficients (Figure 3.3) or real reflectance signals of olivine, spessartine and hypersthene (Figure 3.4). When original factor $\boldsymbol{A}$ contains polynomials or splines without noise 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 on 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. However, matrix $\boldsymbol{M}$ is always computed using integrals for the integral case, as the basis functions are perfectly known. When noise is added to the data, its SNR is equal to 20 dB.

In Figure 3.3 we test the performance of our algorithms using the appropriate functional set $\mathcal{F}$ (polynomials if the original factor $\boldsymbol{A}$ contains polynomials and splines if the original factor $\boldsymbol{A}$ contains splines). 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. They are also less similar to the original signals. This is due to worse filtering of the noise, as indicated by the higher relative residual. 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 a 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).

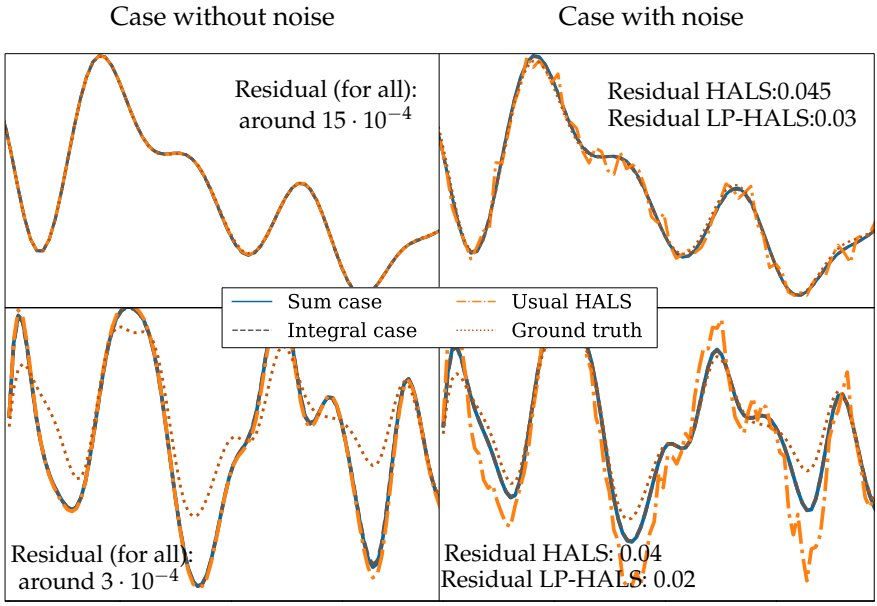


Fig. 3.3 Example of recovered signals. Top: polynomials, Bottom: splines, Left: case without noise, Right: case with noise level 20dB. Our algorithm obtains very similar performances in integral and sum case.

If we observe the recovered signals when $\boldsymbol{A}$ contains real reflectance signals in Figure 3.4, we observe again that the signals recovered by HALS are non-smooth in cases with noise. The relative residual is also worse as it is around 0.04, instead of 0.02 for the others. However, HALS is much better in the noiseless case with a residual around $4 \cdot 10^{-4}$ 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 do not change much between the noise-free and the noisy case, while they are very different between these two cases for HALS. This suggests that our methods are less sensitive to noise than HALS.

The observations made in this section are based over one test which is not enough to draw firm conclusions. In the next section we present tests made several times, to have a better idea of the performance of the algorithms. Moreover, we compare our methods to other similar approaches.

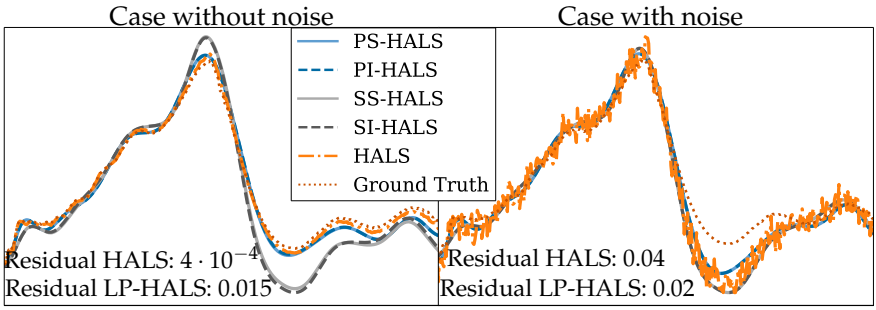


Fig. 3.4 Example of recovered signals of spessartine. Left: noiseless case, Right: noisy case. PS-HALS and PI-HALS obtains very similar performance, like SS-HALS and SI-HALS.

3.3.2 Comparison between sum and integral cases

We compare the performance of our algorithms using the sum or the integral costs. Figure 3.5 present the results when data contains $r = 3$ polynomials of degree 12 with $n = 100$ observations, sampled at $m = [25, 50, 100, 150, 250, 500]$ equally spaced discretization points on $[-1, 1]$ for sum case. Data does not contain noise. LP-HALS uses polynomials of degree 12 or splines with 11 interior knots. Residues are computed over 15000 points equally spaced in $[-1, 1]$.

On polynomial signals, when no noise is added to the data, the integral cost provides better results for both polynomials and splines when few discretization points are available. This is expected as the integrals contain the perfect information about the data, unlike the discretization. However, as soon as 150 discretization points are available, the behavior of the two cost functions becomes very similar.

In Figure 3.6, 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. LP-HALS uses polynomials of degree 20 or splines with 19 interior knots. Due to noise, the integrals in $\boldsymbol{Z}$ must be approximated. We use a piecewise interpolation of rank 1 of the input data.

In this case, the sum case using splines seems slightly better than the integral one. On the other hand, the integral case using polynomials is better

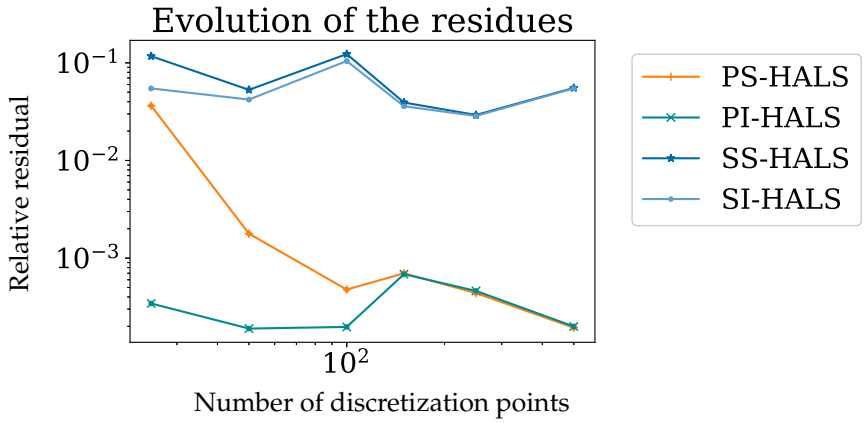


Fig. 3.5 Performance of our algorithms over polynomial signals. No noise is added. Average over 10 problems. Axes are in logarithmic scale.

than the sum case when very few discretization points are available, but the two approaches become very close when the number of discretization points increases. During our experimentation we observed that the approach used to approximately compute the integrals required for the integral case has a large influence on the final results. 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, especially when the number of discretization points is large enough. Therefore, based on the results obtained in these tests, we recommend using the integral cost when the input functions are known and the sum cost if they are provided as vectors.

3.3.3 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.

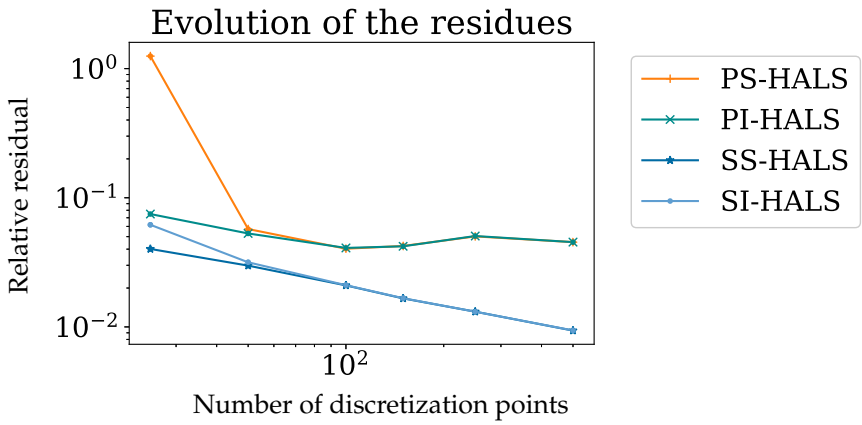


Fig. 3.6 Performance of our algorithms over spline signals with noise of 20 dB. Average over 10 problems. Axes are in logarithmic scale.

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 3.7 illustrates that one

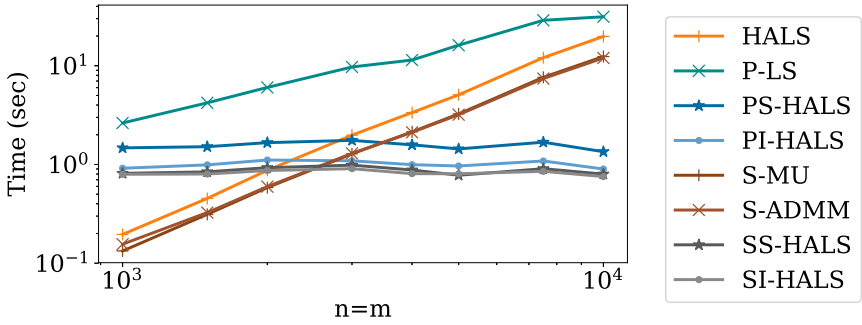


Fig. 3.7 Time for 20 iterations without initialization for increasing $n = m$. S-MU and S-ADMM display very similar performances, as SS-HALS and SI-HALS.

iteration of HALS, S-MU and S-ADMM has $\mathcal{O}(mn)$ complexity, while for LS it is only $\mathcal{O}(n)$ [33]. 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. Moreover, all 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.

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

Performance when the number of coefficients varies

We analyze the influence of the size of the chosen parametrizable set $\mathcal{F}$, i.e. we study the influence of the degree for polynomials or the number of interior knots for splines. In Figure 3.8, 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 LP-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 positive semidefinite matrices in the SDP. 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 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 a slightly higher residual than our methods, unlike S-MU that obtains similar residues. S-MU is also much faster than our methods. When we look at the best recovery of the used re-

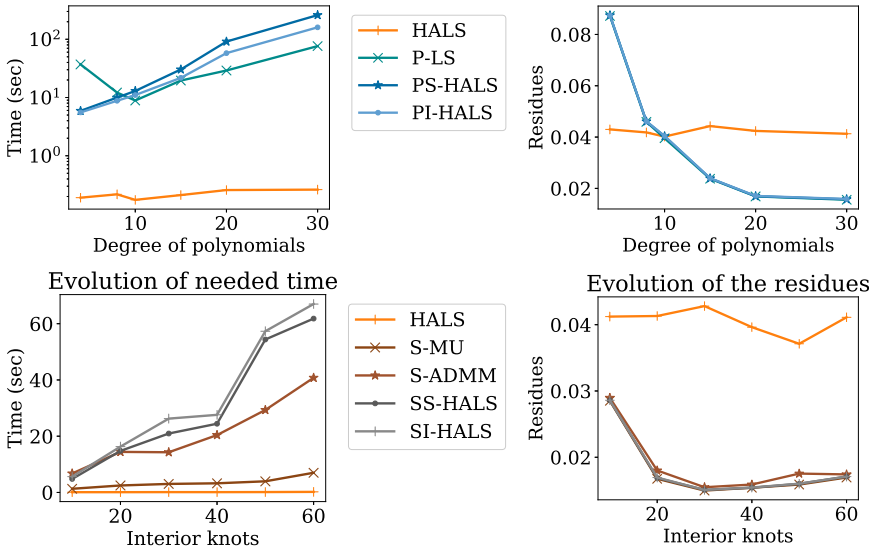


Fig. 3.8 Performance for increasing degree (Top) or increasing number of interior knots (Bottom). Left: needed time, Right: residues. S-MU obtains similar residues than our methods.

flectance 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 compare 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 3.2). Nevertheless, it is interesting to notice that using splines with nonnegative coefficients can be accurate in some situations. Note that this idea could also be used in our LP-HALS approach and would accelerate its projection step. It will be explored in Chapters 5 and 7.

Performance over noisy data

We now pay attention to the performance of the algorithms over various levels of noise on the data, as displayed in Figure 3.9. We observe that when using polynomials or splines in factor $\boldsymbol{A}$, obtained residues 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, except for P-LS that is much faster when data is more

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 3.2 Performance of S-MU and SS-HALS over nonnegative spline signals with negative coefficients in the B-Spline basis. Noise level is 20dB, $r = 3$, $n = m = 500$, and the 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. SIR measure has been presented in Section 2.1.1 and should be as high as possible. Res are the relative residuals multiplied by 10^3, that should be as low as possible.

noisy.

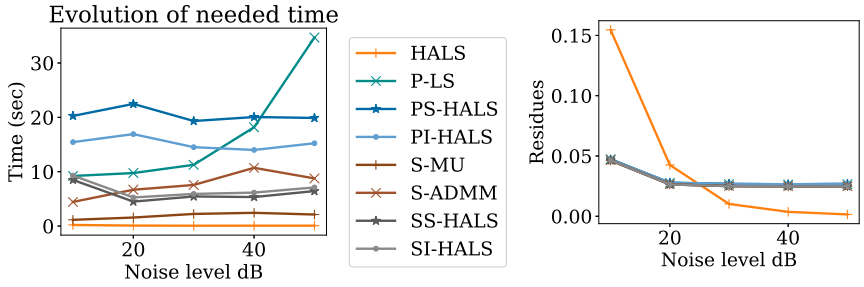


Fig. 3.9 Performance for different noise levels. Rightmost value is the noise-free situation (replacing the 50 dB mark). All algorithms obtain similar residues except HALS.

Detailed analysis of tests with reflectance signals

Table 3.3 contains the average results of the considered algorithms over $n = 250$ observations of reflectance signals using 10 random mixing matrices $\bar{\boldsymbol{X}}$ to build data $\boldsymbol{Y}$. Each algorithm is tested on the 10 inputs using 10 different starting values (100 tests in total). Noise level is 20dB on which functional NMF provides good results (see Figure 3.9). Moreover, based on results from Subsection 3.3.3, we used polynomials of degree 20 and splines with 30 interior knots. We observe that splines obtain better residues than polynomials that obtains better residues than HALS. This is not so surprising as the chosen splines have a higher degree of freedom than the chosen polynomials. Nevertheless, this means that splines managed to avoid overfitting of the noise. Moreover, methods using splines are generally faster than methods using polynomials.

Method	Time	Its	SIR_A	SIR_X	$SIR\ LC_A$	$SIR\ LC_X$	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 3.3 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. SIR and SIR LC have been presented in Section 2.1.1 and should be as high as possible. Res are the residues multiplied by 10^3, that should be as low as possible. The boxplots of the residues for each method are showed in Figure 3.10.

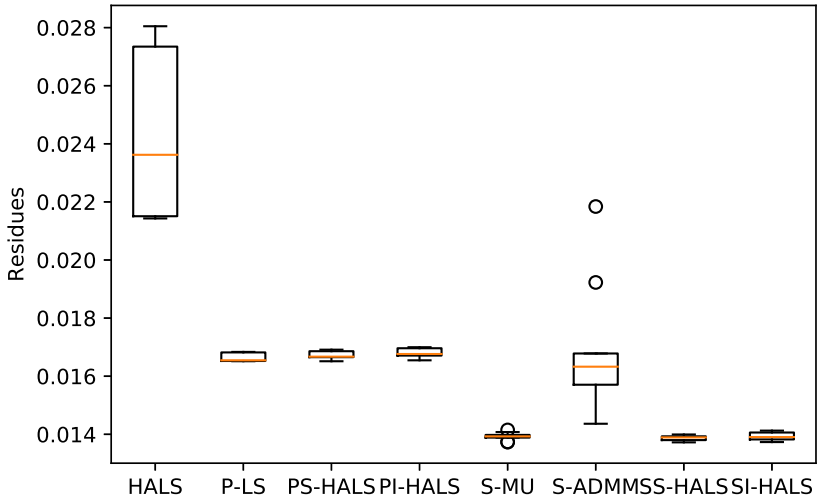


Fig. 3.10 Boxplot of the obtained residues for each method in Table 3.3.

We see also that S-ADMM obtains a significantly higher residual 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. Nevertheless, our method using splines in the sum case leads to

the best residual.

The method leading on average to the closest signals to the ground truth is P-LS. In general, HALS-based methods seem to obtain matrices $\boldsymbol{A}$ and $\boldsymbol{X}$ with worse SIR than the other methods when we assess them by looking only at permutation and rescaling 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 of obtained matrices (SIR LC).

We can observe in Figure 3.11 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 in the presented case.

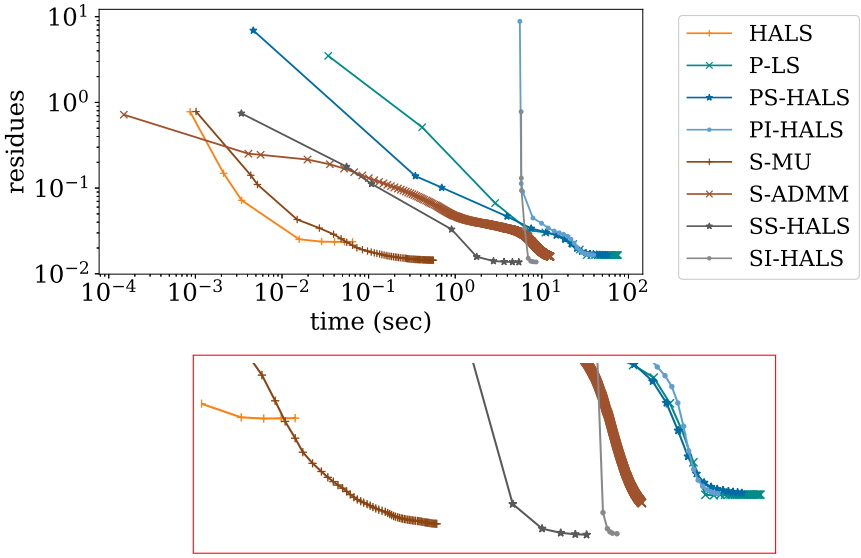


Fig. 3.11 Evolution and zoom on this evolution of the residues for each method with respect to time. A point is plotted every 10 iterations. The y-scale of the zoom is no longer in log-scale to improve readability. Polynomial methods converge in a similar way, while ADMM is the only spline-based method stopping with a residual similar to the polynomial ones.

Discussion

The tests we performed demonstrate that extending the NMF framework to handle polynomial or spline signals enables the recovery of smoother features and is less sensitive to noise when compared to standard NMF applied to discretized signals.

We have adapted the HALS algorithm to this extension. Our new LP-HALS algorithm requires the ability to project over sets of nonnegative linearly 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 important 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. A too low degree affects negatively the precision of the algorithms, while a too high degree has a low impact on the precision, but impacts greatly the computational time. The degree of parametrization is a hyperparameter, and its ideal value can therefore be found with usual techniques such as cross validation. However, it is worth keeping in mind that linear combinations of polynomials and splines do not affect the degree of parametrization. Thus, a reasonable approach to find the most suitable degree is to take some signals from the input data, and analyze their degree, taking into account the fact that the data may be noisy. Finally, when hesitating between several degrees, it is good to remember that a degree that is too high does reduce the accuracy much, but mostly increases the time needed (and conversely for a too low degree). Choosing a degree that is a little too high is therefore rarely a problem in practice.

Moreover, 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).

4

Extending NMF to Hilbert spaces (H-NMF)

In the previous chapter we explored with success the possibility to impose a factor of NMF to be derived from nonnegative polynomials or splines. Such a problem can be solved on matrices, by sampling the polynomials and splines, but we explained that it can also be used on functions, provided that it is possible to compute the integrals of each input function multiplied by each basis element of polynomials or splines. This chapter aims at generalizing the results from the previous chapter, and previous works on NMF using functions (e.g. polynomials, splines or GRBF). It presents a unifying framework that can handle a range of function classes as wide as possible, in Section 4.1. This unified framework is called Nonnegative Matrix Factorization on Hilbert spaces (H-NMF). An associated optimization problem, with a cost function based on inner products, is presented in Section 4.2. This section also presents some situations where the optimization problem can be simplified. Finally, in Section 4.3, we analyze usual algorithms for standard NMF and state the conditions needed to generalize them for H-NMF.

4.1 Generalizing the standard NMF problem

As its name suggests, the Nonnegative Matrix Factorization problem aims at describing an input matrix $\boldsymbol{Y}$ as a product of two nonnegative factors, the matrices $\boldsymbol{A}$ and $\boldsymbol{X}$: $\boldsymbol{Y} \simeq \boldsymbol{A}\boldsymbol{X}^\top$. To reduce the dimension of $\boldsymbol{A}\boldsymbol{X}^\top$ in comparison to $\boldsymbol{Y}$, $\boldsymbol{A}$ and $\boldsymbol{X}$ are imposed to have only r columns. Therefore, $\boldsymbol{Y}$ is expressed as the sum of r rank-one matrices: $\boldsymbol{Y} \simeq \sum_{k=1}^{r} \boldsymbol{A}_{:k}(\boldsymbol{X}_{:k})^\top$. Moreover, each column of $\boldsymbol{Y}$ is a nonnegative linear combination of r characteristic nonnegative factors, the columns of $\boldsymbol{A}$: $\boldsymbol{Y}_{:j} \simeq \sum_{k=1}^{r} \boldsymbol{A}_{:k}\boldsymbol{X}_{jk}$, and similarly for the rows of $\boldsymbol{Y}$, whose characteristic factors are the columns of $\boldsymbol{X}$: $\boldsymbol{Y}_{i:} \simeq \sum_{k=1}^{r} \boldsymbol{A}_{ik}(\boldsymbol{X}_{:k})^\top$. Therefore, each element $\boldsymbol{Y}_{ij}$ of $\boldsymbol{Y}$ is approximated as the sum of the products between the columns of $\boldsymbol{A}$ and $\boldsymbol{X}$ evaluated on i and j respectively, $\boldsymbol{Y}_{ij} \simeq \sum_{k=1}^{r} \boldsymbol{A}_{ik}\boldsymbol{X}_{jk}$. This problem is solved by minimizing the Frobenius norm of the reconstruction error $\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}^\top$ (other cost functions are possible but outside the scope of this work). It is described in Definition 4.1.

Definition 4.1. Nonnegative Matrix Factorization Given an input matrix $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$ and a factorization rank r, find nonnegative matrices $\boldsymbol{A} \in \mathbb{R}_+^{m\times r}$ and $\boldsymbol{X} \in \mathbb{R}_+^{n\times r}$ minimizing

$$\underset{\boldsymbol{A}\in\mathbb{R}_+^{m\times r},\, \boldsymbol{X}\in\mathbb{R}_+^{n\times r}}{\operatorname{argmin}} \left\|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}^\top\right\|_F^2 . \tag{4.1}$$

This leads to matrices $\boldsymbol{A}$ and $\boldsymbol{X}$ such that

$$\boldsymbol{Y} \simeq \boldsymbol{A}\boldsymbol{X}^\top = \sum_{k=1}^{r} \boldsymbol{A}_{:k}(\boldsymbol{X}_{:k})^\top \tag{4.2}$$

which is equivalent to

$$\boldsymbol{Y}_{:j} \simeq \sum_{k=1}^{r} \boldsymbol{A}_{:k}\boldsymbol{X}_{jk}\ \forall j \in \{1,\cdots,n\} \quad , \quad \boldsymbol{Y}_{i:} \simeq \sum_{k=1}^{r} \boldsymbol{A}_{ik}(\boldsymbol{X}_{:k})^\top\ \forall i \in \{1,\cdots,m\} \tag{4.3}$$

and

$$\boldsymbol{Y}_{ij} \simeq \sum_{k=1}^{r} \boldsymbol{A}_{ik}\boldsymbol{X}_{jk} \qquad \forall i \in \{1,\cdots,m\},\ j \in \{1,\cdots,n\}. \tag{4.4}$$

Our goal is now to generalize this problem to an input function of two variables $Y(a,x) : \mathbb{A} \times \mathbb{X} \mapsto \mathbb{R}$, lying in a Hilbert space $\mathcal{H}$. For matrices, those two variables are the indices of rows and columns, i.e. $\mathbb{A} = \{1,2,\dots,m\}$ and $\mathbb{X} = \{1,2,\dots n\}$. In general $\mathbb{A}$ and $\mathbb{X}$ can contain all kinds of sets such as indices, positions (e.g. for images), ranges of spectra (binding energy in chemistry or wavelengths in imaging), times ranges (for time series), etc.

Let us define $\otimes$ the tensor product between two univariate functions $A : \mathbb{A} \mapsto \mathbb{R}$ and $X : \mathbb{X} \mapsto \mathbb{R}$ as

$$\big[A \otimes X\big](a,x) = A(a)X(x) \quad \forall a \in \mathbb{A}, x \in \mathbb{X}. \tag{4.5}$$

We aim at describing the input function Y as a sum of r rank-one terms, and we consider that rank-one functions of two variables are separable functions, i.e. functions $R : \mathbb{A} \times \mathbb{X} \mapsto \mathbb{R}$ such that $R = R_a \otimes R_x$, with $R_a : \mathbb{A} \mapsto \mathbb{R}$ and $R_x : \mathbb{X} \mapsto \mathbb{R}$.

Let $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ be the considered constrained functions sets (with nonnegativity constraints for example). We aim to find $2r$ univariate functions $A_k \in \tilde{\mathcal{A}}, X_k \in \tilde{\mathcal{X}}\ \forall k = 1,\cdots,r$ that approximate the input data Y as $Y \simeq \sum_{k=1}^{r} A_k \otimes X_k$.

This means that, when x is fixed, each function $Y(:,x) : \mathbb{A} \mapsto \mathbb{R}$ is described as a linear combination of functions A_k, $Y(:,x) \simeq \sum_{k=1}^{r} A_k X_k(x)$, and similarly, when a is fixed, each function $Y(a,:) : \mathbb{X} \mapsto \mathbb{R}$ is described as a linear combination of functions X_k, $Y(a,:) \simeq \sum_{k=1}^{r} A_k(a) X_k$. Moreover, each element $Y(a,x)$ is approximated as the sum of the products between the functions A_k and X_k evaluated on a and x respectively, $Y(a,x) \simeq \sum_{i=k}^{r} A_k(a)X_k(x)$. To solve this problem, we suppose that both $\sum_{k=1}^{r} A_k \otimes X_k$ and Y belong to a Hilbert space $\mathcal{H}$, in order to minimize the norm of the reconstruction error $Y - \sum_{k=1}^{r} A_k \otimes X_k$. This leads us to a tentative definition for the H-NMF. This definition is provisional, since we will see in the next section the conditions necessary to implement it in practice. The actual definition of H-NMF is given in Definition 4.4.

Definition 4.2. NMF on Hilbert spaces (H-NMF) - provisional - Given two sets $\mathbb{A}$ and $\mathbb{X}$, two nonnegative sets of functions $\tilde{\mathcal{A}}$ defined on $\mathbb{A} \mapsto \mathbb{R}_+$ and $\tilde{\mathcal{X}}$ defined on $\mathbb{X} \mapsto \mathbb{R}_+$, an input function of two variables $Y : \mathbb{A} \times \mathbb{X} \mapsto \mathbb{R}$ and a factorization rank r. Find $2r$ nonnegative functions

$\{A_k\}_{k=1}^r$, with $A_k \in \tilde{\mathcal{A}}\ \forall k$, and $\{X_k\}_{k=1}^r$, with $X_k \in \tilde{\mathcal{X}}\ \forall k$, minimizing

$$\underset{A_k \in \tilde{\mathcal{A}},\ X_k \in \tilde{\mathcal{X}}\ \forall i}{\operatorname{argmin}} \left\| Y - \sum_{k=1}^{r} A_k \otimes X_k \right\|_{\mathcal{H}}^2 . \tag{4.6}$$

The goal is to obtain factors A_k, X_k such that

$$Y \simeq \sum_{k=1}^{r} A_k \otimes X_k \tag{4.7}$$

which is equivalent to

$$Y(:,x) \simeq \sum_{k=1}^{r} A_k X_k(x) \quad \forall x \in \mathbb{X} \quad , \quad Y(a,:) \simeq \sum_{k=1}^{r} A_k(a) X_k \quad \forall a \in \mathbb{A} \tag{4.8}$$

and

$$Y(a,x) \simeq \sum_{k=1}^{r} A_k(a) X_k(x) \qquad \forall x \in \mathbb{X},\ a \in \mathbb{A}. \tag{4.9}$$

Remark 4.1. Having nonnegative functions in $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ provides an additive representation of Y. However, most of the results presented in next sections do not rely on this nonnegativity. Therefore, we can define a similar problem where sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ are not imposed to contain nonnegative functions: the Constrained Matrix Factorization on Hilbert spaces problem (H-CMF).

Figure 4.1 bellow illustrates this problem on the spectrum images dataset from [79]. Spectrum images contain spectral ($\mathbb{A}$) and spatial ($\mathbb{X}$) information of mixture of chemical elements. Each pixel of Y, $Y(:,x)$, contains the mixture of the binding energy levels of a few chemical elements whose binding energy level is contained in functions A_k. At the same time, each binding energy level of Y, $Y(a,:)$, has an intensity map that is the mixture of the intensity maps of the chemical elements, contained in the functions X_k.

Examples of H-NMF problems

- In standard NMF, $\mathbb{A} = \{1, \cdots, m\}$, $\mathbb{X} = \{1, \cdots, n\}$ and $\tilde{\mathcal{A}} = \{A : \mathbb{A} \mapsto \mathbb{R}_+\} = \mathbb{R}_+^m$, $\tilde{\mathcal{X}} = \mathbb{R}_+^n$.

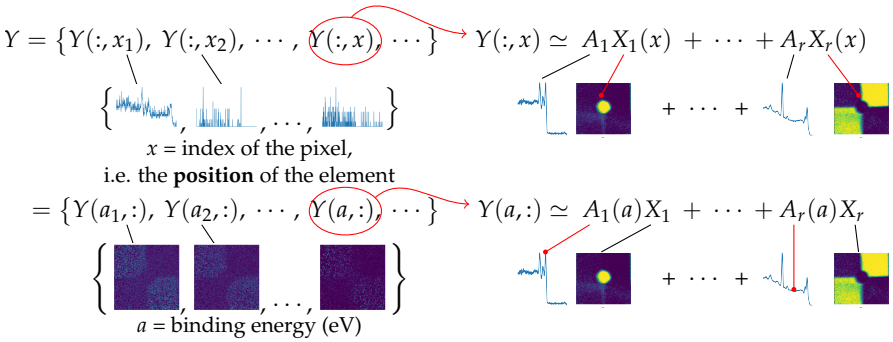


Fig. 4.1 Illustration of H-NMF on the spectrum images dataset from [79].

- Consider that A contains discretization of functions as in [8, 33, 133, 139, 141], or in Chapter 3 (the sum cost), with $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^m$ the set of discretization points and $\mathcal{F}$ the set of considered functions, while X contains vectors. We have $\mathbb{A} = \boldsymbol{\tau}$, $\tilde{\mathcal{A}} = \{f(\boldsymbol{\tau}) | f \in \mathcal{F}\}$, $\mathbb{X} = \{1, \cdots, n\}$ and $\tilde{\mathcal{X}} = \mathbb{R}_+^n$.
- In Figure 4.1, $\mathbb{A}$ is a range of binding energy levels (and therefore contains an infinite number of points) and $\mathbb{X}$ contains the possible positions of the elements (that are two-dimensional). We can consider several constraints in $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$, one example is to consider that $\tilde{\mathcal{A}}$ is a set of nonnegative splines to impose local smoothness on the factors A_k, and to consider only the nonnegative constraint in $\tilde{\mathcal{X}}$: $\tilde{\mathcal{X}} = \{X : \mathbb{X} \mapsto \mathbb{R}_+\}$.
- Consider a hyperspectral image from which we aim to identify the constituent endmembers and their abundance maps (see e.g. [45] for an example of NMF on this kind of dataset). In this case, $\mathbb{A}$ is a range of wavelengths (that can be finite or not) and $\mathbb{X}$ contains the positions of the pixels. We can consider the same constraints as in previous case.

4.2 Optimizing the H-NMF problem using inner products

In this section, we analyze the conditions under which Definition 4.2 is suitable. We also discuss some properties of the cost function (4.6), com-

ment the choice of sets $\mathcal{A}$, $\mathcal{X}$ and constraints $\tilde{\mathcal{A}}$, $\tilde{\mathcal{X}}$ in H-NMF, and present situations where the problem can be simplified.

4.2.1 Needed theory and assumptions for H-NMF

We suppose that input data Y belongs to a Hilbert space $\mathcal{H}$. This implies the existence of a inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and thus of a norm $\| \cdot \|_{\mathcal{H}}$, and allows us to define a cost function. Nevertheless, as shown in Equation (4.9), the goal of H-NMF is to obtain a factorization that is pointwise close to the input data by minimizing the norm of the error. However, it is not always true that functions close in norm are also close pointwise. Consider for example the $L_2^{[-1,1]}$ space, i.e. the space of square integrable functions on domain $[-1, 1]$, with inner product $< f, g >= \int_{-1}^{1} f(a)g(a)\, \mathrm{d}a$. Let $f, g \in L_2^{[-1,1]}$ such that $f(a) = \begin{cases} g(a) & \text{if } a \neq 0 \\ g(0) + M & \text{else} \end{cases}$ with M arbitrary large. Then $\|f - g\| = 0$, while $|f(0) - g(0)| = M$ for some arbitrary parameter M.

Actually, $L_2^{[-1,1]}$ is not a Hilbert space, because it contains elements like $f - g$ that have zero-norm but are not equal to 0. We could work instead on the Hilbert space of equivalence classes of functions in $L_2^{[-1,1]}$. In this Hilbert space, f and g belong to the same class, and are thus considered as equivalent, as they are equal almost everywhere. However, this is also not adequate in our case, because we need to be able to evaluate properly our functions at all points. We therefore consider function spaces with a bounded evaluation functional: the Reproducing Kernel Hilbert Spaces (RKHS) defined below.

Definition 4.3. Reproducing Kernel Hilbert Space (RKHS) Consider a set $\mathbb{A}$ and space $\mathcal{F} = \{f : \mathbb{A} \mapsto \mathbb{R}\}$, the space of all functions from $\mathbb{A}$ to $\mathbb{R}$. The space $\mathcal{A} \subseteq \mathcal{F}$ is a Reproducing Kernel Hilbert space if and only if

1. $\mathcal{A}$ is a Hilbert space, with inner product $\langle \cdot, \cdot \rangle_{\mathcal{A}}$, and
2. $\forall a \in \mathbb{A}$, the linear evaluation functional $\delta_a : \mathcal{A} \mapsto \mathbb{R}$, $\delta_a(f) = f(a)$ is a bounded operator.

RKHS have many interesting properties. Some of them are listed below.

Proposition 4.1. *If $\mathcal{A}$ is a RKHS on $\mathcal{F} = \{f : \mathbb{A} \mapsto \mathbb{R}\}$, then*

1. *$\forall a \in \mathbb{A}$, there is a unique function $k_a \in \mathcal{A}$ such that $f(a) = \langle f, k_a \rangle_{\mathcal{A}}$ $\forall f \in \mathcal{A}$. Function k_a is called the reproducing kernel for the point a.*

2. *The linear span of functions* $\{k_a\}_{a\in\mathbb{A}}$ *is dense in* $\mathcal{A}$.

3. *The function* $K : \mathbb{A}\times\mathbb{A} \mapsto \mathbb{R}$, $K(a,a') = k_a(a') = \langle k_a, k_{a'}\rangle_{\mathcal{A}}$ *is a kernel function, called the reproducing kernel of* $\mathcal{A}$. *This means that for every choice of* n *valid points* $\{a_i\}_{i=1}^n \subseteq \mathbb{A}$, *the matrix* $\boldsymbol{M}$, *defined by* $\boldsymbol{M}_{i,j} = K(a_i, a_j)$, *is positive semi-definite.*

4. *Every kernel function is the reproducing kernel of a RKHS, and this RKHS is unique.*

5. *Suppose* $\{f_n\} \subseteq \mathcal{A}$. *If there is* $f \in \mathcal{A}$ *such that* $\lim_{n\to\infty} \|f_n - f\| = 0$, *then* $f(a) = \lim_{n\to\infty} f_n(a)\ \forall a \in \mathbb{A}$.

Proof. More information about RKHS and kernel functions, as well as proofs of the above statements, can be found in [7, 99]. □

This last property indicates that it makes sense to consider iterative algorithms to solve the H-NMF problem on RKHS, as when the iterates converge in norm, they also converge pointwise, which is the expected desirable behavior.

In fact, it is relatively rare to encounter a Hilbert functional space that is not a RKHS. However, there are space of functions that are not Hilbert spaces, like $L_2^{[-1,1]}$ or the restriction of $L_2^{[-1,1]}$ to continuous functions. Nevertheless, many functions belong to a RKHS. In particular, polynomials of finite degree and splines with a finite number of fixed interior knots are RKHS as shown by lemma below.

Lemma 4.1. *Given* n *linearly independent basis functions* Π_i, *the span of these functions* $\mathcal{H} = \left\{\sum_{i=1}^n \alpha_i\Pi_i\ \forall\alpha \in \mathbb{R}^n\right\}$ *is a RKHS. This is the case for finite degree polynomials or splines with a fixed and finite number of interior knots.*

Proof. A more detailed proof can be found in Chapter 3 of [7]. We aim to find a reproducing kernel function for $\mathcal{H}$, $k_x = \sum_{i=1}^n \beta_i^x\Pi_i$, so that

$$\sum_{i=1}^n \alpha_i\Pi_i(x) = f(x) = \langle f, k_x\rangle = \Big\langle \sum_{i=1}^n \alpha_i\Pi_i, \sum_{i=1}^n \beta_i^x\Pi_i\Big\rangle_{\mathcal{H}} \qquad \forall\alpha_i, x.$$

Let $\boldsymbol{\alpha} = [\alpha_1 \ \cdots \ \alpha_n]^\top \in \mathbb{R}^n$ and $\boldsymbol{\Pi}(x) = [\Pi_1(x) \cdots \Pi_n(x)]^\top \in \mathbb{R}^n$, then

$$f(x) = \langle f, k_x \rangle \Leftrightarrow \boldsymbol{\alpha}^\top \boldsymbol{\Pi}(x) = \sum_{i,j} \alpha_i \beta_j^x \langle \Pi_i, \Pi_j \rangle_{\mathcal{H}} \qquad \forall \boldsymbol{\alpha} \in \mathbb{R}^n, \ \forall x.$$

Let $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ be the Gram matrix, $\boldsymbol{M}_{i,j} = \langle \Pi_i, \Pi_j \rangle_{\mathcal{H}}$. As functions Π_i are linearly independent, $\boldsymbol{M}$ is positive-definite. Let $\boldsymbol{\beta}^x = [\beta_1^x \ \cdots \ \beta_n^x]^\top \in \mathbb{R}^n$.

$$\begin{aligned} f(x) = \langle f, k_x \rangle &\Leftrightarrow \boldsymbol{\alpha}^\top \boldsymbol{\Pi}(x) = \boldsymbol{\alpha}^\top \boldsymbol{M} \boldsymbol{\beta}^x \qquad && \forall \boldsymbol{\alpha} \in \mathbb{R}^n, \ \forall x \\ &\Leftrightarrow \boldsymbol{\beta}^x = \boldsymbol{M}^{-1} \boldsymbol{\Pi}(x) && \forall x \in \mathbb{X}. \end{aligned}$$

We can therefore define the kernel function $K(x, y) = \boldsymbol{\Pi}(y)^\top \boldsymbol{M}^{-1} \boldsymbol{\Pi}(x)$, that is the reproducing kernel of $\mathcal{H}$, and $\mathcal{H}$ is a Reproducing Kernel Hilbert Space. □

We aim at describing input data Y using a sum of products $A_k \otimes X_k$, as defined in Equation (4.5). It turns out that if $A_k \in \mathcal{A} \ \forall k$ and $X_k \in \mathcal{X} \ \forall k$ with $\mathcal{A}$, $\mathcal{X}$ being two RKHS, the space of sums of products $A_k \otimes X_k$ is a RKHS as well. It is actually the tensor product of RKHS $\mathcal{A}$ and $\mathcal{X}$.

Theorem 4.2. ***Tensor product of RKHS*** *Consider two RKHS $\mathcal{A}$ and $\mathcal{X}$ defined on $\mathbb{A} \mapsto \mathbb{R}$ and $\mathbb{X} \mapsto \mathbb{R}$ respectively, and K_a, K_x, their reproducing kernels. The tensor product of $\mathcal{A}$ and $\mathcal{X}$, $\mathcal{L} = \mathcal{A} \otimes \mathcal{X}$ is also a RKHS, with reproducing kernel $K\big((a', x'), (a, x)\big) = K_a(a', a) K_x(x', x)$.*

Tensor product $\mathcal{L}$ is the completion of the set $\{\sum_{s=1}^{j} A_s \otimes X_s\}$ with j finite, $A_s \in \mathcal{A}$, $X_s \in \mathcal{X} \ \forall k$ and $\otimes$ the product from (4.5), i.e.

$$Y \in \mathcal{L} \Leftrightarrow Y = \lim_{j \to \infty} Y_j \text{ with } \{Y_j\} \text{ a Cauchy sequence and } Y_j = \sum_{s=1}^{j} A_s \otimes X_s \tag{4.10}$$

The inner product of space $\mathcal{L}$ is so that:

$$\langle A \otimes X, A' \otimes X' \rangle_{\mathcal{L}} = \langle A, A' \rangle_{\mathcal{A}} \langle X, X' \rangle_{\mathcal{X}} \tag{4.11}$$

Proof. This theorem is Theorem 5.24 from [99]. □

From the results obtained above, using RKHS in H-NMF makes sense and does not add strong constraints to the problem compared to using Hilbert spaces, while giving better theoretical assurances. Moreover, it also allows

us to write the cost function of the H-NMF problem in a different way, as we will see now. We first present a useful lemma for that.

Lemma 4.2. *Given two sets $\mathbb{A}$ and $\mathbb{X}$, two RKHS $\mathcal{A} \subseteq \{A : \mathbb{A} \mapsto \mathbb{R}\}$ and $\mathcal{X} \subseteq \{X : \mathbb{X} \mapsto \mathbb{R}\}$, with $\mathcal{L}$ their tensor product, i.e. $\mathcal{L} = \mathcal{A} \otimes \mathcal{X}$. Suppose $A \in \mathcal{A}$, $X \in \mathcal{X}$ and $Y \in \mathcal{L}$. Define $g : \mathbb{X} \mapsto \mathbb{R}$ with $g(x) = \langle Y(:, x), A\rangle_{\mathcal{A}}$ and $f : \mathbb{A} \mapsto \mathbb{R}$ with $f(a) = \langle Y(a, :), X\rangle_{\mathcal{X}}$.*

Then $g \in \mathcal{X}$, $f \in \mathcal{A}$ and

$$\langle Y, A \otimes X\rangle_{\mathcal{L}} = \langle g, X\rangle_{\mathcal{X}} = \langle f, A\rangle_{\mathcal{A}}. \tag{4.12}$$

Proof. By Equation (4.10), we have $Y = \lim_{j\to\infty} Y_j$ with $\{Y_j\}$ a Cauchy sequence and $Y_j = \sum_{s=1}^{j} A_s \otimes X_s$.

Let us define the sequence of $a_j = \langle Y_j, A \otimes X\rangle_{\mathcal{L}}$. By Cauchy-Schwarz inequality, we have

$$|a_j - a_m| = |\langle Y_j - Y_m, A \otimes X\rangle_{\mathcal{L}}| \leq \|Y_j - Y_m\|_{\mathcal{L}} \|A \otimes X\|_{\mathcal{L}}.$$

As $\{Y_j\}$ is a Cauchy sequence, so is $\{a_j\}$ and $\{a_j\}$ converges to $\langle Y, A \otimes X\rangle$. Moreover, by definition of the inner product of $\mathcal{L}$ (4.11) and the linearity of the inner product we have

$$a_j = \sum_{s=1}^{j} \langle A_s, A\rangle_{\mathcal{A}} \langle X_s, X\rangle_{\mathcal{X}} = \Big\langle \sum_{s=1}^{j} \langle A_s, A\rangle_{\mathcal{A}} X_s, X \Big\rangle_{\mathcal{X}}. \tag{4.13}$$

We now define $g_j \in \mathcal{X}$ as $g_j = \sum_{s=1}^{j} \langle A_s, A\rangle_{\mathcal{A}} X_s$. Sequence $\{g_j\}$ is a Cauchy sequence. Indeed,

$$\begin{aligned}
\|g_j - g_m\|_{\mathcal{X}}^2 &= \sum_{s_1=m}^{j} \sum_{s_2=m}^{j} \langle A_{s_1}, A\rangle_{\mathcal{A}} \langle A_{s_2}, A\rangle_{\mathcal{A}} \langle X_{s_1}, X_{s_2}\rangle_{\mathcal{X}} \\
&= \sum_{s_1=m}^{j} \sum_{s_2=m}^{j} \langle A_{s_1} \otimes X_{s_1}, A \otimes X_{s_2}\rangle_{\mathcal{L}} \langle A_{s_2}, A\rangle_{\mathcal{A}} \\
&= \Big\langle Y_j - Y_m, \sum_{s=m}^{j} A \otimes X_s \langle A_s, A\rangle_{\mathcal{A}} \Big\rangle_{\mathcal{L}} \\
&\leq \|Y_j - Y_m\|_{\mathcal{L}} \Big\| \sum_{s=m}^{j} A \otimes X_s \langle A_s, A\rangle_{\mathcal{A}} \Big\|_{\mathcal{L}} \quad \text{Cauchy-Schwarz ineq.}
\end{aligned}$$

We have

$$\Big\| \sum_{s=m}^{j} A \otimes X_s \langle A_s, A \rangle_{\mathcal{A}} \Big\|_{\mathcal{L}}^2 = \sum_{s_1=m}^{j} \sum_{s_2=m}^{j} \|A\|_{\mathcal{A}}^2 \langle X_{s_1}, X_{s_2} \rangle_{\mathcal{X}} \langle A_{s_1}, A \rangle_{\mathcal{A}} \langle A_{s_2}, A \rangle_{\mathcal{A}}$$

$$= \|A\|_{\mathcal{A}}^2 \Big\langle Y_j - Y_m, \sum_{s=m}^{j} A \otimes X_s \langle A_s, A \rangle_{\mathcal{A}} \Big\rangle_{\mathcal{L}}$$

$$\Big\| \sum_{s=m}^{j} A \otimes X_s \langle A_s, A \rangle_{\mathcal{A}} \Big\|_{\mathcal{L}} \leq \|A\|_{\mathcal{A}}^2 \|Y_j - Y_m\|_{\mathcal{L}} \quad \text{Cauchy-Schwarz inequality.}$$

This allows us to say that $\|g_j - g_m\|_{\mathcal{X}} \leq \|Y_j - Y_m\|_{\mathcal{L}} \|A\|_{\mathcal{A}}$. As $\{Y_j\}$ is a Cauchy sequence, so is $\{g_j\}$, and $\{g_j\}$ converges to $g \in \mathcal{X}$.

As $\mathcal{X}$ is a RKHS, by proposition 4.1.5

$$g(x) = \lim_{j \to \infty} g_j(x) = \lim_{j \to \infty} \langle \sum_{s=1}^{j} A_s X_s(x), A \rangle_{\mathcal{A}} = \lim_{j \to \infty} \langle Y_j(:, x), A \rangle_{\mathcal{A}}.$$

As $\{Y_j\}$ is a Cauchy sequence, $g(x) = \langle Y(:, x), A \rangle_{\mathcal{A}}$.

As $\{g_j\}$ is a Cauchy sequence, sequence $\{a_j\}$ converges to $\langle g, X \rangle_{\mathcal{A}}$. Therefore we have $\langle Y, A \otimes X \rangle_{\mathcal{L}} = \langle g, X \rangle_{\mathcal{X}}$ with $g \in \mathcal{X}$ and $g(x) = \langle Y(:, x), A \rangle_{\mathcal{A}}$. To conclude the proof, a very similar reasoning can be done to find the result using $f : \mathbb{A} \mapsto \mathbb{R}$ with $f(a) = \langle Y(a, :), X \rangle_{\mathcal{X}}$. □

This lemma allows us to express the cost function of the H-NMF problem in three different ways, in the theorem below.

Theorem 4.3. *Given two sets $\mathbb{A}$ and $\mathbb{X}$, two RKHS $\mathcal{A} \subseteq \{A : \mathbb{A} \mapsto \mathbb{R}\}$ and $\mathcal{X} \subseteq \{X : \mathbb{X} \mapsto \mathbb{R}\}$ and a RKHS $\mathcal{H} \subseteq \{Y : \mathbb{A} \times \mathbb{X} \mapsto \mathbb{R}\}$, containing the tensor product of $\mathcal{A}$ and $\mathcal{X}$, i.e. $\mathcal{L} = \mathcal{A} \otimes \mathcal{X} \subseteq \mathcal{H}$. Suppose $\mathcal{H}$ and $\mathcal{L}$ have matching inner product, i.e. $\langle \cdot, \cdot \rangle_{\mathcal{H}} = \langle \cdot, \cdot \rangle_{\mathcal{L}}$ on points in $\mathcal{L}$.*

Considering Y_1 the projection of Y on $\mathcal{L}$, the cost function

$$\Big\| Y - \sum_{k=1}^{r} A_k \otimes X_k \Big\|_{\mathcal{H}}^2 \tag{4.14}$$

is equivalent to both

$$\|Y\|_{\mathcal{H}}^2 + \sum_{k_1=1}^{r}\sum_{k_2=1}^{r}\langle A_{k_1}, A_{k_2}\rangle_{\mathcal{A}}\langle X_{k_1}, X_{k_2}\rangle_{\mathcal{X}} - 2\sum_{k=1}^{r}\langle f_k, A_k\rangle_{\mathcal{A}} \\ \text{with } f_k \in \mathcal{A};\ f_k(a) = \langle Y_1(a,:), X_k\rangle_{\mathcal{X}}\ \forall a \in \mathbb{A} \tag{4.15}$$

and

$$\|Y\|_{\mathcal{H}}^2 + \sum_{k_1=1}^{r}\sum_{k_2=1}^{r}\langle A_{k_1}, A_{k_2}\rangle_{\mathcal{A}}\langle X_{k_1}, X_{k_2}\rangle_{\mathcal{X}} - 2\sum_{k=1}^{r}\langle g_k, X_k\rangle_{\mathcal{X}} \\ \text{with } g_k \in \mathcal{X};\ g_k(x) = \langle Y_1(:,x), A_k\rangle_{\mathcal{A}}\ \forall x \in \mathbb{X}. \tag{4.16}$$

Proof. As $\mathcal{L}$ is a RKHS, it is a closed subspace of $\mathcal{H}$. Using $\mathcal{L}^\top$ the orthogonal complement of $\mathcal{L}$, any element $Y \in \mathcal{H}$ can be written as

$$Y = Y_1 + Y_2 \qquad Y_1 \in \mathcal{L},\ Y_2 \in \mathcal{L}^\perp. \tag{4.17}$$

The decomposition is unique and Y_1 is actually the projection of Y on $\mathcal{L}$: $Y_1 = \text{argmin}_{Y_1 \in \mathcal{L}}\|Y - Y_1\|_{\mathcal{H}}$ [36]. Equation (4.14) can be written as

$$\Big\|Y - \sum_{k=1}^{r} A_k \otimes X_k\Big\|_{\mathcal{H}}^2 = \|Y\|_{\mathcal{H}}^2 + \Big\|\sum_{k=1}^{r} A_k \otimes X_k\Big\|_{\mathcal{H}}^2 - 2\Big\langle Y, \sum_{k=1}^{r} A_k \otimes X_k\Big\rangle_{\mathcal{H}}.$$

Using (4.17) this is equal to

$$\Big\|Y\Big\|_{\mathcal{H}}^2 + \sum_{k_1,k_2=1}^{r}\big\langle A_{k_1}\otimes X_{k_1}, A_{k_2}\otimes X_{k_2}\big\rangle_{\mathcal{H}} - 2\sum_{k=1}^{r}\Big(\big\langle Y_1, A_k\otimes X_k\big\rangle_{\mathcal{H}} + \langle Y_2, A_k\otimes X_k\rangle_{\mathcal{H}}\Big).$$

As $Y_2 \in \mathcal{L}^\perp$ and $A_k \otimes X_k \in \mathcal{L}$, the inner product $\langle Y_2, A_k \otimes X_k\rangle_{\mathcal{H}}$ is equal to zero $\forall k$. The proof is concluded using Lemma 4.2. □

We observe that, in both cases, we can work on Y_1, the projection of the input data Y on $\mathcal{L} = \mathcal{A} \otimes \mathcal{X}$, using inner products $\langle\cdot,\cdot\rangle_{\mathcal{A}}$ and $\langle\cdot,\cdot\rangle_{\mathcal{X}}$ on chosen spaces $\mathcal{A}$ and $\mathcal{X}$, instead of inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$. Note that $\mathcal{A}$ and $\mathcal{X}$ are not the constraints on functions A_k and X_k. Instead, spaces $\mathcal{A}$ and $\mathcal{X}$ should contain the constraints $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$: $\tilde{\mathcal{A}} \subseteq \mathcal{A}$ and $\tilde{\mathcal{X}} \subseteq \mathcal{X}$.

With the information obtained in this section, and especially the results from Theorem 4.3 we can now define the H-NMF problem more precisely.

Definition 4.4. NMF on Hilbert space Given two sets $\mathbb{A}$ and $\mathbb{X}$, two RKHS $\mathcal{A} \subseteq \{A : \mathbb{A} \mapsto \mathbb{R}\}$ and $\mathcal{X} \subseteq \{X : \mathbb{X} \mapsto \mathbb{R}\}$ and two nonnegative functions sets $\tilde{\mathcal{A}} \subseteq \mathcal{A}$ defined on $\mathbb{A} \mapsto \mathbb{R}_+$ and $\tilde{\mathcal{X}} \subseteq \mathcal{X}$ defined on $\mathbb{X} \mapsto \mathbb{R}_+$. Given input function $Y : \mathbb{A} \times \mathbb{X} \mapsto \mathbb{R}$, so that $Y \in \mathcal{H}$, with $\mathcal{H}$ a RKHS containing $\mathcal{L} = \mathcal{A} \otimes \mathcal{X}$, with matching inner product, i.e. $\langle \cdot, \cdot \rangle_{\mathcal{H}} = \langle \cdot, \cdot \rangle_{\mathcal{L}}$ on points in $\mathcal{L}$. Given r a rank of factorization, find $2r$ nonnegative functions $\{A_k\}_{k=1}^r$, with $A_k \in \tilde{\mathcal{A}}\ \forall k$, and $\{X_k\}_{k=1}^r$, with $X_k \in \tilde{\mathcal{X}}\ \forall k$, solving:

$$\underset{A_k \in \tilde{\mathcal{A}}, X_k \in \tilde{\mathcal{X}}\ \forall k}{\operatorname{argmin}} \quad \|Y - \sum_{k=1}^{r} A_k \otimes X_k\|_{\mathcal{H}}^2. \tag{4.18}$$

Considering Y_1 the projection of Y on $\mathcal{L}$, we can rewrite problem (4.18) as

$$\underset{A_k \in \tilde{\mathcal{A}}, X_k \in \tilde{\mathcal{X}}\ \forall k}{\operatorname{argmin}} \sum_{k_1,k_2=1}^{r} \langle A_{k_1}, A_{k_2} \rangle_{\mathcal{A}} \langle X_{k_1}, X_{k_2} \rangle_{\mathcal{X}} - 2 \sum_{k=1}^{r} \langle f_k, A_k \rangle_{\mathcal{A}} \quad \text{with } f_k \in \mathcal{A};\ f_k(a) = \langle Y_1(a,:), X_k \rangle_{\mathcal{X}}\ \forall a \in \mathbb{A} \tag{4.19}$$

or the equivalent

$$\underset{A_k \in \tilde{\mathcal{A}}, X_k \in \tilde{\mathcal{X}}\ \forall k}{\operatorname{argmin}} \sum_{k_1,k_2=1}^{r} \langle A_{k_1}, A_{k_2} \rangle_{\mathcal{A}} \langle X_{k_1}, X_{k_2} \rangle_{\mathcal{X}} - 2 \sum_{k=1}^{r} \langle g_k, X_k \rangle_{\mathcal{X}} \quad \text{with } g_k \in \mathcal{X};\ g_k(x) = \langle Y_1(:,x), A_k \rangle_{\mathcal{A}}\ \forall x \in \mathbb{X}. \tag{4.20}$$

4.2.2 Properties of the cost function

Let us analyze the differentiability of the cost function in Definition 4.4, when $\mathcal{A}$ and $\mathcal{X}$ are RKHS. For this purpose, we first recall the definition of the Fréchet differentiability that can be used for RKHS.

Definition 4.5. Fréchet differentiability Let $\mathbb{D}$ and $\mathbb{G}$ be normed vector spaces and f be a function on an open subset $\mathbb{U}$ of $\mathbb{D}$; $f : \mathbb{U} \subset \mathbb{D} \mapsto \mathbb{G}$. Function f is Fréchet differentiable at $u \in \mathbb{U}$ if there exists a bounded linear operator $L : \mathbb{D} \mapsto \mathbb{G}$ such that

$$\lim_{\|h\|_{\mathbb{D}} \to 0} \frac{\|f(u+h) - f(u) - L(h)\|_{\mathbb{G}}}{\|h\|_{\mathbb{D}}} = 0.$$

By the Riesz representation theorem, when $\mathbb{D}$ is a RKHS, the operator L can be defined as $L(h) = \langle D_f, h \rangle_{\mathbb{D}}$, with $D_f \in \mathbb{D}$. If L exists, it is unique.

Note that for Euclidean spaces, D_f is the gradient of f. Therefore, in the rest of this paper, we will refer to D_f as the gradient. This gradient is a linear operator, in the sense that $D_{af+bg} = aD_f + bD_g$.

Lemma 4.3. *Given a RKHS $\mathbb{D}$, and a fixed element $Y \in \mathbb{D}$. Let $f_1 : \mathbb{D} \mapsto \mathbb{R}$ with $f_1(A) = \langle Y, A\rangle_{\mathbb{D}}$, $f_2 : \mathbb{D} \mapsto \mathbb{R}$ with $f_2(A) = \langle A, Y\rangle_{\mathbb{D}}$, and $f_3 : \mathbb{D} \mapsto \mathbb{R}$ with $f_3(A) = \langle A, A\rangle_{\mathbb{D}}$, we have $D_{f_1} = D_{f_2} = Y$ and $D_{f_3} = 2A$.*

Proof. These gradients can be found using definition 4.5. For example, $G = D_{\langle Y,A\rangle_{\mathbb{D}}}$ is such that

$$\lim_{\|h\|_{\mathbb{D}}\to 0} \frac{|\langle Y, A+h\rangle_{\mathbb{D}} - \langle Y, A\rangle_{\mathbb{D}} - \langle G, h\rangle_{\mathbb{D}}|}{\|h\|_{\mathbb{D}}} = \lim_{\|h\|_{\mathbb{D}}\to 0} \frac{|\langle Y - G, h\rangle_{\mathbb{D}}|}{\|h\|_{\mathbb{D}}} = 0$$

which is satisfied when $G = Y$. □

Proposition 4.4. *The problem defined in Definition 4.4 is differentiable (in Fréchet sense) in each A_i, X_i $\forall i$, $A = \{A_i\}_{i=1}^r$, and $X = \{X_i\}_{i=1}^r$. Let f_V be the cost $\|Y - \sum_{k=1}^r A_k \otimes X_k\|^2$ when all variables are considered as fixed except V. We have*

$$D_{f_{A_i}} = \sum_{k=1}^{r} 2A_k\langle X_k, X_i\rangle_{\mathcal{X}} - 2f_i \qquad D_{f_{X_i}} = \sum_{k=1}^{r} 2X_k\langle A_k, A_i\rangle_{\mathcal{A}} - 2g_i$$

$$D_{f_A} = \Big\{ \sum_{k=1}^{r} 2A_k\langle X_k, X_i\rangle_{\mathcal{X}} - 2f_i \Big\}_{i=1}^{r} \qquad D_{f_X} = \Big\{ \sum_{k=1}^{r} 2X_k\langle A_k, A_i\rangle_{\mathcal{A}} - 2g_i \Big\}_{i=1}^{r}.$$

Proof. Using problem (4.20), we have

$$f_{X_i} = \sum_{k_1,k_2\neq i} \langle A_{k_1}, A_{k_2}\rangle_{\mathcal{A}}\langle X_{k_1}, X_{k_2}\rangle_{\mathcal{X}} - 2\sum_{k\neq i}\Big(\langle g_k, X_k\rangle_{\mathcal{X}} - \langle A_k, A_i\rangle_{\mathcal{A}}\langle X_k, X_i\rangle_{\mathcal{X}}\Big) + \langle A_i, A_i\rangle_{\mathcal{A}}\langle X_i, X_i\rangle_{\mathcal{X}} - 2\langle g_i, X_i\rangle_{\mathcal{X}}$$

Then, using the linearity of the Fréchet differentiation and Lemma 4.3 we obtain:

$$D_{f_{X_i}} = \sum_{k=1}^{r} 2\langle A_k, A_i\rangle_{\mathcal{A}} X_k - 2g_i$$

As all the partial derivatives $\{D_{f_{X_i}}\}_{i=1}^r$ exist and are continuous (because

g_i is independent from X_i), we have

$$D_{f_X} = \Big\{ \sum_{k=1}^{r} 2\langle A_k, A_i \rangle_{\mathcal{A}} X_k - 2g_i \Big\}_{i=1}^{r}.$$

The gradients on A_i and A can be obtained using a similar reasoning, using this time problem (4.19). □

Now we give a condition for which input function Y belongs to space $\mathcal{L} = \mathcal{A} \otimes \mathcal{X}$.

Proposition 4.5. *Let $\mathcal{A}$ and $\mathcal{X}$ be RKHS. Function $Y : \mathbb{A} \times \mathbb{X} \mapsto \mathbb{R}$ belongs to space $\mathcal{L} = \mathcal{A} \otimes \mathcal{X}$ when $\mathcal{A}$ or $\mathcal{X}$ is finite dimensional and $Y(a,:) \in \mathcal{X}\ \forall a \in \mathbb{A}$, $Y(:,x) \in \mathcal{A}\ \forall x \in \mathbb{X}$.*

Proof. Suppose $\mathcal{X}$ is finite dimensional. It is therefore possible to find an orthonormal basis of $\mathcal{X}$, $\{F_x^i\}_{i=1}^n$, with n finite. Let $\{k_x\}$ be the set of reproducing kernels of $\mathcal{X}$, and $f^i : \mathbb{A} \to \mathbb{R}$ defined by $f^i(a) = \langle Y(a,:), F_x^i \rangle_{\mathcal{X}}$. As $\mathcal{X}$ has dimension n, they are at most n linearly independent elements in $\{k_x\}$, and the linear span of functions k_x is a finite-dimensional subspace of the normed space $\mathcal{X}$, and is therefore a complete space. However, it is known that the linear span of functions k_x is dense in $\mathcal{X}$, which means that the closure of this span is equal to $\mathcal{X}$.

As the linear span of functions k_x is complete, it contains its closure, and therefore it is equal to $\mathcal{X}$. Consequently, every $X \in \mathcal{X}$ can be described as a finite linear combination of reproducing kernels, which leads to

$$f^i(a) = \langle Y(a,:), \sum_{j=1}^{n} \alpha_j k_{x_j} \rangle_{\mathcal{X}} = \sum_{j=1}^{n} \alpha_j \langle Y(a,:), k_{x_j} \rangle_{\mathcal{X}} = \sum_{j=1}^{n} \alpha_j Y(a, x_j)$$

$$\Rightarrow f^i = \sum_{j=1}^{n} \alpha_j Y(:, x_j) \in \mathcal{A}\ \forall i \qquad \text{as } \mathcal{A} \text{ is a vector space, and } Y(:,x_j) \in \mathcal{A}.$$

As $Y(a,:) \in \mathcal{X}\ \forall a \in \mathbb{A}$, we also have

$$Y(a,:) = \sum_{i=1}^{n} \langle Y(a,:), F_x^i \rangle_{\mathcal{X}} F_x^i \Rightarrow Y(a,x) = \sum_{i=1}^{n} \langle Y(a,:), F_x^i \rangle_{\mathcal{X}} F_x^i(x)$$

$$\Rightarrow Y = \sum_{i=1}^{n} f^i \otimes F_x^i \in \mathcal{L} \qquad \text{as } f^i \in \mathcal{A}.$$

We can use the same reasoning when $\mathcal{A}$ is finite dimensional. □

We now give some examples of problems studied in the literature, and present them in the H-NMF framework (Definition 4.4).

Proposition 4.6. *Some problems in the H-NMF framework:*

1. *Standard NMF:* $\mathbb{A} = \{1, \cdots, m\}, \mathcal{A} = \{A : \mathbb{A} \mapsto \mathbb{R}\} = \mathbb{R}^m, \tilde{\mathcal{A}} = \mathbb{R}^m_+$.
2. *NMF in transformed domain:* $\mathbb{A} = \{1, \cdots, m\}$, $\mathcal{A} = \{A \in \mathbb{R}^m \mid A = \boldsymbol{W}\boldsymbol{\alpha}$, *with* $\boldsymbol{W} \in \mathbb{R}^{m \times d}$ *fixed,* $\boldsymbol{\alpha} \in \mathbb{R}^{d \times 1}\}$ *where* $\boldsymbol{W}$ *is a transform (such as the wavelet transform), and* $\tilde{\mathcal{A}} = \{A = \boldsymbol{W}\boldsymbol{\alpha}$, *with* $A \geq 0$ *and* $\boldsymbol{\alpha}$ *a sparse vector (with few nonzero elements)* $\}$. *[102]*
3. *Discretization of functions:* $\mathbb{A} = \{a_i\}_{i=1}^m$, $\mathcal{A} = \{A : \mathbb{A} \mapsto \mathbb{R}\} = \mathbb{R}^m$ *and* $\tilde{\mathcal{A}} = \{$*nonnegative functions discretized on* $\mathbb{A}$ *like polynomials [33, 56], splines [9, 56], rational functions [59], or GRBF [139]*$\}$
4. *Polynomial NMF:* $\mathbb{A} = [-1, 1]$, $\mathcal{A} = \{$*polynomials of degree* d *with domain* $\mathbb{A}\}$, *with inner product* $\langle A_1, A_2 \rangle_{\mathcal{A}} = \int_{-1}^{1} A_1(a) A_2(a)\, \mathrm{d}a$ *and* $\tilde{\mathcal{A}} = \{$*polynomials in* $\mathcal{A}$ *nonnegative on* $\mathbb{A}\}$ *(Chapter 3).*
5. *Spline NMF:* $\mathbb{A}=[-1, 1]$, $\mathcal{A}=\{$*splines of degree 3 with fixed interior knots* $\{t_i\}_{i=1}^k$, *with domain* $\mathbb{A}\}$, *with same inner product as above and* $\tilde{\mathcal{A}} = \{$*splines in* $\mathcal{A}$ *nonnegative on* $\mathbb{A}\}$ *(Chapter 3).*
6. *Nonnegative functions: The work of Marteau-Ferey & al.[91] suggests that it is possible to model non-negative functions in a non-parametric way. If set* $\mathbb{A}$ *is finite, of size* m, *and* ϕ *is an universal feature map, we can consider the RKHS* $\mathcal{A} = \{A : \mathbb{A} \mapsto \mathbb{R}\}$ *with inner product* $\langle A_1, A_2 \rangle_{\mathcal{A}} = \sum_{a \in \mathbb{A}} A_1(a) A_2(a)$, $\mathcal{A}$ *is therefore equivalent to* $\mathbb{R}^m$. *The set of nonnegative functions can then be expressed as* $\tilde{\mathcal{A}} = \{A \in \mathcal{A} \mid A(a) = \phi(a)^\top \boldsymbol{M} \phi(a)$ $\forall a \in \mathbb{A}$, *where* $\boldsymbol{M} \succeq 0\}$. *Details for solving problems in such sets can be found in [91].*

The definitions for $\mathcal{X}$ *can be any of these six statements. In case 1, 2, 3 and 6,* $\mathcal{H} = \mathbb{R}^{m \times n}$, *while in cases 4 and 5,* $\mathcal{H} \subset L_2$, *the space of square integrable functions. Note that in the fourth and the fifth case,* $\mathcal{A}$ *could be defined differently, as long as* $\tilde{\mathcal{A}} \subseteq \mathcal{A}$, *and* $\mathcal{A}$ *is a RKHS. We can for example consider higher degree polynomials in* $\mathcal{A}$ *than in* $\tilde{\mathcal{A}}$. *This could make* Y *belong to space* $\mathcal{L} = \mathcal{A} \otimes \mathcal{X}$, *by Proposition 4.5.2, and therefore avoid to project* Y *on* $\mathcal{L}$. *Comments on the choice of spaces* $\mathcal{A}$, $\mathcal{X}$, $\tilde{\mathcal{A}}$ *and* $\tilde{\mathcal{X}}$ *are made in more details in next section.*

4.2.3 Comments on the choice of spaces $\mathcal{A}$, $\mathcal{X}$, $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$

In this section, we analyze the H-NMF problem (Definition 4.4) when spaces $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ are linearly parametrizable, and we comment the choice of RKHS $\mathcal{A}$ and $\mathcal{X}$.

Linearly parametrizable sets of functions

If every element in $\tilde{\mathcal{A}}$ can be described as a linear combination of a finite basis $\Pi^A = \{\Pi_l^A\}_{l=1}^{d_A}$, we say that $\tilde{\mathcal{A}}$ is linearly parametrizable. In this case, every element in $\tilde{\mathcal{A}}$ can be uniquely described using a coefficient matrix $\boldsymbol{C}^A \in \mathbb{R}^{d_A}$: $A = \sum_{l=1}^{d_A} \Pi_l^A \boldsymbol{C}_l^A$. An example of a linearly parametrizable set of functions is the set of polynomials of fixed degree. Note that it is not necessary that all the elements spanned by Π^A belong to $\tilde{\mathcal{A}}$, therefore the set of nonnegative polynomials of fixed degree is also linearly parametrizable.

Let us analyze the expression of the inner products in the cost function of H-NMF when $\tilde{\mathcal{A}}$ and/or $\tilde{\mathcal{X}}$ are linearly parametrizables:

1. If $\tilde{\mathcal{A}}$ is linearly parametrizable, we can precompute $\boldsymbol{M}^A \in \mathbb{R}^{d_A \times d_A}$ with $\boldsymbol{M}_{l,j}^A = \langle \Pi_l^A, \Pi_j^A \rangle_{\mathcal{A}}$ and $\{Z_l^A\}_{l=1}^{d_A}$, with $Z_l^A(x) = \langle Y_1(:,x), \Pi_l^A \rangle_{\mathcal{A}} \in \mathcal{X}$. We have then

   (a) $\langle A_l, A_j \rangle_{\mathcal{A}} = \boldsymbol{C}^{A_l\top} \boldsymbol{M}^A \boldsymbol{C}^{A_j}$,

   (b) $g_k(x) = \langle Y_1(:,x), A_k \rangle_{\mathcal{A}} = \sum_{l=1}^{d_A} \boldsymbol{C}_l^{A_k} Z_l^A(x)$.

   Similarly, if $\tilde{\mathcal{X}}$ is linearly parametrizable, $\langle X_l, X_j \rangle_{\mathcal{X}} = \boldsymbol{C}^{X_l\top} \boldsymbol{M}^X \boldsymbol{C}^{X_j}$, and $f_k(a) = \sum_{l=1}^{d_X} \boldsymbol{C}_l^{X_k} Z_l^X(a)$.

2. If both sets $\tilde{\mathcal{A}}$, $\tilde{\mathcal{X}}$ are linearly parametrizable, we can also precompute matrix $\boldsymbol{Z}^{AX} \in \mathbb{R}^{d_A \times d_X}$, with $\boldsymbol{Z}_{lq}^{AX} = \langle Z_l^A, \Pi_q^X \rangle_{\mathcal{X}} = \langle Y, \Pi_l^A \otimes \Pi_q^X \rangle_{\mathcal{H}}$ (see Lemma 4.2) and have:

   (a) $\langle g_k, X_j \rangle_{\mathcal{X}} = \boldsymbol{C}^{A_k\top} \boldsymbol{Z}^{AX} \boldsymbol{C}^{X_j}$.

   Moreover, as elements in Π^A are linearly independent, $\boldsymbol{M}^A$ is invertible. This matrix is also positive semi-definite, as it is a Gram matrix, and we can therefore find its Cholesky decomposition $\boldsymbol{L}^A$, such that $\boldsymbol{L}^A \boldsymbol{L}^{A\top} = \boldsymbol{M}^A$, and similarly we can find $\boldsymbol{L}^X$. Let $\boldsymbol{C}^A \in \mathbb{R}^{d_A \times r}$ contain in each of its columns $\boldsymbol{C}_{:k}^A$, the coefficients of A_k, and similarly for $\boldsymbol{C}^X \in \mathbb{R}^{d_X \times r}$. Neglecting the constraint, the cost function (4.20) is

equivalent to

$$\sum_{k_1,k_2=1}^{r} {\boldsymbol{C}^A_{:k_1}}^\top \boldsymbol{M}^A \boldsymbol{C}^A_{:k_2} {\boldsymbol{C}^X_{:k_1}}^\top \boldsymbol{M}^X \boldsymbol{C}^X_{:k_2} - 2\sum_{k=1}^{r} {\boldsymbol{C}^A_{:k}}^\top \boldsymbol{Z}^{AX} \boldsymbol{C}^X_{:k}$$

which is equivalent up to a constant to

$$\left\| {\boldsymbol{L}^A}^\top \boldsymbol{C}^A {\boldsymbol{C}^X}^\top \boldsymbol{L}^X - {\boldsymbol{L}^A}^\top {\boldsymbol{M}^A}^{-1} \boldsymbol{Z}^{AX} {\boldsymbol{M}^X}^{-1} \boldsymbol{L}^X \right\|_F^2. \tag{4.21}$$

This means that when both A and X are linearly parametrizable, the problem becomes the minimization of a Frobenius norm on coefficient matrices, after some preprocessing to compute matrices $\boldsymbol{Z}^{AX}$, $\boldsymbol{M}^A$ and $\boldsymbol{M}^X$.

Using linearly parametrizable functions also allows us to compute more efficiently projections on $\tilde{\mathcal{A}}$ or $\tilde{\mathcal{X}}$. Indeed we have:

$$\operatorname*{argmin}_{A^* \in \tilde{\mathcal{A}}} \|A^* - A\|_{\mathcal{A}}^2 = \operatorname*{argmin}_{\sum_{i=1}^{d_A} \boldsymbol{C}_i^* \Pi_i^A \in \tilde{\mathcal{A}}} \left\| {\boldsymbol{L}^A}^\top \left(\boldsymbol{C}^* - \boldsymbol{C}^A\right) \right\|_2^2. \tag{4.22}$$

Therefore we can optimize on the coefficients $\boldsymbol{C}^*$ using a Euclidean norm instead of the function A^* with norm based on the inner product $\langle \cdot, \cdot \rangle_{\mathcal{A}}$. The LP-NMF problem presented in Chapter 3 is thus a special case of the H-NMF problem.

To summarize, some inner products of equations from Theorem 4.3 can be replaced by matrix products using a precomputed Gram matrix containing the inner products of the basis elements Π^A and/or Π^X. Moreover, we can optimize directly on the coefficients instead of the whole function which often decreases the complexity of the problem. This can significantly improve the time performances of the algorithms solving H-NMF, especially if the computation of the inner products is time-consuming.

Choosing appropriate spaces $\mathcal{A}$ and $\mathcal{X}$

As stated in Proposition 4.6, choosing large spaces $\mathcal{A}$ and $\mathcal{X}$ may avoid projecting the input function Y on set $\mathcal{L} = \mathcal{A} \otimes \mathcal{X}$, but the inner product may also become harder to compute. Moreover, this is not always possible as Y is not imposed to belong to a RKHS. On the contrary, having smaller sets $\mathcal{A}$ and $\mathcal{X}$ may simplify the computation of the inner product, but also complicate the computation of the projection of Y. For example, choosing

$\mathcal{A} = \tilde{\mathcal{A}}$ and $\mathcal{X} = \tilde{\mathcal{X}}$, would make the optimization problem trivial, but the projection of Y on $\mathcal{L}$ becomes very difficult, and this choice does not make sense. Moreover, it is not always possible to make this choice, since non-negativity constraints prevent $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ from being RKHS. It is therefore important to be careful when describing a H-NMF problem.

The examples presented in this section are intended to show that it is often possible to use different sets $\mathcal{A}$,$\mathcal{X}$ for the same problem. The choice of these sets should not be neglected as it influences the solution of the problem, as well as its resolution.

Suppose that Y is a continuous function of two variables. In this case, $\mathcal{A}$ and $\mathcal{X}$ can be spaces of continuous functions of one variable. But this can be difficult to handle if function Y is not known exactly. A possibility is to perform a preprocessing step to approximate the input data, using for example Chebfun developed by Battles and Trefethen [11] that uses polynomials interpolant to model a function. The work of Trefethen and its team actually has several similarities with ours. Indeed, in [119] and [118], they develop several approaches to factor matrices via as LU, QR or SVD factorization, in the case where the matrices are what they call "quasimatrices", i.e. matrices where one of the dimensions is infinite, or even "cmatrices", matrices where both dimensions are infinite, and which can therefore be seen as two-dimensional functions. This is done by using linear combinations or inner products of Chebfun approximations, which is close to what we do. However, the use of Chebfun assumes that the input function is known, or at least that it can be calculated at any given point, which is not always the case. Using Chebfun in the context of H-NMF is a very interesting idea for future work, but we do not focus on that here. Instead, we use the input information as provided.

For example, if the input function Y is known via $m \times n$ discretization points, $\{(\boldsymbol{\tau}_i^A, \boldsymbol{\tau}_j^X)\}_{i=1,j=1}^{m,n}$, we can choose to work on $\mathcal{A} = \mathbb{R}^m$ and $\mathcal{X} = \mathbb{R}^n$. If we want to approximate Y using sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$, we can now consider $\tilde{\mathcal{A}}'$ and $\tilde{\mathcal{X}}'$ with $\tilde{\mathcal{A}}' = \{A(\boldsymbol{\tau}^A) \mid A \in \tilde{\mathcal{A}}\}$, where $A(\boldsymbol{\tau}^A) \in \mathbb{R}^m$ is the evaluation of function A at points $\{\tau_i^A\}_{i=1}^m$, and similarly $\tilde{\mathcal{X}}' = \{X(\boldsymbol{\tau}^X) \mid X \in \tilde{\mathcal{X}}\}$.

As another example, consider that $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$ is a matrix containing the information about the mean value over known intervals of functions:

$$\boldsymbol{Y}_{i,j} = \frac{\int_{a_i}^{a_{i+1}} \int_{x_j}^{x_{j+1}} Y(a,x)\, \mathrm{d}a\mathrm{d}x}{(x_{j+1} - x_j)(a_{i+1} - a_i)}.$$

Then we can again work in $\mathcal{A}' = \mathbb{R}^m$ and $\mathcal{X}' = \mathbb{R}^n$ with $\tilde{\mathcal{A}}' = \left\{ \boldsymbol{\alpha} \mid \boldsymbol{\alpha}_i = \frac{\int_{a_i}^{a_{i+1}} A(a)\,\mathrm{d}a}{a_{i+1}-a_i}; A \in \tilde{\mathcal{A}} \right\}$ and $\tilde{\mathcal{X}}' = \left\{ \beta \mid \beta_j = \frac{\int_{x_j}^{x_{j+1}} X(x)\,\mathrm{d}x}{x_{j+1}-x_j};\ X \in \tilde{\mathcal{X}} \right\}$. If $\tilde{\mathcal{A}}$ is linearly parametrizable, we have

$$\frac{\int_{a_i}^{a_{i+1}} A(a)\,\mathrm{d}a}{a_{i+1}-a_i} = \sum_{j=1}^{d_A} \boldsymbol{C}_j^A \frac{\int_{a_i}^{a_{i+1}} \Pi_j^A(a)\,\mathrm{d}a}{a_{i+1}-a_i}.$$

Therefore, in this case, we can use results from Section 4.2.3 with basis $\tilde{\boldsymbol{\Pi}}^A \in \mathbb{R}^{m \times d_A}$, $\tilde{\boldsymbol{\Pi}}^A_{i,j} = \frac{\int_{a_i}^{a_{i+1}} \Pi_j^A(a)\,\mathrm{d}a}{a_{i+1}-a_i}$.

4.3 Solving the H-NMF problem

Many algorithms have been proposed to solve the NMF problem and many of them rely on a block-decomposition of the problem, and solve the problem by alternatively optimizing on each block, considering the other blocks as fixed. Three kinds of decomposition can be considered: either the problem is seen as a unique block, or it is divided in two blocks, the factors $\boldsymbol{A}$ and $\boldsymbol{X}$, or it is divided in $2r$ blocks, the columns of $\boldsymbol{A}$ and $\boldsymbol{X}$ (in H-NMF, the functions A_k and X_k). Below we present those three decompositions, the algorithms used to solve them, and the conditions needed to be able to extend those algorithms to H-NMF.

We consider the original problem (4.1) with Frobenius norm. Extensions of this problem (with regularization, additional constraints, etc.) can be adapted in similar ways.

4.3.1 Considering one block

Consider a vector $\boldsymbol{a} \in \mathbb{R}^m$. The nonnegativity constraint on this vector can easily be expressed as $\boldsymbol{a} = \boldsymbol{h}^{\cdot 2}$ where $\boldsymbol{h} \in \mathbb{R}^m$ and $^{\cdot 2}$ is the square performed element-wise. The NMF problem becomes then $\min_{\boldsymbol{E},\boldsymbol{F}} \|\boldsymbol{Y} - \boldsymbol{E}^{\cdot 2}(\boldsymbol{F}^{\cdot 2})^\top\|^2$ with the factors $\boldsymbol{A} = \boldsymbol{E}^{\cdot 2}$ and $\boldsymbol{X} = \boldsymbol{F}^{\cdot 2}$. This problem is unconstrained and can therefore be solved using any nonlinear least squares (nls) solver. [25]

Extending this method to H-NMF is possible when the cost function can be expressed as a Frobenius norm and the set of constraints $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ can be described as $\tilde{\mathcal{A}} = \{\Phi(\boldsymbol{p});\ \boldsymbol{p} \in \mathbb{R}^{d_A}\}$ and $\tilde{\mathcal{X}} = \{\Psi(\boldsymbol{q});\ \boldsymbol{q} \in \mathbb{R}^{d_X}\}$. The cost function can be expressed as a Frobenius norm when Y is a matrix, or both sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ are linearly parametrizable (see Equation (4.21)), or $\tilde{\mathcal{A}}$ is linearly parametrizable and $\mathcal{X} = \mathbb{R}$, or reversely. It is important that $\Phi(\boldsymbol{p}) \in \tilde{\mathcal{A}}\ \forall \boldsymbol{p} \in \mathbb{R}^{d_A}$, to ensure that the constraints are satisfied, however, it is not necessary that a function is described by a unique vector of coefficient $\boldsymbol{p}$. This is the case for polynomials nonnegative on an interval, for example [101], and this method is used for H-NMF on discrete polynomials in [33]. The algorithm is sketched on Algorithm 4.1.

Algorithm 4.1 H-NMF with one block

Input: $Y : \mathbb{A} \times \mathbb{X} \mapsto \mathbb{R}$, $\Phi : \mathbb{R}^{d_A} \mapsto \tilde{\mathcal{A}}$, $\Psi : \mathbb{R}^{d_X} \mapsto \tilde{\mathcal{X}}$
function 1-BLOCK_H-NMF(Y)
$\underset{\boldsymbol{p}_1,\dots,\boldsymbol{p}_r,\boldsymbol{q}_1,\dots,\boldsymbol{q}_r}{\texttt{nls\_solver}}\left(Y - \sum_{k=1}^{r} \Phi(\boldsymbol{p}_k) \otimes \Psi(\boldsymbol{q}_k)\right)$

4.3.2 Considering two blocks A and X

The idea in this scheme is to alternatively solve the NMF problem on $\boldsymbol{A}$ considering $\boldsymbol{X}$ as fixed, and on $\boldsymbol{X}$ considering $\boldsymbol{A}$ as fixed. Several options exist, we list below the most common ones:

- Multiplicative Updates (MU): this very common approach was proposed in early works on NMF [80]. The idea is to update each element of the factor matrix by multiplying it by a nonnegative number. This ensures the resulting matrix to stay nonnegative. The update is chosen to not increase the cost function. A common way to choose the update, is to perform a gradient descent with appropriate stepsize. To update $\boldsymbol{A}$, for example, the gradient is $-(\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}^\top)\boldsymbol{X}$. Choosing stepsize $\eta_{i,j} = \frac{\boldsymbol{A}_{i,j}}{[\boldsymbol{A}\boldsymbol{X}^\top\boldsymbol{X}]_{i,j}}$, different for each element, we have the following update, with $\odot$ the element-wise multiplication and $\oslash$ the element-wise division:

$$\boldsymbol{A} \leftarrow \boldsymbol{A} + \eta \odot (\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}^\top)\boldsymbol{X} = \boldsymbol{A} \odot \boldsymbol{Y}\boldsymbol{X} \oslash [\boldsymbol{A}\boldsymbol{X}^\top\boldsymbol{X}]$$

  Therefore, the multiplicative updates of the standard NMF problem

are

$$\boldsymbol{A} \leftarrow \boldsymbol{A} \odot \boldsymbol{Y}\boldsymbol{X} \oslash \boldsymbol{A}\boldsymbol{X}^\top\boldsymbol{X} \qquad \boldsymbol{X} \leftarrow \boldsymbol{X} \odot \boldsymbol{Y}^\top\boldsymbol{A} \oslash \boldsymbol{X}\boldsymbol{A}^\top\boldsymbol{A} \quad (4.23)$$

Those updates have the advantage to maintain the nonnegativity of $\boldsymbol{A}$ and $\boldsymbol{X}$ throughout iterations, without increasing the cost function. They also have low complexity, but MU can require many iterations, and is not ensured to converge to a stationary point.

To extend MU to H-NMF, the updates should keep the constraint satisfied, which is not trivial. For example, consider the set of nonnegative polynomials of degree d. This set is not closed under multiplication, and an update similar to (4.23) would not keep the constraint satisfied.

This approach is used in [141] on discrete splines with nonnegative coefficients. This is possible because the constraint is a "simple" nonnegative constraint as we work on splines with nonnegative coefficients instead of nonnegative splines.

- (Other) Gradient-based methods: when using other stepsizes, we can satisfy the constraint by projecting the found update on the feasible set [84]. Consider $[x]_{\mathcal{P}}$ the projection of x on set $\mathcal{P}$ and gradients from Proposition 4.4, the updates for H-NMF would be:

$$A_s \leftarrow \left[A_s + \eta_A\big(-\sum_k A_k\langle X_k, X_s\rangle_{\mathcal{X}} + 2f_s\big)\right]_{\tilde{\mathcal{A}}}, \quad f_s(a) = \langle Y_1(a,:), X_s\rangle_{\mathcal{X}}$$

$$X_s \leftarrow \left[X_s + \eta_X\big(-\sum_k X_k\langle A_k, A_s\rangle_{\mathcal{A}} + 2g_s\big)\right]_{\tilde{\mathcal{X}}}, \quad g_s(x) = \langle Y_1(:,x), A_s\rangle_{\mathcal{A}}$$

Note that all elements A_s are updated at the same time, and the same holds for all X_s. **To be able to perform such an update, linear combinations of A and X must be computable, as well as functions f_s and g_s. Moreover, it must be possible to project onto sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$.**

- As for one-block optimization, we can use functions Φ and Ψ and alternatively optimize on one factor with a non-linear least squares solver while considering the other factor as fixed. The necessary conditions to use this methods are the same as in the one-block case.

- Active set method: this method has been developed for constraints of

the form $\{g_i(\boldsymbol{x}) \geq 0\}_{i=1}^{n}$, with $\boldsymbol{x}$ the variables of the problem. **To be able to use the active set method for H-NMF, the constrained sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ must be describable using a finite number of nonnegativity constraints.** This approach has been used in [139] on discrete GRBF, imposed to be nonnegative on discretization points only.

- Alternating Direction Method of Multipliers (ADMM): this method is used to solve separable cost functions with linear constraints [44]. Using $\Phi_{\tilde{\mathcal{A}}}(\cdot)$ the indicator function of $\tilde{\mathcal{A}}$, the H-NMF problem with X fixed can be written as:

$$\begin{aligned} \underset{A,B}{\operatorname{argmin}} \sum_{k_1,k_2=1}^{r} \langle X_{k_1}, X_{k_2}\rangle_{\mathcal{X}} \langle B_{k_1}, B_{k_2}\rangle_{\mathcal{A}} - 2\sum_{k=1}^{r} \langle f_k, B_k\rangle_{\mathcal{A}} + \sum_{k=1}^{r} \Phi_{\tilde{\mathcal{A}}}(A_k) \\ \text{such that} \quad A_k = B_k \qquad \forall k = 1 \dots r. \\ A_k \in \tilde{\mathcal{A}}\,, \quad B_k \in \mathcal{A} \quad \forall k = 1 \dots r. \end{aligned}$$

To use ADMM for H-NMF, the problem should be easy to solve when A or X is fixed and the constraints $A_k \in \tilde{\mathcal{A}}$ and $X_k \in \tilde{\mathcal{X}}$ are neglected. It should also be easy to project onto sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$. This approach is used in [140] on nonnegative discrete splines.

The 2-block optimization of NMF problem is sketched on Algorithm 4.2. Note that A and X do not need to be updated with the same method.

Algorithm 4.2 H-NMF with two blocks

Input: $\mathcal{A}$ and $\mathcal{X}$ RKHS, $\tilde{\mathcal{A}} \subseteq \mathcal{A}$, $\tilde{\mathcal{X}} \subseteq \mathcal{X}$,
input function Y, initial $X = \{X_k\}_{k=1}^{r}$ with $X_k \in \tilde{\mathcal{X}}\ \forall k$

function 2-BLOCK_H-NMF(Y,X)

Compute Y_1, the projection of Y on $\mathcal{A} \otimes \mathcal{X}$.

repeat

$f_k : \mathbb{A} \mapsto \mathbb{R}, f_k(a) = \langle Y_1(a,:), X_k\rangle_{\mathcal{X}} \quad \forall k \in [1,..,r]$
$A \leftarrow \underset{A_k \in \tilde{\mathcal{A}}\ \forall k}{\operatorname{argmin}} \sum_{k_1,k_2} \langle X_{k_1}, X_{k_2}\rangle_{\mathcal{X}} \langle A_{k_1}, A_{k_2}\rangle_{\mathcal{A}} - 2\sum_k \langle f_k, A_k\rangle_{\mathcal{A}}$

$g_k : \mathbb{X} \mapsto \mathbb{R}, g_k(x) = \langle Y_1(:,x), A_k\rangle_{\mathcal{A}} \quad \forall k \in [1,..,r]$
$X \leftarrow \underset{X_k \in \tilde{\mathcal{X}}\ \forall k}{\operatorname{argmin}} \sum_{k_1,k_2} \langle X_{k_1}, X_{k_2}\rangle_{\mathcal{X}} \langle A_{k_1}, A_{k_2}\rangle_{\mathcal{A}} - 2\sum_k \langle g_k, X_k\rangle_{\mathcal{X}}$

until convergence

4.3.3 Considering 2r blocks A_k, X_k $k = 1, \ldots, r$

For standard NMF, dividing the problem in $2r$ blocks, the columns of A and X, and optimizing alternatively on each of them, allows us to solve each subproblem analytically. Indeed, as the cost function is a squared norm, it is possible to show that the best update is the projection of the best unconstrained update, i.e. the update that cancels the gradient, onto the nonnegative set [75], using $[\xi]_+ = \max(\xi, 0)$:

$$\boldsymbol{A}_{:k} \leftarrow \left[\frac{\boldsymbol{Y}\boldsymbol{X}_{:k} - \sum_{s\neq k} \boldsymbol{A}_{:s}(\boldsymbol{X}_{:s})^\top \boldsymbol{X}_{:k}}{\|\boldsymbol{X}_{:k}\|^2}\right]_+ \forall k$$
$$\boldsymbol{X}_{:k} \leftarrow \left[\frac{\boldsymbol{Y}^\top\boldsymbol{A}_{:k} - \sum_{s\neq k} \boldsymbol{X}_{:s}(\boldsymbol{A}_{:s})^\top \boldsymbol{A}_{:k}}{\|\boldsymbol{A}_{:k}\|^2}\right]_+ \forall k$$

These updates are optimal and unique when $\|\boldsymbol{X}_{:k}\|^2$ and $\|\boldsymbol{A}_{:k}\|^2$ are nonzero. Actually, when $\|\boldsymbol{X}_{k:}\|^2$ and $\|\boldsymbol{A}_{:k}\|^2$ are nonzero throughout all iterations, it is possible to prove that the algorithm converges to a stationary point [75]. This algorithm is called the Hierarchical Alternating Least Squares (HALS) algorithm [27].

To extend this approach to H-NMF, it must be possible to compute the unconstrained update, which is the case if linear combinations of A and X are computable, as well as functions f_s and g_s. Moreover, one must be able to project onto sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$. This approach has been considered in [133] for discrete GRBF, but the projection was estimated using a multiplicative update of the columns of the factors $\boldsymbol{A}$ and $\boldsymbol{X}$. We considered this approach using exact projections in Chapter 3.

An interesting observation to make about this algorithm is that the projection tends to favor solutions at the boundary of the feasible set. In the case of vectors (standard NMF), this results in sparse vectors (since negative values are set to zero). In the case of functions, the result does not necessarily have many zeros. Take the case of polynomials for example: by nature these functions cannot be zero at many points (unless they are zero everywhere), so projection does not tend towards a sparse signal. In the case of splines it is possible to have sparse signals, but it is not guaranteed at all that the projection results in sparse signals, as it depends on the position of the interior knots and the negative/zero zones of the signal to be projected. Sparsity is therefore in general not promoted with H-HALS as opposed to conventional HALS. Nevertheless, it is worth considering whether spar-

sity makes sense when using functions. If it does, the choice of function set should be adapted (like splines), and other techniques should be used to encourage sparsity (e.g. by regularizing the cost function, see the end of Section 5.2 for another comment on sparsity).

In Section 5.2 of the next chapter, it is proved that the Hilbert HALS method (H-HALS), sketched on Algorithm 4.3, preserves the properties of standard NMF, i.e. projecting the unconstrained solution on the feasible set is optimal and, if $\tilde{\mathcal{A}}$, $\tilde{\mathcal{X}}$ are convex and $\|A_k\|^2$, $\|X_k\|^2$ are nonzero throughout all iterations, the algorithm converges to a stationary point. We observe that the LP-HALS algorithm (Algorithm 3.1) is a special case of the H-HALS algorithm for linearly parametrizable functions (see Section 4.2.3).

Note that it is also possible, although much less common, to define $r+1$ blocks, by dividing A in r blocks and keeping X as one block, or the other way around.

Algorithm 4.3 H-NMF with $2r$ blocks (H-HALS)

Input: $\mathcal{A}$ and $\mathcal{X}$ RKHS, $\tilde{\mathcal{A}} \subseteq \mathcal{A}$, $\tilde{\mathcal{X}} \subseteq \mathcal{X}$, input function Y, initials $X = \{X_k\}_{k=1}^r$, $A = \{A_k\}_{k=1}^r$ with $X_k \in \tilde{\mathcal{X}}$, $A_k \in \tilde{\mathcal{A}}\ \forall k$

function H-HALS(Y,A,X)

Compute Y_1, the projection of Y on $\mathcal{A} \otimes \mathcal{X}$.

repeat

for $k \in [1...r]$ **do**

$f_k : \mathbb{A} \mapsto \mathbb{R}, f_k(a) = \langle Y_1(a,:), X_k \rangle_{\mathcal{X}}$

$A_k \leftarrow \left[\frac{f_k - \sum_{s \neq k} A_s \langle X_s, X_k \rangle_{\mathcal{X}}}{\|X_k\|^2_{\mathcal{X}}} \right]_{\tilde{\mathcal{A}}}$

for $k \in [1...r]$ **do**

$g_k : \mathbb{X} \mapsto \mathbb{R}, g_k(x) = \langle Y_1(:,x), A_k \rangle_{\mathcal{A}}$

$X_k \leftarrow \left[\frac{g_k - \sum_{s \neq k} X_s \langle A_s, A_k \rangle_{\mathcal{A}}}{\|A_k\|^2_{\mathcal{A}}} \right]_{\tilde{\mathcal{X}}}$

until Convergence

5

Convergence of H-NMF

In the previous chapter, we have presented a general framework to perform NMF using functions, the H-NMF framework. We have also presented several ways to extend the algorithms of NMF to H-NMF. In particular, we have presented an extension of the HALS algorithm, the H-HALS algorithm, whose iterations are computable under mild conditions, and requires performing projections on the functions of interest. In this chapter, we first show that this algorithm can be described as a Block Coordinate Descent (BCD) method. We then analyze the convergence properties of this algorithm and compare them to the known convergence properties of standard HALS (Section 5.2). We observe that both algorithms have comparable convergence properties when the updates are performed exactly. Moreover, we analyze in Section 5.3 the convergence properties when the updates are not performed exactly. This is very informative as the projection step of the H-HALS algorithm can be difficult to perform, and therefore could be computed using heuristics instead, which results in an inexact update. The iterates are supposed to be solution of smaller scale minimization problems, and we show that under mild conditions H-HALS will converge to a stationary point or a nearly-stationary point when the iterates are ϵ-stationary, or solutions of a minimization problem with less restrictive constraints, or close to the optimal iterate.

5.1 Block Coordinate Descent (BCD) methods for H-NMF

To simplify the notations, we first rewrite the H-NMF problem. Suppose, we have n blocks $\{x_i\}_{i=1}^n$, where each block x_i belongs to a RKHS $\mathcal{F}_i$, and that the constraint on this block is defined by set $\mathcal{P}_i$, i.e. $x_i \in \mathcal{P}_i \subseteq \mathcal{F}_i \ \forall i$. Let $\mathcal{P} = \mathcal{P}_1 \times \cdots \times \mathcal{P}_n$, $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_n$, and $x = (x_1, \cdots, x_n)$. Let f be the cost function, and $f_i(\xi\,; x)$ be the cost function when all blocks are considered as fixed except block i:

$$f_i(\xi\,; x) = f(x_1, \ldots, x_{i-1}, \xi, x_{i+1}, \ldots, x_n) \qquad \xi \in \mathcal{F}_i,\ x \in \mathcal{P}. \tag{5.1}$$

The Block Coordinate Descent (BCD) algorithm, sketched on Algorithm 5.1, aims then to solve:

$$\min_x f(x_1, \cdots, x_n) \quad \text{such that } \ x = (x_1, \cdots, x_n) \in \mathcal{P}_1 \times \cdots \times \mathcal{P}_n = \mathcal{P}. \tag{5.2}$$

For H-NMF, f is one of the cost function from Definition (4.4), and

- when considering 2 blocks, $x_1 = A$, $x_2 = X$, $\mathcal{F}_1 = \mathcal{A}^r$, $\mathcal{P}_1 = \tilde{\mathcal{A}}^r$, $\mathcal{F}_2 = \mathcal{X}^r$, $\mathcal{P}_2 = \tilde{\mathcal{X}}^r$, with $\mathcal{A}^r = \mathcal{A} \times \cdots \times \mathcal{A}$ r times, and similarly for $\tilde{\mathcal{A}}^r$, $\mathcal{X}^r$, $\tilde{\mathcal{X}}^r$,
- when considering $2r$ blocks, $x_i = \begin{cases} A_i & \forall i = 1, \cdots, r \\ X_{i-r} & \forall i = r+1, \cdots, 2r' \end{cases}$
  $\mathcal{F}_i = \begin{cases} \mathcal{A} & \forall i = 1, \cdots, r \\ \mathcal{X} & \forall i = r+1, \cdots, 2r \end{cases}$ and $\mathcal{P}_i = \begin{cases} \tilde{\mathcal{A}} & \forall i = 1, \cdots, r \\ \tilde{\mathcal{X}} & \forall i = r+1, \cdots, 2r \end{cases}$.

Note that we consider in this section that every update of the BCD algorithm is the exact minimizer when all blocks except one are considered as fixed (Equation (5.3)). This approach is sometimes referred to as the alternating minimization method, to specify the exactness of the update. Nevertheless, as we will also consider inexact updates in the next section, we decided to keep the name BCD.

Moreover, for simplicity we consider only a cyclic update of the blocks, i.e. block $i+1$ is updated after block i, and once last block n has been updated, we start a new cycle at block 1.

Algorithm 5.1 Block Coordinate Descent Algorithm (BCD)

Input: $x^0 \in \mathcal{P}$
function BCD(x^0)
  $t = 0$
  Repeat:
    for $i = 1, 2, \ldots, n$ **do**

$$x_i^{t+1} = \underset{\xi}{\operatorname{argmin}} f_i(\xi\,; x^t) \text{ such that } \xi \in \mathcal{P}_i \tag{5.3}$$
$$x_j^{t+1} = x_j^t \ \forall j \neq i$$
$$t \quad = t+1$$

Optimality of updates presented in Algorithm 4.3 We now verify that the updates of Algorithm 4.3 are optimal, and therefore H-HALS is a BCD method. First, let us observe functions $f_i(\xi\,; x)$ when the problem is decomposed in $2r$ blocks.

Lemma 5.1. *Let $O_i = \|A_s\|_{\mathcal{A}}^2$ if block i is X_s and $O_i = \|X_s\|_{\mathcal{X}}^2$, if block i is A_s. For all $x \in \mathcal{P}$, the block functions $f_i(\xi\,; x)$ are quadratic, $2O_i$-strongly convex and*

$$f_i(\xi_1 + \xi_2\,; x) = f_i(\xi_1\,; x) + \langle D_{f_i(\xi\,; x)}(\xi_1), \xi_2 \rangle_{\mathcal{F}_i} + \|\xi_2\|_{\mathcal{F}_i}^2 O_i, \tag{5.4}$$

$$D_{f_i(\xi\,; x)}(\xi_1 + \xi_2) = D_{f_i(\xi\,; x)}(\xi_1) + 2O_i\xi_2 \tag{5.5}$$

Proof. The quadratic structure of the functions can be easily observed in Equations (4.15) and (4.16). The presented result is the Taylor expansion of a quadratic function, where the Hessian has been replaced by its exact value. It can be verified using Equations (4.15), (4.16) and Proposition 4.4. □

Lemma 5.2. *If throughout algorithm, we have $\|x_i\|_{\mathcal{F}_i}^2 \geq \delta > 0 \ \forall i = 1, \ldots, 2r$, then the updates defined in H-HALS Algorithm (4.3) are optimal, meaning that H-HALS is a BCD method corresponding thus to Algorithm 5.1.*

Proof. Updates of H-HALS Algorithm (4.3) can be described as the projection on set $\mathcal{P}_{i_t}$ of the best unconstrained update (that is the update that

cancels the gradient):

$$x_{i_t}^{t+1} = \left[\operatorname*{argmin}_{\xi} f_{i_t}(\xi\,; x^t)\right]_{\mathcal{P}_{i_t}}.$$

We aim to compute

$$x_{i_t}^{t+1} = \operatorname*{argmin}_{\xi} f_{i_t}(\xi\,; x^t) \qquad \text{such that } \xi \in \mathcal{P}_{i_t}.$$

Suppose that $\bar{x}$ is the unconstrained minimizer of $f_{i_t}(\xi\,; x^t)$, and suppose without loss of generality that $\xi = \bar{x} + \epsilon$, we have

$$\begin{aligned} x_{i_t}^{t+1} &= \bar{x} + \operatorname*{argmin}_{\epsilon} f_{i_t}(\bar{x} + \epsilon\,; x^t) && \text{such that } \bar{x} + \epsilon \in \mathcal{P}_{i_t} \\ &= \bar{x} + \operatorname*{argmin}_{\epsilon} \langle D_{f_{i_t}(\xi\,; x^t)}(\bar{x}), \epsilon\rangle + \|\epsilon\|_{\mathcal{F}_i}^2 O_{i_t}^t && \text{such that } \bar{x} + \epsilon \in \mathcal{P}_{i_t} \text{ by (5.4).} \end{aligned}$$

By definition of $\bar{x}$, $D_{f_{i_t}(\xi\,; x^t)}(\bar{x}) = 0$, and by definition, $O_{i_t}^t = \|x_j\|_{\mathcal{F}_i}^2$ for a certain $j \neq i$. By assumption, $O_{i_t}^t \geq \delta > 0$. Therefore we aim to solve

$$x_{i_t}^{t+1} = \bar{x} + \operatorname*{argmin}_{\epsilon} \|\epsilon\|_{\mathcal{F}_i}^2 \qquad \text{such that } \bar{x} + \epsilon \in \mathcal{P}_{i_t}.$$

And thus, the optimal update is

$$x_{i_t}^{t+1} = \operatorname*{argmin}_{\xi} \|\xi - \bar{x}\|_{\mathcal{F}_i}^2 \quad \text{such that} \quad \xi \in \mathcal{P}_{i_t},$$

which is the projection of $\bar{x}$ on set $\mathcal{P}_{i_t}$ and is thus the definition of the update of H-HALS. □

We will see in the next section that the condition $\|x_i\|_{\mathcal{F}_i}^2 \geq \delta > 0\ \forall i = 1, \ldots, 2r$, is often a reasonable condition.

5.2 Convergence analysis of BCD methods for H-NMF

When then H-NMF problem (Definition 4.4) is considered as one unique block, the convergence of Algorithm 4.1 is the convergence of the chosen

nonlinear least-squares solver. When two or $2r$ blocks are considered, we suppose in this section that the update of each block is performed exactly. This is a strong assumption for a two blocks decomposition. Therefore, the following results are more suited for the H-HALS Algorithm 4.3.

It is known that every limit point of HALS on standard NMF is a stationary point if the norm of the columns of $\boldsymbol{A}$ and $\boldsymbol{X}$ never goes to zero [75]. If we furthermore impose the norm of $\boldsymbol{A}_{:k}$ and $\boldsymbol{X}_{:k}$ to stay bounded for all k, the feasible set of the problem is compact, and by the Bolzano-Weierstrass theorem we know that there is a least one converging subsequence, converging thus to a stationary point. This last assumption can be done without loss of generality, see [46], Section 8.1.4 for more details. We observe in this section that similar convergence properties can be found for H-HALS, under certain conditions on the sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$.

Recall the following definition of stationarity, used when the feasible set is convex.

Definition 5.1. Stationarity Consider the real Hilbert space $\mathcal{F}$, the feasible convex set $\mathcal{P} \subseteq \mathcal{F}$, and the minimization problem with continuously differentiable function (in Fréchet sense) $f : \mathcal{F} \to \mathbb{R}$

$$\min_x f(x) \quad \text{such that } x \in \mathcal{P}.$$

We stay that $x^* \in \mathcal{P}$ is a stationary point if and only if $\langle D_f(x^*), x - x^* \rangle_{\mathcal{F}} \geq 0$ $\forall x \in \mathcal{P}$.

Consider $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_n$ with inner product $\sum_{i=1}^n \langle x_i, y_i \rangle_{\mathcal{F}_i}$, and f_i^* defined as $f_i^*(\xi) = f_i(\xi\,; x^*)$ $\forall \xi \in \mathcal{F}_i$. If f is continuously block-differentiable, i.e. all $f_i(\xi\,; x)$ are continuously differentiable for all $x \in \mathcal{P}$, we have for (5.2):

$$\langle D_f(x^*), x - x^* \rangle_{\mathcal{F}} = \sum_{i=1}^n \langle D_{f_i^*}(x_i^*), x_i - x_i^* \rangle_{\mathcal{F}_i}. \tag{5.6}$$

We can also consider a Nash equilibrium whose definition is the following.

Definition 5.2. Nash equilibrium Considering the same problem as in Definition 5.1, x^* is a Nash equilibrium if and only if $x^* \in \mathcal{P}$ and for all $i = 1, \ldots, n$

$$f(x_1^*, \cdots, x_{i-1}^*, x_i^*, x_{i+1}^*, \cdots, x_n^*) \leq f(x_1^*, \cdots, x_{i-1}^*, \xi, x_{i+1}^*, \cdots, x_n^*) \quad \forall \xi \in \mathcal{P}_i.$$

This definition is equivalent to saying that $f(x^*) = f_i(x_i^*; x^*) \leq f_i(\xi\,; x^*)$ $\forall \xi \in \mathcal{P}_i$, $\forall i = 1, \ldots, n$. In certain situations those two definitions are equivalents.

Lemma 5.3. *Suppose that the feasible set $\mathcal{P}$ is a Cartesian product of convex blocks $\mathcal{P}_i$, and $f_i(\xi\,; x)$ is convex and continuously differentiable on ξ for all $x \in \mathcal{P}$ and $i = 1, \ldots, n$. Then, x^* is a Nash equilibrium if and only if it is a stationary point.*

Proof. Suppose x^* is a Nash equilibrium, and assume by contradiction that it is not a stationary point, implying that there exists $x \in \mathcal{P}$ such that $\langle D_f(x^*),\, x - x^* \rangle_{\mathcal{F}} < 0$. As the feasible domain is the Cartesian product of $\mathcal{P}_i$, and by hypothesis on $f_i(\xi\,; x)$, Equation (5.6) holds, which implies that it must exist an index i such that $x_i \in \mathcal{P}_i$ and $\langle D_{f_i^*}(x_i^*), x_i - x_i^* \rangle_{\mathcal{F}_i} < 0$. We have for all $\lambda \in [0, 1]$

$$f_i^*(x_i^* + \lambda(x_i - x_i^*)) = f_i^*(x_i^*) + \lambda \langle D_{f_i^*}(x_i^*), x_i - x_i^* \rangle_{\mathcal{F}_i} + e(\lambda(x_i - x_i^*)),$$

where $\lim_{\lambda \to 0} \frac{|e(\lambda(x_i - x_i^*))|}{\lambda} = 0$ as f_i^* is differentiable. Therefore, $e(\lambda(x_i - x_i^*))$ goes faster to 0 than λ, so that we can find a feasible point (as $\mathcal{P}_i$ is convex) $\xi = x_i^* + \lambda(x_i - x_i^*)$ with λ small enough such that $f_i^*(\xi) < f_i^*(x_i^*)$, which is a contradiction. Therefore a Nash equilibrium is a stationary point.

In the other direction, when x^* is a stationary point, using that f_i^* is convex and differentiable, we have

$$0 \leq \langle D_{f_i^*}(x_i^*), \xi - x_i^* \rangle_{\mathcal{F}_i} \leq f_i^*(\xi) - f_i^*(x_i^*) \quad \forall \xi \in \mathcal{P}_i \; \forall i,$$

and thus $f(x^*) = f_i^*(x_i^*) \leq f_i^*(\xi)$ $\forall \xi \in \mathcal{P}_i$ $\forall i$ which concludes the proof. □

To improve readability, in what follows, the space $\mathcal{F}_i$ is no longer mentioned explicitly in the norms and inner products. We now present some useful lemma and assumptions in order to be able to describe the convergence properties of the BCD Algorithm 5.1.

Lemma 5.4. *Consider a sequence $\{x^t\}$ with n blocks, where x^{t+1} differs from x^t only along block i_t, which is such that $f(x^t) - f(x^{t+1}) \geq \delta \|x_{i_t}^{t+1} - x_{i_t}^t\|^2$ with*

$\delta > 0$, and f is a lower-bounded function. We have

$$\sum_{t=0}^{\infty} \sum_{i=1}^{n} \|x_i^{t+1} - x_i^t\|^2 < \infty.$$

Proof. Using the hypothesis on the update, we have

$$f(x^0) - f(x^T) = \sum_{t=0}^{T-1} f(x^t) - f(x^{t+1}) \geq \sum_{t=0}^{T-1} \delta \|x_{i_t}^t - x_{i_t}^{t+1}\|^2.$$

As f lower bounded, we can say that

$$\sum_{t=0}^{\infty} \|x_{i_t}^t - x_{i_t}^{t+1}\|^2 < \infty.$$

This concludes the proof since x^{t+1} differs from x^t only along block i. □

Corollary 5.1. *Consider a sequence $\{x^t\}$ with n blocks, where x^{t+1} differs from x^t only along block i_t, which is such that $\langle D_{f_{i_t}(\xi\,;x^t)}(x_{i_t}^{t+1}), x_{i_t}^t - x_{i_t}^{t+1}\rangle \geq 0$. Moreover, suppose f is a lower bounded function and $f_{i_t}(\xi\,;x^t)$ is 2δ-strongly convex with $\delta > 0$. Then $\sum_{t=0}^{\infty}\sum_{i=1}^{n} \|x_i^{t+1} - x_i^t\|^2 < \infty$.*

Proof. As $f_{i_t}(\xi\,;x^t)$ is 2δ-strongly convex, we have:

$$f_{i_t}(x_{i_t}^t\,;x^t) \geq f_{i_t}(x_{i_t}^{t+1}\,;x^t) + \langle D_{f_{i_t}(\xi\,;x^t)}(x_{i_t}^{t+1}), x_{i_t}^t - x_{i_t}^{t+1}\rangle + \delta\|x_{i_t}^t - x_{i_t}^{t+1}\|^2.$$

Using the hypothesis on the update,

$$f(x^t) - f(x^{t+1}) \geq \delta\|x_{i_t}^t - x_{i_t}^{t+1}\|^2.$$

Proof is concluded using Lemma 5.4. □

This leads us to one main convergence theorem, where we suppose that block functions are 2δ-strongly convex, a relatively strong assumption. Nevertheless, we will see that this assumption is satisfied by the H-HALS algorithm under mild conditions (see Lemma 5.6) .

The proof hereunder is very similar to those in [14] and [126], but we transcribe it anyway because in [14] and [126] the proofs are written for Euclidean spaces, and we work in RKHS instead. Moreover, we will use this

proof as a basis for other reasonings in the following section. In fact, working in RKHS spaces rather than Euclidean spaces does not raise any particular problem in the proof.

Theorem 5.2. ***Convergence BCD*** *Consider the BCD Algorithm 5.1, where all block functions $f_i(\xi\,; x^t)$ are 2δ-strongly convex, with $\delta > 0$ and continuously differentiable. Moreover, suppose that feasible sets $\mathcal{P}_i$ are closed and convex, and that cost function is lower-bounded. Every limit point of a sequence generated by this algorithm is a stationary point.*

Proof. Suppose $\bar{x}$ is a limit point of the iterates $\{x^t\}$ of the BCD algorithm. As all feasible sets $\mathcal{P}_i$ are closed, $\bar{x}$ is also feasible. By Equation (5.3)

$$f(x^t) \geq f(x^{t+1}) \qquad \forall k. \tag{5.7}$$

Moreover, as the cost function is lower-bounded, $\{f(x^t)\}$ converges to $f(\bar{x})$.

As the update is optimal, $x_{i_t}^t$ is a Nash equilibrium of $f_{i_t}(\xi\,; x^t)$. As $\mathcal{P}_i$ is convex, we have by Lemma 5.3 $\langle D_{f_{i_t}(\xi\,; x^t)}(x_{i_t}^{t+1}), x_{i_t}^t - x_{i_t}^{t+1}\rangle \geq 0$. Therefore, because block functions $f_{i_t}(\xi\,; x^t)$ are 2δ-strongly convex, we can use Corollary 5.1 to obtain:

$$\lim_{t\to\infty} \sum_{i=1}^{n} \|x_i^{t+1} - x_i^t\|^2 = 0.$$

Suppose $\{x^{t_j}\}$ converges to the limit point $\bar{x}$. Then, $\{x^{t_j+1}\}$ also converges to $\bar{x}$, and so does $\{x^{t_j+N}\}$, for any finite number N.

Consider $\{x^{t_j+1}\}$ a subsequence of $\{x^t\}$ that converges to $\bar{x}$ and is the result of an update of block i (x^{t_j+1} differs from x^{t_j} only along the i^{th} block). Such a subsequence exists for all i as in a cycle of $N = n$ updates each block is updated once. In this case, we have by definition of the update that:

$$f_i(x_i^{t_j+1}\,; x^{t_j}) \leq f_i(\xi\,; x^{t_j}) \qquad \forall \xi \in \mathcal{P}_i \qquad \forall i = 1, \dots n.$$

Taking the limit of t_j going to infinity, as f is continuous, we obtain:

$$f_i(\bar{x}_i\,; \bar{x}) \leq f_i(\xi\,; \bar{x}) \qquad \forall \xi \in \mathcal{P}_i \qquad \forall i = 1, \dots n. \tag{5.8}$$

As $\mathcal{P}_i$ is convex, and by the first order optimality condition, we have that:

$$\langle D_{f_i(x\,;\bar{x}_i)}(\bar{x}_i), \xi - \bar{x}_i \rangle \geq 0 \quad \forall \xi \in \mathcal{P}_i \qquad \forall i = 1, \dots n.$$

As the set $\mathcal{P}$ is Cartesian, we can conclude that

$$\langle D_f(\bar{x}), x - \bar{x} \rangle \geq 0 \quad \forall x \in \mathcal{P},$$

which is the definition of a stationary point and conclude the proof. □

We now present three assumptions. The two first ones are useful to ensure that every limit point of a sequence generated by the BCD algorithm for the H-NMF problem converges to a stationary point, while the third one ensures that any sequence generated by the BCD algorithm for the H-NMF problem has a convergent subsequence.

Assumption 5.1. ***Convex and Non null*** *Suppose that $\tilde{\mathcal{A}} \subseteq \mathcal{A}$ and $\tilde{\mathcal{X}} \subseteq \mathcal{X}$ are closed convex sets whose elements have norm larger than δ : $\|A_k\|^2_{\mathcal{A}} \geq \delta > 0$, $\forall A_k \in \tilde{\mathcal{A}}$, and $\|X_k\|^2_{\mathcal{X}} \geq \delta > 0$, $\forall X_k \in \tilde{\mathcal{X}}$.*

Note that the Non null assumption can often be satisfied by imposing factors to be larger than ϵ at all points in their domain (instead of only being nonnegative): $A(a) \geq \epsilon \; \forall a \in \mathbb{A}$ and $X(x) \geq \epsilon \; \forall x \in \mathbb{X}$, and this restriction does not affect the convexity of the set, as shown by lemma below.

Lemma 5.5. *Suppose $\mathcal{A}^*$ is a convex set of functions defined in $\mathbb{A} \mapsto \mathbb{R}$, lying in a RKHS. Let $\tilde{\mathcal{A}}$ be the restriction of $\mathcal{A}^*$ to elements that are pointwise larger than ϵ everywhere: $\tilde{\mathcal{A}} = \{A \in \mathcal{A}^* | A(a) \geq \epsilon \; \forall a \in \mathbb{A}\}$. Then $\tilde{\mathcal{A}}$ is also convex.*

Proof. As the evaluation functional of Hilbert spaces (and thus RKHS) is linear, we have $\forall A_1, A_2 \in \tilde{\mathcal{A}}$, $t \in [0, 1]$

$$(tA_1 + (1-t)A_2)(a) = tA_1(a) + (1-t)A_2(a) \geq \epsilon \; \forall a \in \mathbb{A}.$$

Moreover, as $\mathcal{A}^*$ is convex, $tA_1 + (1-t)A_2 \in \mathcal{A}^*$ and thus $tA_1 + (1-t)A_2 \in \tilde{\mathcal{A}}$. □

We observe with this lemma that imposing nonnegativity, or restricting all functions to be larger than ϵ, does not affect the convexity of the constrained set. Convexity is thus the main requirement in the Convex and Non null assumption.

Assumption 5.2. ***Convex - 2 blocks*** *Suppose $\tilde{\mathcal{A}} \subseteq \mathcal{A}$ and $\tilde{\mathcal{X}} \subseteq \mathcal{X}$ are closed convex sets. Let $\boldsymbol{M}^A \in \mathbb{R}^{r\times r}$ defined by $M^A_{ij} = \langle A_i, A_j\rangle_{\mathcal{A}}$ and $\boldsymbol{M}^X \in \mathbb{R}^{r\times r}$ defined by $M^X_{ij} = \langle X_i, X_j\rangle_{\mathcal{X}}$. Suppose that throughout algorithm $\boldsymbol{M}^A$ and $\boldsymbol{M}^X$ are invertible and trace$\big(\boldsymbol{M}^{A^{-1}}\big) \leq \delta^{-1}$ and trace$\big(\boldsymbol{M}^{A^{-1}}\big) \leq \delta^{-1}$.*

The two assumptions above ensure to have strongly-convex functions, as proved by the Lemma bellow.

Lemma 5.6. *Assumptions 5.1 and 5.2 ensure that the block functions $f_i(\xi\,; x^t)$ are 2δ-strongly convex throughout the BCD algorithm.*

Proof. Assumption 5.1 is for the $2r$ blocks decomposition, and can be easily verified using Lemma 5.1.

Assumption 5.2 is for the two blocks decomposition. Indeed, by definition of strong-convexity $f_1(\xi\,; x)$ is strongly convex if and only if

$$\langle D_{f_1(\xi\,;x)}(\xi_1) - D_{f_1(\xi\,;x)}(\xi_2), \xi_1 - \xi_2\rangle_{\mathcal{F}_i} \geq \mu\|\xi_1 - \xi_2\|^2_{\mathcal{F}_i}$$

Using the definition of the cost function and its derivative, this means

$$\begin{aligned}
&\sum_{i=1}^{r}\Big\langle \sum_{k=1}^{r} 2(A_1 - A_2)_k\langle X_k, X_i\rangle_{\mathcal{X}}, (A_1 - A_2)_i\Big\rangle_{\mathcal{A}} \geq \mu\|A_1 - A_2\|^2_{\mathcal{A}}\\
\Leftrightarrow\ &\sum_{i=1}^{r}\sum_{k=1}^{r} \boldsymbol{M}^X_{ki}\boldsymbol{M}^A_{ki} \geq \mu\sum_{k=1}^{r}\boldsymbol{M}^A_{kk}\\
\Leftrightarrow\ &2\langle \boldsymbol{M}^X, \boldsymbol{M}^A\rangle \geq \mu\langle \boldsymbol{I}^r, \boldsymbol{M}^A\rangle
\end{aligned}$$

Using Cholesky decompositions ${\boldsymbol{L}^X}^\top\boldsymbol{L}^X = \boldsymbol{M}^X$ and ${\boldsymbol{L}^A}^\top\boldsymbol{L}^A = \boldsymbol{M}^A$, this becomes

$$\Leftrightarrow 2\|\boldsymbol{L}^X{\boldsymbol{L}^A}^\top\|^2_F \geq \mu\|\boldsymbol{L}^A\|^2_F$$

If $\boldsymbol{M}^X$ is invertible, so is $\boldsymbol{L}^X$, and $\|\boldsymbol{L}^{X^{-1}}\boldsymbol{L}^X{\boldsymbol{L}^A}^\top\|^2_F \leq \|\boldsymbol{L}^{X^{-1}}\|^2_F\|\boldsymbol{L}^X{\boldsymbol{L}^A}^\top\|^2_F$ and therefore

$$\Leftrightarrow 2\frac{\|\boldsymbol{L}^A\|^2_F}{\|\boldsymbol{L}^{X^{-1}}\|^2_F} \geq \mu\|\boldsymbol{L}^A\|^2_F$$

As $\|\boldsymbol{L}^{X^{-1}}\|_F^2 = \langle \boldsymbol{I}^r, \boldsymbol{M}^{X^{-1}} \rangle = \text{trace}\big(\boldsymbol{M}^{X^{-1}}\big)$, we have

$$\Leftrightarrow \frac{2}{\mu} \geq \text{trace}\big(\boldsymbol{M}^{X^{-1}}\big).$$

The same reasoning can be done for $f_2(\xi\,; x)$, and the block functions are thus 2δ-strongly convex. □

We present now the third assumption.

Assumption 5.3. ***Boundedness*** *$\tilde{\mathcal{A}} \subseteq \mathcal{A}$ and $\tilde{\mathcal{X}} \subseteq \mathcal{X}$ are bounded sets. Moreover, if $\mathcal{A}$ is an infinite dimensional space, for every $\epsilon > 0$ there is a finite-dimensional subspace $W \subset \mathcal{A}$ such that $\inf_{w \in W} \|A - w\| < \epsilon \; \forall A \in \tilde{\mathcal{A}}$. Similarly, if $\mathcal{X}$ is an infinite dimensional space, for every $\epsilon > 0$ there is a finite-dimensional subspace $W \subset \mathcal{X}$ such that $\inf_{w \in W} \|X - w\| < \epsilon \; \forall X \in \tilde{\mathcal{X}}$.*

To satisfy this assumption, we can use the following lemma.

Lemma 5.7. *Consider H-NMF problem under the Convex and Non null Assumption (5.1) where sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ are such that $\forall A_i, A_j \in \tilde{\mathcal{A}}$, $X_i, X_j \in \tilde{\mathcal{X}}$, we have (nonnegativity condition): $\langle A_i, A_j \rangle \langle X_i, X_j \rangle \geq 0$.*

Moreover, suppose that $\alpha A \in \tilde{\mathcal{A}} \; \forall \alpha \geq \min\big(1, \frac{\gamma}{\|A\|_{\mathcal{A}}}\big)$, for a $\gamma > 0$, and similarly $\alpha X \in \tilde{\mathcal{X}} \; \forall \alpha \geq \min\big(1, \frac{\gamma}{\|X\|_{\mathcal{X}}}\big)$, i.e. it is possible to scale elements in $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$.

If problem is solved using BCD Algorithm (5.1), we can consider without loss of generality that sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ are bounded, with $\|A\|_{\mathcal{A}}^2, \|X\|_{\mathcal{X}}^2 \leq \max\big(\gamma^2, \|Y\|_{\mathcal{H}} + \|Y - \sum_{k=1}^r A_k^0 \otimes X_k^0\|_{\mathcal{H}}\big)$, where $\{A_k^0\}_{k=1}^r$, $\{X_k^0\}_{k=1}^r$ are the initial iterates of the algorithm.

Proof. By triangular inequality, and the fact that Algorithm 5.1 decreases the cost function at each iteration, we have:

$$\Big\|Y - \sum_{k=1}^r A_k^0 \otimes X_k^0\Big\|_{\mathcal{H}} \geq \Big\|Y - \sum_{k=1}^r A_k^t \otimes X_k^t\Big\|_{\mathcal{H}} \geq \Big\|\sum_{k=1}^r A_k^t \otimes X_k^t\Big\|_{\mathcal{H}} - \|Y\|_{\mathcal{H}}.$$

By the nonnegativity condition, we have $\big\|\sum_{k=1}^r A_k^t \otimes X_k^t\big\|_{\mathcal{H}}^2 \geq \sum_{k=1}^r \|A_k^t \otimes X_k^t\|_{\mathcal{H}}^2 \geq \|A_s^t \otimes X_s^t\|_{\mathcal{H}}^2 \; \forall s$. Therefore

$$\|Y\|_{\mathcal{H}} + \Big\|Y - \sum_{k=1}^r A_k^0 \otimes X_k^0\Big\|_{\mathcal{H}} \geq \|A_s^t \otimes X_s^t\|_{\mathcal{H}} = \|A_s^t\|_{\mathcal{A}} \|X_s^t\|_{\mathcal{X}} \; \forall s.$$

Hence it is always possible to find $\alpha \geq \min\left(1, \frac{\gamma}{\|A_s^t\|_{\mathcal{A}}}\right)$; $\frac{1}{\alpha} \geq \min\left(1, \frac{\gamma}{\|X_s^t\|_{\mathcal{X}}}\right)$ so that

$$\alpha^2 \|A_s^t\|_{\mathcal{A}}^2, \frac{1}{\alpha^2}\|X_s^t\|_{\mathcal{X}}^2 \leq \max\left(\gamma^2, \|Y\|_{\mathcal{H}} + \left\|Y - \sum_{k=1}^{r} A_k^0 \otimes X_k^0\right\|_{\mathcal{H}}\right).$$

By hypothesis, αA_s^t and $\frac{1}{\alpha} X_s^t$ are feasible points, which lead to the same cost as A_s^t and X_s^t.

An example of feasible α is

$$\alpha = \begin{cases} \sqrt{\frac{\|X_s^t\|}{\|A_s^t\|}} & \text{if } \|A_s^t\|\|X_s^t\| \geq \gamma^2 \\ \frac{\gamma}{\|A_s^t\|} & \text{else if } \|A_s^t\| \geq \gamma \\ \frac{\|X_s^t\|}{\gamma} & \text{else if} \|X_s^t\| \geq \gamma \\ 1 & \text{otherwise.} \end{cases} \tag{5.9}$$

Let us modify slightly Algorithm 5.1: let $\tilde{x}_i^{t+1}$ be the solution of Equation (5.3), and define $x^{t+1} = g(\tilde{x}^{t+1})$, where g is a continuous function that rescales A_s^t, X_s^t as αA_s^t, $\frac{1}{\alpha} X_s^t$ with α from (5.9). It is easy to check that conclusion of Lemma 5.4 becomes $\sum_{t=0}^{\infty}\sum_{i=1}^{n} \|\tilde{x}_i^{t+1} - x_i^k\|^2 < \infty$. Therefore, if $\{x^{t_j}\}$ converges to $\bar{x}$, so does $\{\tilde{x}^{t_j+1}\}$. As g is a continuous function, $g(\tilde{x}^{t_j+1})$ converges to $g(\bar{x})$. The sequence $\{x^{t_j}\}$ is by definition contained in the bounded closed sets $\bar{\mathcal{A}}$ and $\bar{\mathcal{X}}$, therefore $\bar{x}$ is finite, and $g(\bar{x}) = \bar{x}$, and $\{x^{t_j+1}\}$ converges to $\bar{x}$ as well. Therefore, the proof of Theorem 5.2 still holds, and we can consider without loss of generality that $\bar{\mathcal{A}}$ and $\bar{\mathcal{X}}$ are bounded. □

We can now present the convergence theorems for the H-NMF algorithms.

Theorem 5.3. ***Convergence of H-HALS*** *Consider the H-NMF problem (Definition 4.4) under Assumption 5.1, solved using H-HALS (Algorithm 4.3). Every limit point of the generated sequence is a stationary point.*

If in addition the Boundedness Assumption (5.3) is satisfied, any sequence generated by H-HALS has a convergent subsequence converging to a stationary point.

Proof. By Assumption 5.1 and Lemma 5.6, the conditions of Lemma 5.2 are satisfied, and H-HALS is thus a BCD algorithm. Moreover, Assumption

5.1 and Lemma 5.6 also ensure that Theorem 5.2 holds. Therefore, every limit point generated by the H-HALS algorithm is a stationary point.

If the Boundedness assumption is also satisfied, sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ are compact ([92], Proposition 3.9). Therefore, any sequence generated by the H-HALS algorithm has a convergent subsequence, converging thus to a stationary point. □

This theorem thus shows that it is possible to prove the same kind of convergence for Algorithm 5.1 for H-NMF as for standard NMF, under very similar conditions. For two block decomposition, the theorem becomes:

Theorem 5.4. ***Convergence of two-blocks decomposition of H-NMF*** *Consider the H-NMF problem (Definition 4.4) under Assumption 5.2, decomposed in two blocks and solved using BCD Algorithm 5.1. Every limit point of the generated sequence is a stationary point.*

If in addition the Boundedness Assumption (5.3) is satisfied, any sequence generated by the algorithm has a convergent subsequence converging to a stationary point.

Proof. By Assumption 5.2 and Lemma 5.6, Theorem 5.2 holds, and every limit point generated by the BCD algorithm is a stationary point.

If the Boundedness assumption is also satisfied, sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ are compact, and any sequence generated by the BCD algorithm has a convergent subsequence converging to a stationary point. □

Note that as the sets $\mathcal{F}_i$ are all RKHS, $\bar{x}_i(a) = \lim_{t_j \mapsto \infty} x_i^{t_j}(a)\ \forall a$ (Property 4.1.5) which means that the evaluation of the factorization also converges.

Comment about the convergence results We have seen in Assumption 5.1 and Lemma 5.5 that the strongest assumption required for H-HALS to converge is the convexity of the set of constraints. The sets considered in Chapter 3, namely polynomials or splines of fixed degree, satisfy this condition.

Nevertheless, this condition can be restrictive. We will see in Chapter 8 a case where the considered set of functions is not convex, namely the case

of nonnegative rational functions of fixed degree. It turns out that using H-HALS is more delicate in this case. Nevertheless, in practice, the algorithm gives good results on average.

On the other hand, a commonly used constraint in classical NMF is not convex: the sparsity constraint. This means that the tools developed in Chapter 4 do not apply in this case (or at least without theoretical guarantees). Nevertheless, many of the tools developed for classical NMF with sparsity constraints could be adapted to H-NMF (e.g. regularization, or the $\boldsymbol{ASX}^\top$ factorization presented in Section 2.2, coming from [66, 98, 132]). This could be an interesting direction for future research.

It should also be noted that sparsity does not make sense for all functions. For example, looking for polynomials of low degree with many roots is not very judicious, while sparsity has much more sense in the case of splines (see earlier discussion at the end of Section 4.3). Moreover, sparsity may be imposed or promoted on only one of the two factors (and actually, this often makes sense). If, for example, we are working in a situation as in Chapter 3 where one factor contains functions and the other is a matrix, then we can encourage sparsity of the matrix with the usual techniques used in NMF. These techniques do not always have theoretical guarantees, but give good results in practice.

5.3 Convergence using inexact updates

In this section, we analyze the behavior of block-coordinate descent (BCD) algorithm (sketched in Algorithm 5.1) when the updates are inexact. We consider problems where the block functions $f_i(\xi, x)$ derived from the cost function are continuous and strongly convex, and the feasible set is a Cartesian product of the feasible sets of the blocks. This is the case in the H-HALS Algorithm (4.3), provided that the iterates do not become zero during the resolution.

Studying the behavior of inexact BCD has two main interests. First, it allows us to study the behavior of H-HALS when the Convex and Non null Assumption (5.1) is not satisfied. Indeed, the fact that the factors can not have a zero norm is a known problem in NMF and is most often prevented by excluding elements with too small norm from the set of solutions (which

is the equivalent of our Non null assumption in Assumption 5.1) [75, 76]. Some works have considered algorithms that allow a factor to go to zero (e.g. [110]), but this leads to a rank reduction during the factorization, which is usually not a desirable behavior. Indeed, if a factor is set to 0 at some point in the execution of the algorithm, it will often get stuck at 0, regardless of the chosen rank. However, if a factor tends to 0 throughout algorithm, it indicates that the chosen rank is too high and the factor in question can still be manually truncated at the end of the algorithm if this makes sense. In this section, we explore a variation where excluding those null elements from the solution set is not needed, while preventing the iterates from being zero during resolution, and thus ensuring convergence.

The second and main interest is that the projection step in Algorithm 4.3 is not trivial and can not always be performed exactly. The goal of this section is to have some theoretical information about the convergence when the projection is not exact. The presented inexact cases are illustrated using H-NMF on splines using the B-Spline basis.

We analyze three approaches with inaccuracy: using ϵ-stationarity, approximating indicator functions or using general inexact updates.

5.3.1 Using $\epsilon-$stationarity

Modifying the definition of stationarity (Definition 5.1), we describe what is called ϵ-stationarity [67]:

Definition 5.3. ϵ-stationarity Consider the real Hilbert space $\mathcal{F}$, the convex set of constraints $\mathcal{P} \subseteq \mathcal{F}$, and the minimization problem with continuously differentiable function (in Fréchet sense) $f : \mathcal{F} \to \mathbb{R}$

$$\min_x f(x) \quad \text{such that } x \in \mathcal{P}.$$

Given $\epsilon > 0$, point x^* is an ϵ-stationary point if and only if $\langle D_f(x^*), x - x^* \rangle_{\mathcal{F}} \geq -\epsilon \; \forall x \in \mathcal{P}$.

From this definition, we define ϵ-valid updates:

Definition 5.4. ϵ-valid updates Let $\delta > 0$ and $\epsilon > 0$. The iterate $x_i^{t+1} = \xi$ is an ϵ-valid update if and only if

$$\begin{array}{lll} 1.\ \xi \in \mathcal{P}_i & & \text{(feasibility)} \\ 2.\ \langle D_{f_i(x\,;x^t)}(\xi), x_i - \xi \rangle \geq -\epsilon \ \ \forall x_i \in \mathcal{P}_i & & (\epsilon\text{-stationarity}) \\ 3.\ f_i(x_i^t\,;x^t) - f_i(\xi\,;x^t) \geq \delta \|\xi - x_i^t\|^2 & & \text{(improving quality)} \end{array} \tag{5.10}$$

Note that if $f_i(x\,;x^t)$ is 2δ-strongly convex, the optimal value $x_i^* = \underset{x\in\mathcal{P}_i}{\operatorname{argmin}} f_i(x\,;x^t)$ is ϵ-valid. Indeed, as x_i^* is optimal the two first conditions are satisfied. We can see that condition 3 is also satisfied following a similar reasoning as in Corollary 5.1. Therefore, ϵ-valid updates are more general than usual exact updates.

Assessing ϵ-validity can be hard in practice, due to the second criterion. Nevertheless, there are situation where assessing this criterion is possible, for example when using nonnegative vectors (Theorem 5.7), nonnegative splines (see end of this section), or nonnegative polynomials (Lemma 6.2). The ϵ-validity property is then used to allow zero-norm elements in $\mathcal{P}$, or as a criterion to evaluate the quality of a heuristic, or as a stopping criterion for the iterative algorithm used to compute the updates.

The convergence properties of an algorithm with ϵ-valid updates can be analyzed using an approach similar to that of the BCD algorithm (Theorem 5.2).

Theorem 5.5. *Consider problem (5.2) where cost function f is lower-bounded, with n blocks, and blocks $f_i(\xi\,;x)$ are continuously differentiable for all $x \in \mathcal{P}$. Moreover, sets $\mathcal{P}_i$ are closed and convex. Assume that this problem is solved using a BCD approach where Equation (5.3) is replaced by an update satisfying (5.10).*

Every limit point of a sequence generated by such algorithm is a $n\epsilon$-stationary point.

Proof. Suppose $\bar{x}$ is a limit point of $\{x^t\}$. By Lemma 5.1 and the third hypothesis of (5.10), we have $f(x^t) \geq f(x^{t+1})$. Therefore, f monotonically decreases. As f is lower bounded, $\{f(x^t)\}$ converges to $f(\bar{x})$. By hypothesis 3, Lemma 5.4 holds, which means that $\sum_{t=0}^{\infty}\sum_{i=1}^{n} \|x_i^{t+1} - x_i^t\|^2 < \infty$. Moreover, if $\{x^{t_j}\}$ converges to the limit point $\bar{x}$ so do $\{x^{t_j+N}\}$ for any finite N.

Consider $\{x^{t_j+1}\}$ a subsequence of $\{x^t\}$ that converges to $\bar{x}$ and is the re-

sult of an update of block i. As $\mathcal{P}$ is a closed set, $\bar{x}$ is a feasible point by hypothesis 1. By the ϵ-stationarity of the update, we have:

$$\langle D_{f_i(x\,;x^t)}(x_i^{t_j+1}), \xi - x_i^{t_j+1} \rangle \geq -\epsilon \quad \forall \xi \in \mathcal{P}_i \qquad \forall i.$$

As the cost function is continuously block-differentiable, the left part of this equation is continuous. Taking the limit of t_j going to infinity, we obtain for all blocks i

$$\langle D_{f_i(x\,;\bar{x})}(\bar{x}), \xi - \bar{x}_i \rangle \geq -\epsilon \quad \forall \xi \in \mathcal{P}_i \qquad \forall i.$$

As the set $\mathcal{P}$ is the Cartesian product of $\mathcal{P}_i$, we can conclude that

$$\langle D_f(\bar{x}), x - \bar{x} \rangle = \sum_{i=1}^{n} \langle D_{f_i(\xi\,;\bar{x})}(\bar{x}_i), x_i - \bar{x}_i \rangle \geq -n\epsilon \quad \forall x \in \mathcal{P}$$

which means that every limit point of the algorithm is a $n\epsilon$-stationary point. □

Corollary 5.6. *Consider H-NMF from Definition 4.4 with n the number of blocks, equals to two or 2r, and suppose that $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ are closed and convex sets. Note that $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ can contain 0-norm elements. This problem is solved using a BCD approach where Equation (5.3) is replaced by a ϵ-valid update.*

Every limit point of a sequence generated by such algorithm is a $n\epsilon$-stationary point.

This corollary suggests that it is sometimes possible to remove the nonzero norm condition of sets $\mathcal{A}$ and $\mathcal{X}$ in H-NMF and still converge to an ϵ-stationary point. This is confirmed by theorem below.

Theorem 5.7. *Consider the H-NMF problem from Definition 4.4. Suppose $\mathcal{A} = \mathbb{R}^m$, $\mathcal{X} = \mathbb{R}^n$, $\tilde{\mathcal{A}} = [0, M]^m$ and $\tilde{\mathcal{X}} = [0, M]^m$ (standard NMF problem bounded above by M).*

Suppose that the problem is divided in 2r blocks, the columns of $\boldsymbol{A}$ and $\boldsymbol{X}$, and solved using BCD Algorithm 5.1, with update (5.3) replaced by:

$$\begin{aligned} \tilde{x}_i^{t+1} &= \underset{\xi}{argmin} f_i(\xi\,; x^t) \quad \textit{such that } \xi \in \mathcal{P}_i \\ x_i^{t+1} &= \tilde{x}_i^{t+1} + \lambda[x_i^t - \tilde{x}_i^{t+1}]_{\mathcal{P}_i} \end{aligned} \tag{5.11}$$

where $\lambda \in [0,1]$, $\|x_i^{t+1}\|^2 \geq \delta > 0$ *and* x_i^{t+1} *is* ϵ*-valid.*

For any ϵ*, if* δ *and* x^0 *are chosen so that* $2M\sqrt{\delta \max(n,m) f(x^0)} \leq \epsilon$*, and* $\|x_i^0\| \geq \delta$ *for all* i*, then a valid update* x_i^{t+1} *always exists. The resulting algorithm converges to a* $2r\epsilon$*-stationary point.*

Proof. See Appendix A. □

This theorem proves that in certain situation it is always possible to find an ϵ-valid update without restricting sets $\mathcal{A}$ and $\mathcal{X}$ to not contain zero-norm elements.

An example using B-splines with nonnegative coefficients The B-spline basis, denoted $\mathbf{\Pi}$, is often used to represent splines and has the nice property that every element in the basis is nonnegative. Therefore, splines with nonnegative coefficients are nonnegative as well, even though some nonnegative splines have negative coefficients [31]. It is therefore possible to use splines with nonnegative coefficients to represent a subset of nonnegative functions (see e.g. [8, 141]). However, the B-spline basis is not orthogonal and it is not possible to project a spline by simply thresholding its coefficients. Instead, we can solve a constrained least square problem on the coefficients (see Section 3.2 for more details):

$$\min_{\boldsymbol{f}} \|\mathbf{\Pi}(\boldsymbol{f} - \boldsymbol{g})\|^2 \quad \text{s.t. } \boldsymbol{f} \geq 0. \tag{5.12}$$

Nevertheless, solving (5.12) is much more costly than thresholding. Therefore, we use ϵ-stationarity to try to combine the best of both, under the Convex Non null Assumption (5.1).

Based on Theorem 5.7, if $\bar{c}$ contains the coefficients of the best unconstrained update, and c_i^t contains the coefficients of the previous update x_i^t, we aim to use as update:

$$x_i^{t+1} = \Pi\Big([\bar{c}]_+ + \lambda\big[c_i^t - [\bar{c}]_+\big]_+\Big), \quad \text{for } \lambda \in [0,1]. \tag{5.13}$$

In this case, we know by Equation (5.5) that

$$D_{f_i(x\,;x^t)}(x_i^{t+1}) = D_{f_i(x\,;x^t)}(\Pi\bar{c}) + 2O_i\Pi\Big(\lambda\big[c_i^t - [\bar{c}]_+\big]_+ + [\bar{c}]_+ - \bar{c}\Big)$$

As $D_{f_{i_t}(x\ ;x^t)}(\Pi\bar{c}) = 0$, and splines with nonnegative coefficients are nonnegative, this gradient is nonnegative. This means that $\langle D_{f_i(x\ ;x^t)}(x_i^{t+1}), x\rangle \geq 0\ \forall x \in \mathcal{P}_i$. Therefore, condition 2 of ϵ-validity becomes $\langle D_{f_i(x\ ;x^t)}(x_i^{t+1}), -x_i^{t+1}\rangle \geq -\epsilon$, which can be easily verified.

We decided to work with $\epsilon = 10^{-3}$. If the thresholding of the coefficients $\Pi[\bar{c}]_+$ is ϵ-valid (i.e. when $\lambda = 0$), it is chosen as update. Otherwise, we chose the smallest $\lambda \in [0, 1]$ so that x_i^{t+1} is ϵ-valid. However, such a λ does not always exist. If Equation (5.13) is never valid, the update is performed using the exact projection. The hope is that situation does not occur too often.

To test this approach, we created a synthetic dataset $\boldsymbol{Y}^* = \bar{\boldsymbol{A}}\bar{\boldsymbol{X}}^\top$, where $\bar{\boldsymbol{A}}$ contains discretization over 200 points of 5 nonnegative splines with 30 equispaced interior knots in [-1,1]. Factor $\bar{\boldsymbol{X}} \in \mathbb{R}_+^{5\times 200}$ is a mixing matrix generated using a normal distribution, thresholded to be nonnegative. When indicated, a Gaussian noise $\boldsymbol{N}$ with noise level $20dB$ is added: $\boldsymbol{Y} = \boldsymbol{Y}^* + \boldsymbol{N}$. The accuracy of the methods is measured through the relative residues as presented in Section 2.1.1

Figure 5.1 illustrates that using a combination of thresholding and exact projection leads to the best results in terms of accuracy, just like exact projection, because these two methods converge to stationary points. However, it is faster than exact projection as it is accelerated by using some non-exact thresholding steps.

5.3.2 Using approximations of indicator functions

The following approach has been inspired from the block successive upper-bound minimization algorithm (BSUM) from [103]. The objective function of problem (5.2) can be written as

$$\min_x f(x) + \sum_{i=1}^{n} r_i(x_i),$$

where r_i is an indicator function of the considered set, meaning that $r_i(x_i) = 0$ if $x_i \in \mathcal{P}_i$, ∞ otherwise. In this section we aim to relax this problem. In [103], function f was upper bounded with an easier function u. Let us in our case replace indicators r_i with convex functions u_i. The relaxed prob-

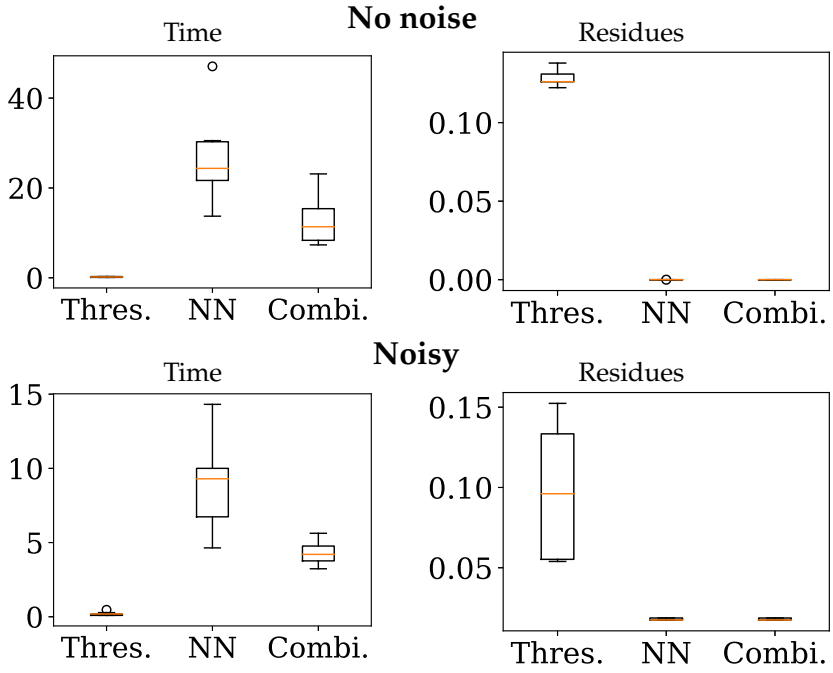


Fig. 5.1 Comparison of H-NMF on splines using thresholding (Thres.), splines with nonnegative coefficients (NN) or a combination of both (Combi.). Tests repeated three times over three initializations (9 tests in total). The upper figures are for a test without noise, while the lower figures are for a test with noise level 20dB.

lem is

$$\min_x f(x) + \sum_{i=1}^{n} u_i(x_i). \tag{5.14}$$

Note that the problem is now unconstrained. Let us analyze the convergence of a BCD algorithm on (5.14), where we allow to modify u during iterations such that

$$0 \leq u_i^{t+1}(x) \leq u_i^t(x) \qquad \forall i, t, x. \tag{5.15}$$

This condition ensures that the sequence of functions $\{u_i^t\}$ converges to a function $\bar{u}_i \ \forall i$. This condition is illustrated in Figure 5.2. We observe in this figure that an element infeasible for u_i^t has to be infeasible for all u_i^k

with $k \leq t$. At the end of this section, we present an example using two constraints u, with u^1 an easier but also more restrictive constraint that u^2.

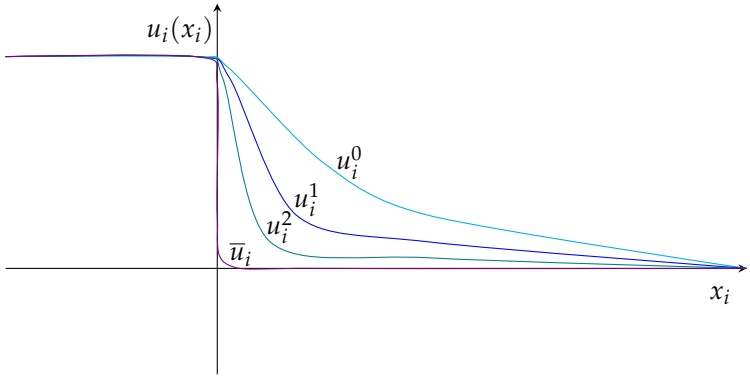


Fig. 5.2 Illustration of a valid choice of constraint functions u_i^t. In this example, points x_i should be nonnegative.

To analyze this approach, let us define $F^t(x) = f(x) + \sum_{i=1}^n u_i^t(x_i)$. We can adapt Lemma 5.4:

Lemma 5.8. *Consider the sequence $\{x^t\}$ with n blocks, where x^{t+1} differs from x^t only along block i_t, and the sequence $\{u_i^t\}$ with convex functions u_i^t satisfying (5.15). Moreover, suppose that $\langle D_{f_{i_t}(\xi\,;x^t)}(x_{i_t}^{t+1}) + p_{i_t}^{t+1}, x_{i_t}^t - x_{i_t}^{t+1}\rangle \geq 0$, for some $p_{i_t}^{t+1} \in \partial u_{i_t}^{t+1}(x_{i_t}^{t+1})$. If f is a lower bounded function and $f_{i_t}(\xi\,;x^t)$ is 2δ-strongly convex, then*

$$\sum_{t=0}^{\infty}\sum_{i=1}^{n} \|x_i^{t+1} - x_i^t\|^2 < \infty.$$

Proof. Using the fact that $u_{i_t}^{t+1}$ is convex, and $f_{i_t}(\xi\,;x^t)$ is 2δ-strongly con-

vex, we have that (for some $p_{i_t}^{t+1} \in \partial u_{i_t}^{t+1}(x_{i_t}^{t+1})$):

$$\begin{aligned} F^t(x^t) - F^{t+1}(x^{t+1}) &= f(x^t) - f(x^{t+1}) + \sum_{i=1}^{n} \left[u_i^t(x_i^t) - u_i^{t+1}(x_i^t)\right] \\ &\geq f_{i_t}(x_{i_t}^t\,; x^t) - f_{i_t}(x_{i_t}^{t+1}\,; x^t) + u_{i_t}^{t+1}(x_{i_t}^t) - u_{i_t}^{t+1}(x_{i_t}^{t+1}) \\ &\geq \langle D_{f_i(\xi\,; x^t)}(x_{i_t}^{t+1}) + p_{i_t}^{t+1}, x_{i_t}^t - x_{i_t}^{t+1}\rangle + \delta\|x_{i_t}^t - x_{i_t}^{t+1}\|^2 \\ &\geq \delta\|x_{i_t}^t - x_{i_t}^{t+1}\|^2 \quad \text{by hypothesis.} \end{aligned}$$

We can then conclude as in Lemma 5.4, using F instead of f. □

We can now prove the convergence theorem:

Theorem 5.8. *Consider problem (5.14), with convex u_i^t satisfying (5.15). Suppose that function f is lower bounded, with block functions $f_i(\xi\,; x^t)$ 2δ-strongly convex and continuously differentiable.*

Every limit point of a sequence generated by a BCD algorithm is a stationary point of

$$\min f(x) + \sum_{i=1}^{n} \bar{u}_i(x_i).$$

Proof. The proof is similar to Theorem 5.2. Suppose $\bar{x}$ is a limit point of the sequence $\{x^t\}$.

$$\begin{aligned} F^t(x^t) - F^t(x^{t+1}) &\geq F^t(x^t) - F^{t+1}(x^t) && \text{as the update is optimal} \\ &= f(x^t) + \sum_{i=1}^{n} u_i^t(x_i^t) - f(x^t) - \sum_{i=1}^{n} u_i^{t+1}(x_i^t) \\ &\geq 0 && \text{by Equation(5.15).} \end{aligned}$$

Therefore, the objective function is decreased at each step. Moreover, this objective function is lower bounded, and the sequence $\{F^t(x^t)\}$ converges to $f(\bar{x}) + \sum_{i=1}^{n} \bar{u}_i(\bar{x})$. As the update is optimal, if x^{t+1} differs from x^t only along block i_t, we know by first order optimality condition that

$$\langle D_{f_{i_t}(\xi\,; x^{t})}(x_{i_t}^{t+1}) + p_{i_t}^{t+1}, x_{i_t}^t - x_{i_t}^{t+1}\rangle \geq 0 \quad \text{for some } p_{i_t}^{t+1} \in \partial u_{i_t}^{t+1}(x_{i_t}^{t+1}) \quad \forall t \tag{5.16}$$

Therefore, Lemma 5.8 holds, which means that $\sum_{k=0}^{\infty}\sum_{i=1}^{n} \|x_i^{t+1} - x_i^t\|^2 < \infty$. Moreover, if $\{x^{t_j}\}$ converges to the limit point $\bar{x}$ so do $\{x^{t_j+N}\}$ for any

finite N. Consider $\{x^{t_j+1}\}$ a subsequence of $\{x^t\}$ that converges to $\bar{x}$ and is the result of an update of block i. As the update is optimal, we have:

$$f_i(x_i^{t_j+1}; x^{t_j}) + u_i^{t_j+1}(x_i^{t_j+1}) \leq f_i(\xi; x^{t_j}) + u_i^{t_j+1}(\xi) \; \forall \xi.$$

Taking the limit when t_j goes to infinity, we have for all blocks i

$$f_i(\bar{x}_i; \bar{x}) + \bar{u}_i(\bar{x}_i) \leq f_i(\xi; \bar{x}) + \bar{u}_i(x) \quad \forall \xi.$$

As ξ is unconstrained, using first order optimality conditions we have:

$$\langle D_{f_i(x;\bar{x})}(\bar{x}_i) + \bar{p}_i, \xi - \bar{x}_i \rangle \geq 0 \quad \forall \xi \text{ , for some } \bar{p}_i \in \partial \bar{u}_i(\bar{x}_i) \qquad \forall i.$$

We can conclude that

$$\langle D_f(\bar{x}) + \bar{p}, \xi - \bar{x} \rangle \geq 0 \quad \forall \xi, \text{ for some } \bar{p} \in \partial \bar{u}(\bar{x})$$

which means that every limit point of the algorithm is a stationary point of $\min f(x) + \sum_{i=1}^{n} \bar{u}_i(x_i)$ and concludes the proof. □

Example using B-splines As stated in previous section, splines with nonnegative coefficients are useful to represent nonnegative functions, but are not as complete as nonnegative splines [31]. Projecting splines on their nonnegative set is not trivial and uses conic optimization on a sum of squares (SOS) representation of nonnegative splines (see Section 3.2.2). This is quite costly compared to constrained least squares. From what is explained above, we can first optimize on splines with nonnegative coefficients and then on nonnegative splines.

Indeed, as the set of nonnegative splines contains the set of splines with nonnegative coefficients, its indicator function is a lower bound on the indicator function of splines with nonnegative coefficients. Function u is thus defined as follows. Let k be the first iteration so that $f(x^t) - f(x^{t+1}) \leq 10^{-3} f(x^t)$ (the algorithm starts to converge), we have:

$$u_i^t = \begin{cases} \text{indicator function of splines with nonnegative coefficients} & \text{if } t \leq k \\ \text{indicator function of nonnegative splines} & \text{if } t > k. \end{cases} \tag{5.17}$$

Additionally, we also set a time limit of 100 seconds to switch between the

two indicator functions, but this limit was never reached in practice (so the change always took place thanks to the condition on the convergence of f). Also, as the projection must be exact, we can not anymore use the thresholding from previous section. Figure 5.3 illustrates this method on same data as in Section 5.3.1. We observe that using first splines with nonnegative coefficients and then nonnegative splines help the algorithm to converge faster on average to the solution and the quality of the solution is not impacted by this change.

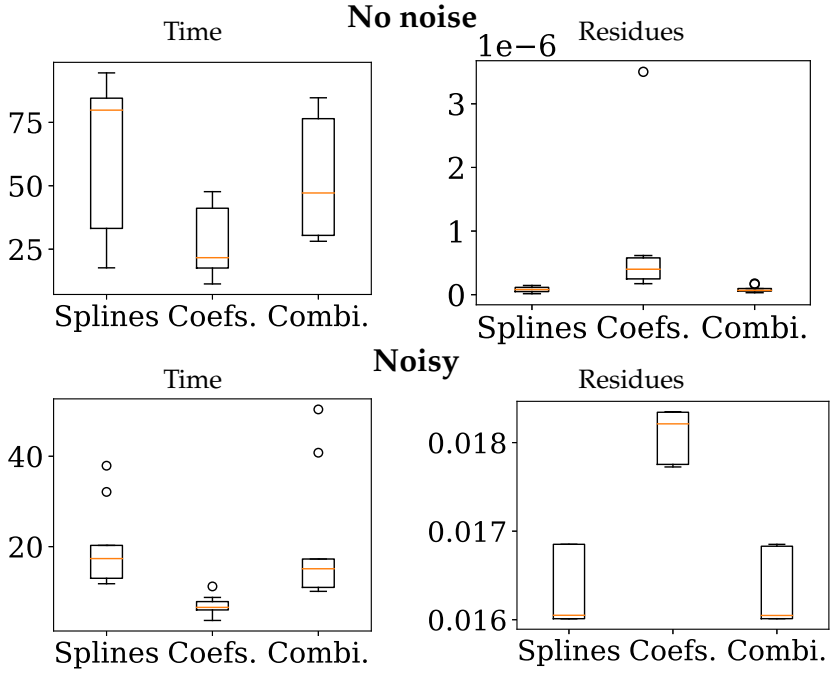


Fig. 5.3 Illustration of the use of splines with nonnegative coefficients as a first step for H-NMF on nonnegative splines (Combi). Tests repeated three times over three initializations (9 tests in total). The upper figures are for a test without noise, while the lower figures are for a test with noise level 20dB.

5.3.3 Updates close in norm to optimal updates

Let us consider the following inexact update with error ϵ, decreasing condition $\delta > 0$, and where x^{t+1} differs from x^t only along block i_t:

$$x_{i_t}^{t+1} \in \{\xi \mid f_{i_t}(x_{i_t}^t\,; x^t) - f_{i_t}(\xi\,; x^t) \geq \delta\|\xi - x_{i_t}^t\|^2;\ \|\xi - \operatorname*{argmin}_{x\in\mathcal{P}} f_{i_t}(x\,; x^t)\| < \epsilon\} \tag{5.18}$$

Note that this set can contain unfeasible points, and that if $f_{i_t}(x\,; x^t)$ is 2δ-strongly convex, this set is never empty as the true minimizer is always feasible (this can be proved using a similar reasoning as in Corollary 5.1).

Theorem 5.9. *Consider problem (5.2), where every update is in set (5.18). If objective function f is lower bounded, and block functions $f_i(\xi\,; x)$ are strongly convex and continuously differentiable, then every limit point of a sequence of updates $\{x^k\}$, $\bar{x}$ is such that:*

$$f(\bar{x}) \leq f_i(x\,; \bar{x}) + \|D_{f_i(\xi\,;\bar{x})}(\bar{x})\|\,\epsilon \quad \forall x \in \mathcal{P}_i.$$

Furthermore, suppose that $\|x_{i_t}^{t+1} - argmin_{x\in\mathcal{P}} f_{i_t}(x\,; x^t)\| \leq \epsilon^t$. If for all $\epsilon > 0$ there is an iteration k so that $\epsilon^t \leq \epsilon$ for all $t \geq k$ (the norm of the becomes arbitrary close to zero), and if the norm of the gradient is bounded, and sets $\mathcal{P}_i$ are closed and convex, then every limit point of a sequence of such update is a stationary point.

Proof. As f is lower bounded, Lemma 5.4 holds and $\sum_{k=1}^{\infty}\sum_{i=1}^{n}\|x_i^{t+1} - x_i^t\|^2 < \infty$. Suppose that $\bar{x}$ is a limit point of $\{x^k\}$. Then there exists a sequence $\{x^{t_j}\}$ that converges to $\bar{x}$, and so does $\{x^{t_j+1}\}$. Consider such a sequence, where each x^{t_j} is the result of an update on block i.

Suppose x_i^* is the optimal value for $f_i(\xi\,; x^{t_j+1})$. As $f_i(\xi\,; x^{t_j+1})$ is 2δ-strongly convex, at each iteration satisfying (5.18), we have (with $\alpha_i^{t_j+1} = x_i^{t_j+1} - x_i^*$)

$$\begin{aligned} f_i(x_i^{t_j+1}, x^{t_j+1}) &\leq f_i(x_i^*\,; x^{t_j+1}) + \langle D_{f_i(\xi\,;x^{t_j+1})}(x_i^{t_j+1}), \alpha_i^{t_j+1}\rangle - \delta\|\alpha_i^{t_j+1}\|^2 \\ &\leq f_i(x\,; x^{t_j+1}) + \langle D_{f_i(\xi\,;x^{t_j+1})}(x_i^{t_j+1}), \alpha_i^{t_j+1}\rangle \qquad \forall x \in \mathcal{P}_i \\ &\leq f_i(x\,; x^{t_j+1}) + \|D_{f_i(\xi\,;x^{t_j+1})}(x_i^{t_j+1})\|\,\epsilon \quad \forall x \in \mathcal{P}_i, \end{aligned}$$

where the last inequality is obtained using the Cauchy-Schwarz inequality,

and the second hypothesis in (5.18) that ensures that $\|\alpha^{t_j+1}\| \leq \epsilon$. As f is continuously block-differentiable, taking the limit when t_j goes to infinity, we have

$$f(\bar{x}) \leq f_i(x_i\,;\bar{x}) + \|D_{f_i(\xi\,;\bar{x})}(\bar{x})\|\epsilon \qquad \forall x_i \in \mathcal{P}_i\ \forall i. \tag{5.19}$$

This last expression is computable and provides a bound on the distance to a Nash equilibrium (Definition 5.2) in each block-direction.

If $\|x_{i_t}^{t+1} - \operatorname{argmin}_{x\in\mathcal{P}} f_{i_t}(x\,;x^t)\| \leq \epsilon^t$ and for all $\epsilon' > 0$ there is a iteration k so that $\epsilon^t \leq \epsilon'$ for all $t \geq k$, we can do exactly the same reasoning using ϵ' instead of ϵ and $t \geq k$ to obtain again Equation (5.19). As ϵ' can be arbitrarly small and $\|D_{f_i(\xi\,;\bar{x})}(\bar{x})\|$ is bounded, this expression is equivalent to

$$f(\bar{x}) \leq f_i(x_i\,;\bar{x}) \qquad \forall x_i \in \mathcal{P}_i\ \forall i. \tag{5.20}$$

If sets $\mathcal{P}_i$ are closed, $\bar{x}_i \in \mathcal{P}_i\ \forall i$. Furthermore, if sets $\mathcal{P}_i$ are convex, $\bar{x}$ is a stationary point of f by Lemma 5.3. □

For H-NMF using $2r$-block decomposition, a way to ensure $\|D_{f_i(\xi\,;\bar{x})}(\bar{x})\|$ to be bounded is to bound x_i^t, as proved by Lemma 5.9 below. This can be done by restricting domain $\mathcal{P}$ to elements with norm below a chosen threshold. In most situations this can be done without loss of generality, using a similar reasoning as in Lemma 5.7.

Lemma 5.9. *For H-NMF from Definition 4.4 using $2r$-blocs decomposition, the norm of the gradient $\|D_{f_i(\xi\,;x^t)}(x_i^t)\|$ is bounded by the initial cost and $O_i^t = \|x_l^t\|^2$ for a $l \neq i$:*

$$\|D_{f_i(\xi\,;x^t)}(x_i^t)\|^2 \leq 4f(0)O_i^t.$$

Proof. If x_i^t is an element $X_s \in \tilde{\mathcal{X}}$, then

$$\|D_{f_i(\xi\,;x^t)}(x_i^t)\| = \Big\| \sum_{k=1}^{r} 2X_k^t\langle A_k^t, A_i\rangle_{\mathcal{A}} - 2g_i^t \Big\|_{\mathcal{X}}^2 \qquad \text{From Proposition 4.4.}$$

Let $E(a,x) = -2Y_1(a,x) + 2\sum_{k=1}^{r} X_k^t(x)A_k^t(a) \in \mathcal{A}\otimes\mathcal{X}$, then

$$\|D_{f_i(\xi\,;x^t)}(x_i^t)\| = \big\|\langle -E(:,x), A_i^t\rangle_{\mathcal{A}}\big\|_{\mathcal{X}}^2.$$

Suppose that $\{F_x^n\}$ is an orthonormal basis of $\mathcal{X}$. Then, $E(a,:) = \sum_n \langle E(a,:), F_x^n \rangle_{\mathcal{X}} F_x^n$. Let $E_n'(a) = \langle E(a,:), F_x^n \rangle_{\mathcal{X}}$, $E_n' \in \mathcal{A}$, which implies $E(:,x) = \sum_n E_n' F_x^n(x)$, and thus $E = \sum_n E_n' \otimes F_x^n$. Therefore

$$\begin{aligned}\|D_{f_i(\xi\,;x^t)}(x_i^t)\| &= \sum_{n_1,n_2} \langle E_{n_1}', A_i^t \rangle_{\mathcal{A}} \langle E_{n_2}', A_s \rangle_{\mathcal{A}} \langle F_x^{n_1}, F_x^{n_2} \rangle_{\mathcal{X}} = \sum_n (\langle E_n', A_i^t \rangle_{\mathcal{A}})^2 \\ &\leq \sum_n \|E_n'\|_{\mathcal{A}}^2 \|A_s\|_{\mathcal{A}}^2 \qquad \text{by Cauchy Schwarz inequality}\end{aligned}$$

We have $\|E\|_{\mathcal{L}}^2 = \sum_{n_1,n_2} \langle E_{n_1}', E_{n_2}' \rangle_{\mathcal{A}} \langle F_x^{n_1}, F_x^{n_2} \rangle_{\mathcal{X}} = \sum_n \|E_n'\|_{\mathcal{A}}^2$. Moreover, $\|E\|_{\mathcal{L}}^2 = 4f(x^t) - 4\|Y_2\|_{\mathcal{H}}^2$. Therefore $\|D_{f_i(\xi\,;x^t)}(x_i^t)\| \leq 4f(0)O_i^t$.

A similar proof can be performed for elements in $\mathcal{A}$. □

Corollary 5.10. *Consider the H-NMF from Definition 4.4 using 2r-block decomposition. Suppose that $M \geq \|x\|^2 \geq \delta > 0 \; \forall x \in \mathcal{P}_i \; \forall i$, for chosen M, δ (M can be chosen using Lemma 5.7 for example). Moreover, suppose that every update is in set (5.18) with ϵ^t changing for each update such that for all $\epsilon > 0$ there exist k such that $\epsilon^t \leq \epsilon$ for all $t \geq k$. Then every limit point of a sequence of updates is a stationary point.*

Example using B-splines The interior-point method used to solve the conic problem computing the projection on nonnegative splines is iterative. It is thus possible to progressively increase the number of iterations and/or to decrease the tolerance of the method throughout iterations, in order to have an error on the projection that goes to zero. We decided to start with a tolerance in the solver of 10^{-2}. This tolerance is divided by two every ten iterations of H-HALS.

Another possibility is to project on the set of splines that are nonnegative on a set of points (discretization of the problem), and not on the whole interval. If the cardinality of this set of points increases during the iterations, the error of the projection will also go to zero. We decided to start with 100 discretization points. The number of discretization points is multiplied by two every ten iterations of H-HALS. Note that in this case we are not sure that the iterates are admissible. Both methods use the previous iterate if the first condition of set (5.18) is not satisfied.

Figure 5.4 shows that using discretization is a good idea in general, but as this approach may find infeasible points, the found points are feasible only asymptotically. On the other hand, decreasing progressively the tolerance

of the solver does not seem to help much in this case. This is most probably because interior-point method used to solve the conic problem converges in few iterations and increasing or decreasing the tolerance does not influence it much. However, in other cases, considering iterative methods with decreasing tolerance/increasing number of iterations could have more impact, such as for projecting polynomials on the nonnegative set using a nonlinear least squares solver. This will be explored in next chapter.

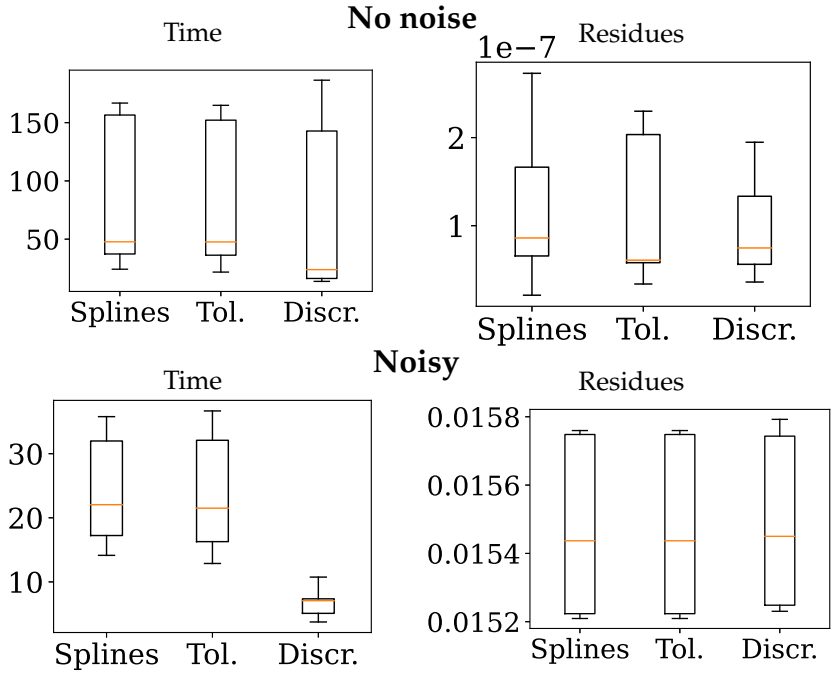


Fig. 5.4 Illustration of the decreasing of the tolerance (Tol.) and the discretization of the problem (Discr.). Tests repeated three times over three initializations (9 tests in total). The upper figures are for a test without noise, while the lower figures are for a test with noise level 20dB.

6

Accelerating NMF using polynomials via inexact projections

In Chapter 3 we have presented an algorithm, LP-HALS, to perform NMF using nonnegative polynomials or splines. We have shown that this algorithm can lead to a more accurate factorization of the input data in terms of reconstruction ability and noise filtering than standard NMF on vectors. Moreover, LP-HALS has the same convergence properties as the standard HALS problem, as shown in Chapter 5. However, this algorithm relies on repeated computation of projections over the set of nonnegative polynomials or splines at each iteration. This projection can dominate the computational effort of the algorithm, as illustrated in Section 6.1, and LP-HALS can be significantly slower than HALS, especially for small and moderate size datasets and for polynomials of high degree. As splines are piecewise polynomials of degree 3, the problem is less present in this case, even though it still exists.

In this chapter, we focus on the case of LP-HALS using polynomials. Our goal is to alleviate the cost of the projection onto nonnegative polynomials, by developing faster projection approaches, possibly approximate. This is

motivated by the results obtained in Chapter 5 where it has been shown that the projections do not need to be exact in order to guarantee convergence of the LP-HALS algorithm. To the best of our knowledge, previous works to improve performance of polynomial optimization have focused mainly on polynomials with high degree, and/or with several variables [60, 96], which does not really correspond to our goal since the polynomials we consider are univariate and of low degree. Some works focus on the shape of positive semidefinite matrices involved in the SDP formulation, like in [1, 51, 143], where the Alternating Direction Method of Multipliers (ADMM) is used to exploit sparsity of the positive semidefinite matrices, but again this is of main interest when the degree of the polynomials is high. On the other hand, a common method to improve performance when solving SDPs is to use the Burer-Monteiro heuristic instead of interior-point methods [16, 21]. Moreover, the set of nonnegative polynomials is the intersection between two sets: the set of nonnegative functions and the set of polynomials. Projection on this intersection can also be solved in principle using alternating projections on these two sets [34, 37]. As projecting on nonnegative functions is not possible, we need to discretize the polynomials to use this idea, and therefore to optimize on another projection problem, close to the original one.

In Section 6.2 we explore this idea as well as several other algorithms based on a discretization of the projection problem. Those algorithms do not have theoretical guarantees on the original (continuous) projection problem, but present the advantage of being very fast. Despite the lack of theoretical guarantees, those heuristic projections accelerate the algorithm very significantly while attaining essentially the same final accuracy. This is presented in the numerical experiments reported in Section 6.2.4.

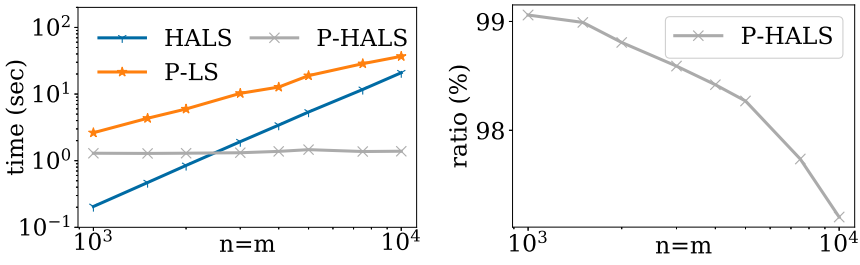
In Section 6.3 we use the theoretical results from Chapter 5 to design projection methods adapted to the NMF problem using polynomials. Those methods are mainly iterative projection methods that are stopped before convergence, once they meet some conditions, using ADMM or Burer-Monteiro, as those approaches have shown their usefulness for solving polynomial optimization or SDPs. Again, the presented methods are much faster than LP-HALS using exact projection, without affecting significantly the obtained accuracy. It is however difficult for them to beat the heuristic methods from Section 6.2, even though some are competitive. Nevertheless, this chapter confirms that LP-HALS using polynomials can be significantly accelerated without affecting the accuracy performance by using

approximate projection instead of exact projection on the set of nonnegative polynomials.

6.1 Computational effort to solve NMF using polynomials

Let P-LS denote the method of [33] that solves the NMF problem using polynomials via an unconstrained parametrization of nonnegative polynomials (this method has been presented in Section 2.2). P-HALS denotes the LP-HALS algorithm using nonnegative polynomials. Figure 6.1(a) displays time needed to compute 20 iterations of both P-LS and P-HALS as well as usual HALS (which does not enforce the polynomial structure), for $r = 3$ polynomials of degree $d = 12$, and for a growing number n of observations and number m of discretization points (with $n = m$).

Standard HALS is faster than polynomial-based algorithms for small-scale problems, but P-HALS and later P-LS become competitive as the size of the problem increases. Moreover, iteration times for P-HALS do not vary much with the size of the problem, as the semidefinite optimization-based projections over nonnegative polynomials are by far the most time-consuming part of the algorithm, as seen in Figure 6.1(b) (more than 97% of the computational time spent). We therefore explore ways to accelerate those polynomial projections in the next sections.



(a) Needed time (in seconds)

(b) Ratio projection/total time

Fig. 6.1 Performances of the P-HALS algorithm over large datasets, using $r = 3$, $d = 12$, $10^3 \leq m = n \leq 10^4$ and a signal to noise ratio of 20dB (see Section 6.2.4 for details about the used datasets).

Reminder: projection on nonnegative polynomials. Properties of nonnegative polynomials have been presented in Section 2.3.1, and the projection on polynomials nonnegatives on a fixed interval has been presented in Section 3.2.1. Let $\mathcal{P}_+^d(I)$ be the set of polynomials of degree d nonnegatives on interval I. As a reminder, the projection of polynomial f of degree d with coefficients $\boldsymbol{f}$ on the set $\mathcal{P}_+^d(I)$ can be expressed as (using matrix $\boldsymbol{L} \in \mathbb{R}^{(d+1)\times(d+1)}$):

$$\underset{\boldsymbol{g} \text{ s. t. } g \in \mathcal{P}_+^d(I)}{\operatorname{argmin}} \quad \|\boldsymbol{L}(\boldsymbol{f}-\boldsymbol{g})\|_2^2 \tag{6.1}$$

This problem has a conic formulation, using $\boldsymbol{R}^d \in \mathbb{R}^{(d+1)\times(d_a^2+d_b^2)}$:

$$\begin{aligned} & \min t \\ \text{such that} \quad & \boldsymbol{u} = \boldsymbol{L}^\top \left(\boldsymbol{R}^d \begin{bmatrix} vec(\boldsymbol{S}_a) \\ vec(\boldsymbol{S}_b) \end{bmatrix} - \boldsymbol{f} \right) \\ & (\boldsymbol{u}, t) \in \mathbb{L}^{d+1},\ \boldsymbol{S}_a \in \mathbb{S}_+^{d_a},\ \boldsymbol{S}_b \in \mathbb{S}_+^{d_b}. \end{aligned} \tag{6.2}$$

The projection is then recovered from optimal values $\boldsymbol{S}_a^*$ and $\boldsymbol{S}_b^*$ using:

$$\boldsymbol{g} = \boldsymbol{R}^d \begin{bmatrix} vec(\boldsymbol{S}_a^*) \\ vec(\boldsymbol{S}_b^*) \end{bmatrix} \tag{6.3}$$

where $\boldsymbol{g}$ contains the coefficients of the projected polynomial. We use Mosek fusion to solve this problem [6].

Interior-point solvers are able to solve this projection problem exactly in an efficient way as the degree d of the polynomials is supposed to be small. However, it is known [69] that interior-point solvers cannot take advantage of prior solutions of nearby instances, i.e. they cannot warm-start. Actually, giving as initialization a point close to the solution may even tend to slow down the algorithms. This is due to the fact that interior-point methods rely on the central path to reach the solution. However, a point close to the solution is unlikely to belong to this central path or even to be close from it. This is a shortcoming in our context where the projection is a subproblem of an iterative algorithm, whose iterates are not expected to be very far from each other, especially when the algorithm starts converging. Therefore, it is reasonable to think that using information from previous projections could have a benefit, but it is not really possible to exploit it when using the interior-point solver.

6.2 Acceleration using heuristics

We introduce below several new heuristic algorithms designed to replace the costly projection over polynomials nonnegative on interval I. They allow one to find a polynomial g that is close to the polynomial to project f. This polynomial g is constrained to be nonnegative on a set of discretization points $\mathcal{D} \subset I$, but is no longer strictly enforced to be nonnegative over the whole interval I. We are thus solving another projection problem, close to the original one.

6.2.1 General heuristics

Discretization heuristic. A first approach consists in solving the projection from (6.1) with constraint $g(x) \geq 0 \; \forall x \in I$ relaxed to $g(x) \geq 0 \; \forall x \in \mathcal{D}$ where set $\mathcal{D} \subset I$ contains a finite number D of points ($\#_D = D$). This relaxed projection problem becomes a second-order cone optimization problem, and is solved faster than the exact version.

Figure 6.2 shows how the chosen number D of discretization points impacts this method: both the computation times and the accuracy (norm of the error with respect to exact projection) are reported, as well as an estimate of the total proportion of I where the projection is negative (the lower, the better). Choosing more discretization points significantly slows down the algorithm, but also significantly decreases the projection error as well as the length of negative subintervals.

Two iterative heuristics The next idea is derived from the fact that polynomial curve fitting, i.e. projection on the set of polynomials, is much faster than the projections (exact or the discretization heuristic) presented previously, as it corresponds to solving a linear system. Consequently, another way to project a given discretized polynomial $f(x)$ is to repeatedly truncate the negative parts of $f(x)$ (projection on the nonnegative set) and apply polynomial curve fitting on the obtained points (projection on the set of polynomials). In other words, this performs alternating projection [16, 21]. This is illustrated in Figure 6.3.

More concretely, we consider a set $\mathcal{D} \subset I$ containing a finite number of points D. We fit a degree-d polynomial to points (λ_i, y_i) $(\forall \lambda_i \in D)$ where

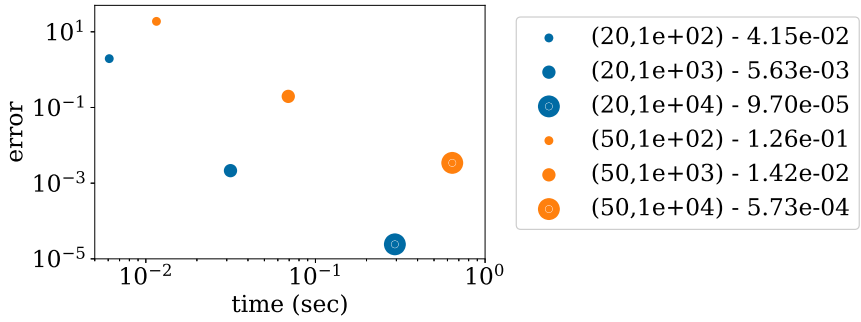


Fig. 6.2 Influence of discretization parameter D on projection using discretization heuristic. The width of symbols increases with D. Legend indicates (d, D) and an estimate of the length of subintervals where projection is negative.

$y_i = f(\lambda_i)$ for every abscissa λ_i such that $f(\lambda_i) > 0$, and $y_i = \epsilon$ otherwise (with ϵ a small positive number). As the fitted polynomial still contains negative values in general, we propose two iterative approaches to ultimately obtain a nonnegative polynomial:

H1: At each iteration, use the obtained approximation as the initial polynomial for the next fit [34].

H2: At each iteration, add ϵ to the values y_i where the obtained approximation is still negative.

Pseudo-code for these two approaches is presented in Algorithm 6.1. To improve the performance of these heuristics, the value of ϵ is doubled at each iteration (but bounded by 0.1, this upper bound has been chosen experimentally). We observed that this doubling of ϵ significantly speeds up H2, while it is less important for H1.

Figure 6.4 shows the influence of the chosen number of discretization points, D, and (initial) parameter ϵ for the first heuristic, H1. We observe that the choice of ϵ does not influence the performance much. On the other hand, higher-degree polynomials require more discretization points to obtain a more accurate result. However, considering too many discretization points does not improve performance, as we can observe in the figure for polynomials of degree 20. Our second heuristic shows very similar performance, except that it is roughly twice as slow, as it typically requires more inner iterations to converge (around 15 instead of 7).

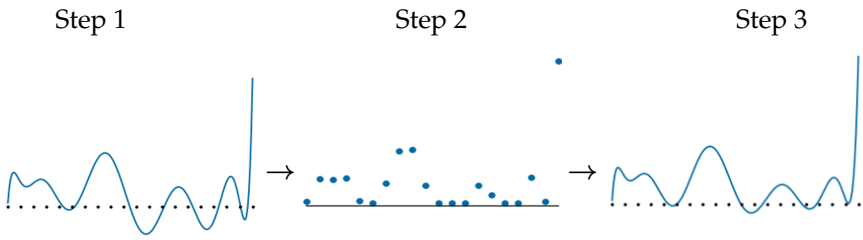


Fig. 6.3 Illustration of the discretization/thresholding (step 2) and the resulting fitting (step 3).

Algorithm 6.1 Iterative heuristics for projection on nonnegative polynomials

Input: $\boldsymbol{f}$ is the polynomial to project and $\boldsymbol{\Pi f}$ evaluates it at points $\mathcal{D}$.

function HEURIPROJ($\boldsymbol{f}$, $\mathcal{D}$, $\boldsymbol{\Pi}$, $\epsilon > 0$, maxiter)
  iter = 0, d = degree of $\boldsymbol{f}$
  vals = $\boldsymbol{\Pi f}$, $\boldsymbol{g} = \boldsymbol{f}$
  while min($\boldsymbol{\Pi g}$)< 0 and iter<maxiter **do**
    if H1 **then**
      vals = $\boldsymbol{\Pi g}$
      vals[vals<0] = ϵ
    else
      neg = $\boldsymbol{\Pi g} < 0$
      vals[neg] = max(vals[neg],0) + ϵ
    $\boldsymbol{g}$ = `PolynomialCurveFitting`($\mathcal{D}$, vals, d)
    iter=iter+1, $\epsilon = \min(0.1, 2\epsilon)$
  return $\boldsymbol{g}$

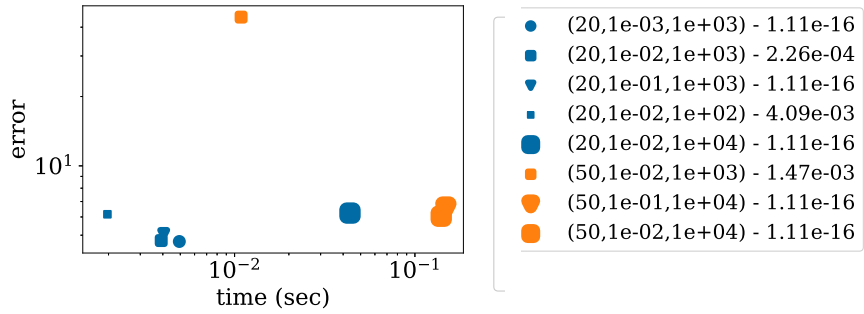


Fig. 6.4 Influence of the parameters on first heuristic. Different symbols correspond to different values of parameter ϵ, while bigger symbols correspond to more discretization points). Legend indicates (d, ϵ, D) - estimated length of negative subintervals.

6.2.2 Proximal projection

This approach aims to take advantage from the fact that iterations of P-HALS tend to become close from each other, and thus that information about nearby projections is known (especially when the algorithm starts to converge). Suppose that the projection of $f_1(x)$ is $g_1(x)$, and that $f_2(x)$ is close to $f_1(x)$, with $\delta(x) = f_1(x) - f_2(x)$ and $\boldsymbol{\delta}$ the coefficient vector of $\delta(x)$, with $\|\boldsymbol{\delta}\|$ small. The problem to solve is:

$$\boldsymbol{g_2} = \text{argmin}_{\boldsymbol{g}} \|\boldsymbol{L}(\boldsymbol{f_1} - \boldsymbol{\delta} - \boldsymbol{g})\|_2^2 \qquad \text{such that } \boldsymbol{g} \in \mathcal{P}_+^d(I)$$

which is equivalent to

$$\boldsymbol{g_2} + \boldsymbol{\delta} = \text{argmin}_{\tilde{\boldsymbol{g}}} \|\boldsymbol{L}(\boldsymbol{f_1} - \tilde{\boldsymbol{g}})\|_2^2 \qquad \text{such that } \tilde{\boldsymbol{g}} - \boldsymbol{\delta} \in \mathcal{P}_+^d(I)$$

This last problem is very similar to the one used to find $\boldsymbol{g_1}$, except that $\tilde{\boldsymbol{g}} - \boldsymbol{\delta}$ must belong to $\mathcal{P}_+^d(I)$ instead of $\tilde{\boldsymbol{g}} \in \mathcal{P}_+^d(I)$. Nevertheless, as $\delta(x)$ is a small perturbation, we propose to estimate $\boldsymbol{g_2}$ as $\boldsymbol{g_2} = \boldsymbol{g_1} - \boldsymbol{\delta}$. Of course, polynomial $g_2(x)$ obtained in such a way can contain negative values. Therefore, we search for the maximal $\gamma \in [0, 1]$ such that $\boldsymbol{g_2} = \boldsymbol{g_1} - \gamma\boldsymbol{\delta}$ is nonnegative over the considered interval. As $\boldsymbol{g_1}$ is nonnegative such a γ always exists and should be as large as possible in order to stay close to the estimated solution $\boldsymbol{g_1} - \boldsymbol{\delta}$. Since $\mathcal{P}_+^d(I)$ is a convex set, we can use a bisection search to find the maximal γ (nonnegativity of $\boldsymbol{g_2}$ is checked on

a set $\mathcal{D}$ of D discretization points defined as for the other heuristics). The pseudo-code of this method is presented in Algorithm 6.2.

Figure 6.5 illustrates performance of the proximal algorithm. In this case, increasing the number of discretization points does not reduce the error too much but decreases the length of negative subintervals at the price of a larger needed time. As expected, the algorithm performs better when the known polynomial is closer to the polynomial to project.

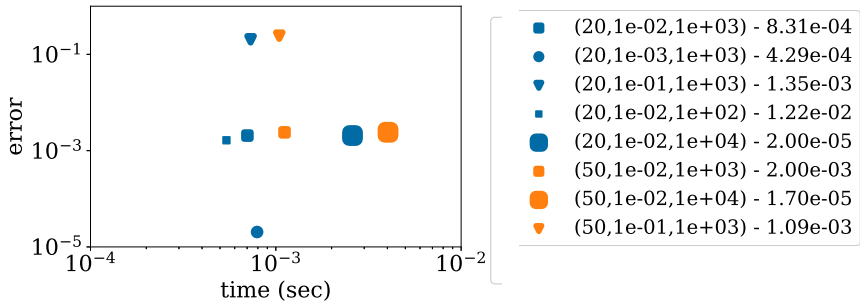


Fig. 6.5 Influence of the parameters on proximal heuristic. Different symbols correspond to different distances between the polynomial to project and the polynomial with known projection. Bigger symbols are for more discretization points. Legend indicates (d, $\|\delta\|$, D) - estimated length of negative subintervals.

6.2.3 Performance comparison for heuristics

Figure 6.6 shows performance of each heuristic in comparison to the exact projection. The polynomial to project is compared to its projection on 100 equispaced points (i.e., matrix $\boldsymbol{L}$ is build using 100 discretization points, see Section 3.2 for more information), while set $\mathcal{D}$ contains 10^3 equispaced points and the length of negative subintervals is estimated trough 10^6 points. Parameter ϵ is chosen as 10^{-2} for heuristics 1 and 2. To evaluate the proximal projection, we use a polynomial with known projection and impose the coefficient vector of this polynomial $\boldsymbol{f_1}$ to be at distance 10^{-2} from $\boldsymbol{f}$, the coefficient vector of the polynomial to project ($\|\boldsymbol{f_1} - \boldsymbol{f}\|_2 = 10^{-2}$).

Figure 6.6(a) shows that the heuristics are significantly faster than the exact projection, especially for large degrees. Proximal projection is the fastest,

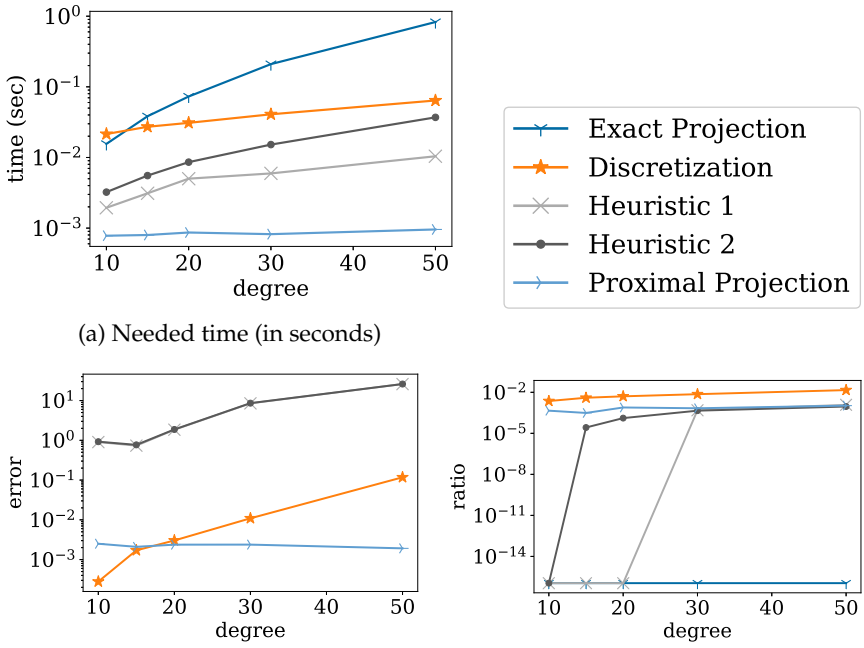
Algorithm 6.2 Projection of close polynomial

Input: $\boldsymbol{f}$ is the polynomial to project and $\boldsymbol{\Pi f}$ evaluates it at points $\mathcal{D}$. $\boldsymbol{f_1}$ is a polynomial close to $\boldsymbol{f}$ and with known projection $\boldsymbol{g_1}$.

function PROXIPROJ($\boldsymbol{f}$, $\boldsymbol{\Pi}$, $\boldsymbol{f_1}$, $\boldsymbol{g_1}$, maxiter=100)
    iter = 0, d = degree of $\boldsymbol{f}$
    $\boldsymbol{\delta} = \boldsymbol{f_1} - \boldsymbol{f}, \boldsymbol{g} = \boldsymbol{g_1} - \boldsymbol{\delta}$
    vals = $\boldsymbol{\Pi g_1}$, valsd = $\boldsymbol{\Pi \delta}$
    $\gamma_{\min} = 0$, $\gamma_{\max} = 1$
    for iter **in** (0,maxiter) **do**:
        $\gamma = (\gamma_{\min}+\gamma_{\max})/2$
        if min(vals - γ valsd)$<$0 **then**:
            $\gamma_{\max} = \gamma$
        else
            $\gamma_{\min} = \gamma$
    return $\boldsymbol{g_1}$-$\gamma_{\min}\boldsymbol{\delta}$

but requires to know the projection of a close polynomial. The two iterative heuristics display similar performance, even though heuristic 2 is slightly slower, and are faster than the discretization heuristic. In Figure 6.6(b), we observe that their approximation error is similar, and much larger than the error of the two other approaches. Figure 6.6(c) illustrates that all methods lead to a solution with only very few negative points (on average less than 2% of the total interval length for all methods). The discretization heuristic performs worst on that aspect.

From this test, it appears that if information about close polynomials is known, using the proximal projection leads to very good results almost instantly. Otherwise, the discretization leads to more accurate results but is slower than the two iterative heuristics, which obtain similar results with fewer negative values but also a larger approximation error than the other approximations. All approximations are faster than the exact projection, which was our main objective. Nevertheless, the ultimate effect of these approximation errors and times has to be evaluated in the context of the whole algorithm for NMF using polynomials.



(a) Needed time (in seconds)

(b) Projection error

(c) Ratio of negative points

Fig. 6.6 Performance of the heuristics presented in this section.

6.2.4 Performance of heuristics for NMF using polynomials

In this section we analyze how the use of our heuristics to compute approximate projections impacts the P-HALS algorithm. In particular, we study whether the approximation errors have any negative effect on the convergence of the algorithm and the accuracy of the solutions, and whether the use of faster projections decreases the total computational time.

In our experiments, we choose to use $D = 10^3$ discretization points for all algorithms, parameter ϵ equal to 10^{-2} for heuristics H1 and H2, and we decide to apply the proximal projection heuristic when the last updated polynomial is closer than 10^{-1}, i.e. the norm of the difference between vectors of coefficients of polynomial to project and last updated polynomial is lower than 10^{-1}. Degree of the sought basis polynomials is $d = 12$. All algorithms are stopped when iterations no longer improve the cost function: $|\text{cost} - \text{previous cost}|/\text{cost} < 10^{-8}$.

We perform our tests over synthetic input signals, created as $\boldsymbol{Y} = \bar{\boldsymbol{A}}\bar{\boldsymbol{X}}^\top + \boldsymbol{N}$ where matrix $\bar{\boldsymbol{A}} \in \mathbb{R}^{m\times r}$ contains some ground truth signals, mixing matrix $\bar{\boldsymbol{X}} \in \mathbb{R}^{n\times r}$ is randomly generated using a normal distribution $\mathcal{N}(0,1)$ with negative values replaced by zero, and $\boldsymbol{N} \in \mathbb{R}^{m\times n}$ is an additive Gaussian noise with a signal-to-noise ratio (SNR) of 20 dB. The use of synthetic signals allows us to compute the error with respect to ground truth, which is $\boldsymbol{Y}^* = \bar{\boldsymbol{A}}\bar{\boldsymbol{X}}$, as explained in Section 2.1.1.

We now proceed to compare the average performance of several algorithms for NMF, including standard HALS (that does not consider polynomials), the P-LS method based on least squares [33] as well as P-HALS with and without approximate projections (using the discretization, H1 or H2 heuristics). The proximal projection heuristic is tested in combination with both standard P-HALS and P-HALS using H1.

Polynomial input signals. We first test with an input dataset containing $r = 3$ polynomials of degree 12 (contained in matrix $\bar{\boldsymbol{A}}$), with $n = m = 500$. We observe in Table 6.1 that HALS is fastest but obtains a final solution with significantly worse accuracy than all other methods, as it ignores the polynomial structure of the input signals. The P-LS method is more accurate than HALS, but much slower than all methods. All methods based on P-HALS compute even more accurate solutions. Among those, methods using approximate projections are, as expected, faster or much faster. More surprising is the fact that, despite their use of inexact projections, they converge to final solutions with accuracy similar to P-HALS with exact projections, and even sometimes a little better and using fewer iterations. Use of the proximal projection heuristic further decreases computational times, at the cost of very slightly larger residues, and thus the accuracy is slightly impacted.

Real reflectance input signals. We also tested our algorithms with $r = 5$ real reflectance signals discretized over $m = 414$ points equally spaced over $[-1,1]$, coming from the U.S. Geological Survey database [77][1], contained in matrix $\bar{\boldsymbol{A}}$.

Figure 6.7 compares performance of the same algorithms, over 100 tests. Again, we see that our heuristics allows P-HALS to converge faster, especially when using heuristics H1 or H2. Using the proximal projection re-

[1] Adulania, Clinochlore, Hypersthene, Olivine and Spessartine from https://www.usgs.gov/labs/spec-lab/capabilities/spectral-library

Method	Time	Iterations	Res
HALS	0.66	224.62	0.01437
P-LS	36.1	448.39	0.01075
P-HALS	10.87	186.84	0.00890
P-HALS+Prox	6.94	150.67	0.00896
P-HALS with Discr.	10.56	178.32	0.00891
P-HALS with H1	2.55	170.73	0.00792
P-HALS with H2	4.94	161.13	0.00793
P-HALS with H1+Prox	2.03	151.44	0.00795

Table 6.1 Average performance on synthetic polynomial signals over 10 tests using 10 initializations (100 tests in total), of HALS, P-LS, P-HALS, and five of our heuristic variants of P-HALS.

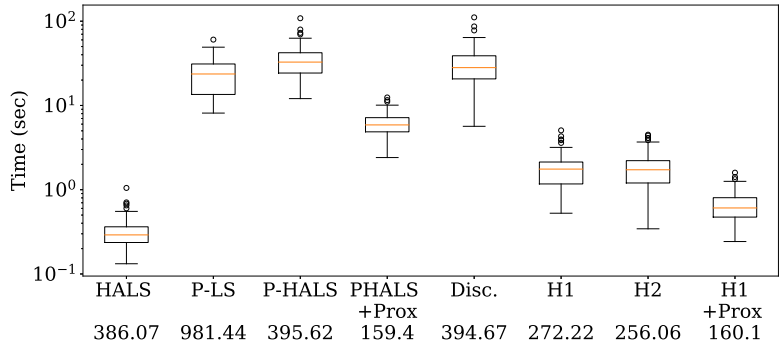
sults in even faster computations using fewer iterations, but with slightly larger final residues. This is coherent with what we observed in the previous section, and is maybe due to a premature stop of the algorithm because the proximal projection becomes too close to the previous iterate.

From the presented tests, we advise to use heuristic H1 with P-HALS as it obtains the best performance and performs better projection than H2, as presented in section 6.2.3. The use of proximal projection in P-HALS leading to slightly less accurate solutions further accelerates the algorithm.

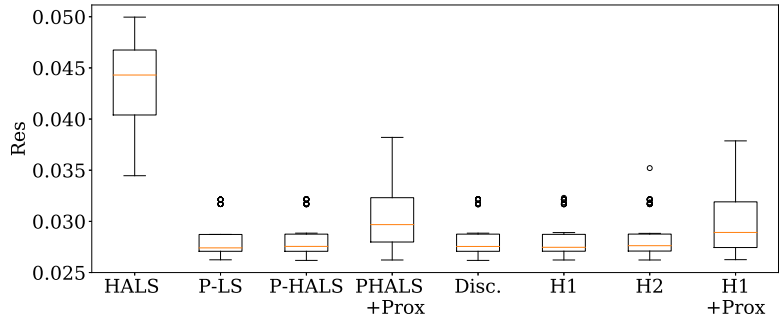
In general, our numerical experiments suggest that performing an exact projection is not necessary for P-HALS to converge, and show that the heuristic projection algorithms introduced in this section significantly accelerate the P-HALS algorithm while preserving the final accuracy of the computed solutions. However, there is no theoretical guarantee of convergence for these heuristic projections, and it is thus possible that in certain situations these heuristics prevent the convergence of the algorithm. Therefore, in the next section, we will use results from Chapter 5.3 to define approximate projections for P-HALS with some theoretical guarantees.

6.3 Acceleration using algorithms with early stopping

In this section, we analyze several ways to perform projections onto the set of nonnegative polynomials of fixed degree using iterative algorithms with early stopping, inspired from results of Section 5.3.3 and from previ-



(a) Distribution of CPU time (in seconds) + mean iterations in legend



(b) Final residual (res)

Fig. 6.7 P-NMF algorithms applied to real reflectance signals.

ous work on optimization problems on nonnegative polynomials. Traditionally, problems on nonnegative polynomials are solved using semidefinite programming (SDP) via interior-point methods [2, 111], as presented in the beginning of this chapter. But LP-HALS is an iterative algorithm that requires to project onto nonnegative polynomials several times, and on similar instances. It would therefore be good if the projection algorithm could be warm-started, but this is not feasible for interior-point methods.

We consider the use of the ADMM or the Burer-Monteiro approach to compute the projection on nonnegative polynomials, as both of these approaches can be warm-started, unlike interior-point methods, and have already been used successfully in the context of polynomials optimization, or SDP [1, 16, 143]. Moreover, those methods are iterative, and from the

results of Section 5.3.3 it is expected that they allow the P-HALS to converge to a stationary point even if they are stopped early, as long as they are asymptotically run until convergence. Of course, in practice it is not possible to run indefinitely the algorithms, but we expect that this good theoretical property makes it possible to have an effective algorithm in practice. We could observe during testing that LP-HALS can indeed be significantly sped up by using ADMM or Burer-Monteiro approaches instead of interior-point methods, without impacting negatively the accuracy.

6.3.1 Projection using ADMM

Using Alternating Direction Method of Multipliers (ADMM) to solve problems on nonnegative polynomials has been considered in the past. Indeed, the matrix $\boldsymbol{R}$ in (6.2) is sparse, especially when the degree of the polynomials increases. Using ADMM allows us to take into account this sparsity and to improve the performance of the algorithm when the degree of polynomials is high [1, 143]. However this does not correspond to our objective as the degree of the considered polynomials is too low for algorithms presented in [1] and [143] to be more efficient than a good interior-point method.

However, we try to take advantage of the knowledge of the solution of problems close to the one to be solved, and this is allowed by ADMM. Let us rewrite problem (6.2) in an appropriate way to use ADMM. Let $\delta_{\mathbb{S}^d_+}(\cdot)$ be the indicator function of $\mathbb{S}^d_+$. The projection problem is

$$\begin{aligned} \min_{\boldsymbol{S}_a, \boldsymbol{S}_b, \boldsymbol{Y}_a, \boldsymbol{Y}_b} & \left\| \boldsymbol{L}^\top \left(\boldsymbol{R}^d \begin{bmatrix} vec(\boldsymbol{S}_a) \\ vec(\boldsymbol{S}_b) \end{bmatrix} - \boldsymbol{f} \right) \right\|^2 + \delta_{\mathbb{S}^{d_a}_+}(\boldsymbol{Y}_a) + \delta_{\mathbb{S}^{d_b}_+}(\boldsymbol{Y}_b) \\ & \text{such that } \begin{bmatrix} \boldsymbol{S}_a \\ \boldsymbol{S}_b \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y}_a \\ \boldsymbol{Y}_b \end{bmatrix}. \end{aligned} \tag{6.4}$$

Considering the two sets of variables $(\boldsymbol{S}_a, \boldsymbol{S}_b)$ and $(\boldsymbol{Y}_a, \boldsymbol{Y}_b)$, the augmented

Lagrangian of (6.4) is

$$\mathcal{L}\big((\boldsymbol{S}_a,\boldsymbol{S}_b),(\boldsymbol{Y}_a,\boldsymbol{Y}_b),(\boldsymbol{\Lambda}_a,\boldsymbol{\Lambda}_b),\rho\big) = \left\|\boldsymbol{L}^\top\left(\boldsymbol{R}^d\begin{bmatrix}vec(\boldsymbol{S}_a)\\vec(\boldsymbol{S}_b)\end{bmatrix}-\boldsymbol{f}\right)\right\|^2 + \delta_{\mathrm{S}_+^{d_a}}(\boldsymbol{Y}_a)$$
$$+\delta_{\mathrm{S}_+^{d_b}}(\boldsymbol{Y}_b) + \langle\boldsymbol{\Lambda}_a,\boldsymbol{Y}_a-\boldsymbol{S}_a\rangle + \langle\boldsymbol{\Lambda}_b,\boldsymbol{Y}_b-\boldsymbol{S}_b\rangle + \frac{\rho}{2}\|\boldsymbol{Y}_a-\boldsymbol{S}_a\|^2 + \frac{\rho}{2}\|\boldsymbol{Y}_b-\boldsymbol{S}_b\|^2 \tag{6.5}$$

To use ADMM, we need to identify the minimum of $\mathcal{L}$ with respect to $(\boldsymbol{S}_a,\boldsymbol{S}_b)$ and with respect to $(\boldsymbol{Y}_a,\boldsymbol{Y}_b)$, the other variables being considered as fixed.

Minimization of the Lagrangian with respect to $(\boldsymbol{S}_a,\boldsymbol{S}_b)$

In (6.5), $\boldsymbol{S}_a$ and $\boldsymbol{S}_b$ are used in their matrix and vectorized form, which complicates the computation of the minimum of $\mathcal{L}$ according to $(\boldsymbol{S}_a,\boldsymbol{S}_b)$. We can choose to work on $\big(vec(\boldsymbol{S}_a),vec(\boldsymbol{S}_b)\big)$ as $\langle\boldsymbol{A},\boldsymbol{B}\rangle = vec(\boldsymbol{A})^\top vec(\boldsymbol{B})$. Nevertheless, we can also choose to impose the symmetry of $\boldsymbol{S}_a$ and $\boldsymbol{S}_b$ as positive semidefinite matrices are symmetric. This will also simplify the update of $(\boldsymbol{Y}_a,\boldsymbol{Y}_b)$, as explained in Section 6.3.1. Therefore, we can work on the following representation of the matrices, which takes into account their symmetry:

$$svec(\boldsymbol{S}_a) = \left[\frac{\boldsymbol{S}_{a_{(1,1)}}}{\sqrt{2}},\boldsymbol{S}_{a_{(1,2)}},\boldsymbol{S}_{a_{(1,3)}},\dots,\frac{\boldsymbol{S}_{a_{(2,2)}}}{\sqrt{2}},\boldsymbol{S}_{a_{(2,3)}},\boldsymbol{S}_{a_{(2,4)}},\dots,\frac{\boldsymbol{S}_{a_{(d_a,d_a)}}}{\sqrt{2}}\right] \tag{6.6}$$

This representation is not exactly the same as the commonly used svec presented for example in [117], but has similar properties. Indeed, with this representation one can easily compute the inner product:

$$\langle\boldsymbol{A},\boldsymbol{B}\rangle = 2\,svec(\boldsymbol{A})^\top svec(\boldsymbol{B}). \tag{6.7}$$

Moreover, it is quite straightforward from equation (6.6) to manipulate the columns of $\boldsymbol{R}$ to find $\tilde{\boldsymbol{R}}$ so that

$$\boldsymbol{R}\begin{bmatrix}vec(\boldsymbol{S}_a)\\vec(\boldsymbol{S}_b)\end{bmatrix} = \tilde{\boldsymbol{R}}\begin{bmatrix}svec(\boldsymbol{S}_a)\\svec(\boldsymbol{S}_b)\end{bmatrix}. \tag{6.8}$$

To optimize $\mathcal{L}$ on $(\boldsymbol{S}_a,\boldsymbol{S}_b)$ we compute the gradient of $\mathcal{L}$ using the svec representation (note that using the svec representation slightly modifies the problem, as it imposes that $\boldsymbol{S}_a$ and $\boldsymbol{S}_b$ are symmetric matrices, but this

is not an issue).

$$\frac{\partial \mathcal{L}}{\partial \begin{bmatrix} svec(\boldsymbol{S}_a) \\ svec(\boldsymbol{S}_b) \end{bmatrix}} = 2\tilde{\boldsymbol{R}}^\top \boldsymbol{L}\boldsymbol{L}^\top \tilde{\boldsymbol{R}} \begin{bmatrix} svec(\boldsymbol{S}_a) \\ svec(\boldsymbol{S}_b) \end{bmatrix} - 2\tilde{\boldsymbol{R}}^\top \boldsymbol{L}\boldsymbol{L}^\top \boldsymbol{f} - 2 \begin{bmatrix} svec(\boldsymbol{\Lambda}_a) \\ svec(\boldsymbol{\Lambda}_b) \end{bmatrix} - 2\rho \begin{bmatrix} svec(\boldsymbol{Y}_a - \boldsymbol{S}_a) \\ svec(\boldsymbol{Y}_b - \boldsymbol{S}_b) \end{bmatrix} \tag{6.9}$$

As the problem is unconstrained and convex, the minimization of $\mathcal{L}$ on $(\boldsymbol{S}_a, \boldsymbol{S}_b)$ can therefore be obtained by setting the gradient to zero:

$$\begin{bmatrix} svec(\boldsymbol{S}_a) \\ svec(\boldsymbol{S}_b) \end{bmatrix} = (\tilde{\boldsymbol{R}}^\top \boldsymbol{L}\boldsymbol{L}^\top \tilde{\boldsymbol{R}} + \rho \boldsymbol{I})^{-1} \Big(\tilde{\boldsymbol{R}}^\top \boldsymbol{L}\boldsymbol{L}^\top \boldsymbol{f} + \begin{bmatrix} svec(\boldsymbol{\Lambda}_a + \rho \boldsymbol{Y}_a) \\ svec(\boldsymbol{\Lambda}_b + \rho \boldsymbol{Y}_b) \end{bmatrix} \Big). \tag{6.10}$$

Minimization of the Lagrangian with respect to (Y_a, Y_b)

The problem we aim to solve can be written as

$$\underset{(\boldsymbol{Y}_a, \boldsymbol{Y}_b)}{\operatorname{argmin}} \|\boldsymbol{Y}_a - \boldsymbol{S}_a + \boldsymbol{\Lambda}_a/\rho\|^2 + \|\boldsymbol{Y}_b - \boldsymbol{S}_b + \boldsymbol{\Lambda}_b/\rho\|^2 \quad \text{s.t.} \quad \boldsymbol{Y}_a \in \mathbb{S}_+^{d_a}, \boldsymbol{Y}_b \in \mathbb{S}_+^{d_b}. \tag{6.11}$$

We can easily see that this problem consists of two projections, the projection of $\boldsymbol{S}_a - \boldsymbol{\Lambda}_a/\rho$ onto $\mathbb{S}_+^{d_a}$ and the projection of $\boldsymbol{S}_b - \boldsymbol{\Lambda}_b/\rho$ onto $\mathbb{S}_+^{d_b}$. Let $[\cdot]_{\mathcal{S}}$ represent the projection onto set $\mathcal{S}$. The solution of (6.11) is therefore

$$(\boldsymbol{Y}_a, \boldsymbol{Y}_b) = \Big([\boldsymbol{S}_a - \boldsymbol{\Lambda}_a/\rho]_{\mathbb{S}_+^{d_a}}, \, [\boldsymbol{S}_b - \boldsymbol{\Lambda}_b/\rho]_{\mathbb{S}_+^{d_b}} \Big) \tag{6.12}$$

As $\boldsymbol{\Lambda}_a$ and $\boldsymbol{\Lambda}_b$ are obtained by linear combinations of $\boldsymbol{S}_a, \boldsymbol{Y}_a$ and $\boldsymbol{S}_b, \boldsymbol{Y}_b$ (see Algorithm 6.3), if $\boldsymbol{S}_a$ and $\boldsymbol{S}_b$ are symmetric matrices (which is the case if they are obtained using equation (6.10)), matrices $\boldsymbol{S}_a - \boldsymbol{\Lambda}_a/\rho$ and $\boldsymbol{S}_b - \boldsymbol{\Lambda}_b/\rho$ are symmetric. If $\boldsymbol{Q}_A \boldsymbol{E}_A \boldsymbol{Q}_A^\top$ is the eigenvalue decomposition of $\boldsymbol{S}_a - \boldsymbol{\Lambda}_a/\rho$ and $\boldsymbol{Q}_B \boldsymbol{E}_B \boldsymbol{Q}_B^\top$ is the eigenvalue decomposition of $\boldsymbol{S}_b - \boldsymbol{\Lambda}_b/\rho$, and $[\cdot]_+$ is an operator that sets all negative values to 0, the projections are

$$(\boldsymbol{Y}_a, \boldsymbol{Y}_b) = \big(\boldsymbol{Q}_A [\boldsymbol{E}_A]_+ \boldsymbol{Q}_A^\top, \, \boldsymbol{Q}_B [\boldsymbol{E}_B]_+ \boldsymbol{Q}_B^\top \big). \tag{6.13}$$

ADMM algorithm

Thanks to the developments carried out in the two previous sections, we can now write a sketch of the ADMM algorithm to project a polynomial

of degree d on the nonnegative set $\mathcal{P}_+^d$. Note that the algorithm returns $(\boldsymbol{Y}_a, \boldsymbol{Y}_b)$ as the computed projection, because this ensures to have a feasible point, but if the algorithm has converged, it should be equal to $(\boldsymbol{S}_a, \boldsymbol{S}_b)$. Moreover, if ρ is fixed, the iterates of the algorithm converge to a solution of (6.4). Indeed, the considered functions are both convex, and we are working with 2-block ADMM (see Theorem 8 of [38]).

Algorithm 6.3 Projection on nonnegative polynomials using ADMM

Input: $\boldsymbol{f}$ the coefficients of the polynomial to project, $\boldsymbol{Y}_a^0 \in \mathbb{S}_+^{d_a}$, $\boldsymbol{Y}_b^0 \in \mathbb{S}_+^{d_b}$

function PROJ_ADMM($\boldsymbol{f}$, $\boldsymbol{Y}_a^0$,$\boldsymbol{Y}_b^0$)

l=0, $\rho^0 = 1$, $\boldsymbol{\Lambda}_a = 0^{(d_a)\times(d_a)}$, $\boldsymbol{\Lambda}_b = 0^{d_b \times d_b}$

while not `EndCond` **do**

$$\begin{bmatrix} svec(\boldsymbol{S}_a)^{l+1} \\ svec(\boldsymbol{S}_b)^{l+1} \end{bmatrix} = \left(\tilde{\boldsymbol{R}}^\top \boldsymbol{L}\boldsymbol{L}^\top \tilde{\boldsymbol{R}} + \rho^l \boldsymbol{I}\right)^{-1} \left(\tilde{\boldsymbol{R}}^\top \boldsymbol{L}\boldsymbol{L}^\top \boldsymbol{f} + \begin{bmatrix} svec(\boldsymbol{\Lambda}_a^l + \rho^l \boldsymbol{Y}_a^l) \\ svec(\boldsymbol{\Lambda}_b^l + \rho^l \boldsymbol{Y}_b^l) \end{bmatrix}\right)$$

$$(\boldsymbol{Y}_a^{l+1}, \boldsymbol{Y}_b^{l+1}) = \left([\boldsymbol{S}_a^{l+1} - \boldsymbol{\Lambda}_a^l/\rho^l]_{\mathbb{S}_+^{d_a}}, [\boldsymbol{S}_b^{l+1} - \boldsymbol{\Lambda}_b^l/\rho^l]_{\mathbb{S}_+^{d_b}}\right)$$

$$(\boldsymbol{\Lambda}_a^{l+1}, \boldsymbol{\Lambda}_b^{l+1}) = (\boldsymbol{\Lambda}_a^l + \rho^l(\boldsymbol{Y}_a^{l+1} - \boldsymbol{S}_a^{l+1}), \boldsymbol{\Lambda}_b^l + \rho^l(\boldsymbol{Y}_b^{l+1} - \boldsymbol{S}_b^{l+1}))$$

ρ^{l+1} = `UpdateRho`; $l = l + 1$

Return $\tilde{\boldsymbol{R}} \begin{bmatrix} svec(\boldsymbol{Y}_a^{\text{end}}) \\ svec(\boldsymbol{Y}_b^{\text{end}}) \end{bmatrix}$

As the projection is used inside an iterative algorithm, we use $(\boldsymbol{Y}_a^0, \boldsymbol{Y}_b^0) = (\boldsymbol{Y}_a^{\text{prev}}, \boldsymbol{Y}_b^{\text{prev}})$, the result of projection performed at the previous iteration. If we are at the first iteration, $(\boldsymbol{Y}_a^0, \boldsymbol{Y}_b^0)$ are initialized as random rank-one positive semidefinite matrices. Finding a good `EndCond` and `UpdateRho` is not trivial. As this algorithm is part of an outer iterative algorithm, an idea inspired from Section 5.3.3 is to aim for an approximate projection during the first iterations and to project exactly only when the algorithm starts to converge. We can hope that such a behavior is achieved when we fix the number of iterations in Algorithm 6.3 to a small number, and keep ρ unchanged. Indeed, as we use the result of previous projection as a first guess, the hope is that when the algorithm starts to converge the projections are more and more precise.

Otherwise, we can also use Boyd's method from [17]. Let us assume that we have fixed the parameters tol, maxRho, minRho, and let $e^{l+1} =$

$$\frac{\sqrt{\left\|\boldsymbol{Y}_a^{l+1}-\boldsymbol{S}_a^{l+1}\right\|^2+\left\|\boldsymbol{Y}_b^{l+1}-\boldsymbol{S}_b^{l+1}\right\|^2}}{\max\left(\sqrt{\left\|\boldsymbol{S}_a^{l+1}\right\|^2+\left\|\boldsymbol{S}_b^{l+1}\right\|^2},\sqrt{\left\|\boldsymbol{Y}_a^{l+1}\right\|^2+\left\|\boldsymbol{Y}_b^{l+1}\right\|^2}\right)} \text{ and } e_D^{l+1}=\frac{\rho^l\sqrt{\left\|\boldsymbol{Y}_a^{l+1}-\boldsymbol{Y}_a^{l}\right\|^2+\left\|\boldsymbol{Y}_b^{l+1}-\boldsymbol{Y}_b^{l}\right\|^2}}{\sqrt{\left\|\boldsymbol{\Lambda}_a^{l+1}\right\|^2+\left\|\boldsymbol{\Lambda}_b^{l+1}\right\|^2}}.$$

We have

$$\texttt{EndCond} = \min(e^{l+1},e_D^{l+1})<\text{tol}, \texttt{ UpdateRho} = \begin{cases} \min(2\rho^l,\text{maxRho}) & \text{if } e^{l+1} > 10e_D^{l+1} \\ \max(\rho^l/2,\text{minRho}) & \text{if } e_D^{l+1} > 10e^{l+1} \\ \rho^l & \text{else.} \end{cases} \tag{6.14}$$

Using stationarity information

The ϵ-stationarity measure is an approximate stationarity measure defined in Definition 5.3. From Theorem 5.5 we know the following.

Theorem 6.1. *Let $\boldsymbol{B}$ be the coefficients of polynomials contained in $\boldsymbol{A}$. Then $\boldsymbol{B}_{:k}^* = \frac{(\boldsymbol{M})^{-1}\boldsymbol{Z}\boldsymbol{X}_{:k}-\sum_{s\neq k}\boldsymbol{B}_{:s}\boldsymbol{X}_{:s}{}^\top\boldsymbol{X}_{:k}}{\boldsymbol{X}_{:k}{}^\top\boldsymbol{X}_{:k}}$ is the unconstrained update of $\boldsymbol{B}$ (see Equation (3.6)). Suppose that $\boldsymbol{X}$ is such that $\boldsymbol{X}_{kj} \geq \delta$, and $\boldsymbol{X}$ is updated using the standard HALS algorithm. Every limit point of P-HALS is $r\epsilon$-stationary if the updates of $\boldsymbol{B}_{:k}$ satisfy*

1. $\boldsymbol{B}_{:k}{}^{t+1} \in \mathcal{P}_+^d(I)$
2. $2\langle \boldsymbol{L}\boldsymbol{L}^\top(\boldsymbol{B}_{:i}{}^{t+1} - \boldsymbol{B}_{:k}^{*\,t+1}), \boldsymbol{x} - \boldsymbol{B}_{:k}{}^{t+1}\rangle \geq -\epsilon \;\; \forall \boldsymbol{x} \in \mathcal{P}_+^d(I)$
3. $2\langle \boldsymbol{L}\boldsymbol{L}^\top(\boldsymbol{B}_{:k}{}^{t+1} - \boldsymbol{B}_{:k}^{*\,t+1}), \boldsymbol{B}_{:k}{}^{t} - \boldsymbol{B}_{:k}{}^{t+1}\rangle \geq 0$
4. $\|\boldsymbol{B}_{:k}{}^{t+1}\| \geq \delta$ (6.15)

Proof. The two first conditions are the same as in Theorem 5.5.

By the fourth condition, the NMF problem on a column of $\boldsymbol{B}$ or $\boldsymbol{X}$ considering all other variables as fixed is a strongly convex problem with $\mu \geq \delta$. Using the third condition and a similar reasoning as in Corollary 5.1, the third condition of Theorem 5.5 holds. □

Using Algorithm 6.3, condition 1 is always satisfied. Moreover, condition 4 is a common problem concerning the convergence of HALS. Nevertheless, if the chosen rank is not too high, it is rare that elements become zero, especially when the algorithm starts to converge. We can therefore reasonably neglect this constraint. However, conditions 2 and 3 are not negligible and not always met. Let us analyze them more deeply.

Lemma 6.1. *At convergence, conditions 2 and 3 of equation (6.15) are satisfied, with $\epsilon = 0$.*

Proof. As the ADMM algorithm converges to a solution of (6.4) [38], it converges to a $\epsilon = 0$ - stationary point and conditions 2 and 3 are satisfied at convergence. □

Lemma 6.2. *Let* $\boldsymbol{G}_1^{end} \in \mathbb{R}^{\frac{d_a(d_a+1)}{2}}$, $\boldsymbol{G}_2^{end} \in \mathbb{R}^{\frac{d_b(d_b+1)}{2}}$, $\begin{bmatrix}\boldsymbol{G}_1^{end}\\ \boldsymbol{G}_2^{end}\end{bmatrix} =$

$$2(\tilde{\boldsymbol{R}}^\top \boldsymbol{L}\boldsymbol{L}^\top \tilde{\boldsymbol{R}})\begin{bmatrix}svec(\boldsymbol{Y}_a^{end} - \boldsymbol{S}_a^{end})\\ svec(\boldsymbol{Y}_b^{end} - \boldsymbol{S}_b^{end})\end{bmatrix} + \rho^{end-1}\begin{bmatrix}svec(\boldsymbol{Y}_a^{end-1} - \boldsymbol{Y}_a^{end})\\ svec(\boldsymbol{Y}_b^{end-1} - \boldsymbol{Y}_b^{end})\end{bmatrix} + \begin{bmatrix}svec(\boldsymbol{\Lambda}_a^{end})\\ svec(\boldsymbol{\Lambda}_b^{end})\end{bmatrix}.$$

Condition 2 is equivalent to $svec^{-1}(\boldsymbol{G}_1^{end}) \in \mathbb{S}_+^{d_a}$ *and* $svec^{-1}(\boldsymbol{G}_2^{end}) \in \mathbb{S}_+^{d_b}$ *and* $2\left\langle \begin{bmatrix}\boldsymbol{G}_1^{end}\\ \boldsymbol{G}_2^{end}\end{bmatrix}, \begin{bmatrix}svec(\boldsymbol{Y}_a^{end})\\ svec(\boldsymbol{Y}_b^{end})\end{bmatrix} \right\rangle \leq \epsilon.$

Proof. As $x \in \mathcal{P}_+^d(I)$, it can be described as $\boldsymbol{x} = \tilde{\boldsymbol{R}}\begin{bmatrix}svec(\boldsymbol{X}_a)\\ svec(\boldsymbol{X}_b)\end{bmatrix}$ with $\boldsymbol{X}_a \in \mathbb{S}_+^{d_a}$ and $\boldsymbol{X}_b \in \mathbb{S}_+^{d_b}$, by Equations (6.3) and (6.8). Moreover, every $\boldsymbol{X}_a \in \mathbb{S}_+^{d_a}$ and $\boldsymbol{X}_b \in \mathbb{S}_+^{d_b}$ define a feasible $\boldsymbol{x}$. Considering that $\boldsymbol{B}_{:k}{}^{t+1}$ is found using Algorithm 6.3 on $\boldsymbol{f} = \boldsymbol{B}_{:k}^{*\,t+1}$, we can rewrite condition 2 as:

$$2\left\langle \boldsymbol{L}\boldsymbol{L}^\top\left(\tilde{\boldsymbol{R}}\begin{bmatrix}svec(\boldsymbol{Y}_a^{end})\\ svec(\boldsymbol{Y}_b^{end})\end{bmatrix} - \boldsymbol{B}_{:k}^{*\,t+1}\right), \tilde{\boldsymbol{R}}\begin{bmatrix}svec(\boldsymbol{X}_a - \boldsymbol{Y}_a^{end})\\ svec(\boldsymbol{X}_b - \boldsymbol{Y}_b^{end})\end{bmatrix} \right\rangle \geq -\epsilon. \quad (6.16)$$

From Algorithm 6.3, we can observe that $\tilde{\boldsymbol{R}}^\top \boldsymbol{L}\boldsymbol{L}^\top \boldsymbol{B}_{:k}^{*\,t+1} = (\tilde{\boldsymbol{R}}^\top \boldsymbol{L}\boldsymbol{L}^\top \tilde{\boldsymbol{R}} + \rho^l \boldsymbol{I})\begin{bmatrix}svec(\boldsymbol{S}_a^{l+1})\\ svec(\boldsymbol{S}_b^{l+1})\end{bmatrix} - \begin{bmatrix}svec(\boldsymbol{\Lambda}_a^l + \rho^l \boldsymbol{Y}_a^l)\\ svec(\boldsymbol{\Lambda}_b^l + \rho^l \boldsymbol{Y}_b^l)\end{bmatrix}$ $\forall l$. Using the definition of $\boldsymbol{\Lambda}_a^{l+1}$ and $\boldsymbol{\Lambda}_b^{l+1}$ and the definition of $\boldsymbol{G}_1^{end}, \boldsymbol{G}_2^{end}$ presented in the statement of this lemma, we observe that condition 2 is equivalent to

$$2\left\langle \begin{bmatrix}\boldsymbol{G}_1^{end}\\ \boldsymbol{G}_2^{end}\end{bmatrix}, \begin{bmatrix}svec(\boldsymbol{X}_a - \boldsymbol{Y}_a^{end})\\ svec(\boldsymbol{X}_b - \boldsymbol{Y}_b^{end})\end{bmatrix} \right\rangle \geq -\epsilon. \quad (6.17)$$

Let us note that if there exist $(\boldsymbol{X}_a, \boldsymbol{X}_b)$ such that $\left\langle \begin{bmatrix}\boldsymbol{G}_1^{end}\\ \boldsymbol{G}_2^{end}\end{bmatrix}, \begin{bmatrix}svec(\boldsymbol{X}_a)\\ svec(\boldsymbol{X}_b)\end{bmatrix} \right\rangle < 0$, condition 2 cannot be satisfied. Indeed, $(\alpha\boldsymbol{X}_a, \alpha\boldsymbol{X}_b)$ is also feasible for

all $\alpha \geq 0$, therefore when α goes to infinity, condition 2 goes to $-\infty$. To avoid that, as the positive semidefinite cone is self dual, we must have $svec^{-1}(\boldsymbol{G}_1^{end}) \in \mathbb{S}_+^{d_a}$ and $svec^{-1}(\boldsymbol{G}_2^{end}) \in \mathbb{S}_+^{d_b}$ to satisfy condition 2. The value of ϵ is then $2\left\langle \begin{bmatrix} \boldsymbol{G}_1^{end} \\ \boldsymbol{G}_2^{end} \end{bmatrix}, \begin{bmatrix} svec(\boldsymbol{Y}_a^{end}) \\ svec(\boldsymbol{Y}_b^{end}) \end{bmatrix} \right\rangle \geq 0$.

□

Thanks to Lemma 6.2, condition 2 is computable. Therefore, Theorem 6.1 provides a stopping criterion: as soon as conditions 2 and 3 are satisfied, the projection algorithm can stop while still ensuring convergence of the outer algorithm to a $r\epsilon-$stationary point.

6.3.2 Projection using Burer-Monteiro approach

From Equation (2.13), we know that all nonnegative polynomials can be expressed using rank-one matrices $\boldsymbol{S}_a$ and $\boldsymbol{S}_b$ in problem (6.2). Let $\otimes_K$ be the Kronecker product, we can rewrite problem (6.2) as

$$\min_{\boldsymbol{s}_a \in \mathbb{R}^{d_a}, \boldsymbol{s}_b \in \mathbb{R}^{d_b}} \left\| \boldsymbol{L}^\top \left(\boldsymbol{R}^d \begin{bmatrix} \boldsymbol{s}_a \otimes_K \boldsymbol{s}_a^\top \\ \boldsymbol{s}_b \otimes_K \boldsymbol{s}_b^\top \end{bmatrix} - \boldsymbol{f} \right) \right\|, \tag{6.18}$$

The Jacobian of the residual in (6.18) can be easily computed and is equal to:

$$J_{\boldsymbol{s}_a, \boldsymbol{s}_b} = \boldsymbol{L}^\top \boldsymbol{R}^d \begin{bmatrix} \boldsymbol{I}^{d_a} \otimes_K \boldsymbol{s}_a + \boldsymbol{s}_a \otimes_K \boldsymbol{I}^{d_a} & 0^{d_a^2 \times d_b} \\ 0^{d_b^2 \times d_a} & \boldsymbol{I}^{d_b} \otimes_K \boldsymbol{s}_b + \boldsymbol{s}_b \otimes_K \boldsymbol{I}^{d_b} \end{bmatrix} \tag{6.19}$$

The problem can then be solved using a nonlinear least squares solver. We use the `least_squares` solver of python[2] with default parameters. At each projection, the previous solution is used as first guess, as for ADMM. This approach is comparable to the Burer-Monteiro approach of rank 1 for solving the projection problem in SDP form (6.3). Indeed, in the Burer-Monteiro approach the positive semidefinite matrices are constrained to be low rank and factorized using low-rank matrices: $\boldsymbol{S} = \boldsymbol{F}\boldsymbol{F}^\top$ where $\boldsymbol{F} \in \mathbb{R}^{s \times s}$ with s much smaller than d. In our case $s = 1$. We also consider the use of other ranks in this Burer-Monteiro approach, i.e. use

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html#id2

$\sum_{k=1}^{s} \boldsymbol{s}_{ak} \otimes_K \boldsymbol{s}_{ak}^\top$ and $\sum_{k=1}^{s} \boldsymbol{s}_{bk} \otimes \boldsymbol{s}_{bk}^\top$ instead of $\boldsymbol{S}_a$ and $\boldsymbol{S}_b$. The Jacobian is close to expressions (6.19) for each $\boldsymbol{s}_{ak}$, $\boldsymbol{s}_{bk}$.

6.3.3 Parameters of proposed projection methods

We now analyze the performance of the proposed projection methods with various parameters.

Datasets and comparison tools

To build input data $\boldsymbol{Y}$, we first build two matrices $\boldsymbol{A}$ and $\boldsymbol{X}$. $\boldsymbol{A}$ can have two different forms:

- **Pol:** $\boldsymbol{A}$ contains the discretization of r polynomials of degree d non-negative on $[-1, 1]$. The coefficients of the r polynomials are built using random positive-semidefinite matrices $\boldsymbol{S}_a^*$ and $\boldsymbol{S}_b^*$ of rank 1, using representation of Equation (6.3).
- **Refl:** $\boldsymbol{A}$ contains discretization of reflectance signals. The reflectance signals are the signals of Adulania, Clinochlore, Hypersthene, Olivine, Spessartine, Andesine, Celestine and Kaolinite from [77], r is therefore equal to 8. Those signals are discretized on 414 points and are illustrated in Figure 6.8.

Factor $\boldsymbol{X} \in \mathbb{R}_+^{n \times r}$ is a random matrix whose rows follow a Dirichlet distribution with parameter $\alpha = 1/r$. We then create input data $\boldsymbol{Y}$ as $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}^\top + \boldsymbol{N}$, where $\boldsymbol{N}$ is an additive Gaussian noise with a chosen signal-to-noise ratio (SNR). The algorithms are compared in term of accuracy using the relative residual, SIR and SIR LC criteria presented in Section 2.1.1.

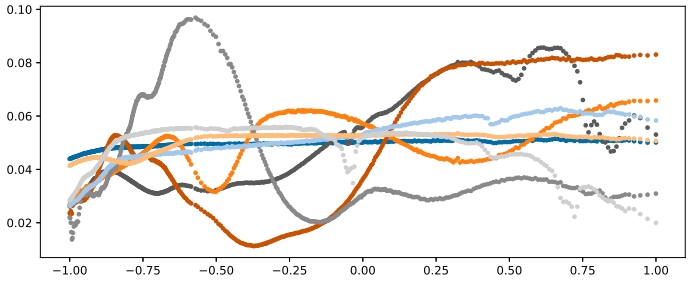


Fig. 6.8 The reflectance signals of Adulania, Clinochlore, Hypersthene, Olivine, Spessartine, Andesine, Celestine and Kaolinite.

To determine the parameters of the ADMM and Burer-Monteiro approaches, we tested various choices on a dataset containing polynomials of degree $d = 20$ (**Pol**). We use $m = 200$, $n = 500$, $r = 5$ and data does not contain noise. All parameters are tested 10 times over different initialization. The problem we aim to solve at this stage is a subproblem of the NMF problem. Indeed, we suppose that matrix $\boldsymbol{X}$ is fixed and known, so that the problem becomes convex: $\min_{\boldsymbol{A}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}_0^\top\|_F$. We consider five cases:

- $\boldsymbol{X}_0$ is random (**Random**).
- $\boldsymbol{X}_0$ is such that $\frac{\|\boldsymbol{Y}-\tilde{\boldsymbol{A}}\boldsymbol{X}_0^\top\|}{\|\boldsymbol{Y}\|} = 0.1$ or 0.05 or 0.01, with $\tilde{\boldsymbol{A}}$ the initialization of $\boldsymbol{A}$ (**0.1, 0.05, 0.01**).
- $\boldsymbol{X}_0$ is the ground truth mixing matrix, $\boldsymbol{Y} = \boldsymbol{A}_0\boldsymbol{X}_0^\top$ (**Exact**).

This lets us observing whether the methods behave differently throughout the outer NMF process, when we are close or not to the solution.

This convex problem is solved by updating iteratively the columns of matrix $\boldsymbol{A}$. Iterations are stopped once either $\|\boldsymbol{A}^t - \boldsymbol{A}^{t-1}\|| < 10^{-12}$, or $\frac{\left|\|\boldsymbol{Y}-\boldsymbol{A}^t\boldsymbol{X}_0^\top\|-\|\boldsymbol{Y}-\boldsymbol{A}^{t-1}\boldsymbol{X}_0^\top\|\right|}{\|\boldsymbol{Y}\|} < 10^{-16}$, or time exceeds 1000 seconds. Figures summarizing the tests display the following error: $\frac{\|\boldsymbol{Y}-\boldsymbol{A}\boldsymbol{X}_0^\top\|-\|\boldsymbol{Y}-\boldsymbol{A}^*\boldsymbol{X}_0^\top\|}{\|\boldsymbol{Y}\|}$ with $\boldsymbol{A}^*$ the optimal solution. Obtaining a zero error is thus possible.

A second test is performed, on the same input data $\boldsymbol{Y}$ but this time on the full NMF problem. This test shows the average evolution of the relative residual of tested approaches. All approaches are run during exactly 100 seconds.

Parameters of ADMM

We considered the following parameters in the ADMM projection from Algorithm 6.3:

- Algorithm performs exactly 10 iterations with $\rho = 1$ fixed (**10 its**)
- Algorithm performs exactly 50 iterations with $\rho = 1$ fixed (**50 its**)
- Algorithm performs maximum 50 iterations, the stopping criterion and the update of ρ follow Boyd's method, presented in equation (6.14) with `maxRho`$= 10^4$, `minRho`$= 10^{-4}$ (**rho_1e-4**)
- Same except that `maxRho`$= 10^8$, `minRho`$= 10^{-8}$ (**rho_1e-8**)

- Algorithm performs maximum 50 iterations,with $\rho = 1$ fixed. The stopping criterion is the one presented in Section 6.3.1 (**statio**).
- Same except that if algorithm was not able to stop in 50 iterations, we use an interior-point algorithm to solve the projection problem (**combi**).

The last approach ensures that the NMF algorithm converges to an almost stationary point by Theorem 6.1. Indeed, either the algorithm has stop because conditions (5.10) are satisfied, or the projection has been done using the interior point algorithm that converges to the optimal solution and satisfies thus conditions (5.10). The hope in this last approach is that the ADMM algorithm converges often in less than 50 iterations and that the interior point solver is not very much used.

Results are summarized in Figure 6.9. On the convex problem, we observe that all methods are very good in the "Random" situation, as they are fast and find the optimal solution. In this case using exactly 10 iterations is the fastest, using the combination is the slowest and other parameters are comparable in terms of time.

Moreover, the combination is not much influenced by the situation: times are comparable and recovered errors are always very good, most of the time around 10^{-12} or below, except in the "Exact" situation where the worst case is around 10^{-7} . This method is the most accurate and among the fastest (except in "Random" situation). It is therefore very suited for the convex case.

On the contrary, using 10 iterations is not suited for the convex problem as it obtains the worse error on average and is among the slowest methods, except in "Random" case where it is the fastest. Approaches using maximum 50 iterations have very similar behaviors. Therefore, using a stopping criterion does not seem to help. The time gained in performing fewer iterations is probably compensated by the time needed to compute the criterion.

When looking at performance on the NMF problem, conclusions are surprisingly the opposite from the convex case. Indeed, in this case using exactly 10 iterations is the fastest and the most accurate, while the combination is significantly slower. Methods using maximum 50 iterations have speed in between and obtain again very similar results. A plausible explanation that could explain this very different behavior is the following. The

goal in NMF is to obtain quickly a "good enough" projection, to update $\boldsymbol{A}$ quickly, knowing that $\boldsymbol{X}$ will still change, so it is not necessary to find the optimal $\boldsymbol{A}$ for $\boldsymbol{X}$ given. As using only 10 iterations is faster than combining methods, this approach is the most efficient. In the convex case, the combination needs fewer iterations to converge as each iteration is better and it is faster thanks to that.

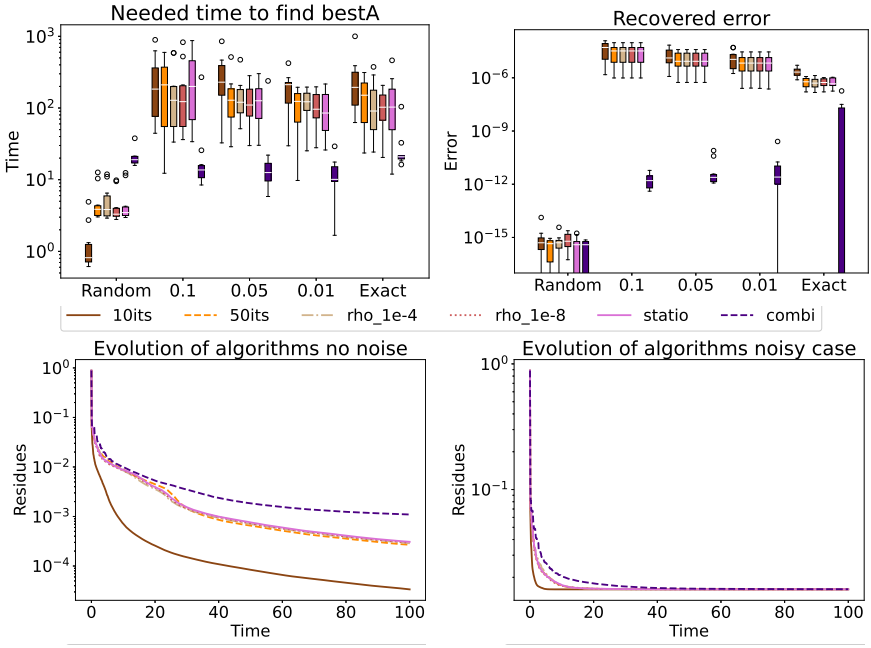


Fig. 6.9 Performance of the ADMM approach for the projection, using various parameters. Methods using a stopping criterion are limited to 50 iterations. Top: performance on convex problem with fixed $\boldsymbol{X}$. Bottom: performance on NMF problem.

Parameters of Burer-Monteiro method

The main parameters of the Burer-Monteiro method are the rank and precision of the projection, that can be controlled through the maximal number of iterations and the tolerance. Three quantities are evaluated throughout the algorithm: the change of the objective function, the difference between successive iterates, and the norm of the gradient. When one of these quantities becomes smaller than the tolerance, the algorithm is stopped. Let us

first evaluate good values of precision parameters for rank 1.

We tested the following parameters:

- Maximum 10 iterations - tolerance = 10^{-12} (**10 its**)
- Maximum 50 iterations - tolerance = 10^{-12} (**50 its**)
- Maximum 10^6 iterations - tolerance = 10^{-2} (**1e-2**)
- Maximum 10^6 iterations - tolerance = 10^{-4} (**1e-4**)
- Maximum 10^6 iterations - tolerance = 10^{-6} (**1e-6**)
- Maximum 10^6 iterations - tolerance = 10^{-8} (**1e-8**)

We observe in Figure 6.10 that on the convex problem, using maximum 10 iterations during the projection is faster than using maximum 50 iterations. Similarly, a tolerance of 10^{-2} leads to a faster resolution than when using a tolerance of 10^{-4} that is faster than 10^{-6} which itself is faster than 10^{-8}. This might seem logical since methods with a higher tolerance will interrupt the projection algorithm earlier. However, it is important to remember that this projection is only one step in the iterative algorithm and that these more tolerant methods might require more iterations to converge, which does not seem to be the case. It is also interesting to note that random and exact cases are solved quicker than during NMF process.

When we look at the error obtained by each algorithm on the convex case, we observe that a tolerance of 10^{-2} or 10^{-4} almost never allows the algorithm to converge to the solution. This is not so surprising because these two tolerances are quite high. Allowing to have maximum 10 iterations may also sometimes lead to difficulties to find the exact solution, but most of the time some tests among the ten tests performed are able to converge to the exact solution. Other parameters are able to find the exact solution at least once. Nevertheless, all methods have comparable average error.

Looking now at the test on the full NMF problem, we observe that using a tolerance of 10^{-2} leads to the fastest method, but when there is no noise this method converges to factors with higher residual. However, there is no such problem in the noisy case. This is not very surprising as in the convex case the Burer-Monteiro approach with tolerance 10^{-2} was also not able to converge to the solution. However, using maximum 10 iterations is also very fast and obtains the best residual after 100 seconds in both noisy and not noisy cases. This method was not significantly faster than the others in

the convex case, but stands out in terms of speed in the NMF problem. A plausible explanation for the good behavior of this method is the fact that its precision evolves during the solving process. Indeed, at the beginning, the initial guess of the projection is inaccurate and allowing only 10 iterations leads to a poor result, but as the algorithm evolves, the initial guess of the projection becomes more and more accurate, and the found solutions are more and more precise. This behavior is probably more helpful for the NMF problem, which explains the supremacy of the approaches using 10 iterations, both for Burer-Monteiro and ADMM approaches.

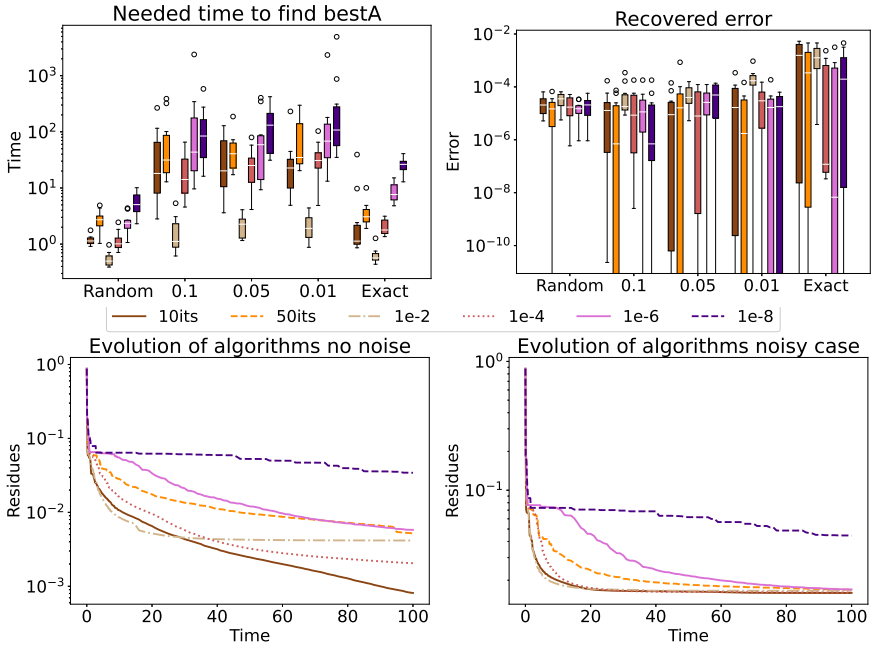


Fig. 6.10 Performance of the Burer-Monteiro approach using various parameters. Top: performance on convex problem with fixed $\boldsymbol{X}$. Bottom: performance on NMF problem.

We now analyze the performance of the Burer-Monteiro approach when rank varies. We test this approach on ranks $1, 2, 3, 4$ and 5 using maximum 10 iterations. Results are summarized in Figure 6.11. We observe on this figure that, in the convex case, increasing the rank clearly increases the needed time. Moreover, on average, using only a rank equals to 1 obtains worse results than using a rank equals to 2 or more. For higher ranks

there is no significant differences in terms of error. These results on the convex problem suggest using a rank-2 Burer-Monteiro approach. This is confirmed by the test performed on the full NMF problem as this approach is on average the fastest method and leads to residues among the best.

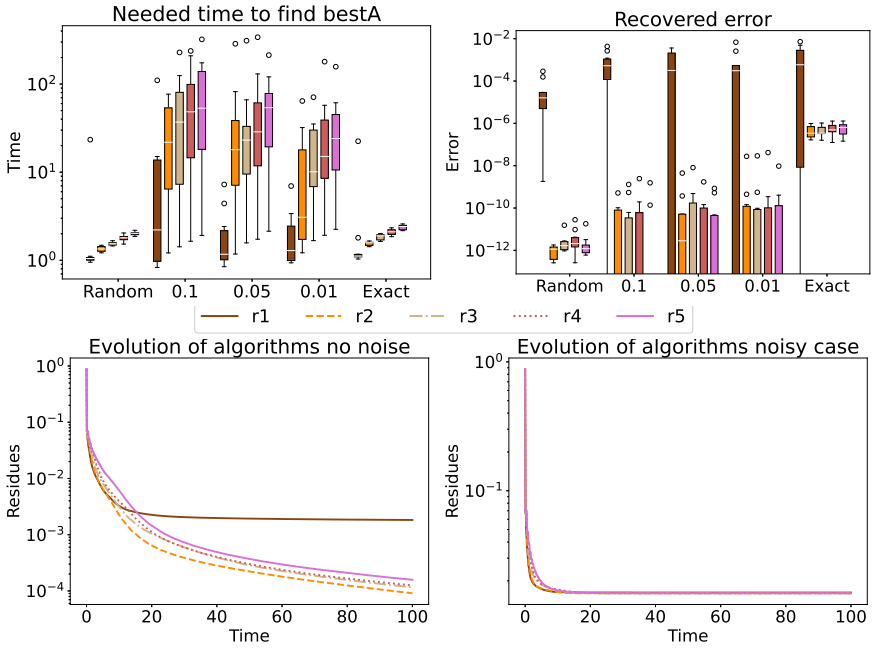


Fig. 6.11 Performance of Burer-Monteiro approach using various ranks. Top: performance on convex problem with fixed $\boldsymbol{X}$. Bottom: performance on NMF problem.

This section allowed us to determine the most interesting parameters for the ADMM and Burer-Monteiro approaches. We will now compare the performance of these two algorithms with their best parameters.

6.3.4 Comparison of projection approaches for P-HALS

We first analyze the performance of the different projections proposed in the context of the NMF problem where original signals are polynomials (dataset **Pol**). In this case, it is possible for the methods to recover exactly the factors of the input data.

The comparison includes the following algorithms:

- **Vectors:** the standard HALS algorithm, solving the problem on vectors and not polynomials,
- **IP:** P-HALS using interior-point projection method,
- **BM:** P-HALS using Burer-Monteiro projection, with maximum 10 iterations and rank 2,
- **ADMM:** P-HALS using ADMM with exactly 10 iterations,
- **ADMM_p:** P-HALS using ADMM with Boyd's stopping criterion and `MaxRho`=10^4, `MinRho`=10^{-4}, with maximum 10 iterations,
- **ADMM_s:** P-HALS using ADMM stopping criterion on stationarity, with maximum 10 iterations,
- **Heuri:** P-HALS using heuristic H_1 as projection.

Those algorithms are then compared in several situations where data is either noiseless or with noise level 20 dB, and where $(m, n, r, d) = (500, 500, 5, 20)$; $(500, 500, 10, 20)$; $(50, 500, 5, 20)$; $(500, 50, 5, 20)$ or $(500, 500, 5, 6)$. Each set of parameters is tested 10 times.

On noiseless data

We observe in Figure 6.12 that HALS using vectors is very accurate: it is fast and obtains a low residual as there is no noise to perturb the algorithm. P-HALS using heuristic H_1 is the only method that is able to beat it. Indeed, this approach obtains a lower residual and obtains comparable SIR and SIR LC in comparable time. In general, methods can be classified as follows (from best to worst): Heuri - Vectors - ADMM-based approaches - BM - IP. We see that the goal to accelerate P-HALS using interior point projection (IP) is achieved for all methods. Heuristic H_1 is ultimately faster than the projections based on theory.

When m is smaller ($m = 50$ instead of 500), HALS using vectors obtains a lower residual and a higher SIR LC, while heuristic H_1 obtains a worse residual, but its SIR LC stays good on average. Both methods obtain a lower SIR. Other approaches have unchanged performance. Having only 50 discretization points decreases the size of the dataset, but this is not enough to explain the better performance of HALS using vectors, as this method is not better when $n = 50$. The explanation may come from the fact that observing 50 points of a degree 20 polynomials leads to a observed

signal that is less smooth, with less structure, that can be easier to factor by standard NMF. This would also explain why the SIR is a bit lower (having less information about structure makes it more difficult to reconstruct the original signals), while the two other criteria are higher. The fact that the heuristic approach obtains a high residual and a high SIR LC on average may look strange, but is explained by the fact that the heuristic obtains a high residual and a low SIR LC in a few tests and good results for the others. As the residues have value close to 0, their average value is more impacted than the average value of the SIR LC (e.g. $\frac{10^{-1}+9\cdot 10^{-8}}{10} \simeq 10^{-2}$ while $\frac{40+9\cdot 100}{10} = 94$). P-HALS using heuristic H_1 is therefore not able to converge in all ten tests performed. As this approach has no convergence properties, it is not surprising that its performance is degraded when the signals are known less precisely.

When changing other parameters (n, r, d) the results do not change significantly, and are therefore not presented in Figure 6.12.

On noisy data

We observe in Figure 6.13 that HALS using vectors has difficulties to filter out the noise. Indeed, it obtains a higher residual and a lower SIR LC. The SIR is also a bit lower but not significantly. This is worsened when the number of observation is low ($n = 50$ instead of 500). Among methods using polynomials, the heuristic is the fastest, followed by ADMM approaches, then Burer-Monteiro and finally the interior-point method. This is the same order than in the noiseless case. All methods obtain comparable results in terms of SIR, SIR LC and residual after 100 seconds.

We observe in Figure 6.14 that when m is decreased to 50, HALS using vectors obtains a residual comparable to the other approaches, while P-HALS using projections relying on Burer-Monteiro approach or interior-point method are the methods obtaining the best SIR and SIR LC. Moreover, P-HALS using ADMM projection obtains results (very) slightly better than P-HALS using heuristic H_1. When changing other parameters (r, d) the results do not change significantly, and are therefore not presented.

In general, on polynomial signals, NMF using vectors obtain good results in the noiseless case, but has difficulties to filter out the noise. The only method able to compete with NMF using vectors on the noiseless case, both in terms of time and accuracy, is P-HALS using heuristic H_1. In general, this method obtains very good results, even though it is less ef-

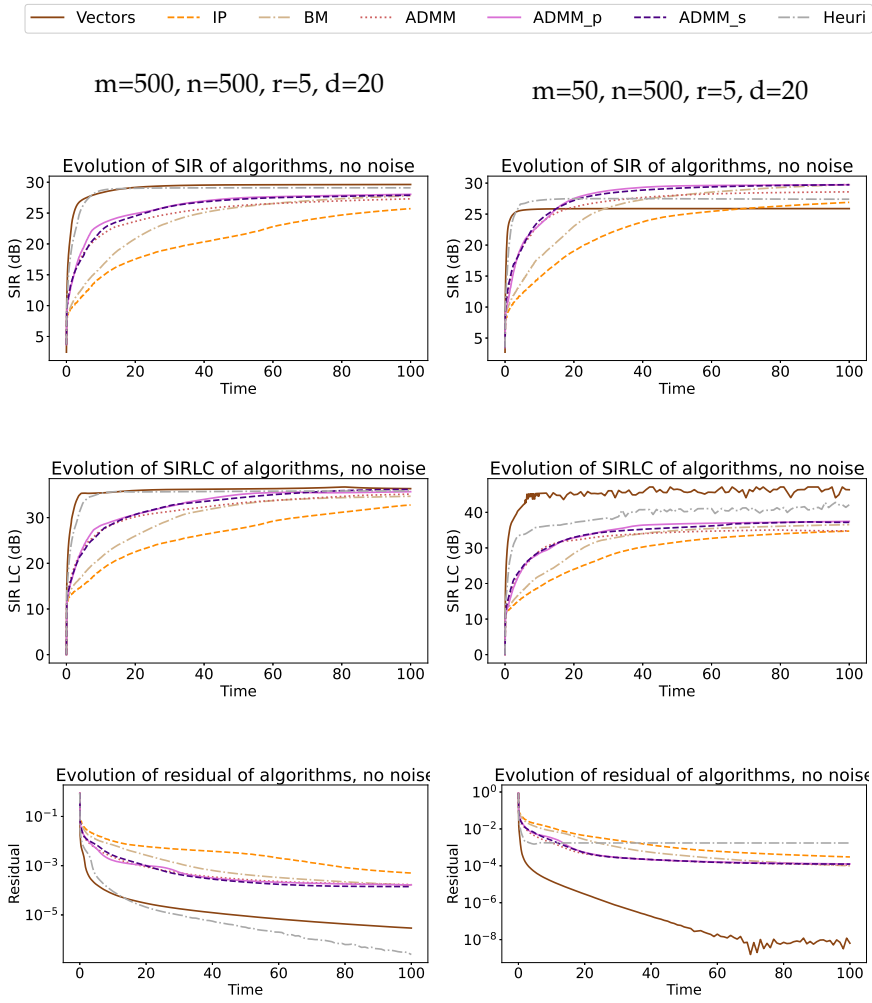


Fig. 6.12 Evolution graph of SIR, SIR LC and residual of the tested methods, when $(m, n, r, d) = (500, 500, 5, 20)$ and $(m, n, r, d) = (50, 500, 5, 20)$. The data is noiseless. Each evolution graph is the average of 10 tests.

ficient when the number of observation points (m) is not large enough. P-HALS using ADMM projection is in general faster than P-HALS using Burer-Monteiro projection that is faster than P-HALS using interior-point projection, while all methods obtain comparable results in terms of SIR,

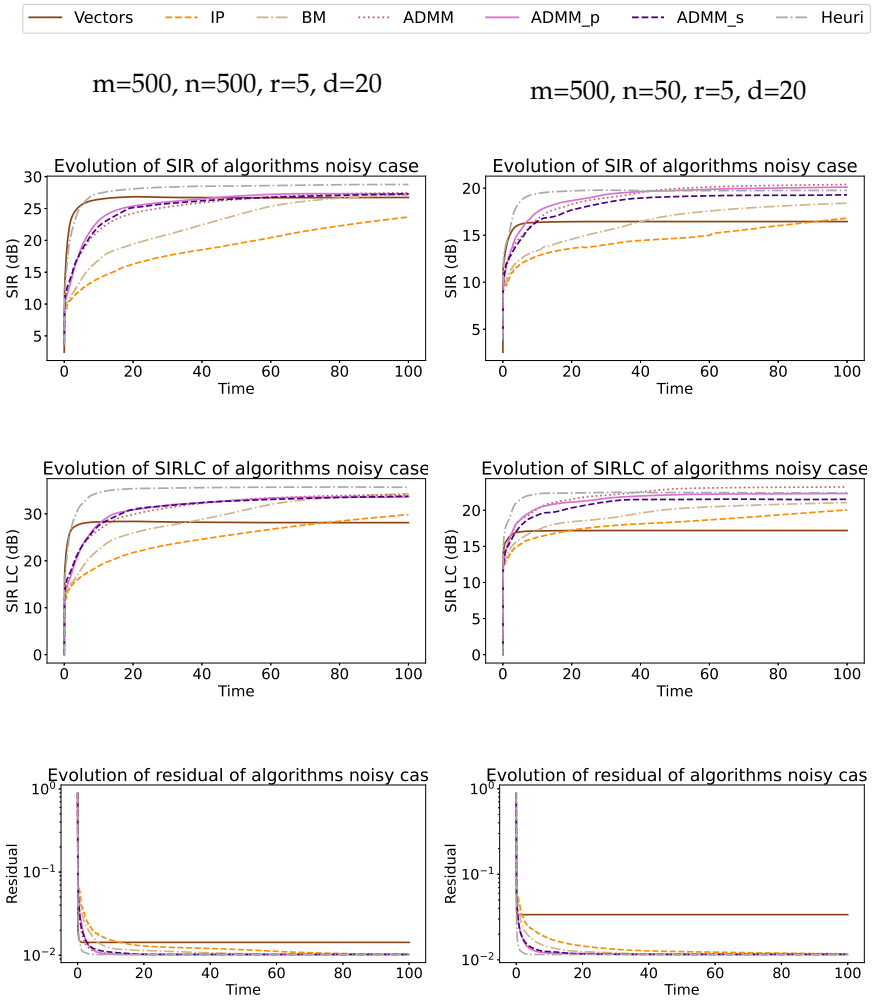


Fig. 6.13 Evolution graph of SIR, SIR LC and residual of the tested methods, when $(m,n,r,d) = (500,500,5,20)$ and $(m,n,r,d) = (500,50,5,20)$. The data is noisy. Each evolution graph is the average of 10 tests.

SIR LC and residual. The objective of this work, that is accelerating NMF using polynomials with interior-point projection, is therefore met.

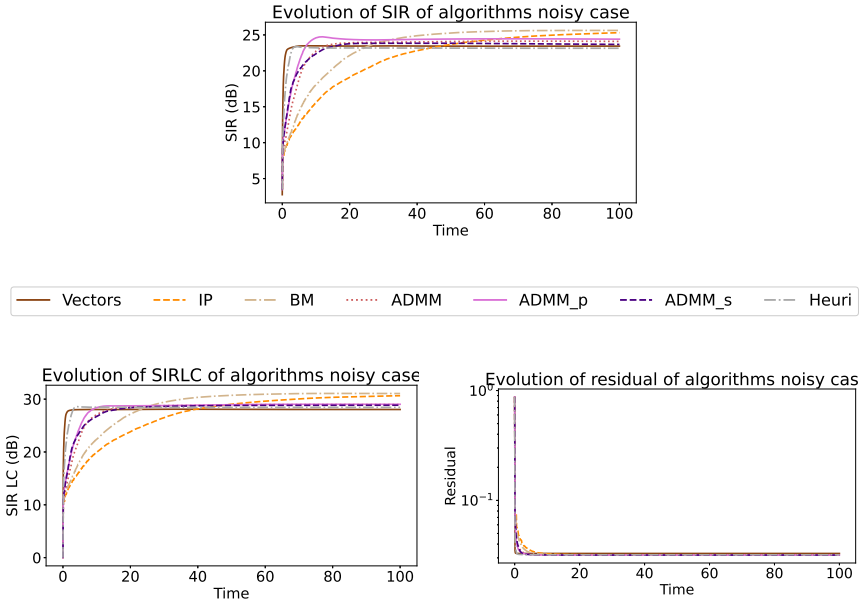


Fig. 6.14 Evolution graph of SIR, SIR LC and residual of the tested methods, when $(m, n, r, d) = (50, 500, 5, 20)$. The data is noisy. Each evolution graph is the average of 10 tests.

Performance on reflectance signals ***Refl***

We also test the methods when the input signals are mixtures of reflectance signals (**Refl**), to confirm the results when signals do not come from polynomials. Our methods can therefore not recover the input signals exactly. We tested the factorization over $n = 100$ observations using degree 12, 20 or 40 polynomials, but they were no main differences between these three cases: using a degree equals to 12 leads to worse residuals, while using a degree equals to 40 leads to slightly worse factors in terms of SIR LC, but the comparison between the different approaches stays identical. We tested the factorization on noiseless data, and noisy data with noise level 20 dB.

Figure 6.16 shows the results for degree $d = 20$ with or without noise. We observe that polynomial-based approaches are better in terms of both SIR LC and residual than HALS based on vectors. However, all methods are barely able to recover the original signals, as it can observed by comparing

Figures 6.8 and 6.15.

Among polynomial-based methods, using as projection heuristic H_1 is again the fastest and leads to very accurate results, especially when there is no noise in the data. ADMM-based methods are also quite efficient, while Burer-Monteiro approach is slower and a bit less accurate, and interior-point projection leads to the slowest and the less accurate results after 100 seconds.

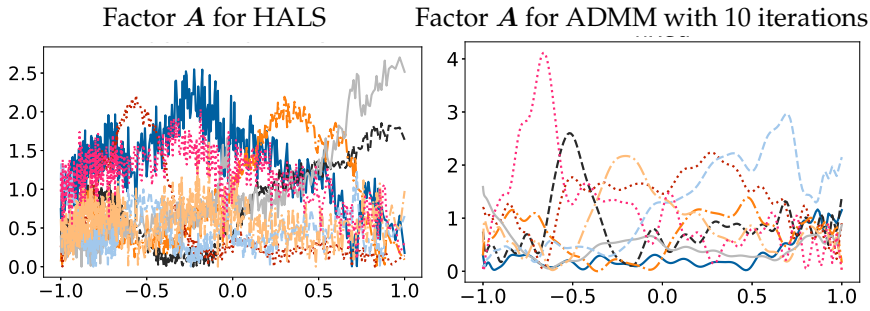


Fig. 6.15 Illustration of signals recovered by HALS using vectors (left) and ADMM using 10 iterations (right).

6.4 Discussion

In this chapter, we have presented several ways to project polynomials on the set of polynomials of same degree that are nonnegative on a given interval. We have first analyzed heuristic projections, that are not ensured to converge but find quickly a good approximation. We observed that the heuristic named H_1 was the most promising. This heuristic projects a polynomial by truncating its negative values, and fitting the obtained signal.

We then tested two methods based on iterative algorithms with early stopping. One is based on the Alternating Direction Method of Multipliers (ADMM), and the other one is the result of the application of the Burer-Monteiro approach on the projection problem. Those two methods can be warm started by a good guess of the solution, which makes them particularly interesting for use in iterative algorithms that can use the solutions of the previous iteration as first guess.

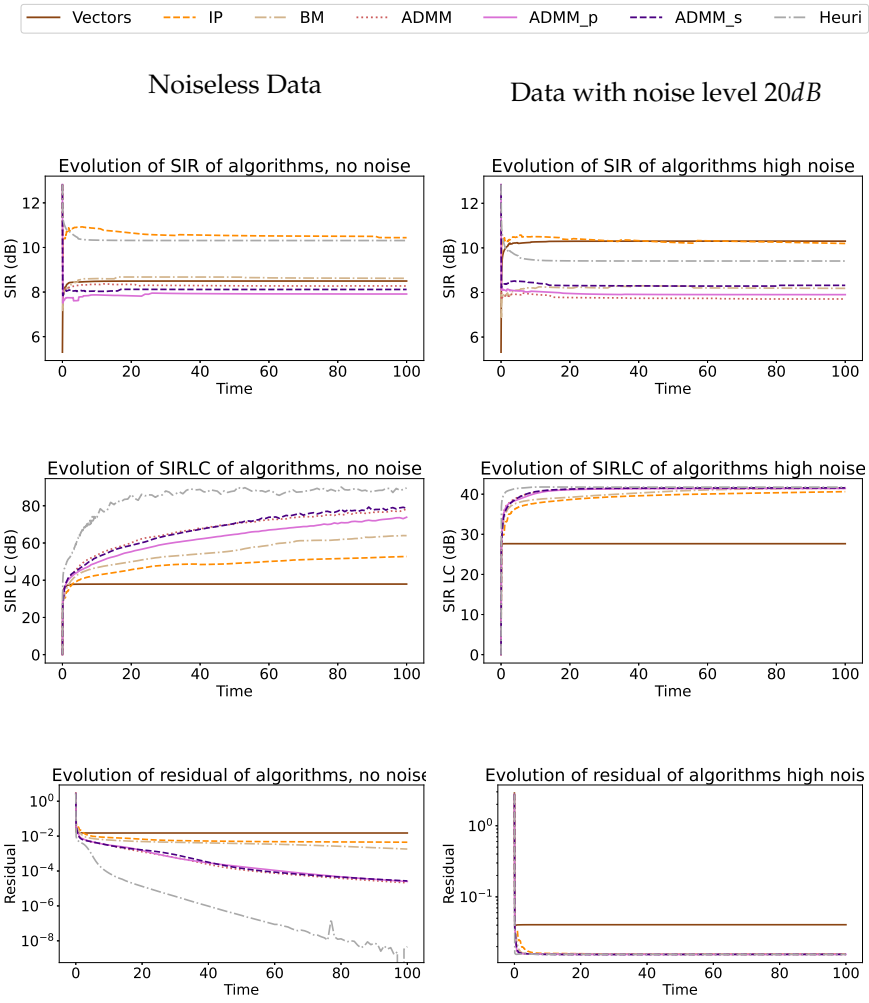


Fig. 6.16 Evolution graph of SIR, SIR LC and residual of the tested methods over $n = 100$ mix of real reflectance signals. The used degree for polynomials is $d = 20$. Each evolution graph is the average of 10 tests.

We observed during our tests that many of the proposed approximate projections are indeed accelerating the P-HALS algorithm, without impacting the quality of the recovered solution. Actually, the most heuristic methods, i.e. ADMM using 10 iterations and our best heuristic H_1 are on average

the more efficient methods, as they are faster and not less accurate than the other approaches. The heuristic H_1 is even comparable to HALS using vectors in terms of time, while having the good accuracy properties of P-HALS most of the time. However, when the number of discretization point is too low, P-HALS using the heuristic method has more difficulties to converge to a good point and obtains worse results than the other versions of P-HALS. Otherwise this method is recommended.

Therefore, if the number of observation points is large compared to degree of the used polynomials, using P-HALS with the heuristic projection H1 is competitive with standard HALS in terms of time while being very accurate. If the number of observation points is smaller, it is better to use ADMM with 10 iterations to perform the projection. This algorithm is also quite fast, while always being accurate (in the tests we performed at least).

To conclude, we explored the possibility of using iterative algorithms with early stopping to perform the projection in P-HALS as well as some heuristic projections, and showed that this accelerates the algorithm. However, we also observed in our tests that a projection approach can be very fast and accurate for the convex problem $\min_{\boldsymbol{A}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}^\top\|$ (that considers $\boldsymbol{X}$ as fixed), while being very slow and inaccurate for the real non convex full NMF problem $\min_{\boldsymbol{A},\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}^\top\|$. It is therefore particularly important to test the approximate projections within the P-HALS algorithm to see if they are relevant, and not use only theoretical information.

7

Application: Using NMF with splines for image completion

Many state-of-the art methods for image completion rely on two main assumptions: images are smooth, except at edges, and low-rank [65, 86, 134]. As images are composed of nonnegative pixels, NMF can be an interesting approach since it factorizes a matrix as the product of two low-rank matrices (resulting thus in a low-rank matrix) [106, 114, 127]. The smoothness is usually enforced via regularization, either on the produced image [86], or on its factors [134].

Recently, Sadowski and Zdunek proposed a different technique where the smoothness constraint is imposed on one factor of the NMF problem by constraining it to be a linear combination of piece-wise smooth elements, namely B-Splines [106]. This approach leads to promising results, but at the cost of a relatively high computational time. Our goal is to use a similar idea for image completion while keeping a low computational effort, using for this our LP-HALS algorithm from Chapter 3 for NMF using splines.

We test two approaches: either using our LP-HALS algorithm to constrain one factor of the NMF problem to contain splines and therefore be smooth,

using an approach very similar to Sadowski and Zdunek, or imposing both factors of the NMF problem to contain splines. We observe that using LP-HALS to impose smoothness on one factor is faster and slightly more accurate than the original approach of Sadowski and Zdunek, while imposing smoothness simultaneously on both factors of the NMF problem is more robust, and more accurate for difficult problems (with many missing pixels or with noisy images), but a bit less accurate in easier situations.

7.1 Image completion using smooth NMF

Smooth-NMF, presented in [140], consists in the following problem: given a matrix $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$, a factorization rank r and a (discretized) B-Spline basis with d splines $\boldsymbol{S} \in \mathbb{R}^{m \times d}$, find matrices $\boldsymbol{A} \in \mathbb{R}^{m \times r}$, $\boldsymbol{B} \in \mathbb{R}^{d \times r}$ and $\boldsymbol{X} \in \mathbb{R}^{r \times n}$ minimizing

$$\|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}^\top\|_F^2 \quad \text{such that} \quad \boldsymbol{A} = \boldsymbol{S}\boldsymbol{B} \geq 0, \boldsymbol{X} \geq 0. \tag{7.1}$$

To use smooth-NMF for image completion, Sadowski and Zdunek proposed Algorithm 7.1 in [106], inspired from [134]. This algorithm successively solves a smooth-NMF problem based on a guess $\boldsymbol{Y}$ of the image $\boldsymbol{W}$ to be recovered. This guess is updated using the low-rank matrix $\boldsymbol{A}\boldsymbol{X}^\top$ produced by NMF, where known pixels are replaced by their true value. Iterations are stopped once $\boldsymbol{A}\boldsymbol{X}^\top$ is close enough to $\boldsymbol{W}$ on the known pixels. To ensure smoothness on both the rows and the columns, smooth-NMF is performed on the image and on its transpose.

Note that lines 5 to 7 in Algorithm 7.1 where inferred from descriptions in [106], as the authors did not provide pseudocode of these elements. Moreover, the condition of the while loop has been slightly modified by removing the absolute value to stop the algorithm when the error increases. As the image can be smoother in its rows or its column, the error can increase between line 5 and 7. Nevertheless, we want this error to decrease after each full iteration, otherwise the algorithm is stopped.

To solve the smooth-NMF problem, the authors used the algorithm from [140], which solves the problem alternatively on matrices $\boldsymbol{A}$ and $\boldsymbol{X}$. Matrix $\boldsymbol{X}$ is updated using the popular Hierarchical Alternating Least Squares (HALS) method (Algorithm 2.1), while matrix $\boldsymbol{A}$ is updated with the Alter-

Algorithm 7.1 B-Splines-based Algorithm for Image Completion (BSA-IC)

This algorithm iteratively uses NMF over splines in one factor to smooth the rows or the columns of the resulting image.
Input: $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ is the incomplete matrix with support Ω. $\boldsymbol{Y}$ is a first guess for $\boldsymbol{W}$, by default $0^{m \times n}$, $\boldsymbol{A} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{X} \in \mathbb{R}^{n \times r}$ are initialized randomly if not provided. $\boldsymbol{A}$ is initialized to contain nonnegative splines.

function BSA-IC($\boldsymbol{W}, \Omega, \boldsymbol{Y}, \boldsymbol{A} \geq 0, \boldsymbol{X} \geq 0, \delta = 10^{-2}$)
2: $\boldsymbol{E} = 0, \boldsymbol{E}_\Omega = \left(\boldsymbol{W} - \boldsymbol{A}\boldsymbol{X}^\top\right)_\Omega = 0$ ▷ $\boldsymbol{E}$ contains the initial error
$e_2 = \|\boldsymbol{E}\|_F, e_1 = e_2 + \delta + 1$
4: $\boldsymbol{Y}_\Omega = \boldsymbol{W}_\Omega$
while $\frac{e_1 - e_2}{e_2} > \delta$ **do**
6: $(\boldsymbol{A}, \boldsymbol{X})$ = `Smooth-NMF`$(\boldsymbol{Y}, \boldsymbol{A}, \boldsymbol{X})$ ▷ Smooth rows
$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}^\top, \boldsymbol{Y}_\Omega = \boldsymbol{W}_\Omega$
8: $(\boldsymbol{X}, \boldsymbol{A})$ = `Smooth-NMF`$(\boldsymbol{Y}^\top, \boldsymbol{X}, \boldsymbol{A})$ ▷ Smooth columns
$\boldsymbol{E} = 0, \boldsymbol{E}_\Omega = \left(\boldsymbol{W} - \boldsymbol{A}\boldsymbol{X}^\top\right)_\Omega = 0$
10: $e_1 = e_2, e_2 = \|\boldsymbol{E}\|_F$
$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}^\top, \boldsymbol{Y}_\Omega = \boldsymbol{W}_\Omega$
12: **return** $\boldsymbol{Y}$

nating Direction Method of Multipliers (ADMM) [48]. The ADMM update of $\boldsymbol{A}$ solves

$$\min_{\boldsymbol{A},\boldsymbol{B}} \tfrac{1}{2}\|\boldsymbol{Y} - \boldsymbol{S}\boldsymbol{B}\boldsymbol{X}^\top\|_F^2 + \Phi(\boldsymbol{A}) \quad \text{such that} \quad \boldsymbol{S}\boldsymbol{B} = \boldsymbol{A} \tag{7.2}$$

where $\Phi(\boldsymbol{A})$ is the indicator function of the nonnegative set ($\Phi(a) = 0$ if $a \geq 0$, $+\infty$ otherwise). Applying ADMM on this problem gives the following iterative scheme, using $[\xi]_+ = \max\{0, \xi\}$, $\tau > 0$ and the pseudo-inverse $\boldsymbol{S}^\dagger = (\boldsymbol{S}^\top\boldsymbol{S})^{-1}\boldsymbol{S}^\top$:

$$\begin{aligned}
\boldsymbol{B}^{t+1} &= \boldsymbol{S}^\dagger\left[\boldsymbol{Y}\boldsymbol{X} + \boldsymbol{\Lambda}^t + \rho\boldsymbol{A}^t\right]\left(\boldsymbol{X}^\top\boldsymbol{X} + \rho\boldsymbol{I}^r\right)^{-1} \\
\boldsymbol{A}^{t+1} &= \left[\boldsymbol{S}\boldsymbol{B}^{t+1} - \rho^{-1}\boldsymbol{\Lambda}^t\right]_+,\ \boldsymbol{\Lambda}^{t+1} = \boldsymbol{\Lambda}^t + \rho(\boldsymbol{A}^{t+1} - \boldsymbol{S}\boldsymbol{B}^{t+1}).
\end{aligned}$$

Algorithm 7.1 is able to complete images properly but at the cost of a relatively high CPU time compared to existing approaches. We propose to tackle the smooth-NMF subproblem using LP-HALS from Chapter 3 on splines, that we denote as S-HALS. This method is potentially faster than the ADMM approach, especially when fast heuristics are used to project splines onto their nonnegative set. Indeed, it has been shown that using heuristics for the projection step could benefit the algorithm in terms of CPU time without impacting significantly its accuracy (see Section 5.3 and Chapter 6).

Our S-HALS method alternates between updates of matrix $\boldsymbol{B}$ (the coefficients of $\boldsymbol{A} = \boldsymbol{S}\boldsymbol{B}$) and matrix $\boldsymbol{X}$. This algorithm is explained in details in Algorithm 3.1, and is recalled in Algorithm 7.2 below. Projection in line 5 of `updateB` can be done exactly (see Section 3.2.2), but instead we choose to compute it in an approximate but much faster way: we project each B-Spline coefficients on the nonnegative set, which ensures that the resulting spline is nonnegative (although not all nonnegative splines can be obtained in this way). This is a heuristic way to perform the projection with the advantage of being very fast.

Both S-HALS and the ADMM algorithm are limited to 500 iterations (we observed during our tests that it was necessary to allow the algorithm to make a few hundred iterations to reach a satisfactory convergence) and are stopped when the relative residual is below 10^{-7}. Let $\boldsymbol{A}^t$ and $\boldsymbol{X}^t$ be the

matrices obtained at iteration t. The relative residual is:

$$\frac{\|\boldsymbol{Y} - \boldsymbol{A}^t \boldsymbol{X}^{t\top}\| - \|\boldsymbol{Y} - \boldsymbol{A}^{t+1} \boldsymbol{X}^{t+1\top}\|}{\|\boldsymbol{Y} - \boldsymbol{A}^{t+1} \boldsymbol{X}^{t+1\top}\|} < 10^{-7}. \tag{7.3}$$

Algorithm 7.2 S-HALS (a Smooth-NMF method)

$$\boldsymbol{M} = \boldsymbol{S}^\top \boldsymbol{S},\ \boldsymbol{Z} = \boldsymbol{S}^\top \boldsymbol{Y},\ \boldsymbol{M}_1 = \boldsymbol{M}^{-1}\boldsymbol{Z}$$

function UPDATEB($\boldsymbol{M}_1, \boldsymbol{B}, \boldsymbol{X}$)
  $\boldsymbol{P} = \boldsymbol{M}_1\boldsymbol{X}, \boldsymbol{Q} = \boldsymbol{X}^\top\boldsymbol{X}$
  for $\boldsymbol{B}_{:k}$ in $\boldsymbol{B}$ **do**
    $\boldsymbol{t} = \boldsymbol{P}_{:k} - \sum_{j\neq k} \boldsymbol{B}_{:j}\boldsymbol{Q}_{jk}$
    $\boldsymbol{B}_{:k} \leftarrow$ `Projection`$(\boldsymbol{t}/\boldsymbol{Q}_{kk})$
  return $\boldsymbol{B}$

function UPDATEX($\boldsymbol{Z}, \boldsymbol{M}, \boldsymbol{B}, \boldsymbol{X}$)
  $\boldsymbol{P} = \boldsymbol{Z}^\top\boldsymbol{B}, \boldsymbol{Q} = \boldsymbol{B}^\top\boldsymbol{M}\boldsymbol{B}$
  for $\boldsymbol{X}_{k:}$ in $\boldsymbol{X}$ **do**
    $\boldsymbol{t} = \boldsymbol{P}_{k:} - \sum_{j\neq k} \boldsymbol{Q}_{kj}\boldsymbol{X}_{j:}$
    $\boldsymbol{X}_{k:} \leftarrow \max(0, \boldsymbol{t}/\boldsymbol{Q}_{kk})$
  return $\boldsymbol{X}$

7.2 Image completion using splines in both factors

In Algorithm 7.1, the image is smoothed by alternatively smoothing its rows and its columns. We consider in this section the possibility to impose both $\boldsymbol{A}$ and $\boldsymbol{X}$ to contain splines at the same time, in order to smooth both rows and columns of the image at the same time. Given a matrix $\boldsymbol{W} \in \mathbb{R}^{m\times n}$ whose entries are known on support Ω, we aim to minimize the distance between $\boldsymbol{W}$ and $\boldsymbol{A}\boldsymbol{X}^\top$ on Ω . Moreover, we decide to have the possibility to use a guess of the solution, $\boldsymbol{Y}$, and we aim to minimize the distance between $\boldsymbol{Y}$ and $\boldsymbol{A}\boldsymbol{X}^\top$ on $\overline{\Omega}$ (the entries not belonging to Ω). A parameter λ allows us to balance the weight between these two distances. Let $\boldsymbol{S}_A$ be the B-Spline basis for splines in $\boldsymbol{A}$, with $\boldsymbol{B}_A$ the coefficients of the splines, and similarly for $\boldsymbol{S}_X$ and $\boldsymbol{B}_X$. The problem is thus

$$\min_{\boldsymbol{B}_A\geq 0, \boldsymbol{B}_X \geq 0} C(\boldsymbol{B}_A, \boldsymbol{B}_X) = \sum_{(i,j)\in\Omega} \left(\boldsymbol{W}_{i,j} - (\boldsymbol{S}_A\boldsymbol{B}_A\boldsymbol{B}_X^\top\boldsymbol{S}_X^\top)_{i,j}\right)^2 \tag{7.4}$$

$$+ \lambda \sum_{(i,j)\in\overline{\Omega}} \left(\boldsymbol{Y}_{i,j} - (\boldsymbol{S}_A\boldsymbol{B}_A\boldsymbol{B}_X^\top\boldsymbol{S}_X^\top)_{i,j}\right)^2 \tag{7.5}$$

We solve this problem using an approach close to S-HALS. The gradient of

$C(\boldsymbol{B}_A, \boldsymbol{B}_X)$ with respect to a column of $\boldsymbol{B}_A$ is

$$\begin{aligned}\frac{\partial C(\boldsymbol{B}_A, \boldsymbol{B}_X)}{\boldsymbol{B}_{A:k}} =& 2 \sum_{(i,j)\in\Omega} \boldsymbol{S}_{A_{i:}}^\top \big(\boldsymbol{W}_{i,j} - (\boldsymbol{S}_A \boldsymbol{B}_A \boldsymbol{B}_X^\top \boldsymbol{S}_X^\top)_{i,j}\big) \boldsymbol{S}_{X_{j:}} \boldsymbol{B}_{X:k} \\ &+ 2\lambda \sum_{(i,j)\in\overline{\Omega}} \boldsymbol{S}_{A_{i:}}^\top \big(\boldsymbol{Y}_{i,j} - (\boldsymbol{S}_A \boldsymbol{B}_A \boldsymbol{B}_X^\top \boldsymbol{S}_X^\top)_{i,j}\big) \boldsymbol{S}_{X_{j:}} \boldsymbol{B}_{X:k} \quad (7.6)\end{aligned}$$

We then equal the gradient to zero, to obtain

$$\begin{aligned}&\sum_{(i,j)\in\Omega} \boldsymbol{S}_{A_{i:}}^\top \boldsymbol{S}_{Ai:} \boldsymbol{B}_{A:k} \boldsymbol{B}_{X:k}^\top \boldsymbol{S}_{X_{j:}}^\top \boldsymbol{S}_{Xj:} \boldsymbol{B}_{X:k} + \lambda \sum_{(i,j)\in\overline{\Omega}} \boldsymbol{S}_{A_{i:}}^\top \boldsymbol{S}_{Ai:} \boldsymbol{B}_{A:k} \boldsymbol{B}_{X:k}^\top \boldsymbol{S}_{X_{j:}}^\top \boldsymbol{S}_{Xj:} \boldsymbol{B}_{X:k} \\ &= \sum_{(i,j)\in\Omega} \boldsymbol{S}_{A_{i:}}^\top \big(\boldsymbol{W}_{i,j} - \sum_{l\neq k} \boldsymbol{S}_{Ai:} \boldsymbol{B}_{A:l} \boldsymbol{B}_{X:l}^\top \boldsymbol{S}_{X_{j:}}^\top\big) \boldsymbol{S}_{Xj:} \boldsymbol{B}_{X:k} \\ &\quad + \lambda \sum_{(i,j)\in\overline{\Omega}} \boldsymbol{S}_{A_{i:}}^\top \big(\boldsymbol{Y}_{i,j} - \sum_{l\neq k} \boldsymbol{S}_{Ai:} \boldsymbol{B}_{A:l} \boldsymbol{B}_{X:l}^\top \boldsymbol{S}_{X_{j:}}^\top\big) \boldsymbol{S}_{Xj:} \boldsymbol{B}_{X:k}.\end{aligned}$$

Let Ω_i be the set of indices j such that $(i,j) \in \Omega$, and similarly for $\overline{\Omega}_i$. Suppose also that $\boldsymbol{D}_X$ is a diagonal matrix, with diagonal element $\boldsymbol{D}_{Xi,i} = \sum_{j\in\Omega_i} \boldsymbol{B}_{X:k}^\top \boldsymbol{S}_{X_{j:}}^\top \boldsymbol{S}_{Xj:} \boldsymbol{B}_{X:k} + \lambda \sum_{j\in\overline{\Omega}_i} \boldsymbol{B}_{X:k}^\top \boldsymbol{S}_{X_{j:}}^\top \boldsymbol{S}_{Xj:} \boldsymbol{B}_{X:k}$. We have then

$$\begin{aligned}\boldsymbol{B}_{A:k} = (\boldsymbol{S}_A^\top \boldsymbol{D}_X \boldsymbol{S}_A)^{-1} \Big(&\sum_{(i,j)\in\Omega} \boldsymbol{S}_{A_{i:}}^\top \big(\boldsymbol{W}_{i,j} - \sum_{l\neq k} \boldsymbol{S}_{Ai:} \boldsymbol{B}_{A:l} \boldsymbol{B}_{X:l}^\top \boldsymbol{S}_{X_{j:}}^\top\big) \boldsymbol{S}_{Xj:} \boldsymbol{B}_{X:k} \\ &+ \lambda \sum_{(i,j)\in\overline{\Omega}} \boldsymbol{S}_{A_{i:}}^\top \big(\boldsymbol{Y}_{i,j} - \sum_{l\neq k} \boldsymbol{S}_{Ai:} \boldsymbol{B}_{A:l} \boldsymbol{B}_{X:l}^\top \boldsymbol{S}_{X_{j:}}^\top\big) \boldsymbol{S}_{Xj:} \boldsymbol{B}_{X:k}\Big).\end{aligned}$$

Finally, we threshold the coefficients to zero. This is not the optimal update of $\boldsymbol{B}_{A_{:k}}$ when $\boldsymbol{B}_{A_{:j}}$, $j \neq k$ and $\boldsymbol{X}$ are fixed, but has the advantage of being very fast:

$$\boldsymbol{B}_{A:k}^* = \big[\boldsymbol{B}_{A:k}\big]_+. \quad (7.7)$$

As the problem is symmetric in its two factors, we can derive the updates

of the columns of $\boldsymbol{X}$ from the updates of the columns of $\boldsymbol{A}$:

$$\boldsymbol{B}_{X:k}^{*} = \Bigg[(\boldsymbol{S}_X^\top \boldsymbol{D}_A \boldsymbol{S}_X)^{-1} \Big(\sum_{(i,j)\in\Omega} \boldsymbol{S}_{Xi:}^\top (\boldsymbol{W}_{ij}^\top - \sum_{l\neq k} \boldsymbol{S}_{Xi:} \boldsymbol{B}_{X:l} \boldsymbol{B}_{A:l}^\top \boldsymbol{S}_{Aj:}^\top) \boldsymbol{S}_{Aj:} \boldsymbol{B}_{A:k} + \lambda \sum_{(i,j)\in\overline{\Omega}} \boldsymbol{S}_{Xi:}^\top (\boldsymbol{Y}_{ij}^\top - \sum_{l\neq k} \boldsymbol{S}_{Xi:} \boldsymbol{B}_{X:l} \boldsymbol{B}_{A:l}^\top \boldsymbol{S}_{Aj:}^\top) \boldsymbol{S}_{Aj:} \boldsymbol{B}_{A:k} \Big) \Bigg]_+ \tag{7.8}$$

We can then solve this problem using an iterative algorithm, presented in Algorithm 7.3. As each inner iteration of this algorithm is very time-consuming, we limit the number of inner iterations to 10. We observed during experiments that this limit was enough. A particular case of this Algorithm is when $\lambda = 0$. In this case the guess of $\boldsymbol{W}$, namely $\boldsymbol{Y}$, is not used. It makes therefore no sense to repeat the main loop in this case, and the limit number of inner iterations is increased to 500. Note that even when λ is chosen as different from 0, we use $\lambda = 0$ for the very first iteration, as no previous guess is known at this stage.

Figure 7.1 shows results for various values for λ on an image of a boat with 512×512 pixels, with 90% of missing pixels. B-Spline basis for both $\boldsymbol{A}$ and $\boldsymbol{X}$ contains 100 interior knots equally spaced in interval $[0, 1]$, and the splines are discretized on 512 equally spaced points in $[0, 1]$, see next section for more details on the tests.

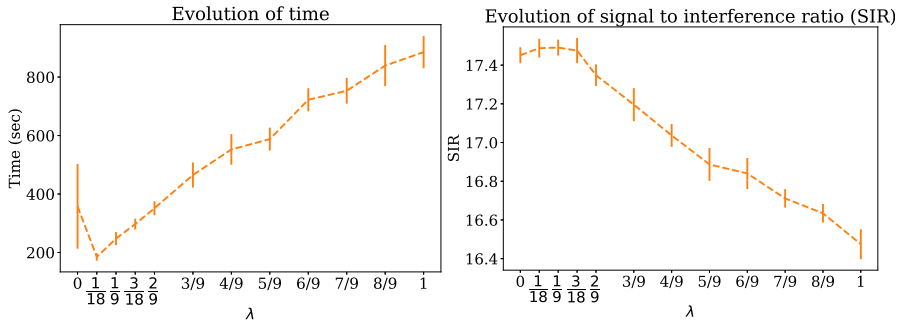


Fig. 7.1 Performance of 2BSA-IC algorithm with various values of λ on an image with 90% of missing pixels. Evolution of time (left) and signal to interference ratio (right).

We observe that the needed time increases with λ , but the most precise λ

Algorithm 7.3 2B-Splines-based Algorithm for Image Completion (2BSA-IC)

This algorithm uses NMF over splines in both factors to smooth both row and columns of the resulting image at the same time.

Input: $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ is the incomplete matrix with support Ω. λ_0 is the parameter of the algorithm, by default $\lambda_0 = \frac{\#_\Omega}{\#_{\overline{\Omega}}}$. $\boldsymbol{Y}$ is an initial guess for $\boldsymbol{W}$. If $\boldsymbol{Y}$ is not provided, the very first λ is equal to 0. $\boldsymbol{B}_A^0 \in \mathbb{R}_+^{d_A \times r}$ and $\boldsymbol{B}_X^0 \in \mathbb{R}_+^{d_X \times r}$ are initialized randomly if not provided.

function 2BSA-IC($\boldsymbol{W}$, Ω, λ_0, $\boldsymbol{Y}$, $\boldsymbol{B}_A^0 \geq 0$, $\boldsymbol{B}_X^0 \geq 0$, $\delta = 10^{-2}$, ftol = 10^{-7})

$e_2 = C(\boldsymbol{B}_A, \boldsymbol{B}_X)$, $e_1 = e_2 + \delta + 1$

if $\boldsymbol{Y}$ is None **then**

  $\lambda = 0$, $\boldsymbol{Y} = 0^{m \times n}$

while $\frac{e_1 - e_2}{e_2} > \delta$ **do** ▷ Outer loop

  $it = 0$

  $r_2 = C(\boldsymbol{B}_A, \boldsymbol{B}_X)$, $r_1 = r_2 + 1 + ftol$

  while $it < 10$ and $\frac{r_1 - r_2}{r_2} >$ftol **do** ▷ Inner loop

    for k in $0, \ldots, r$ **do**

      $\boldsymbol{B}_{A:k}$ = Equation (7.7)

    for k in $0, \ldots, r$ **do**

      $\boldsymbol{B}_{X:k}$ = Equation (7.8)

    if $\lambda == 0$ and $it == 0$ **then** ▷ Only first it. using $\lambda = 0$

      $\lambda = \lambda_0$

    $r_1 = r_2$, $r_2 = C(\boldsymbol{B}_A, \boldsymbol{B}_X)$

    $it = it + 1$

  $e_1 = e_2$, $e_2 = C(\boldsymbol{B}_A, \boldsymbol{B}_X)$

  $\boldsymbol{Y} = \boldsymbol{S}_A \boldsymbol{B}_A \boldsymbol{S}_X{}^\top \boldsymbol{B}_X{}^\top$

return $\boldsymbol{Y}$

(with higher SIR) are $\lambda = 1/18, 1/9, 3/18$. Choosing $\lambda = \frac{\#_\Omega}{\#_{\overline{\Omega}}} = \frac{1}{9}$ is thus a reasonable choice ($\#_\Omega$ is the cardinality of Ω, i.e. the number of elements in Ω). Note that using $\lambda = 1$ has similarities with BSA-IC using the S-HALS algorithm presented in the previous section. Indeed, when $\lambda = 1$, it is not necessary to split the problem in two sums $\sum_{(i,j)\in\Omega}$ and $\sum_{(i,j)\in\overline{\Omega}}$, but we can sum on all (i,j) instead, using matrix $\overline{\boldsymbol{W}}$:

$$\overline{\boldsymbol{W}}_{i,j} = \begin{cases} \boldsymbol{W}_{i,j} & \text{if } (i,j) \in \Omega \\ \boldsymbol{Y}_{i,j} & \text{else} \end{cases}$$

This simplifies and accelerates the computation a lot. However, in the presented test we did not use this acceleration. As this special case turns out to be much less accurate compared to the others, we decided to not improve our implementation.

On the other hand, based on the results for 2BSA-IC, we considered to use also a λ in the BSA-IC Algorithm (7.1), but results were not convincing in this case. Moreover, algorithms where much slower than when not using λ. Those results are therefore not presented in what follows.

7.3 Experiments

Methods to perform image completion are compared on two criteria: the needed time and the signal to interference ratio (SIR). The needed time should be as low as possible, while the SIR computes the similarity between the obtained image and the original (ground truth) image (before removing pixels). It should therefore be as high as possible. This measure has been presented in Equation (2.4). We use cubic B-Splines as basis for splines. The tests presented below are performed on a 512×512 boat image (see Figure 7.6) with 90% of missing pixels. The pixels are removed randomly.

Choosing the number of B-splines to be used is an important component in all the proposed algorithms: one can either choose a fixed number (50 or 100), or perform a run with some number of splines and use its final image as input for a subsequent run with more splines (we tried $50 \to 100$ and $25 \to 50 \to 100$). Two more types of runs are reported: in an 'inner' run, as suggested in previous work [106], the number of B-splines is incremented

progressively from 3 up to 100 during the update of matrices $\boldsymbol{A}$ and $\boldsymbol{X}$ [140]. However, we did not find that tweak, where the increase happens during each iteration, to be very effective. Instead we designed an 'outer' run, similar in spirit to the successive $50 \to 100$ and $25 \to 50 \to 100$ runs, where a different number of B-Splines is used at each iteration: we use $\min(3 \cdot i + 10, 100)$ splines for the ith iteration (increasing the number of splines by 3 at each iteration prevents too quick stagnation). All tests are performed with a rank $r = 50$.

Figure 7.2 (left) establishes that our methods (Algorithm 7.1 with S-HALS, and 2BSA-IC) are both more efficient than Algorithm 7.1 with ADMM (more than four times faster). The signal to interference ratio (SIR) is on average higher for S-HALS than for the other methods, even though ADMM is competitive in terms of accuracy when the number of splines is increased progressively. Increasing progressively the number of splines is a good idea and provide the most accurate results for all methods, and in particular the 'outer' increase of splines is the best. An explanation for this improvement could be that the image is first reconstructed from its low-rank elements, and then improved [134].

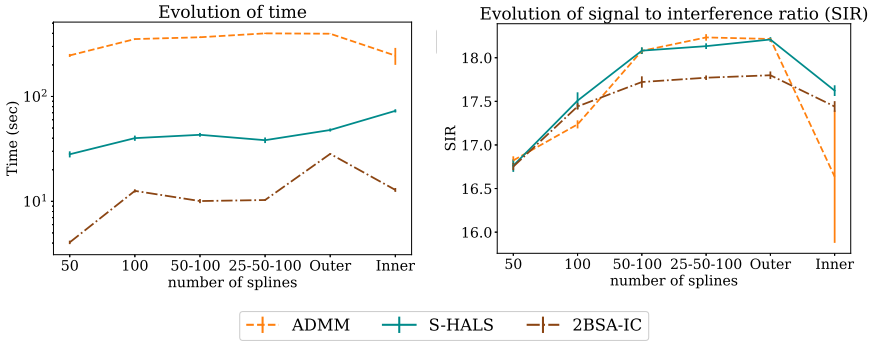


Fig. 7.2 Performance of the methods with varying number of B-Splines. The y-axis for time is in logarithmic scale.

We also analyze the methods with different choices of factorization rank r, using $r = 25$, $r = 50$ or $r = 100$. We include in the comparison a case where r is increased at each iteration: $r = 25 + 5 \cdot i$ (Evolutive), inspired from [134]. All methods are now tested with an 'outer' increment of the number of splines, as it is the most efficient. We observe in Figure 7.3 that

all methods obtain better results with higher rank, but at the cost of more computational effort. For example, for ADMM, the benefits for $r = 100$ compared to $r = 50$ is small compared to the increase in time. Moreover, using an evolutive rank is not interesting as it is not significantly better nor faster than using a constant rank

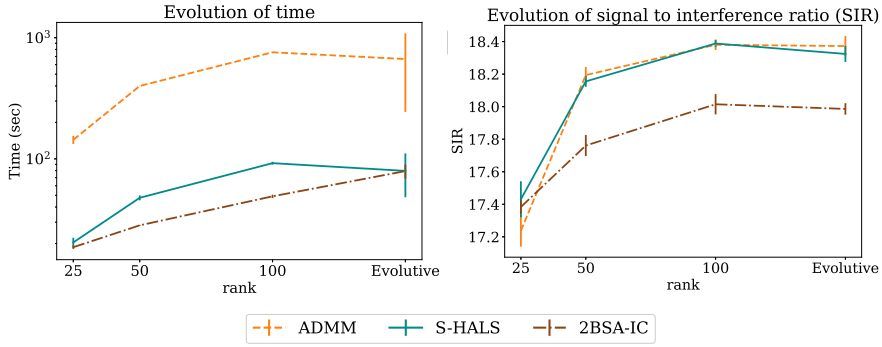


Fig. 7.3 Performance of the methods with varying rank. The y-axis for time is in logarithmic scale.

Let us now observe the behavior of the algorithms when they are many missing pixels. The methods are tested with rank $r = 100$ and an 'outer' increment in the number of splines. We observe in Figure 7.4 that increasing the number of missing pixels decreases the accuracy of the algorithms, which is quite logical. However, the degradation is rather small and linear when less than 90% of pixels are missing, while the degradation becomes exponential when more than 90% of pixels are missing, the problem becoming much more difficult in this case. For example, when 99% of pixels are missing, Algorithm 7.1 using both S-HALS and ADMM is highly degraded, and the recovered SIR is very low (only 6dB), while the needed time increases significantly. However, the performance of 2BSA-IC is much less degraded: the needed time does not increase, and the SIR stays around 13-14 dB. This is illustrated in Figure 7.7.

Let us now observe what happens when noise is added to the data (i.e. the known points of the input image). The input image has 90% of missing pixels, and the methods are tested with rank $r = 100$ and an 'outer' increment of the number of splines. We observe in Figure 7.5 that when the noise is not too high (SNR above 40 dB), the performances are not much

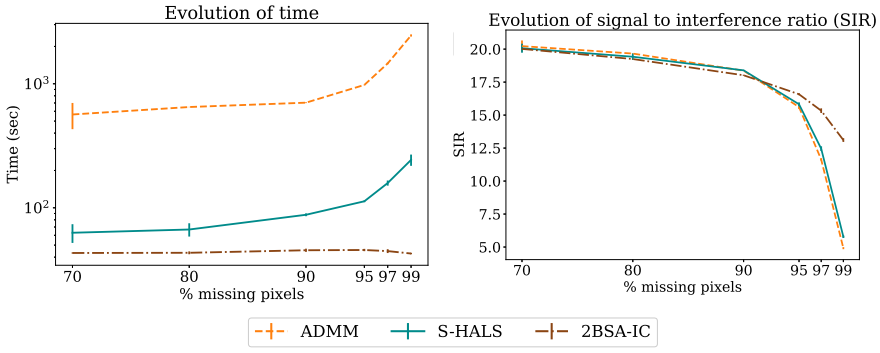


Fig. 7.4 Performance of the methods when the number of missing pixels increase. The y-axis for time is in logarithmic scale.

impacted. When the noise level increases to a SNR of 20dB, the performances are degraded a bit in terms of accuracy, and they are even more degraded when noise level increase to an SNR of 10dB. Note however that 2BSA-IC is less impacted than the other methods and becomes better than them in this case. From this case and the previous one (with the increasing number of missing pixels), we can conclude that the 2BSA-IC method is the most robust, even though it is a bit less accurate than the BSA-IC Algorithm when only 90% of pixels are missing.

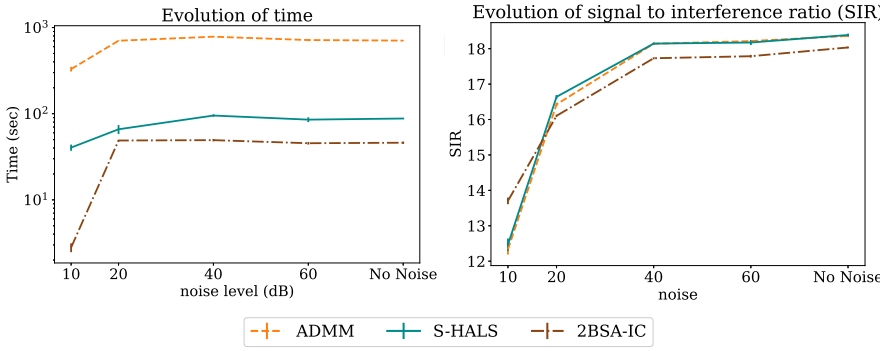


Fig. 7.5 Performance of the methods for different levels of noise. The y-axis for time is in logarithmic scale.

To illustrate the results obtained in this section, Figure 7.6 displays sev-

eral examples of recovered images with 90% of missing pixels, with rank fixed to 100 for the 512×512 grayscale image and 50 for the 256×256 color images. For color images, algorithms were applied on each color channel independently. This is not optimal as the color channels are not really independent. Extending our approaches to tensors or to multivariate splines could therefore be a way to deal better with color images, which we leave as interesting future work. All algorithms use an 'outer' run with a maximal number of splines equal to 100. We observe on all three images that S-HALS is significantly faster and slightly more accurate than ADMM, while 2BSA-IC is the fastest method but also the least accurate, although its accuracy is close to that of the other two methods

Figure 7.7 displays several examples of recovered images with 99 % missing pixels. The other parameters are the same. We observe that all images are significantly degraded when compared to Figure 7.6, but the images obtained by 2BSA-IC are this time by far the best and this method is significantly faster than the others. This illustrates well that 2BSA-IC is more robust than BSA-IC. Both images also illustrate that S-HALS is more appropriate than ADMM for BSA-IC as it is faster and slightly more accurate.

Some ideas for further improvement To improve the methods, one idea is to impose the sparsity of the factors in addition to their smoothness, imposed thanks to the splines. One could thus consider splines with few non-zero coefficients. This could be done either by adding a regularization term in the minimized cost functions, or during the projection. Indeed, rather than simply setting all the negative coefficients of splines to zero, one could consider keeping only the significant coefficients (either a small number of coefficients, or only the coefficients above a certain value).

Alternatively, one could consider rational functions rather than splines in the NMF. Indeed, as presented in Chapter 8, rational functions are smooth and with some peaks, characteristics shared by images, which are generally smooth except at the edges of the objects. We leave the testing of these two ideas for further research.

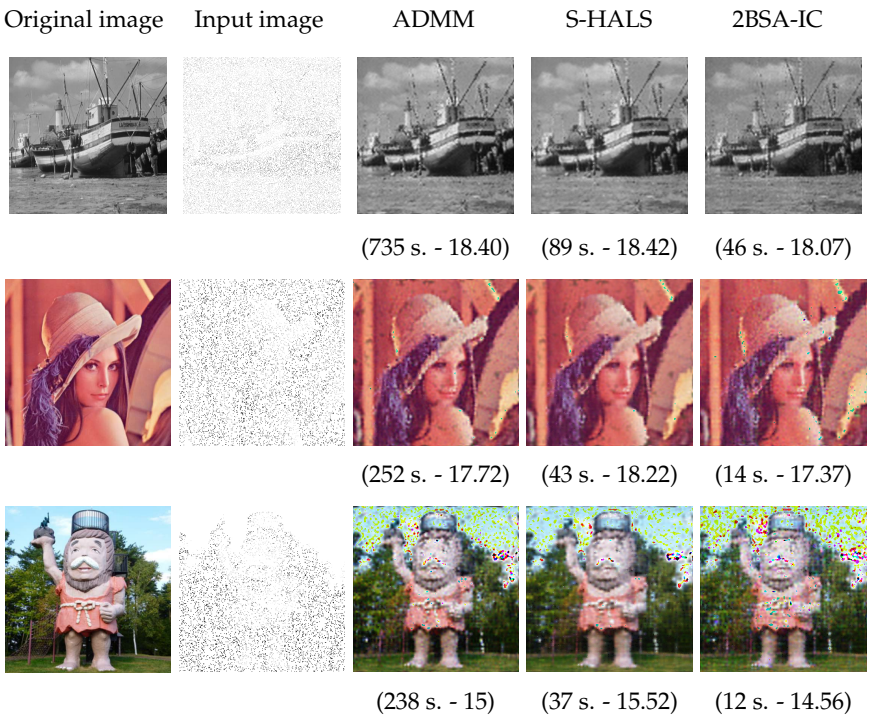


Fig. 7.6 Example of recovered images for 90% of missing pixels. (CPU time - SIR). The black pixels in input image are replaced by white pixels in order to better handle the given information.

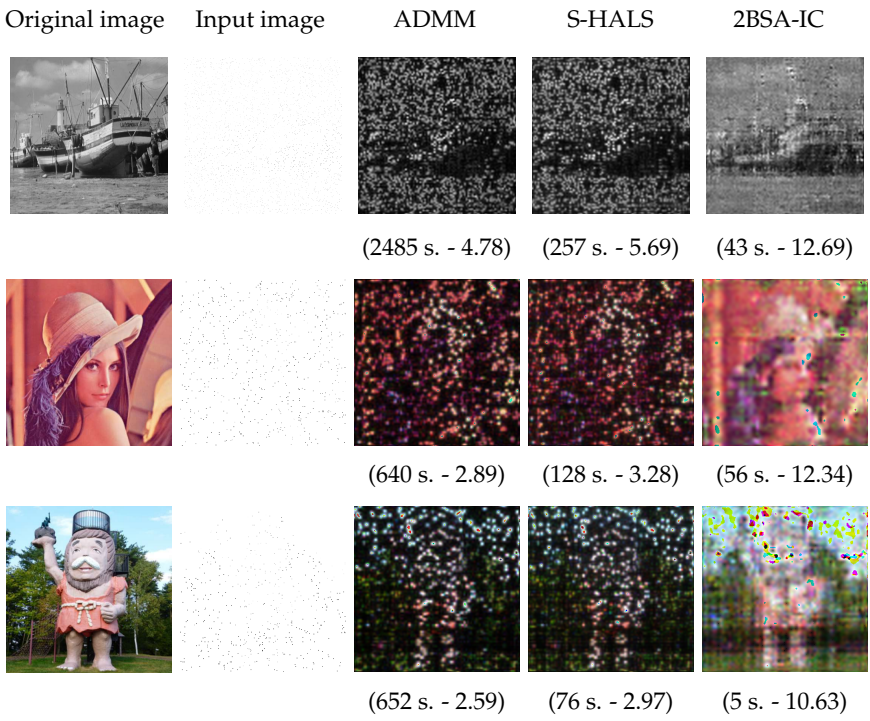


Fig. 7.7 Example of recovered images for 99% of missing pixels. (CPU time - SIR). The black pixels in input image are replaced by white pixels in order to better handle the given information

8

NMF using rational functions

We observed in previous chapters that using polynomials or splines as factors $\boldsymbol{A}$ and/or $\boldsymbol{X}$ in NMF may improve the quality of the factorization and allow it to be less sensitive to noise. When the columns of the input matrix $\boldsymbol{Y}$ are samples of nonnegative continuous signals, mostly smooth with possibly some peaks, it makes sense to consider that they are samples of nonnegative 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 [64].

NMF over rational functions, named R-NMF, is presented in Section 8.1. In Section 8.2, 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 behind data, as in blind source separation.

Unlike for nonnegative polynomials or splines, the set of nonnegative rational functions of fixed degree is not convex, and the projection on it is not easy to compute, to use the H-HALS algorithm. Therefore, we explore in Section 8.4 several 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, i.e. lead to

better factorizations.

One of the main advantages of HALS for standard NMF is the simplicity of its iterations, allowing for a quick resolution of the problem. However, when using rational functions, each iteration is difficult due to the projection. It is thus questionable whether this approach is suitable for this problem. Therefore, we consider other block-coordinate descent (BCD) approaches involving fewer blocks in Section 8.3, the R-ANLS and R-NLS methods. Those approaches solve at each iteration larger problems closer to the original problem. Methods are then analyzed and compared in Section 8.5, where we find that R-HANLS is suited for large-scale data and/or to obtain quickly a good factorization, while R-ANLS can obtain a more accurate factorization but is slower and more resource-demanding. R-NLS can only be used for very small datasets.

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.

8.1 The NMF using rational functions (R-NMF) problem

Consider an input data matrix $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$, 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 taken equidistantly. 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 $\boldsymbol{Y}$, i.e. $\boldsymbol{Y}_{:j}$ for $j = 1, 2, \ldots, n$, as a nonnegative linear combination of r functions in $\mathcal{F}^{\boldsymbol{d},T}$

$$\boldsymbol{Y}_{:j} \simeq \sum_{k=1}^{r} A_k(\boldsymbol{\tau}) \boldsymbol{X}_{jk} \; \forall j. \qquad A_k \in \mathcal{F}_+^{\boldsymbol{d},T} \; \forall j \; , \boldsymbol{X} \in \mathbb{R}_+^{n \times r} \tag{8.1}$$

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 $\boldsymbol{Y}$ as objective function.

Definition 8.1 (R-NMF). Given an input matrix $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$, 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 $\boldsymbol{A} \in \mathbb{R}_+^{m \times r}$ containing elements of $\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ in each of its columns, i.e. $\boldsymbol{A}_{:k} \in \mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ $\forall k$, and a nonnegative matrix $\boldsymbol{X} \in \mathbb{R}_+^{n \times r}$ solving

$$\min_{\boldsymbol{A}_{:k} \in \mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}, \boldsymbol{X} \in \mathbb{R}_+^{n \times r}} \sum_{j=1}^{n} \left\| \boldsymbol{Y}_{:j} - \sum_{k=1}^{r} \boldsymbol{A}_{:k} \boldsymbol{X}_{jk} \right\|_F^2. \tag{8.2}$$

The choice of rational functions is motivated by their ability to represent a large range of shapes and their utility in applications; in particular they generalize polynomials or splines [120], and they represent the natural way of modeling linear dynamical systems in the frequency domain [64].

A rational function is defined as the ratio of two polynomials: $f(x) = \frac{h(x)}{g(x)}$. Throughout this work, we consider univariate rational functions with fixed degree $\boldsymbol{d} = (d_1, d_2)$, such 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 (for example 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 [68], and nonnegative polynomials can be parametrized using sums of squares [101]. 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 Section 2.3.3. Consider for example the case of a rational function of degree $\boldsymbol{d} = (2d_1', 2d_2')$ nonnegative on $[-1, 1]$. Let $\boldsymbol{V}_{\boldsymbol{\tau}}^{d}$ be the Vandermonde-like matrix for the chosen basis of polynomials (in our case, the Chebyshev basis). Using the coefficients $(\boldsymbol{h_1}, \boldsymbol{h_2}, \boldsymbol{g_1}, \boldsymbol{g_2}) \in \mathbb{R}^{d_1'+1} \times \mathbb{R}^{d_1'} \times \mathbb{R}^{d_2'+1} \times \mathbb{R}^{d_2'}$ the evaluation of this function on points $\boldsymbol{\tau}$ can be written as (the multiplication '$\cdot$' and the division are performed element-wise):

$$f_{\boldsymbol{\tau}}(\boldsymbol{h_1}, \boldsymbol{h_2}, \boldsymbol{g_1}, \boldsymbol{g_2}) = \frac{(\boldsymbol{V}_{\boldsymbol{\tau}}^{d_1'} \boldsymbol{h_1})^2 + (1 - \boldsymbol{\tau}^2) \cdot (\boldsymbol{V}_{\boldsymbol{\tau}}^{d_1'-1} \boldsymbol{h_2})^2}{(\boldsymbol{V}_{\boldsymbol{\tau}}^{d_2'} \boldsymbol{g_1})^2 + (1 - \boldsymbol{\tau}^2) \cdot (\boldsymbol{V}_{\boldsymbol{\tau}}^{g_2'-1} \boldsymbol{g_2})^2 + \epsilon}. \tag{8.3}$$

However, this representation is redundant, as multiplying the numerator and the denominator by the same constant leads to the same rational func-

tion with other coefficients. Therefore, we impose the denominator g to be monic using $(\boldsymbol{g}_1)_{d_2'+1} = \frac{1}{2}\sqrt{8 + (\boldsymbol{g}_2)_{d_2'}^2}$ (see Equation (2.26)).

8.2 Uniqueness

In this section, we focus on exact factorizations $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}^\top$. In such a factorization, if the column $\boldsymbol{A}_{:k}$ is scaled by a factor α_k while the column $\boldsymbol{X}_{:k}$ is scaled by factor $\frac{1}{\alpha_k}$, $\boldsymbol{A}\boldsymbol{X}^\top$ remains unchanged, as it is the sum of the same rank-one terms. Moreover, applying the same permutation to the columns of $\boldsymbol{A}$ and $\boldsymbol{X}$ also keeps $\boldsymbol{A}\boldsymbol{X}^\top$ unchanged. This defines essentially unique factorizations.

Definition 8.2. $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}^\top$ is said to have an **essentially unique factorization** if all the factorizations of $\boldsymbol{Y}$ can be obtained only using consistent permutations and scalings/counterscaling of columns of $\boldsymbol{A}/\boldsymbol{X}$.

As shown in Lemma 8.1, a matrix $\boldsymbol{Y}$ with factorization $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}^\top$ admits an infinite number of other factorizations not resulting from permutations and scalings. To have an essentially unique factorization, we must thus add constraints on $\boldsymbol{A}$ and/or $\boldsymbol{X}$. In NMF, the factors $\boldsymbol{A}$ and $\boldsymbol{X}$ are nonnegative. This constraint restricts the possibilities of equivalent factorizations and allows, under certain conditions, for an essentially unique factorization. However, these conditions are quite restrictive, and are not met in general, see [43] and [46, Chap. 4] and the references therein.

Lemma 8.1. *Let $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}^\top$, with $\boldsymbol{A} \in \mathbb{R}^{m\times r}$, $\boldsymbol{X} \in \mathbb{R}^{n\times r}$, and $\operatorname{rank}(\boldsymbol{Y}) = r$. Matrices $\boldsymbol{A}' \in \mathbb{R}^{m\times r}$ and $\boldsymbol{X}' \in \mathbb{R}^{n\times r}$ factorize $\boldsymbol{Y}$ if and only if $\boldsymbol{A}' = \boldsymbol{A}\boldsymbol{Q}$ and $\boldsymbol{X}'^\top = \boldsymbol{Q}^{-1}\boldsymbol{X}^\top$ where $\boldsymbol{Q} \in \mathbb{R}^{r\times r}$ is an invertible matrix.*

Proof. For all invertible matrices $\boldsymbol{Q} \in \mathbb{R}^{r\times r}$, $\boldsymbol{A}' = \boldsymbol{A}\boldsymbol{Q}$ and $\boldsymbol{X}'^\top = \boldsymbol{Q}^{-1}\boldsymbol{X}^\top$ have has product $\boldsymbol{Y}$. On the other hand, suppose $\boldsymbol{A}'\boldsymbol{X}'^\top = \boldsymbol{A}\boldsymbol{X}^\top$. As $\operatorname{rank}(\boldsymbol{Y}) = r$, matrices $\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{A}'$ and $\boldsymbol{X}'$ must be rank r as well, and matrices $\boldsymbol{A}'^\top\boldsymbol{A}'$ and $\boldsymbol{X}'^\top\boldsymbol{X}'$ are invertible. We have

$$\boldsymbol{A}'\boldsymbol{X}'^\top = \boldsymbol{A}\boldsymbol{X}^\top \Rightarrow \boldsymbol{X}'^\top = (\boldsymbol{A}'^\top\boldsymbol{A}')^{-1}\boldsymbol{A}'^\top\boldsymbol{A}\boldsymbol{X}^\top = \boldsymbol{Q}_A\boldsymbol{X}^\top$$
$$\text{and } \boldsymbol{A}'\boldsymbol{X}'^\top = \boldsymbol{A}\boldsymbol{X}^\top \Rightarrow \boldsymbol{A}' = \boldsymbol{A}\boldsymbol{X}^\top\boldsymbol{X}'(\boldsymbol{X}'^\top\boldsymbol{X}')^{-1} = \boldsymbol{A}\boldsymbol{Q}_X.$$

As $\boldsymbol{A}'$ and $\boldsymbol{A}$ have rank r, $\boldsymbol{Q}_X$ must be invertible. We have

$$\boldsymbol{Q}_X^{-1} = \boldsymbol{Q}_A \Leftrightarrow \boldsymbol{A}'^{\top}\boldsymbol{A}'\boldsymbol{X}'^{\top}\boldsymbol{X}' = \boldsymbol{A}'^{\top}\boldsymbol{A}\boldsymbol{X}^{\top}\boldsymbol{X}'$$

And therefore $\boldsymbol{A}' = \boldsymbol{A}\boldsymbol{Q}_X$ and $\boldsymbol{X}'^{\top} = \boldsymbol{Q}_X^{-1}\boldsymbol{X}^{\top}$. □

If we consider that the columns of matrix $\boldsymbol{A}$ are samples of rational functions, it is possible to prove that the product $\boldsymbol{A}\boldsymbol{X}^{\top}$ is essentially unique under certain conditions on the rational functions contained in the columns of $\boldsymbol{A}$ [32]. Indeed, at most one column can contain a polynomial and the poles of all rational functions must be distinct. The number of discretization points must also be greater than twice the sum of the degrees of the rational functions in $\boldsymbol{A}$, for example $m > 2r(d_1 + d_2)$ in R-NMF.

In R-NMF, the considered rational functions must be nonnegatives and of the same degrees. The exact R-NMF problem described below is thus a special case of [32]. Theorem 8.1 shows that it is possible to ensure that exact R-NMF is essentially unique with milder conditions on $\boldsymbol{A}$.

Definition 8.3. Exact R-NMF Given $\boldsymbol{Y} \in \mathbb{R}_+^{m\times n}$, $\boldsymbol{\tau}$, $\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ and r as in R-NMF, compute, if possible, $\boldsymbol{A} \in \mathbb{R}_+^{m\times r}$ with $\boldsymbol{A}_{:k} \in \mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ for all k and $\boldsymbol{X} \in \mathbb{R}_+^{n\times r}$ such that $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}^{\top}$.

Let us introduce some useful lemmas and notations. A rational function $f(x)$ of degree $\boldsymbol{d} = (d_1, d_2)$ can be written as follows, with $\alpha \neq 0$:

$$f(x) = \frac{\alpha \prod_{i=1}^{d_1}(x - z_i)}{\prod_{j=1}^{d_2}(x - p_j)} \quad z_i, p_j \in \mathbb{C}, z_i \neq p_j \; \forall i, j, \tag{8.4}$$

with $\mathcal{Z} = \{z_i\}_{i=1}^{d_1}$ the zeros of $f(x)$, 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. 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, i.e. poles appearing in exactly one function is $\{p_3\}$.

Lemma 8.2. *Let $\{f_l\}_{l=1}^r$ be a collection of rational functions in the form (8.4), 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, i.e. 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. Let $\mathcal{U}$ be the set of all poles in $\{f_l\}_{l=1}^r$. The function f can be written as:

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

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 $x = s_k$, only the l^{th} 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 $x = s_k$ 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 8.3. *Let $\{f_l\}_{l=1}^r$ be a collection of r rational functions in form (8.4), of degree $\boldsymbol{d} = (d_1, d_2)$, and $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^m$ be a set of distinct discretization points with $m > d_1 + rd_2$, such that the denominators of functions f_l do not cancel at these points. If there exist a rational function f^* of degree $\boldsymbol{d}$ such 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

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

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

$$\Leftrightarrow \Big(\tilde{\alpha} \prod_{i=1}^{d_1}(\boldsymbol{\tau} - \tilde{z}_i)\Big)\Big(\prod_{l=1}^r \prod_{j=1}^{d_2}(\boldsymbol{\tau} - p_{lj})\Big) = \tag{8.5}$$

$$\Big(\prod_{j=1}^{d_2}(\boldsymbol{\tau} - \tilde{p}_j)\Big)\Big(\sum_{l=1}^r \beta_l \alpha_l \prod_{i=1}^{d_1}(\boldsymbol{\tau} - z_{li}) \prod_{k \neq l}^r \prod_{j=1}^{d_2}(\boldsymbol{\tau} - p_{kj})\Big) \tag{8.6}$$

Elements (8.5) and (8.6) 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 $\boldsymbol{A}$ and $\boldsymbol{X}$ that imply that the exact R-NMF $\boldsymbol{A}\boldsymbol{X}^\top$ is essentially unique.

Theorem 8.1. *Let $\boldsymbol{A} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{X} \in \mathbb{R}^{n\times r}$ be of rank r. Suppose all columns of $\boldsymbol{A}$ 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, they are at least $d_2 + 1$ unique poles, i.e. poles appearing in exactly one function. Then the exact R-NMF $\boldsymbol{A}\boldsymbol{X}^\top$ is essentially unique.*

Proof. Let $\boldsymbol{A}', \boldsymbol{X}'$ be such that $\boldsymbol{A}'\boldsymbol{X}'^\top = \boldsymbol{A}\boldsymbol{X}^\top$. As $\boldsymbol{A}, \boldsymbol{X}$ are of rank r, we know by Lemma 8.1 that each column $\boldsymbol{A}'_{:j}$ can be written as a linear combination of the columns of $\boldsymbol{A}$: $\boldsymbol{A}'_{:j} = \sum_{l=1}^{r} \beta_l \boldsymbol{A}_{:l} = \sum_{l=1}^{r} \beta_l A_l(\boldsymbol{\tau})$. To be valid, $\boldsymbol{A}'_{:j}$ 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 (8.3), A'_j must be the linear combination of the rational functions in $\boldsymbol{A}$: $A'_j = \sum_l \beta_l A_l$.

To avoid the trivial case of permutation and scaling, there must be at least one $\boldsymbol{A}'_{:j}$ that is the combination of two or more columns of $\boldsymbol{A}$. As all sets $\{A_j\}$ containing two functions or more have at least $d_2 + 1$ unique poles, using Lemma 8.2 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 $\boldsymbol{A}'$ such that $\boldsymbol{A}'\boldsymbol{X}'^\top = \boldsymbol{A}\boldsymbol{X}^\top$ and the factorization $\boldsymbol{A}\boldsymbol{X}^\top$ is essentially unique. □

Corollary 8.2. *Let $\boldsymbol{A} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{X} \in \mathbb{R}^{n\times r}$ be of rank r, with the columns of $\boldsymbol{A}$ obtained through evaluation of rational functions of degree (d_1, d_2) on m distinct points $\boldsymbol{\tau} \in \mathbb{R}^m$, with $m > d_1 + rd_2$, and $\boldsymbol{\tau}$ not containing poles of functions in $\boldsymbol{A}$. 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 $\boldsymbol{A}\boldsymbol{X}^\top$ is essentially unique.*

Note that the nonnegativity constraint is not necessary for Theorem 8.1 and Corollary 8.2 to hold. Nevertheless, when using a representation like (8.3), functions A_j does not have real poles on interval $T = [\boldsymbol{\tau}_{\min}, \boldsymbol{\tau}_{\max}]$,

thanks to the ϵ added to the denominator. This means that in this case the condition "$\boldsymbol{\tau}$ not containing poles of functions in $\boldsymbol{A}$" is always met.

8.3 Algorithms for R-NMF

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

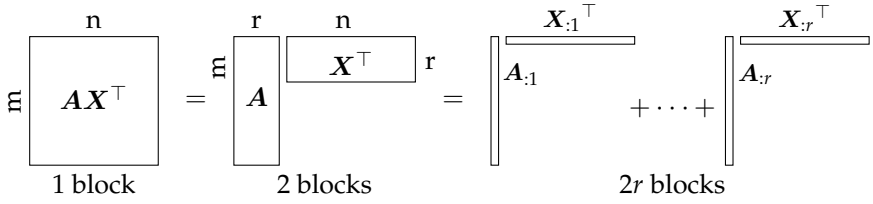


Fig. 8.1 Illustration of the three block-decomposition.

8.3.1 General Nonlinear Least Squares approach (R-NLS)

We substitute in (8.2) $\boldsymbol{A}_{:k}$ by $f_{\boldsymbol{\tau}_k}$ from Equation (8.3), and $\boldsymbol{X}_{jk}$ by $\boldsymbol{C}^2_{jk}$ to express the R-NMF problem in an unconstrained way:

Algorithm 8.1 Nonlinear Least Squares

function R-NLS($\boldsymbol{Y}$)

$$\boldsymbol{A}, \boldsymbol{X} \leftarrow \underset{\substack{\boldsymbol{h_{1k}}, \boldsymbol{h_{2k}}, \\ \boldsymbol{g_{1k}}, \boldsymbol{g_{2k}}, \boldsymbol{C}}}{\operatorname{argmin}} \sum_{j=1}^{n} \left\| \boldsymbol{Y}_{:j} - \sum_{k=1}^{r} f_{\boldsymbol{\tau}_k}(\boldsymbol{h_{1k}}, \boldsymbol{h_{2k}}, \boldsymbol{g_{1k}}, \boldsymbol{g_{2k}}) \boldsymbol{C}^2_{jk} \right\|^2. \tag{8.7}$$

return $\boldsymbol{A}$, $\boldsymbol{X}$

This problem can be solved using a standard nonlinear least squares solver. The same approach for polynomials has been proposed in [33]. Note however that in the cited work a compression method is suggested to preprocess the data and reduce the complexity of the problem, but this is not possible in our case because rational function are not linearly parametriz-

able, that is, they cannot be described using a linear combination of some basis elements.

8.3.2 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, many NMF algorithms consider instead alternating schemes [27],[74],[80],[84]. The problem is then solved by alternating on $\boldsymbol{A}$ and $\boldsymbol{X}$ considering the other matrix as fixed, as sketched in Algorithm 8.2. As $f_{\boldsymbol{\tau}_k}$ is a nonlinear function, each sub-problem is nonlinear, and this method is called alternating nonlinear least squares.

Algorithm 8.2 Alternating Nonlinear Least Squares

function R-ANLS($\boldsymbol{Y}$, $\boldsymbol{A}$, $\boldsymbol{X}$)
while Stop condition not encountered **do**:

$$\boldsymbol{A} \leftarrow \underset{\substack{\boldsymbol{h_{1k}},\boldsymbol{h_{2k}},\\ \boldsymbol{g_{1k}},\boldsymbol{g_{2k}}}}{\operatorname{argmin}} \sum_{j=1}^{n} \left\| \boldsymbol{Y}_{:j} - \sum_{k=1}^{r} f_{\boldsymbol{\tau}_k}(\boldsymbol{h_{1k}}, \boldsymbol{h_{2k}}, \boldsymbol{g_{1k}}, \boldsymbol{g_{2k}}) \boldsymbol{X}_{jk} \right\|^2 \tag{8.8}$$

$$\boldsymbol{X} \leftarrow \left(\underset{\boldsymbol{C} \in \mathbb{R}^{n \times r}}{\operatorname{argmin}} \sum_{j=1}^{n} \left\| \boldsymbol{Y}_{:j} - \sum_{k=1}^{r} \boldsymbol{A}_{:k} \boldsymbol{C}_{jk}^2 \right\|^2 \right)^2 \tag{8.9}$$

return $\boldsymbol{A}$, $\boldsymbol{X}$

Problems (8.8) and (8.9) are unconstrained and can be solved using a standard nonlinear least squares solver. Note that problem (8.9) is separable in n independent sub-problems, as the rows of $\boldsymbol{X}$, are independent (which is not the case for the rows of $\boldsymbol{A}$):

$$\boldsymbol{X}_{j:} \leftarrow \left(\underset{\boldsymbol{C}_{j:} \in \mathbb{R}^{r}}{\operatorname{argmin}} \left\| \boldsymbol{Y}_{:j} - \sum_{k=1}^{r} \boldsymbol{A}_{:k} \boldsymbol{C}_{jk}^2 \right\|^2 \right)^2 \quad \forall j \in \{1, \cdots, n\}.$$

8.3.3 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 $\boldsymbol{A}/\boldsymbol{X}$ are updated successively, considering all the other elements as fixed [27]; see also [47]. Because of the quadratic structure of the objective function, minimizing (8.2) when all variables are fixed except a column of $\boldsymbol{A}$ or $\boldsymbol{X}$ can be done by projecting the unconstrained minimizer on the corresponding feasible region (see Section 5.2 and in particular Lemma 5.2). This region is the set $\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}$ of nonnegative rational functions with fixed degrees (for $\boldsymbol{A}$), or the set $\mathbb{R}_+^n$ of nonnegative vectors (for $\boldsymbol{X}$).

The unconstrained minimizer can easily be found for columns of $\boldsymbol{A}$ and $\boldsymbol{X}$ by canceling the gradient. Algorithm 8.3 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 (8.11) is separable: the value of $\boldsymbol{X}_{ik}$ can be computed independently from $\boldsymbol{X}_{jk}$, but this is not the case for $\boldsymbol{A}_{:k}$ in Equation (8.10), as the projection is not separable unlike the thresholding operation.

Algorithm 8.3 Hierarchical Alternating Nonlinear Least Squares

function R-HANLS($\boldsymbol{Y}$, $\boldsymbol{A}$, $\boldsymbol{X}$)
  while Stop condition not met **do**
    for $\boldsymbol{A}_{:k} \in \boldsymbol{A}$ **do**

$$\boldsymbol{A}_{:k} \leftarrow \left[\frac{\boldsymbol{Y}\boldsymbol{X}_{:k} - \sum_{s\neq k} \boldsymbol{A}_{:s}(\boldsymbol{X}_{:s})^\top \boldsymbol{X}_{:k}}{\|\boldsymbol{X}_{:k}\|^2}\right]_{\mathcal{R}_{\boldsymbol{\tau}}^{\boldsymbol{d},T}} \tag{8.10}$$

    for $\boldsymbol{X}_{:k} \in \boldsymbol{X}$ **do**

$$\boldsymbol{X}_{:k} \leftarrow \left[\frac{\boldsymbol{Y}^\top \boldsymbol{A}_{:k} - \sum_{s\neq k} \boldsymbol{X}_{:s}(\boldsymbol{A}_{:s})^\top \boldsymbol{A}_{:k}}{\|\boldsymbol{A}_{:k}\|^2}\right]_{\mathbb{R}_+^n} \tag{8.11}$$

  return $\boldsymbol{A}$, $\boldsymbol{X}$

8.4 Projection on nonnegative rational functions

As mentioned in Section 8.1, rational functions nonnegative on a fixed interval T can be described as a 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 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 $z(x)$ 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:

$$\min_{h \in \mathcal{P}^{d_1}_+(T), g \in \mathcal{P}^{d_2}_{++}(T)} \left\| \boldsymbol{z} - \frac{h(\boldsymbol{\tau})}{g(\boldsymbol{\tau})} \right\|_2^2. \tag{8.12}$$

8.4.1 Existing approaches to approximate (nonnegative) rational functions

Solving problem (8.12) is not trivial, even when neglecting the nonnegativity constraints. If many works exist in the unconstrained case, most of them consider the infinity norm in (8.12) [121], and there are very few works imposing nonnegativity: to the best of our knowledge this problem is only addressed in [105, 113], 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

$$f(x) = \sum_{i=1}^{d} \frac{\omega_i z_i}{x - \alpha_i} \Bigg/ \sum_{i=1}^{d} \frac{\omega_i}{x - \alpha_i}. \tag{8.13}$$

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 [42], or Froissart doublets [94]. Moreover, when $x \to \alpha_i$, then $f(x) \to z_i$, which allows one to optimize only the ω_i. The most common method using this representation is the adaptive Antoulas–Anderson method (AAA) [94]. This method gradually increases the size of the basis by judiciously choosing the α_i points to be added. It does not seek to optimize a particular norm, but is a good initialization for future optimization [29, 42, 62, 73]. On the other hand, even if it is not presented as such, one can see Vector Fitting as using the same representation. In this method, the whole basis is chosen at once. Then one optimizes iteratively, using at each iteration the poles of the denominator found at previous iteration as new basis [50].

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.

In general, many methods try to get rid of the denominator which is difficult to optimize. Thus, [109] but also [87] and [125] have proposed to solve the problem iteratively, using a guess of the denominator, g^{k-1}, improved throughout iterations, by solving

$$(g^k, h^k) = \underset{g\in\mathcal{P}^{d_2}, h\in\mathcal{P}^{d_1}}{\operatorname{argmin}} \left\| \frac{\boldsymbol{z}g(\boldsymbol{\tau}) - h(\boldsymbol{\tau})}{g^{k-1}(\boldsymbol{\tau})} \right\|. \tag{8.14}$$

In the same idea, a special case of the RKFIT algorithm from [12] focuses on finding a good denominator by solving the following problem iteratively:

$$\min_{g^k\in\mathcal{P}^{d_2}} \left\| \frac{\boldsymbol{z}g^k - h'(g^k; \boldsymbol{z}, g^{k-1})}{g^{k-1}} \right\|, \tag{8.15}$$

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

When using the infinity norm in (8.12), if $g(\boldsymbol{\tau}_i)$ is positive for all i, the problem can be rewritten as:

$$\min_{h\in\mathcal{P}^{d_1}, g(\boldsymbol{\tau}_i)>0, u} u \quad \text{s.t.} \quad \begin{cases} \boldsymbol{z}_i g(\boldsymbol{\tau}_i) - h(\boldsymbol{\tau}_i) \leq u g(\boldsymbol{\tau}_i) \\ h(\boldsymbol{\tau}_i) - \boldsymbol{z}_i g(\boldsymbol{\tau}_i) \leq u g(\boldsymbol{\tau}_i) \end{cases}. \tag{8.16}$$

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 [105, 113] to solve the problem on nonnegative rational functions. The numerator and the denominator of the rational functions are modeled using Sum Of Squares (SOS), which makes problem (8.16) a SDP feasibility problem for u fixed.

Finally, using Equation (8.3), it is possible to see problem (8.12) as an unconstrained nonlinear least squares problem and to solve it using standard methods [121].

8.4.2 Proposed projection methods

Let us present five approaches to solve the projection problem on nonnegative rational functions. Some details of implementation are omitted and presented in Appendix B instead, to lighten the text.

Least Squares: Using Equation (8.3), the projection problem (8.12) can be rewritten in an unconstrained way and solved using a standard nonlinear least squares solver, as in R-NLS or R-ANLS.

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:

$$\underset{h \in \mathcal{P}_+^{d_1}(T)}{\operatorname{argmin}} \left\| z - \frac{h(\tau)}{g(\tau)} \right\|^2. \tag{8.17}$$

This problem is described in more details in Appendix B.

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 (8.3). As second problem has fewer variables than the original one, we can hope that it will be solved faster.

Algorithm 8.4 Alternating Least Squares

Input: z: signal to approximate, d_1, d_2: degree of the numerator/denominator, τ: discretization points, g: initial guess of the denominator, tol: tolerance of the algorithm

function ALTERNATING LS(z, d_1, d_2, τ, g, tol)
  while $\frac{\text{err}_{\text{prev}} - \text{err}}{\text{err}} >$ tol **do**
    $h =$ argmin(8.17)
    $g = \underset{g \in \mathcal{P}_{++}^{d_2}(T)}{\operatorname{argmin}} \|z - h(\tau)/g(\tau)\|^2$.
    $f(\tau) = h(\tau)/g(\tau)$
    $\text{err}_{\text{prev}} = \text{err}$, $\text{err} = \|z - f(\tau)\|^2$
  return $f(\tau)$

Conic: This method is inspired by Equation (8.14). From a given estimate of the denominator $\tilde{g}$, we aim to recover the rational function by optimiz-

ing a problem without variables at the denominator. The problem we aim to solve is not the same as in (8.14), and is motivated in Appendix B. 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

$$\underset{h\in\mathcal{P}^{d_1}_{+}(T),\delta\in\mathcal{P}^{d_2}_{+}(T)}{\operatorname{argmin}} \left\| \frac{\boldsymbol{z}\tilde{g}(\boldsymbol{\tau}) + \boldsymbol{z}\delta(\boldsymbol{\tau}) - h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} \right\|^2. \tag{8.18}$$

Note that the parametrization $f(\boldsymbol{\tau}) = \frac{h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})+\delta(\boldsymbol{\tau})}$ allows representing any rational function nonnegative on a fixed interval, and that the cost function of problem (8.18) is an upper bound of the cost function of problem (8.12). Moreover, if $\boldsymbol{z}$ is a nonnegative rational function of appropriate degrees, for any $\tilde{g}$ it is possible to find h and d such that cost function (8.18) 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 (8.18) and (8.12). Thus problem (8.18) 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 8.5.

Algorithm 8.5 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: $h, \delta = \operatorname{argmin}(8.18)$
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 [12]. The idea of this method is to first recover a good candidate for the denominator and then find the best numerator knowing the denominator.

This last problem is a convex problem on polynomials, and has been described for the Alternating Least Squares method in Equation (8.17).

To find a good denominator, we consider (8.18) 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 B for more details). This leads us to the following problem:

$$\operatorname*{argmin}_{\delta \in \mathcal{P}_+^{d_2}(T)} \left\| \boldsymbol{z} + \frac{\boldsymbol{z}\delta(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} - \frac{h'(\tilde{g}, \delta, \boldsymbol{z}, \boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} \right\|^2. \tag{8.19}$$

To find a good projection on the set of nonnegative rational functions we iterate over instances of problem (8.19). An iterative scheme is useful because problem (8.19) relies on the fixed parameter $\tilde{g}$. The pseudo-code of RKFIT+ is quite similar to the one of Conic (Algorithm 8.5). Line 7 is deleted, and lines 3 and 4 are replaced by $g = \operatorname{argmin}(8.19) + \tilde{g}$. Moreover, problem (8.17) is solved after the while loop to recover the numerator.

LinProj: This approach has been inspired from [105, 113]. 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 (8.16). These feasibility problems can even have linear constraints if we impose h and g to be nonnegative on points $\boldsymbol{\tau}_i \in \boldsymbol{\tau}$ instead of being nonnegative on interval T (this is different from what is done in [105, 113]). The feasibility problem is then:

$$\text{find}(h, g) \text{ s. t. } \begin{cases} \boldsymbol{z}_i g(\boldsymbol{\tau}_i) - h(\boldsymbol{\tau}_i) \leq u g(\boldsymbol{\tau}_i) \\ h(\boldsymbol{\tau}_i) - \boldsymbol{z}_i g(\boldsymbol{\tau}_i) \leq u g(\boldsymbol{\tau}_i) \quad \forall i, \\ h(\boldsymbol{\tau}_i) \geq 0;\ g(\boldsymbol{\tau}_i) \geq 1 \end{cases} \tag{8.20}$$

and a bisection algorithm is sketched in Algorithm 8.6. Note that g is prevented from containing values smaller than 1 at points $\boldsymbol{\tau}_i$ without loss of generality, to simplify the feasibility problem, preventing $[-u g(\boldsymbol{\tau}_i), u g(\boldsymbol{\tau}_i)]$ from being too small.

Algorithm 8.6 LinProj

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

function LINPROJ($z, d_1, d_2, \boldsymbol{\tau}$, 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 (8.20) on u_{med} is feasible **then**
      $u_{\max} = u_{\text{med}}$
    else
      $u_{\min} = u_{\text{med}}$
  Find h, g a feasible solution of (8.20) using $u_{\max}$
  return $\frac{h(\boldsymbol{\tau})}{g(\boldsymbol{\tau})}$

8.4.3 Comparison of the projection methods

We now compare the five proposed approaches to approximately project onto nonnegative rational functions. Algorithms have a tolerance `tol` of 10^{-8}. We consider two sets of inputs:

- The signals to project are the discretization of nonnegative 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 $[-1, 1]$. Figure 8.2 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 obtain more accurate results. Therefore, we do not consider Alternating Least

Squares in what follows. The 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 the Least Squares and the RK-FIT+ methods seem the more promising projection methods, but they are not always better than the others, and do not outperform them significantly.

8.5 Performance and Comparison of R-NMF algorithms

In this section, we briefly discuss the computational complexity of the proposed algorithms, before to compare the R-NMF algorithms presented in Section 8.3 on purely synthetic datasets to analyze their reconstruction ability and their efficiency. Then the most promising methods are compared to standard HALS and LP-HALS using polynomials or splines from Chapter 3 on semi-synthetic datasets. The least squares solver used for the experimentation is the function `least_squares` from python[1], with default parameters, using thus a trust region reflective algorithm [19].

Note that unlike LP-HALS using polynomials or splines, R-NMF approaches do not satisfy the hypothesis of Theorems 5.3 and 5.4, as the set of rational functions of fixed degree is not convex. Moreover, updates of R-NMF algorithms are not exact due this non convexity. Indeed, the minimization problem solved in R-NLS and R-ANLS are nonconvex, and there is no guarantee of convergence, and the same hold for the projection problem solved in R-HANLS. R-NMF methods have thus no theoretical guarantees. Nevertheless, we observe their actual behavior in practice in this section.

8.5.1 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. r is the rank of factorization, n is the

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

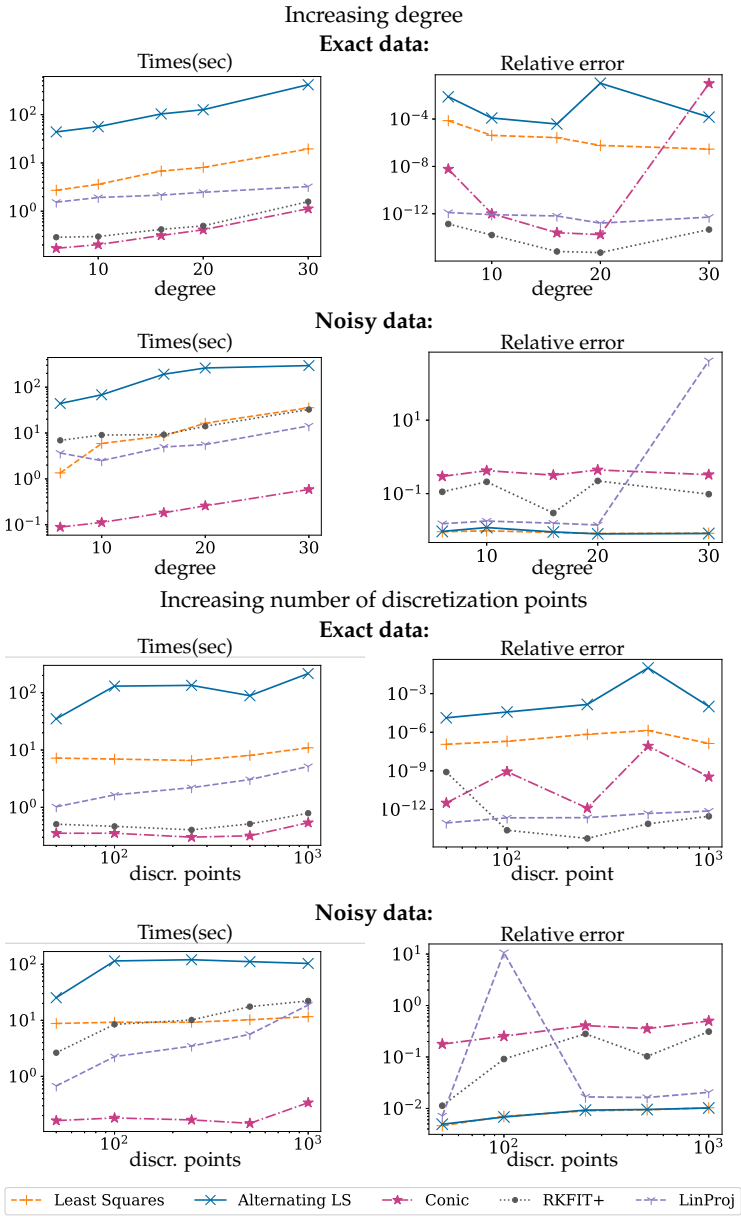


Fig. 8.2 Comparison of the projection methods. The results are averaged over 10 trials. The plots represents the time needed for computations (left) and the relative error (right).

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 $\boldsymbol{X}$ has complexity $\mathcal{O}(rmnI)$, where I is the number of iterations. The complexities of LP-HALS, R-HANLS using least-squares projection, R-ANLS and R-NLS can also be computed. Their value is summarized in Table 8.1. 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.

HALS	Poly/splines	R-HANLS
$\mathcal{O}(rmnI)$	$\mathcal{O}(rdnI + rp(d^2)I)$	$\mathcal{O}(rmnI + r\,ls(md)I)$

R-ALS	R-LS
$\mathcal{O}(rmnI + ls(rdmn)I)$	$\mathcal{O}(ls(rmn(n+d)))$

Table 8.1 Computational complexity of the various NMF methods.

8.5.2 Datasets

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

- a "purely synthetic" $\bar{\boldsymbol{A}}$ 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" $\bar{\boldsymbol{A}}$ whose columns are the real reflectance signals of Adulania, Clinochlore, Hypersthene, Olivine, Spessartine, Andesine,

Celestine and Kaolinite evaluated on 414 nonequally spaced points. These signals are showed in Figure 8.3 (left) and come from the U.S. Geological Survey (USGS) database [77]. Those 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 $\boldsymbol{A}^t$, $\boldsymbol{X}^t$ denote the factors obtained at iteration t. The stopping criterion of the algorithms is the following:

$$sc^t = \frac{\|\boldsymbol{Y} - \boldsymbol{A}^{t-1}\boldsymbol{X}^{t-1^\top}\| - \|\boldsymbol{Y} - \boldsymbol{A}^t\boldsymbol{X}^{t^\top}\|}{\|\boldsymbol{Y} - \boldsymbol{A}^t\boldsymbol{X}^{t^\top}\|} < 10^{-12}. \tag{8.21}$$

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 8.1, and selected to be not too important, while allowing the algorithm to converge in most cases, as we will see in the experiments. The quality of the factorization is evaluated using the relative residual and the SIR, presented in Section 2.1.1.

In what follows, each test is performed 10 times, using different initializations. To summarize the performance, 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 t if $\frac{sc^t - sc^o}{sc^t} < 10^{-3}$ for all $o \geq t$. This is used to evaluate the time needed by each algorithm, that it is the time needed to converge.

8.5.3 Initialization of the projections in R-HANLS

R-HANLS, described in Algorithm 8.3, is an iterative algorithm that uses a projection at each iteration. An important characteristic of this algorithm is

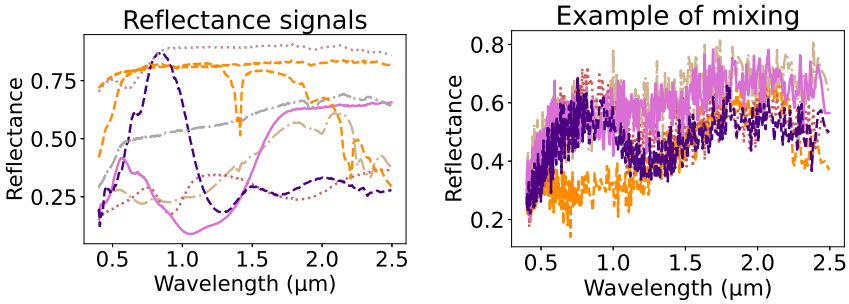


Fig. 8.3 Left: Considered real reflectance signals. Right: Example of mixing of those signals with noise level 20dB. Each of the five signals is a column of $\boldsymbol{Y}$.

that the successive iterates for the same block tend to become close to each other. Therefore, we should consider exploiting knowledge from previous iterations:

- 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 $u_{\max} = \max_i\{|\boldsymbol{z}_i - f_{\text{prev}}(\boldsymbol{\tau}_i)|\}$.

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 allows us to obtain accurate results in a reasonable time.

Figure 8.4 compares all projection methods in HALS, without using information about previous projection (dotted lines) or using this information (straight lines). We used a semi-synthetic dataset with $r = 3$ signals, and $n = 100$ observations, that does not contain noise. We observe that using information about previous projections is mostly interesting for the Least Squares projection. Indeed, in this case, it accelerates significantly the convergence of the algorithm and reduce its oscillations. Otherwise, the interest is lower, but it still makes sense to use information about previous iterations.

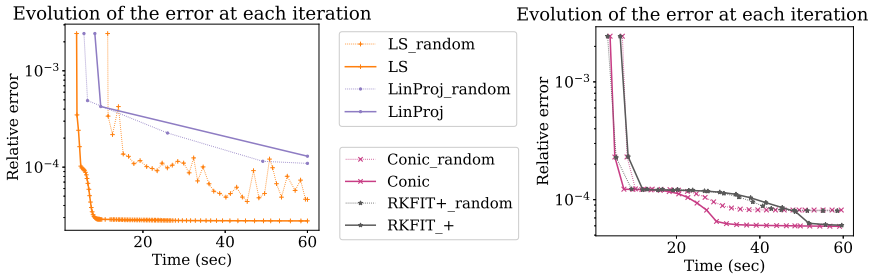


Fig. 8.4 Evolution of the relative error with respect to time, using previous iterates as initialization or not (Random case). Left figure presents results for Least Squares and LinProj projection. Right figure presents results for Conic and RKFIT+ projections.

8.5.4 Purely synthetic dataset

Let us present the result with and without noise separately.

Case without noise: In this case there is no noise to filter, but it is still interesting to analyze the data and find the factors behind them. By the uniqueness property of rational functions presented in Section 8.2, we can hope that the methods based on rational functions are able to recover the original signals. We observe in Figure 8.5 that even though the SIR of methods using rational functions are on average better than the SIR recovered by HALS, 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 much 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 presented in what follows. The other R-NMF methods have similar performance, except in terms of computation time. Nevertheless, it seems that R-ANLS is the most accurate method in terms of obtained residual, while R-HANLS-based methods are faster.

We observe in Figure 8.6 that when the number of observations n is small

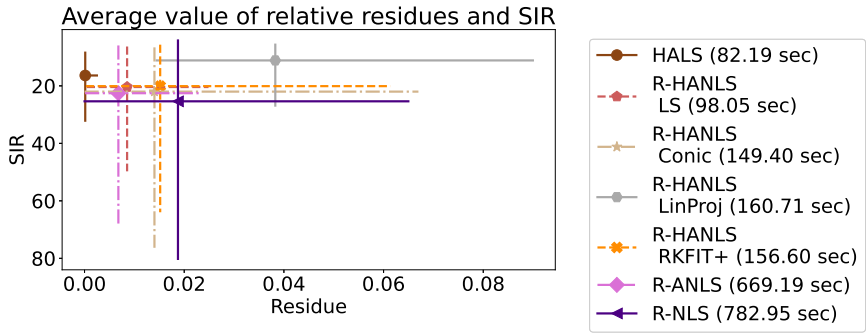


Fig. 8.5 Average performance, with $n = [20, 100]$, $d = [6, 10]$, $r = [5, 10]$. Data is not noisy.

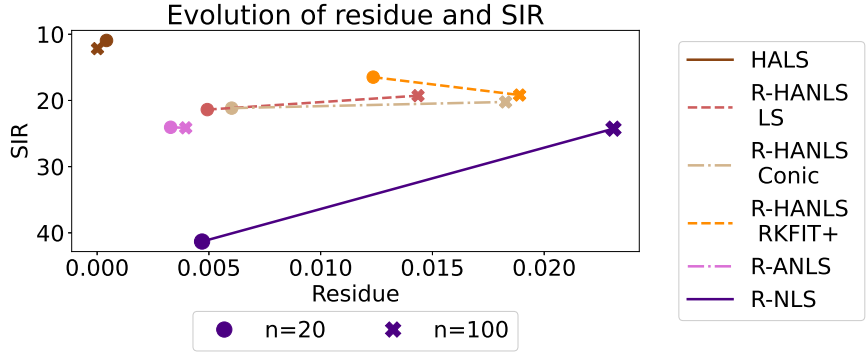


Fig. 8.6 Performance for varying n. Data is not noisy.

($n = 20$), R-NLS is able to recover the original signals, as this method obtains a low residual 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 residual 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 residual: a low relative residual means a good ability to filter the noise.

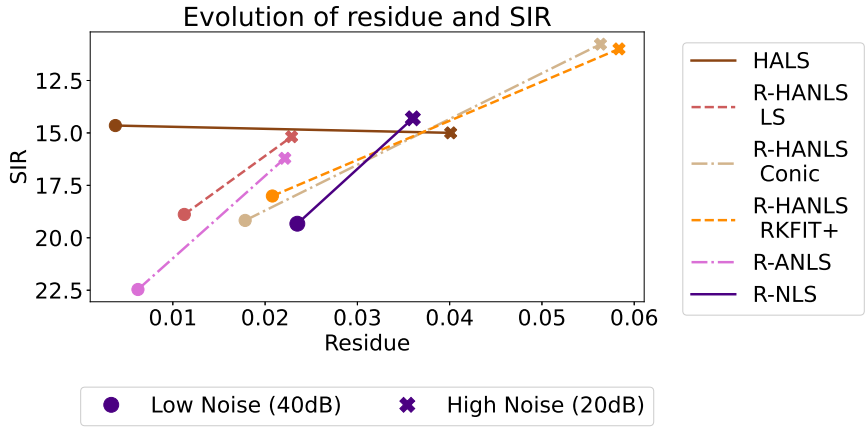


Fig. 8.7 Average performance for varying level of noise, with $n = [20, 100]$, $d = [6, 10]$, $r = [5, 10]$.

Figure 8.7 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. Moreover, the noise level has a high impact on the residual of HALS, which means that this method is not good at filtering the noise on the data. However, the quality of the recovered factors is not much impacted by the noise level and stays around 35 dB. R-HANLS LS and R-ANLS obtains the best performance when the noise level is high both in terms of SIR and residue. We see in Figure 8.8 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.

Combining R-NMF methods We analyzed the possibility of combining methods using rational functions to see if such combinations could have better performance. To do so, we performed a new test on synthetic data, with $d_1 = d_2 = 10$. The data was constructed with polynomials perturbed with a (1,10) polynomial (not (1,2) unlike in Section 8.5.2). The parameters are (m,n,r) = (200,100,5). It can be seen in Figure 8.9 that in this case R-NLS fails to find a good solution, probably because of the number of observations. R-ANLS and R-HANLS have quite comparable performance in terms of accuracy, but R-HANLS is faster.

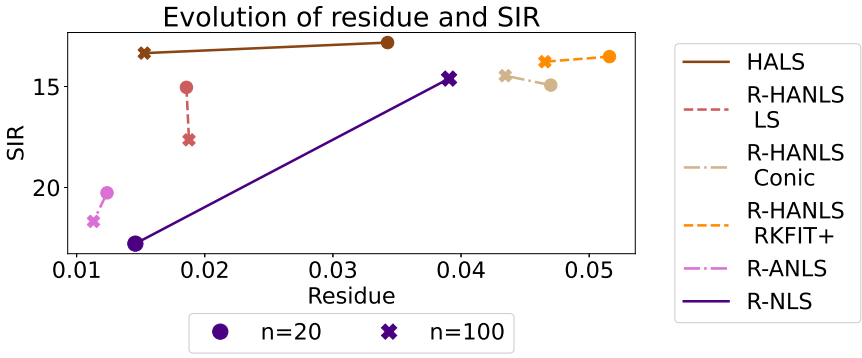


Fig. 8.8 Average performance for varying n, data is noisy.

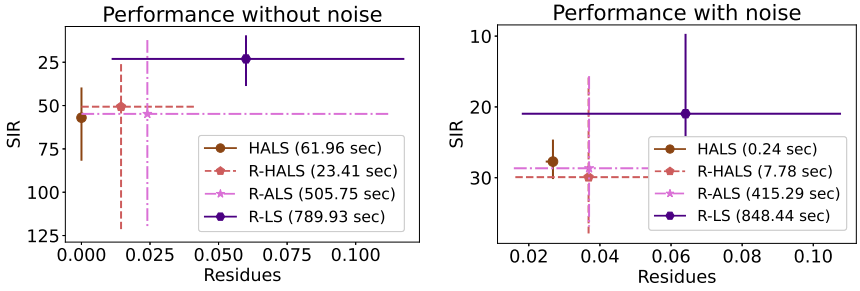


Fig. 8.9 Performance on a synthetic dataset, without noise (Left) and with noise level 20dB (Right).

We combine the methods in pairs as follows: the first method is executed until it reaches a tolerance $sc^t < 10^{-2}$ (8.21). Then, the second method is initialized with the factors obtained by the first one, and executed until convergence. We therefore use one partition to initialize another. These hybrid methods are called "method 1 - method 2".

We observe in Table 8.2 that associating the methods makes it possible to improve at least 2 criteria out of 3 (namely the time, the residual and the SIR) in more than 40 % of the cases, and even the methods which are the least easy to help are improved on at least two criteria in more than 30 % of the cases. Moreover, we see in the Tables 8.3 and 8.4 that the most accurate methods are always hybrid methods. Combining two methods can therefore help to improve the performance of the algorithm, and allows

to obtain the best results in terms of residual and SIR. However, there is no particular effect of these associations on the variability of the methods, which is illustrated in Figure 8.10 (the most accurate methods may also perform poorly in some tests).

	Initialize with another method is better for at least		
	1 criterion	2 criteria	3 criteria
HALS	60%	31%	5 %
R-HANLS	56%	37%	3 %
R-ANLS	76%	44%	17 %
R-NLS	86%	55%	41 %
All methods	70%	42 %	15%

Table 8.2 Percentage of cases where initializing with another method improves one or more of the three efficiency criteria: time, residual and SIR.

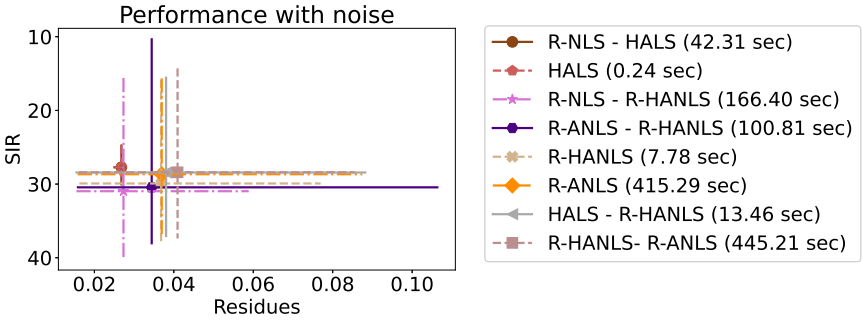


Fig. 8.10 Performance of some hybrid methods in noisy case.

It is also interesting to note that, in the noisy case, initializing the methods with R-NLS gives the most accurate results (Table 8.4). However, in Figure 8.9 we saw that this method used alone is the least accurate of the four on average. This is a bit counter-intuitive, since one could suppose that it is more interesting to initialize with a fast method which solves the problem locally, then to refine with the slower method which takes into account the whole problem, but it is here the reverse which is more efficient in the noisy case. We also observe that the hybrid methods ending with HALS or R-HANLS are faster than the others, which is quite logical. Indeed, obtaining a tolerance of 10^{-2} is quite easy and requires only a few iterations, so the complexity is dominated by the second method.

In the tests carried out, the most accurate method when there is no noise

Best	On average	Over all tests
time	HALS - R-HANLS (20 sec)	HALS - R-HANLS (2.57 sec)
residual	R-HANLS - HALS ($9.5 \cdot 10^{-6}$)	R-HANLS - R-ANLS ($1.9 \cdot 10^{-10}$)
SIR	R-HANLS - R-ANLS (70.64)	R-HANLS - R-ANLS (176.58)

Table 8.3 Best methods when there is no noise.

Best	On average	Over all tests
time	HALS (0.24 sec)	HALS (0.05 sec)
residual	R-NLS - HALS (0.027)	R-NLS - R-ANLS (0.015)
SIR	R-NLS - R-HANLS (30.96)	R-NLS - R-HANLS (40.17)

Table 8.4 Best methods in the noisy case.

is the association LR-HANLS - R-ANLS, and in the noisy case R-NLS - R-ANLS (this method obtains a residual close to R-NLS - R-HANLS, but a much higher SIR). It is important to keep in mind that methods using rational functions obtain very variable results depending on the problem to be solved, and even on the initialization, and that these methods will therefore not necessarily be the best in other situations. A fairly robust conclusion remains: combining the different partitions in a hybrid method often improves the performance of the algorithm.

8.5.5 Semi-synthetic dataset

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, i.e. 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 func-

tion is beneficial in such situations. We compared the methods to HALS as before, but also to LP-HALS using polynomials or splines from Chapter 3. Using results of previous section, we also considered the combinations R-ANLS - R-HANLS and R-NLS - R-HANLS.

Figure 8.11 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, the combination R-NLS - R-HANLS obtains the best SIR in a reasonable time (it is among the fastest R-NMF methods). LP-HALS using polynomials obtains also low residues and is competitive with R-NMF methods, while LP-HALS using splines is a bit less accurate in this case. Nevertheless, all methods using functions are able to filter the noise unlike HALS using vectors that obtains high residues. Figure 8.12 shows that

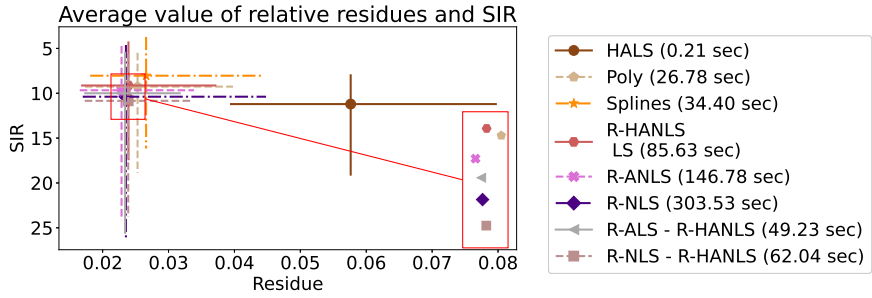


Fig. 8.11 Performance on semi-synthetic dataset.

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, for $r = 5$ or 8, the recovered signals do not really resemble the original ones, which is illustrated in Figure 8.13. We also observe in this figure that the signals recovered by HALS are very nonsmooth.

On the other hand, changing the degree does not influence much the SIR. However, Figure 8.14 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 because rational functions are able to express a larger variety of shapes than polynomials or splines for the same degrees of freedom. However, this advantage turns into a drawback when the number of degrees of freedom is too

high. Indeed, the performance 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 approaches using 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).

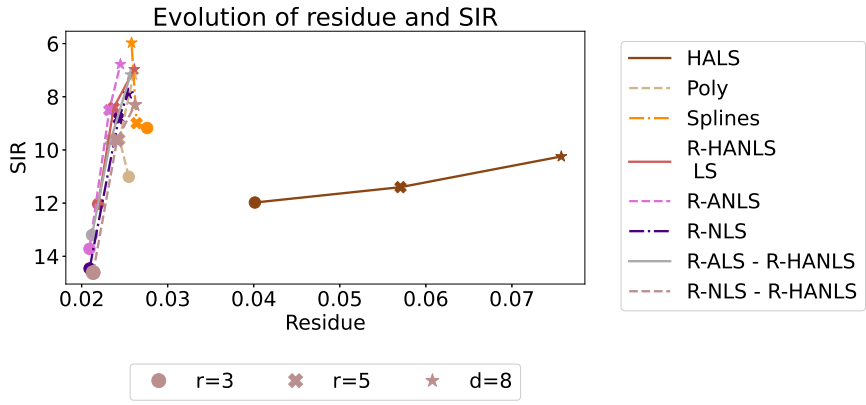


Fig. 8.12 Performance for varying rank.

8.6 Two applications

After the analysis from previous section on synthetic data, we now observe the performance of our methods in two real-world problems.

8.6.1 Using (R-)NMF for spectrum images unmixing

We test our method to unmix difficult spectrum images of multicomponent nanostructures from [18] and [79], where the authors use a method called MCR-LLM (Multivariate Curve Resolution by Log-Likelihood Maximization) to solve the unmixing problem. We apply the same preprocessing (see the cited papers for more information about the datasets), and scale the input data matrix so that each row has unit Manhattan norm and normalize the rows of matrix $\bar{\boldsymbol{X}}$ in the decomposition (this is done without loss of generality in NMF by scaling $\bar{\boldsymbol{A}}$ so that $\bar{\boldsymbol{A}}\bar{\boldsymbol{X}}^\top$ stays unchanged). We use k-means clustering to initialize weights in $\boldsymbol{X}$, and matrix $\boldsymbol{A}$ is initialized

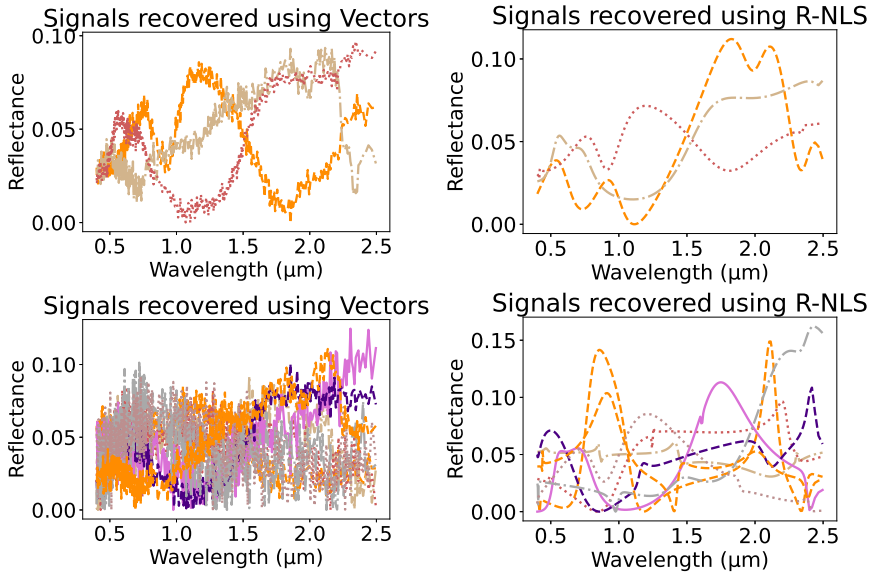


Fig. 8.13 Example of recovered factor A for $r = 3$ (up) or 8 (down), for HALS using vectors (left) or R-NLS (right).

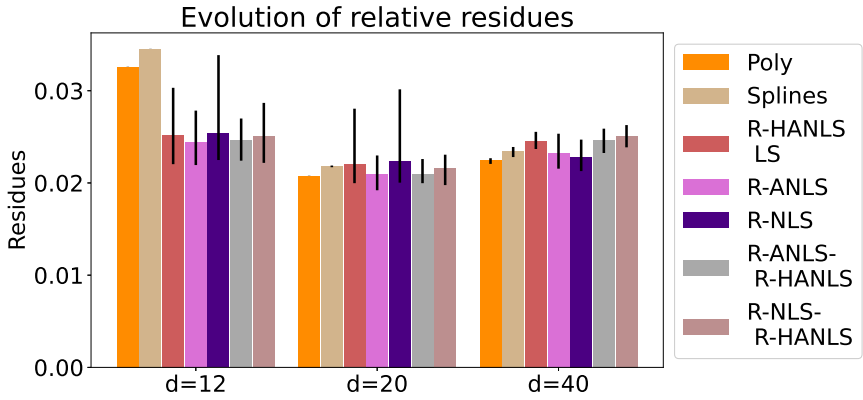


Fig. 8.14 Performance for varying degree of freedom (d).

using the unconstrained minimizer corresponding to this fixed $\boldsymbol{X}$ (an initial iterate that does not satisfy the constraints does not cause any trouble in our implementation).

A first one-dimensional dataset [18] contains the energy loss of the components compared to their position. The shape of such an energy loss does not look like a rational function. However, the relative abundance of each element should vary smoothly with its position, hence we will approximate it using rational functions of degrees $d_1 = d_2 = 20$. Because we use a random initialization, we report the best result out of ten runs for each method (as noise seems to be Poisson [18], we use the Kullback-Leibler divergence on the error to pick the best out of the ten tests). Then we compare the results between different methods visually, as depicted Figure 8.15.

We observe that R-NMF gives results similar to those of MCR-LLM, except that the relative abundances are smoothed thanks to the rational functions. When using standard NMF the result displays more noise, and features several unexpected peaks for all abundances, especially for SiO_2 (blue curve). Using splines or polynomials also leads to some noise but with a lower level than standard NMF. However our lack of knowledge in chemistry does not allow us to clearly determine the best method between MCR-LLM and R-NMF.

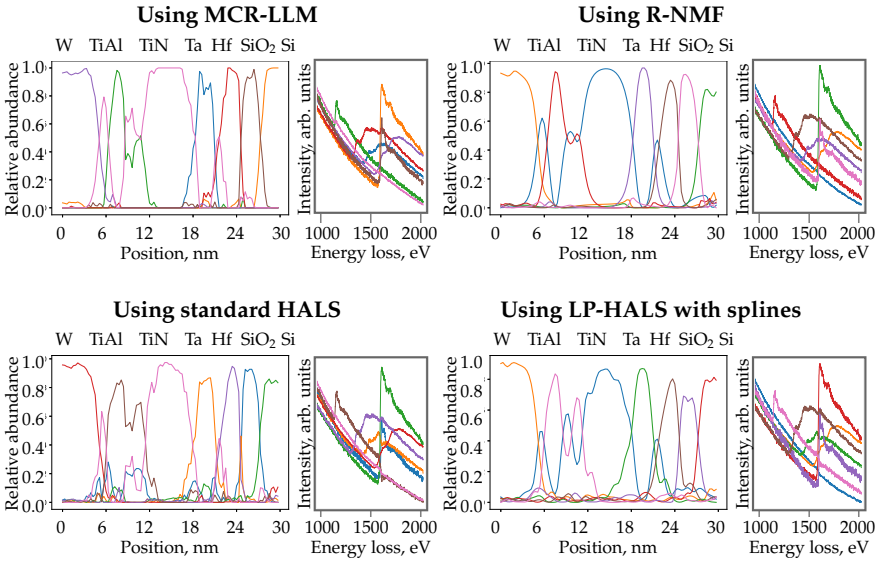


Fig. 8.15 Results of different methods on 1D dataset. Result obtained using polynomials were close to the ones obtained with splines.

For the next two-dimensional dataset [79], we reshape all images into vec-

tors to form the input matrix. The dataset contains the binding energy (instead of the energy loss), which resembles rational functions more. Furthermore, relative abundances are computed in a 2D space, so that we cannot properly represent them with univariate rational functions. Therefore, we use rational functions to represent the binding energy, with degrees $d_1 = d_2 = 44$, and report again on the best out of ten tests in Figure 8.16.

All methods are able to discriminate the three categories C_0, C_1 and C_2 in the abundance maps, and there is no noticeable difference between the three approaches using NMF. This time, NMF methods obtain abundances that appear less discriminate than MCR-LLM. Note however that other state-of-the-art methods presented in [79] are unable to distinguish between the three categories, so our method still makes sense in this case.

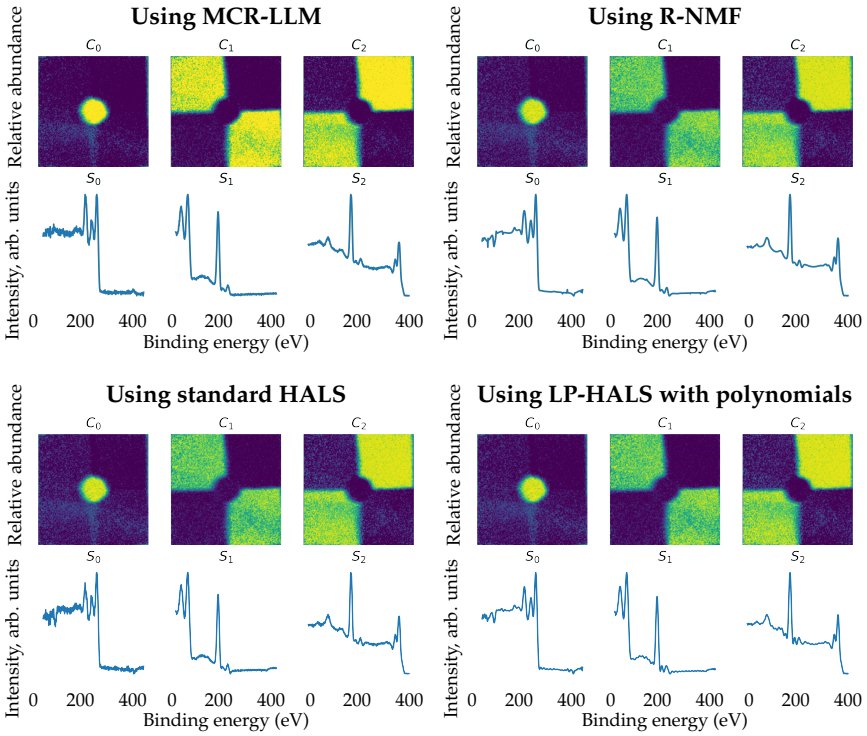


Fig. 8.16 Results of different methods on 2D dataset. Results obtained using splines are not represented but they are very close to results obtained with polynomials.

8.6.2 Using (R-)NMF for classification

We explore now the possibility of using R-NMF in a real problem: the Indian Pines classification problem. Classification is performed using the k-nearest-neighbors (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 $\boldsymbol{Y} \in \mathbb{R}^{200\times 21025}$ 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 $\boldsymbol{Y}$ as $\boldsymbol{A}\boldsymbol{X}^\top$ using NMF, where $\boldsymbol{A}$ contains in its columns sampled functions (note that there is no knowledge of labels at this stage). We use the R-HANLS methods for rational functions due to the high number of observations. Then the classification is performed on $\boldsymbol{X}^\top$ instead of $\boldsymbol{Y}$. The hope is that NMF filters noise in the data, while limiting the number of factors.

We also considered PCA to do the preprocessing (PCA does not have a nonnegativity constraint). Moreover, we tested the method of [32] 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 optimal 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 the case without nonnegativity constraints, 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 $r = 200$.

Figure 8.17 shows the accuracy obtained according to the factorization rank considered during pre-processing. It confirms the interest of using 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

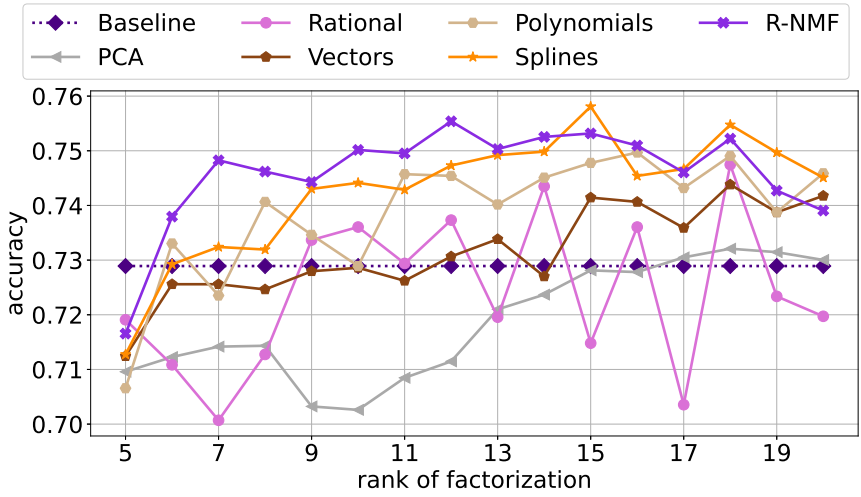


Fig. 8.17 Accuracy of classification using NMF as preprocessing with various factorization ranks.

sense, since PCA and Rational, which do not enforce this constraint, obtain the worst results. Note that the classification problem has 16 classes, and PCA starts to be competitive with the baseline from $r = 16$. It appears that, in this case, PCA needs at least as many features as classes to be able to describe properly the dataset, while NMF-based methods are able to describe the dataset properly with fewer features, thanks to their better ability to filter noise.

On the other hand, 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.

8.7 Discussion

We introduced R-NMF, a factorization model using nonnegative rational functions to unmix sampled signals, and presented three approaches to solve the problem. We observed that R-NMF performs better than other NMF approaches on semi-synthetic data or real-life data. A likely expla-

nation 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 8.2.

To explain this phenomenon, note that for R-ANLS and R-HANLS each update is not guaranteed to be optimal, and these two methods do thus inexact BCD. But doing inexact BCD was not a problem for polynomials or splines (see Section 5.3 and Chapter 6), so this explanation is not enough. 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. To overcome this sensitivity problem, the best thing we can do for now is to solve the problem with several initializations, and to choose the best factorization among those obtained. This makes particular sense when using R-NMF in the context of machine learning algorithms that have methods for selecting the best initialization, such as the K-fold used for classification in Section 8.6.2.

From our results, it appears that R-HANLS obtains on average worse results than R-ANLS. Moreover, R-NLS is able to obtain good results on very small problems, but when the problem size increases the method slows down very strongly and has difficulties to converge. Furthermore, R-NLS is resource demanding, and R-ANLS also but to a lesser degree. Therefore, we recommend using R-NLS only for very small problems, when $n < 50$ for example. For small problems, R-ANLS is accurate and not too slow (when $n < 1000$). However, for larger problems, R-HANLS is more appropriate as it is much less demanding. However, when possible, it should be initialized by a few iterations of R-NLS or R-ANLS to improve performance.

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 con-

dition is not trivial, like barycentric representation [42, 94] or sum of fractions [50], which is left for future work.

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 comment highlights the many existing possibilities when performing R-NMF.

9

Discussion and conclusion

In this thesis, we explored the possibility of using parametrizable functions in NMF to obtain better factorizations of datasets containing sampling of continuous signals. Moreover, we generalized the NMF problem to the H-NMF problem that handles two-dimensional functions and not only matrices, allowing thus to avoid the sampling of the input signals. For this purpose, we defined a product between two univariate functions $A : \mathbb{A} \mapsto \mathbb{R}$ and $X : \mathbb{X} \mapsto \mathbb{R}$ as $Y(a, x) = (A \otimes X)(a, x) = A(a)X(x)$, with $Y : \mathbb{A} \times \mathbb{X} \mapsto \mathbb{R}$. Our generalization of NMF aims at expressing a two-dimensional function $Y(a, x) : \mathbb{A} \times \mathbb{X} \mapsto \mathbb{R}$ as the sum of r products of two unidimensional functions, the factors A_k, X_k for $k = 1 \dots r$. We have thus $Y = \sum_{k=1}^{r} A_k \otimes X_k$, with $A_k : \mathbb{A} \mapsto \mathbb{R}$ and $X_k : \mathbb{X} \mapsto \mathbb{R}$ for all $k = 1, \dots, r$.

Some theoretical results This generalization is possible under some conditions. Input function Y must belong to a RKHS. Moreover, we impose factor functions A_k and X_k to belong to RKHS as well, respectively $\mathcal{A}$ and $\mathcal{X}$ for all k. This ensures that the evaluations of the factor functions is well defined. Of course, it is still possible to impose furthers constraints on the factors A_i, X_i such as nonnegativity constraints. We have then $A_k \in \tilde{\mathcal{A}} \subseteq \mathcal{A}\ \forall k$, and similarly $X_k \in \tilde{\mathcal{X}} \subseteq \mathcal{X}\ \forall k$.

We presented several ways to solve the H-NMF problem, inspired from methods solving the standard NMF problem. However, we mainly focused

on the H-HALS algorithm that generalizes the HALS algorithm. This algorithm optimizes the problem by iteratively minimizing it alternatively on a factor A_k or X_k while considering all other factors as fixed. When sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ are closed and convex, the H-HALS algorithms has convergence properties similar to those of the HALS algorithm for standard NMF. Indeed, if it is possible to avoid factors A_k, X_k to become 0-norm throughout algorithm, then H-HALS converges to a stationary point of the H-NMF problem. However, if computing each iteration of HALS is straightforward for standard NMF, H-HALS requests to project on sets $\tilde{\mathcal{A}}$, $\tilde{\mathcal{X}}$ at each iteration. This projection can be difficult, or at least difficult to be performed exactly.

H-HALS using inexact iterations We therefore explored several situations where the iterations of H-HALS are not performed exactly, mostly due to inexact projections. If all iterates are ϵ-stationary point of the intermediate problems (that consider the problem on one factor, considering the others as fixed), the solution found by H-HALS will be at worse $2r\epsilon$-stationary. In another setting, the constraints can be handled with approximations of the indicator functions of sets $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$, named u^t, and those approximations functions can evolute throughout iterations. If $0 \leq u^{t+1} \leq u^t$ for all iteration t, H-NMF converges to a stationary point of the problem using indicator function u^∞. Finally, if all iterations have a distance to the exact update smaller than ϵ, the H-NMF algorithm converges to a point close to a Nash equilibrium (the distance to the Nash equilibrium depends on ϵ). Moreover, if ϵ tends to 0 , H-HALS converges to a stationary point under mild conditions.

The first result provides a stopping criterion for iterative projection algorithms and gives a general condition to know if found points are good enough. Moreover, this result on ϵ-stationary points can allow dealing with cases where the functions A_k, X_k may have 0-norm, see Theorem 5.7. This result has been used for splines (Section 5.3.1) and polynomials (Section 6.3.1) to accelerate the H-HALS algorithm . The second result allows one to project on subsets of $\tilde{\mathcal{A}}, \tilde{\mathcal{X}}$ and increase progressively the size of these subsets, as illustrated for splines in Section 5.3.2 . Finally, the third result is also the most general and allows one to have information about the quality of the factorization, if the error can be quantified. Moreover, it indicates that computing projections more and more precisely allows converging to a stationary point. This is especially useful for iterative projec-

tion algorithms. It has been used with success to accelerate H-HALS using polynomials in Section 6.3. In the same Chapter 6, we also considered other heuristics to project on nonnegative polynomials, in order to accelerate the algorithm. Those heuristic projections obtain results similar than the accelerations based on theory, and are even often faster, but their convergence to a stationary point is not guaranteed.

H-NMF in practice If NMF generalizes well to H-NMF from a theoretical point of view, manipulating functions is more difficult in practice. This is why we considered the use of functions parametrizable using a low number of parameters. The simplest case is when the considered functions are linear combinations with nonnegative coefficients of a few basis elements, denoted as $\Pi = \{\Pi_l\}$, for example splines with nonnegative coefficients in the B-spline basis (see Chapter 7). Indeed, in this case, the problem is very close to the standard NMF problem. If Π^A is the basis for $\tilde{\mathcal{A}}$, and Π^X the basis for $\tilde{\mathcal{X}}$, we can compute matrices $\boldsymbol{Z}$ with $\boldsymbol{Z}_{i,j} = \langle Y, \Pi_i^A \otimes \Pi_j^X \rangle_{\mathcal{H}}$, $\boldsymbol{M}^A$ with $\boldsymbol{M}^A_{i,j} = \langle \Pi_i^A, \Pi_j^A \rangle_{\mathcal{A}}$ and $\boldsymbol{M}^X$ with $\boldsymbol{M}^X_{i,j} = \langle \Pi_i^X, \Pi_j^X \rangle_{\mathcal{X}}$. Then the problem can be solved on $\boldsymbol{Z}$ instead of Y, and on the coefficients of A_k, X_k, denoted as $\boldsymbol{C}^A_{:k}, \boldsymbol{C}^X_{:k}$, using the fact that $\langle A_i, A_j \rangle_{\mathcal{A}} = \boldsymbol{C}^{A\top}_{:i} \boldsymbol{M} \boldsymbol{C}^A_{:j}$ and similarly for $\langle X_i, X_j \rangle_{\mathcal{X}}$. Matrices $\boldsymbol{C}^A_{:k}$ and $\boldsymbol{C}^X_{:k}$ must be nonnegative.

In the case of linearly parametrizable functions, whose coefficients are not imposed to be nonnegative, like nonnegative polynomials and nonnegative splines, it is also possible to work with matrices $\boldsymbol{Z}$, $\boldsymbol{M}^A$ and $\boldsymbol{M}^X$, but the constraints on $\boldsymbol{C}^A$ and $\boldsymbol{C}^X$ become more complex. This is presented for polynomials and splines in Chapter 3.

It is also possible to work with functions that are not linearly parametrizable, as rational functions. In this case, it is not anymore possible to compute matrices $\boldsymbol{Z}$, $\boldsymbol{M}^A$ and $\boldsymbol{M}^X$, but it is necessary to compute inner products between Y, A_k and X_k at each iteration instead. This is presented for discrete rational functions in Chapter 8.

Computing the inner products between Y, A_k and X_k is not always easy, especially as the input data does not always have the expected shape. Most of the time, input data contains in fact samples of continuous functions. In this case, we could attempt to interpolate the functions, but it is easier to work on the known points instead. The problem then becomes again a problem over matrices and is thus easier to solve. We can then wonder why working on functions, if they are discretized anyway?

The fact that behind each column of $\boldsymbol{A}$ and $\boldsymbol{X}$ there is a function has a main advantage: the linear combinations of the columns of $\boldsymbol{A}$ and $\boldsymbol{X}$ preserve the properties and the structure of the functions behind them, which can improve a lot the performance of NMF in terms of accuracy. For example, polynomials are globally smooth, splines are piece-wise smooth, and rational functions are generally smooth with some peaks.

Moreover, even though we did not explore much this path in this thesis, working directly on functions is possible. Indeed, Trefethen & al. [11, 118, 119] have develop Chebfun to represent functions relying on very precise polynomial interpolations, which allows modeling the input data Y. Moreover, Marteau-Ferey & al. [91] have proposed a structure to model nonnegative functions, which is an interesting approach to model factors A_k and X_k, and would be an exciting future work.

Using functions in NMF is therefore possible in theory and several approaches to use H-NMF in practice have been explored in this work, using polynomials, splines, or rational functions. Even though the approach of this thesis is quite theoretical, we showed in Chapter 7 that using H-NMF can be very useful in practice, for example in the context of image or matrix completion.

In Chapter 8 we analyzed the case of rational functions of fixed degree, that do not form a convex set (and the set of rational functions of fixed degree is not even a vector space). Therefore, this case does not meet the conditions to have theoretical guarantees. Nevertheless, we could show that even in this case, the results of H-NMF are interesting in terms of accuracy of the factorization. For example, H-NMF using rational functions is more accurate than H-NMF using polynomials or splines to model the dataset from the real Indian Pines classification problem using a small rank, and leads to better classification results. H-NMF using polynomials or splines are also more accurate than standard NMF for this problem. This good result comes at the price of a slower algorithm on average, because rational functions are not linearly parametrizable, and H-NMF using rational functions is also more sensitive to the initialization, probably due to the non-convexity of that set of parametrizable functions.

Perspectives We considered polynomials, splines and rational functions in this work, but many other functions with interesting properties can be explored in the H-NMF framework, such as sums of exponentials or more

general descriptions of nonnegative functions, for example the one proposed in [91]. It is also possible to use different functions for factors $\boldsymbol{A}$ and $\boldsymbol{X}$ (for example splines in $\boldsymbol{A}$ and rational functions in $\boldsymbol{X}$), or to consider ratios of splines (instead of ratios of polynomials) or even piece-wise rational functions. The possibilities offered by H-NMF are therefore quite broad.

Moreover, it would be interesting to consider input functions rather than discretizations of functions, using for example Chebfun [11, 118, 119].

Furthermore, this work focused on the minimization of the Frobenius norm of the error, but other cost functions can be considered to be more adapted to the problem to be solved, creating several interesting algorithmic challenges due to the loss of the Euclidean structure.

The final word To conclude, in this thesis we have presented a general framework allowing to perform NMF using functions: the H-NMF (Chapter 4). To solve this new problem, we focused on the generalization of the HALS algorithm, namely the H-HALS, which has very similar theoretical properties as the HALS (Chapter 5). We used these results to perform H-NMF with polynomials or splines (Chapter 3), and showed that the performance in terms of factorization quality could be greatly improved compared to standard NMF, and in particular could allow image/matrix completion to be performed efficiently (Chapter 7). This performance improvement comes at the cost of a slower algorithm, especially for small and medium instances, or when the degree of the polynomials or the number of interior points of the splines is high. To overcome this problem, we have considered several heuristics, allowing to speed up the algorithm significantly, without deteriorating its performance (Chapter 6).

Finally, we have analyzed a case where the functions of interest are not convex, and which therefore does not allow us to have any theoretical assurance of convergence: the case of H-NMF using rational functions (Chapter 8). In this case also, very good results have been observed in terms of average quality of the factorization. However, the algorithm is then very sensitive to its initialization and may also obtain very bad results. Nevertheless, H-NMF with rational functions obtains better results than H-NMF using polynomials or splines in a classification problem. In such machine learning problems, there are ways to select the most promising classification, and thus smooth out the initialization sensitivity problem of H-NMF using rational functions.

A

Proof of Theorem 5.7

Let us first recall Theorem 5.7.

Theorem A.1. *Consider the H-NMF problem from Definition 4.4. Suppose $\mathcal{A} = \mathbb{R}^m$, $\mathcal{X} = \mathbb{R}^n$, $\tilde{\mathcal{A}} = [0, M]^m$ and $\tilde{\mathcal{X}} = [0, M]^m$ (standard NMF problem bounded above by M).*

Suppose that the problem is divided in $2r$ blocks, the columns of $\boldsymbol{A}$ and $\boldsymbol{X}$, and solved using BCD Algorithm 5.1, with update (5.3) replaced by:

$$\begin{aligned} \tilde{x}_i^{t+1} &= \operatorname*{argmin}_{\xi} f_i(\xi\,; x^t) \quad \text{such that } \xi \in \mathcal{P}_i \\ x_i^{t+1} &= \tilde{x}_i^{t+1} + \lambda[x_i^t - \tilde{x}_i^{t+1}]_{\mathcal{P}_i} \end{aligned} \tag{A.1}$$

where $\lambda \in [0, 1]$, $\|x_i^{t+1}\|^2 \geq \delta > 0$ and x_i^{t+1} is ϵ-valid.

For any ϵ, if δ and x^0 are chosen so that $2M\sqrt{\delta \max(n, m) f(x^0)} \leq \epsilon$, and $\|x_i^0\| \geq \delta$ for all i, then a valid update x_i^{t+1} always exists. The resulting algorithm converges to a $2r\epsilon$-stationary point.

Proof. The projection on $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ is a threshold operation:

$$\left([\xi]_{\tilde{\mathcal{A}}/\tilde{\mathcal{X}}}\right)_j = \begin{cases} 0 & \text{if } \xi_j \leq 0 \\ \xi_j & \text{if } \xi_j \in [0, M] \\ M & \text{if } \xi_j \geq M \end{cases}.$$

When $(x_i^t - \tilde{x}_i^{t+1})_j \leq 0$, then $(x_i^{t+1})_j = (\tilde{x}_i^{t+1})_j \in [0, M]$. Otherwise, $0 \leq (x_i^{t+1})_j = (\tilde{x}_i^{t+1})_j + \lambda(x_i^t - \tilde{x}_i^{t+1})_j \leq \lambda(x_i^t)_j + (1-\lambda)(x_i^t)_j \leq M$. Therefore, $x_i^{t+1} \in \mathcal{P}_i \; \forall \lambda \in [0,1]$ and condition 1 is always respected.

Concerning condition 3, we know by Lemma 5.2 that $\tilde{x_i}^{t+1} = [x^*]_{\mathcal{P}_i}$ where x^* is such that $D_{f_i(\xi\,;x^t)}(x^*) = 0$. We have then, using Equation (5.5):

$$\begin{aligned}\langle D_{f_i(\xi\,;x^t)}(x_i^{t+1}), x_i^t - x_i^{t+1}\rangle \geq 0 &\Leftrightarrow\; 2O_i^t\langle x_i^{t+1} - x^*, x_i^t - x_i^{t+1}\rangle \geq 0 \\ &\Leftrightarrow\; 2O_i^t \sum_j \left(x_i^{t+1} - x^*\right)_j \left(x_i^t - x_i^{t+1}\right)_j \geq 0. \quad \text{(A.2)}\end{aligned}$$

- If $(x)_j^* \leq 0$ then $(\tilde{x}_i^{t+1})_j = 0$ and $(x_i^{t+1})_j = \lambda(x_i^t)_j$. This means that $\left(x_i^{t+1} - x^*\right)_j \left(x_i^t - x_i^{t+1}\right)_j = \left(\lambda x_i^t - x^*\right)_j \left((1-\lambda)x_i^t\right)_j$ which is nonnegative when $\lambda \in [0,1]$.

- If $(x)_j^* \in [0, M]$ then $(\tilde{x}_i^{t+1})_j = (x)_j^*$. This means that $\left(x_i^{t+1} - x^*\right)_j \left(x_i^t - x_i^{t+1}\right)_j = \left(\lambda[x_i^t - x_i^*]_{\mathcal{P}_i})_j (x_i^t - x^* + \lambda[x_i^t - x^*]_{\mathcal{P}_i}\right)_j$. If $x_i^t - x_i^* \leq 0$ the first element of the product is 0. Otherwise, both elements are nonnegative, which means that the product is always nonnegative.

- Finally, if $(x)_j^* \geq M$, then $(\tilde{x}_i^{t+1})_j = M$ and $(x_i^{t+1})_j = M$. This means that $\left(x_i^{t+1} - x^*\right)_j \left(x_i^t - x_i^{t+1}\right)_j = \left(M - (x^*)_j\right)\left((x_i^t)_j - M\right)$, which is nonnegative.

Therefore, all elements in the sum in (A.2) are nonegative, and as O_i is always nonnegative, condition 3 is always respected.

Concerning condition 2, first observe that $\tilde{x}_i^{t+1}$ always respect this condition as it is optimal. Therefore, if $\|\tilde{x}_i^{t+1}\| \geq \delta$, choosing $\lambda = 0$ is a valid choice.

Otherwise, observe that when $\lambda = 1$, $\left(x_i^{t+1}\right)_j \geq \left(x_i^t\right)_j \; \forall j$, and thus $\|x_i^{t+1}\|^2 \geq \|x_i^t\|^2 \geq \delta$. On the contrary, when $\lambda = 0$, $\|x_i^{t+1}\|^2 = \|\tilde{x}_i^{t+1}\|^2 < \delta$. As the function determining x_i^{t+1} is continuous in λ, it is possible to choose $\lambda^* \in [0,1]$ such that $\|x_i^{t+1}\|^2 = \delta$.

Now the question is if x_i^{t+1} using λ^* satisfies condition 2.

$$\begin{aligned}
&\langle D_{f_i(\xi\ ;x^t)}(x_i^{t+1}), x_i - x_i^{t+1}\rangle \geq -\epsilon \quad \forall x_i \in \mathcal{P}_i \\
\Leftrightarrow &\left(\min_{x_i\in\mathcal{P}_i} \langle D_{f_i(\xi\ ;x^t)}(x_i^{t+1}), x_i - x_i^{t+1}\rangle\right) \geq -\epsilon \\
\Leftrightarrow &\, 2O_i^t \min_{x_i\in\mathcal{P}_i} \sum_j \left(x_i^{t+1} - x^*\right)_j \left(x_i - x_i^{t+1}\right)_j \geq -\epsilon
\end{aligned}$$

At minimum, $\left(x_i\right)_j = \begin{cases} M & \text{if } \left(x_i^{t+1} - x^*\right)_j < 0 \rightarrow \left(x_i^{t+1}\right)_j = M \\ 0 & \text{else} \end{cases}$,

$$\Leftrightarrow 2O_i^t \sum_{\left(x_i^{t+1}-x^*\right)_j>0} \left(x_i^{t+1} - x^*\right)_j \left(x_i^{t+1}\right)_j \leq \epsilon$$

As $\sum_i a_i b_i \leq \sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}$ (Cauchy-Schwarz inequality):

$$\begin{aligned}
&\Leftarrow 2O_i^t \sqrt{\sum_{\left(x_i^{t+1}-x^*\right)_j>0} \left(x_i^{t+1} - x^*\right)_j^2} \sqrt{\sum_{\left(x_i^{t+1}-x^*\right)_j>0} \left(x_i^{t+1}\right)_j^2} \leq \epsilon \\
&\Leftarrow -2O_i^t \|x_i^{t+1} - x^*\| \, \|x_i^{t+1}\| \geq -\epsilon
\end{aligned}$$

Note that $O_i^t \|x_i^{t+1} - x^*\|^2 = f_i(x_i^{t+1}\ ; x^t) - f_i(x^*\ ; x^t)$ by Lemma (5.1)

$$\Leftrightarrow 2\sqrt{O_i^t (f_i(x_i^{t+1}\ ; x^t) - f_i(x^*\ ; x^t))\delta} \leq \epsilon \qquad \text{using } \lambda^*$$

We have $\sqrt{O_i^t} = \|x_k^t\| \leq M\sqrt{\max(m,n)}$, $f_i(x_i^{t+1}\ ; x^t) \leq f(x^0)$, and $-f(x^*) \leq 0$, so

$$\Leftarrow 2M\sqrt{\max(n,m)\left(f(x^0)\right)\delta} \leq \epsilon$$

It is then possible to satisfy all conditions when $2M\sqrt{\max(n,m)\left(f(x^0)\right)\delta} \leq \epsilon$. Note that the constraint on ϵ is stronger that needed due to the numerous simplifications. Nevertheless, using $\epsilon-$stationarity we manage to consider the whole spaces $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{X}}$ and not subspaces without zeros elements, and we theoretically bounded the stationarity of the obtained point.

□

B

Description and implementation of the projection methods for rational functions

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 a trust region reflective algorithm. The algorithm is stopped when either the cost function is not enough improved anymore, or the iterates are too close to each others, or the norm of the gradient is very small.

Alternating Least Squares: problem (8.17) is as a conic problem. Indeed, using Markov-Lukacs theorem, nonnegative polynomials can be expressed using sum of squares polynomials (SOS), and SOS can be expressed using positive semidefinite matrices [15]. Therefore, problem (8.17) can be rewritten using appropriate matrices: $\boldsymbol{V}_{\boldsymbol{\tau}}(g)$ a Vandermonde-like matrix taking into account the known denominator, $\boldsymbol{V}_{\boldsymbol{\tau}}(g) \in \mathbb{R}^{m \times (d_1+1)}$ and $\boldsymbol{R} \in$

$\mathbb{R}^{(d_1+1)\times\frac{d_1^2}{2}+d_1+1}$ the matrix recovering the coefficients of the polynomial from the positive semi-definite matrices. $\boldsymbol{R}$ is built using Gram matrices, see Section 2.3.1. Let $\mathcal{S}_+^d$ be the set of positive semidefinite matrices in $\mathbb{R}^{d\times d}$. We have

$$\min_{(\boldsymbol{S}_1,\boldsymbol{S}_2)\in\mathcal{S}_+^{\frac{d_1}{2}+1}\times\mathcal{S}_+^{\frac{d_1}{2}}} \left\| \boldsymbol{z} - \boldsymbol{V}_{\boldsymbol{\tau}}(g)\boldsymbol{R}\begin{bmatrix} vec(\boldsymbol{S}_1) \\ vec(\boldsymbol{S}_2)\end{bmatrix}\right\|^2. \tag{B.1}$$

Problem (B.1) can be compressed using the singular value decomposition of $\boldsymbol{V}_{\boldsymbol{\tau}}(g) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^\top$. It can be proved that using $\tilde{\boldsymbol{V}} = \boldsymbol{\Sigma}\boldsymbol{W}^\top$ and $\tilde{\boldsymbol{z}} = \boldsymbol{U}^\top\boldsymbol{z}$ leads to the same minimization problem, to one constant. It is solved using Mosek 9.1 [6]. 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 [10] on which we add nonnegativity constraints:

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

where $\tilde{g} \in \mathcal{P}_+^{d_2}(T)$ is fixed so that $\tilde{g}(\tau_m) = 1$. This equation is equivalent to (8.12) when $g(\boldsymbol{\tau}) = \tilde{g}(\boldsymbol{\tau}) > 0$. It transforms the problem into a simpler problem on polynomials.

The normalization 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 the discretization of a nonnegative rational function. We observed that the error is often much smaller on (B.2) than on (8.12). For example, suppose that $g(\boldsymbol{\tau}_i)$ and $h(\boldsymbol{\tau}_i)$ are very small and $\tilde{g}(\boldsymbol{\tau}_i) = 1$. In this case, $\frac{\boldsymbol{z}_i g(\boldsymbol{\tau}_i) - h(\boldsymbol{\tau}_i)}{\tilde{g}(\boldsymbol{\tau}_i)}$ can be much smaller than $\boldsymbol{z}_i - \frac{h(\boldsymbol{\tau}_i)}{g(\boldsymbol{\tau}_i)}$. 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 $\|\boldsymbol{z} - f(\boldsymbol{\tau})\|^2 =$

$$\left\| \frac{\boldsymbol{z}\tilde{g}(\boldsymbol{\tau}) + \boldsymbol{z}\delta(\boldsymbol{\tau}) - h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} \cdot \frac{\tilde{g}(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau}) + \delta(\boldsymbol{\tau})} \right\|^2. \tag{B.3}$$

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 (B.4) is thus an upper bound of the cost function of problem (8.12):

$$\min_{h \in \mathcal{P}_+^{d_1}(T), \delta \in \mathcal{P}_+^{d_2}(T)} \left\| |\boldsymbol{z} + \frac{\boldsymbol{z}\delta(\boldsymbol{\tau}) - h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} \right\|^2. \tag{B.4}$$

Solving problem (B.4) ensures to have a rational function that leads also to a low cost in problem (8.12), which was not the case when solving (B.2). It can be solved in a similar way as (B.1). Using appropriate matrices $\boldsymbol{V}_{\boldsymbol{\tau}}(\tilde{g}, \boldsymbol{z}) \in \mathbb{R}^{m \times (d_1 + d_2 + 2)}$ and $\boldsymbol{R} \in \mathbb{R}^{(d_1+d_2+2) \times \frac{d_1^2}{2} + d_1 + \frac{d_2^2}{2} + d_2 + 2}$, we have:

$$\min_{\substack{(\boldsymbol{S}_1, \boldsymbol{S}_2, \boldsymbol{D}_1, \boldsymbol{D}_2) \in \\ \mathcal{S}_+^{\frac{d_1}{2}+1} \times \mathcal{S}_+^{\frac{d_1}{2}} \times \mathcal{S}_+^{\frac{d_2}{2}+1} \times \mathcal{S}_+^{\frac{d_2}{2}}}} \left\| \boldsymbol{z} + \boldsymbol{V}_{\boldsymbol{\tau}}(\tilde{g}, \boldsymbol{z}) \boldsymbol{R} \begin{bmatrix} vec(\boldsymbol{S}_1) \\ vec(\boldsymbol{S}_2) \\ vec(\boldsymbol{D}_1) \\ vec(\boldsymbol{D}_2) \end{bmatrix} \right\|^2. \tag{B.5}$$

Problem (B.5) can be compressed, using the singular value decomposition of $\boldsymbol{V}_{\boldsymbol{\tau}}(\tilde{g}, \boldsymbol{z}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^\top$, with $\tilde{\boldsymbol{V}} = \boldsymbol{\Sigma}\boldsymbol{W}^\top$ and $\tilde{\boldsymbol{z}} = \boldsymbol{U}^\top \boldsymbol{z}$. This problem is solved using Mosek 9.1 solver.

RKFIT+: operator h' from (8.19) can be solved analytically using matrix $\boldsymbol{V}_1$ such that $\frac{h(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} = \boldsymbol{V}_1 \boldsymbol{h}$, where $\boldsymbol{h}$ is the coefficient vector of h (such a matrix always exists). Problem becomes:

$$\frac{h'(\tilde{g}, \delta, \boldsymbol{z}, \boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} = \boldsymbol{V}_1 \operatorname{argmin}_{\boldsymbol{h}} \left\| \boldsymbol{z} + \frac{\boldsymbol{z}\delta(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} - \boldsymbol{V}_1 \boldsymbol{h} \right\|^2. \tag{B.6}$$

The solution of this problem can be expressed using $\boldsymbol{V}_1^\dagger$ the pseudo-inverse of $\boldsymbol{V}_1$ as:

$$\frac{h'(\tilde{g}, \delta, \boldsymbol{z}, \boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} = \boldsymbol{V}_1 \boldsymbol{V}_1^\dagger \left(\boldsymbol{z} + \frac{\boldsymbol{z}\delta(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} \right). \tag{B.7}$$

Similarly, we can define $\boldsymbol{V}_2$ so that $\frac{\boldsymbol{z}\delta(\boldsymbol{\tau})}{\tilde{g}(\boldsymbol{\tau})} = \boldsymbol{V}_2\boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is the coefficient vector of δ. Problem (8.19) is then $\min_{\delta \in \mathcal{P}_+^{d_2}(T)} \|(\boldsymbol{I} - \boldsymbol{V}_1\boldsymbol{V}_1^\dagger)(\boldsymbol{z} + V_2\boldsymbol{\delta})\|^2$. This problem can be compressed, using SVD decomposition of $(\boldsymbol{I} - \boldsymbol{V}_1\boldsymbol{V}_1^\dagger)\boldsymbol{V}_2$: $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^\top$. The cost becomes $\|\boldsymbol{U}^\top(\boldsymbol{I} - \boldsymbol{V}_1\boldsymbol{V}_1^\dagger)\boldsymbol{z} + \boldsymbol{\Sigma}\boldsymbol{W}^\top\boldsymbol{\delta}\|^2$. The problem can then be solved using Mosek 9.1, in a similar way than (B.1).

The problem of finding the best numerator for a fixed denominator is problem B.1, presented in the Alternating Least Squares approach.

LinProj: this problem is solved using Mosek 9.1. 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 $\max_i \left(\left|\boldsymbol{z}_i - \frac{h(\boldsymbol{\tau}_i)}{g(\boldsymbol{\tau}_i)}\right|\right)$, $g(\boldsymbol{\tau})$ is imposed to be greater than 1.

C

Implementation

The code used for this thesis is available on Code Ocean, with some representative tests: https://codeocean.com/capsule/5065386/tree.

The code is written in Python, using MOSEK version 9.1.3 [6] to solve some optimization problems. The structure of the code is the following:

- Main files:
  - **ClassesA.py**: this file describes the factors A/X. It contains an abstract class `Factors` with all functions that must be implemented by a class of factors to be usable by H-HALS. It also contains class `PosMat` that implement all these functions when factors contains nonnegative vectors (standard case).
  - **ClassesY.py**: this file describes the input data $Y : \mathbb{A} \times \mathbb{X} \mapsto \mathbb{R}$, with all useful functions for H-HALS. Three kind of data are considered:
    - `Matrices`: Both domains $\mathbb{A}$ and $\mathbb{X}$ are discrete and Y can thus be described as a matrix.
    - `HilbFun`: Both domains $\mathbb{A}$ and $\mathbb{X}$ are continuous and Y is thus described as a function of two variables.

  - `ListFun`: Domain $\mathbb{A}$ is continuous while domain $\mathbb{X}$ is discrete. Input Y is then described as a list of functions.

  This file also contains class `realData` that describes input Y when both factors $\{A_k\}_{k=1}^r$ and $\{X_k\}_{k=1}^r$ are known. This is useful for testing.

- ◇ **HilbertHALS**: this file contains the implementation of the H-HALS algorithm.

- Additional folders:
  - ◇ **Poly**: this folder contains functions needed for H-HALS using polynomials. In particular, file PHALS is the class for factors containing nonnegative polynomials.
  - ◇ **Spline**: this folder contains functions needed for H-HALS using splines. In particular, file SHALS is the class for factors containing nonnegative splines.
  - ◇ **Ratio**: this folder contains functions needed for H-NMF using rational functions. In particular, file Ratio is the class for factors containing nonnegative rational functions (R-HANLS) and file RatioLS contains the implementation of R-NLS and R-ANLS.
  - ◇ **Test:** this folder contains used datasets, helpers to perform tests and notebooks with main results. Those notebooks are also very useful to better understand how to use our implementation of H-HALS in practice.

Our implementation uses results from Section 4.2.3 and work on precomputable Z instead of Y (when neither A nor X contains linearly parametrizable functions, $Z = Y$).

Bibliography

[1] A. A. Ahmadi, G. Hall, A. Papachristodoulou, J. Saunderson, and Y. Zheng. Improving efficiency and scalability of sum of squares optimization: Recent advances and limitations. In *2017 IEEE 56th annual conference on decision and control (CDC)*, pages 453–462. IEEE, 2017.

[2] A. A. Ahmadi and A. Majumdar. Some applications of polynomial optimization in operations research and real-time decision making. *Optimization Letters*, 10(4):709–729, 2016.

[3] M. Ahookhosh, L. T. K. Hien, N. Gillis, and P. Patrinos. Multi-block Bregman proximal alternating linearized minimization and its application to orthogonal nonnegative matrix factorization. *Computational Optimization and Applications*, 79(3):681–715, 2021.

[4] A. M. S. Ang and N. Gillis. Accelerating nonnegative matrix factorization algorithms using extrapolation. *Neural computation*, 31(2):417–439, 2019.

[5] F. Anowar, S. Sadaoui, and B. Selim. Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA,LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE). *Computer Science Review*, 40:100378, 2021.

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

[7] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

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

[9] D. Backenroth, R. T. Shinohara, J. A. Schrack, and J. Goldsmith. Nonnegative decomposition of functional count data. *Biometrics*, 76(4):1273–1284, 2020.

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

[11] Z. Battles and L. N. Trefethen. An extension of matlab to continuous functions and operators. *SIAM Journal on Scientific Computing*, 25(5):1743–1770, 2004.

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

[13] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

[14] D. P. Bertsekas. Nonlinear programming. *Scientific, Athena*, 1999. Using errata updated the 1/18/2016, available at http://www.athenasc.com/nlperrata.pdf.

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

[16] N. Boumal, V. Voroninski, and A. Bandeira. The non-convex Burer-Monteiro approach works on smooth semidefinite programs. *Advances in Neural Information Processing Systems*, 29, 2016.

[17] S. Boyd, N. Parikh, and E. Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[18] N. Braidy and R. Gosselin. Unmixing noisy co-registered spectrum images of multicomponent nanostructures. *Scientific reports*, 9(1):1–8, 2019.

[19] M. A. Branch, T. F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained mini-

mization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

[20] I. Buciu, N. Nikolaidis, and I. Pitas. Nonnegative matrix factorization in polynomial feature space. *IEEE Transactions on Neural Networks*, 19(6):1090–1100, 2008.

[21] S. Burer and R. D. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical programming*, 103(3):427–444, 2005.

[22] D. Cai, X. He, J. Han, and T. S. Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1548–1560, 2010.

[23] T. Chen, H. Li, Q. Yang, and Y. Yu. General functional matrix factorization using gradient boosting. In *International Conference on Machine Learning*, pages 436–444. PMLR, 2013.

[24] S. Choi. Algorithms for orthogonal nonnegative matrix factorization. *Neural Networks IJCNN*, pages 1828–1832, 2008.

[25] M. Chu, F. Diele, R. Plemmons, and S. Ragni. Optimality, computation, and interpretation of nonnegative matrix factorizations. In *SIAM Journal on Matrix Analysis*. Citeseer, 2004.

[26] A. Cichocki, H. Lee, Y.-D. Kim, and S. Choi. Non-negative matrix factorization with α-divergence. *Pattern Recognition Letters*, 29(9):1433–1440, 2008.

[27] 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.

[28] 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.

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

[30] A. M. Darsono, C. C. Toh, S. Saat, A. A. M. Isa, N. A. Manap, and M. M. Ibrahim. β-divergence nonnegative matrix factorization on

biomedical blind source separation. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(2):1–4, 2017.

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

[32] 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.

[33] 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.

[34] F. Deutsch and F. Deutsch. *Best approximation in inner product spaces*, volume 7. Springer, 2001.

[35] D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems 16*, pages 1141–1148. MIT Press, 2004.

[36] J.-M. Dufour. Hilbert spaces. https://jeanmariedufour.github.io/ResE/Dufour_1999_C_TS_HilbertSpaces.pdf, 1999.

[37] R. L. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

[38] J. Eckstein and D. P. Bertsekas. On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1):293–318, 1992.

[39] J. Feng, X. Huo, L. Song, X. Yang, and 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):553–565, 2013.

[40] X.-R. Feng, H.-C. Li, J. Li, Q. Du, A. Plaza, and W. J. Emery. Hyperspectral unmixing using sparsity-constrained deep nonnegative matrix factorization with total variation. *IEEE Transactions on Geoscience and Remote Sensing*, 56(10):6245–6257, 2018.

[41] C. Févotte, N. Bertin, and J.-L. Durrieu. Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis. *Neural computation*, 21(3):793–830, 2009.

[42] 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.

[43] 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.

[44] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2014.

[45] N. Gillis. The why and how of nonnegative matrix factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, 12(257), 2014.

[46] N. Gillis. *Nonnegative matrix factorization*. SIAM, 2020.

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

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

[49] R. Gu, Q. Du, and S. J. Billinge. A fast two-stage algorithm for non-negative matrix factorization in streaming data. *arXiv preprint arXiv:2101.08431*, 2021.

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

[51] Y. Hachez. Convex optimization over non-negative polynomials: structured algorithms and applications. *Université Catholique de Louvain*, 2003.

[52] C. Hautecoeur, L. De Lathauwer, N. Gillis, and F. Glineur. Least-squares methods for nonnegative matrix factorization over rational functions. 2022.

[53] C. Hautecoeur and F. Glineur. Accelerating nonnegative matrix factorization over polynomial signals with faster projections. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019.

[54] C. Hautecoeur and F. Glineur. Nonnegative matrix factorization with polynomial signals via hierarchical alternating least squares. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 125–130, 2019.

[55] C. Hautecoeur and F. Glineur. Image completion via nonnegative matrix factorization using HALS and B-splines. In *28th European Symposium on Artificial Neural Networks-Computational Intelligence and Machine Learning (ESANN)*, pages 73–78, 2020.

[56] C. Hautecoeur and F. Glineur. Nonnegative matrix factorization over continuous signals using parametrizable functions. *Neurocomputing*, 416:256–265, 2020.

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

[58] C. Hautecoeur and F. Glineur. H-NMF: Nonnegative and constrained matrix factorization on Hilbert spaces; a unifying framework for NMF on signals. 2022.

[59] 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.

[60] D. Henrion and J. Malick. Projection methods in conic optimization. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 565–600. Springer, 2012.

[61] L. T. K. Hien and N. Gillis. Algorithms for nonnegative matrix factorization with the Kullback–Leibler divergence. *Journal of Scientific Computing*, 87(3):1–32, 2021.

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

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

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

[65] T.-Y. Ji, T.-Z. Huang, X.-L. Zhao, T.-H. Ma, and G. Liu. Tensor completion using total variation and low-rank matrix factorization. *Information Sciences*, 326:243–257, 2016.

[66] S. Jia and Y. Qian. Constrained nonnegative matrix factorization for hyperspectral unmixing. *IEEE Transactions on Geoscience and Remote Sensing*, 47(1):161–173, 2008.

[67] B. Jiang, T. Lin, S. Ma, and S. Zhang. Structured nonconvex and nonsmooth optimization: algorithms and iteration complexity analysis. *Computational Optimization and Applications*, 72(1):115–157, 2019.

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

[69] E. John and E. A. Yıldırım. Implementation of warm-start strategies in interior-point methods for linear programming in fixed dimension. *Computational Optimization and Applications*, 41(2):151–183, 2008.

[70] H. Kameoka, M. Nakano, K. Ochiai, Y. Imoto, K. Kashino, and S. Sagayama. Constrained and regularized variants of non-negative matrix factorization incorporating music-specific constraints. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5365–5368. IEEE, 2012.

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

[72] S. Khoshsokhan, R. Rajabi, and H. Zayyani. Sparsity-constrained distributed unmixing of hyperspectral data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(4):1279–1288, 2019.

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

[74] 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.

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

[76] T. Kimura and N. Takahashi. Global convergence of a modified HALS algorithm for nonnegative matrix factorization. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015 IEEE 6th International Workshop on*, pages 21–24. IEEE, 2015.

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

[78] V. Kumar and S. Minz. Feature selection: a literature review. *SmartCR*, 4(3):211–229, 2014.

[79] F. B. Lavoie, N. Braidy, and R. Gosselin. Including noise characteristics in MCR to improve mapping and component extraction from spectral images. *Chemometrics and Intelligent Laboratory Systems*, 153:40–50, 2016.

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

[81] V. Leplat, N. Gillis, and A. M. Ang. Blind audio source separation with minimum-volume beta-divergence NMF. *IEEE Transactions on Signal Processing*, 68:3400–3410, 2020.

[82] H.-C. Li, G. Yang, W. Yang, Q. Du, and W. J. Emery. Deep nonsmooth nonnegative matrix factorization network with semi-supervised learning for sar image change detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, 160:167–179, 2020.

[83] L. Li, G. Lebanon, and H. Park. Fast Bregman divergence NMF using Taylor expansion and coordinate descent. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 307–315, 2012.

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

[85] H. Liu, W. Wang, L. Xue, J. Yang, Z. Wang, and C. Hua. Speech enhancement based on discrete wavelet packet transform and Itakura-Saito nonnegative matrix factorisation. *Archives of Acoustics*, pages 565–572, 2020.

[86] Y. Liu, Z. Long, and C. Zhu. Image completion using low tensor tree rank and total variation minimization. *IEEE Transactions on Multimedia*, 21(2):338–350, 2018.

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

[88] Y. Lu, Z. Lai, Y. Xu, J. You, X. Li, and C. Yuan. Projective robust nonnegative factorization. *Information Sciences*, 364:16–32, 2016.

[89] P. Luo, X. Qu, L. Tan, X. Xie, W. Jiang, L. Huang, W. H. Ip, and K. L. Yung. Robust ensemble manifold projective non-negative matrix factorization for image representation. *IEEE Access*, 8:217781–217790, 2020.

[90] X. Luo, M. Zhou, Y. Xia, and Q. Zhu. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics*, 10(2):1273–1284, 2014.

[91] U. Marteau-Ferey, F. Bach, and A. Rudi. Non-parametric models for non-negative functions. *Advances in neural information processing systems*, 33:12816–12826, 2020.

[92] R. Melrose. Functional analysis lecture notes for 18.102. https://math.mit.edu/ rbm/18-102-S17/Chapter3.pdf, Mai 2017.

[93] G. R. Naik. *Non-negative Matrix Factorization Techniques*. Springer, 2016.

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

[95] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, 26(09):917–922, 1977.

[96] J. Nie. Regularization methods for sum of squares relaxations in large scale polynomial optimization. *Submitted for publication., September*, 2009.

[97] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[98] A. Pascual-Montano, J. M. Carazo, K. Kochi, D. Lehmann, and R. D. Pascual-Marqui. Nonsmooth nonnegative matrix factorization (nsNMF). *IEEE transactions on pattern analysis and machine intelligence*, 28(3):403–415, 2006.

[99] V. I. Paulsen and M. Raghupathi. *An introduction to the theory of reproducing kernel Hilbert spaces*, volume 152. Cambridge university press, 2016.

[100] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, et al. Recent advances in techniques for hyperspectral image processing. *Remote sensing of environment*, 113:S110–S122, 2009.

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

[102] J. Rapin, J. Bobin, A. Larue, and J.-L. Starck. NMF with sparse regularizations in transformed domains. *SIAM journal on Imaging Sciences*, 7(4):2020–2047, 2014.

[103] M. Razaviyayn, M. Hong, and Z.-Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

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

[105] 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.

[106] T. Sadowski and R. Zdunek. Image completion with smooth nonnegative matrix factorization. In *International Conference on Artificial Intelligence and Soft Computing*, pages 62–72. Springer, 2018.

[107] P. Sajda, S. Du, T. R. Brown, R. Stoyanova, D. C. Shungu, X. Mao, and 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):1453–1465, 2004.

[108] Y. E. Salehani and S. Gazor. Smooth and sparse regularization for NMF hyperspectral unmixing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(8):3677–3692, 2017.

[109] 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.

[110] T. Sano, T. Migita, and N. Takahashi. A novel update rule of HALS algorithm for nonnegative matrix factorization and Zangwill's global convergence. *Journal of Global Optimization*, pages 1–27, 2022.

[111] P. Seiler. Sosopt: A toolbox for polynomial optimization. *arXiv preprint arXiv:1308.1889*, 2013.

[112] Z. Shu, X.-J. Wu, C. You, Z. Liu, P. Li, H. Fan, and F. Ye. Rank-constrained nonnegative matrix factorization for data representation. *Information Sciences*, 528:133–146, 2020.

[113] 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.

[114] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *2010 IEEE international conference on data mining*, pages 1055–1060. IEEE, 2010.

[115] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad. A review of unsupervised feature selection methods. *Artificial Intelligence Review*, 53(2):907–948, 2020.

[116] D. L. Sun and C. Fevotte. Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6201–6205. IEEE, 2014.

[117] M. J. Todd, K.-C. Toh, and R. H. Tütüncü. On the Nesterov-Todd direction in semidefinite programming. *SIAM Journal on Optimization*, 8(3):769–796, 1998.

[118] A. Townsend and L. N. Trefethen. Continuous analogues of matrix factorizations. *Proc. R. Soc. A*, 471(2173):20140585, 2015.

[119] L. N. Trefethen. Householder triangularization of a quasimatrix. *IMA journal of numerical analysis*, 30(4):887–897, 2009.

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

[121] 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.

[122] A. Vandaele, N. Gillis, F. Glineur, and D. Tuyttens. Heuristics for exact nonnegative matrix factorization. *Journal of Global Optimization*, 65(2):369–400, 2016.

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

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

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

[126] Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789, 2013.

[127] Y. Xu, W. Yin, Z. Wen, and Y. Zhang. An alternating direction algorithm for matrix completion with nonnegative factors. *Frontiers of Mathematics in China*, 7(2):365–384, 2012.

[128] F. Yahaya, M. Puigt, G. Delmaire, and G. Roussel. Random projection streams for (weighted) nonnegative matrix factorization. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3280–3284. IEEE, 2021.

[129] J. Yan, B. Zhang, N. Liu, S. Yan, Q. Cheng, W. Fan, Q. Yang, W. Xi, and Z. Chen. Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing. *IEEE transactions on Knowledge and Data Engineering*, 18(3):320–333, 2006.

[130] S. Yang and M. Ye. Multistability of α-divergence based NMF algorithms. *Computers & Mathematics with Applications*, 64(2):73–88, 2012.

[131] Z. Yang, H. Zhang, Z. Yuan, and E. Oja. Kullback-Leibler divergence for nonnegative matrix factorization. In *International Conference on Artificial Neural Networks*, pages 250–257. Springer, 2011.

[132] Z. Yang, Y. Zhang, W. Yan, Y. Xiang, and S. Xie. A fast non-smooth nonnegative matrix factorization for learning sparse representation. *IEEE access*, 4:5161–5168, 2016.

[133] T. Yokota, R. Zdunek, A. Cichocki, and Y. Yamashita. Smooth nonnegative matrix and tensor factorizations for robust multi-way data analysis. *Signal Processing*, 113:234–249, 2015.

[134] T. Yokota, Q. Zhao, and A. Cichocki. Smooth parafac decomposition for tensor completion. *IEEE Transactions on Signal Processing*, 64(20):5423–5436, 2016.

[135] J. Yu, G. Zhou, A. Cichocki, and S. Xie. Learning the hierarchical parts of objects by deep non-smooth nonnegative matrix factorization. *IEEE Access*, 6:58096–58105, 2018.

[136] Y. Yuan, M. Fu, and X. Lu. Substance dependence constrained sparse NMF for hyperspectral unmixing. *IEEE Transactions on Geoscience and Remote Sensing*, 53(6):2975–2986, 2015.

[137] Z. Yuan and E. Oja. Projective nonnegative matrix factorization for image compression and feature extraction. In *Scandinavian Conference on Image Analysis*, pages 333–342. Springer, 2005.

[138] S. Zafeiriou and M. Petrou. Nonlinear non-negative component analysis algorithms. *IEEE Transactions on Image Processing*, 19(4):1050–1066, 2010.

[139] 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.

[140] 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.

[141] 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.

[142] D. Zhang, Z.-H. Zhou, and S. Chen. Non-negative matrix factorization on kernels. In *Pacific Rim International Conference on Artificial Intelligence*, pages 404–412. Springer, 2006.

[143] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou. Exploiting sparsity in the coefficient matching conditions in sum-of-squares programming using ADMM. *IEEE control systems letters*, 1(1):80–85, 2017.

[144] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 315–324, 2011.

[145] F. Zhu, P. Honeine, and M. Kallas. Kernel nonnegative matrix factorization without the curse of the pre-image-application to unmixing hyperspectral images. *arXiv preprint arXiv:1407.4420*, 2014.