Optimization algorithms for the tensor rank approximation problem: application to clustering in machine learning

Rima Khouja

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES FONDAMENTALES ET APPLIQUÉES

THÈSE DE DOCTORAT

Algorithmes d'optimisation pour le problème d'approximation des décompositions en rang tensoriel: application au clustering en apprentissage automatique

Rima KHOUJA

Inria d'Université Côte d'Azur

Présentée en vue de l'obtention du grade de docteur en Mathématiques
de l'Université Côte d'Azur
et Université Libanaise

Dirigée par : Bernard MOURRAIN, Directeur de Recherche, Inria, Sophia-Antipolis, France
Co-encadrée par : Houssam KHALIL, Maître de conférence, Université Libanaise, Liban
Soutenue le : 9 Juin 2022

Devant le jury, composé de :
Lieven DE LATHAUWER, Professeur, Katholieke Universiteit Leuven, Belgique
Lek-Heng LIM, Professeur, University of Chicago, États-Unis
Alessandra BERNARDI, Professeur associé, Università Di Trento, Italy
André GALLIGO, Professeur Émérite, Université Côte d'Azur, France
Jean-Claude YAKOUBSOHN, Professeur Émérite, Université de Toulouse, France

ALGORITHMES D'OPTIMISATION POUR LE PROBLÈME D'APPROXIMATION DES DÉCOMPOSITIONS EN RANG TENSORIEL: APPLICATION AU CLUSTERING EN APPRENTISSAGE AUTOMATIQUE

Optimization algorithms for the tensor rank approximation problem: application to clustering in machine learning

Rima KHOUJA

Jury :

Rapporteurs

Lieven DE LATHAUWER, Professeur, Katholieke Universiteit Leuven, Belgique
Lek-Heng LIM, Professeur, University of Chicago, États-Unis

Examinateurs

Alessandra BERNARDI, Professeur associé, Università Di Trento, Italy
André GALLIGO, Professeur Émérite, Université Côte d'Azur, France
Jean-Claude YAKOUBSOHN, Professeur Émérite, Université de Toulouse, France

---

Université Côte d'Azur

Rima KHOUJA

Algorithmes d'optimisation pour le problème d'approximation des décompositions en rang tensoriel: application au clustering en apprentissage automatique

Ce document a été préparé avec LaTeX2e et la classe `these-ISSS` version v. 2.10.

Impression : output.tex – 15/6/2022 – 12:18

Révision pour la classe : these-ISSS.cls,v 2.10 2020/06/24 14:16:37 mpelleau

Aux victimes de l'explosion du port de Beyrouth le 4 Août 2020 et
à toutes les victimes de la corruption au Liban

Algorithmes d'optimisation pour le problème d'approximation des décompositions en rang tensoriel: application au clustering en apprentissage automatique

Résumé

Les tenseurs sont une généralisation d'ordre supérieur des matrices. Ils apparaissent dans une myriade d'applications. La décomposition de rang de tenseur décompose le tenseur en une somme minimale de tenseurs simples de rang 1. En pratique, la présence de bruit dans les entrées du tenseur fait que le calcul d'une décomposition de petit rang approchée est plus pertinente que de son calcul exact. Ce problème est connu comme le problème d'approximation des décompositions en rang tensoriel. Dans cette thèse, nous étudions ce problème pour les tenseurs symétriques, c.à.d pour les tenseurs avec des entrées invariantes par les permutations d'indices. Nous considérons des tenseurs symétriques avec des valeurs complexes. Parsuite en utilisant le lien entre les tenseurs et les polynômes homogènes, ainsi que des techniques d'optimisation complexe, nous proposons une approche d'optimisation riemannienne et nous développons un algorithme de Newton riemannien et un algorithme de Gauss–Newton riemannien pour résoudre ce problème. Nous abordons également le problème de diagonalisation simultanée de matrices, qui est étroitement lié au problème de décomposition tensorielle. Nous considérons ce problème sous deux angles : la certification et l'approximation. Pour la première partie, nous développons une suite de type Newton à convergence quadratique locale, et nous proposons un teste de certification. Pour la deuxième partie, nous développons un algorithme de gradient conjugué riemannien qui calcule localement un faisceau de matrices simultanément diagonalisables approché. En combinant cet algorithme avec un problème linéaire de moindres carrés, nous introduisons un algorithme d'optimisation alterné qui calcule une approximation de la décomposition pour les tenseurs tridimensionnels, quand le rang d'approximation est supérieur à la dimension de deux premiers modes. Enfin, en se basant sur les deux approches : tenseurs symétriques et diagonalisation simultanée de matrices, nous abordons le problème de clustering en apprentissage automatique pour les modèles de mélanges de Gaussienne sphériques. Nous utilisons ces méthodes pour implémenter la méthode des moments, afin de fournir un bon point initial pour l'algorithme de maximisation de vraissemblance.

Mots-clés : Tenseurs, algorithmes d'optimisation, apprentissage automatique, clustering, optimisation riemannienne, diagonalisation simultanée de matrices, mélanges Gaussiènnes, optimisation complex, variétés différentielles.

Optimization algorithms for the tensor rank approximation problem: application to clustering in machine learning

Abstract

Tensors are higher order generalization of matrices. They appear in a myriad of applications. The tensor rank decomposition is to write the tensor as a minimal sum of simple rank-1 tensors. In practice, the presence of noise in the tensor's inputs means that computing an approximated low rank decomposition is more relevant than computing the exact tensor rank decomposition. This problem is known as the low rank tensor approximation problem. In this thesis, we study the low rank tensor approximation problem for symmetric tensors i.e. tensors with unchanged entries under any permutation of their indices. We consider symmetric tensors with complex values, and using the basic link between tensors and homogeneous polynomials, and techniques from complex optimization, we develop a Riemannian optimization approach proposing Riemannian Newton and Gauss–Newton algorithms to solve this problem. We also address the simultaneous matrix diagonalization problem, which is closely related to the tensor decomposition problem. Indeed, we consider this problem from two points of view: certification and approximation. For the first point, we develop a Newton-type sequence with local quadratic convergence, and we exhibit a certification test. For the second point, we develop a Riemannian conjugate gradient algorithm which approximates locally a pencil of matrices by a pencil of simultaneously diagonalizable matrices. Moreover, by combining this algorithm with a linear least-squares problem, we introduce an alternate optimization algorithm that approximates the decomposition of three-dimensional tensors with approximation rank larger than the first two mode dimensions. Finally, based on both approaches symmetric tensors and simultaneous diagonalization, we address the machine learning clustering problem for spherical Gaussian mixture models, where we use our developed methods to implement the method of moments, which provides an initial point for the expectation maximization algorithm.

Keywords: Tensors, optimization algorithms, machine learning, clustering, Riemannian optimization, simultaneous matrix diagonalization, Gaussian-mixtures, complex optimization, manifolds.

Remerciements

Je tiens d'abord à remercier mon directeur de thèse Bernard Mourrain. Merci à vous d'avoir accepté de m'encadrer tout au long de cette thèse, d'avoir écouté mes idées avec un esprit ouvert, pour l'accompagnement idéal que vous m'avez fourni, et pour les efforts de qualité que vous avez mis afin que cette thèse se déroule bien malgré les conditions particulières. Vous m'avez beaucoup apporté, grâce à vous j'ai pu découvrir le monde de la recherche, ce fut un grand plaisir de travailler avec vous, en espérant que nous continuerons à collaborer ensemble. Je tiens encore à remercier mon co-encadrant Houssam Khalil d'avoir me proposer l'opportunité de faire avec lui et Bernard Mourrain le stage de M2, c'était pour moi la passerelle vers le monde de la recherche et c'est grâce à ça que j'ai vécu cette précieuse expérience dans ma vie, merci d'avoir accordé votre confiance, et merci pour votre engagement. En plus, je veux remercier Mustapha Jazar le coordinateur du département de mathématiques à l'Université Libanaise.

Je veux également remercier les membres de mon jury de thèse. Merci à André Galligo d'avoir accepté d'être président de mon jury. Merci à Lieven De Lathauwer et Lek-Heng Lim pour avoir la gentillesse de rapporter ma thèse. Merci à mes deux examinateurs Alessandra Bernardi et Jean-Claude Yakoubsohn.

Je veux remercier de plus les chercheurs avec qui j'ai collaboré pendant ma thèse. Merci à Jean-Claude Yakoubsohn, c'était un plaisir d'élargir mes horizons de recherche pour que ça touche un domaine assez intéressant comme l'algèbre linéaire numérique. Merci Pierre-Alexandre Mattei pour toutes les discussions que nous avons eu à propos de machine learning et data science. En fait la collaboration avec vous n'a pas juste aboutit à un article mais aussi m'a encouragé à aller plus loin dans ce domaine et à faire une longue formation en data science.

Je remercie tous les membres de l'équipe Aromath à l'Inria. Merci Erick, Ahmad, Fatmanur, pour vos conseils scientifiques et humains. Merci Pablo, Tobias, Michelangelo, Lorenzo pour les bon moments conviviaux qu'on a partagé dans l'institut. Merci pour les sorties, pour nos discussions autour d'un bon café. Merci à Angelos, Laurent, Evelyne pour votre gentillesse et encouragement. Un grand merci à l'assistante de l'équipe Sophie Honnorat, pour son aide administrative, les discussions qu'on a eu, pour les conseils. Je remercie aussi les collègues que j'ai eu la chance de rencontrer à l'Inria : Hugo, Sidhant, Amine, Riham....

Des remerciements à ma famille, merci à ma mère pour son amour inconditionnel, je ne peux pas décrire ce que je ressens quand je vois ses yeux fiers. Merci à mon père pour ses conseils et d'avoir partager avec moi ses idées qui reflètent son esprit ouvert, ce qui a contribué à ma vision sur de nombreux sujets. Merci à ma grande soeur Nourane d'avoir supporté mes plaintes tout au long de ce doctorat avec patience et merci pour son soutien et amour. Merci à mes trois frères Mohamad, Ahmad, Omar. Ils réussissent toujours à me faire rire même aux instants les plus difficiles. Je remercie mon beau-frère Ahmad, je le considère comme un grand frère pour moi, et enfin je remercie de tout mon coeur mon neveu Boudi, après un long jour de travail il suf-

fit de me recharger de la joie et de l'espoir en regardant ses sourires et ses jolies yeux verts brillants.

Pour conclure ces remerciements, merci à tous mes amis et mes proches. Merci Rada (ma grande soeur et meilleure amie), Maysam (pour son soutien, d'avoir partagé avec moi sa sagesse grâce à ses conseils précieux). Merci Ali Zoebi, Mohamad Said et tous mes amis au Liban (en particulier Alya, Abir, Majida, Rouba, Abed), en France (en particulier Amina, Rabab, Arij, Tarek, Joe, Mohamad Khayri, Ali Darwich).

Pour conclure, le doctorat était pour moi une expérience très riche au niveau scientifique et humain, et je suis très contente de l'avoir vécu.

Table of contents

Contributions

CHAPTER 1

Introduction

Tensors are multidimensional arrays. They constitute a powerful tool from multilinear algebra and have a central role in many important applications. The roots of tensor computation can be traced back to the end of the nineteenth century in differential calculus [58, 174]. Then they became essential in many fields of applications in conjunction with the widespread use of big datasets, where they showed high capability in extracting hidden structures in the data, outperforming in this regard matrix-based methods. Mathematically speaking, tensors are more than simply a data structure. Indeed, if f is a linear map on a $\mathbb{K}$-vector space V onto another $\mathbb{K}$-vector space V', i.e. $f(\alpha_1 x_1 + \alpha_2 x_2) = \alpha_1 f(x_1) + \alpha_2 f(x_2),\ \forall x_1, x_2 \in V,\ \forall \alpha_1, \alpha_2 \in \mathbb{K}$, where the two vector spaces are of the same dimension then this map can be represented by a matrix of coordinates with respect to fixed basis on respectively V and V'. Similarly, f is said to be multilinear map from $V_1 \times \ldots \times V_d$ onto $\mathbb{K}$, where $V_1, \ldots, V_d$ are $\mathbb{K}$-vector spaces, if f is linear with respect to every variable x_k in $V_k,\ \forall k \in \{1, \ldots, d\}$, and thus as in the linear case the multilinear map f is represented by an array of coordinates, once the basis of V_k for $k \in \{1, \ldots, d\}$ have been fixed, such that the entries of this array depend on d indices. The *tensor product* of $V_1, \ldots, V_d$ denoted by $V_1 \otimes \ldots \otimes V_d$ is defined by universal property such that any multilinear map f on $V_1 \times \ldots \times V_d$ lift to a linear map on $V_1 \otimes \ldots \otimes V_d$. For simplicity, we will look at tensors as multidimensional arrays of data. The Canonical Polyadic Decomposition (CPD) of tensors is at the core of many applications such that Signal Processing and Machine Learning [51], [189], Sensor array processing [195], Chemometrics [34], Principal components analysis [114], and recently in Deep Learning [160, 88, 16]. It consists of expressing a given tensor as a sum of rank-1 indecomposables tensors. The tensor rank is by definition the smallest number of rank-1 tensors needed in the CPD to generate the tensor. The CPD is also known as the rank decomposition when the number of rank-1 components is equal to the rank of the tensor. In particular, symmetric tensors i.e. higher order generalization of symmetric matrices, can be decomposed as a linear combination of simple symmetric tensors of symmetric rank one. To illustrate the interest of tensor rank decomposition, let us present briefly two applications as motivational examples :

Example 1.0.1 – (Blind Source Identification). Blind source identification consists of recovering source signals from observed signals without knowing the recording environment. For instance, let us consider this problem for a sensor array consisting of $n > 2$ displaced but otherwise identical subarrays of l sensors i.e. $ln := I$ sensors in total. The array output can be described by the following model

$$X = AS^t + E,$$

where $A \in \mathbb{C}^{I \times r}$ is the global array response, $S \in \mathbb{C}^{m \times r}$ contains m snapshots of r sources, and E is an additive noise. Let $J_k \in \mathbb{C}^{l \times I}$ be a row-selection matrix such that $J_k X \in \mathbb{C}^{l \times m}$ is the k-th subarray for $k \in \{1, \ldots, n\}$. Let $\mathcal{X} \in \mathbb{C}^{l \times m \times n}$ be the tensor of frontal slices (see

Section 2.1) equal to the matrices $J_k X$. This tensor has a unique rank-r decomposition (except of the elementary indeterminacies of scaling and permutation)

$$\mathcal{X} \simeq \sum_{i=1}^{r} a_i^1 \otimes a_i^2 \otimes a_i^3,$$

with the factor matrices $L = [a_i^1]_{1\leq i\leq r} \in \mathbb{C}^{l\times r}$, $M = [a_i^2]_{1\leq i\leq r} \in \mathbb{C}^{m\times r}$, $N = [a_i^3]_{1\leq i\leq r} \in \mathbb{C}^{n\times r}$. The generic uniqueness of rank-r decomposition of $\mathcal{X}$ stays valid even when the system is underdetermined i.e. the number of sources exceeds the number of sensors. Using the rank decomposition of $\mathcal{X}$, $J_k A$ and the sources S can be identified as follows [187, 51]

$$J_k A = L\,\mathrm{diag}(a_{k,1}^3, \ldots, a_{k,r}^3),\ S = M.$$

Example 1.0.2 – (Gaussian mixtures). Suppose that we have a mixture of r Gaussian distributions with r spherical covariance matrices (i.e. each covariance matrix is equal to the identity matrix multiplied by a scalar) such that we aim to estimate the proportion w_i, the mean $\mu_i \in \mathbb{R}^n$ and the covariance matrix $\Sigma_i = \sigma_i^2 I_n$ of each Gaussian distribution within the mixture, for $i \in \{1, \ldots, r\}$. Assume that we have enough number of samples N that allows us computing significant statistics. The following theorem of Hsu-Kakade [107] can be used to implement an algorithm to find these latent variables.

Theorem 1.0.1. Assume $r \leq n$. Let

- *$\tilde{\sigma}^2$ be the smallest eigenvalue of $\mathbb{E}[(x - \mathbb{E}[x]) \otimes (x - \mathbb{E}[x])]$ and v a corresponding unit eigenvector,*
- *$M_1 = \mathbb{E}[(v^t(x - \mathbb{E}[x]))^2 x]$,*
- *$M_2 = \mathbb{E}[x \otimes x] - \tilde{\sigma}^2 I_n$,*
- *$M_3 = \mathbb{E}[x \otimes x \otimes x] - \sum_{i=1}^{n} (M_1 \otimes e_i \otimes e_i + e_i \otimes M_1 \otimes e_i + e_i \otimes e_i \otimes M_1)$,*

where $(e_i)_{1\leq i\leq n}$ denotes the canonical basis of $\mathbb{R}^n$, and $\mathbb{E}$ is the expectation (can also called the mean). Then $\tilde{\sigma}^2 = \sum_{i=1}^{r} \omega_i\, \sigma_i^2$ and

$$M_1 = \sum_{i=1}^{r} \omega_i\, \sigma_i^2\, \mu_i, \quad M_2 = \sum_{i=1}^{r} \omega_i\, \mu_i \otimes \mu_i, \quad M_3 = \sum_{i=1}^{r} \omega_i\, \mu_i \otimes \mu_i \otimes \mu_i.$$

This theorem tells us that the symmetric rank decomposition of the symmetric tensor M_3 contains information on the parameters w_i and μ_i that we want to find.

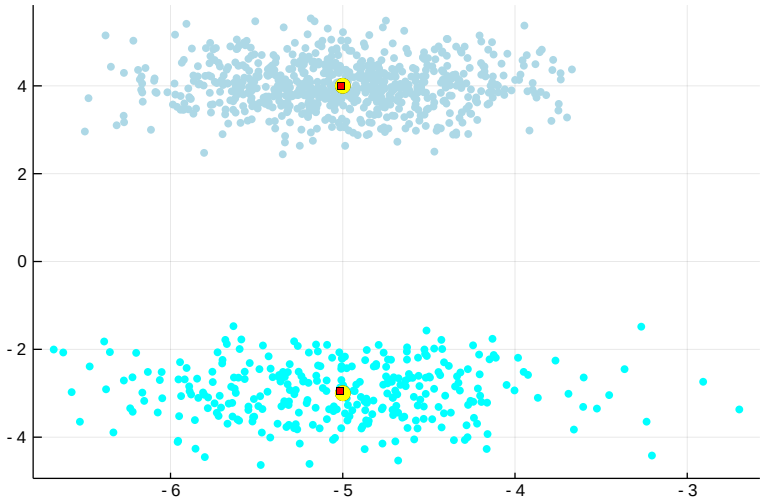


Figure 1.1 – Illustration of a Gaussian mixture in the plane with 2 clusters. The first cluster has mean $\mu_1 = (-5.0, -3.0)^t$, variance $\sigma_1^2 = 0.4$ and proportion $w_1 = 0.3$. The second cluster has $\mu_2 = (-5.0, 4.0)^t$, $\sigma_2^2 = 0.3$ and $w_2 = 0.7$. Number of samples $N = 1000$. The yellow circles represent the means of the clusters and the red squares are the estimated means by the method of moments that we will detail in Chapter 6.

We can notice in the first example the presence of noise in the model of the sensor array processing, and in the second example the symmetric tensor is built from empirical moments. In fact, this is often the case for many tensor decomposition applications, where the data from which the tensor is constructed is rarely free of noise. For this reason, the problem to solve in practice is to approximate the tensor by a tensor of low rank, rather than computing the exact rank decomposition. This consists of formulating the approximation problem as a cost function that measures the distance (in general by using the Frobenius norm) between the tensor to approximate and a tensor in the set of tensors with a bounded number of terms in their rank decompositions. In this thesis we first consider the low rank tensor approximation problem for symmetric tensors with complex coefficients, where we investigate the geometric structure of the constraint set and show how it is possible, when combining this with efficient computation tools, to develop concrete efficient approximation algorithms.

We then move to the simultaneous matrix diagonalization problem which is closely related to tensors. In fact, suppose that we have a pencil of matrices, a tensor of order three can be built by stacking the matrices of the pencil one after the other along the third dimension. Conversely, unfolding a three dimensional tensor along the third dimension yields a pencil of matrices (this can also generalized for higher order tensors). The rank decomposition of the tensor corresponds to a simultaneous diagonalization of the matrices in the pencil. The relation between these two representations is heavily exploited in the literature. We address the simultaneous matrix diagonalization problem as a certification and approximation problem and briefly address its connection to tensor rank approximation problem. Finally, we highlight the importance and the applicability of the proposed approaches throughout an interesting application in recovering hidden structure in spherical Gaussian mixtures.

In this thesis we use techniques from numerical optimization, tensor analysis, linear algebra, and differential geometry. Concretely, we utilize tools such as Riemannian optimization, manifolds, matrix decomposition, complex optimization, and several others.

We start with two new algorithms for the low rank tensor approximation problem for symmetric tensors with complex coefficients. The first algorithm is a Riemannian Newton algorithm with quadratic local convergence. The second algorithm is a Riemannian Gauss-Newton algorithm over Veronese manifolds. The local convergence of the aforementioned algorithm is not affected by large scale differences between rank-one components. The two algorithms show their competitiveness in comparison with other state-of-the-art methods in terms of accuracy and running time.

We then move to the simultaneous matrix diagonalization problem closely related to tensors, and discussing in this context the certification and the approximation problems. Indeed, for a pencil of simultaneously diagonalizable matrices, we introduce a Newton-type sequence that converges quadratically towards the solution when starting from an initial point verifying a sufficient condition. Moreover, we introduce a Riemannian conjugate gradient algorithm for the approximate simultaneous diagonalization of matrices problem, and use this algorithm to develop an algorithm for the low rank tensor approximation problem for three-dimensional tensors when the approximation rank is higher than the dimension of the two first modes.

Finally, we show how the studied approaches can be used in machine learning applications for instance in clustering problems. Mainly, when the dataset that we aim to cluster is obeying spherical Gaussian mixture distribution, we propose to apply the Expectation Maximization algorithm, with an initial point given by the method of moments and show the impact of this choice especially in term of accuracy in comparison with other state-of-the-art methods.

1.1 Context and literature review

In the sequel we state the main research axis that we will consider in this thesis equipped with an overview of some important related works.

Low rank symmetric tensor approximation problem A symmetric tensor T of order d and dimension n in $\mathcal{T}^d(\mathbb{C}^n) = \mathbb{C}^n \otimes \cdots \otimes \mathbb{C}^n$ is a special case of tensors, where its entries do not change under any permutation of its d indices. We denote their set by $\mathcal{S}^d(\mathbb{C}^n)$. The symmetric tensor decomposition problem consists of decomposing a symmetric tensor $T \in \mathcal{S}^d(\mathbb{C}^n)$ into a linear combination of symmetric tensors of rank one i.e.

$$T = \sum_{i=1}^{r} w_i \underbrace{v_i \otimes ... \otimes v_i}_{d \text{ times}}, \quad w_i \in \mathbb{C},\ v_i \in \mathbb{C}^n \tag{1.1}$$

For a multilinear tensor, its decomposition as a minimal sum of tensor products of vectors is called the Canonical Polyadic Decomposition [101]. We have a correspondence between $\mathcal{S}^d(\mathbb{C}^n)$ and the set of homogeneous polynomials of degree d in n variables denoted $\mathbb{C}[x_1, \ldots, x_n]_d =: \mathbb{C}[\mathbf{x}]_d$. Using this correspondence, (1.1) is equivalent to express the homogeneous polynomial $\boldsymbol{p}$ associated to T as a sum of powers of linear forms, which is by definition the classical Waring decomposition i.e.

$$\boldsymbol{p} = \sum_{i=1}^{r} w_i (v_{i,1}x_1 + \cdots + v_{i,n}x_n)^d, \quad w_i \in \mathbb{C},\ v_i \in \mathbb{C}^n \tag{1.2}$$

The smallest r such that this decomposition exists is by definition the symmetric rank of $\boldsymbol{p}$ denoted by $\mathrm{rank}_s(\boldsymbol{p})$. Let $d \geq 3$. The generic symmetric rank denoted by r_g, is given by Alexander–Hirschowitz theorem [10] as follows : $r_g = \lceil \frac{1}{n}\binom{n+d-1}{d} \rceil$ for all $n, d \in \mathbb{N}$, except for the following cases : $(d, n) \in \{(3, 5), (4, 3), (4, 4), (4, 5)\}$, where it should be increased by 1. We say that T is of subgeneric rank, if its rank $\mathrm{rank}_s(T) = r$ in (1.2) is strictly lower than r_g. In this case, a strong property of uniqueness of the Waring decomposition holds [47], and the symmetric tensor T is called identifiable, unless in three exceptions which are cited in [47, Theorem 1.1], where there are exactly two Waring decompositions. This identifiability property forms an important key strength of Waring decomposition. It can explain why this decomposition problem appears in many applications for instance in the areas of mobile communications, in blind identification of under-determined mixtures, machine learning, factor analysis of k-way arrays, statistics, biomedical engineering, psychometrics, and chemometrics. See e.g. [52, 56, 63, 190] and references therein. The decomposition of the tensor is often used to recover structural information in the application problem.

The Symmetric Tensor Approximation problem (STA) consists of finding the closest symmetric tensor to a given symmetric tensor $T \in \mathcal{S}^d(\mathbb{C}^n)$, of low symmetric rank. Equivalently, for a given $r \in \mathbb{N}^*$, it consists of approximating a homogeneous polynomial $\boldsymbol{p}$ associated to a symmetric tensor T by an element in Σ_r, where $\Sigma_r = \{\boldsymbol{q} \in \mathbb{C}[\mathbf{x}]_d \mid \mathrm{rank}_s(\boldsymbol{q}) \leq r\}$, i.e.

$$\text{(STA)} \quad \min_{\boldsymbol{q} \in \Sigma_r} \tfrac{1}{2}||\boldsymbol{p} - \boldsymbol{q}||_d^2.$$

Since in many problems, the input tensors are often computed from measurements or statistics, they are known with some errors on their coefficients and computing an approximate decomposition of low rank often gives better structural information than the exact or accurate decomposition of the approximate tensor [11, 13, 86].

For matrices, the best low rank approximation can be computed via Singular Value Decomposition (SVD). Higher Order Singular Value Decomposition (HOSVD) has been investigated to compute a multilinear rank approximation of a tensor [63, 64, 215], this, in contrast to the matrix case, does not give the best multilinear rank approximation (see for instance inequality (5) in [122]).

A classical approach for computing an approximate tensor decomposition of low rank is the so-called Alternating Least Squares (ALS) method. It consists of minimizing the distance between a given tensor and a low rank tensor by alternately updating the different factors of the tensor decomposition, solving a quadratic minimization problem at each step. See e.g. [42, 44, 93, 118]. This approach is well-suited for tensor represented in $\mathcal{T}^d(\mathbb{C}^n)$ but it looses the symmetry property in the internal steps of the algorithm. The space in which the linear operations are performed is of large dimension n^d compared to the dimension $\binom{n+d-1}{d}$ of $\mathcal{S}^d(\mathbb{C}^n)$ when n and d grow. Moreover the convergence is slow [78, 208].

Other iterative methods such as quasi-Newton methods have been considered for low rank tensor approximation problems to improve the convergence speed. See for instance [97, 162, 169, 183, 192, 206]. A Riemannian Gauss–Newton algorithm with trust region scheme was presented in [33], to approximate a given real multilinear tensor by one of low rank. The Riemannian optimization set is a Cartesian product of Segre manifolds (i.e. manifolds of real multilinear tensors of rank one). The retraction on the Segre manifold, called ST-HOSVD, is based on sequentially truncated HOSVD [122, 91, 215]. Moreover, an algorithm, called hot restarts, was introduced in [33] to avoid ill-conditioned decompositions. Closely related to these iterative methods, the condition number of join decompositions such as tensor decompositions is studied in [32].

Optimization techniques based on quasi-Newton iterations for block term decompositions of multilinear tensors over the complex numbers have also been presented in [191, 192]. In [183] quasi-Newton and limited memory quasi-Newton methods for distance optimization on products of Grassmannian manifolds are designed to deal with the Tucker decomposition of a tensor and applied for a low multilinear rank tensor approximation. In all these approaches, an approximation of the Hessian is used to compute the descent direction, and the local quadratic convergence cannot be guaranteed.

Specific investigations have been developed, in the case of best rank-1 approximation. The problem is equivalent to the optimization of a polynomial on the product of unitary spheres (see e.g. [64, 227]). Global polynomial optimization methods can be employed over the real or complex numbers, using for instance convex relaxations and semidefinite programming [155]. However, the approach is facing scalability issues in practice for large size tensors.

In relation with polynomial representation and multivariate Hankel matrix properties, another least square optimization problem is presented in [154], for low rank symmetric tensor approximation. Good approximations of the low rank approximation are obtained for small enough perturbations of low rank tensors. More recently, a method for decomposing real even-order symmetric tensors, called Subspace Power Method (SPM), has been proposed in [116]. It is based on a power method associated to the projection on subspaces of eigenvectors of the Hankel operators and has a linear convergence.

Simultaneous diagonalization of matrices Let us consider s *diagonalizable* matrices $M_1, \cdots, M_s$ in $\mathbb{C}^{n\times n}$ which pairwise commute. A classical result states that these matrices are simultaneously diagonalizable, i.e., there exists an invertible matrix E and diagonal matrices Σ_i, $1 \leqslant i \leqslant s$, such that $EM_iE^{-1} = \Sigma_i$, $1 \leqslant i \leqslant s$, see e.g. [105]. Our objective is to compute numerically a solution (E, F, Σ) of the system of equations

$$f(E, F, \Sigma) := \begin{pmatrix} FE - I_n \\ FME - \Sigma \end{pmatrix} = 0 \tag{1.3}$$

where $\Sigma = (\Sigma_1, \ldots, \Sigma_s)$ and $EMF - \Sigma := (EM_1F - \Sigma_1, \ldots, EM_sF - \Sigma_s)$. Notice that this system is multi-linear in the unknowns E, F, Σ. We verify that when $s = 1$ and M_1 is a generic matrix, this system has a solution set of dimension $2\,n^2 - n^2 - (n^2 - n) = n$. However, for $s > 1$ and generic matrices M_i, there is no solution. To have a solution, the pencil M must be on the manifold of s-tuples of simultaneously diagonalizable matrices.

The system (1.3) can be generalized to the following system :

$$f'(E, F, \Sigma') := \begin{pmatrix} FM_0E - \Sigma_0 \\ FME - \Sigma \end{pmatrix} = 0 \tag{1.4}$$

where $\Sigma' = (\Sigma_0, \Sigma_1, \ldots, \Sigma_s)$, $M_0 \in \mathbb{C}^{n\times n}$ is replacing I_n and Σ_0 is a diagonal matrix replacing I_n in the first equation of (1.3). When the pencil $M' = (M_0, M_1, \ldots, M_s)$ contains an invertible matrix, the solutions of the two systems are closely related. If M_0 is invertible, a solution (E, F, Σ') of (1.4) for $M' = (M_0, M_1, \ldots, M_s)$ gives the solution $(FM_0, E\Sigma_0^{-1}, \Sigma\Sigma_0^{-1})$ of (1.3) for $M = (M_0^{-1}M_1, \ldots, M_0^{-1}M_s)$. A similar correspondence between the solution sets can be obtained if a linear combination $M_0' = \sum_{i=1}^s \lambda_i M_i$ is invertible.

As (1.4) can be seen as an homogeneization of (1.3) and appears in several contexts and applications, we will also study Newton-type methods for this homogenized system.

To solve the system of equations (1.3), we propose to apply a Newton-like method and to analyze the Newton map associated to an iteration. These ideas have also been developed in the literature, for instance, in a technical report for the fast computation of the singular value decomposition [103], in [143] where a Newton method is used for the symmetric eigenvalue problem.

We say that we have a quadratic sequence associated to a system of equations if the sequence converges quadratically towards a solution.

The classical Newton map defines $(E+X, F+Y, \Sigma+S)$ from (E, F, Σ) in order to cancel the linear part in the Taylor expansion of $f(E+X, F+Y, \Sigma+S)$. An easy computation shows that the perturbations X, Y and S are solutions of such a Sylvester-type linear system

$$\begin{pmatrix} FE - I_n + FX + YE \\ FME - \Sigma - S + XMF + EMY \end{pmatrix} = 0. \tag{1.5}$$

The technical background to solve this linear system is the Kronecker product, see [104]. In this way, the size of the linear system that one needs to invert is n^2.

The construction of the methods studied here is based on perturbations of such type $(E(I_n + X), (I_n + Y)F, \Sigma + S)$ rather than $(E+X, F+Y, \Sigma+S)$. More precisely the perturbations X, Y and S that we consider are perturbations which cancel the linear part of the Taylor expansion of $f(E(I_n+X), (I_n+Y)F, \Sigma+S)$. In this case, we can produce explicit solutions for the linear system in X, Y and S given by :

$$\begin{pmatrix} Z + X + Y \\ \Delta - S + \Sigma X + Y\Sigma \end{pmatrix} = 0. \tag{1.6}$$

where $Z = FE - I_n$ and $\Delta = FME - \Sigma$. We will see that the linear system (1.6) admits an explicit solution (X, Y, S) with respect to Z and Δ for $s = 1, 2$ in (1.3). This is because Σ is a diagonal matrix. From these considerations, we define and analyze a sequence that converges quadratically towards a solution of the system (1.3) without inverting a linear system at each step of this Newton-like method.

Simultaneous matrix diagonalization is required by many algorithms as it was pointed out in [36]. There is quite a body of literature on exactly or approximately commuting matrices and exact or approximate joint diaonalization, see for instance [121, 211]. A numerical analysis for two normal commuting matrices is proposed in [37] using Jacobi-like methods. Their method adjusts the classical Jacobi method in successively solving $\frac{n(n-1)}{2}$ two-real-variables optimization problems at each sweep of the algorithm. Their main result states a local quadratic convergence and can be summarized in the following way. Let $\text{off}_2(A,B)^2 = \sum_{i \neq j} |A_{i,j}|^2 + |B_{i,j}|^2$. Let $\{\alpha_1, \ldots, \alpha_n\}$ (resp. $\{\beta_1, \ldots, \beta_n\}$) be the set of the eigenvalues of A (resp. B). Let A^k and B^k the matrices obtained at the step k of the Jacobi-like method and $\rho_k = \text{off}_2(A^k, B^k)$. If

$$\rho_0 < \frac{1}{2}\delta := \frac{1}{4} \min_{i \neq j} \left(|\alpha_i - \alpha_j|, |\beta_i - \beta_j| \right),$$

then

$$\rho_{k+1} < 2n(9n-13)\, \frac{\rho_k^2}{\delta}.$$

We will see in Theorems 5.1.5 and 5.1.8 that the local conditions of the quadratic convergence do not depend on n. Many other papers studied the so-called Jacobi-like methods (see e.g. [139], [146] and references therein).

In [102] a sequence with proof of its convergence towards a numerical solution of the system (1.3) when $s = 1$ i.e. for M_1, with the assumption of M_1 being a diagonalizable matrix, is presented. It requires matrix inversion. Furthermore, under some extra assumptions, its quadratic convergence is established.

Simultaneous matrix diagonalization appears in many applications. For instance, in the solution of multivariate polynomial equations by algebraic methods, the isolated roots of the system are obtained from the computation of common eigenvectors of commuting operators of multiplication in the quotient ring and from their eigenvalues [57], [77]. In the case of simple roots, this reduces to simultaneous diagonalization of a pencil of matrices. Further, simultaneous matrix diagonalization is used for blind source separation, direction of arrival estimation, multi-dimensional harmonic retrieval, Canonical Polyadic Decomposition (CPD), econometric (see e.g. [27], [212], [89], [65], [24] and references therein).

The approximate simultaneous diagonalization problem aims to approximate locally a pencil of matrices to a pencil of simultaneously diagonalizable matrices. This problem is widely studied in the literature for a pencil of real *symmetric* matrices $C = (C_1, \ldots, C_s)$, in particular several algorithms based on Riemannian optimization methods (see [2]) have been developed in order to find an *approximate joint diagonalizer* for the pencil C (see e.g. [25, 1, 171, 113]). The idea is to find a local minimizer $B \in \mathbb{R}^{n\times n}$ of an objective function f which measures the degree of non-diagonality of the pencil $(BC_1B^T, \ldots, BC_sB^T)$ over a Riemannian manifold (see [219, 25, 7] for some examples of objective functions). This Riemannian manifold is defined according to the geometric constraints considered on B. For instance, the diagonalizer is supposed to be orthogonal in some of these algorithms after a pre-whitening step (see e.g. [40, 41, 80, 171, 74, 113, 156, 157]). Due to inaccuracies in the computation of the diagonalizer with orthogonality constraints (see. [224]), *oblique* constraints, i.e. all the rows of the diagonalizer have unit Euclidean norm, have also been considered instead of the former constraints in more recent works (see e.g. [1, 25]).

The approach of approximate joint diagonalizer for a pencil of real *symmetric* matrices is used to solve Blind Source Separation (BSS) problem, with potential applications in wide domains of engineering (see e.g. [55]).
In more general context, when the matrices of the pencil M to approximate are general square matrices, there exists algorithms to find an invertible matrix E such that $(EM_1E^{-1}, \ldots, EM_sE^{-1})$ is the most diagonal. The majority of these algorithms is based on Jacobi-like method (see e.g. [141, 85, 110]). Nevertheless, some other approaches have been addressed this problem. In [218] the authors split the optimization problem into a sequence of simpler second order subproblems, and present an algorithm that works with no restriction on the transformation matrix E. More recently, the authors in [9] present an algorithm based on two main steps. The first step approximates the pencil to nearly simultaneously diagonalizable pencil of matrices and this by solving a structured low-rank approximation problem. The second step computes a transformation matrix that diagonalizes exactly the pencil of simultaneously diagonalizable matrices obtained from the first step. One advantage for this approach over the other optimization methods regarding this problem, is that it has a guaranty to find an exact common diagonalizer if the pencil to approximate is already simultaneously diagonalizable.

Simultaneous matrix diagonalization of pencils of matrices appears in the Canonical Polyadic Decomposition (CPD) of tensors [60]. Under certain conditions the rank decomposition (i.e. the CP decomposition with number of rank-1 simple tensors equal to the rank of the tensor), is unique

[188]. In this case simultaneous matrix diagonalization allows to compute the rank decomposition. Direct methods based on simultaneous diagonalisation of matrices built from slices of tensors have been investigated for 3rd order multilinear tensors, e.g. in [93, 179, 134, 73, 59] or for multilinear tensors of rank smaller than the lowest dimension in [60, 139, 65]. For the low rank tensor approximation problem (i.e. approximation of the tensor to CP decomposition of lower rank) via simultaneous matrix diagonalization see for instance [65, 176]. In his proof on lower bounds of tensor ranks, Strassen showed in [203, Theorem 4.1] that a 3rd order multilinear tensor is of rank r if it can be embedded into a tensor with slices of rank r matrices, which are simultaneously diagonalizable. The CPD of tensors plays a crucial role in numerous applications such that Psychometric [43], signal processing and machine learning [51], [189], sensor array processing [195], arithmetic complexity [38], wireless communications [197], multidimensional harmonic retrieval [193], [194], Chemometrics [34], and Principal components analysis [114].

Tensor decomposition for learning Gaussian mixtures from moments With the relatively recent evolutions of information systems over the last decades, many observations, measurements, data are nowadays available on a variety of subjects. However, too much information can kill the information and one of the main challenges remains to analyse and to model these data, in order to recover and exploit hidden structures.

To tackle this challenge, popular Machine Learning technologies have been developed and used successfully in several application domains (e.g. in image recognition [99]). These techniques can be grouped in two main classes : Supervised machine learning techniques are approximating a model by optimising the parameters of an enough general model (e.g. a Convolution Neural Network) from training data. Unsupervised machine learning techniques are deducing the parameters characterising a model directly from the given data, using an apriori knowledge on the model. The supervised approach requires annotated data, with a training step that can introduce some bias in the learned model. The unsupervised approach can be applied directly on a given data set avoiding the costly step of annotating data, but the quality of the output strongly depends on the type of models to be recovered.

We consider the latter approach and show how methods from effective algebraic geometry help finding hidden structure in data that can be modelled by mixtures of Gaussian distributions. The algebraic-geometric tool that we consider is tensor decomposition. It consists in decomposing a tensor into a minimal sum of rank-1 tensors. This decomposition generalises the rank decomposition of a matrix, with specific and interesting features. Contrarily to matrix rank decomposition, the decomposition of a tensor is usually unique (up to permutations and scaling) when the rank of the tensor, that is the minimal number of rank-1 terms in a decomposition, is small compared to the dimension of the space(s) associated to the tensor (see for instance [46, 48, 47]). Such a tensor is called *identifiable*. This property is of particular importance when the decomposition is used to recover the parameters of a model. It guaranties the validity of the recovering process and its convergence when the number of data increases.

In [107], symmetric tensor decompositions for moment tensors are studied for spherical Gaussian mixtures. Moment methods have been further investigated for Latent Dirichlet Allocation models, topic or multiview models in [13, 111]. In [178], a tensor decomposition technique based on Alternate Least Squares (ALS) is used to initialise the Expectation Maximisation (EM) algorithm [96, 151, 222], for a mixture of discrete distributions (which are not Gaussian distributions). An overview of tensor decomposition methods in Machine Learning can be found in [170].

1.2 Contributions

In this section we summarize our main contributions in this thesis.

Low rank symmetric tensor approximation problem In Chapter 4, we present two Riemannian Newton-type algorithms for the low rank tensor approximation problem (STA) for symmetric tensors with complex coefficients.

- The first algorithm is a Riemannian Newton algorithm (Section 4.2). We use the parametrization of the set of tensors of rank at most r by weights and vectors on the unit sphere. Exploiting the properties of the apolar product on homogeneous polynomials combined with efficient tools from complex optimization, we provide an explicit and tractable formulation of the Riemannian gradient and Hessian, leading to Newton iterations with local quadratic convergence. We prove that under some regularity conditions on non-defective tensors in the neighborhood of the initial point, the iteration (completed with a trust-region scheme) is converging to a local minimum (Proposition 4.2.13).

- The second algorithm is a Riemannian Gauss–Newton method on the Cartesian product of the manifolds of symmetric rank-1 tensors called Veronese manifolds (Section 4.2.2). We describe an explicit orthonormal basis of the tangent space of this Riemannian manifold. We use this basis to obtain the Riemannian gradient and the Gauss–Newton approximation of the Riemannian Hessian. We present an approximation method for a given homogeneous polynomial in $\mathbb{C}[\mathbf{x}]_d$ into linear form to the d^{th} power, based on the rank-1 truncation of the SVD of Hankel matrix associated to the homogeneous polynomial. From this approximation method, we propose a new retraction operator on the Veronese manifold. The design of the algorithm depends on the geometry of the Veronese manifold and its tangent space. In this context, the Riemannian Gauss–Newton iteration that we describe is adapted to the symmetric setting by considering the reduced vector space $\mathbb{C}[\mathbf{x}]_d$ and by exploiting the apolar identities. In our approach we consider symmetric tensors with complex coefficients. The constraint set is parameterized via the complex Veronese manifolds, which leads us to a complex optimization problem with geometric constraints, and this, to the best of our knowledge, has not been addressed previously in tensor approximation.

- We analyze the numerical behavior of these methods, choosing for the initial point the approximate decomposition provided by the Simultaneous Matrix Diagonalisation (SMD) of a pencil of Hankel matrices [92, 150] (Section 4.3.1). Numerical experiments (Section 4.3) show the good numerical behavior of the new methods for the best rank-1 approximation of real-valued symmetric tensors, for low rank approximation of sparse symmetric tensors, and against perturbations of symmetric tensors of low rank. Comparisons with existing state-of-the-art methods corroborate this analysis.

Simultaneous diagonalization of matrices In Chapter 5, we investigate the simultaneous diagonalization of matrices from three points of view : Certification, approximation, and relation with tensor rank decomposition.

- In Section 5.1, our contributions are a new iteration for the simultaneous diagonalization of matrices, with a local quadratic convergence and its analysis. The iteration is different

from a Newton iteration. It does not require to invert a large linear system, but performs simple matrix operations. We analyse the numerical behavior of the method and provide a certification test for the convergence. Sections 5.1.2, 5.1.3, 5.1.4, and 5.1.5 are devoted to respectively constructing a sequence to solve numerically :
— $FE - I_n = 0$,
— the system (1.3) when $s = 1$,
— the system (1.4) when $s = 1$,
— the system (1.3) for any s.
Moreover, we provide for these cases, a certification that the sequence converges to a nearby solution, and a test to detect when this convergence is quadratic from an initial point. More precisely, in Section 5.1.3 we show that a triplet (E_0, F_0, Σ_0) must satisfy a property depending on the quantity $\varepsilon_0 := \max(\kappa_0^2 K_0^2 \|Z_0\|, \kappa_0^2 K_0 \|\Delta_0\|)$ to get a quadratic convergence where
1– $Z_0 = E_0 F_0 - I_n$,
2– $\Delta_0 = E_0 M F_0 - \Sigma_0$,
3– $\kappa_0 = \max\left(1, \max\limits_{1 \leqslant j < k \leqslant n} \dfrac{1}{|\sigma_{0,k} - \sigma_{0,j}|}\right)$,
4– $K_0 = \max_k \left(1, |\sigma_{0,k}|\right)$,
such that $\sigma_{0,1}, \ldots, \sigma_{0,n}$ denote the diagonal entries of Σ_0. The quantity κ is the condition number of the studied methods. Based on the same methodology of Section 5.1.3, in Sections 5.1.4 and 5.1.5 we exhibit a certification of the convergence of the sequence constructed to the studied case towards the solution with a sufficient condition on the initial point. In Section 5.1.6 we perform numerical experimentation to corroborate the theoretical analysis.

- In Section 5.2, we address the problem of approximate simultaneous diagonalization of matrices. We consider a pencil of square matrices $M = [M_1, \ldots, M_s]$ without imposing any restrictions on the matrices to be diagonalized approximately, regarding their symmetry or definiteness. Unlike the common familiar case where one transformation or diagonalizer E is considered, we look in more general context, for two transformation matrices E and F that diagonalize approximately the pencil M in such a way that FM_kE^t is the most diagonal for $k \in \{1, \ldots, s\}$. This structure allows for an implementation of an algorithm for the tensor rank approximation problem of three-dimensional tensors with approximation rank higher than the first two dimensional sizes (Section 5.3). The approximation problem is formulated as a minimization problem of a cost function that measures the norm of the off-diagonal terms of the pencil's matrices. Moreover, restrictions on E and F are imposed to be in the oblique manifold in order to avoid undesirable singular points. Hence, the problem is formulated as a non-linear least squares problem over the Cartesian product of two oblique manifolds. We develop a Riemannian conjugate gradient algorithm to solve this problem. To develop our algorithm, we essentially use the framework presented recently in [25] regarding the Riemannian gradient-based optimization methods over the oblique manifold.

- In Section 5.3, we present a new algorithm for the low rank tensor approximation problem of trilinear tensors of size (n_1, n_2, n_3) with approximation rank $r \geq \max(n_1, n_2)$. We consider the pencil of matrices associated to the tensor to be approximated that contains

the slices of this tensor obtained by the flattening according to the third mode. Then we extend the matrices of this pencil of size $n_1 \times n_2$ to matrices of size $r \times r$. The algorithm then works on the pencil of the extended matrices. At each iteration, the algorithm alternates between two steps. The first step finds E and F that diagonalize approximately the extended pencil, and this by using the Riemannian conjugate gradient algorithm for approximate simultaneous diagonalization of matrices (Section 5.2). The second step updates the entries of the extdended part in each matrix of the pencil by solving a linear least-squares problem. At the end an approximated rank-r decomposition is obtained for an extended tensor of size (r, r, n_3), from which we extract an approximated rank-r decomposition for the original tensor of size (n_1, n_2, n_3).

Tensor decomposition for learning Gaussian mixtures from moments Chapter 6 is the chapter where we show one application of symmetric tensors in machine learning, by addressing the clustering problem of a dataset following a spherical Multivariate Gaussian Mixture (MGM) distribution. Indeed, this done by using the method of moments.

It has been shown in [47] that for symmetric tensors, if the rank of the tensor is strictly less than the rank r_g of a generic tensor of the same size, then the tensor is generically identifiable, except in three cases. We show in Section 6.2.5 a more specific result : for a symmetric tensor T having a decomposition with r points, if the Hankel matrix associated to T in a degree strictly bigger than the degree of interpolation of the r points is of rank r, then the tensor is identifiable. We show in Section 6.2.3, that under some assumption on the spherical Gaussian mixtures, a tensor of moments of order 3 of the distribution is identifiable and its decomposition allows to recover the parameters of the Gaussian mixture.
For symmetric tensor decomposition, a method based on flat extension of Hankel matrices or commutation of multiplication operators has been proposed in [30] and extended to multi-symmetric tensors in [18]. This approach is closely related to the simultaneous diagonalisation of tensor slices, but follows a more algebraic perspective. Eigenvectors of symmetric tensors have been used to compute their decompositions in [158]. In [92], Singular Value Decomposition and eigenvector computation are used to decompose a symmetric tensor, when its rank is smaller than the smallest size of its Hankel matrix in degree less than half the order of the tensor. In Section 6.2, we describe a new algorithm, involving Singular Value Decomposition and simultaneous diagonalisation, to compute the decomposition of an identifiable tensor, which interpolation degree is smaller that half the order of the tensor. In Section 6.3, we apply the method of moments [107] for recovering Gaussian mixtures and show, throughout examples on synthetic and real datasets, its impact on providing good initialisation point in the Expectation Maximization algorithm, in comparison with other state-of-the-art approaches.

1.3 Main notation

In this section we introduce some of the notation we use throughout the thesis. The superscripts $.^t$, $.^*$ and $.^{-1}$ are used respectively for the transpose, Hermitian conjugate, and the inverse matrix. Let $M \in \mathbb{C}^{m\times n}$, we denote by $\|A\|$ the Frobenius norm of A given by $\|M\| := \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|m_{i,j}|^2}$. Let $A \in \mathbb{C}^{n\times n}$, We denote by $\sqrt{A}$ a matrix $B \in \mathbb{C}^{n\times n}$ such that $A = B^2$. The complex conjugate is denoted by an overbar, e.g., $\bar{w}$. We use parentheses to denote vectors e.g. $W = (w_i)_{1\leq i\leq r}$, and the square brackets to denote matrices e.g. $V = [v_i]_{1\leq i\leq r}$ where v_i

are column vectors. The concatenation of vectors $v_1, v_2, \ldots$ is denoted $(v_1; v_2; \ldots)$. The trace of a matrix A is denoted by $\mathrm{tr}(A)$. For a matrix $M \in \mathbb{C}^{n\times n}$, let $\mathrm{ddiag}(M)$ be the diagonal matrix with the same diagonal as M and let $\mathrm{off}(M)$ be the matrix where the diagonal term of M are replaced by 0. We have $M = \mathrm{ddiag}(M) + \mathrm{off}(M)$.We say that M is an off-matrix if $M = \mathrm{off}(M)$. In addition, $\mathrm{diag(M)}$ returns a vector containing the diagonal entries of M, and for $(\lambda_1, \ldots, \lambda_n) \in \mathbb{C}^n$, $\mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ is the diagonal matrix in $\mathbb{C}^{n\times n}$ of diagonal entries $\lambda_1, \ldots, \lambda_n$, and for a family of matrices $(A_i)_{1\leq i\leq n}$, $\mathrm{diag}(A_i)_{1\leq i\leq n}$ is the matrix with diagonal blocks A_i.

1.4 Organization of the thesis

The thesis is divided in three parts : *Preliminaries*, *Contributions*, and *Conclusions and Perspectives*.

Preliminaries

This part contains the background needed in the contributions part, with no original contributions.

- **Chapter 2 :** We introduce a review on tensors given the main notion and information in this regard.
- **Chapter 3 :** We introduce the necessary concepts of Riemannian optimization.

Contributions

This part contains our contributions in this thesis.

- **Chapter 4 :** This chapter contains our contributions to *low rank symmetric tensor approximation problem.*
  - — **Section 4.1 :** We describe the set of non-defective rank-r symmetric tensors.
  - — **Section 4.2 :** This section contains the Riemannian Newton algorithm and the Riemannian Gauss–Newton algorithm for the STA problem
  - — **Section 4.3 :** This section carries out numerical experiments.
  - — **Section 4.3 :** This section contains our conclusions in this chapter.
- **Chapter 5 :** This chapter contains our contributions to *simultaneous matrix diagonalization problem*.
  - — **Section 5.1 :** This section contains the Newton-type methods for simultaneous matrix diagonalization problem.
  - — **Section 5.2 :** This section contains the Riemannian conjugate gradient algorithm for the approximate simultaneous diagonalization of matrices problem.
  - — **Section 5.3 :** This section contains the algorithm for the low rank tensor approximation problem for real three-dimensional tensors with approximation rank larger than the size of the first two modes.
  - — **Section 5.4 :** This section contains our conclusions in this chapter.
- **Chapter 6 :** This chapter contains our contributions to *tensor decomposition for learning Gaussian mixtures from moments*.
  - — **Section 6.1 :** In this section we review Gaussian mixtures and moments methods.
  - — **Section 6.2 :** This section contains an algebraic symmetric tensor decomposition method for identifiable tensors.

— **Section 6.3 :** This section contains numerical examples for simulations on synthetic and real datasets.

Conclusions and Perspectives

This part contains the chapter of our conclusions and perspectives.

- **Chapter 7 :** We summary the main results of the thesis and give some perspectives and open questions.

1.5 Publications

- The contributions of Chapter 4 were a joint work with Bernard Mourrain and Houssam Khalil appeared as a journal paper in : Rima Khouja, Houssam Khalil, Bernard Mourrain. Riemannian Newton optimization methods for the symmetric tensor approximation problem. Linear Algebra and its Applications. Volume 637, 2022, pages 175-211, ISSN 0024-3795, https://doi.org/10.1016/j.laa.2021.12.008.

- The contributions of Section 5.1 were a joint work with Jean-Claude Yakoubsohn and Bernard Mourrain, the content is in the paper : Rima Khouja, Bernard Mourrain, Jean-Claude Yakoubsohn. Newton-type methods for simultaneous matrix diagonalization. In preparation, 2022, submitted to Calcolo journal, and it is under revision. https://hal.archives-ouvertes.fr/hal-03390265.

- The contributions of Chapter 6 were a joint work with Pierre-Alexandre Mattei and Bernard Mourrain, the content is in the paper : Rima Khouja, Pierre-Alexandre Mattei, Bernard Mourrain. Tensor decomposition for learning Gaussian mixtures from moments. Accepted for publication in the special issue on Algebraic Geometry and Machine Learning of journal of Symbolic Computation. https://hal.archives-ouvertes.fr/hal-03244448.

Preliminaries

CHAPTER 2

Tensors

Tensors can be seen as a generalization of matrices to multiple dimensions. Formally, a tensor of order d is an element of the tensor product of d vector spaces. It can be represented by a multidimensional array of numerical values from a field like $\mathbb{R}$ or $\mathbb{C}$ with respect to fixed basis on the vector spaces. Indeed, a scalar is a tensor of order zero, a vector is a tensor of order one and a matrix is a tensor of order two.

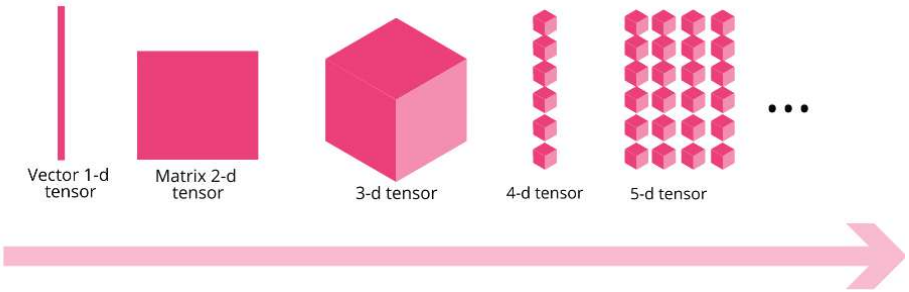


Figure 2.1 – Illustration of tensors of order $1, 2, 3, 4$ and 5.

Researchers pay high attention to tensors due to their applications in wide fields such as signal processing, numerical linear algebra, computer vision, numerical analysis, neuroscience, chemometrics, linguistics and more. In particular, the popularity of data science and machine learning has grown considerably during the last decade, and tensors have expanded to these domains to provide the required high computing capacities. For instance, tensors appear in the name of the popular Google's machine learning package "Tensorflow", since they are counted from the main tools used in this package, where they provide a convenient data format that allows applying very efficient operations in order to extract important information within the dataset and also to build strong predictive models. There is a number of comprehensive and important reviews concerning tensors and their applications. We mention for instance : Kolda et al. [118], Bro [34], Sidiropoulos et al. [189], Cichoki et al. [51], and Rabanser et al. [170].

2.1 Introduction to tensors

Unless otherwise stated, in this chapter tensors are assumed real valued.
Let $\mathbb{R}^{n_1} \otimes \ldots \otimes \mathbb{R}^{n_d}$ denotes the vector space of the tensor product of $\mathbb{R}^{n_1}, \ldots, \mathbb{R}^{n_d}$. The vector space $\mathbb{R}^{n_1} \otimes \ldots \otimes \mathbb{R}^{n_d}$ is defined by the universal property [28] such that any multilinear map f on $\mathbb{R}^{n_1} \times \ldots \times \mathbb{R}^{n_d}$ lift to a linear map on $\mathbb{R}^{n_1} \otimes \ldots \otimes \mathbb{R}^{n_d}$.
An element $\mathcal{A}$ in the vector space of d-way arrays denoted by $\mathbb{R}^{n_1 \times \ldots \times n_d}$ can be written in the

form of an array $\mathcal{A} = [a_{i_1,\ldots,i_d}]_{1\leq i_1\leq n_1,\ldots,1\leq i_d\leq n_d}$, such that $a_{i_1,\ldots,i_d} \in \mathbb{R}$ is the $(i_1,\ldots,i_d)$-entry of the array.
Let u, v be two vectors in respectively $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$. Their *outer product* is defined by :

$$u \otimes v = [u_{i_1} u_{i_2}]_{1\leq i_1\leq n_1, 1\leq i_2\leq n_2}.$$

This produces a matrix i.e. a 2-way array, thus the extension of this notion to the outer product of d vectors yields a d-way array, and more generally the outer product of a d-way array with a k-way array produces a $d+k$-way array. The so-called *Segre map* given by

$$\begin{aligned} \varphi : \mathbb{R}^{n_1} \times \ldots \times \mathbb{R}^{n_d} &\to \mathbb{R}^{n_1\times\ldots\times n_d} \\ (a^1,\ldots,a^d) &\mapsto a^1 \otimes \ldots \otimes a^d := [a^1_{i_1}\ldots a^d_{i_d}]_{1\leq i_1\leq n_1,\ldots,1\leq i_d\leq n_d}, \end{aligned} \tag{2.1}$$

is a multilinear map, thus by the universal property of the tensor product [28] there exists a unique linear map θ such that the following diagram commutes :

$$\begin{array}{ccc} & & \mathbb{R}^{n_1} \otimes \ldots \otimes \mathbb{R}^{n_d} \\ & \nearrow & \downarrow \theta \\ \mathbb{R}^{n_1} \times \ldots \times \mathbb{R}^{n_d} & \xrightarrow{\varphi} & \mathbb{R}^{n_1\times\ldots\times n_d}. \end{array}$$

Since $\dim(\mathbb{R}^{n_1\times\ldots\times n_d}) = \dim(\mathbb{R}^{n_1} \otimes \ldots \otimes \mathbb{R}^{n_d})$, θ is an isomorphism. Consider the canonical basis of $\mathbb{R}^{n_1} \otimes \ldots \otimes \mathbb{R}^{n_d}$:

$$\left\{ \mathbf{e}^{(1)}_{i_1} \otimes \cdots \otimes \mathbf{e}^{(d)}_{i_d} \mid 1 \leq i_1 \leq n_1, \ldots, 1 \leq i_d \leq n_d \right\},$$

where $\left\{\mathbf{e}^{(\ell)}_1, \ldots, \mathbf{e}^{(\ell)}_{n_\ell}\right\}$ denotes the canonical basis in $\mathbb{C}^{n_\ell}$, $\ell = 1,\ldots,d$. Then θ may be described by

$$\theta\left(\sum_{i_1,\ldots,i_d=1}^{n_1,\ldots,n_d} a_{i_1,\ldots,i_d} \mathbf{e}^{(1)}_{i_1} \otimes \cdots \otimes \mathbf{e}^{(d)}_{i_d}\right) = [\![a_{i_1\cdots i_d}]\!]^{n_1,\ldots,n_d}_{i_1,\ldots,i_d=1} \in \mathbb{C}^{n_1\times\ldots\times n_d}.$$

Hence, a *multilinear tensor* of order or mode d and size $(n_1,\ldots,n_d)$ in $\mathbb{R}^{n_1} \otimes \ldots \otimes \mathbb{R}^{n_d}$ can be represented by a d-way array in $\mathbb{R}^{n_1\times\ldots\times n_d}$ up to a choice of basis on $\mathbb{R}^{n_1},\ldots,\mathbb{R}^{n_d}$.

Thus, without loss of generality, we refer hereafter to a tensor in $\mathbb{R}^{n_1}\otimes\ldots\otimes\mathbb{R}^{n_d}$ by an element $\mathcal{A} = [a_{i_1,\ldots,i_d}]_{1\leq i_1\leq n_1,\ldots,1\leq i_d\leq n_d}$ in $\mathbb{R}^{n_1\times\ldots\times n_d}$, and we note that all the notion that we will work on in this chapter and throughout this thesis are independent from the choice of basis of the vector spaces.

We also mention that we can associate to a tensor $\mathcal{A}$ a multilinear polynomial of total degree d in the variables $x_1 = (x^1_1,\ldots,x^{n_1}_1),\ldots,x_d = (x^1_d,\ldots,x^{n_d}_d)$ of the form $\sum a_{i_1,\ldots,i_d} x^{i_1}_1 \ldots x^{i_d}_d$, $1 \leq i_1 \leq n_1,\ldots,1 \leq i_d \leq n_d$. This bijection between the set of tensors and the set of multilinear polynomials is important since it allows to study tensors from an algebraic geometric point of view (see for instance [128]).

Some basics. We can create sub-dimensional sections of a tensor by fixing some of the tensor's indices. *Fibers* are the higher-order generalization of matrix rows and columns. They are obtained by fixing all the indices of the tensor but one. *Slices* of a tensor are obtained by fixing all the indices of the tensor except of two. In particular, if $\mathcal{A}$ is a three-dimensional tensor, then the fibers

$\mathcal{A}[:, j, k]$, $\mathcal{A}[i, :, k]$ and $\mathcal{A}[i, j, :]$ are called respectively column, row and tube fibers. Further, the slices $\mathcal{A}[i, :, :]$, $\mathcal{A}[:, j, :]$ and $\mathcal{A}[:, :, k]$ are called respectively horizontal, lateral and frontal slices. The *inner product* of $\mathcal{A} = [a_{i_1,\ldots,i_d}]_{1\leq i_1\leq n_1,\ldots,1\leq i_d\leq n_d}$ and $\mathcal{B} = [b_{i_1,\ldots,i_d}]_{1\leq i_1\leq n_1,\ldots,1\leq i_d\leq n_d}$ such that $\mathcal{A}$ and $\mathcal{B}$ are in $\mathbb{R}^{n_1\times\ldots\times n_d}$ is given by :

$$\langle \mathcal{A}, \mathcal{B}\rangle = \sum_{i_1=1}^{n_1} \ldots \sum_{i_d=1}^{n_d} a_{i_1,\ldots,i_d} b_{i_1,\ldots,i_d}.$$

The *Frobenius* norm of $\mathcal{A}$ is as follows :

$$\|\mathcal{A}\| = \sqrt{\langle \mathcal{A}, \mathcal{A}\rangle} = \sqrt{\sum_{i_1=1}^{n_1} \ldots \sum_{i_d=1}^{n_d} |a_{i_1,\ldots,i_d}|^2}.$$

It is analogous to the Frobenius matrix norm.

A tensor $\mathcal{A} \in \mathbb{R}^{n_1\times\ldots\times n_d}$ is called of rank one if it can be written as the outer product of d vectors as follows :

$$\mathcal{A} = a^1 \otimes \ldots \otimes a^d,$$

where $a^k \in \mathbb{R}^{n_k}$, $\forall 1 \leq k \leq d$. We can notice that the set of rank-one tensors (also called simple tensors) is given by the image of the Segre map

$$\begin{aligned}\varphi : \mathbb{R}^{n_1} \times \ldots \times \mathbb{R}^{n_d} &\to \mathbb{R}^{n_1\times\ldots\times n_d}\\ (a^1, \ldots, a^d) &\mapsto a^1 \otimes \ldots \otimes a^d = [a^1_{i_1} \ldots a^d_{i_d}]_{1\leq i_1\leq n_1,\ldots,1\leq i_d\leq n_d}.\end{aligned}$$

Some types of tensors. A tensor $\mathcal{A} \in \mathbb{R}^{n_1\times\ldots n_d}$ is *diagonal* if $a_{i_1,\ldots,i_d} \neq 0$ only if $i_1 = \ldots = i_d$. A tensor is called cubical if each mode is of the same size i.e. $n_1 = \ldots = n_d = n$. We denote by $\mathcal{T}^d_n(\mathbb{R}) = \mathbb{R}^n \otimes \cdots \otimes \mathbb{R}^n$ the outer product d times of $\mathbb{R}^n$. A cubical tensor $\mathcal{A}$ in $\mathcal{T}^d_n(\mathbb{R})$ is said to be *symmetric* if each entries remains unchanged under any permutation of its d indices i.e.

$$a_{i_1\ldots i_d} = a_{i_{\sigma(1)}\ldots i_{\sigma(d)}} \ , \forall \sigma \in \mathbb{S}_d,$$

where $\mathbb{S}_d$ denotes the symmetric group of permutations on $\{1, \ldots, d\}$.
In particular, a tensor can be partially symmetric and it is called *multi-symmetric*. For example, a tensor $\mathcal{A} \in \mathbb{R}^{n\times n\times s}$ is symmetric in modes one and two if

$$a_{i_1 i_2 i_3} = a_{i_2 i_1 i_3} \ , \forall\ 1 \leq i_1, i_2 \leq n,\ \forall\ 1 \leq i_3 \leq s,$$

in other words, all the frontal slices are symmetric matrices.

Since low rank symmetric tensor approximations constitutes a relevant part of our contributions in this thesis (Chapter 4), the next section is devoted to mention notation and definitions related to symmetric tensors, that will be used later in the contributions part.

2.2 Symmetric tensors

In this section, tensors are considered with complex coefficients.
The set of complex valued symmetric tensors in $\mathcal{T}^d_n(\mathbb{C})$ is denoted $\mathcal{S}^d_n(\mathbb{C})$. We have a correspondence between $\mathcal{S}^d_n(\mathbb{C})$ and the set of the homogeneous polynomials of degree d in n variables

$\mathbb{C}[x_1,\dots,x_n]_d := \mathbb{C}[\mathbf{x}]_d$. This allows to reduce the dimension of the ambient space from n^d (dimension of $\mathcal{T}_n^d(\mathbb{C})$) to $s_{n,d} := \binom{n+d-1}{d}$ (dimension of $\mathcal{S}_n^d(\mathbb{C}) \sim \mathbb{C}[\mathbf{x}]_d$). Bold letters such as $\boldsymbol{p}, \boldsymbol{q}$ denote homogeneous polynomials in $\mathbb{C}[\mathbf{x}]_d$ or equivalently elements in $\mathcal{S}_n^d(\mathbb{C})$. A homogeneous polynomial $\boldsymbol{p}$ in $\mathbb{C}[\mathbf{x}]_d$ can be written as : $\boldsymbol{p} = \sum_{|\alpha|=d} \binom{d}{\alpha} p_\alpha \mathbf{x}^\alpha$, where $\mathbf{x} := (x_1,\dots,x_n)$ is the vector of the variables $x_1,\dots,x_n$, $\alpha = (\alpha_1,\dots,\alpha_n)$ is a vector of the multi-indices in $\mathbb{N}^n$, $|\alpha| = \alpha_1 + \cdots + \alpha_n$, $p_\alpha \in \mathbb{C}$, $\mathbf{x}^\alpha := x_1^{\alpha_1} \dots x_n^{\alpha_n}$ and $\binom{d}{\alpha} := \frac{d!}{\alpha_1!\dots\alpha_n!}$.

A symmetric tensor $\mathcal{A}$ in $\mathcal{S}_n^d(\mathbb{C})$ is of symmetric rank one if it can be written as the outer product d times of the same vector a in $\mathbb{C}^n$ i.e.

$$\mathcal{A} = \underbrace{a \otimes \dots \otimes a}_{d \text{ times}}.$$

Let $\boldsymbol{p}$ be the associate homogeneous polynomial to $\mathcal{A}$ in $\mathbb{C}[\mathbf{x}]_d$. This means that $\boldsymbol{p}$ can be written as the linear form given by the vector a to the d^{th} power i.e.

$$\boldsymbol{p} = (a^t \mathbf{x})^d.$$

Definition 2.2.1. For $\boldsymbol{p} = \sum_{|\alpha|=d} \binom{d}{\alpha} p_\alpha \mathbf{x}^\alpha$ and $\boldsymbol{q} = \sum_{|\alpha|=d} \binom{d}{\alpha} q_\alpha \mathbf{x}^\alpha$ in $\mathbb{C}[\mathbf{x}]_d$, their apolar product is

$$\langle \boldsymbol{p}, \boldsymbol{q} \rangle_d := \sum_{|\alpha|=d} \binom{d}{\alpha} \bar{p}_\alpha q_\alpha.$$

Obviously, the apolar product for real symmetric tensors $\boldsymbol{p}$ and $\boldsymbol{q}$ in $\mathbb{R}[\mathbf{x}]_d$ is given by $\langle \boldsymbol{p}, \boldsymbol{q} \rangle_d = \sum_{|\alpha|=d} \binom{d}{\alpha} p_\alpha q_\alpha$.
The apolar norm of $\boldsymbol{p}$ is $||\boldsymbol{p}||_d = \sqrt{\langle \boldsymbol{p}, \boldsymbol{p} \rangle_d} = \sqrt{\sum_{|\alpha|=d} \binom{d}{\alpha} \bar{p}_\alpha p_\alpha}$.

The apolar product is invariant by a linear change of variables of the unitary group U_n : $\forall u \in U_n, \langle p(u\,\mathbf{x}), q(u\,\mathbf{x}) \rangle_d = \langle p(\mathbf{x}), q(\mathbf{x}) \rangle_d$.
The following properties of the apolar product can be verified by direct calculus :

Lemma 2.2.1. *Let $\boldsymbol{l} = (v_1 x_1 + \cdots + v_n x_n)^d := (v^t \mathbf{x})^d \in \mathbb{C}[\mathbf{x}]_d$ where $v = (v_i)_{1 \le i \le n}$ is a vector in $\mathbb{C}^n$, $\boldsymbol{q} \in \mathbb{C}[\mathbf{x}]_{(d-1)}$, we have the following two properties :*

1. $\langle \boldsymbol{l}, \boldsymbol{p} \rangle_d = \boldsymbol{p}(\bar{v})$, $\langle \boldsymbol{p}, \boldsymbol{l} \rangle_d = \bar{\boldsymbol{p}}(v)$,
2. $\langle \boldsymbol{p}, x_i \boldsymbol{q} \rangle_d = \frac{1}{d} \langle \partial_{x_i} \boldsymbol{p}, \boldsymbol{q} \rangle_{d-1}$, $\langle x_i \boldsymbol{q}, \boldsymbol{p} \rangle_d = \frac{1}{d} \langle \boldsymbol{q}, \partial_{x_i} \boldsymbol{p} \rangle_{d-1}$, $\forall 1 \le i \le n$.

These properties are called the apolar identities.

2.3 Tensor reorderings : vectorization and matricization

Let $\mathcal{A} = [a_{i_1,\dots,i_d}]_{1 \le i_1 \le n_1, \dots, 1 \le i_d \le n_d} \in \mathbb{R}^{n_1 \times \dots \times n_d}$.

- The *vectorization* of $\mathcal{A}$ is to stacking vertically all the entries of $\mathcal{A}$ in one vector.

$$vec(\mathcal{A}) = [a_{1,\dots,1}\ a_{2,1,\dots,1}\ \dots\ a_{n_1,\dots,n_{d-1},n_d-1}\ a_{n_1,\dots,n_d}]^t.$$

- Let $k \in \{1, \ldots, d\}$. *The kth mode of matricization* or *flattening* of $\mathcal{A}$ is a rearrangement of the entries of $\mathcal{A}$ into a matrix $\mathcal{A}_{(k)}$ such that the *kth* mode becomes the row index and all the other $(d-1)$ modes become column indices, i.e. $\mathcal{A}_{(k)}$ has n_k rows and $\prod_{1 \leq j \neq k \leq d} n_j$ columns. More precisely, the matricization of mode k maps an element of the tensor $\mathcal{A}$ of index $(i_1, \ldots, i_d)$ into an element of the matrix $\mathcal{A}_{(k)}$ of index (i_k, j) such that

$$j = 1 + \sum_{\substack{l=1 \\ l \neq k}}^{d} (i_l - 1) I_l \text{ with } I_l = \prod_{\substack{m=1 \\ m \neq k}}^{l-1} n_m.$$

Example 2.3.1 – Let $\mathcal{A}$ be a tensor in $\mathbb{R}^{3 \times 2 \times 4}$ such that its frontal slices are as follows :

$$\mathcal{A}_1 = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, \mathcal{A}_2 = \begin{bmatrix} 7 & 10 \\ 8 & 11 \\ 9 & 12 \end{bmatrix}, \mathcal{A}_3 = \begin{bmatrix} 13 & 16 \\ 14 & 17 \\ 15 & 18 \end{bmatrix}, \mathcal{A}_4 = \begin{bmatrix} 19 & 22 \\ 20 & 23 \\ 21 & 24 \end{bmatrix}.$$

We have :

$$vec(\mathcal{A}) = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ \ldots\ 22\ 23\ 24]^t,$$

$$\mathcal{A}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix},$$

$$\mathcal{A}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 7 & 8 & 9 & \ldots & 19 & 20 & 21 \\ 4 & 5 & 6 & 10 & 11 & 12 & \ldots & 22 & 23 & 24 \end{bmatrix},$$

$$\mathcal{A}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \\ 19 & 20 & 21 & 22 & 23 & 24 \end{bmatrix}.$$

2.4 Important matrix and tensor products

- The k-mode (matrix) product of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ with a matrix $U \in \mathbb{R}^{I \times n_k}$ is a tensor of order d and size $(n_1, \ldots, n_{k-1}, I, n_{k+1}, \ldots, n_d)$ denoted by $\mathcal{A} \times_k U$ such that

$$(\mathcal{A} \times_k U)_{i_1, \ldots, i_{k-1}, i, i_{k+1}, \ldots, i_d} = \sum_{i_k=1}^{n_k} a_{i_1, \ldots, i_{k-1}, i_k, i_{k+1}, \ldots, i_d} u_{i, i_k}.$$

  This is nothing else but the multiplication of each mode-k fiber with the matrix U. Alternatively, the k-mode product can be expressed in terms of matricized tensors as follows :

$$\mathcal{Y} = \mathcal{A} \times_k U \Leftrightarrow \mathcal{Y}_{(k)} = U \mathcal{A}_{(k)}.$$

  There is no matter concerning the order of multiplication i.e.

$$\mathcal{A} \times_k U \times_l V = \mathcal{A} \times_l V \times_k U \ \ (k \neq l),$$

and when $k = l$, we have :

$$\mathcal{A} \times_k U \times_k V = \mathcal{A} \times_k (VU).$$

The k-mode product also exists for tensors with vectors. Let $v \in \mathbb{R}^{n_k}$, the k-mode product of $\mathcal{A}$ with v denoted by $\mathcal{A} \times_k v$ is a tensor of order $d-1$ and size $(n_1, \ldots, n_{k-1}, n_{k+1}, \ldots, n_d)$. Element-wise, this can be expressed as follows :

$$(\mathcal{A} \times_k v)_{i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_d} = \sum_{i_k=1}^{n_k} a_{i_1,\ldots,i_{k-1},i_k,i_{k+1},\ldots,i_d} v_{i_k}.$$

- The Kronecker product of $A \in \mathbb{R}^{n_1 \times n_2}$ with $B \in \mathbb{R}^{m_1 \times m_2}$ is denoted by $A \otimes B$ and yields a matrix in $\mathbb{R}^{n_1 m_1 \times n_2 m_2}$ such that :

$$\begin{aligned} A \otimes B &= \begin{bmatrix} a_{1,1}B & a_{1,2}B & \ldots & a_{1,n_2}B \\ a_{2,1}B & a_{2,2}B & \ldots & a_{2,n_2}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_1,1}B & a_{n_1,2}B & \ldots & a_{n_1,n_2}B \end{bmatrix} \\ &= \begin{bmatrix} a_1 \otimes b_1 & a_1 \otimes b_2 & a_1 \otimes b_3 & \ldots & a_{n_2} \otimes b_{m_2-1} & a_{n_2} \otimes b_{m_2} \end{bmatrix}, \end{aligned}$$

where $a_1, \ldots, a_{n_2}$ (resp. $b_1, \ldots, b_{m_2}$) denote the columns of the matrix A (resp. B). More precisely $(A \otimes B)_{(i_1-1)m_1+j_1,(i_2-1)m_2+j_2} = a_{i_1,i_2} b_{j_1,j_2}$.

We have the following property [117] which relates the Kronecker product of matrices to the k-mode product of tensors as follows : Let $A_k \in \mathbb{R}^{I_k \times n_k}$,

$$\mathcal{Y} = \mathcal{A} \times_1 A_1 \ldots \times_d A_d \Leftrightarrow \mathcal{Y}_{(k)} = A_k \mathcal{A}_{(k)} (A_d \otimes \ldots \mathcal{A}_{k-1} \otimes A_{k+1} \otimes \ldots \otimes A_d)^t, \; \forall \, 1 \leq k \leq d. \tag{2.2}$$

- The Khatri-Rao product is the column-wise Kronecker product. For two matrices of the same number of columns $A \in \mathbb{R}^{n_1 \times m}$ and $B \in \mathbb{R}^{n_2 \times m}$, their Khatri-Rao product denoted by $A \odot B$ is a matrix of size $n_1 n_2 \times m$ defined as follows :

$$A \odot B = \begin{bmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 & \ldots & a_m \otimes b_m \end{bmatrix}.$$

In particular, the Kronecker product of two vectors is identical to their Khatri-Rao product.

- The Hadamard product is the element-wise matrix product. For two matrices A and B of the same size $n_1 \times n_2$, the Hadamard product is denoted by $A * B$, and it is given by :

$$A * B = \begin{bmatrix} a_{1,1}b_{1,1} & a_{1,2}b_{1,2} & \ldots & a_{1,n_2}b_{1,n_2} \\ a_{2,1}b_{2,1} & a_{2,2}b_{2,2} & \ldots & a_{2,n_2}b_{2,n_2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_1,1}b_{n_1,1} & a_{n_1,2}b_{n_1,2} & \ldots & a_{n_1,n_2}b_{n_1,n_2} \end{bmatrix}.$$

2.5 Canonical Polyadic Decomposition

The Canonical Polyadic Decomposition (CPD) proposed by Hitchcock [101], express the tensor as a sum of rank-one simple tensors in the form

$$\mathcal{A} = \sum_{i=1}^{r} a_i^1 \otimes \ldots \otimes a_i^d, \tag{2.3}$$

where r is a positive integer and $a_i^k \in \mathbb{R}^{n_k}, \forall\ 1 \leq i \leq r,\ \forall\ 1 \leq k \leq d$.
The *factor matrices* are given by :

$$A_k = [a_1^k \dots a_r^k] \in \mathbb{R}^{n_k \times r},\ \forall 1 \leq k \leq d.$$

Thus, using the factor matrices the CPD decomposition can be expressed as follows

$$\mathcal{A} = [\![A_1, \dots, A_d]\!] = \sum_{i=1}^{r} a_i^1 \otimes \dots \otimes a_i^d.$$

In particular, if $\mathcal{A}$ is a three-dimensional tensor, the frontal slices of $\mathcal{A}$ can be expressed in terms of factor matrices denoted by A, B and C as follows

$$\mathcal{A}_k = A\,\mathrm{diag}(C[k,:])B^t,\ \forall\ 1 \leq k \leq n_3.$$

This expression does not easily extend to tensors of order higher than three.

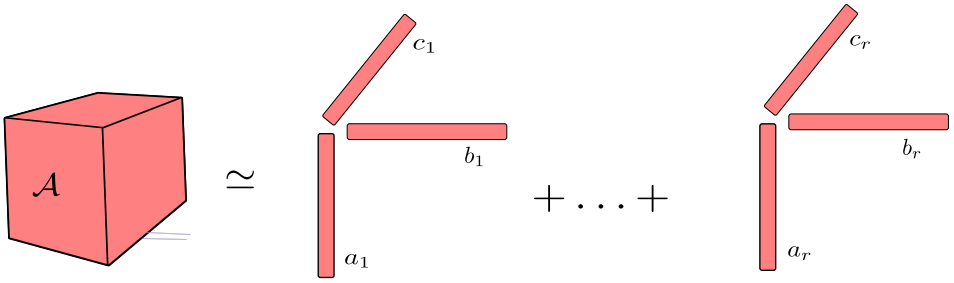


Figure 2.2 – CPD of a three-dimensional tensor.

In general, it is recommended from computation perspectives to normalize the columns of the factor matrices, by using a *weight vector* $\lambda = (\lambda_i)_{1 \leq i \leq r} \in \mathbb{R}^r$ so that

$$\mathcal{A} = [\![\lambda, A_1, \dots, A_d]\!] = \sum_{i=1}^{r} \lambda_i a_i^1 \otimes \dots \otimes a_i^d.$$

Equivalently, $\mathcal{A}$ can be expressed as a multilinear product with a diagonal tensor

$$\mathcal{A} = \mathcal{D} \times_1 A_1 \dots \times_d A_d,$$

where $\mathcal{D}$ is a diagonal tensor in $\mathcal{T}_r^d$ with $\mathcal{D}_{i,\dots,i} = \lambda_i,\ \forall\ 1 \leq i \leq r$.
It follows from (2.2) that the kth mode of matricization of $\mathcal{A}$ is given by

$$\mathcal{A}_{(k)} = A_k \Lambda (A_d \odot \dots A_{k-1} \odot A_{k+1} \odot \dots \odot A_1)^t,$$

where $\Lambda = \mathrm{diag}(\lambda)$. The Kronecker product in (2.2) is replaced here by the Hadamard product since all the factor matrices A_k have the same number of columns equal to r.

Similarly, symmetric tensors can be expressed as a sum of symmetric rank-one tensors i.e. for $\mathcal{A} \in \mathcal{S}_n^d$

$$\mathcal{A} = \sum_{i=1}^{r} w_i \underbrace{a_i \otimes \dots \otimes a_i}_{d \text{ times}},\ \ w_i \in \mathbb{R},\ a_i \in \mathbb{R}^n. \tag{2.4}$$

Equivalently, the homogeneous polynomial $\boldsymbol{p}$ associated to $\mathcal{A}$ is written as a sum of powers of linear forms i.e.

$$\boldsymbol{p} = \sum_{i=1}^{r} w_i (a_{i,1}x_1 + \cdots + a_{i,n}x_n)^d, \;\; w_i \in \mathbb{R}, \; a_i \in \mathbb{R}^n. \tag{2.5}$$

2.5.1 Tensor rank

The *rank* of a tensor $\mathcal{A}$ denoted by $\mathrm{rank}(\mathcal{A})$ is the minimal number of rank-one tensors needed in the CPD decomposition in (2.3) to generate the tensor $\mathcal{A}$. The CPD decomposition with r equal to the rank of the tensor is called the *rank decomposition*. The tensor rank is not affected by mode permutation.
Similarly, *the symmetric tensor rank* of a symmetric tensor $\mathcal{A}$ denoted by $\mathrm{rank}_s(\mathcal{A})$ is the minimal number of symmetric rank-one tensors in a decomposition of the form (2.5). The decomposition (2.5) is called *Waring decomposition* when r is equal to the symmetric rank of the symmetric tensor. As a direct property, we have $\mathrm{rank}_s(\mathcal{A}) \geq \mathrm{rank}(\mathcal{A})$, since the constraint on the rank-one tensors to be symmetric may increase the rank. Nevertheless, Comon et al. show in [54] that equality holds when $\mathrm{rank}_s(\mathcal{A}) = 1, 2$, and holds generically when $\mathrm{rank}_s(\mathcal{A}) \leq n$ and when d is sufficiently large with respect to n. More recently, Shitov shows a counterexample in [186] for a symmetric tensor in $\mathcal{S}^3_{800}(\mathbb{C})$, that has rank strictly less than its symmetric rank.
The rank may depend on the field, where it may be different over $\mathbb{R}$ and $\mathbb{C}$. This idea is demonstrated throughout two examples : The first (can be found in [118]) for a tensor $\mathcal{A} \in \mathbb{R}^{2\times2\times2}$ and the second for a symmetric tensor $\mathcal{A} \in \mathcal{S}^3_2(\mathbb{R})$ (from [54]).

Example 2.5.1 – Let $\mathcal{A} \in \mathbb{R}^{2\times2\times2}$ such that the frontal slices are given by :

$$\mathcal{A}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathcal{A}_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

This tensor is of rank 3 over $\mathbb{R}$ and rank 2 over $\mathbb{C}$.
The factor matrices for the rank decomposition over $\mathbb{R}$ are as follows :

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix},$$

and over $\mathbb{C}$ are as follows :

$$A = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -\mathbf{i} & \mathbf{i} \end{bmatrix}, B = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ \mathbf{i} & -\mathbf{i} \end{bmatrix}, C = \begin{bmatrix} 1 & 1 \\ \mathbf{i} & -\mathbf{i} \end{bmatrix}.$$

Example 2.5.2 – Let $\mathcal{A} \in \mathcal{S}^3_2(\mathbb{R})$ such that the frontal slices are given by :

$$\mathcal{A}_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \mathcal{A}_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

This tensor is of symmetric rank 3 over $\mathbb{R}$:

$$\mathcal{A} = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix}^{\otimes 3} + \frac{1}{2} \begin{pmatrix} 1 \\ -1 \end{pmatrix}^{\otimes 3} - 2 \begin{pmatrix} 1 \\ 0 \end{pmatrix}^{\otimes 3},$$

and symmetric rank 2 over $\mathbb{C}$:

$$\mathcal{A} = \frac{\mathbf{i}}{2}\begin{pmatrix} -\mathbf{i} \\ 1 \end{pmatrix}^{\otimes 3} - \frac{\mathbf{i}}{2}\begin{pmatrix} \mathbf{i} \\ 1 \end{pmatrix}^{\otimes 3}.$$

The *maximum rank* is the largest attainable rank. For instance, a week upper bound on the maximum rank that can be reached by a three-dimensional tensor in $\mathbb{R}^{n_1 \times n_2 \times n_3}$ is established in [125]

$$\operatorname{rank}(\mathcal{A}) \leq \min\{n_1 n_2, n_1 n_3, n_2 n_3\}.$$

For example, it has been shown in the same reference [125] that the maximum rank for $3 \times 3 \times 3$ tensors over $\mathbb{R}$ is equal to 5.

Two other important notions in term of tensor rank is the *typical rank* and the *generic rank*. The typical rank is any rank that occurs with probability greater than zero, whereas the *generic rank* is the rank which is true almost everywhere (i.e. with probability one), when their entries are chosen independently according to a continuous probability distribution. For tensors over $\mathbb{R}$ there may be more than one typical rank, for example $2 \times 2 \times 2$ tensors over $\mathbb{R}$ have two typical rank 2 and 3 [125]. However, the smallest typical rank over $\mathbb{R}$ is the generic rank over $\mathbb{C}$ [23], and thus the generic rank of $\mathbb{C}^{2\times 2\times 2}$ is 2. When there is only one typical rank, then it may be called the generic rank. There is always a single typical rank over $\mathbb{C}$, and is thus generic.
The expected generic rank for real or complex tensors of size $(n_1, \ldots, n_d)$ [23] is given by

$$r_g = \left\lceil \frac{n_1 n_2 \ldots n_d}{n_1 + n_2 + \ldots + n_d - d + 1} \right\rceil. \tag{2.6}$$

The generic symmetric rank for real or complex symmetric tensors of order d and dimension n is given by Alexander–Hirschowitz theorem [10] as follows

$$r_g = \left\lceil \frac{1}{n}\binom{n+d-1}{d} \right\rceil, \tag{2.7}$$

except for when $(d, n) \in \{(3, 5), (4, 3), (4, 4), (4, 5)\}$, where it should be increased by 1. For further discussion concerning typical and generic rank see for instance [23, 54, 46, 48].

The last type of tensor rank that we aim to present in this section is the *border rank*. A tensor $\mathcal{A}$ is of border rank r if it is the limit of tensors of rank r but not the limit of tensors of rank smaller than r. Rank and border rank of a tensor may actually be different, let us take the following example from [53] to illustrate this idea :

Example 2.5.3 – Let $\varepsilon > 0$ and u, v be two no-collinear vectors such that

$$\mathcal{A}_\varepsilon = \frac{1}{\varepsilon}\left[(u + \varepsilon v)^{\otimes 3} - u^{\otimes 3}\right] = \mathcal{A} + O(\varepsilon),$$

where

$$\mathcal{A} = u \otimes u \otimes v + u \otimes v \otimes u + v \otimes u \otimes u.$$

The tensor $\mathcal{A}$ is of rank 3 whereas its border rank is equal 2, since $\mathcal{A}$ is the limit of rank-2 tensors $(\mathcal{A}_\varepsilon)_{\varepsilon>0}$.

Remark 2.5.1 – Unlike the matrix case, there is no straightforward algorithm to compute the rank of a given tensor as the problem is NP hard [109].

2.5.2 Uniqueness

Let $\mathcal{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ of rank r with rank decomposition given by

$$\mathcal{A} = [\![A_1, \ldots, A_d]\!] = \sum_{i=1}^{r} a_i^1 \otimes \ldots \otimes a_i^d. \tag{2.8}$$

Uniqueness means that the rank decomposition (2.8) is unique up to the elementary indeterminacies of scaling and permutation. In this case, the tensor $\mathcal{A}$ is called *r-identifiable*. The scaling undeterminacy is due to the fact that we can scale vectors such that

$$\mathcal{A} = \sum_{i=1}^{r} (\lambda_{i,1} a_i^1) \otimes \ldots \otimes (\lambda_{i,d} a_i^d),$$

with $\lambda_{i,1} \ldots \lambda_{i,d} = 1,\ \forall 1 \leq i \leq r$. The permutation undeterminacy is due to the fact that rank-1 component tensors can be ordered arbitrarily such that

$$\mathcal{A} = [\![A_1, \ldots, A_d]\!] = [\![A_1 P, \ldots, A_d P]\!], \text{ for any } r \times r \text{ permutation matrix } P.$$

One can notice that rank decomposition for matrices is not unique. Indeed, let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2}$ be of rank r with rank decomposition

$$\mathcal{A} = \sum_{i=1}^{r} a_i \otimes b_i = AB^t.$$

Let $U\Sigma V^t$ be the SVD decomposition of $\mathcal{A}$. We can take $A = U\Sigma$ and $B = V$, thus equivalently for some $r \times r$ orthogonal matrix W, it is possible to obtain a completely different rank-1 matrices (i.e. the rank-1 components in the rank decomposition of the matrix $\mathcal{A}$) by taking $A = U\Sigma W$, $B = VW$. We remark that the SVD decomposition of a matrix is unique (for instance assuming that all the singular values are distinct) but this is only because of the strong condition on the matrices U and V (orthogonal matrices) with the diagonal matrix Σ of ordered singular values in the middle. Nevertheless, the uniqueness in the case of higher order tensors can be obtained with much weaker conditions. For instance, a classical sufficient condition for uniqueness for third order tensors is due to Kruskal [127]

$$k_A + k_B + k_C \geq 2r + 2,$$

where k_M of a matrix M denotes its k-rank given by the maximum value k such that any k columns from M are linearly independent. This condition is extended to the general case of order-d tensors in [188] as follows

$$\sum_{i=1}^{d} k_{A_i} \geq 2r + (d-1).$$

More generally, Liu and Sidiropoulos showed in [137] the following necessary conditions for uniqueness

$$\min_{k=1,\ldots,d} \operatorname{rank}(A_1 \odot \ldots \odot A_{k-1} \odot A_{k+1} \odot \ldots \odot A_d) = r,$$

and

$$\min_{k=1,\ldots,d} \Big(\prod_{\substack{i=1 \\ i\neq k}}^{d} \operatorname{rank}(A_i) \Big) \geq r.$$

On the other hand some conditions for *generic* uniqueness are established. For instance, Delathauwer has proved in [60] that the rank decomposition in (2.8) for third respectively fourth order tensors is generically (i.e. with probability one) unique if for third order tensors

$$r \leq n_3 \text{ and } r(r-1) \leq \frac{1}{2} n_1(n_1-1)n_2(n_2-1),$$

respectively for fourth order tensors

$$r \leq n_4 \text{ and } r(r-1) \leq \frac{1}{4} n_1 n_2 n_3 (3n_1 n_2 n_3 - n_1 n_2 - n_1 n_3 - n_2 n_3 - n_1 - n_2 - n_3 + 3).$$

Further, Chiantini et al. investigate in [46, 48] the generic uniqueness or generic identifiability of tensors in $\mathbb{C}^{n_1 \times \ldots \times n_d}$, where they show that, for instance, a generic tensor $\mathcal{A} \in \mathbb{C}^{n_1 \times \ldots \times n_d}$ of *subgeneric* rank (i.e. of rank strictly lower than the expected generic rank in (2.6)) is r-identifiable if $\prod_{k=1}^{d} n_k \leq 15000$ unless some exceptions (see [46, Theorem 1]). They also studied this problem for symmetric tensors where they proved that the Waring decomposition of a generic symmetric tensor in $\mathcal{S}_n^d(\mathbb{C})$ of subgeneric rank (i.e. of rank strictly lower than the generic rank in (2.7)) is unique unless in three exceptions [47, Theorem 1.1]. For a thorough study of uniqueness and generic uniqueness see the aforementioned references in this section as well the references therein.

2.5.3 Exact computation

There exists in the literature various algorithms for exact computation of the rank decomposition. The approaches used to develop these algorithms are essentially based on linear algebra by solving sets of linear equations and computing generalized eigenvalue decomposition. We provide for instance some pointers among many others e.g. [180, 73, 204, 168], and for the Waring decomposition of symmetric tensors e.g. [19, 30, 158, 92]. For further algorithms for the rank decomposition based on simultaneous or joint diagonalization of matrices see e.g. [60, 59, 196, 140, 72].

2.5.4 Low rank tensor approximation problem

In practice measurements are rarely free of noise, consequently the rank decomposition of the tensor built from these measurements is rarely exact. For this reason, one often prefers to approximate the given tensor by a low rank tensor decomposition. This is the so-called low rank tensor approximation problem. It aims to best approximate a given tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ by a rank decomposition with number of components equal to the approximation rank r. More concretely, this can be done by minimizing the Frobenius norm between the given tensor $\mathcal{A}$ and the decomposition of rank r

$$\min_{\hat{\mathcal{A}}} \|\mathcal{A} - \hat{\mathcal{A}}\|, \text{ with } \hat{\mathcal{A}} = \sum_{i=1}^{r} a_i^1 \otimes \ldots \otimes a_i^d = [\![A_1, \ldots, A_d]\!]. \tag{2.9}$$

2.5.4.1 Existence and degeneracies

Before taking a look at popular approaches to solve the low rank tensor approximation problem, let us recall briefly some facts about this optimization problem. Unfortunately, this approximation problem is not always well-posed [67, 100]. In fact, De Silva and Lim show in [67] that the set of input tensors that fail to have a solution i.e. a best low rank approximation has a positive Lebesgue measure, meaning that this case is not rare. Ill-posedness comes from the fact that the set of tensors of rank bounded by r is not closed. As shown in Example 2.5.3, we have a sequence of tensors $(\mathcal{A}_\epsilon)_{\epsilon>0}$ of rank at most 2 which converges towards a tensor of rank 3 and not 2, thus the set of tensors of rank at most 2 is not closed. Equivalently, this means also that the tensor $\mathcal{A}$ of rank 3 does not have a best rank-2 approximation, and that the low rank approximation problem of the tensor $\mathcal{A}$ by a tensor of rank bounded by 2 has only an infimum. It follows that if a tensor $\mathcal{A}$ of rank strictly higher than r does not have a best rank-r approximation and we attempt to minimize

$$\|\mathcal{A} - \lambda_1 a_1^1 \otimes \ldots \otimes a_1^d - \cdots - \lambda_r a_r^1 \otimes \ldots \otimes a_r^d\|,$$

such that the vectors a_i^j for $1 \leq i \leq r$, and $1 \leq j \leq d$ are normalized or uniformly bounded, then at least some of the coefficients λ_i become unbounded, i.e. the magnitude of some terms in the decomposition go to infinity. This phenomenon is referred to as *degeneracy*. More precisely, when there are k diverging terms such that their sum is bounded is called *k-factor degeneracies* [124, 126, 163, 198, 199], this phenomenon also exists for symmetric tensors [54]. Any algorithm that minimizes the cost function in (2.9) will yield a degeneracy if this problem does not have an optimal solution (i.e. its infimum is not reached). Nevertheless, there exists in the literature a family of approaches to avoid degeneracies by imposing constraints in the CPD, like for instance imposing orthogonality between columns of factor matrices [55], bounding the coefficients λ_i [163, 135], imposing a minimal angle between columns of factor matrices [136], imposing, when the tensor has positive entries, the non negativity on the components of the CP decomposition (i.e. that is impose on the entries of the rank-1 tensors to be positive) [135] (for more references in this regard see [53] and references therein). However, in practice this problem remains open since there is no fully satisfactory solution.

2.5.4.2 Algorithms

We come now to methods for solving the low rank approximation problem. Several algorithms were proposed for this purpose in the literature. One of the common approaches is to parameterize the set of tensors of rank bounded by r using factor matrices and then to formulate the low rank approximation problem as an unconstrained optimization problem as follows

$$\min_{(A_1,\ldots,A_d)\in \mathrm{D}} \frac{1}{2}\|[\![A_1,\ldots,A_d]\!] - \mathcal{A}\|^2, \tag{2.10}$$

with $\mathrm{D} := \mathbb{R}^{n_1\times r} \times \ldots \times \mathbb{R}^{n_d\times r}$, and $[\![A_1,\ldots,A_d]\!] = \sum_{i=1}^r a_i^1 \otimes \ldots \otimes a_i^d$. General optimization methods such as, for instance, alternate least squares [42, 94, 44], conjugate gradient [162] and quasi-Newton methods [5, 200, 98, 161, 169, 192, 206] are widely used in the literature to solve this problem. Alternate least squares (ALS) methods solve sequentially simple linear subproblems, where at each step the cost function is optimized for

one of the factor matrices while the other factor matrices are considered fixed, which leads to solve overdetermined sets of linear equations. Despite its simplicity, the method has some drawbacks. For instance, it has a slow convergence especially in the presence of ill-conditioned cases [148, 172]. It has been shown that numerical optimization methods like conjugate gradient methods and quasi-Newton methods outperform the ALS-like strategies in this case [162, 5], though this comes at a higher computational cost per iteration. The aforementioned methods differ from each other for instance in terms of performance, computation complexity, and in how they address the ill-conditioning problem which appears in the iterative algorithms as the degeneracy phenomenon that we described previously and yields numerical difficulties. Let us mention one complication in the Gauss–Newton method which is a quasi-Newton method used in this context in many efficient algorithms e.g. [5, 98, 161, 169, 192, 206]. An essential step in this method is to solve at each iteration the normal equation of the form $J^tJx = -J^tr$ such that r is the residual vector i.e. $r = vec([\![A_1, \ldots, A_d]\!] - \mathcal{A})$ and J is the Jacobian matrix of the objective function in (2.10). The matrix J^tJ, called the Gauss–Newton approximation of the Hessian, is never of full rank [214] and thus methods that use this approach apply strategies such that for example the use of a regularization term or the Moore–Penrose pseudoinverse. We mention that there exists efficient implementation in term of complexity and storage to compute the terms J^tr [166, 215] and J^tJ [161, 169, 192]. Recently, the structure of rank-1 components in the CP decomposition was exploited in [33], where the low rank approximation problem is formulated as a Riemannian optimization problem over the Cartesian product of rank-1 manifolds (namely Segre manifolds), and a Riemannian Gauss–Newton iteration is developed to solve the problem. One advantage in this approach, is that the local convergence of the Gauss–Newton method is not affected by large difference scaling between the rank-1 components.

There exists also methods in the literature for the low rank symmetric tensor approximation problem, which aims to best approximates a given symmetric tensor to a Waring decomposition of low symmetric rank. We mention some of them. The algorithm CCPD-NLS from Tensorlab [217] employs a Gauss–Newton iteration to solve, for a given $\mathcal{A} \in \mathcal{S}^d_n(\mathbb{R})$, the non-linear least squares optimization problem

$$\min_{A \in \mathbb{R}^{n \times r}} \frac{1}{2} \|\mathcal{A} - \sum_{i=1}^{r} a_i^{\otimes d}\|^2,$$

where a_i are the columns of A. In [154] a method is developed for this purpose by expressing the linear relations among the entries of low rank symmetric tensors as polynomials called generating polynomials, and using approximate common zeros of these polynomials. Recently, Kileel and Pereira introduced the Subspace Power Method (SPM) [116] to solve this approximation problem for real symmetric tensors of even orders, mainly the method uses the tensor power method called SS-HOPM [119] on a modified tensor built from the flattening of the original tensor to a square matrix, and then applies deflation steps.

2.5.5 Applications of CPD

The CP decomposition is interesting in applications thank to its uniqueness property, which occurs for all higher order tensors. Because of this property, it is heavily used to recover

hidden structures in several applications including blind source separation, dimensionality reduction, pattern and image recognition, machine learning, data mining, data analysis, signal processing, biomedical engineering, chemometrics, and multidimensional harmonic retrieval ; we mention for instance [35, 49, 50, 182, 190, 13, 6, 149, 118, 193, 194].

2.6 Tucker decomposition

In this thesis we are interested in particular in the rank decomposition. Nevertheless, in this section we present briefly another important type of tensor decomposition called the Tucker decomposition [207].

- The Tucker decomposition can be viewed as a higher order Principal Component Analysis (PCA). It consists in decomposing the tensor into a so-called core tensor multiplied by a matrix along each mode : for $\mathcal{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ we have

$$\begin{aligned}\mathcal{A} = \mathcal{S} \times_1 A_1 \times_2 \ldots \times_d A_d &= \sum_{i_1=1}^{r_1} \ldots \sum_{i_d=1}^{r_d} s_{i_1 \ldots i_d} a_{i_1}^1 \otimes \ldots \otimes a_{i_d}^d \\ &:= [\![\mathcal{S}; A_1, \ldots, A_d]\!],\end{aligned}$$

  where $A_i = [a_1^i, \ldots, a_{r_i}^i] \in \mathbb{R}^{n_i \times r_i}$, $\forall\ 1 \leq i \leq d$, and $\mathcal{S} \in \mathbb{R}^{r_1 \times \ldots \times r_d}$ is the core tensor. Tucker decomposition appears in many applications such as classification, feature extraction, and subspace-based harmonic retrieval [138, 216, 90, 167]. Unlike the rank decomposition, the Tucker decomposition is generically not unique.

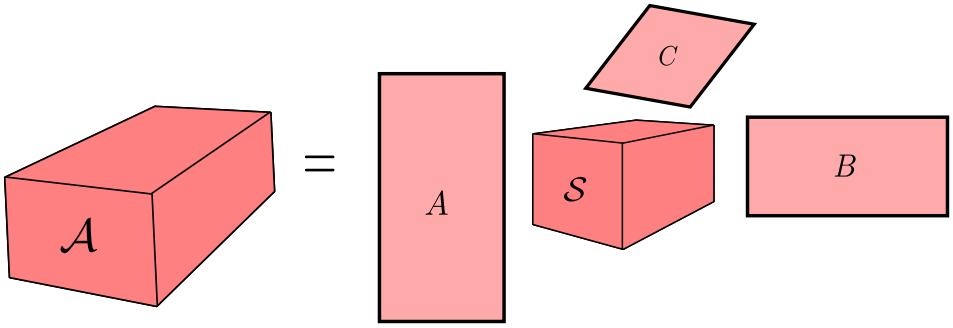


Figure 2.3 – Tucker decomposition of a three-dimensional tensor.

- The multilinear rank of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ is the tuple $(r_1, \ldots, r_d)$ such that r_i is the column rank of the mode-i matricization of $\mathcal{A}$, i.e. $r_i = \mathrm{rank}(\mathcal{A}_{(i)})$. Any tensor of multilinear rank $(r_1, \ldots, r_d)$ has an orthogonal Tucker decomposition such that the core tensor $\mathcal{S} \in \mathbb{R}^{r_1 \times \ldots \times r_d}$ and the factor matrices are orthonormal i.e. $A_i^t A_i = I_{r_i}$.
- The Higher Order Singular Value Decomposition (HOSVD) [63, 215] is a multilinear singular value decomposition described by the successive application of the SVD to each mode of matricization of $\mathcal{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ of multilinear rank $(r_1, \ldots, r_d)$. Let $U_i \in \mathbb{R}^{n_i \times r_i}$ denotes the matrix of the left singular vectors from the SVD decomposition of $\mathcal{A}_{(i)}$, we have by orthonormality

$$\mathcal{A} = [\![\mathcal{S}; U_1, \ldots, U_d]\!],$$

such that

$$\mathcal{S} = \mathcal{A} \times_1 U_1^t \times_2 \ldots \times_d U_d^t.$$

- Truncated Higher Order Singular Value Decomposition (THOSVD) [63, 215] allows to approximate a tensor $\mathcal{A}$ of multilinear rank $(r_1, \ldots, r_d)$ by a tensor $\hat{\mathcal{A}}$ of multilinear rank $(s_1, \ldots, s_d)$ such that $s_i \leq r_i, \ \forall \ 1 \leq i \leq d$, by applying successive truncated rank-s_i SVD to each mode-i flattening $\mathcal{A}_{(i)}$. It is well-known that truncated SVD for matrices yields best low rank matrix approximation, but it is not the case for truncated HOSVD where in general it does not yield best multilinear rank $(s_1, \ldots, s_d)$ approximation. Nevertheless, we have the following quasi best approximation property [63]

$$\|\mathcal{A} - \hat{\mathcal{A}}\| \leq \sqrt{d}\|\mathcal{A} - \mathcal{A}_t\|,$$

  where $\mathcal{A}_t$ denotes the best multilinear rank $(s_1, \ldots, s_d)$ approximation.
- The Tucker decomposition has a link with the tensor rank decomposition. Indeed, let $\mathcal{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ be a tensor of rank r with an orthogonal Tucker decomposition

$$\mathcal{A} = [\![\mathcal{S}; U_1, \ldots, U_d]\!] = \sum_{i_1=1}^{r_1} \ldots \sum_{i_d=1}^{r_d} s_{i_1 \ldots i_d} u_{i_1}^1 \otimes \ldots \otimes u_{i_d}^d,$$

  then one rank decomposition of $\mathcal{A}$ can be written as $\mathcal{A} = \sum_{i=1}^{r} (U_1 s_i^1) \otimes \ldots \otimes (U_d s_i^d)$, where $\mathcal{S} = \sum_{i=1}^{r} s_i^1 \otimes \ldots \otimes s_i^d$ is a rank decomposition of the core tensor $\mathcal{S}$. This means once an orthogonal Tucker decomposition is computed to the tensor $\mathcal{A}$ it is sufficient to compute a rank decomposition of the core tensor $\mathcal{S}$ instead of the tensor $\mathcal{A}$ itself. In this regards, Tucker compression techniques are useful to reduce the computational complexity, especially when r_i are significantly smaller than n_i, there exists efficient algorithms for Tucker compression, for instance [39, 75, 159].

CHAPTER 3

Riemannian Optimization

In this chapter we review Riemannian optimization techniques. Note that we address this topic in a concise way where we incorporate the essential information and notions to our purpose. Nevertheless, for a detailed, precised and pedagogic presentation of the notions introduced in this chapter, it is recommended to consult references like [2, 132, 87]. In Section 3.1, we define Riemannian manifolds, then in Section 3.2 we describe the concept of Riemannian optimization on Riemannian manifolds. Next, Section 3.3 is devoted to present the main Riemannian optimization tools : the Riemannian gradient and Hessian, vector transport, and retraction operator, also we introduce these notions for Riemannian submanifolds in Section 3.3.5. First and second order Riemannian optimization algorithms that we will use are presented in Section 3.4 : Riemannian conjugate gradient algorithm, Riemannian Newton algorithm, Riemannian Gauss-Newton algorithm, and Riemannian trust-region scheme (with dogleg steps). Finally, in Section 3.5 we present the Riemannian manifolds that interest us in this thesis : Sphere, Segre and Veronese manifolds, the general linear group and the oblique manifold.

3.1 Riemannian manifold

A smooth manifold $\mathcal{M}$ is a set which is locally in diffeomorphism with a vector space, and which admits a globally defined differential structure. In other words, for each point x in $\mathcal{M}$ the tangent space $T_x\mathcal{M}$ contains the tangent vectors on $\mathcal{M}$ at x which generalize the notion of a directional derivative. Let $\gamma : \mathbb{R} \to \mathcal{M}$ be a smooth curve on $\mathcal{M}$ that passes through the point x at 0 i.e. $\gamma(0) = x$, the derivative of γ at 0 is given by the classical formula :

$$\dot{\gamma}(0) := \lim_{t \to 0} \frac{\gamma(t) - \gamma(0)}{t}.$$

The tangent space on $\mathcal{M}$ at x is $\{\xi_x := \dot{\gamma}(0) : \gamma \text{ is a smooth curve on } \mathcal{M} \text{ such that } \gamma(0) = x\}$. It can be seen as a linear approximation of the manifold $\mathcal{M}$ in the neighborhood of x.
A Riemannian manifold is a couple $(\mathcal{M}, g)$ such that $\mathcal{M}$ is a smooth manifold and g is a smoothly varying inner product on $\mathcal{M}$ called a Riemannian metric on $\mathcal{M}$. More precisely, at each point x the tangent space $T_x\mathcal{M}$ is endowed with an inner product $g_x := \langle ., . \rangle_x$ (i.e. a bilinear, symmetric positive-definite form). This induces the following norm on $T_x\mathcal{M}$

$$\|\xi_x\|_x := \sqrt{\langle \xi_x, \xi_x \rangle}.$$

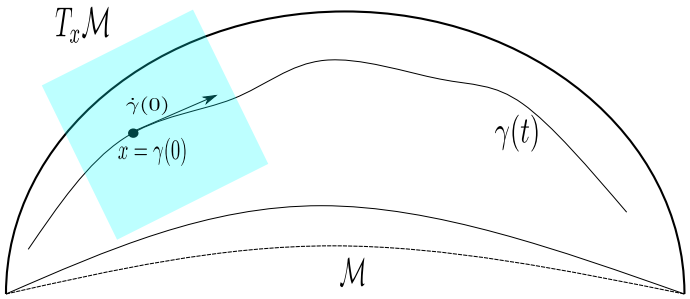


Figure 3.1 – Illustration of a tangent vector $\dot{\gamma}(0)$ of $\mathcal{M}$ at x.

3.2 Riemannian optimization

A Riemannian optimization problem consists in minimizing a smooth function $f : \mathcal{M} \to \mathbb{R}$ on the domain $\mathcal{M}$ which is a Riemannian manifold i.e.

$$\min f(x) \text{ s.t. } x \in \mathcal{M}.$$

Riemannian optimization methods aim to generate, starting from an initial guess $x_0 \in \mathcal{M}$, a sequence $x_1, x_2, \ldots$ that remains in $\mathcal{M}$ and that converges to a local minimum of f constrained to $\mathcal{M}$.

In a common basic form, Riemannian optimization methods are following the process given by :

$$x_{i+1} = R_{x_i}(t\xi_i).$$

- The tangent vector $\xi_i \in T_{x_i}\mathcal{M}$ is the search direction i.e. the direction according to which f will decrease locally, that is the directional derivative of f at x_i according to the search direction also called descent direction ξ_i denoted by $Df(x_i)[\xi_i]$ is strictly negative. It can be obtained using the first-order (gradient) or second-order (Hessian) information.
- The scalar $t > 0$ is called the step size which is used to guarantee a sufficient decrease of f in x_{i+1} according to the descent direction ξ_i. Two typical strategies to choose t are the line-search methods like Armijo backtracking, and trust region methods.
- As we can see $t\xi_i$ is in $T_{x_i}\mathcal{M}$. To define the new point on the Riemannian manifold $\mathcal{M}$ we use a smooth operator $R_x : T_x\mathcal{M} \to \mathcal{M}$ called retraction which replaces the straight ray in $T_{x_i}\mathcal{M}$ with a curve that locally lies on $\mathcal{M}$ such that moving along this curve is considered as moving in the direction of ξ_i.

An advantage of Riemannian optimization is that there exists, in the general case, proofs of convergence of the Riemannian optimization algorithms. These proofs are closely related to the results from Euclidean unconstrained optimization (see [2]).

In our current context, we consider in particular the case where the real valued function f takes the form :

$$f : \mathcal{M} \to \mathbb{R},\ x \mapsto \tfrac{1}{2}\|F(x)\|^2,\ \text{with } F : \mathcal{M} \to \mathcal{E},\ x \mapsto F(x), \tag{3.1}$$

where $\mathcal{E}$ is a Euclidean space. Thus minimizing this f is a *Riemannian least-squares minimization problem*.

We review in Section 3.3 in more details the ingredients of Riemannian optimization methods (gradient, Hessian, retraction, ...). Then in Section 3.4 we recall the Riemannian optimization

algorithms which we will use in this thesis. In Section 3.5 we review the Riemannian manifolds involved in the Riemannian least-squares problems which we will consider in this thesis.

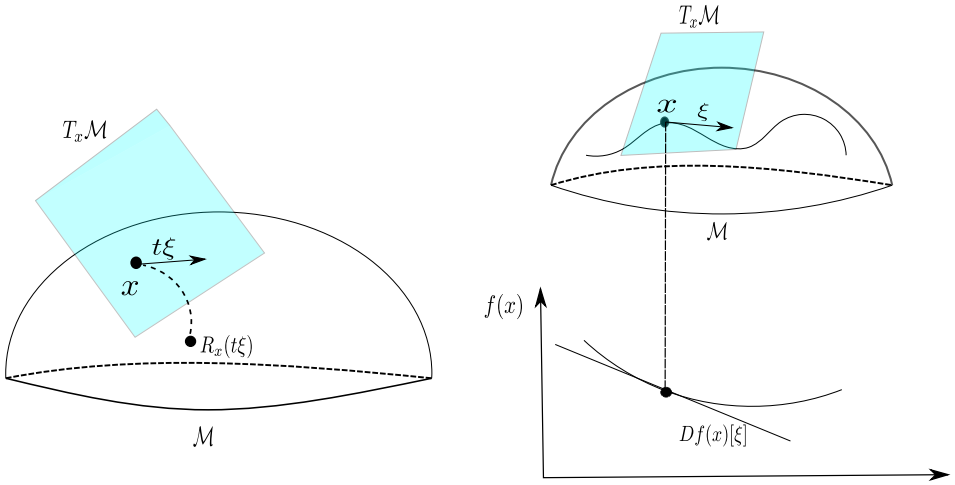


Figure 3.2 – Illustration of one step of a typical Riemannian optimization method. On the right at x a descent direction ξ which leads to a decrease in f (i.e. $Df(x)[\xi] < 0$). On the left, the new point is defined on the manifold $\mathcal{M}$ by using a retraction operator.

3.3 Riemannian optimization tools

Hereafter in this section we present the main tools for Riemannian optimization methods.

3.3.1 Riemannian gradient

Let f be a smooth real valued function on a Riemannian manifold $\mathcal{M}$ the *gradient* of f at x denoted by $\operatorname{grad} f(x)$ is defined as the unique element of $T_x\mathcal{M}$ such that :

$$\langle \operatorname{grad} f(x), \xi \rangle_x = Df(x)[\xi], \ \forall \xi \in T_x\mathcal{M}.$$

An interesting property of $\operatorname{grad} f(x)$ is that its direction is the steepest-ascent direction of f at x. A tangent vector ξ in $T_x\mathcal{M}$ is called a descent direction if $\langle \operatorname{grad} f(x), \xi \rangle_x < 0$.
If we assume f is as in (3.1) then for all $\xi \in T_x\mathcal{M}$ we have :

$$\langle \operatorname{grad} f(x), \xi \rangle_x = Df(x)[\xi] = \langle DF(x)[\xi], F(x) \rangle = \langle \xi, (DF(x))^*[F(x)] \rangle,$$

where $\langle ., . \rangle$ denotes the inner product on $\mathcal{E}$, and $(DF(x))^*$ denotes the adjoint of the operator $DF(x) : T_x\mathcal{M} \to \mathcal{E}$. Thus we have :

$$\operatorname{grad} f(x) = (DF(x))^*[F(x)]. \tag{3.2}$$

3.3.2 Riemannian Hessian

To define the Riemannian Hessian operator, we need first to introduce some important notions. A vector field ξ is a function such that for each $x \in \mathcal{M}$ it associates a tangent vector $\xi_x \in T_x\mathcal{M}$.

An affine connection on a smooth manifold is also an important geometric object in differential geometry. It permits to connect nearby tangent spaces and generalizes directional derivatives of vector fields. Coming back to our Riemannian manifolds context, let $\mathcal{M}$ be a Riemannian manifold with metric $\langle ., . \rangle$, the connection of Levi-Civita denoted by ∇ is the unique affine connection that verifies Koszul formula

$$2\left\langle \nabla_{\xi_x}\eta_x, \nu_x \right\rangle_x = \mathrm{D}\left\langle \eta_x, \nu_x \right\rangle_x [\xi_x] + \mathrm{D}\left\langle \xi_x, \nu_x \right\rangle_x [\eta_x] - \mathrm{D}\left\langle \xi_x, \eta_x \right\rangle_x [\nu_x] + \left\langle \nu_x, [\xi_x, \eta_x] \right\rangle_x + \left\langle \eta_x, [\nu_x, \xi_x] \right\rangle_x - \left\langle \xi_x, [\eta_x, \nu_x] \right\rangle_x ,$$

where ξ_x, η_x and ν_x are the vector fields ξ, η and ν evaluated at x and $[., .]$ is the Lie bracket [*].
The connection of Levi-Civita is very important on a Riemannian manifold $\mathcal{M}$ with a metric $\langle ., . \rangle$, it permits for instance to define the Hessian operator, geodesics and distance on $\mathcal{M}$.
Note that the gradient operator of f is a vector field, since for each $x \in \mathcal{M}$, $\operatorname{grad} f(x)$ is a tangent vector in $T_x\mathcal{M}$. Thus the Hessian operator is by definition the linear mapping from $T_x\mathcal{M}$ to itself such that :

$$\operatorname{Hess} f(x)[\xi_x] = \nabla_{\xi_x} \operatorname{grad} f(x). \tag{3.3}$$

If we assume that f is as in (3.1) then for all ξ, η in $T_x\mathcal{M}$ we have :

$$\langle \operatorname{Hess} f(x)[\eta], \xi \rangle_x = \langle DF(x)[\xi], DF(x)[\eta] \rangle + \langle F(x), \nabla^2 F(x)[\xi, \eta] \rangle, \tag{3.4}$$

where $\langle ., . \rangle$ denotes the inner product on $\mathcal{E}$ and $\nabla^2 F$ denotes *the second covariant derivative of* F (see [2, Chapter 8 and Section 5.6] for the proof).

3.3.3 Retraction

The geodesics on a Riemannian manifold $\mathcal{M}$ with metric $\langle ., . \rangle$ generalize the notion of *straight lines*. They are given by curves $\gamma : I \subset \mathbb{R} \to \mathcal{M}$ on $\mathcal{M}$ with zero acceleration. In our context, this means by using the connection of Levi-Civita :

$$\nabla_{\dot\gamma(t)} \dot\gamma(t) = 0.$$

Further, they are characterised by the initial choice of the initial point $\gamma(0) = x \in \mathcal{M}$ and the initial direction $\dot\gamma(0) = \xi \in T_x\mathcal{M}$. Geodesics allow to define distance on $\mathcal{M}$ with the metric $\langle ., . \rangle$. Indeed, for x, $y \in \mathcal{M}$ and γ a geodesic on $\mathcal{M}$ such that $\gamma(0) = x$ and $\gamma(1) = y$, the distance $d(x, y)$ between x and y is defined by :

$$d^2(x, y) = \int_0^1 \langle \dot\gamma(t), \dot\gamma(t) \rangle_{\gamma(t)} dt.$$

For every $\xi \in T_x\mathcal{M}$ there exits an interval $I = [0, 1]$ and a unique geodesic $\gamma(t; x, \xi) : I \to \mathcal{M}$ such that $\gamma(0) = x$, $\dot\gamma(0) = \xi$. The mapping

$$\operatorname{Exp}_x : T_x\mathcal{M} \to \mathcal{M}, \xi \mapsto \gamma(1; x, \xi),$$

is called the *exponential* map at x. It maps along geodesics in direction of the tangent vector. Intuitively, exponential map is a natural way from a differential geometry point of view used in Riemannian optimization methods, to turn an increment $x + \xi$ in $T_x\mathcal{M}$ into a new point on the

∗. For two matrices A, B in $\mathbb{R}^{n\times n}$ the Lie bracket is given by $[A, B] = AB - BA$.

Riemannian manifold $\mathcal{M}$. Nevertheless, the exponential map may be very complicated or very expensive to compute. Herein, a *retraction* map R which corresponds to the first order approximation of the exponential map i.e.

$$R_x(\xi) = x + \xi + o(\|\xi\|),$$

is an adequate alternative choice. Formally, the definition of a retraction map is given by the following :

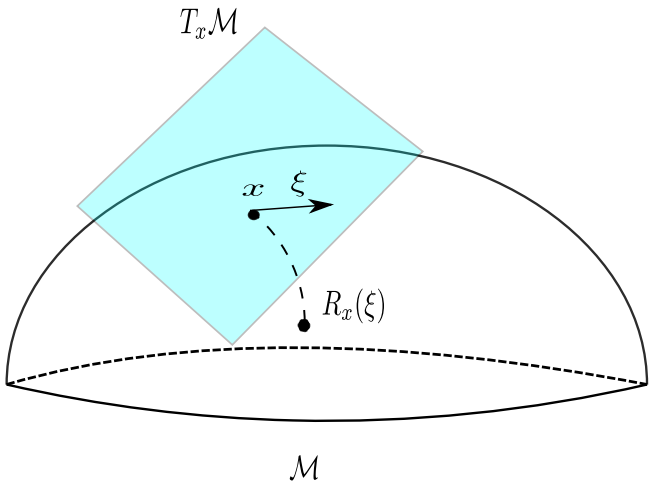


Figure 3.3 – Retraction.

Definition 3.3.1 (Retraction)**.** [2, Chapter 4] Let $\mathcal{M}$ be a manifold and $x \in \mathcal{M}$. A retraction R_x is a map $T_x\mathcal{M} \to \mathcal{M}$, which satisfies the following properties :

1. $R_x(0_x) = x$;
2. there exists an open neighborhood $\mathcal{U}_x \subset T_x\mathcal{M}$ of 0_x such that the restriction on $\mathcal{U}_x$ is well-defined and smooth;
3. R_x satisfies the local rigidity condition

$$DR_x(0_x) = id_{T_x\mathcal{M}},$$

where $id_{T_x\mathcal{M}}$ denotes the identity map on $T_x\mathcal{M}$.

The rigidity condition insures that for every tangent vector $\xi \in T_x\mathcal{M}$, moving along the curve $\gamma_\xi : t \mapsto R_x(t\xi)$ is moving in the direction of ξ, since $\dot{\gamma}_\xi(0) = \frac{d}{dt}R_x(t\xi) \mid_{t=0} = DR_x(0_x)[\xi] = id_{T_x\mathcal{M}}[\xi] = \xi$. It is an important property for Riemannian optimization purposes.

Finally, we recall the following easy-to-proof property of retraction on a Cartesian product of Riemannian manifolds :

Lemma 3.3.1. *Let $\mathcal{M}_1, \ldots, \mathcal{M}_r$ be manifolds, $x_i \in \mathcal{M}_i$ and $\mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_r$ and $x = (x_1, \ldots, x_r) \in \mathcal{M}$. Let $R_i : T_{x_i}\mathcal{M}_i \to \mathcal{M}_i$ be retractions. Then $R_x : T_x\mathcal{M} \to \mathcal{M}$ defined as follows : $R_x(\xi_1, \ldots, \xi_r) = (R_{x_1}(\xi_1), \ldots, R_{x_r}(\xi_r))$ for $\xi_i \in T_{x_i}\mathcal{M}_i$, $1 \leq i \leq r$, is a retraction on $\mathcal{M}$.*

3.3.4 Vector transport

Another important notion in Riemannian optimization, is the *vector transport* [2, Section 8.1], which is used to transport a tangent vector ξ in $T_x\mathcal{M}$ to a tangent vector in $T_{R_x(\eta)}\mathcal{M}$ (see Figure 3.5). Similarly to the retraction which approximates the exponential mapping to the first-order, vector transport is the first-order approximation of parallel translation along geodesics, and it is used rather than the parallel translation for the same reason which the retraction is used rather than the exponential map i.e. to reduce the computational cost while keeping the convergence properties of the algorithm. Concerning parallel transport, briefly, it corresponds to transport a tangent vector $\nu(0)$ in $T_{\gamma(0)}\mathcal{M}$ along the curve $\gamma(t)$ in such a way $\nu(t)$ in $T_{\gamma(t)}\mathcal{M}$ corresponds to the tangent vector $\nu(0)$ in $T_{\gamma(0)}\mathcal{M}$ (see Figure 3.4) in the sense that the transport along the curve γ is parallel according to the metric $\langle ., . \rangle$ on $\mathcal{M}$. In other words, using the Levi-Civita connection, the parallel transport ν along the curve γ is given by the solution of :

$$\nabla_{\dot\gamma(t)}\nu(t) = 0.$$

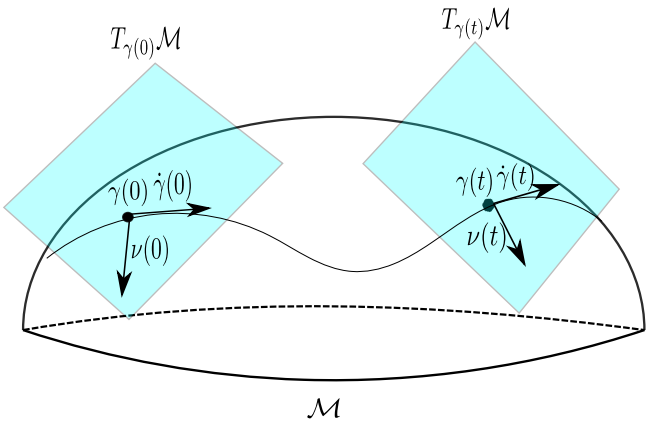


Figure 3.4 – Parallel transport ν on $\mathcal{M}$ along the curve γ.

Coming back to vector transport, its definition uses the so-called *Whitney sum* given by :

$$T\mathcal{M} \oplus T\mathcal{M} = \bigcup_{x\in\mathcal{M}} \{x\} \times T_x\mathcal{M} \times T_x\mathcal{M} = \{(x, \xi, \eta), x \in \mathcal{M}, \xi \in T_x\mathcal{M}, \eta \in T_x\mathcal{M}\}.$$

Definition 3.3.2 (Vector transport)**.** A vector transport on the manifold $\mathcal{M}$ is a smooth mapping : $T\mathcal{M} \oplus T\mathcal{M} \to T\mathcal{M},\ (x, \eta, \xi) \mapsto \mathcal{T}_\eta(\xi)$ satisfying the following properties for all $x \in \mathcal{M}$.

1. There exists a retraction R such that, for all $(x, \eta, \xi) \in T\mathcal{M} \oplus T\mathcal{M}$, it holds that $\mathcal{T}_\eta(\xi) \in T_{R_x(\eta)}\mathcal{M}$.
2. $\mathcal{T}_0(\xi) = \xi$ for all $\xi \in T_x\mathcal{M}$.
3. For all $(x, \eta) \in T\mathcal{M}$, the mapping $\mathcal{T}_\eta : T_x\mathcal{M} \to T_{R_x(\eta)}\mathcal{M}, \xi \mapsto \mathcal{T}_\eta(\xi)$ is linear.

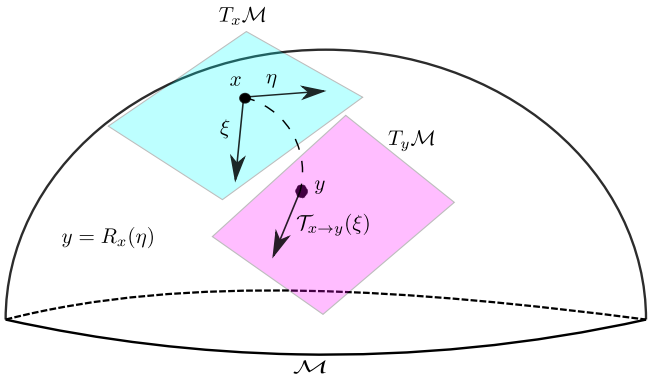


Figure 3.5 – Vector transport on $\mathcal{M}$.

3.3.5 Riemannian optimization tools for Riemannian submanifolds

Let $\mathcal{M}$ be an embedded submanifold of a Riemannian manifold $\overline{\mathcal{M}}$ endowed with a metric $\langle .,. \rangle$. For each $x \in \mathcal{M}$, the tangent space $T_x\mathcal{M}$ is a subspace of $T_x\overline{\mathcal{M}}$. Hence, $\mathcal{M}$ turns into a Riemannian manifold, called a Riemannian submanifold of $\overline{\mathcal{M}}$, by simply inheriting $\mathcal{M}$ the metric $\langle .,. \rangle$ from $\overline{\mathcal{M}}$.
Any element $\xi \in T_x\overline{\mathcal{M}}$ can be uniquely decomposed as follows :

$$\xi = P_x(\xi) + P_x^{\perp}(\xi),$$

where P_x denotes the orthogonal projection onto $T_x\mathcal{M}$ and $P_x^{\perp}$ denotes the orthogonal projection onto the orthogonal complement of $T_x\mathcal{M}$ called the *normal space to $\mathcal{M}$ at x* and given by

$$(T_x\mathcal{M})^{\perp} := \{\eta \in T_x\overline{\mathcal{M}} : \langle \eta, \xi \rangle_x = 0,\ \forall \xi \in T_x\mathcal{M}\}.$$

It follows that the connection of Levi-Civita on $\mathcal{M}$ denoted by ∇ can be deduced from the connection of Levi-Civita on $\overline{\mathcal{M}}$ denoted by $\overline{\nabla}$:

$$\nabla_{\xi_x}\eta_x = P_x(\overline{\nabla}_{\xi_x}\eta_x), \tag{3.5}$$

where $x \in \mathcal{M}$, and ξ_x, η_x are vector fields evaluated at x.
Moreover, if $\overline{f}$ is a smooth real valued function on $\overline{\mathcal{M}}$ and f is the restriction of $\overline{f}$ on $\mathcal{M}$ then $\forall x \in \mathcal{M}$ we have :

$$\operatorname{grad} f(x) = P_x(\operatorname{grad} \overline{f}(x)). \tag{3.6}$$

Further, the Hessian of f at x is obtained by substituting (3.5) and (3.6) in (3.3).

In particular if $\mathcal{M}$ is an embedded submanifold of a Euclidean space $\mathcal{E}$, then (3.5) reads :

$$\nabla_{\xi_x}\eta_x = P_x(D\eta_x[\xi_x]),$$

with $D\eta_x[\xi_x]$ is the classical directional derivative of η at x in the direction of ξ_x. Also, in this case (3.6) reads :

$$\operatorname{grad} f(x) = P_x(\partial_x \overline{f}(x)),$$

where $\partial_x \overline{f}(x)$ is the classical Euclidean gradient of $\overline{f}$ at x. Finally, if $x, y \in \mathcal{M}$ such that $y = R_x(\eta)$ where $\eta \in T_x\mathcal{M}$, given a tangent vector $\xi \in T_x\mathcal{M}$, we have $\xi \in T_x\mathcal{M} \subset T_x\mathcal{E} \simeq \mathcal{E}$, thus

ξ can be transported to $T_y\mathcal{M}$ by simply use the orthogonal projector from $\mathcal{E}$ onto $T_y\mathcal{M}$ (see [2, Section 8.1.3]) i.e.

$$\mathcal{T}_\eta(\xi) := P_{R_x(\eta)}(\xi).$$

We note that in our context we will, in most cases, deal with the aforementioned type of Riemannian manifold i.e. embedded Riemannian submanifold in a Euclidean space.

3.4 Riemannian optimization algorithms

In this section we review some of the existing Riemannian optimization algorithms in the literature, which we will use in this thesis in the contributions part.
Let $f : \mathcal{M} \to \mathbb{R}$ be a smooth real valued function defined on a Riemannian manifold $\mathcal{M}$ with a Riemannian metric $\langle ., . \rangle$. We recall from Section 3.2 that a Riemannian optimization algorithm aims to solve locally the minimization problem :

$$\min f(x) \text{ s.t. } x \in \mathcal{M}. \tag{3.7}$$

We note that our review does not include an analysis for the convergence of these algorithms. For this purpose as well as for more details concerning these algorithms see for instance [2].

3.4.1 Riemannian conjugate gradient method

The Riemannian conjugate gradient method generalizes the typical conjugate gradient method on Euclidean space. The search direction at step i in a classical conjugate gradient method is computed by combining the steepest descent direction i.e. $\xi_i := -\operatorname{grad} f(x_i)$ with the search direction in the previous step η_{i-1} so that the new search direction is given by :

$$\eta_i = \xi_i + \beta_i \eta_{i-1},$$

and then

$$x_{i+1} = x_i + \alpha_i \eta_i,$$

where α_i is the step-size that determines how far x_i should moves along the direction η_i.
To generalize the method to Riemannian manifolds, we have to take into account the following points :

- The search direction step involves the sum of an element η_{i-1} lies in $T_{x_{i-1}}\mathcal{M}$ with $\xi_i \in T_{x_i}\mathcal{M}$, thus η_{i-1} should be transported to $T_{x_i}\mathcal{M}$ by using vector transport $\mathcal{T}_{x_{i-1}\to x_i}$.
- We can choose one of the two following popular choice to compute β_i :

$$\beta_i = \frac{\langle \operatorname{grad} f(x_i), \operatorname{grad} f(x_i) - \mathcal{T}_{x_{i-1}\to x_i}(\operatorname{grad} f(x_{i-1})) \rangle}{\langle \operatorname{grad} f(x_{i-1}), \operatorname{grad} f(x_{i-1}) \rangle} \text{ (Polak-Ribière)}, \tag{3.8}$$

$$\beta_i = \frac{\langle \operatorname{grad} f(x_i), \operatorname{grad} f(x_i) \rangle}{\langle \operatorname{grad} f(x_{i-1}), \operatorname{grad} f(x_{i-1}) \rangle} \text{ (Fletcher-Reeves)}. \tag{3.9}$$

- We need to use a retraction operator $R_{x_i} : T_{x_i}\mathcal{M} \to \mathcal{M}$ to map the tangent vectors to the manifold.

— Perform a line-search with the conjugate direction η_i along the curve $t \mapsto R_{x_i}(t\eta_i)$. A standard Armijo backtracking step size can be used to choose t :
Let $\alpha > 0, \beta, \sigma \in (0,1)$, find the smallest nonnegative integer m such that :

$$f(x_i) - f(R_{x_i}(\beta^m \alpha \eta_i)) \geq -\sigma \langle \operatorname{grad} f(x_i), \beta^m \alpha \eta_i \rangle,$$

take $t = \beta^m \alpha$.

We give the skeleton of the Riemannian conjugate gradient algorithm with Armijo backtracking line search step in Algorithm 3.1.

Require: Riemannian manifold $\mathcal{M}$; vector transport $\mathcal{T}$ on $\mathcal{M}$ with associated retraction R; cost function f on $\mathcal{M}$; initial iterate $x_1 \in \mathcal{M}$, tangent vector $\eta_0 = 0$.

1: **for** $i = 1, 2, \ldots$ **do**
2: Compute the gradient $\xi_i := \operatorname{grad} f(x_i)$;
3: Compute a conjugate direction $\eta_i := -\xi_i + \beta_i \mathcal{T}_{x_{i-1} \to x_i}(\eta_{i-1})$;
4: Perform Armijo backtracking to find the smallest integer $m \geq 0$ such that for given $\alpha > 0, \beta, \sigma \in (0,1)$

$$f(x_i) - f(R_{x_i}(\beta^m \alpha \eta_i)) \geq -\sigma \langle \xi_i, \beta^m \alpha \eta_i \rangle;$$

5: Compute the next new point $x_{i+1} := R_{x_i}(\beta^m \alpha \eta_i)$;
6: **end for**

Algorithm 3.1 – Riemannian conjugate gradient method with Armijo line-search.

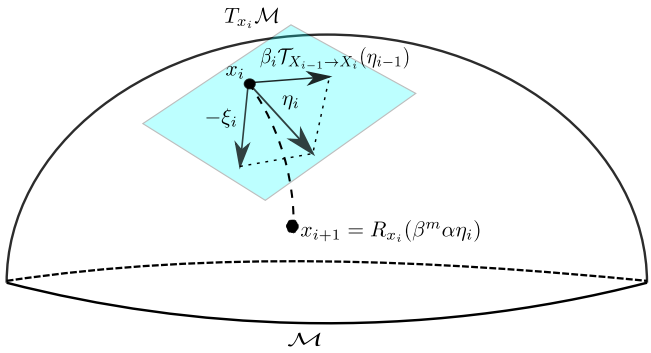


Figure 3.6 – Visualization of one iteration of Algorithm 3.1.

3.4.2 Riemannian Newton and Riemannian Gauss–Newton methods

A Riemannian Newton method is a geometric generalization of the classical Newton method. Recall that if F is a smooth function from $\mathbb{R}$ to $\mathbb{R}$ such that $F(x_*) = 0$, then Newton's method consists of starting with an initial guess $x_0 \in \mathbb{R}$ and generating a sequence $x_1, x_2, \ldots$ in $\mathbb{R}$, such that :

$$x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)},$$

where F' is the derivative of F. By rewritten this expression as

$$F(x_k) + F'(x_k)(x_{k+1} - x_k) = 0,$$

this means graphically that x_{k+1} is the intersection of the tangent to the graph of F at x with the horizontal axis. This can be generalize to a function $F : \mathbb{R}^n \to \mathbb{R}^n$ by taking :

$$F(x_k) + DF(x_k)[x_{k+1} - x_k] = 0, \tag{3.10}$$

where $DF(x_k)[z]$ denotes the directional derivative of F at x along z. Now, if we consider the minimization problem in (3.7) but with constrained set equal to $\mathbb{R}^n$, since the aim is to find a local minimum of f, this means that we seek a critical point i.e. $\operatorname{grad} f(x_*) = 0$, thus by replacing F by $\operatorname{grad} f$, the Newton equation (3.10) reads :

$$\operatorname{grad} f(x_k) + D(\operatorname{grad} f)(x_k)[x_{k+1} - x_k] = 0.$$

Finally, to generalize this approach to Riemannian manifolds, $x_{k+1} - x_k$ is replaced by a tangent vector $\eta_k \in T_{x_k}\mathcal{M}$, $\operatorname{grad} f(x_k)$ is the Riemannian gradient of f at x_k and $D(\operatorname{grad} f)(x_k)$ is replaced by the Riemannian Hessian $\operatorname{Hess} f(x_k)$. Using a retraction operator R on $\mathcal{M}$, the new point x_{k+1} is obtained from η_k by $x_{k+1} = R_{x_k}(\eta_k)$.

Formally, a Riemannian Newton method for solving (3.7) [2, Chapter 6] consists of starting with an initial guess $x_0 \in \mathcal{M}$ and generating a sequence $x_1, x_2, \ldots$ in $\mathcal{M}$, with respect to the following process :

$$x_{k+1} \leftarrow R_{x_k}(\eta_k) \quad \text{with} \operatorname{Hess} f(x_k)[\eta_k] = -\operatorname{grad} f(x_k); \tag{3.11}$$

where $\operatorname{grad} f(x_k)$ and $\operatorname{Hess} f(x_k)$ are respectively the Riemannian gradient and Hessian of f at x_k on $\mathcal{M}$, and $R_{x_k} : T_{x_k}\mathcal{M} \to \mathcal{M}$ is a retraction operator from the tangent space $T_{x_k}\mathcal{M}$ to $\mathcal{M}$.

Riemannian Gauss–Newton method is a quasi–Newton method i.e. an approximation of the Riemannian Newton method for the case where $f = \frac{1}{2}\|F\|^2$ as in (3.1). Recall from (3.4), that with this f, we have :

$$\langle \operatorname{Hess} f(x)[\eta], \xi\rangle_x = \langle DF(x)[\xi], DF(x)[\eta]\rangle + \langle F(x), \nabla^2 F(x)[\xi, \eta]\rangle, \ \forall \xi, \eta \in T_x\mathcal{M}.$$

The Riemannian Gauss–Newton method consists in approximating $\langle \operatorname{Hess} f(x)[\eta], \xi\rangle_x$ by the first term $\langle DF(x)[\xi], DF(x)[\eta]\rangle$. By injecting this in the Riemannian Newton equation in (3.11), we obtain the so-called Riemannian Gauss–Newton equation

$$((DF(x_k))^* \circ DF(x_k))[\eta_k] = -(DF(x_k))^*[F(x_k)],$$

since in this case $\operatorname{grad} f(x_k) = (DF(x_k))^*[F(x_k)]$ (see (3.2)). The operator $(DF(x_k))^* \circ DF(x_k)$ is called the Riemannian Gauss–Newton approximation of the Riemannian Hessian $\operatorname{Hess} f(x_k)$. Thus, technically, the Riemannian Gauss–Newton method differs from the Riemannian Newton method by solving the Gauss–Newton equation instead of the exact Newton equation. Obviously, the Riemannian Gauss–Newton method is of lower computational complexity than the Riemannian Newton method. On the other hand, Riemannian Gauss–Newton method is in general not superlinearly convergent, in contrast with the Riemannian Newton method which has a local superlinear rate of convergence (see [2, Theorem 6.3.2]). Nevertheless, in practice, Riemannian Newton and Gauss–Newton methods are combined with a line-search step or with a trust-region scheme in order to ensure a sufficient decrease in the cost function f at each iteration. The next section is devoted to describe the Riemannian trust-region scheme with a dogleg step.

3.4.3 Riemannian trust region scheme with dogleg steps

The Riemannian Newton (resp. Gauss–Newton) method is looking for a critical point of a real-valued function f, without distinguishing between local minimizer, saddle point and local maximizer. Furthermore, the convergence of this algorithm may not occur from the beginning. For these reasons, a trust region scheme is usually added to such algorithm in order to enhance the algorithm, with the desirable properties of convergence to a local minimum, with a local superlinear rate of convergence. In fact, trust region method ensures that f decreases at each iteration, which reinforces, when convergence occurs, the possibility of finding a local minimizer. Nevertheless, a global convergence to a local minimizer from any initial points is not guaranteed even after adding the trust region scheme (see [2, Subsection 7.4.1] for the global convergence of Riemannian trust-region methods). The idea is to approximate the objective function f by its second order Taylor series expansion in a ball of center $0_{x_k} \in T_{x_k}\mathcal{M}$ and radius Δ_k denoted by $B_{\Delta_k} := \{\eta \in T_{x_k}\mathcal{M} \mid ||\eta|| \leq \Delta_k\}$, and to solve the so-called *trust-region subproblem*

$$\min_{\eta \in B_{\Delta_k}} m_{x_k}(\eta), \tag{3.12}$$

where $m_{x_k}(\eta) := f(x_k) + \langle \operatorname{grad} f(x_k), \eta \rangle_{x_k} + \frac{1}{2}\langle H_k[\eta], \eta \rangle_{x_k}$, H_k is some symmetric operator on $T_{x_k}\mathcal{M}$ (in the case of Riemannian Newton method H_k is the Hessian operator of f at x_k and it is the Riemannian Gauss–Newton approximation $(DF(x_k))^* \circ DF(x_k)$ for the Riemannian Gauss–Newton method).

By solving (3.12), we obtain a solution $\eta_k \in T_{x_k}\mathcal{M}$. Accepting or rejecting the candidate new point $x_{k+1} = R_{x_k}(\eta_k)$, as well as updating the trust region Δ_k is based on the quotient

$$\rho_k = \frac{f(x_k) - f(x_{k+1})}{m_{x_k}(0) - m_{x_k}(\eta_k)}.$$

One approach to solve the trust-region subproblem (3.12) is the so-called dogleg method :
Let η_N be the Newton direction given by $H_k[\eta_N] = -\operatorname{grad} f(x_k)$, let η_c denote the Cauchy point given by $\eta_c = -\frac{\langle \operatorname{grad} f(x_k), \operatorname{grad} f(x_k) \rangle_{x_k}}{\langle H_k[\operatorname{grad} f(x_k)], \operatorname{grad} f(x_k) \rangle_{x_k}} \operatorname{grad} f(x_k)$, and let η_I be the intersection of the boundary of the sphere B_Δ and the vector pointing from η_c to η_N. Then the optimal solution η^* of (3.12) by the dogleg method is given as follows :

$$\eta^* = \begin{cases} \eta_N & \text{if } ||\eta_N|| \leq \Delta, \\ -\frac{\Delta}{||\operatorname{grad} f(x_k)||} \operatorname{grad} f(x_k) & \text{if } ||\eta_N|| > \Delta \text{ and } ||\eta_c|| \geq \Delta, \\ \eta_I & \text{otherwise.} \end{cases}$$

3.5 Riemannian manifolds of interest

In this section we present five manifolds that we use thereafter in this thesis : the sphere, Segre and Veronese manifolds, the general linear group and the oblique manifold.

3.5.1 The unit sphere

The unit sphere in $\mathbb{R}^n$

$$\mathcal{S}^{n-1} := \{x \in \mathbb{R}^n \mid x^T x = 1\};$$

is one of the simplest Riemannian manifolds. The inner product is inherited from the standard inner product on $\mathbb{R}^n$ i.e.

$$\langle \xi, \eta \rangle_x := \xi^T \eta.$$

Its tangent space at $x \in \mathcal{S}^{n-1}$ is given by

$$T_x\mathcal{S}^{n-1} = \{z \in \mathbb{R}^n \mid x^T z = 0\}.$$

The normal space is

$$(T_x\mathcal{S}^{n-1})^\perp = \{x\alpha \mid \alpha \in \mathbb{R}\}.$$

The projection on $T_x\mathcal{S}^{n-1}$ respectively on $(T_x\mathcal{S}^{n-1})^\perp$ are given by

$$P_x(\xi) = (I_n - xx^T)\xi,\ P_x^\perp(\xi) = xx^T\xi.$$

The map $R_x : T_x\mathcal{S}^{n-1} \to \mathcal{S}^{n-1},\ \xi \mapsto \frac{x+\xi}{\|x+\xi\|}$ is a retraction operator on $\mathcal{S}^{n-1}$.

All the aforementioned information with their proofs can be found in [2].

3.5.2 Segre manifold

The *Segre manifold* in $\mathbb{R}^{n_1 \times \ldots n_d}$ consists of all tensors of rank 1 :

$$\mathcal{S}_{n_1,\ldots,n_d} := \{a_1 \otimes \ldots \otimes a_d \mid a_k \in \mathbb{R}^{n_k}\} - \{0\}.$$

It is an embedded submanifold in $\mathbb{R}^{n_1 \times \ldots \times n_d}$, thus $\mathcal{S}_{n_1,\ldots,n_d}$ is a Riemannian submanifold by tacking the metric induced by this ambient Euclidean space. The Segre manifold can be parameterized using the following local diffeomorphism :

$$\begin{aligned}\Phi : \mathbb{R}_+^* \times \mathcal{S}^{n_1-1} \times \ldots \times \mathcal{S}^{n_d-1} &\to \mathcal{S}_{n_1,\ldots,n_d} \\ (\alpha, a_1, \ldots, a_d) &\mapsto \alpha a_1 \otimes \ldots \otimes a_d.\end{aligned}$$

Let $x = (\alpha, a_1, \ldots, a_d)$. This allows to identify the tangent space of $\mathcal{S} := \mathcal{S}_{n_1,\ldots,n_d}$ at x as follows :

$$T_x(\mathbb{R}_+^* \times \mathcal{S}^{n_1-1} \times \ldots \times \mathcal{S}^{n_d-1}) \cong \mathbb{R} \times T_{a_1}\mathcal{S}^{n_1-1} \times \ldots \times T_{a_d}\mathcal{S}^{n_d-1},$$

with

$$\begin{aligned}D_x\Phi : \mathbb{R} \times T_{a_1}\mathcal{S}^{n_1-1} \times \ldots \times T_{a_d}\mathcal{S}^{n_d-1} &\to T_{\alpha a_1 \otimes \ldots \otimes a_d}\mathcal{S} \\ (\dot\alpha, \dot y_1, \ldots, \dot y_d) &\mapsto \dot\alpha a_1 \otimes \ldots \otimes a_d + \alpha(\dot y_1 \otimes a_2 \otimes \ldots \otimes a_d + \ldots \\ &\qquad + a_1 \otimes \ldots \otimes a_{d-1} \otimes \dot y_d).\end{aligned}$$

There exists a retraction operator on $\mathcal{S}$ called T-HOSVD retraction [122]. It is defined as the rank-$(1, \ldots, 1)$ T-HOSVD approximation of $\mathcal{A} + p$, such that $\mathcal{A} \in \mathcal{S}$, $p \in T_\mathcal{A}\mathcal{S}$, as follows :

$$R_\mathcal{A}(p) = (q_1 q_1^T, \ldots, q_d q_d^T).(\mathcal{A} + p),$$

where q_k is the first left singular vector from the SVD decomposition of the matrix $(\mathcal{A}+p)_{(k)}$ i.e. the *kth* mode of flattening of $\mathcal{A} + p$.

3.5.3 Veronese manifold

Similarly to the Segre manifold, the *Veronese manifold* in $\mathcal{T}^d(\mathbb{R}^n)$, i.e. the space of real symmetric tensors of dimension n and order d, contains all rank-1 symmetric tensors in $\mathcal{T}^d(\mathbb{R}^n)$. By considering the correspondence between $\mathcal{T}^d(\mathbb{R}^n)$ and $\mathbb{R}[\mathbf{x}]_d$, the Veronese manifold is given by the set of linear forms to the d^{th} power :

$$\mathcal{V}_n^d(\mathbb{R}) := \{(v^t x)^d \mid v \in \mathbb{R}^n\} - \{0\}.$$

Let $p = (v^t x)^d \in \mathcal{V}_n^d(\mathbb{R})$, the tangent space $T_p\mathcal{V}_n^d(\mathbb{R})$ is given by :

$$T_p\mathcal{V}_n^d(\mathbb{R}) = \{(u^t x)(v^t x)^{d-1} \mid u \in \mathbb{R}^n\}.$$

Similarly to the Segre manifold, the Veronese manifold is an embedded submanifold in $\mathbb{R}[\mathbf{x}]_d$. Herein, endowed with the metric inherited from $\mathbb{R}[\mathbf{x}]_d$ (for instance the apolar product defined on $\mathbb{R}[\mathbf{x}]_d \times \mathbb{R}[\mathbf{x}]_d$ in Definition 2.2.1 which is an inner product on $\mathbb{R}[\mathbf{x}]_d$) $\mathcal{V}_n^d(\mathbb{R})$ is a Riemannian submanifold.
We will investigate this Riemannian manifold in Chapter 4 in more details especially in the complex field, in order to developp Riemannian based optimization algorithms for the symmetric low rank approximation problem for symmetric tensors with complex coefficients.

3.5.4 The general linear group

The information presented in this section can be found in [14, 133, 147, 213, 225, 25]. The set of invertible matrices of size $n \times n$ denoted by GL_n is open in $\mathbb{R}^{n\times n}$, thus it is a manifold. In addition, it is a group under matrix multiplication. Consequently, GL_n is by definition a Lie matrix group, called the general linear group. Since it is open in $\mathbb{R}^{n\times n}$, its tangent space $T_B\mathrm{GL}_n$ at any point $B \in \mathrm{GL}_n$ can be identified with $\mathbb{R}^{n\times n}$. Since GL_n is a Lie matrix group, thus an element $\xi^B \in T_B\mathrm{GL}_n$ can be written as $\xi^B = BX$ or $\xi^B = XB$ such that X is in the Lie algebra which is by definition $T_{I_n}\mathrm{GL}_n \simeq \mathbb{R}^{n\times n}$, herein $\xi^B = BX$ or $\xi^B = XB$ with $X \in \mathbb{R}^{n\times n}$. We will use the aforementioned information in Section 5.1 to apply perturbations on elements of GL_n. In Section 5.1, $\mathrm{GL}_n(\mathbb{C})$ is viewed as a complex Lie matrix group. However, the same information remains applicable in this case.
The manifold GL_n is equipped with either the left-invariant metric or the right-invariant which are for all $B \in \mathrm{GL}_n$, $\xi^B, \eta^B \in T_B\mathrm{GL}_n$ given by :

$$\langle \xi^B, \eta^B \rangle_B^\ell = \mathrm{tr}(B^{-1}\xi^B(B^{-1}\eta^B)^t),\ \langle \xi^B, \eta^B \rangle_B^r = \mathrm{tr}(\xi^B B^{-1}(\eta^B B^{-1})^t)$$

Let $\gamma_\ell : \mathbb{R} \to \mathrm{GL}_n,\ \gamma_r : \mathbb{R} \to \mathrm{GL}_n$ be the geodesics of GL_n with respect to respectively the left- and right-invariant metrics, for all $B \in \mathrm{GL}_n$, $\xi^B \in T_B\mathrm{GL}_n$ they are given by :

$$\gamma_\ell(t) = B\exp(t(B^{-1}\xi^B)^T)\exp(t(B^{-1}\xi^B - (B^{-1}\xi^B)^T)) \tag{3.13}$$

$$\gamma_r(t) = \exp(t(\xi^B B^{-1} - (\xi^B B^{-1})^T))\exp(t(\xi^B B^{-1})^T)B \tag{3.14}$$

where $\exp(.)$ denote the matrix exponential. We denote the exponential map resulting from (3.13) and (3.13) by $\exp_B^\ell : T_B\mathrm{GL}_n \to \mathrm{GL}_n,\ \exp_B^r : T_B\mathrm{GL}_n \to \mathrm{GL}_n$. Naturally, these exponential maps are retractions on GL_n.
The vector transports on GL_n according to the left- and right-invariant metrics are given by :

$$\mathcal{T}_\ell(B, \xi^B, \eta^B) = \exp_B^\ell(\xi^B)B^{-1}\eta^B \text{ and } \mathcal{T}_r(B, \xi^B, \eta^B) = \eta^B B^{-1}\exp_B^r(\xi^B).$$

Finally, the Riemannian gradients of a cost function $f : \mathrm{GL}_n \to \mathbb{R}$ at $E \in \mathrm{GL}_n$ equipped with the left or right-invariant metric are given by :

$$\mathrm{grad}_\ell\, f(B) = BB^t\, \mathrm{grad}_{\mathrm{Euc}}\, f(B),\ \mathrm{grad}_r\, f(B) = \mathrm{grad}_{\mathrm{Euc}}\, f(B) B^t B,$$

where $\mathrm{grad}_{\mathrm{Euc}}\, f(B)$ is the classical Euclidean gradient of f at $B \in \mathrm{GL}_n$.

3.5.5 Oblique manifold

The results that we are going to present in this section come from [25, 1].
Let

$$\mathcal{M}_n^o := \{B \in \mathrm{GL}_n : \mathrm{ddiag}(BB^t) = I_n\}$$

denote the set of all invertible matrices in GL_n with normalized rows. The set $\mathcal{M}_n^o$ is a submanifold of GL_n. For $B \in \mathcal{M}_n^o$, its tangent space $T_B\mathcal{M}_n^o$ is as follows :

$$T_B\mathcal{M}_n^o = \{\xi \in \mathbb{R}^{n\times n} \mid \mathrm{ddiag}(\xi B^t) = 0\}.$$

The manifold $\mathcal{M}_n^o$ is a Riemannian manifold inheriting the left or right-invariant metric of GL_n. We summarize in Table 3.1, the necessary tools according to the left- or right-invariant metric, that we need for a Riemannian gradient based method (we refer the reader to [25] for the proofs of the formulas stated in the table).

metric	$\langle ., . \rangle^\ell_B$	$\langle ., . \rangle^r_B$
projection map onto $T_B\mathcal{M}_n^o$	$P_B^{ob,\ell}(Y) = Y - BB^t\Delta B$, Δ is a diagonal matrix with $\mathrm{diag}(\Delta) = (BB^t * BB^t)^{-1}\,\mathrm{diag}(YB^t)$	$P_B^{ob,r}(Y) = Y - \Delta BB^tB$, $\Delta = \mathrm{ddiag}(YB^t)\,\mathrm{ddiag}((BB^t)^2)^{-1}$
gradient	$\mathrm{grad}^\ell_{ob}\, f(B) = P_B^{ob,\ell}(\mathrm{grad}_\ell\, f(B))$	$\mathrm{grad}^r_{ob}\, f(B) = P_B^{ob,r}(\mathrm{grad}_r\, f(B))$
retraction	$R_B^{ob,\ell}(\xi^B) = \Delta(\exp^\ell_B(\xi^B))\exp^\ell_B(\xi^B)$ with $\Delta(Y) = \mathrm{ddiag}(YY^t)^{-\frac{1}{2}}$	$R_B^{ob,r}(\xi^B) = \Delta(\exp^r_B(\xi^B))\exp^r_B(\xi^B)$
vector transport	$\mathcal{T}^{ob,\ell}_{B\to R_B^{ob,\ell}(\xi^B)}(\eta^B) = P^{ob,\ell}_{R_B^{ob,\ell}(\xi^B)}(R_B^{ob,\ell}(\xi^B)B^{-1}\eta^B)$	$\mathcal{T}^{ob,r}_{B\to R_B^{ob,r}(\xi^B)}(\eta^B) = P^{ob,r}_{R_B^{ob,r}(\xi^B)}(\eta^B B^{-1} R_B^{ob,r}(\xi^B))$

Table 3.1 – Main tools for Riemannian optimization on the oblique manifold $\mathcal{M}_n^o$.

Contributions

CHAPTER 4

Riemannian Newton optimization algorithms for the symmetric tensor approximation problem

The Symmetric Tensor Approximation problem (STA) consists of approximating a symmetric tensor or a homogeneous polynomial by a linear combination of symmetric rank-1 tensors or powers of linear forms of low symmetric rank. In this chapter, we present two Riemannian Newton-type algorithms for low rank approximation of symmetric tensor with complex coefficients.

The first algorithm uses the parametrization of the set of tensors of rank at most r by weights and unit vectors. Exploiting the properties of the apolar product on homogeneous polynomials combined with efficient tools from complex optimization, we provide an explicit and tractable formulation of the Riemannian gradient and Hessian, leading to Newton iterations with local quadratic convergence. We prove that under some regularity conditions on non-defective tensors in the neighborhood of the initial point, the Newton iteration (completed with a trust-region scheme) is converging to a local minimum.

The second algorithm is a Riemannian Gauss–Newton method on the Cartesian product of Veronese manifolds. An explicit orthonormal basis of the tangent space of this Riemannian manifold is described. We deduce the Riemannian gradient and the Gauss–Newton approximation of the Riemannian Hessian. We present a new retraction operator on the Veronese manifold.

We analyze the numerical behavior of these methods, with an initial point provided by Simultaneous Matrix Diagonalisation (SMD). Numerical experiments show the good numerical behavior of the two methods in different cases and in comparison with existing state-of-the-art methods.

Keywords : *symmetric tensor decomposition, homogeneous polynomials, Riemannian optimization, Newton method, retraction, complex optimization, trust region method, Veronese manifold.*

In this chapter, we describe in section 4.1 the set of non-defective rank-r symmetric tensors. In subsection 4.2.1, we formulate the STA problem as a Riemannian least square optimization problem using the parametrization by weights and unit vectors. We compute explicitly the Riemannian gradient vector and the Hessian matrix in subsection 4.2.1.1 and describe the retraction in subsection 4.2.1.2. In subsection 4.2.2, we describe the Riemannian Gauss–Newton method on the product of Veronese manifolds. We present in subsection 4.2.2.1 a new retraction operator on the Veronese manifold with its analysis. In subsection 4.2.3, we recall the trust-region extension scheme, and prove under some regularity assumptions the convergence of the exact Riemannian Newton method with trust region steps to a local minimum of the distance function. Numerical experiments are featured in section 4.3. The final section 4.5 is for our conclusions.

4.1 The set of non-defective rank-r symmetric tensors

Let $\Sigma_r \subset \mathcal{S}_n^d$ be the set of symmetric tensors of symmetric rank at most r. A symmetric tensor $\boldsymbol{t} \in \Sigma_r$ is the sum of d^{th} powers

$$\boldsymbol{t} = \sum_{i=1}^{r} (v_i^t \mathbf{x})^d, \text{ for } v_i \in \mathbb{C}^n. \tag{4.1}$$

It is a point in the image of the following map :

$$\begin{aligned} \psi_r : \mathbb{C}^{n\times r} := \mathbb{C}^n \times \cdots \times \mathbb{C}^n &\longrightarrow \mathbb{C}[\mathbf{x}]_d \\ [v_i]_{1\le i\le r} &\longmapsto \psi_r((v_i)_{1\le i\le r}) = \sum_{i=1}^{r} (v_i^t \mathbf{x})^d. \end{aligned}$$

The d^{th} power $(v_i^t \mathbf{x})^d$ with $v_i \neq 0$ are symmetric tensors of rank-1, which are on the so-called Veronese manifold.

Definition 4.1.1. Let $\psi : \mathbb{C}^n \to \mathbb{C}[\mathbf{x}]_d,\ v \mapsto (v^t\mathbf{x})^d = \sum_{|\alpha|=d} \binom{d}{\alpha} v^\alpha \mathbf{x}^\alpha$. The Veronese manifold in $\mathbb{C}[\mathbf{x}]_d$ denoted by $\mathcal{V}^{n,d}$ is the set of linear forms in $\mathbb{C}[\mathbf{x}]_1 - \{0\}$ to the d^{th} power. It is the image of $\mathbb{C}^n - \{0\}$ by ψ.

The Veronese variety studied in algebraic geometry is the algebraic variety of the projective space $\mathbb{P}^{s_{n,d}-1}$ associated to $\mathcal{V}^{n,d}$, where $s_{n,d} = \dim \mathbb{C}[\mathbf{x}]_d$ [112, 226, 129]. The tangent space of $\mathcal{V}^{n,d}$ at a point $p = (v^t\mathbf{x})^d$ is the vector space spanned by $\langle x_1(v^t\mathbf{x})^{d-1}, \ldots, x_n(v^t\mathbf{x})^{d-1}\rangle$, that is the linear space $T_p(\mathcal{V}^{n,d}) = \{(u^t\mathbf{x})(v^t\mathbf{x})^{d-1} \mid u \in \mathbb{C}^n\}$.

The Zariski closure $\overline{\Sigma}_r$ of Σ_r is called the $(r-1)$*th-secant variety* of the Veronese variety. For $r > 1$, the algebraic variety $\overline{\Sigma}_r$ is not smooth and contrarily to the case of matrices, singular points of $\overline{\Sigma}_r$ can have a rank $> r$, as shown in the following example. For $d > 2$, $\boldsymbol{p} = (v_0^t\mathbf{x})(v_1^t\mathbf{x})^{d-1} \in \mathbb{C}[\mathbf{x}]_d$ with $v_0 \neq v_1 \in \mathbb{C}^n$ is in the (Zariski) closure of $\overline{\Sigma}_2$ since $(v_0^t\mathbf{x})(v_1^t\mathbf{x})^{d-1} = \lim_{\delta\to 0} \frac{1}{d\delta}(((v_1+\delta v_0)^t\mathbf{x})^d - (v_1^t\mathbf{x})^d)$ but its symmetric rank is $d > 2$ [54, Proposition 5.6].

To avoid these singularities, we will restrict our theoretical analysis to points of Σ_r where the map ψ_r is a local embedding, since in the vicinity of singularities, the best low rank approximation problem is ill-posed (as shown by the previous example). The map ψ_r is a local embedding at $y = [v_i]_{1\le i\le r} \in \mathbb{C}^{n\times r}$ iff

$$J\psi_r(y) = d\,[x_1(v_1^t\mathbf{x})^{d-1}, \ldots, x_n(v_1^t\mathbf{x})^{d-1}, \ldots, x_1(v_r^t\mathbf{x})^{d-1}, \ldots, x_n(v_r^t\mathbf{x})^{d-1}]$$

is of rank $n\,r$. The tensors $\psi_r(y)$ with $y \in \mathbb{C}^{n\times r}$ such that $\operatorname{rank} J\psi_r(y) = n\,r$ are called *non-defective*. The set of non-defective tensors of rank r, locally embedded in $\mathbb{C}[\mathbf{x}]_d$, is the image by a local diffeomorphism of a Riemannian manifold and it is denoted Σ_r^{reg}. The map ψ_r is a local diffeomorphism between an open subset of $\mathbb{C}^{n\times r}$ and $\Sigma_r^{\text{reg}} \subset \overline{\Sigma}_r$.

Hereafter, we consider the cases where $d > 2$ and the rank r is strictly subgeneric, i.e. $r < r_g = \lceil \frac{1}{n}\binom{n+d-1}{d} \rceil$, where r_g is the generic symmetric rank (except for $(d,n) \in \{(3,5),(4,3),(4,4),(4,5)\}$ or $d = 2$) by Alexander–Hirschowitz theorem [10]. Using "Terracini's lemma" (see e.g. [129, Lemma 5.3.1.1]), we have that Σ_r^{reg} is a dense open subset of $\overline{\Sigma}_r$ iff the dimension of $\overline{\Sigma}_r$ is the expected dimension $n\,r$. In this case, $\overline{\Sigma}_r$ is also said to be *non-defective*. Alexander and Hirschowitz [10] proved that $\overline{\Sigma}_r$ is non-defective when $r < r_g$ (the exceptional defective cases for $d > 2$ being $(d,n,r) \in \{(3,5,7),(4,3,5),(4,4,9),(4,5,14)\}$).

It is also known that for $r < r_g$, generic tensors of ψ_r have a unique decomposition, i.e. a unique inverse image by ψ_r up to permutations, except for $(d,n,r) \in \{(6,2,9),(4,3,8),(3,5,9)\}$, see [47, Theorem 1.1].

4.2 Riemannian optimization for the STA problem

In this section, we use the framework of Riemannian optimization [2] to solve the STA problem. See also [33, 91, 123] for real multilinear tensors. We develop a Riemannian Newton algorithm and a Riemannian Gauss–Newton algorithm exploiting the properties of symmetric tensors to obtain explicit and simplified formulation. We consider distance minimization problems for symmetric tensors with complex decompositions for both algorithms.
Given $\boldsymbol{p} \in \mathcal{S}_n^d \sim \mathbb{C}[\mathbf{x}]_d$, we consider hereafter the following least square minimization problem

$$\min_{y\in\mathcal{M}} f(y) \tag{4.2}$$

where $f : \mathcal{M} \to \mathbb{R}$ is half the square distance function to $\boldsymbol{p}$ i.e. $f(y) = \frac{1}{2}\|F(y)\|_d^2$ with $F(y) = \Phi_r(y) - \boldsymbol{p}$, such that $\Phi_r : \mathcal{M} \to \mathbb{C}[\mathbf{x}]_d$ is a parametrization map of Σ_r the set of symmetric tensors of symmetric rank bounded by r, and $\mathcal{M}$ is a Riemannian manifold. A Riemannian optimization method for solving (4.2) requires a Riemannian metric. Since we will assume that $\mathcal{M}$ is embedded in some space $\mathbb{R}^M$, we will take the metric induced by the Euclidean space $\mathbb{R}^M$.

We propose to parametrize Σ_r, first via weights and unit vectors. We describe an exact Riemannian Newton method for this formulation in subsection 4.2.1. Secondly, we parametrize Σ_r via sums of the d^{th} power of linear forms that is as sums of tensors in $\mathcal{V}^{n,d}$. We develop a Riemannian Gauss–Newton method for this formulation in subsection 4.2.2. A dogleg trust-region scheme in subsection 4.2.3 is added to the two algorithms.

4.2.1 Riemannian Newton method for STA

We normalize the decomposition (4.1) by choosing unit vectors for v_i and positive weights. Namely, we decompose a symmetric tensor $\boldsymbol{p} \in \Sigma_r$ as $\boldsymbol{p} = \sum_{i=1}^r w_i\,(v_i^t\mathbf{x})^d$ with $w_i \in \mathbb{R}_+^*$ and $||v_i|| = 1$, for $1 \le i \le r$; by normalizing v_i and multiplying by $w_i := ||v_i||^d$ if v_i is not a unit vector. The vector $(w_i)_{1\le i\le r}$ in this decomposition is called "the weight vector", and is denoted by W. Let $V = [v_i]_{1\le i\le r} \in \mathbb{C}^{n\times r}$ be the matrix of the normalized vectors.

The objective function expressed in terms of these weights and unit vectors is given by $f(W,V) = \frac{1}{2}||F(W,V)||_d^2$, with $F(W,V) = \sum_{i=1}^r w_i\,(v_i^t\mathbf{x})^d - \boldsymbol{p}$.

The function f is a real valued function of complex variables; such function is non-analytic, because it cannot satisfy the Cauchy–Riemann conditions [175]. To apply the Riemannian Newton method, we need the second order differentials of f. As discussed in [191], we overcome the non-analytic problem by converting the optimization problem to the real domain, regarding f as a function of the real and imaginary parts of its complex variables.

Let $\mathcal{N}_r = \{(W, \Re(V), \Im(V)) \mid W \in {\mathbb{R}_+^*}^r, V \in \mathbb{C}^{n\times r}, (\Re(v_i), \Im(v_i)) \in \mathbb{S}^{2n-1}, \forall\, 1 \leq i \leq r\}$, where $\mathbb{S}^{2n-1}$ is the unit sphere in $\mathbb{R}^{2n}$. Let $\varphi_r : (w, v_1, \ldots, v_r, v'_1, \ldots, v'_r) \in \mathcal{N}_r \mapsto \sum_{i=1}^r w_i((v_i + \mathbf{i}\, v'_i)^t \mathbf{x})^d$. Hereafter in this subsection, we use the following formulation to compute the different ingredients of a Riemannian Newton method :

$$\text{(STA)}_{\mathcal{N}_r} \quad \min_{y\in\mathcal{N}_r} f(y),$$

where $f(y) = \frac{1}{2}||F(y)||_d^2$, with $F(y) = \varphi_r(y) - \boldsymbol{p}$.

4.2.1.1 Computation of the gradient vector and the Hessian matrix

In this section, we present the explicit expressions of the Riemannian gradient and Hessian on $\mathcal{N}_r$. We first describe an orthonormal basis of $T_y\mathcal{N}_r$ for $y \in \mathcal{N}_r$. Then we detail the computation of the gradient and Hessian in this basis, via the differentials of maps in complex and conjugate variables.

Lemma 4.2.1. *Let $y = (w, v_1, \ldots, v_r, v'_1, \ldots, v'_r) \in \mathcal{N}_r$. For all $i = 1, \ldots, r$ let $\check{v}_i = (v_i; v'_i) \in \mathbb{S}^{2n-1}$ and let*

$$(I_{2n} - \check{v}_i\check{v}_i^t) = Q_i R_i P_i$$

be a rank-revealing QR-decomposition of the projector on $\check{v}_i^{\perp}$ in $\mathbb{R}^{2n}$, where $Q_iQ_i^t = I_{2n}$, R_i is upper triangular, and P_i is a permutation matrix.

Let $Q_{i,re}$ (resp. $Q_{i,im}$) be the matrix given by the first n rows (resp. the last n rows) and the first $2n-1$ columns of Q_i. Let $\tilde{Q} = \begin{bmatrix} Q_{re} \\ Q_{im} \end{bmatrix} \in \mathbb{R}^{2nr\times(2n-1)r}$, where $Q_{re} = \operatorname{diag}(Q_{i,re})_{1\leq i\leq r}$ and $Q_{im} = \operatorname{diag}(Q_{i,im})_{1\leq i\leq r}$. Then the columns of $Q = \operatorname{diag}(I_r, \tilde{Q})$ form an orthonormal basis of $T_y\mathcal{N}_r$.

Proof. We have $T_y\mathcal{N}_r \simeq T_w(\mathbb{R}_+^*)^r \times T_Z\mathcal{S}_r$, where $\mathcal{S}_r = \{(\Re(V), \Im(V)) \mid V \in \mathbb{C}^{n\times r}, ||v_i||^2 = 1,\ \forall 1 \leq i \leq r\}$ and $Z = (\Re(V), \Im(V)) = (v_1, \ldots, v_r, v'_1, \ldots, v'_r) \in \mathbb{R}^{n\times 2r}$.

As $T_w(\mathbb{R}_+^*)^r = \mathbb{R}^r$, I_r represents an orthonormal basis of $T_w(\mathbb{R}_+^*)^r$.

We verify now that $\tilde{Q}$ is an orthonormal basis of $T_Z\mathcal{S}_r$. For $i = 1, \ldots, r$, $\check{v}_i \in \mathbb{S}^{2n-1} \subset \mathbb{R}^{2n}$ and the first $(2n-1)$ columns of the factor Q_i of a rank-revealing QR-decomposition of $I_{2n} - \check{v}_i\check{v}_i^t$ give an orthonormal basis of the image $\check{v}_i^{\perp}$ of $(I_{2n} - \check{v}_i\check{v}_i^t)$, that is $T_{\check{v}_i}\mathbb{S}^{2n-1}$.

The vector space $T_Z\mathcal{S}_r$, of dimension $r(2n-1)$, is the Cartesian product of the tangent spaces $T_{\check{v}_i}\mathbb{S}^{2n-1}$. Therefore, by construction, the columns of $\tilde{Q}$ form an orthonormal basis of $T_Z\mathcal{S}_r$.

We deduce that $Q = \operatorname{diag}(I_r, \tilde{Q})$ represents an orthonormal basis of $T_y\mathcal{N}_r$ in the canonical basis of $\mathbb{R}^{2nr}$. □

Let $\mathcal{R}_r = \{(W, \Re(V), \Im(V)) \in \mathbb{R}^r \times \mathbb{R}^{n\times r} \times \mathbb{R}^{n\times r} \mid W \in \mathbb{R}^r, V \in \mathbb{C}^{n\times r}\}$ and let f_R be the function f seen as a function on $\mathcal{R}_r$. The gradient and the Hessian of f_R at a point $p^R \in \mathcal{R}_r$

are called the real gradient and the real Hessian. We denote them by G^R and H^R. We will describe their computation, after the next proposition, relating them to the Riemannian gradient and Hessian.

Proposition 4.2.2. *Let $p = (w, v_1, \ldots, v_r, v'_1, \ldots, v'_r) \in \mathcal{N}_r$, $Q \in \mathbb{R}^{(r+2nr)\times(r+(2n-1)r)}$ such that its columns form an orthonormal basis of $T_y\mathcal{N}_r$. Let $G^R = (g_0, g_1, \ldots, g_r, g'_1, \ldots, g'_r) \in \mathbb{R}^{r+2nr}$ (resp. $H^R \in \mathbb{R}^{(r+2nr)\times(r+2nr)}$) be the gradient vector (resp. the Hessian matrix) of f_R at p^R in the canonical basis. The Riemannian gradient vector (resp. Hessian matrix) of f at p with respect to the basis Q is given by :*

$$G = Q^t G^R, H = Q^t(H^R + S)Q,$$

where $S = \mathrm{diag}(0_{r\times r}, \tilde{S}, \tilde{S})$, with $\tilde{S} = \mathrm{diag}(s_1 I_n, \ldots, s_r I_n)$, $s_i = \langle v_i, g_i\rangle + \langle v'_i, g'_i\rangle$.

Proof. Let $y = (w, v_1, \ldots, v_r, v'_1, \ldots, v'_r) \in \mathcal{N}_r$. Let $\mathcal{P}_y$ be the orthogonal projector on $T_y\mathcal{N}_r$. Let $Q \in \mathbb{R}^{(r+2nr)\times(r+(2n-1)r)}$ such that its columns form an orthonormal basis of the image of $\mathcal{P}_y$ or equivalently of $T_y\mathcal{N}_r$. As the Riemannian gradient of f is the projection of Df_R, the first order differentials of f_R, on the tangent space $T_y\mathcal{N}_r$ [2, Chapter 5], we have $G = Q^t G^R$, where G^R is the vector which represents the classical first order partial derivatives of f_R at y^R in the canonical basis.
Let $\eta \in T_y\mathcal{N}_r$, $z \in T_y\mathcal{N}_r^\perp$. We have from [3] that the Riemannian Hessian matrix of f at y is given by the formula : $H\eta = \mathcal{P}_y H^R \eta + \mathfrak{U}_y(\eta, \mathcal{P}_y^\perp G^R)$, where H^R is the matrix of the second order derivatives of f_R at y^R in the canonical basis, $\mathfrak{U}_y$ is the Weingarten map on $\mathcal{N}_r$ at y given by $\mathfrak{U}_y(\eta, z) = \mathcal{P}_y D_\eta \mathcal{P} z$, where $\mathcal{P}$ is a matrix valued function on $\mathcal{N}_r$ determined as follows : $\mathcal{P} : y \in \mathcal{N}_r \mapsto \mathcal{P}_y$, and $D_\eta \mathcal{P} z$ represent the time derivative of $y \mapsto \mathcal{P}_y z$ in terms of the time derivative of y i.e. $\dot{y} \in T_y\mathcal{N}_r$ applied at $\dot{y} = \eta$, and $\mathcal{P}_y^\perp = I - \mathcal{P}_y$ is the orthogonal projector on $T_y\mathcal{N}_r^\perp$.
As $y \in \mathcal{N}_r$ we have $w \in {\mathbb{R}_+^*}^r$, and $\check{v}_i := (v_i, v'_i) \in \mathbb{S}^{2n-1}$, $\forall 1 \le i \le r$. Let $u = (u_0, u_1, \ldots, u_r, u'_1, \ldots, u'_r) \in \mathbb{R}^{r+2nr}$, such that $\check{u}_i = (u_i, u'_i)$, $\forall 1 \le i \le r$. Let $\mathcal{P}_w$ (resp. $\mathcal{P}_{\check{v}_i}$) denote the orthogonal projector on $T_w(\mathbb{R}_+^*)^r = \mathbb{R}^r$ (resp. $T_{\check{v}_i}\mathbb{S}^{2n-1}$), we have that : $\mathcal{P}_w(u_0) = u_0$, $\mathcal{P}_{\check{v}_i}\check{u}_i = (I_{2n} - \check{v}_i\check{v}_i^t)\check{u}_i$, $\forall 1 \le i \le r$, thus :

$$\mathcal{P}_y u = \begin{pmatrix} u_0 \\ ((I_{2n} - \check{v}_1\check{v}_1^t)\check{u}_1)[1:n] \\ \vdots \\ ((I_{2n} - \check{v}_r\check{v}_r^t)\check{u}_r)[1:n] \\ ((I_{2n} - \check{v}_1\check{v}_1^t)\check{u}_1)[n+1:2n] \\ \vdots \\ ((I_{2n} - \check{v}_r\check{v}_r^t)\check{u}_r)[n+1:2n] \end{pmatrix} = \begin{pmatrix} u_0 \\ u_1 - v_1\check{v}_1^t\check{u}_1 \\ \vdots \\ u_r - v_r\check{v}_r^t\check{u}_r \\ u'_1 - v'_1\check{v}_1^t\check{u}_1 \\ \vdots \\ u'_r - v'_r\check{v}_r^t\check{u}_r \end{pmatrix}, \mathcal{P}_y^\perp u = \begin{pmatrix} 0_r \\ v_1\check{v}_1^t\check{u}_1 \\ \vdots \\ v_r\check{v}_r^t\check{u}_r \\ v'_1\check{v}_1^t\check{u}_1 \\ \vdots \\ v'_r\check{v}_r^t\check{u}_r \end{pmatrix}.$$

Let $\mathfrak{U}_{\check{v}_i}$ be the Weingarten map on $\mathbb{S}^{2n-1}$ at $\check{v}_i$. For $\eta = (\eta_0, \eta_1, \ldots, \eta_r, \eta'_1, \ldots, \eta'_r) \in T_y\mathcal{N}_r$, and $z = (z_0, z_1, \ldots, z_r, z'_1, \ldots, z'_r) \in T_y\mathcal{N}_r^\perp$ with $\check{\eta}_i = (\eta_i, \eta'_i) \in T_{\check{v}_i}\mathbb{S}^{2n-1}$ and $\check{z}_i = (z_i, z'_i) \in T_{\check{v}_i}\mathbb{S}^{2n-1\perp}$, $\forall 1 \le i \le r$, we have from [3] : $\mathfrak{U}_{\check{v}_i}(\check{\eta}_i, \check{z}_i) = -\check{\eta}_i\check{v}_i^t\check{z}_i$. Thus, with respect to the

parameterization that we consider we find that :

$$\mathfrak{U}_y(\eta, z) = -\begin{pmatrix} 0_r \\ \eta_1 \check{v}_1^t \check{z}_1 \\ \vdots \\ \eta_r \check{v}_r^t \check{z}_r \\ \eta_1' \check{v}_1^t \check{z}_1 \\ \vdots \\ \eta_r' \check{v}_r^t \check{z}_r \end{pmatrix}.$$

Let $G^R = (g_0; g_1; \ldots; g_r; g_1'; \ldots; g_r') \in \mathbb{R}^{r+2nr}$ and $\check{g}_i = (g_i, g_i')$, $l_i = \check{v}_i \check{v}_i^t \check{g}_i$ for $i = 1, \ldots, r$. We obtain $\mathfrak{U}_y(\eta, \mathcal{P}_y^{\perp} G^R)$ by substituting $\check{z}_i$ by l_i in $\mathfrak{U}_y(\eta, z)$. Since $\check{v}_i^t \check{v}_i = ||\check{v}_i||^2 = 1$, we find that

$$\mathfrak{U}_y(\eta, \mathcal{P}_y^{\perp} G^R) = \begin{pmatrix} 0_r \\ \eta_1 \check{v}_1^t \check{g}_1 \\ \vdots \\ \eta_r \check{v}_r^t \check{g}_r \\ \eta_1' \check{v}_1^t \check{g}_1 \\ \vdots \\ \eta_r' \check{v}_r^t \check{g}_r \end{pmatrix} = S\eta, \text{ where } S = \operatorname{diag}(0_{r\times r}, \tilde{S}, \tilde{S}), \text{ with } \tilde{S} = \operatorname{diag}(s_1 I_n, \ldots, s_r I_n),$$

$s_i = \check{v}_i^t \check{g}_i = \langle v_i, g_i \rangle + \langle v_i', g_i' \rangle$. Since, $\mathfrak{U}_y(\eta, z) = \mathcal{P}_y D_\eta \mathcal{P} z$, and $\mathcal{P}_y \circ \mathcal{P}_y = \mathcal{P}_y$, we can write $\mathfrak{U}_y(\eta, z) = \mathcal{P}_y \mathfrak{U}_y(\eta, z)$. Hence, $\mathfrak{U}_y(\eta, \mathcal{P}_y^{\perp} G^R) = \mathcal{P}_y S \eta = \mathcal{P}_y S \mathcal{P}_y \eta$, since $\mathcal{P}_y \eta = \eta$ for $\eta \in T_y \mathcal{N}_r$. Thus we have : $H\eta = \mathcal{P}_y(H^R + S)\mathcal{P}_y \eta$, and then $H = \mathcal{P}_y(H^R + S)\mathcal{P}_y$. Herein, H can be written with respect to the basis Q as follows : $H = Q^t(H^R + S)Q$, which ends the proof. □

We describe now the real gradient and Hessian, by using complex variables and their conjugates. Recall from Brandwood [31] that transforming the pair $(\Re(z), \Im(z))$ of real and imaginary parts of a given complex variable z into the pair $(z, \overline{z})$ is a simple linear transformation, which allows us to achieve explicit and simple computation of the gradient and Hessian of f.
Let $\mathcal{C}_r = \left\{(W, V, \overline{V}) \in \mathbb{R}^r \times \mathbb{C}^{n\times r} \times \mathbb{C}^{n\times r} \mid W \in \mathbb{R}^r,\ V \in \mathbb{C}^{n\times r}\right\}$ and

$$K = \begin{bmatrix} I_r & 0_{r\times 2nr} \\ 0_{2nr\times r} & J \end{bmatrix} \tag{4.3}$$

where $J = \begin{bmatrix} I_{nr} & \mathbf{i} I_{nr} \\ I_{nr} & -\mathbf{i} I_{nr} \end{bmatrix}$. The linear map K is an isomorphism between the $\mathbb{R}$-vector spaces $\mathcal{R}_r$ and $\mathcal{C}_r$. Its inverse is given by $K^{-1} = \begin{bmatrix} I_r & 0_{r\times 2nr} \\ 0_{2nr\times r} & \frac{1}{2} J^* \end{bmatrix}$.

Let f_C be the function f seen as a function on $\mathcal{C}_r$. Considering f_C for the computation of the gradient and the Hessian yields more elegant expressions than considering f_R. For this reason, we compute first the gradient and the Hessian of f_C, and then we use the isomorphism K in (4.3) to get the real gradient and the Hessian of f_R.

Lemma 4.2.3. *The complex gradient G^C can be transformed into the real gradient G^R as follows :*

$$G^R = K^t G^C. \tag{4.4}$$

Similarly H^R and H^C are related by the following formula :

$$H^R = K^t H^C K. \tag{4.5}$$

Proof. See [191] and the references therein. □

Let us describe now explicitly the real gradient G^R :

Proposition 4.2.4. *The gradient G^R of f_R on $\mathcal{R}_r$ is the vector*

$$G^R = \begin{pmatrix} G_1 \\ \Re(G_2) \\ -\Im(G_2) \end{pmatrix} \in \mathbb{R}^{r+2nr},$$

where
- $G_1 = (\sum_{i=1}^r w_i \Re((v_j^* v_i)^d) - \Re(\bar{\boldsymbol{p}}(v_j)))_{1\leq j\leq r} \in \mathbb{R}^r$,
- $G_2 = (d\sum_{i=1}^r w_i w_j (v_i^* v_j)^{(d-1)} \bar{v}_i - w_j \nabla_{\mathbf{x}} \bar{\boldsymbol{p}}(v_j))_{1\leq j\leq r} \in \mathbb{C}^{nr}$.

Proof. We can write f_C as $f_C = \frac{1}{2}(f_1 - f_2 - f_3 + f_4)$, where

$$f_1 = \Big|\Big| \sum_{i=1}^r w_i (v_i^t \mathbf{x})^d \Big|\Big|_d^2 = \sum_{|\alpha|=d} \binom{d}{\alpha} \Big(\sum_{i=1}^r w_i \bar{v}_i^\alpha\Big)\Big(\sum_{i=1}^r w_i v_i^\alpha\Big) \quad \text{(by definition 2.2.1)},$$

$$f_2 = \big\langle \sum_{i=1}^r w_i (v_i^t \mathbf{x})^d, \boldsymbol{p} \big\rangle_d = \sum_{i=1}^r w_i \boldsymbol{p}(\bar{v}_i) \quad \text{(by 1. in lemma 2.2.1)},$$

$$f_3 = \bar{f_2} = \sum_{i=1}^r w_i \bar{\boldsymbol{p}}(v_i), \text{and } f_4 = ||\boldsymbol{p}||_d^2.$$

Let us decompose G^C as $G^C = \begin{pmatrix} G_1 \\ \tilde{G}_2 \\ \tilde{G}_3 \end{pmatrix}$, with $G_1 = (\frac{\partial f_C}{\partial w_j})_{1\leq j\leq r}$, $\tilde{G}_2 = (\frac{\partial f_C}{\partial v_j})_{1\leq j\leq r}$ and $\tilde{G}_3 = (\frac{\partial f_C}{\partial \bar{v}_j})_{1\leq j\leq r}$. As f_C is a real valued function, we have that $\frac{\partial f_C}{\partial \bar{v}_j} = \overline{\frac{\partial f_C}{\partial v_j}}$ [152, 175], thus $\tilde{G}_3 = \overline{\tilde{G}_2}$. Let us start by the computation of G_1 :

$$\begin{aligned} \frac{\partial f_1}{\partial w_j} &= \frac{\partial}{\partial w_j}\Big(\sum_{|\alpha|=d} \binom{d}{\alpha}\Big(\sum_{i=1}^r w_i \bar{v}_i^\alpha\Big)\Big(\sum_{i=1}^r w_i v_i^\alpha\Big)\Big) \\ &= \sum_{|\alpha|=d} \binom{d}{\alpha}\Big(\bar{v}_j^\alpha\Big(\sum_{i=1}^r w_i v_i^\alpha\Big) + v_j^\alpha\Big(\sum_{i=1}^r w_i \bar{v}_i^\alpha\Big)\Big) \\ &= \sum_{i=1}^r w_i (v_j^* v_i)^d + \sum_{i=1}^r w_i (v_i^* v_j)^d = 2\sum_{i=1}^r w_i \Re((v_j^* v_i)^d); \end{aligned}$$

the third equality is deduced by using definition 2.2.1 and 1. of lemma 2.2.1. In addition, we have $\frac{\partial f_2}{\partial w_j} = \frac{\partial}{\partial w_j}(\sum_{i=1}^r w_i \boldsymbol{p}(\bar{v}_i)) = \boldsymbol{p}(\bar{v}_j)$, $\frac{\partial f_3}{\partial w_j} = \bar{\boldsymbol{p}}(v_j)$, and $\frac{\partial f_4}{\partial w_j} = 0$. Thus, $\frac{\partial f_C}{\partial w_j} = \sum_{i=1}^r w_i \Re((v_j^* v_i)^d) - \Re(\bar{\boldsymbol{p}}(v_j))$.

Now, for the computation of $\tilde{G}_2$, let $\boldsymbol{p} = \sum_{|\alpha|=d} \binom{d}{\alpha} \check{v}_\alpha \mathbf{x}^\alpha$, and $1 \leq k \leq n$,

$$\frac{\partial f_1}{\partial v_{j,k}} = \sum_{|\alpha|=d} \binom{d}{\alpha} \Big(\sum_{i=1}^{r} w_i \bar{v}_i^\alpha \Big) (w_j \alpha_k v_j^{\alpha - e_k}) = w_j \sum_{i=1}^{r} w_i \langle \partial_{x_k} (v_i^t \mathbf{x})^d, (v_j^t \mathbf{x})^{d-1} \rangle_{d-1}$$
$$= d w_j \sum_{i=1}^{r} w_i \langle (v_i^t \mathbf{x})^d, x_k (v_j^t \mathbf{x})^{d-1} \rangle_d = d w_j \sum_{i=1}^{r} w_i \bar{v}_{i,k} (v_i^* v_j)^{d-1},$$

the second (resp. third and fourth) equality are deduced by using lemma 2.2.1. Moreover, we have $\frac{\partial f_2}{\partial v_{j,k}} = 0$, $\frac{\partial f_3}{\partial v_{j,k}} = w_j \sum_{|\alpha|=d} \binom{d}{\alpha} \bar{\check{v}}_\alpha \alpha_k v_j^{\alpha - e_k} = w_j \partial_{x_k} \bar{\boldsymbol{p}}(v_j)$, and $\frac{\partial f_4}{\partial v_{j,k}} = 0$. Thus, $\frac{\partial f_C}{\partial v_j} = \frac{1}{2} \Big(d w_j \sum_{i=1}^{r} w_i (v_i^* v_j)^{(d-1)} \bar{v}_i - w_j \nabla_{\mathbf{x}} \bar{\boldsymbol{p}}(v_j) \Big)$.

We have $G^R = K^t G^C$ from (4.4). By multiplication of these two matrices, we obtain : $G^R = \begin{pmatrix} G_1 \\ \tilde{G}_2 + \overline{\tilde{G}_2} \\ \mathbf{i}(\tilde{G}_2 - \overline{\tilde{G}_2}) \end{pmatrix} = \begin{pmatrix} G_1 \\ 2\Re(\tilde{G}_2) \\ -2\Im(\tilde{G}_2) \end{pmatrix}$. Finally dividing by 2, we get $G^R = \begin{pmatrix} G_1 \\ \Re(G_2) \\ -\Im(G_2) \end{pmatrix}$, where $G_2 = 2\tilde{G}_2$, which ends the proof. □

The matrix of the real Hessian can be computed as follows :

Proposition 4.2.5. *The real Hessian matrix H^R is the following block matrix :*

$$H^R = \begin{bmatrix} A & \Re(B)^t & -\Im(B)^t \\ \Re(B) & \Re(C+D) & -\Im(C+D) \\ -\Im(B) & \Im(D-C) & \Re(D-C) \end{bmatrix} \in \mathbb{R}^{(r+2nr)\times(r+2nr)},$$

with

- $A = \Re([(v_i^* v_j)^d]_{1 \leq i,j \leq r}) \in \mathbb{R}^{r \times r}$,
- $B = [d w_i (v_j^* v_i)^{d-1} \bar{v}_j + \delta_{i,j} (d \sum_{l=1}^{r} w_l (v_l^* v_i)^{d-1} \bar{v}_l - \nabla_{\mathbf{x}} \bar{\boldsymbol{p}}(v_j))]_{1 \leq i,j \leq r} \in \mathbb{C}^{nr \times r}$, *where $\delta_{i,j}$ is the Kronecker delta,*
- $C = \mathrm{diag}[d(d-1)[\sum_{i=1}^{r} w_i w_j \overline{v_{i,k} v_{i,l}} (v_i^* v_j)^{d-2}]_{1 \leq k,l \leq n} - w_j \Delta_{\mathbf{x}} \bar{\boldsymbol{p}}(v_j)]_{1 \leq j \leq r} \in \mathbb{C}^{nr \times nr}$, *where $\Delta_{\mathbf{x}} \bar{\boldsymbol{p}}(v_j) := [\partial_{x_k} \partial_{x_l} \bar{\boldsymbol{p}}(v_j)]_{1 \leq k,l \leq n}$,*
- $D = [d w_i w_j (v_i^* v_j)^{d-2} ((v_i^* v_j) I_n + (d-1) v_j v_i^*)]_{1 \leq i,j \leq r} \in \mathbb{C}^{nr \times nr}$.

Proof. H^C is given by the following block matrix :

$$H^C = \begin{bmatrix} \Big[\frac{\partial^2 f_C}{\partial w_i \partial w_j}\Big]_{1 \leq i,j \leq r} & \Big[\frac{\partial^2 f_C}{\partial w_i \partial v_j^t}\Big]_{1 \leq i,j \leq r} & \Big[\frac{\partial^2 f_C}{\partial w_i \partial \bar{v}_j^t}\Big]_{1 \leq i,j \leq r} \\ \Big[\frac{\partial^2 f_C}{\partial v_i \partial w_j}\Big]_{1 \leq i,j \leq r} & \Big[\frac{\partial^2 f_C}{\partial v_i \partial v_j^t}\Big]_{1 \leq i,j \leq r} & \Big[\frac{\partial^2 f_C}{\partial v_i \partial \bar{v}_j^t}\Big]_{1 \leq i,j \leq r} \\ \Big[\frac{\partial^2 f_C}{\partial \bar{v}_i \partial w_j}\Big]_{1 \leq i,j \leq r} & \Big[\frac{\partial^2 f_C}{\partial \bar{v}_i \partial v_j^t}\Big]_{1 \leq i,j \leq r} & \Big[\frac{\partial^2 f_C}{\partial \bar{v}_i \partial \bar{v}_j^t}\Big]_{1 \leq i,j \leq r} \end{bmatrix}.$$

We have that $\frac{\partial^2 f}{\partial \bar{z} \partial \bar{z}^t} = \overline{\frac{\partial^2 f}{\partial z \partial z^t}}$, and $\frac{\partial^2 f}{\partial z \partial \bar{z}^t} = \frac{\partial^2 f}{\partial \bar{z} \partial z^t}$, for a complex variable z and a real valued function with complex variables f. Using these two relations, we find that $\Big[\frac{\partial^2 f_C}{\partial w_i \partial w_j}\Big]_{1 \leq i,j \leq r}$,

$\left[\frac{\partial^2 f_C}{\partial v_i \partial w_j}\right]_{1\leq i,j\leq r}$, $\left[\frac{\partial^2 f_C}{\partial v_i \partial v_j^t}\right]_{1\leq i,j\leq r}$, and $\left[\frac{\partial f_C}{\partial \bar{v}_i \partial v_j^t}\right]_{1\leq i,j\leq r}$ determine H^C. We denote them respectively by A, $\tilde{B}$, $\tilde{C}$, and $\tilde{D}$. Herein, we can decompose H^C as :

$$H^C = \begin{bmatrix} A & \tilde{B}^t & \tilde{B}^* \\ \tilde{B} & \tilde{C} & \tilde{D}^t \\ \overline{\tilde{B}} & \tilde{D} & \overline{\tilde{C}} \end{bmatrix}.$$

The computation of these four matrices can be done by taking the formula of $\frac{\partial f_C}{\partial w_j}$ and $\frac{\partial f_C}{\partial v_j}$ obtained in the proof of proposition 4.2.4, and using the apolar identities in lemma 2.2.1. Using (4.5) we obtain : $H^R = \begin{bmatrix} A & 2\Re(\tilde{B})^t & -2\Im(\tilde{B})^t \\ 2\Re(\tilde{B}) & 2\Re(\tilde{C}+\tilde{D}) & -2\Im(\tilde{C}+\tilde{D}) \\ -2\Im(\tilde{B}) & 2\Im(\tilde{D}-\tilde{C}) & 2\Re(\tilde{D}-\tilde{C}) \end{bmatrix}$. Finally, for the simplification by 2, as in the previous proof, we redefine the formula of H^R as it is given in proposition 4.2.5, where B, C, and D are respectively equal to two times $\tilde{B}$,$\tilde{C}$, and $\tilde{D}$. □

4.2.1.2 Retraction on $\mathcal{N}_r$

To complete this Riemannian Newton method, we need to define a retraction operator on $\mathcal{N}_r$. Let us assume that the Riemannian Newton equation is solved at a point $y = (w, v_1, \ldots, v_r, v'_1, \ldots v'_r) \in \mathcal{N}_r$, in local coordinates with respect to the basis Q as in lemma 4.2.1. It yields a solution vector $\hat{\eta} \in \mathbb{R}^{r+r(2n-1)}$. The tangent vector $\eta \in T_y\mathcal{N}_r$ of size $r + 2nr$ is given by $\eta = Q\,\hat{\eta} = (\nu, \eta_1, \ldots, \eta_r, \eta'_1, \ldots, \eta'_r)$. The new point $R_y(\eta) = (\tilde{w}, \tilde{v}_1, \ldots, \tilde{v}_r, \tilde{v}'_1, \ldots, \tilde{v}'_r) \in \mathcal{N}_r$ is defined using the product of the retractions on each component, that is the identity map on $\mathbb{R}^r$ and the projection map on the sphere $\mathbb{S}^{2n-1}$ [4] as follows :

— $\tilde{w} = R_w(\nu) = w + \nu$;

— $(\tilde{v}_j, \tilde{v}'_j) = R_{(v_j;v'_j)}(\eta_j, \eta'_j) = \frac{(v_j+\eta_j;v'_j+\eta'_j)}{||(v_j+\eta_j;v'_j+\eta'_j)||}$.

By lemma 3.3.1, this defines a retraction from $T_y\mathcal{N}_r$ to $\mathcal{N}_r$ since R_w (resp. $R_{(v_j;v'_j)}$) is a retraction on $\mathbb{R}^r$ (resp. $\mathbb{S}^{2n-1}$).

4.2.2 Riemannian Gauss–Newton for STA

In this subsection, we consider the STA problem over the product of r Veronese manifolds $\mathcal{V}^{n,d}$. By separating the real and imaginary parts of the coefficients of a polynomial, the non-zero points $(v^t\mathbf{x})^d$ with $v \in \mathbb{C}^n \setminus \{0\}$ form a smooth Riemannian variety in $\mathbb{C}[\mathbf{x}]_d$. We equip the $\mathbb{R}$-vector space $\mathbb{C}[\mathbf{x}]_d \sim \mathbb{R}^{2s_{n,d}}$ with the real inner product :

$$\forall \boldsymbol{p}, \boldsymbol{q} \in \mathbb{C}[\mathbf{x}]_d, \quad \langle \boldsymbol{p}, \boldsymbol{q} \rangle_d^{\mathbb{R}} = \Re(\langle \boldsymbol{p}, \boldsymbol{q} \rangle_d).$$

Let $\mathcal{V}_r := \mathcal{V}^{n,d} \times \cdots \times \mathcal{V}^{n,d}$. The map $\sigma_r : y = (y_1, \ldots, y_r) \in \mathcal{V}_r \mapsto y_1 + \cdots + y_r \in \mathbb{C}[\mathbf{x}]_d$ is a parameterization of the set Σ_r of symmetric tensors of symmetric rank at most r. We formulate the STA problem as a Riemannian least square problem over $\mathcal{V}_r$ as follows :

$$\text{(STA)}_{\mathcal{V}_r} \quad \min_{y\in\mathcal{V}_r} f(y),$$

where $f(y) = \frac{1}{2}||F(y)||_d^2$, with $F(y) = \sigma_r(y) - \boldsymbol{p}$ for $y \in \mathcal{V}_r$.

The differential map $DF = D\sigma_r$ at $y = (y_1, \ldots, y_r) \in \mathcal{V}_r$ with $y_i = (v_i^t \mathbf{x})^d$, $v_i \in \mathbb{C}^n$ is

$$\begin{array}{rcl} D\sigma_r(y) : T_{y_1}\mathcal{V}^{n,d} \times \cdots \times T_{y_r}\mathcal{V}^{n,d} & \to & T_{\sigma_r(y)}\mathbb{C}[\mathbf{x}]_d = \mathbb{C}[\mathbf{x}]_d \\ (\eta_1, \ldots, \eta_r) & \mapsto & \eta_1 + \cdots + \eta_r, \end{array}$$

where $T_{y_i}\mathcal{V}^{n,d} = \left\{ (u^t\mathbf{x})(v_i^t\mathbf{x})^{d-1} |\ u \in \mathbb{C}^n \right\}$ is of dimension $2n$ over $\mathbb{R}$.

The Gauss–Newton equation is given by :

$$(DF(y))^* \circ (DF(y))[\eta] = -(DF(y))^*[F(y)], \tag{4.6}$$

where $DF(y) : T_y\mathcal{V}_r \to \mathbb{C}[\mathbf{x}]_d$ is the differential map of F at y, and $(DF(y))^* \circ (DF(y)) : T_y\mathcal{V}_r \to T_y\mathcal{V}_r$ is the Gauss–Newton approximation of the Hessian of f at y.

We are going to describe explicitly the matrix of this map in a convenient basis of $T_y\mathcal{V}_r$. For a non-zero complex vector $v \in \mathbb{C}^n$, we define the inner product : $\forall u, u' \in \mathbb{C}^n$,

$$\langle u, u' \rangle_v = \Re(u^*u' + (d-1)(u^*v)(v^*u')\|v\|^{-2})$$

It is a positive definite inner product on $\mathbb{C}^n \sim \mathbb{R}^{2n}$ since $\langle u, u \rangle_v = \|u\|^2 + (d-1)|(v^*u)|^2\|v\|^{-2} \geq 0$ and it vanishes iff $u = 0$. Notice that $\langle v, v \rangle_v = d\|v\|^2$. The symmetric matrix associated to this inner product in the canonical basis of $\mathbb{R}^{2n}$ is

$$M_v := I_{2n} + (d-1)\|v\|^{-2}(v_R v_R^t + v_I v_I^t)$$

where $v_R = (\Re(v); \Im(v)), v_I = (-\Im(v); \Re(v))$ are the vectors of $\mathbb{R}^{2n}$ obtained by concatenating the real and imaginary part (resp. opposite imaginary and real part) of $v \in \mathbb{C}^n$.
Let $u_1 = \frac{v_R}{\|v\|}$ and $u_2 = \frac{v_I}{\|v\|}$. We notice that u_1 and u_2 can be completed to an orthonormal basis of $\mathbb{R}^{2n}$. Let U denotes the matrix of this basis i.e. $U = [u_1, \ldots, u_{2n}]$. Then $UU^t = I_{2n}$, so that an eigenvalue decomposition of the symmetric matrix M_v of $\langle \cdot, \cdot \rangle_v$ in the canonical basis of $\mathbb{R}^{2n}$ can be written as follows :

$$M_v = U\mathrm{diag}(1 + (d-1), 1 + (d-1), 1, \ldots, 1)U^t = U\mathrm{diag}(d, d, 1, \ldots, 1)U^t. \tag{4.7}$$

For shortness, we denote the strictly positive diagonal matrix $\mathrm{diag}(d, d, 1, \ldots, 1)$ by S.

Lemma 4.2.6. *Let $v \neq 0 \in \mathbb{C}^n \sim \mathbb{R}^{2n}$ and $p = (v^t\mathbf{x})^d \in \mathbb{C}[\mathbf{x}]_d$. Let $u_1, \ldots, u_{2n} \in \mathbb{C}^n$ be an orthonormal $\mathbb{R}$-basis for the inner product $\langle \cdot, \cdot \rangle_v$ with $u_1 = \frac{v}{\sqrt{d}\|v\|}$. Then*

$$\boldsymbol{q}_i = \sqrt{d}\|v\|^{-d+1}(u_i^t\mathbf{x})(v^t\mathbf{x})^{d-1}, i = 1, \ldots, 2n$$

is an orthonormal basis of $T_p\mathcal{V}^{n,d}$ for the inner product $\langle \cdot, \cdot \rangle_d^{\mathbb{R}}$.

Proof. Using the apolar identities in lemma 2.2.1, we have

$$\begin{array}{rcl} \langle \boldsymbol{q}_i, \boldsymbol{q}_j \rangle_d^{\mathbb{R}} & = & \sqrt{d}\|v\|^{-d+1}\Re(\langle (u_i^t\mathbf{x})(v^t\mathbf{x})^{d-1}, \boldsymbol{q}_j \rangle_d) \\ & = & \sqrt{d^{-1}}\|v\|^{-d+1}\Re((u_i^*\nabla_{\mathbf{x}}\boldsymbol{q}_j)(\overline{v})) \\ & = & \|v\|^{-2d+2}\Re((u_i^*u_j)(v^*v)^{d-1} + (d-1)(u_i^*v)(v^*u_j)(v^*v)^{d-2}) \\ & = & \Re((u_i^*u_j) + (d-1)(u_i^*v)(v^*u_j)\|v\|^{-2}) = \langle u_i, u_j \rangle_v. \end{array}$$

We deduce that $\langle \boldsymbol{q}_i, \boldsymbol{q}_j \rangle_d^{\mathbb{R}} = \delta_{i,j}$ and $(\boldsymbol{q}_i)_{i=1,\ldots,2n}$ is an orthonormal basis of $T_{(v^t\mathbf{x})^d}\mathcal{V}^{n,d}$ for the inner product $\langle \cdot, \cdot \rangle_d^{\mathbb{R}}$. □

We describe now how to compute an orthonormal basis for $\langle \cdot, \cdot \rangle_v$.

Lemma 4.2.7. *Let $M_v = USU^t$ be the eigenvalue decomposition of M_v as in (4.7). Let $\hat{u}_1 = \sqrt{S^{-1}}U^t\frac{v_R}{\sqrt{d}\|v\|}$ and let $Q \in \mathbb{R}^{2n\times 2n}$ be the orthogonal factor of a rank-revealing QR-decomposition of $I_{2n} - \hat{u}_1\hat{u}_1^t = QRP$ where R is upper triangular and P is a permutation matrix. Let*

$$u_{R,1} = \frac{v_R}{\sqrt{d}\|v\|}, u_{R,i} = U\sqrt{S^{-1}}Q_{[:,i-1]} \quad i = 2,\dots,2n.$$

Then the orthonormal $\mathbb{R}$-basis $u_1,\dots,u_{2n} \in \mathbb{C}^n$ for $\langle\cdot,\cdot\rangle_v$ is such that $u_i = (u_{R,i})_{[1:n]} + \mathbf{i}\,(u_{R,i})_{[n+1:2n]} \in \mathbb{C}^n$ for $i = 1,\dots,2n$.

Proof. As $M_v = USU^t$ with $UU^t = I_{2n}$ and $S \in \mathbb{R}^{2n\times 2n}$ a strictly positive diagonal matrix, we have $\sqrt{S^{-1}}U^tM_vU\sqrt{S^{-1}} = I_{2n}$. Thus the column vectors of $U\sqrt{S^{-1}}$ form an orthonormal basis of $\mathbb{R}^{2n}$ for $\langle\cdot,\cdot\rangle_v$.

The vector $\hat{u}_1 = \sqrt{S^{-1}}U^t\frac{v_R}{\sqrt{d}\|v\|}$ is representing the vector $\frac{v_R}{\sqrt{d}\|v\|}$ in this orthonormal basis.

The first $2n-1$ columns of the factor Q in a rank-revealing QR-decomposition of $I_{2n}-\hat{u}_1\hat{u}_1^t = QRP$ are orthonormal vectors $\hat{u}_2,\dots,\hat{u}_{2n}$ for $\langle\cdot,\cdot\rangle_v$, expressed in the basis $U\sqrt{S^{-1}}$. An orthonormal basis $u_{R,1}, u_{R,2},\dots,u_{R,2n} \in \mathbb{R}^{2n}$ for $\langle\cdot,\cdot\rangle_v$ is thus given by $u_{R,1} = \frac{v_R}{\sqrt{d}\|v\|}, u_{R,i} = U\sqrt{S^{-1}}Q_{[:,i-1]}, i = 2,\dots,2n$. The corresponding vectors $\in \mathbb{C}^n$ are $u_i = (u_{R,i})_{[1:n]} + \mathbf{i}\,(u_{R,i})_{[n+1:2n]} \in \mathbb{C}^n$ for $i = 1,\dots,2n$. □

Notice that when v is real and u, u' are real such that $\langle v,u\rangle = \langle v,u'\rangle = 0$, $\langle u,u'\rangle_v = \langle u,u'\rangle$ is the standard inner product of u, u'. Consequently in the real case, an orthonormal basis $(u_i)_{i=1,\dots,n} \subset \mathbb{R}^n$ can be obtained directly from $u_1 = \frac{v}{\|v\|}$ and a rank-revealing QR-decomposition of $I_n - u_1u_1^t$.

For $y = (y_1,\dots,y_r) \in \mathcal{V}_r$ with $y_i = (v_i^t\mathbf{x})^d \in \mathcal{V}^{n,d}$, $\forall 1 \le i \le r$, let $(\boldsymbol{q}_{i,j})_{j=1,\dots,2n}$ be the orthonormal basis associated to v_i defined in lemma 4.2.6 and let $Q_i = [\boldsymbol{q}_{i,1},\dots,\boldsymbol{q}_{i,2n}] \in \mathbb{R}^{2s_{n,d}\times 2n}$ be the coefficient matrix of the polynomials $(\boldsymbol{q}_{i,j})_{j=1,\dots,2n}$ in the canonical $\mathbb{R}$-basis of $\mathbb{C}[\mathbf{x}]_d$. The columns of the matrix

$$Q = \operatorname{diag}(Q_i)_{1\le i\le r},$$

represent an orthonormal basis of $T_y\mathcal{V}_r$ for the inner product induced by $\langle\cdot,\cdot\rangle_d^{\mathbb{R}}$ on each component.

Therefore, the Jacobian matrix J of σ_r at y, which is the matrix associated to $D\sigma_r(y) = DF(y)$, with respect to the orthonormal basis Q on $T_y\mathcal{V}_r$ and the standard real basis on $\mathbb{C}[\mathbf{x}]_d$ is given by :

$$J = [Q_1,\dots,Q_r] \in \mathbb{R}^{2s_{n,d}\times 2nr}.$$

Proposition 4.2.8. *The Gauss–Newton equation (4.6) in the orthonormal basis Q of $T_y\mathcal{V}_r$ is of the form*

$$H\,\tilde{\eta} = -G,$$

where $\tilde{\eta}^t = (\tilde{\eta}_1^t,\dots,\tilde{\eta}_r^t) \in \mathbb{R}^{2nr}$ is the unknown coordinate vector of an element of the tangent space $T_y\mathcal{V}_r$ in the basis Q and

- *$G = [G_k]_{k=1,\dots,2nr}$ with for $1 \le i \le r,\ 1 \le j \le 2n$,*
  $G_{2n(i-1)+j} = \sqrt{d^{-1}}\|v_i\|^{-d+1}\Big(d\sum_{k=1}^r \Re\big((u_{i,j}^*v_k)(v_i^*v_k)^{d-1}\big) - \Re\big(u_{i,j}^*\nabla_{\mathbf{x}}\boldsymbol{p}(\overline{v_i})\big)\Big)$,
- *$H = [H_{k,k'}]_{1\le k,k'\le 2nr}$ with for $1 \le i,i' \le r,\ 1 \le j,j' \le 2n$,*
  $H_{2n(i-1)+j,2n(i'-1)+j'} = \|v_i\|^{-d+1}\|v_{i'}\|^{-d+1}\Big(\Re\big((u_{i,j}^*u_{i',j'})(v_i^*v_{i'})^{d-1}\big) + (d-1)\Re\Big((u_{i,j}^*v_{i'})(v_i^*u_{i',j'})(v_i^*v_{i'})^{d-2}\Big)\Big)$.

Proof. As the matrix of $D\sigma_r(y) = DF(y)$ in the orthonormal basis Q on $T_y\mathcal{V}_r$ and the standard real basis on $\mathbb{C}[\mathbf{x}]_d$ is J, we have that the Gauss–Newton equation (4.6) is $H\tilde{\eta} = -G$ with
— $G = J^t vec(\sigma_r(y) - \boldsymbol{p}) = (\langle \boldsymbol{q}_{i,j}, \sigma_r(y) - \boldsymbol{p}\rangle_d^{\mathbb{R}})$,
— $H = J^t J = [Q_1, \ldots, Q_r]^t[Q_1, \ldots, Q_r] = (\langle \boldsymbol{q}_{i,j}, \boldsymbol{q}_{i',j'}\rangle_d^{\mathbb{R}})$.
By the apolar identities in lemma 2.2.1, we have

$$\begin{aligned}\langle \boldsymbol{q}_{i,j}, \sigma_r(y) - \boldsymbol{p}\rangle_d^{\mathbb{R}} &= \sqrt{d^{-1}}\|v_i\|^{-d+1}\Big(\sum_{k=1}^{r} \Re(u_{i,j}^* \nabla_{\mathbf{x}}(v_k^t\mathbf{x})^d(\overline{v_i}) - u_{i,j}^*\nabla_{\mathbf{x}}\boldsymbol{p}(\overline{v_i}))\Big)\\ &= \sqrt{d^{-1}}\|v_i\|^{-d+1}\Big(\sum_{k=1}^{r} \Re\big(d(u_{i,j}^*v_k)(v_i^*v_k)^{d-1} - u_{i,j}^*\nabla_{\mathbf{x}}\boldsymbol{p}(\overline{v_i})\big)\Big).\end{aligned}$$

Similarly,

$$\begin{aligned}&\langle \boldsymbol{q}_{i,j}, \boldsymbol{q}_{i',j'}\rangle_d^{\mathbb{R}}\\ &= \|v_i\|^{-d+1}\|v_{i'}\|^{-d+1}\Re(u_{i,j}^*\nabla_{\mathbf{x}}((u_{i',j'}^t\mathbf{x})(v_{i'}^t\mathbf{x})^{d-1})(\overline{v_i}))\\ &= \|v_i\|^{-d+1}\|v_{i'}\|^{-d+1}\Re((u_{i,j}^*u_{i',j'})(v_i^*v_{i'})^{d-1} + (d-1)(u_{i,j}^*v_{i'})(v_i^*u_{i',j'})(v_i^*v_{i'})^{d-2}),\end{aligned}$$

which ends the proof of the proposition. □

The Gauss–Newton equation

$$H\,\tilde{\eta} = -G,$$

solved in local coordinate with respect to the basis Q, yields a vector $\tilde{\eta} = (\tilde{\eta}_1; \ldots; \tilde{\eta}_r) \in \mathbb{R}^{2nr}$. The components of the tangent vector $\eta = (\eta_1, \ldots, \eta_r) \in T_y\mathcal{V}_r \in \mathbb{C}[\mathbf{x}]_d$ are then

$$\eta_i = \sqrt{d}||v_i||^{-d+1}(v_i^t\mathbf{x})^{d-1}\sum_{k=1}^{2n}\tilde{\eta}_{i,k}(u_{i,k}^t\mathbf{x}), \quad i = 1, \ldots, r.$$

4.2.2.1 Retraction on the Veronese manifold

We define the retraction of a tangent vector $\eta \in T_y\mathcal{V}_r$ to a new point $\tilde{y}$ on the manifold $\mathcal{V}_r$ as follows :

$$\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_r) = (R_{y_1}(\eta_1), \ldots, R_{y_r}(\eta_r)),$$

where $R_{y_i} : T_{y_i}\mathcal{V}^{n,d} \to \mathcal{V}^{n,d}$ is a retraction operator on the Veronese manifold for $i \in \{1, \ldots, r\}$ that we describe hereafter (see lemma 3.3.1).

We will use the following matrix construction to define the retraction on $\mathcal{V}^{n,d}$.

Definition 4.2.1. The Hankel matrix of degree $(k, d-k)$ associated to a polynomial $\boldsymbol{p}$ in $\mathbb{C}[\mathbf{x}]_d$ is given by :

$$H_{\boldsymbol{p}}^{k,d-k} = (\langle \boldsymbol{p}, \mathbf{x}^{\alpha+\beta}\rangle_d)_{|\alpha|=k,|\beta|=d-k}.$$

This matrix is also known as the *Catalecticant matrix* of the symmetric tensor $\boldsymbol{p}$ in degree $(k, d-k)$ or the *flattening* of $\boldsymbol{p}$ in degree $(k, d-k)$. In this definition, we implicitly assume that we have chosen a monomial ordering (for instance the lexicographic ordering on the monomials indexing the rows and columns of $H_{\boldsymbol{p}}^{k,d-k}$) to build the Hankel matrix. The properties of Hankel matrices that we will use are independent of this ordering. Such a matrix is called a Hankel matrix

since, as in the classical case, the entries of the matrix depend on the sum of the exponents of the monomials indexing the corresponding rows and columns.

When $k = 1$, using the apolar relations $\langle \boldsymbol{p}, x_i \mathbf{x}^\beta \rangle_d = \frac{1}{d} \langle \partial_{x_i} \boldsymbol{p}, \mathbf{x}^\beta \rangle_{d-1}$, we see that $H_{\boldsymbol{p}}^{1,d-1}$ is nothing else than the transposed of the coefficient matrix of the gradient $\frac{1}{d} \nabla_{\mathbf{x}} \boldsymbol{p}$ in the basis $\left(\mathbf{x}^\beta \binom{d-1}{\beta}^{-1} \right)_{|\beta|=d-1}$. When $\boldsymbol{p} = (v^t \mathbf{x})^d \in \mathcal{V}^{n,d}$, $H_{\boldsymbol{p}}^{1,d-1}$ can thus be written as the rank-1 matrix $v \otimes (v^t \mathbf{x})^{d-1}$.

Our construction of a retraction on $\mathcal{V}^{n,d}$ is described in the following definition.

Definition 4.2.2. For $v \in \mathbb{C}^n \setminus \{0\}$, let $\pi_v : \mathbb{C}[\mathbf{x}]_d \to \mathcal{V}^{n,d}$ be the map such that $\forall \boldsymbol{q} \in \mathbb{C}[\mathbf{x}]_d$,

$$\pi_v(\boldsymbol{q}) = \frac{\langle \psi(v), \boldsymbol{q} \rangle_d}{\|\psi(v)\|_d^2} \psi(v), \tag{4.8}$$

where $\psi : v \in \mathbb{C}^n \mapsto (v^t \mathbf{x})^d \in \mathcal{V}^{n,d}$ is the parametrization of the Veronese variety. For $\boldsymbol{p} \in \mathbb{C}[\mathbf{x}]_d$, let $\theta(\boldsymbol{p}) \in \mathbb{C}^n$ be the first left singular vector of $H_{\boldsymbol{p}}^{1,d-1}$. For $\boldsymbol{p} \in \mathcal{V}^{n,d}$, let

$$\begin{array}{rcl} R_{\boldsymbol{p}} : T_{\boldsymbol{p}} \mathcal{V}^{n,d} & \to & \mathcal{V}^{n,d} \\ \boldsymbol{q} & \mapsto & \pi_{\theta(\boldsymbol{p}+\boldsymbol{q})}(\boldsymbol{p}+\boldsymbol{q}). \end{array}$$

The retraction that we are going to describe on the Veronese manifold is closely related to the one on the Segre manifold used in [33]. In fact, since the Segre manifold coincides with the manifold of tensors of multilinear rank $(1, \ldots, 1)$, the retraction in [33] is deduced from the truncated multilinear rank $(1, \ldots, 1)$ HOSVD of a real multilinear tensor, i.e. from the truncated rank one SVD of the matricization in the different modes [122]. For a symmetric tensor, the matricization with respect to any mode gives the same Catalecticant matrix in degree (1, d-1). Hereafter, we show, by different techniques, that a single truncated SVD of the Catalecticant matrix in degree $(1, d-1)$ gives a retraction on the Veronese manifold.

By the apolar identities, we check that $R_{\boldsymbol{p}}(\boldsymbol{q}) = (\boldsymbol{p}(\bar{u}) + \boldsymbol{q}(\bar{u}))\,(u^t \mathbf{x})^d$ where $u = \theta(\boldsymbol{p}+\boldsymbol{q})$. We also verify that $\pi_{\lambda u} = \pi_u$ for any $\lambda \in \mathbb{C} \setminus \{0\}$ and any $u \in \mathbb{C}^n \setminus \{0\}$.

By the relation (4.8), for any $v \in \mathbb{C}^n \setminus \{0\}$, $\pi_v(\boldsymbol{q})$ is the vector on the line spanned by $\psi(v)$, which is the closest to $\boldsymbol{q}$ for the apolar norm. In particular, we have $\pi_v(\psi(v)) = \psi(v)$.

We verify now that $R_{\boldsymbol{p}}$ is a retraction on $\mathcal{V}^{n,d}$.

Lemma 4.2.9. *Let $\boldsymbol{p} \in \mathcal{V}^{n,d}$. Then, $\boldsymbol{p}$ is a fixed point by π_u where u is the first left singular vector of $H_{\boldsymbol{p}}^{1,d-1}$.*

Proof. If $\boldsymbol{p} = (v^t \mathbf{x})^d = \psi(v) \in \mathcal{V}^{n,d}$ with $v \in \mathbb{C}^n \setminus \{0\}$, then the first left singular vector u of $H_{\boldsymbol{p}}^{1,d-1}$ is up to a scalar equal to v. Thus we have $\pi_u(\boldsymbol{p}) = \pi_v(\psi(v)) = \psi(v) = \boldsymbol{p}$. □

Proposition 4.2.10. *Let $\boldsymbol{p} \in \mathcal{V}^{n,d}$. There exists a neighborhood $\mathcal{U}_{\boldsymbol{p}} \subset \mathbb{C}[\mathbf{x}]_d$ of $\boldsymbol{p}$ such that the map $\rho : \boldsymbol{q} \in \mathcal{U}_{\boldsymbol{p}} \mapsto \pi_{\theta(\boldsymbol{q})}(\boldsymbol{q})$ is well-defined and C^∞ smooth.*

Proof. Let $\boldsymbol{p} \in \mathcal{V}^{n,d}$ and $\theta : \boldsymbol{q} \in \mathbb{C}[\mathbf{x}]_d \to q \in \mathbb{C}^n$ where q is the first left singular vector of the SVD decomposition of $H_{\boldsymbol{q}}^{1,d-1}$. Let $\gamma : \mathbb{C}[\mathbf{x}]_d \to \mathcal{V}^{n,d} = \psi \circ \theta$ be the composition map by the parametrization map ψ of $\mathcal{V}^{n,d}$.

By construction, we have $\rho : \boldsymbol{q} \mapsto \langle \boldsymbol{q}, \gamma(\boldsymbol{q}) \rangle_d \, \gamma(\boldsymbol{q})$. Let $\mathcal{O}$ denotes the open set of homogeneous polynomials $\boldsymbol{q} \in \mathbb{C}[\mathbf{x}]_d$ such that the Hankel matrix $H_{\boldsymbol{q}}^{1,d-1}$ has a nonzero gap between the

first and the second singular values. It follows from [45] that the map θ is well-defined and smooth on $\mathcal{O}$. As $\boldsymbol{p}$ is in $\mathcal{V}^{n,d}$ and $H^{1,d-1}_{\boldsymbol{p}}$ is of rank 1, $\boldsymbol{p} \in \mathcal{O}$. Let $\mathcal{U}_{\boldsymbol{p}}$ be a neighborhood of $\boldsymbol{p}$ in $\mathbb{C}[\mathbf{x}]_d$ such that $\psi_{|\mathcal{U}_{\boldsymbol{p}}}$ is well-defined and smooth. As the apolar product $\langle \cdot, \cdot \rangle_d$ and the multiplication are well-defined and smooth on $\mathbb{C}[\mathbf{x}]_d \times \mathbb{C}[\mathbf{x}]_d$, ρ is well-defined and smooth on $\mathcal{U}_{\boldsymbol{p}}$, which ends the proof. □

As $\psi : v \in \mathbb{C}^n \mapsto (v^t\mathbf{x})^d \in \mathcal{V}^{n,d}$ is a parametrization of the Veronese variety $\mathcal{V}^{n,d}$, the tangent space of $\mathcal{V}^{n,d}$ at a point $\psi(v)$ is spanned by the first order vectors $D\psi(v)\, q$ of the Taylor expansion of $\psi(v + t\, q) = \psi(v) + t\, D\psi(v)\, q + O(t^2)$ for $q \in \mathbb{C}^n$. We are going to use this observation to prove the rigidity property of R_p.

Proposition 4.2.11. *For $\boldsymbol{p} \in \mathcal{V}^{n,d}, \boldsymbol{q} \in T_{\boldsymbol{p}}(\mathcal{V}^{n,d})$,*

$$\boldsymbol{p} + t\, \boldsymbol{q} - R_{\boldsymbol{p}}(t\, \boldsymbol{q}) = O(t^2).$$

Proof. As $\boldsymbol{p} \in \mathcal{V}^{n,d}, \boldsymbol{q} \in T_{\boldsymbol{p}}\mathcal{V}^{n,d}$, there exist $v, q \in \mathbb{C}^n$ such that $\boldsymbol{p} = \psi(v), \boldsymbol{q} = D\psi(v)\, q$. In particular, we have $\boldsymbol{p} + t\, \boldsymbol{q} - \psi(v + t\, q) = O(t^2)$. This implies that $H^{1,d-1}_{\boldsymbol{p}+t\, \boldsymbol{q}} - H^{1,d-1}_{\psi(v+t\, q)} = O(t^2)$. By differentiability of simple non-zero singular values and their singular vectors [202], we have $u_t - v_t = O(t^2)$ where $u_t = \theta(\boldsymbol{p} + t\, \boldsymbol{q})$ and $v_t = \theta(\psi(v + t\, q))$ are respectively the first left singular vectors of $H^{1,d-1}_{\boldsymbol{p}+t\, \boldsymbol{q}}$ and $H^{1,d-1}_{\psi(v+t\, q)}$.

Since $H^{1,d-1}_{\psi(v+t\, q)}$ is a matrix of rank 1 and its image is spanned by $v + t\, q$, v_t is a non-zero scalar multiple of $v + t\, q$ and we have $\pi_{v_t} = \pi_{v+t\, q}$. By continuity of the projection on a line, we have

$$\pi_{u_t}(\boldsymbol{p} + t\, \boldsymbol{q}) = \pi_{v_t}(\boldsymbol{p} + t\, \boldsymbol{q}) + O(t^2) = \pi_{v+t\, q}(\boldsymbol{p} + t\, \boldsymbol{q}) + O(t^2).$$

Since $\psi(v + t\, q) = \psi(v) + t\, D\psi(v)q + O(t^2) = \boldsymbol{p} + t\, \boldsymbol{q} + O(t^2)$, we have

$$\pi_{v+t\, q}(\boldsymbol{p} + t\, \boldsymbol{q}) = \pi_{v+t\, q}(\psi(v + t\, q)) + O(t^2) = \psi(v + t\, q) + O(t^2).$$

We deduce that

$$\begin{aligned} \boldsymbol{p} + t\, \boldsymbol{q} - R_{\boldsymbol{p}}(t\, \boldsymbol{q}) &= \boldsymbol{p} + t\, \boldsymbol{q} - \pi_{u_t}(\boldsymbol{p} + t\, \boldsymbol{q}) \\ &= \boldsymbol{p} + t\, \boldsymbol{q} - \psi(v + t\, q) + (\psi(v + t\, q) - \pi_{v_t}(\boldsymbol{p} + t\, \boldsymbol{q})) \\ &\quad + (\pi_{v_t}(\boldsymbol{p} + t\, \boldsymbol{q}) - \pi_{u_t}(\boldsymbol{p} + t\, \boldsymbol{q})) \\ &= \psi(v) + t\, D\psi(v)q - \psi(v + t\, q) + O(t^2) = O(t^2), \end{aligned}$$

which proves the proposition. □

Proposition 4.2.12. *Let $\boldsymbol{p} \in \mathcal{V}^{n,d}$. The map $R_{\boldsymbol{p}} : T_{\boldsymbol{p}}\mathcal{V}^{n,d} \to \mathcal{V}^{n,d},\ \boldsymbol{q} \mapsto R_{\boldsymbol{p}}(\boldsymbol{q}) = \pi_{\theta(\boldsymbol{p}+\boldsymbol{q})}(\boldsymbol{p} + \boldsymbol{q})$ is a retraction operator on the Veronese manifold $\mathcal{V}^{n,d}$.*

Proof. We have to prove that $R_{\boldsymbol{p}}$ verifies the three properties in definition 3.3.1.

1. $R_{\boldsymbol{p}}(0_{\boldsymbol{p}}) = \pi_{\theta(\boldsymbol{p})}(\boldsymbol{p} + 0_{\boldsymbol{p}}) = \pi_{\theta(\boldsymbol{p})}(\boldsymbol{p}) = \boldsymbol{p}$, by using lemma 4.2.9.
2. Let $S_{\boldsymbol{p}} : T_{\boldsymbol{p}}\mathcal{V}^{n,d} \to \mathbb{C}[\mathbf{x}]_d, \boldsymbol{q} \mapsto \boldsymbol{p} + \boldsymbol{q}$. The map $S_{\boldsymbol{p}}$ is well-defined and smooth on $T_{\boldsymbol{p}}\mathcal{V}^{n,d}$. By proposition 4.2.10, π is well-defined and smooth in a neighborhood $\mathcal{U}_{\boldsymbol{p}}$ of $\boldsymbol{p} \in \mathcal{V}^{n,d}$. Thus $R_{\boldsymbol{p}} = \rho \circ S_{\boldsymbol{p}}$ is well-defined and smooth in a neighborhood $\mathcal{U}'_{\boldsymbol{p}} \subset T\mathcal{V}^{n,d}$ of $0_{\boldsymbol{p}}$.

3. By proposition 4.2.11,
$$(\boldsymbol{p} + t\boldsymbol{q}) - R_{\boldsymbol{p}}(t\,\boldsymbol{q}) = O(t^2),$$
which implies that $\frac{d}{dt}R_{\boldsymbol{p}}(t\,\boldsymbol{q})\mid_{t=0} = \boldsymbol{q}$, or equivalently $DR_{\boldsymbol{p}}(0_{\boldsymbol{p}})\boldsymbol{q} = \boldsymbol{q}$. Therefore we have $DR_{\boldsymbol{p}}(0_{\boldsymbol{p}}) = id_{T_{\boldsymbol{p}}\mathcal{V}^{n,d}}$.

□

4.2.3 Adding a trust-region scheme

As discussed in 3.4.3, Riemannian Newton-type algorithm is usually combined with Riemannian trust region method to ensure a sufficient decrease in the cost function, and to enhance the algorithm, with the desirable properties of convergence to a local minimum, with a local superlinear rate of convergence. In this section, we add a Riemannian trust region scheme to the Newton (resp. Gauss–Newton) method described respectively in 4.2.1 and 4.2.2. Moreover, We prove in proposition 4.2.13 that under regularity assumptions, a local convergence for the Riemannian–Newton algorithm with trust region scheme can be obtained.

Let $\mathcal{M}$ denote the Riemannian manifold $\mathcal{N}_r$ in subsection 4.2.1 (resp. $\mathcal{V}_r$ in subsection 4.2.2), and let $y_k \in \mathcal{M}$. The subproblem to solve is

$$\min_{\eta \in B_{\Delta_k}} m_{y_k}(\eta), \tag{4.9}$$

where $m_{y_k}(\eta) := f(y_k) + G_k^t\eta + \frac{1}{2}\eta^t H_k \eta$, G_k is the gradient of f at y_k and H_k is respectively the Hessian of f at y_k for the Riemannian Newton method and the Gauss–Newton approximation of f at y_k for the Riemannian Gauss–Newton method, and $B_{\Delta_k} := \{\eta \in T_{y_k}\mathcal{M} \mid ||\eta|| \leq \Delta_k\}$.

By solving (4.9) using the dogleg method (3.4.3), we obtain a solution $\eta_k \in T_{y_k}\mathcal{M}$. Accepting or rejecting the candidate new point $y_{k+1} = R_{y_k}(\eta_k)$ is based on the quotient $\rho_k = \frac{f(y_k)-f(y_{k+1})}{m_{y_k}(0)-m_{y_k}(\eta_k)}$.
If ρ_k exceeds 0.2 then the current point y_k is updated, otherwise the current point y_k remains unchanged.
The radius of the trust region Δ_k is also updated based on ρ_k. We choose to update the trust region as in [33] with a few changes.

Let $\Delta_{y_0} := 10^{-1}\sqrt{\frac{d}{r}\sum_{i=1}^{r}||w_i^0||^2}$ in the Riemannian Newton iteration (resp. $\Delta_{y_0} := 10^{-1}\sqrt{\frac{d}{r}\sum_{i=1}^{r}||v_i^0||^{2d}}$ in the Riemannian Gauss–Newton iteration), $\Delta_{\max} := \frac{1}{2}||\boldsymbol{p}||_d$. We take the initial radius as $\Delta_0 = \min\{\Delta_{y_0}, \Delta_{\max}\}$, if $\rho_k > 0.6$ then the trust region is enlarged as follows : $\Delta_{k+1} = \min\{2||\eta_k||, \Delta_{\max}\}$. Otherwise the trust region is shrinked by taking $\Delta_{k+1} = \min\{(\frac{1}{3} + \frac{2}{3}(1 + e^{-14(\rho_k - \frac{1}{3})})^{-1})\Delta_k, \Delta_{\max}\}$.

The algorithm of the Riemannian Newton (resp. Gauss–Newton) method with trust region scheme for the STA problem is denoted by RNE-N-TR (resp. RGN-V-TR) and is given in pseudocode by Algorithm 4.1.

The Algorithm 4.1 is stopped when $\Delta_k \leq \Delta_{\min}$ (by default $\Delta_{\min} = 10^{-3}$), or when the maximum number of iterations exceeds $N_{\max}$.

Remark 4.2.1 – In order to handle ill-conditioned Hessian (resp. Gauss–Newton Hessian approximation) matrices in Algorithm 4.1, we use the Moore-Penrose pseudoinverse [22, 120, 201]. This can appear in cases where some vectors v_i of the rank-r approximation span close lines, which

Input : The homogeneous polynomial $\boldsymbol{p} \in \mathbb{C}[\mathbf{x}]_d$ associated to the symmetric tensor to approximate, $r < r_g$.
Choose initial point $y_0 \in \mathcal{N}_r$ (resp. $y_0 \in \mathcal{V}_r$).
while the method has not converged **do**

1. Compute the gradient vector and the Hessian matrix (resp. Gauss–Newton Hessian approximation);
2. Solve the subproblem (4.9) for the search direction $\eta_k \in B_{\Delta_k}$ by using the dogleg method;
3. Compute the candidate next new point $y_{k+1} = R_{y_k}(\eta_k)$;
4. Compute the quotient ρ_k;
5. Accept or reject y_{k+1} based on the quotient ρ_k;
6. Update the trust region radius Δ_k.

Output : $y_* \in \mathcal{N}_r$ (resp. $y_* \in \mathcal{V}_r$).

Algorithm 4.1 – Riemannian Newton (resp. Gauss–Newton) algorithm with trust region sheme for the STA problem "RNE-N-TR"(resp. "RGN-V-TR")

yields a singularity problem in the iteration. In particular, this is the case when the symmetric border rank of the symmetric tensor is not equal to its symmetric rank [33, 54], [129, section 2.4]. For example, the tensor $\boldsymbol{p} = (v_0^t\mathbf{x})(v_1^t\mathbf{x})^{d-1} + \epsilon\, T$, with $v_0, v_1 \in \mathbb{R}^n$, $T \in \mathbb{R}[\mathbf{x}]_d$ and ϵ very small, is close to the tensor $(v_0^t\mathbf{x})(v_1^t\mathbf{x})^{d-1} = \lim_{\delta\to 0} \frac{1}{d\delta}(((v_1 + \delta v_0)^t\mathbf{x})^d - (v_1^t\mathbf{x})^d)$ of border rank 2 and symmetric rank d. It can be very well approximated by a tensor of rank 2, with two vectors of almost the same direction.

Under some regularity assumption, it is possible to guarantee that RNE-N-TR algorithm converges to a local minimum of the distance function f.

Proposition 4.2.13. *Let $\boldsymbol{p} \in \mathbb{C}[\mathbf{x}]_d$, let $\boldsymbol{p}_0 \in \Sigma_r$ be the initial point of RNE-N-TR and let $B_0 = B(\boldsymbol{p}, ||\boldsymbol{p} - \boldsymbol{p}_0||_d)$ be the ball of center $\boldsymbol{p}$ and radius $||\boldsymbol{p} - \boldsymbol{p}_0||_d$ in $\mathbb{C}[\mathbf{x}]_d$. Assume that $B_0 \cap \overline{\Sigma}_r \subset \Sigma_r^{\mathrm{reg}}$ (i.e. all points of $\overline{\Sigma}_r$ in B_0 are non-defective), then RNE-N-TR converges to a local minimum $y \in \mathcal{N}_r$ of the distance function f to Σ_r.*

Proof. Let $\Sigma_r^0 := B_0 \cap \overline{\Sigma}_r = B_0 \cap \Sigma_r^{\mathrm{reg}}$ be the set of non-defective tensors of rank r in B_0. As $\varphi_r : (W, V_R, V_I) \in \mathcal{N}_r \mapsto \sum_{i=1}^r w_i((v_{R,i} + \mathbf{i}\, v_{I,i})^t\mathbf{x})^d \subset \overline{\Sigma}_r \subset \mathbb{C}[\mathbf{x}]_d$ is locally injective at a non-defective tensor, it defines a local diffeomorphism between $\mathcal{N}_{r,0} = \varphi_r^{-1}(\Sigma_r^0)$ and Σ_r^0. As B_0 is compact, $\mathcal{N}_{r,0} = \varphi_r^{-1}(\Sigma_r^0)$ is a compact Riemannian manifold. By construction, the distance between $\boldsymbol{p}$ and the iterates $\boldsymbol{p}_i$ is decreasing in RNE-N-TR, so that their decomposition is in $\mathcal{N}_{r,0} = \varphi_r^{-1}(\Sigma_r^{\mathrm{reg}} \cap B_0)$. As $\mathcal{N}_{r,0}$ is a compact Riemannian manifold and f is smooth on $\mathcal{N}_{r,0}$ (as a polynomial function), [2, Corollary 7.4.6] implies that the iterates of RNE-N-TR of Riemannian Newton method with a trust region sheme on $\mathcal{N}_{r,0}$ converge to a local minimum of the distance function f. □

The regularity assumption $B_0 \cap \overline{\Sigma}_r \subset \Sigma_r^{\mathrm{reg}}$ implies that the ball centered at $\boldsymbol{p}$ and containing the initial point of the iteration does not contain a defective tensor. In this case, the iterates, which distance to $\boldsymbol{p}$ decreases, remain in the ball and the limit decomposition is a non-defective low rank tensor. This assumption, satisfied when $\boldsymbol{p}$ is far enough from the singular locus of $\overline{\Sigma}_r$, is a sufficient condition to ensure the regularity of the iteration points and their limit.

4.3 Numerical experiments

In this section, we present four numerical experiments using the RNE-N-TR and RGN-V-TR algorithms. These algorithms are implemented in the package `TensorDec.jl` *. We use a Julia implementation for the method SPM tested in subsection 4.3.4. The solvers from Tensorlab v3 [217] are run in MATLAB 7.10. The experimentation was done on a Dell Windows desktop with 8 GB memory and Intel Core i5-5300U, 2.3 GHz CPU.

4.3.1 Choice of the initial point

The choice of the initial point is a crucial step in iterative methods. We use the direct algorithm of [92], based on the computation of generalized eigenvectors and generalized eigenvalues of pencils of Hankel matrices (see also [150]), to compute an initial rank-r approximation. This algorithm, denoted SMD, works only with $r < r_g$ such that $\iota \leq \lfloor \frac{d-1}{2} \rfloor$ where ι denotes the interpolation degree of the points in the rank-r decomposition [76, Chapter 4]. This implies that $r < \binom{n+d'-1}{d'}$ where $d' = \lfloor \frac{d-1}{2} \rfloor$. It first computes a SVD decomposition of the Hankel matrix of the tensor $\boldsymbol{t}$ in degree $\left(\lfloor \frac{d-1}{2} \rfloor, d - \lfloor \frac{d-1}{2} \rfloor\right)$, extracts the first r singular vectors, computes a simultaneous diagonalisation of the matrices of multiplication by the variables x_i by taking a random combination of them, computing its eigenvectors and deducing the points and weights in the approximate decomposition of $\boldsymbol{t}$. The rationale behind choosing the initial point with this method is when the symmetric tensor is already of symmetric rank r with $r < r_g$ and $\iota \leq \lfloor \frac{d-1}{2} \rfloor$, then this computation gives a good numerical approximation of the exact decomposition, so that the Riemannian Newton (resp. Gauss–Newton) algorithm needs few iterations to converge numerically. We will see in the following numerical experiments that this initial point is an efficient choice to get a good low rank approximation of a symmetric tensor.

4.3.2 Best rank-1 approximation and spectral norm

Let $\boldsymbol{p} \in \mathcal{S}_n^d(\mathbb{R})$, a best real rank-1 approximation of $\boldsymbol{p}$ is a minimizer of the optimization problem

$$\text{dist}_1(\boldsymbol{p}) := \min_{\boldsymbol{t} \in \mathcal{S}_n^d(\mathbb{R}), \text{rank}_s(\boldsymbol{t})=1} ||\boldsymbol{p} - \boldsymbol{t}||_d^2 = \min_{(w,v) \in \mathbb{R} \times \mathbb{S}^{n-1}} ||\boldsymbol{p} - w(v^t \mathbf{x})^d||_d, \tag{4.10}$$

where $\mathbb{S}^{n-1} = \{v \in \mathbb{R}^n \mid ||v|| = 1\}$ is the unit sphere. This problem is equivalent to $\min_{\boldsymbol{t} \in \mathcal{T}^d(\mathbb{R}^n), \text{rank}(\boldsymbol{t})=1} ||\boldsymbol{p} - \boldsymbol{t}||_F^2$ since at least one global minimizer is a symmetric rank-1 tensor [227].

The real spectral norm of $\boldsymbol{p} \in \mathcal{S}_n^d(\mathbb{R})$, denoted by $||\boldsymbol{p}||_{\sigma,\mathbb{R}}$ is by definition :

$$||\boldsymbol{p}||_{\sigma,\mathbb{R}}^2 := \max_{v \in \mathbb{S}^{n-1}} |\boldsymbol{p}(v)|. \tag{4.11}$$

The two problems (4.10) and (4.11) are related by the following equality :

$$\text{dist}_1(\boldsymbol{p})^2 = ||\boldsymbol{p}||_d^2 - ||\boldsymbol{p}||_{\sigma,\mathbb{R}}^2,$$

∗. It can be obtained from `https://gitlab.inria.fr/AlgebraicGeometricModeling/TensorDec.jl` and run in Julia version 1.1.1. See functions rne_n_tr and rgn_v_tr.

which we deduce by simple calculus and properties of the apolar norm (see also [64, 227]) :

$$\begin{aligned}
\mathrm{dist}_1(\boldsymbol{p})^2 &= \min_{(w,v)\in\mathbb{R}\times\mathbb{S}^{n-1}} ||\boldsymbol{p} - w(v^t\mathbf{x})^d||_d^2 \\
&= \min_{(w,v)\in\mathbb{R}\times\mathbb{S}^{n-1}} ||\boldsymbol{p}||_d^2 - 2\langle \boldsymbol{p}, w(v^t\mathbf{x})^d\rangle_d + ||w(v^t\mathbf{x})^d||_d^2 \\
&= \min_{(w,v)\in\mathbb{R}\times\mathbb{S}^{n-1}} ||\boldsymbol{p}||_d^2 - 2w\,\boldsymbol{p}(v) + w^2 \\
&= \min_{v\in\mathbb{S}^{n-1}} ||\boldsymbol{p}||_d^2 - |\boldsymbol{p}(v)|^2 = ||\boldsymbol{p}||_d^2 - \max_{v\in\mathbb{S}^{n-1}} |\boldsymbol{p}(v)|^2 = ||\boldsymbol{p}||_d^2 - ||\boldsymbol{p}||_{\sigma,\mathbb{R}}^2.
\end{aligned}$$

Therefore, if v is a global maximizer of (4.11) such that $w = \boldsymbol{p}(v)$, then $w\,v^{\otimes d}$ is a best rank-1 approximation of $\boldsymbol{p}$. Herein, a rank-1 approximation $w\,v^{\otimes d}$, such that $w = \boldsymbol{p}(v)$ and $||v|| = 1$, is better when $|w|$ is higher. Therefore, in the following experimentation, we report the weight w obtained by the different methods.

In [155] the authors present an algorithm called “SDP" based on semidefinite relaxations to find a best real rank-1 approximation of a real symmetric tensor by finding a global optimum of $\boldsymbol{p}$ on $\mathbb{S}^{n-1}$. We choose two examples from [155], on which we apply the RNE-N-TR with initial point chosen according to the SMD algorithm adapted for 1×1 matrices. The reason behind using RNE-N-TR instead of RGN-V-TR is to take advantage of the local quadratic rate of convergence that distinguishes the exact Riemannian Newton iteration in RNE-N-TR [2, Theorem 6.3.2]. We compare these methods with the method CCPD-NLS which is a non-linear least-square solver for the symmetric decomposition from Tensorlab v3 [217] in MATLAB 7.10. For CCPD-NLS we use two initialization strategies, in the first one we run 50 instances (i.e. 50 random initial points obeying Gaussian distributions), and we take the absolute value of the weight in average for this method. In the second one we use the same initialization as RNE-N-TR i.e. CCPD-NLS is initialized by SMD. We also compare with the algorithm CPD from Tensorlab v3 [217] for multilinear tensors initialized by the determinist method from Tensolab called GEVD.

We denote respectively by $|w_{\mathrm{sdp}}|$, $|w_{\mathrm{rne}}|$, $|w_{\mathrm{ccpd\_smd}}|$, and $|w_{\mathrm{cpd\_gevd}}|$ the weight in absolute value given respectively by SDP, RNE-N-TR, CCPD-NLS initialized by SMD, and CPD initialized by GEVD, and $|w_{\mathrm{ccpd}}|$ denotes the absolute value of the weight in average given by CCPD-NLS. Note that $|w_{\mathrm{sdp}}|$ is the spectral norm of $\boldsymbol{p}$, since SDP gives a best rank-1 approximation. We report the time spent by SDP from [155] (resp. RNE-N-TR including the computation time of the initial point) in seconds (s) and we denote it by t_{sdp} (resp. t_{rne}). We denote by N_{rne} the number of iterations in RNE-N-TR. We denote by d_0 the norm between $\boldsymbol{p}$ and the initial point of RNE-N-TR, and by d_* the norm between $\boldsymbol{p}$ and the solution obtained by RNE-N-TR. We denote by $t_{\mathrm{ccpd\_smd}}$ and $N_{\mathrm{ccpd\_smd}}$ the consumed time and number of iterations of CCPD-NLS initialized by SMD. We denote by $t_{\mathrm{cpd\_gevd}}$ the time spent by CPD initialized by GEVD. We denote by t_{ccpd} (resp. N_{ccpd}) the time in seconds (s) (resp. number of iterations) in average for CCPD-NLS.

Example 4.3.1 – [155, Example 3.5]. Consider the tensor $\boldsymbol{p}\in\mathcal{S}_n^3(\mathbb{R})$ with entries :

$$(\boldsymbol{p})_{i_1,i_2,i_3} = \frac{(-1)^{i_1}}{i_1} + \frac{(-1)^{i_2}}{i_2} + \frac{(-1)^{i_3}}{i_3},$$

corresponding to the polynomial $\boldsymbol{p} = \sum_{|\alpha|=3}(\sum_{i=1}^n \alpha_i \frac{(-1)^i}{i}) \binom{3}{\alpha} \mathbf{x}^\alpha$.

Example 4.3.2 – [155, Example 3.7]. Consider the tensor $\boldsymbol{p} \in \mathcal{S}_n^5(\mathbb{R})$ given as :

$$(\boldsymbol{p})_{i_1,\ldots,i_5} = (-1)^{i_1}\log(i_1) + \cdots + (-1)^{i_5}\log(i_5),$$

corresponding to the polynomial $\boldsymbol{p} = \sum_{|\alpha|=5}(\sum_{i=1}^{n}\alpha_i(-1)^i\log(i))\binom{5}{\alpha}\mathbf{x}^\alpha$.

	Example 4.3.1					Example 4.3.2				
n	10	20	30	40	50	5	10	15	20	25
$\lvert w_{\mathrm{rne}}\rvert$	**17.8**	**34.2**	**50.1**	**65.9**	**81.6**	**1.100e+2**	**8.833e+2**	**2.697e+3**	**6.237e+3**	**11.504e+3**
d_0	32.4	28.4	44	64.6	78.3	526.1	6.559e+3	26.318e+3	64.268e+3	132.213e+3
d_*	13.2	28.3	43.8	59.5	75.3	477.5	6.096e+3	24.643e+3	60.435e+3	121.892e+3
t_{rne}	0.038	0.304	1.5	3.3	12.1	0.058	0.282	3.8	18.3	34.8
N_{rne}	5	4	4	4	6	5	4	6	6	6
$\lvert w_{\mathrm{ccpd}}\rvert$	14.0	29.3	43.3	60.0	75.6	78.9	8.68e+2	2.354e+3	6.148e+3	10.587e+3
t_{ccpd}	0.173	0.109	0.105	0.122	0.143	0.093	0.187	1.2	5.5	16.7
N_{ccpd}	27	25	22	23	22	19	29	16	23	17
$\lvert w_{\mathrm{ccpd\_smd}}\rvert$	**17.8**	**34.2**	**50.1**	**65.9**	**81.6**	**1.100e+2**	**8.833e+2**	**2.697e+3**	**6.237e+3**	**11.504e+3**
$t_{\mathrm{ccpd\_smd}}$	0.194	0.158	0.118	0.123	0.192	0.228	0.261	1.2	4.8	14.1
$N_{\mathrm{ccpd\_smd}}$	33	32	24	23	27	40	31	19	13	21
$\lvert w_{\mathrm{cpd\_gevd}}\rvert$	**17.8**	**34.2**	**50.1**	**65.9**	**81.6**	**1.100e+2**	**8.833e+2**	**2.697e+3**	**6.237e+3**	**11.504e+3**
$t_{\mathrm{cpd\_gevd}}$	0.148	0.152	0.207	0.289	0.177	0.521	0.361	0.947	3.4	9.7
$\lvert w_{\mathrm{sdp}}\rvert$	**17.8**	**34.2**	**50.1**	**65.9**	**81.6**	**1.100e+2**	**8.833e+2**	**2.697e+3**	**6.237e+3**	
t_{sdp}	2.0	6.0	30.0	245.0	1965.0	1.0	22.0	78.0	1350.0	

Table 4.1 – Symmetric rank-1 approximation for Example 4.3.1 and Example 4.3.2 : RNE-N-TR (rne), CCPD-NLS (ccpd_smd) initialized by SMD (Section 4.3.1), CCPD-NLS (ccpd) initialized by 50 random initial points obeying Gaussian distributions and the reported results for this method are in average, CPD (cpd_gevd) is initialized by default by the Tensorlab method called GEVD. The method RNE-N-TR stops when the maximum number of iterations is reached (by default 500) or when the radius of the trust region Δ_k is less than $\Delta_{\min}$ (by default 10^{-3}). Tensorlab's methods stop when the stop criteria given by $\mathrm{Display} = 10$ are verified. The method SDP (sdp) is a global optimization to which we compare the aforementioned local optimization methods if they can find best symmetric rank-1 approximation.

The results in Table 4.1 show that the RNE-N-TR finds a global minimizer, starting from the initial point given by the SMD algorithm. The RNE-N-TR algorithm converges to this point in few iterations, and with very reduced time compared to the SDP algorithm especially when n grows, where the SDP method works in a greedy manner. On the other hand, $|w_{\mathrm{ccpd}}|$ is smaller than $|w_{\mathrm{sdp}}|$, implying that CCPD computes, in several cases, a local minimum, which is not a global minimum i.e. a best rank-1 approximation. In comparison for these cases, RGN-V-TR took more iterations ($\sim$20) than RNE-N-TR and consequently more time, while reaching the same optimimum.

The fact that RNE-N-TR finds the best rank-1 approximation in these examples comes from the good initial point provided by SMD algorithm. However, we have no guarantee that RNE-N-TR with this initial point will always converge to a best rank-1 approximation. This experimentation shows that RNE-N-TR combined with SMD algorithm for the initial point is an efficient method to get a good real rank-1 approximation of a real symmetric tensor.

4.3.3 Symmetric rank-r approximation

We consider two examples of a real and a complex valued sparse symmetric tensors, in order to compare the performance of RNE-N-TR and RGN-V-TR with state-of-the-art non-linear least-square solvers CCPD-NLS and SDF-NLS for symmetric decomposition from Tensolab v3 with

random initial point following a standard normal distribution. We note that `ccpd_smd` means that CCPD-NLS is initialized by SMD as for the Riemannian Newton and Riemannian Gauss–Newton algorithms. These solvers employ factor matrices as parameterization and use a Gauss–Newton method with dogleg trust region steps called "NLS-GNDL". We also compare with the algorithm CPD from Tensorlab v3 [217] for multilinear tensors initialized by the determinist method from Tensolab called GEVD, we mention that this algorithm does not impose symmetry. We fix 200 iterations as maximal number of iterations, and we run 50 instances for these methods and we report the minimal, median and maximal residual error denoted 'err', such that, $\mathrm{err} := ||\boldsymbol{p}-\boldsymbol{p}_*||_d$, where $\boldsymbol{p}$ is the symmetric tensor to approximate and $\boldsymbol{p}_*$ is the approximate symmetric tensor of rank-r. In the computation of the initial point by SMD algorithm in RNE-N-TR and RGN-V-TR, we compute eigenvectors of a random linear combination of multiplication operators. This computation is sensitive to the choice of the linear combination, when the operators are not commuting, which explains why we report also the minimal, median and maximal err for these two methods. The average of time t is in seconds, and the average number of iterations N is rounded to the closest integer.

Example 4.3.3 – Let $\boldsymbol{p} \in \mathcal{S}^3_{10}(\mathbb{R})$ such that :

$$(\boldsymbol{p})_{i_1,i_2,i_3} = \begin{cases} i_1^2+1 & \text{if } i_1=i_2=i_3, \\ 1 & \text{if } [i_1,i_2,i_3] \equiv [i,i,j] \text{ with } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

($[i_1,i_2,i_3] \equiv [j_1,j_2,j_3]$ iff there exists a permutation $\sigma \in S_3$ such that $[i_{\sigma(1)},i_{\sigma(2)},i_{\sigma(3)}] = [j_1,j_2,j_3]$). This sparse symmetric tensor corresponds to the polynomial $\boldsymbol{p} = \sum_{i=1}^{10} i^2 x_i^3 + (\sum_{i=1}^{10} x_i^2) \times (\sum_{i=1}^{10} x_i)$.

Example 4.3.4 – Let $\boldsymbol{p} \in \mathcal{S}^3_{10}(\mathbb{C})$ such that :

$$(\boldsymbol{p})_{i_1,i_2,i_3} = \begin{cases} e^{\sqrt{i_1}+i_1^2\sqrt{-1}} + \frac{i_1}{10}\sqrt{-1} & \text{if } i_1=i_2=i_3, \\ \frac{i}{10}\sqrt{-1} & \text{if } [i_1,i_2,i_3] \equiv [i,i,j] \text{ with } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

This sparse symmetric tensor corresponds to the polynomial $\boldsymbol{p} = \sum_{i=1}^{10} e^{\sqrt{i}+i^2\sqrt{-1}} x_i^3 + \sqrt{-1}(\sum_{i=1}^{10} \frac{i}{10} x_i^2) \times (\sum_{i=1}^{10} x_i)$.

The numerical results in Table 4.2 show that the number of iterations of RNE-N-TR and RGN-V-TR method is low compared to the other methods that impose symmetry in this test i.e. CCPD-NLS initialized by SMD, CCPD-NLS and SDF-NLS with random initialization. The iterations in RNE-N-TR and RGN-V-TR are more expensive. The numerical quality of approximation is better for RNE-N-TR and RGN-V-TR than the other methods that impose symmetry. It is of the same order as the other methods for $r = 3, 5$ but much better for $r = 10$. On the other hand, we notice that the approximation obtained by CPD algorithm look better than the ones obtained by RNE-N-TR and RGN-V-TR, even though we checked that the decompositions obtained by CPD are not symmetric.

Example 4.3.3						Example 4.3.4					
r	$\mathrm{err}_{\mathrm{rne}}$			t_{rne}	N_{rne}	r	$\mathrm{err}_{\mathrm{rne}}$			t_{rne}	N_{rne}
	min	med	max	avg	avg		min	med	max	avg	avg
3	70.6	96	134.3	0.03	2	3	22.4	28.8	30.9	0.04	2
5	33.3	54.2	91.8	0.08	3	5	14.1	17.4	24.6	0.07	3
10	0.884	0.884	94.1	0.465	6	10	0.164	0.168	0.369	0.113	2
r	$\mathrm{err}_{\mathrm{rgn}}$			t_{rgn}	N_{rgn}	r	$\mathrm{err}_{\mathrm{rgn}}$			t_{rgn}	N_{rgn}
	min	med	max	avg	avg		min	med	max	avg	avg
3	70.6	96	136.8	0.064	3	3	22.4	27.6	36.1	0.065	3
5	33.3	48.8	105.3	0.149	4	5	14.1	17.1	24.6	0.101	3
10	0.886	0.886	10.1	0.836	7	10	0.162	0.164	0.169	0.219	2
r	$\mathrm{err}_{\mathrm{ccpd}}$			t_{ccpd}	N_{ccpd}	r	$\mathrm{err}_{\mathrm{ccpd}}$			t_{ccpd}	N_{ccpd}
	min	med	max	avg	avg		min	med	max	avg	avg
3	71	102	137.1	0.067	14	3	22.9	26.8	35.2	0.084	14
5	34.2	54.7	121	0.116	26	5	14.9	17	26.6	0.104	18
10	7.8	7.8	9.7	0.5	90	10	4.8	4.8	11.2	0.506	60
r	$\mathrm{err}_{\mathrm{ccpd\_smd}}$			$t_{\mathrm{ccpd\_smd}}$	$N_{\mathrm{ccpd\_smd}}$	r	$\mathrm{err}_{\mathrm{ccpd\_smd}}$			$t_{\mathrm{ccpd\_smd}}$	$N_{\mathrm{ccpd\_smd}}$
	min	med	max	avg	avg		min	med	max	avg	avg
3	71	96.3	142.5	0.044	5	3	22.9	27	30.1	0.068	10
5	34.2	47.4	105.7	0.06	8	5	14.9	17.1	24.6	0.084	11
10	7.8	7.8	47.1	0.726	44	10	4.8	4.8	4.8	0.09	12
r	$\mathrm{err}_{\mathrm{cpd\_gevd}}$			$t_{\mathrm{cpd\_gevd}}$	$N_{\mathrm{cpd\_gevd}}$	r	$\mathrm{err}_{\mathrm{cpd\_gevd}}$			$t_{\mathrm{cpd\_gevd}}$	$N_{\mathrm{cpd\_gevd}}$
	min	med	max	avg	avg		min	med	max	avg	avg
3	70.9	70.9	70.9	0.069	1	3	22.6	22.6	22.6	0.065	1
5	33.8	33.8	33.8	0.071	1	5	14.2	14.2	14.2	0.071	1
10	2.3e-14	2.3e-14	2.3e-14	0.049	1	10	5.3e-15	5.3e-15	5.3e-15	0.049	1
r	$\mathrm{err}_{\mathrm{sdf}}$			t_{sdf}	N_{sdf}	r	$\mathrm{err}_{\mathrm{sdf}}$			t_{sdf}	N_{sdf}
	min	med	max	avg	avg		min	med	max	avg	avg
3	71	96.3	136	0.155	14	3	22.9	27.4	35.2	0.254	15
5	34.2	49.4	105.3	0.212	16	5	14.9	17.8	26.5	0.35	19
10	7.8	8.2	38.3	2.3	158	10	4.8	6.2	12.6	2.5	144

Table 4.2 – Computational results for Examples 4.3.3 and 4.3.4 : RNE-N-TR (rne), RGN-V-TR (rgn), and CCPD-NLS (ccpd_smd) initialized by 50 points given by SMD, CPD (cpd_gevd) does not impose symmetry (i.e. the three factor matrices obtained by this method are not the same) and it is initialized by default by the Tensorlab's method GEVD, CCPD-NLS (ccpd) and SDF-NLS (sdf) initialized by 50 random initial points obeying Gaussian distributions. The methods RNE-N-TR and RGN-V-TR stop when the maximum number of iterations is reached (fixed to 200) or when the radius of the trust region Δ_k is less than $\Delta_{\min}$ (by default 10^{-3}). Tensorlab's methods stop when the stop criteria given by $\mathrm{Display} = 10$ are verified.

4.3.4 Approximation of perturbations of low rank symmetric tensors

In this section, we consider perturbations of random low rank tensors. For a given rank r, we choose r random vectors v_i of size n, obeying Gaussian distributions and compute the symmetric tensor $\boldsymbol{t} = \sum_{i=1}^{r}(v_i^t \mathbf{x})^d$ of order d. We choose a random symmetric tensor $\boldsymbol{t}_{\mathrm{err}}$ of order d, with coefficients also obeying Gaussian distributions, normalize it so that its apolar norm is ϵ and add it to $\boldsymbol{t}$: $\tilde{\boldsymbol{t}} = \boldsymbol{t} + \epsilon \frac{\boldsymbol{t}_{\mathrm{err}}}{\|\boldsymbol{t}_{\mathrm{err}}\|_d}$. We apply the different approximation algorithms to $\tilde{\boldsymbol{t}}$ and compute the relative error factor ref $:= \frac{\|\boldsymbol{t}_* - \boldsymbol{t}\|_d}{\epsilon}$ between the approximation $\boldsymbol{t}_*$ of rank r computed by the algorithm and the rank-r tensor $\boldsymbol{t}$. We run this computation for 100 random instances and report the geometric average of the relative error. The average number of iterations N is rounded to the closest integer, and the average time t is in seconds.

As the initial tensor $\tilde{\boldsymbol{t}}$ is in a ball of radius ϵ centered at the tensor $\boldsymbol{t}$ of rank r, we expect $\boldsymbol{t}_*$ to be at distance to $\boldsymbol{t}$ smaller than ϵ and the relative error factor to be less than 1.

We compare the RNE-N-TR and RGN-V-TR methods with the initial point computed by SMD algorithm, with the recent Subspace Power Method (SPM) of [116] and the state-of-the-art implementation CPD-NLS of the package Tensorlab v3. Note that CPD-NLS is designed for the canonical polyadic decomposition [101]. Nevertheless, in practice it is often observed that applying a general tensor rank approximation method (like CPD-NLS) from a symmetric starting point will usually result in a symmetric approximation. Since CPD-NLS is an efficient tensor decomposition routine of Tensorlab v3, we choose to compare our methods with this algorithm in this numerical experiment, using symmetric initial points and verifying that the obtained tensor approximations are symmetric. As SPM works for even order tensors with real coefficients, the comparison in Table 4.3 is run for tensors in $\mathcal{S}^4_{10}(\mathbb{R})$. In Table 4.4, we compare CPD-NLS, RNE-N-TR, and RGN-V-TR for tensors in $\mathcal{S}^d_{10}(\mathbb{C})$ of order $d = 4$ and with complex coefficients. These tables also provide a numerical comparison with the low rank approximation methods tested in Example 5.4 of [154], since the setting is the same. We also run this tensor perturbation test on some complex examples in which the approximation rank is higher than the mode size of the tensor (see Table 4.5). We test this with the three methods RNE-N-TR, RGN-V-TR, and CPD-NLS. We run 20 instances, for each example of tensor and ϵ.

The computational time for the methods RNE-N-TR and RGN-V-TR includes the computation of the initial point by the SMD algorithm. We fix 200 iterations as maximal number of iterations for RNE-N-TR, RGN-V-TR and CPD-NLS. For SPM, the iterations are stopped when the distance between two consecutive iterates is less than 10^{-10} or when the maximal number of iterations ($N = 400$ in this experimentation) is reached.

In Tables 4.3, 4.4, the number of iterations of the RNE-N-TR and RGN-V-TR methods is significantly smaller than the number of iterations of the other methods. In SPM, the number of iterations to get an approximation of a single rank-1 term of the approximation is about 30, indicating a practical linear convergence as predicted by the theory [116, Theorem 5.10]. As the method CPD-NLS is based on a quasi-Newton iteration, its local convergence is sub-quadratic, which also explains the relatively high number of iterations. The low number of iterations in RNE-N-TR and RGN-V-TR can be explained by the choice of the initial point by SMD algorithm. This provides a good initialization such that a solution by RNE-N-TR and RGN-V-TR can be obtained in a few number of iterations.

The cost of an iteration appears to be higher in RNE-N-TR and RGN-V-TR than in the other methods. Nevertheless, the total time is of the same order. Note that the cost of an iteration seems higher in RGN-V-TR than RNE-N-TR. Despite the fact that the first algorithm computes the Gauss–Newton approximation of the Hessian matrix, whereas the second algorithm computes the exact Hessian matrix. This can be explained by the use of a parametrization in the first algorithm (i.e. the Cartesian product of Veronese manifolds), which involves a more expensive retraction using SVD decomposition on larger matrices.

These experimentation also show a good numerical behavior for the Riemannian methods. In particular, the numerical quality of the low rank approximation is good for RNE-N-TR and RGN-V-TR, in comparison with SPM and CPD-NLS. The average of the relative error factor in RNE-N-TR and RGN-V-TR is less than 1. The numerical results in [154, Example 5.4] for GP method and small perturbations ($\epsilon \in \{10^{-2}, 10^{-4}, 10^{-6}\}$), shows that the numerical quality in GP-OPT method is worse than with these methods.

We also compare CPD-NLS, RNE-N-TR and RGN-V-TR for perturbation of random tensors of rank $r > n$ and report the minimal and maximal relative error with the average number of iterations N (rounded to the closest integer) and the average time t (in seconds) in Table 4.5. The

r	ϵ	ref_{spm}	t_{spm}	N_{spm}	ref_{rne}	t_{rne}	N_{rne}	ref_{rgn}	t_{rgn}	N_{rgn}
1	1	0.103	0.04	28	0.105	0.07	2	0.11	0.083	3
	10^{-1}	0.103	0.039	28	0.104	0.04	2	0.11	0.069	3
	10^{-2}	0.1	0.04	28	0.1	0.04	2	0.103	0.058	2
	10^{-4}	0.101	0.041	29	0.101	0.041	2	0.166	0.044	2
	10^{-6}	0.104	0.041	30	0.104	0.041	2	0.17	0.045	2
2	1	0.15	0.1	69	0.175	0.137	3	0.159	0.16	3
	10^{-1}	0.15	0.091	65	0.153	0.076	2	0.159	0.13	3
	10^{-2}	0.144	0.086	66	0.149	0.072	2	0.15	0.111	2
	10^{-4}	0.148	0.089	66	0.157	0.076	2	0.199	0.076	2
	10^{-6}	0.146	0.087	67	0.151	0.073	2	0.195	0.073	2
3	1	0.185	0.126	109	0.194	0.172	3	0.194	0.208	3
	10^{-1}	0.185	0.135	111	0.195	0.128	2	0.195	0.198	3
	10^{-2}	0.187	0.119	113	0.208	0.099	2	0.195	0.175	2
	10^{-4}	0.182	0.102	106	0.197	0.092	2	0.217	0.095	2
	10^{-6}	0.183	0.101	105	0.196	0.094	2	0.206	0.097	2
4	1	0.217	0.159	168	0.25	0.546	8	0.225	0.278	3
	10^{-1}	0.218	0.161	168	0.245	0.319	4	0.228	0.241	3
	10^{-2}	0.211	0.163	162	0.241	0.134	2	0.219	0.239	3
	10^{-4}	0.216	0.167	169	0.26	0.128	2	0.261	0.136	2
	10^{-6}	0.227	0.167	168	0.259	0.126	2	0.259	0.133	2
5	1	0.244	0.207	217	0.339	1.199	13	0.252	0.594	5
	10^{-1}	0.244	0.221	220	0.255	0.252	2	0.252	0.317	3
	10^{-2}	0.247	0.223	218	0.292	0.175	2	0.254	0.321	3
	10^{-4}	0.246	0.215	213	0.304	0.16	2	0.304	0.165	2
	10^{-6}	0.249	0.231	226	0.307	0.158	2	0.311	0.165	2

Table 4.3 – Computational results of SPM (spm) (initialized by a random vector of size $n = 10$ obeying normal distribution for each tensor instance) RNE-N-TR (rne), and RGN-V-TR (rgn) (initialized by the method SMD for each tensor instance) for rank-r approximations in $\mathcal{S}^4_{10}(\mathbb{R})$. The method SPM stops when the distance between two consecutive iterates is less than 10^{-10} or when the maximal number of iterations (fixed to 400) is reached. The methods RNE-N-TR and RGN-V-TR stop when the maximum number of iterations is reached (fixed to 200) or when the radius of the trust region Δ_k is less than $\Delta_{\min}$ (by default 10^{-3}).

considered cases in Table 4.5 are for the degree d, the number of variables n and the rank r such that (d, n, r) is respectively (5,4,10), (5,15,20), (6,5,12), and (7,8,15). We see that the maximal relative error factor *ref* reached by RNE-N-TR and RGN-V-TR with initial point by SMD is less than 1. There is an exception in the first case when $\epsilon = 1$, where a large number of iterations is needed for RNE-N-TR and RGN-V-TR. On the other hand, the minimal relative error of CPD-NLS is less than 1 in almost all Table 4.5, whereas its maximal relative error is higher than 1 in all Table 4.5.

This numerical experiment indicates that for these examples of random low rank tensors with random noise, SMD provides a good initial point, close enough to a good solution, so that RNE-N-TR and RGN-V-TR need a few number of iterations. In this context, the combination of an adaptive choice of initial point and a Newton-type method is successful.

4.3.5 Symmetric tensor with large differences in the scale of the weight vector

Consider the case of a real symmetric tensor $\boldsymbol{t} = \sum_{i=1}^{r} w_i (v_i^t \boldsymbol{x})^d$, $\|v_i\| = 1$, $w_i > 0$, with large differences in the scale of the weights w_i i.e. $\frac{\max_i w_i}{\min_i w_i}$ is large. More precisely, there are large differences in the norms of the rank-1 symmetric tensors $w_i (v_i^t \boldsymbol{x})^d$. We randomly sample real symmetric tensors of order $d = 3$ and dimension $n = 7$ with $r \in \{5, 10, 15, 20\}$, according to the

r	ϵ	ref_{cpd}	t_{cpd}	N_{cpd}	ref_{rne}	t_{rne}	N_{rne}	ref_{rgn}	t_{rgn}	N_{rgn}
1	1	0.117	0.05	10	0.11	0.06	2	0.115	0.069	3
	10^{-1}	0.118	0.046	10	0.108	0.054	2	0.112	0.084	3
	10^{-2}	0.116	0.043	10	0.107	0.044	2	0.11	0.06	2
	10^{-4}	0.114	0.042	10	0.107	0.037	2	0.227	0.038	2
	10^{-6}	0.113	0.037	11	0.112	0.036	2	0.237	0.037	2
2	1	0.167	0.072	14	0.162	0.078	2	0.166	0.118	3
	10^{-1}	0.169	0.077	14	0.164	0.063	2	0.167	0.111	3
	10^{-2}	0.162	0.071	14	0.163	0.061	2	0.163	0.09	2
	10^{-4}	0.171	0.071	14	0.163	0.062	2	0.204	0.063	2
	10^{-6}	0.175	0.069	13	0.162	0.062	2	0.23	0.064	2
3	1	0.201	0.115	16	0.204	0.135	2	0.204	0.163	3
	10^{-1}	0.223	0.109	17	0.206	0.091	2	0.203	0.157	3
	10^{-2}	0.228	0.117	17	0.209	0.086	2	0.203	0.152	2
	10^{-4}	0.202	0.103	15	0.205	0.091	2	0.243	0.093	2
	10^{-6}	0.284	0.124	19	0.211	0.088	2	0.234	0.091	2
4	1	0.235	0.149	18	0.234	0.192	3	0.234	0.23	3
	10^{-1}	0.232	0.165	19	0.244	0.132	2	0.238	0.215	3
	10^{-2}	0.237	0.142	17	0.25	0.113	2	0.232	0.219	3
	10^{-4}	0.238	0.158	19	0.25	0.112	2	0.255	0.117	2
	10^{-6}	0.232	0.161	19	0.254	0.111	2	0.274	0.116	2
5	1	0.275	0.21	22	0.261	0.269	3	0.261	0.345	3
	10^{-1}	0.264	0.186	19	0.269	0.211	2	0.261	0.288	3
	10^{-2}	0.266	0.211	22	0.305	0.148	2	0.264	0.292	3
	10^{-4}	0.265	0.169	18	0.293	0.158	2	0.299	0.163	2
	10^{-6}	0.266	0.206	21	0.298	0.158	2	0.301	0.161	2

Table 4.4 – Computational results of CPD-NLS (initialized by a random symmetric initial point obeying normal distribution for each tensor instance), RNE-N-TR, and RGN-V-TR (initialized by the method SMD for each tensor instance) for rank-r approximations in $\mathcal{S}^4_{10}(\mathbb{C})$. The method CPD-NLS stops when the stop criteria given by $\text{Display} = 10$ in Tensorlab are verified. The methods RNE-N-TR and RGN-V-TR stop when the maximum number of iterations is reached (fixed to 200) or when the radius of the trust region Δ_k is less than $\Delta_{\min}$ (by default 10^{-3}).

following model :

$$\boldsymbol{t} = \sum_{i=1}^{r} 10^{\frac{is}{r}} (v_i^t \boldsymbol{x})^d, \ \|v_i\| = 1.$$

The components of the weight vector increase exponentially from $10^{\frac{s}{r}}$ to 10^s.

We aim to compare the performance of RNE-N-TR and RGN-V-TR methods (hereafter called respectively RNE and RGN for shortness) in this configuration. We run the following test :

— Take $\boldsymbol{t}$ as above, and create a perturbated tensor $\boldsymbol{t}_p = \frac{\boldsymbol{t}}{\|\boldsymbol{t}\|} + 10^{-5} \frac{\boldsymbol{t}_{\text{err}}}{\|\boldsymbol{t}_{\text{err}}\|}$, where $\boldsymbol{t}_{\text{err}} \in \mathbb{R}[\mathbf{x}]_d$ is a random symmetric tensor with coefficients obeying Gaussian distributions ;

— run 20 random initial points obeying Gaussian distributions ;

— run RNE and RGN with a maximum of iterations $N_{\max} = 500$, and report in average respectively : the relative error (in geometric average) $\text{err}_{\text{rel}} := \left\| \frac{\boldsymbol{t}}{\|\boldsymbol{t}\|} - \boldsymbol{t}_* \right\|_d$, where $\boldsymbol{t}_*$ is a rank-r symmetric decomposition obtained by these methods ; the number of iterations N_{iter} ; and the computation time t in seconds (s). We also report the number N_{opt} of instances where $\text{err}_{\text{rel}} \leq 1.1.10^{-5}$.

d n r	ϵ	ref_{cpd}		t_{cpd}	N_{cpd}	ref_{rne}		t_{rne}	N_{rne}	ref_{rgn}		t_{rgn}	N_{rgn}
		min	max	avg	avg	min	max	avg	avg	min	max	avg	avg
5 4 10	1	0.803	7.8	2.1	162	0.834	25.7	3.4	273	0.815	1.1	1.2	49
	10^{-2}	0.849	881.2	2.3	172	0.718	0.933	0.197	16	0.718	0.933	0.078	4
	10^{-4}	1.5	9.8e+4	2	157	0.711	0.933	0.0379	3	0.711	0.933	0.0535	3
	10^{-6}	776.4	1.9e+7	2	184	0.789	0.912	0.042	3	0.789	0.912	0.071	4
5 15 20	1	0.15	1.9e+3	9.8	45	0.153	0.172	22.6	3	0.153	0.172	27.1	3
	10^{-2}	0.149	1.2e+5	12.7	62	0.151	0.183	13.6	2	0.148	0.169	26.9	3
	10^{-4}	0.152	9.2e+6	13.8	67	0.152	0.181	13.6	2	0.152	0.181	14.7	2
	10^{-6}	0.155	1.4e+9	11.9	59	0.156	0.173	13.8	2	0.156	0.174	14.8	2
6 5 12	1	0.515	109.7	1.4	61	0.467	0.706	0.342	4	0.467	0.622	0.353	4
	10^{-2}	0.519	2.4e+4	3.8	143	0.472	0.62	0.155	3	0.472	0.62	0.205	3
	10^{-4}	0.518	9.6e+5	2.9	137	0.493	0.622	0.222	4	0.493	0.622	0.35	5
	10^{-6}	1.1	9.7e+7	2.3	112	0.647	0.6	0.098	2	0.492	0.591	0.211	3
7 8 15	1	0.183	2.1e+3	54.9	46	0.171	0.21	8.3	3	0.171	0.21	8.5	3
	10^{-2}	0.174	5.3e+4	52.3	47	0.137	0.171	4.7	2	0.169	0.201	8.1	3
	10^{-4}	0.168	3.5e+6	63.1	54	0.138	0.169	4.5	2	0.138	0.169	4.7	2
	10^{-6}	0.179	1.1e+9	75.6	65	0.142	0.177	4.4	2	0.142	0.177	4.6	2

Table 4.5 – Computational results of CPD-NLS (initialized by a random symmetric initial point obeying normal distribution for each tensor instance), RNE-N-TR, and RGN-V-TR (initialized by the method SMD for each tensor instance). The method CPD-NLS stops when the stop criteria given by $\text{Display} = 10$ in Tensorlab are verified. The methods RNE-N-TR and RGN-V-TR stop when the maximum number of iterations is reached (fixed to 200) or when the radius of the trust region Δ_k is less than $\Delta_{\min}$ (by default 10^{-3}).

$r = 5$						
s	1		2		3	
Alg	RNE	RGN	RNE	RGN	RNE	RGN
err_{rel}	0.456	5.8e-6	0.411	5.5e-6	0.246	1.4e-5
N_{iter}	120	39	165	61	175	77
t	2.0	1.1	2.4	1.4	2.5	1.8
N_{opt}	0	20	0	20	0	17

$r = 10$						
s	1		2		3	
Alg	RNE	RGN	RNE	RGN	RNE	RGN
err_{rel}	0.372	9.4e-6	0.195	1.6e-6	0.224	6.8e-5
N_{iter}	423	87	270	186	392	206
t	16.9	6.3	10.8	13.7	15.5	15.0
N_{opt}	0	18	0	20	0	9

$r = 15$						
s	1		2		3	
Alg	RNE	RGN	RNE	RGN	RNE	RGN
err_{rel}	0.198	9.9e-6	0.242	9.9e-6	0.184	1.1e-5
N_{iter}	500	17	476	21	426	25
t	37.3	2.1	34.6	2.6	30.9	3.2
N_{opt}	0	20	0	20	0	20

$r = 20$						
s	1		2		3	
Alg	RNE	RGN	RNE	RGN	RNE	RGN
err_{rel}	0.098	9.9e-6	0.165	1.0e-5	0.221	9.9e-6
N_{iter}	500	12	469	12	483	12
t	54.4	2.4	59.3	2.9	59.1	2.6
N_{opt}	0	20	0	20	0	20

Table 4.6 – Computational results of RNE-N-TR and RGN-V-TR for scaled weights. The two methods are initialized for each s by the same 20 random initial points obeying Gaussian distributions. They stop when the maximum number of iterations is reached (fixed to 500) or when the radius of the trust region Δ_k is less than $\Delta_{\min}$ (by default 10^{-3}).

The results in Table 4.6 show that RGN outperforms RNE. In fact, the average of the relative error in RGN is better, up to five order of magnitude, than in RNE. Moreover, starting from the same 20 random initial points in the two methods ; RGN succeeded to reach an optimum, at least in 9 instances with the different order of scale s, while RNE could not find any optimum. Notice that, as we mentionned before, the cost of one iteration in RGN is higher than in RNE. The good performance of RGN compared to RNE in this test was expected, since the orthonormal basis of the tangent space computed in RGN method is independent of the weight factor. This behavior was also observed in [33, Subsection 3.4] for real multilinear tensors, parametrized by Segre manifolds.

4.4 Practical session

We present two simple examples to show the implementation of the methods described in this chapter, available in the Julia package TensorDec.jl†.

[2]:
```
using TensorDec
using DynamicPolynomials
using MomentTools
using CSDP, JuMP
# The function "Optimizer" is a global optimization solver based on positive␣
 ↪semi-definite programming
optimizer = CSDP.Optimizer
using LinearAlgebra
```

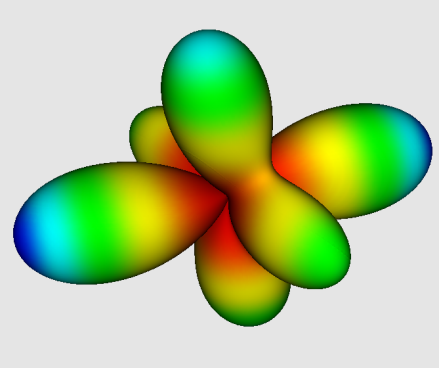
Example 1:

[3]:
```
# Define the parameters
X = @polyvar x1 x2 x3
```

[3]:
```
(x1, x2, x3)
```

[4]:
```
# P is a homogeneous polynomial of degree 4 in 3 variables
P = (x1+x2+0.75*x3)^4+1.5*(x1-x2)^4-2*(x1-x3)^4;
```

The graph of P in polar coordinates on the sphere looks like this:



Let us use the function "optimizer" to minimize and maximize P on the unit sphere. The maximum evaluation of P in absolute value on the unit sphere (and that is why we have to use both maximize and minimize functions) gives the spectral norm of P and equivalently a best rank-1 approximation of the symmetric tensor associated to P.

[5]:
```
v1, M1 = minimize(P, [x1^2+x2^2+x3^2-1], [], X, 8, optimizer);
v2, M2 = maximize(P, [x1^2+x2^2+x3^2-1], [], X, 8, optimizer);
```

†. https://gitlab.inria.fr/AlgebraicGeometricModeling/TensorDec.jl

```
CSDP 6.2.0
Iter:  0 Ap: 0.00e+000 Pobj:  0.0000000e+000 Ad: 0.00e+000 Dobj:  0.0000000e+000
Iter:  1 Ap: 6.21e-001 Pobj: -4.7686654e+001 Ad: 3.23e-001 Dobj:  7.7359277e+000
Iter:  2 Ap: 1.00e+000 Pobj: -6.7310911e+002 Ad: 3.23e-001 Dobj:  9.7555745e+000
Iter:  3 Ap: 1.00e+000 Pobj: -6.2816076e+002 Ad: 8.92e-001 Dobj:  1.2402191e+000
⋮
Iter: 20 Ap: 9.60e-001 Pobj: -7.7015795e+000 Ad: 9.55e-001 Dobj: -7.7015795e+000
Success: SDP solved
Primal objective value: -7.7015795e+000
Dual objective value: -7.7015795e+000
Relative primal infeasibility: 1.69e-014
Relative dual infeasibility: 1.46e-010
Real Relative Gap: -2.15e-011
XZ Relative Gap: 9.09e-010
DIMACS error measures: 3.03e-014 0.00e+000 4.40e-009 0.00e+000 -2.15e-011
9.09e-010
CSDP 6.2.0
Iter:  0 Ap: 0.00e+000 Pobj:  0.0000000e+000 Ad: 0.00e+000 Dobj:  0.0000000e+000
Iter:  1 Ap: 6.19e-001 Pobj: -4.7582096e+001 Ad: 3.23e-001 Dobj: -8.7051506e+000
Iter:  2 Ap: 1.00e+000 Pobj: -6.7805576e+002 Ad: 3.21e-001 Dobj: -1.1060713e+001
Iter:  3 Ap: 1.00e+000 Pobj: -6.3256373e+002 Ad: 8.92e-001 Dobj: -1.4167349e+000
⋮
Iter: 21 Ap: 9.60e-001 Pobj: -6.5652500e+000 Ad: 9.53e-001 Dobj: -6.5652500e+000
Success: SDP solved
Primal objective value: -6.5652500e+000
Dual objective value: -6.5652500e+000
Relative primal infeasibility: 2.80e-015
Relative dual infeasibility: 4.47e-010
Real Relative Gap: -7.76e-011
XZ Relative Gap: 2.82e-009
DIMACS error measures: 5.01e-015 0.00e+000 1.35e-008 0.00e+000 -7.76e-011
2.82e-009
```

The minimum evaluation of P on the unit sphere is: -7.701579459519532.

The maximum evaluation of P on the unit sphere is: 6.565249952183416.

Thus, the maximum weight in absolute value which is the spectral norm of P is: 7.701579459519532.

The unit vectors that give this value are: [0.6805571886747267, 0.048429787707634814, -0.7310926539221407] and [-0.6805571886747267, -0.048429787707634814, 0.7310926539221407].

Let us compute a rank-1 approximation of P. We will compute an initial point by the method SMD, the Julia function that corresponds to this method in the package TensorDec is called "decompose", then we will use the Riemannian Newton algorithm with trust region scheme for the real case, the corresponding Julia function in the package TensorDec is called "rne_n_tr_r".

[16]:
```
# Compute an initial point
w1, V1 = decompose(P,1)
```

[16]:
```
([-7.089690728998527], [0.6841218273996007; 0.04220544010380849;
-0.7281456077606144], Dict{String,Any}("diagonalization" =>
Dict{String,Any}("case" => "1x1")))
```

We can notice that the weight given by decompose is close to the one given by Optimizer. Let us refine this point by using a few number of iterations of rne_n_tr_r, for example 5 iterations.

[26]:
```
w_end, V_end = rne_n_tr_r(P, w1, V1, Dict{String,Any}("maxIter" => 5,"epsIter"␣
↪=> 1.e-3))
```

[26]:
```
([-7.701576525649196], [0.68061769889553; 0.04831896554028091;
-0.7310436550024019], Dict{String,Any}("d*" => 8.819736152809341,"d0" =>
8.841950556430849,"nIter" => 5,"epsIter" => 0.001,"maxIter" => 5))
```

The weight in absolute value given by rne_n_tr_r initialized by decompose for rank-1 symmetric tensor approximation is: 7.701576525649196.

The unit vector given by rne_n_tr_r initialized by decompose for rank-1 symmetric tensor approximation is: [0.68061769889553; 0.04831896554028091; -0.7310436550024019].

Verifying with the global optimization method "optimizer" from the package CSDP, the symmetric rank-1 approximation $w_{end}(v_{end}^t X)^4$ of P given by "rne_n_tr_r" initialized by "decompose" is a best rank-1 approximation.

Example 2:

Let us take a random symmetric tensor normally distributed with complex coefficients of order 4 and dimension 3 (the generic symmetric rank is 5), and let us compute by the Riemannian Newton algorithm "rne_n_tr" and the Riemannian Gauss–Newton algorithm "rgn_v_tr" initialized by a random initial point obeying normal distribution an approximated rank-3 symmetric tensor.

[38]:
```
# Take a random symmetric tensor
using Tensors
n = 3; d = 4; r = 3
T = randn(SymmetricTensor{d, n})+randn(SymmetricTensor{d, n})*im
T = convert(Array,T)
# show the first 3 arrays of T
T[:,:,:,1]
```

[38]:
```
3×3×3 Array{Complex{Float64},3}:
[:, :, 1] =
    0.304489-2.14852im   -0.00230603+2.81im      -0.621412-0.185586im
 -0.00230603+2.81im          1.26159-0.687682im   0.677236-0.314596im
   -0.621412-0.185586im     0.677236-0.314596im   0.652919-1.50752im
```

```
[:, :, 2] =
  0.114497+0.472617im  -0.454587+2.23814im    -0.215587-1.08674im
 -0.454587+2.23814im    0.765382+0.970151im      2.2932-1.98527im
 -0.215587-1.08674im      2.2932-1.98527im   -0.0570214+0.405164im

[:, :, 3] =
 -1.38547-0.945676im   0.319933+0.53044im   0.554848+1.60772im
 0.319933+0.53044im   -0.323876+0.698715im  -1.67809-2.0286im
 0.554848+1.60772im    -1.67809-2.0286im    -1.38897+0.225804im
```

[39]:
```
# Take the associate homogeneous polynomial P to T by applying the function ahp␣
 ↪(for associate homogeneous polynomial)
X = (@polyvar x[1:n])[1]
P = ahp(T, X);
```

[58]:
```
# Take an initial point
w = ones(r) + fill(0.0+0.0im,r);
V = randn(ComplexF64,n,r);

# Apply rne_n_tr
w_end, V_end, Info = rne_n_tr(P, w, V, Dict{String,Any}("maxIter" => 500,␣
 ↪"epsIter" => 1.e-3))
```

[58]:
```
([3.3533378582723405, 1.7528646663899954, 5.021876225230323],
Complex{Float64}[-0.610048013088563 + 0.2278059897255048im 0.7944923164400758 +
0.034592585203756895im -0.04277847243642945 - 0.4515491977610789im;
0.5078071901018043 + 0.4498236972987523im 0.3999428679776847 +
0.36458354731448234im 0.4336335307863758 - 0.3802580876158061im;
-0.3373082465526216 + 0.044266224244519994im -0.26111415928223225 +
0.08080375839222821im 0.6788726483700402 - 0.027766875461433867im],
Dict{String,Any}("d*" => 7.856359646235264,"d0" => 31.118041861573218,"nIter" =>
10,"epsIter" => 0.001,"maxIter" => 500))
```

[59]:
```
# Adjust the initial point to use with rgn_v_tr since this function takes only␣
 ↪the matrix V as parameter without the weight vector
for i in 1:r
    V[:,i]=(w[i])^(1/d)*V[:,i]
end

# Apply rgn_v_tr
V_end, Info = rgn_v_tr(P, V, Dict{String,Any}("maxIter" => 500,"epsIter" => 1.
 ↪e-3))
```

[59]:
```
(Complex{Float64}[-1.197237252086016 + 0.4498363042506125im 0.6784383624116359 +
0.5660791048163021im -0.7849864638452732 - 0.014835935134524971im;
0.3788396401411271 + 0.2511463936422992im -0.974262084689121 -
0.2574674292040156im -0.6433234546249749 - 0.670531077035846im;
```

```
-0.5027488452521038 + 0.6166859533178 09im 0.9073543082281365 +
0.35258183865185777im -0.0088033550999379 - 0.9139555194450166im],
Dict{String,Any}("d*" => 2.8904022783502117,"d0" => 31.118041861573218,"nIter"
=> 27,"epsIter" => 0.001,"maxIter" => 500))
```

The reported error is the apolar norm between P and the approximated polynomial.

The initial error "d0" is ~ 31.11.

The algorithm rne_n_tr takes 10 iterations and decreases to the final error d* ~ 7.85, while the algorithm rgn_v_tr decreases to the final error ~ 2.89, after 27 iterations.

4.5 Conclusion

We presented two Riemannian Newton optimization methods for approximating a given complex-valued symmetric tensor by a low rank symmetric tensor. We used in subsection 4.2.1 the weighted normalized factor matrices parametrization for the constraint set. We developed an exact Riemannian Newton iteration with exact computation of the Hessian matrix (RNE-N-TR). We exploited in subsection 4.2.1.1 the properties of the apolar product and of partial complex derivatives, to deduce a simplified and explicit computation of the gradient and Hessian of the square distance function in terms of the points, weights of the decomposition and the tensor to approximate. We proved that under some regularity conditions on non-defective tensors in the neighborhood of the initial point, the iteration is converging to a local minimum. In subsection 4.2.2, we parametrized the constraint set via Cartesian product of Veronese manifolds. Taking into account the geometry of the Veronese manifold, we constructed a suitable basis for its tangent space at a given point on this manifold. Using this basis, we developed a Gauss–Newton iteration (RGN-V-TR). In subsection 4.2.2.1, we presented a retraction operator on the Veronese manifold. We showed that, combined with SMD method for choosing the initial point, the two methods have a good practical behavior in several experiments : in subsection 4.3.2 to compute a best real rank-1 approximation of a real symmetric tensor, in subsection 4.3.3 to compute a low rank approximation of sparse symmetric tensors, and in subsection 4.3.4 to compute low rank approximations of random perturbations of low rank symmetric tensors. In subsection 4.3.5, we showed that the numerical behavior of RNE-N-TR is affected by large differences in the scaling of the rank-1 symmetric tensor, where RGN-V-TR outperformed this algorithm in this case.

It was clear throughout the numerical experiments in this chapter, the good impact of the initial point chosen by SMD algorithm, which is based on simultaneous diagonalization of a pencil of matrices built from the symmetric tensor to approximate, on the numerical performance of the Riemannian Newton and the Riemannian Gauss–Newton algorithms presented in this chapter. Herein, in the next chapter, we will focus on the simultaneous diagonalization problem of a pencil of matrices, and its connection to the tensor rank approximation problem.

CHAPTER 5

On the simultaneous matrix diagonalization problem

In this chapter we study the simultaneous matrix diagonalization problem. Mainly, a pencil of matrices $M = [M_1, \ldots, M_s]$ is called in this chapter simultaneously diagonalizable, if there exists two invertible matrices E and F such that $\Sigma_i := FM_kE$ is a diagonal matrix, for $k \in \{1, \ldots, s\}$. Our results are presented in three sections. In the first section, we assume that the pencil of matrices is simultaneously diagonalizable, and we construct a Newton-type sequence that converges quadratically towards the solution $(E, F, (\Sigma_i)_{1 \leq i \leq s})$. Moreover, we exhibit a certification test that the sequence converges towards the solution. In the second section, the considered pencil of matrices is not simultaneously diagonalizable, and thus the objective is to find two matrices E and F that diagonalize approximately the pencil, i.e. to approximate the pencil to a pencil of simultaneously diagonalizable matrices. To solve this problem, we present a Riemannian conjugate gradient algorithm. As an application of the aforementioned algorithm, the third section shows that it can be used to compute an approximated rank-r decomposition for a three dimensional tensor with r higher than the two first dimensional sizes. To this end in the third section, we connect tensor decomposition and simultaneous diagonalization of matrices. We develop an algorithm based on alternate optimization method that combines two steps. The first step uses the Riemannian conjugate gradient algorithm from the second section and the second step solves a linear least-squares problem.
Keywords : *Simultaneous diagonalization, Newton-type method, eigenproblem, eigenvalues, high precision computation, approximate simultaneous matrix diagonalization, Riemannian conjugate gradient algorithm, alternate optimization algorithm.*

In this chapter, we present in Section 5.1 a Newton-type approach for the simultaneous matrix diagonalization problem. In Section 5.2, we present a Riemannian conjugate gradient algorithm for the approximate simultaneous matrix diagonalization problem. Finally, we introduce in Section 5.3, an algorithm based on simultaneous diagonalization of matrices for the low rank approximation problem for three dimensional real tensors with approximation rank higher than the first two mode dimensions.

5.1 Newton-type methods for simultaneous matrix diagonalization

In this Section we introduce a Newton-type method for the simultaneous matrix diagonalization problem. Sections 5.1.2, 5.1.3, 5.1.4, and 5.1.5 are respectively devoted to constructing a sequence that converges quadratically towards the numerical solution and to provide a certification test for its quadratic convergence for respectively the following systems :

— $FE - I_n = 0$,
— the system of one diagonalizable matrix,
— the system of two simultaneously diagonalizable matrices,
— the system of a pencil of simultaneously diagonalizable matrices.

We perform numerical experimentation in Section 5.1.6.

5.1.1 Notation and preliminaries

Throughout this section, we will use the infinity vector norm and the corresponding matrix norm. For a given vector $v \in \mathbb{C}^n$ and matrix $M \in \mathbb{C}^{n\times n}$, they are respectively given by :

$$\begin{aligned} \|v\|_{\max} &= \max\{|v_1|, \ldots, |v_n|\} \\ \|M\|_{\max} &= \sup_{\|v\|_{\max}=1} \|Mv\|_{\max}. \end{aligned}$$

Explicitly, $\|M\|_{\max} = \max\{|m_{i,1}| + \ldots + |m_{i,n}| : 1 \leq i \leq n\}$.
For a second matrix $N \in \mathbb{C}^{n\times n}$, we have

$$\begin{aligned} \|M+N\|_{\max} &\leqslant \|M\| + \|N\|_{\max} \text{ (sub-additivity)} \\ \|MN\|_{\max} &\leqslant \|M\|_{\max}\|N\|_{\max} \text{ (sub-multiplicativity).} \end{aligned}$$

Moreover, for a given matrix $M \in \mathbb{C}^{n\times n}$, we denote by $\|M\|_{\mathrm{L}}$ the following :

$$\|M\|_{\mathrm{L,Tri}} := \max_{\substack{1\leq i\leq n \\ 1\leq j\leq i-1}} |m_{i,j}|,$$

i.e the max matrix norm of the lower triangular part of M,

Furthermore, we consider in this section the regular case of diagonalizable matrices, that is, the matrices are diagonalizable with simple eigenvalues. Thus we will use the following notation

$$\mathcal{W}_n := \{M \in \mathbb{C}^{n\times n} \mid M \text{with pairwise distinct eigenvalues}\}.$$

It is well-known that $\mathcal{W}_n$ is dense in $\mathbb{C}^{n\times n}$.

We denote by $\mathcal{D}_n \subset \mathbb{C}^{n\times n}$, the vector space of diagonal matrices of size n and $\mathcal{D}'_n$ denotes the subset of $\mathcal{D}_n$ in which the diagonal matrices are of n distinct diagonal entries. Let $E, F \in \mathrm{GL}_n$ and $\Sigma \in \mathcal{D}'_n$. The perturbation of respectively E, F and Σ that we consider in this section are of the following form : $E + \dot{E}$, $F + \dot{F}$ and $\Sigma + \dot{\Sigma}$, where $\dot{E}$ and $\dot{F}$ are respectively in $T_E\mathrm{GL}_n$ and $T_F\mathrm{GL}_n$ and $\dot{\Sigma}$ is in $T_\Sigma\mathcal{D}'_n$.
As GL_n is a Lie group, $\dot{E}$ and $\dot{F}$ can be written as EX and YF such that X, Y are in $\mathbb{C}^{n\times n}$ (see Section 3.5.4).
As $\mathcal{D}'_n$ is open in $\mathcal{D}_n$ then $T_\Sigma\mathcal{D}'_n = \mathcal{D}_n$, herein $\dot{\Sigma} = S \in \mathcal{D}_n$.

Finally, the perturbations of E, F and Σ that we consider are as follows :
$E+EX$, $F+YF$ and $\Sigma+S$, such that X and Y are in $\mathbb{C}^{n\times n}$ and S is a diagonal matrix in $\mathbb{C}^{n\times n}$.

We state the following lemma which will be used in some of the proofs in this section.

Lemma 5.1.1. *Let* $\varphi(\varepsilon, u) = \frac{\prod_{j\geqslant 0}(1+u\varepsilon^{2^j})-1}{\varepsilon u}$. *Given* $\varepsilon \leqslant \frac{1}{2}$, $u \leqslant 1$, *and* $i \geqslant 0$, *we have*

$$\prod_{j\geqslant 0}(1 + u\varepsilon^{2^{j+i}}) \quad \leqslant \quad 1 + 2u\varepsilon^{2^i} \tag{5.1}$$

Proof. Modulo taking ε^{2^i} instead of ε, it suffices to consider the case when $i = 0$. Now $\varphi(\varepsilon, u)$ is an increasing function in ε and u, since its power series expansion in ε and u admits only positive coefficients. Consequently, $\varphi(\varepsilon, u) \leqslant \varphi(\frac{1}{2}, 1) = 2$. □

5.1.2 Newton-type method for the system $FE - I_n = 0$.

Let $f : \mathrm{GL}_n \times \mathrm{GL}_n \to \mathbb{C}^{n\times n},\ (E, F) \mapsto FE - I_n$. We consider the following perturbations $E + EX$, $F + YF$ of respectively E and F where $X,\ Y \in \mathbb{C}^{n\times n}$.
To define the Newton sequence we have to solve the linear system obtained by canceling the linear part in the Taylor expansion of $f(E + EX, F + YF)$. The same methodology will be adopted in the next sections for the other considered systems. Hereafter, we detail the computation of the Newton sequence associated to the system $FE - I_n = 0$. Moreover, a sufficient condition on the initial point for the quadratic convergence of this Newton sequence will be established.
Let $Z = FE - I_n$. We observe that

$$\begin{aligned} f(E + EX, F + YF) &= (F + YF)(E + EX) - I_n & (5.2)\\ &= Z + (Z + I_n)X + Y(Z + I_n) + Y(Z + I_n)X. & (5.3)\end{aligned}$$

We assume here that Z is of small norm i.e. we start from an initial point (E_0, F_0) close from the solution of the system $FE - I_n = 0$.
Consequently, the linear system of first order terms to solve is

$$Z + X + Y = 0. \tag{5.4}$$

Hence $X = Y = -\frac{Z}{2}$ is a solution of Constraint (5.4). Moreover we get, by substituting in Equation (5.3) X and Y by $-\frac{Z}{2}$,

$$(F + YF)(E + EX) - I_n \quad = \quad Z^2\left(-\frac{3}{4}I_n + \frac{Z}{4}\right). \tag{5.5}$$

Proposition 5.1.2. *Let $Z_0 = F_0E_0 - I_n$. Define $X_0 = -\frac{Z_0}{2}$, $E_1 = E_0(I_n + X_0)$, $F_1 = (I_n + X_0)F_0$ and $Z_1 = F_1E_1 - I_n$. Assume that $\|Z_0\|_{\max} \leqslant 1$. Then*

$$\|Z_1\|_{\max} \leqslant \|Z_0\|_{\max}^2 \tag{5.6}$$

Proof. It follows easily from (5.5). □

Theorem 5.1.3. *Let E_0 and F_0 two complex square matrices of size n. Let $Z_0 = F_0E_0 - I_n$ and assume that $\varepsilon = \|Z_0\|_{\max} < \frac{1}{2}$. The sequences defined for $i \geqslant 0$*

$$\begin{aligned} Z_i &= F_iE_i - I_n \\ X_i &= -\frac{Z_i}{2} \\ E_{i+1} &= E_i(I_n + X_i) \\ F_{i+1} &= (I_n + X_i)F_i \end{aligned}$$

converge quadratically towards the solution of $FE - I_n = 0$. Each E_i, respectively F_i are invertible and, if E_∞ and F_∞ are respectively the limits of sequences $(E_i)_{i\geqslant 0}$ and $(F_i)_{i\geqslant 0}$ we have for $i \geqslant 0$,

$$\begin{aligned} \|E_i - E_\infty\|_{\max} &\leqslant (1 + 2\varepsilon)2^{-2^{i+1}+1}\varepsilon\|E_0\|_{\max}, \\ \|F_i - F_\infty\|_{\max} &\leqslant (1 + 2\varepsilon)2^{-2^{i+1}+1}\varepsilon\|F_0\|_{\max}. \end{aligned}$$

Proof. Let us prove by induction that $\|Z_k\|_{\max} \leqslant 2^{-2^k+1}\varepsilon$. Since $\varepsilon < \frac{1}{2}$, we have

$$\begin{aligned} \|Z_{k+1}\|_{\max} &\leqslant \|Z_k\|_{\max}^2 \qquad \text{from (5.6)} \\ &\leqslant \varepsilon 2^{-2^{k+1}+2}\varepsilon \\ &\leqslant 2^{-2^{k+1}+1}\varepsilon. \end{aligned}$$

Consequently $Z_\infty = 0$. Since $X_k = -\frac{Z_k}{2}$ we deduce

$$\|X_k\|_{\max} \leqslant 2^{-2^k}\varepsilon.$$

It follows $X_\infty = 0$. We have

$$\begin{aligned} E_k &= E_{k-1}(I_n + X_{k-1}) \\ &= E_0(I_n + X_0)\cdots(I_n + X_{k-1}). \end{aligned}$$

Denoting $W_i = \prod_{0\leqslant k\leqslant i}(I_n + X_k)$, $W_\infty = \prod_{k\geqslant 0}(I_n + X_k)$ we compute

$$\begin{aligned} \|W_\infty - I_n\|_{\max} &\leqslant \prod_{k\geqslant 0}(1 + 2^{-2^k}\varepsilon) - 1 \\ &\leqslant 2\varepsilon \qquad \text{by using Lemma 5.1.1.} \end{aligned}$$

Then W_∞ is invertible and $\|W_\infty^{-1}\|_{\max} \leqslant \dfrac{1}{1 - 2\varepsilon}$. Let $E_\infty = E_0W_\infty$. Hence $E_0 = E_\infty W_\infty^{-1}$. In the same way $F_0 = W_\infty^{-1}F_\infty$. Finally, the identity $F_\infty E_\infty - I_n = 0$ permits to conclude that E_0

and F_0 are invertible. In the same way we prove easily that $\|W_i - I_n\|_{\max} \leqslant 2\varepsilon$. It follows that W_i is invertible. Since $E_i = E_0 W_i$ we deduce that E_i is invertible. Moreover

$$\begin{aligned}
\|W_i - W_\infty\|_{\max} &\leqslant \|W_i\|_{\max} \left\| 1 - \prod_{k \geqslant i+1} (1 + \|X_k\|_{\max}) \right\|_{\max} \\
&\leqslant (1 + \|W_i - I_n\|_{\max}) \left\| \prod_{k \geqslant 0} (1 + 2^{-2^{k+i+1}}\varepsilon) - 1 \right\|_{\max} \\
&\leqslant (1 + 2\varepsilon) 2^{-2^{i+1}+1} \varepsilon \qquad \text{by using Lemma 5.1.1.}
\end{aligned}$$

We deduce that

$$\|E_i - E_\infty\|_{\max} \leqslant (1 + 2\varepsilon) 2^{-2^{i+1}+1} \varepsilon \|E_0\|_{\max}.$$

These properties also hold for the F_i's. The theorem is proved. □

5.1.3 Newton-like method for diagonalizable matrices.

Let $M \in \mathcal{W}_n$, $\Sigma \in \mathcal{D}'_n$, $E,\ F \in \mathrm{GL}_n$. We aim to construct Newton sequences which converge towards the numerical solution of $f(E, F, \Sigma) = 0$ where $f : \mathrm{GL}_n \times \mathrm{GL}_n \times \mathcal{D}'_n \to \mathbb{C}^{n\times n} \times \mathbb{C}^{n\times n},\ (E, F, \Sigma) \mapsto (FE - I_n, FME - \Sigma)$. We consider in the same way as before the perturbations $E + EX$ and $F + YF$ of respectively E and F and in addition the perturbation $\Sigma + S$ of Σ such that $S \in \mathcal{D}_n$. We get with $Z = FE - I_n$ and $\Delta = FME - \Sigma$:

$$\begin{aligned}
&(F + YF)(E + EX) - I_n \\
&= Z + (Z + I_n)X + Y(Z + I_n) + Y(Z + I_n)X \qquad (5.7)\\
&(F + YF)M(E + EX) - \Sigma - S \\
&= FME - \Sigma - S + FMEX + YFME + YFMEX \\
&= \Delta - S + \Sigma X + Y\Sigma + \Delta X + Y\Delta + Y(\Delta + \Sigma)X \qquad (5.8)
\end{aligned}$$

As in the previous subsection we assume that (E, F, Σ) is sufficiently close to the solution of $f(E, F, \Sigma) = 0$, thus the linear system that we obtain from (5.7) and (5.8) is

$$\begin{cases} Z + X + Y & = 0 \\ \Delta - S + \Sigma X + Y\Sigma & = 0 \end{cases}$$

The following lemma gives a solution of this linear system.

Lemma 5.1.4. *Let $\Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_n)$, $Z = (z_{i,j})_{1\leq i,j\leq n}$ and $\Delta = (\delta_{i,j})_{1\leq i,j\leq n}$ be given matrices in $\mathbb{C}^{n\times n}$. Assume that $\sigma_i \neq \sigma_j$ for $i \neq j$. Let S, X and Y be matrices defined by*

$$\begin{aligned}
S &= \mathrm{ddiag}(\Delta - Z\Sigma) && (5.9)\\
x_{i,i} &= 0 && (5.10)\\
x_{i,j} &= \frac{-\delta_{i,j} + z_{i,j}\sigma_j}{\sigma_i - \sigma_j}, \qquad i \neq j && (5.11)\\
y_{i,i} &= -z_{i,i} && (5.12)\\
y_{i,j} &= \frac{\delta_{i,j} - z_{i,j}\sigma_i}{\sigma_i - \sigma_j}, \qquad i \neq j. && (5.13)
\end{aligned}$$

Then we have

$$\begin{aligned} Z + X + Y &= 0 \quad (5.14) \\ \Delta - S + \Sigma X + Y\Sigma &= 0. \quad (5.15) \end{aligned}$$

Proof. It easy to verify that $X + Y + Z = 0$. In this way the equation (5.15) is equivalent to

$$\Delta - S - Z\Sigma + \Sigma X - X\Sigma = 0.$$

Since $\mathrm{ddiag}(\Delta - S - Z\Sigma) = \mathrm{ddiag}(\Sigma X - X\Sigma) = 0$ the formulas which define X follow easily. □

In the next theorem we introduce the Newton sequences associated to the system $f(E, F, \Sigma) = 0$ with a sufficient condition on the initial point for its quadratic convergence.

Theorem 5.1.5. *Let $E_0, F_0 \in GL_n$ and $\Sigma_0 \in \mathcal{D}'_n$ be given such that they define the sequences for $i \geqslant 0$,*

$$\begin{aligned} Z_i &= F_i E_i - I_n \\ \Delta_i &= F_i M E_i - \Sigma_i \\ S_i &= \mathrm{diag}(\Delta_i - Z_i\Sigma_i) \\ E_{i+1} &= E_i(I_n + X_i) \\ F_{i+1} &= (I_n + Y_i)F_i \\ \Sigma_{i+1} &= \Sigma_i + S_i, \end{aligned}$$

where S_i, X_i and Y_i are defined by the formulas (5.9–5.13). Let us define $\kappa_0 = \max\left(1, \max_{i\neq j} \frac{1}{|\sigma_{0,i} - \sigma_{0,j}|}\right)$, $K_0 = \max(1, \max_i|\sigma_{0,i}|)$ and $\varepsilon_0 = \max(\kappa_0^2 K_0^2 \|Z_0\|_{\max}, \kappa_0^2 K_0 \|\Delta_0\|_{\max})$. Assume that

$$\varepsilon_0 \leqslant 0.033. \quad (5.16)$$

Then the sequences $(\Sigma_i, E_i, F_i)_{i\geqslant 0}$ converge quadratically to the solution of $(FE - I_n, FME - \Sigma) = 0$. More precisely E_0 and F_0 are invertible and

$$\begin{aligned} \|E_i - E_\infty\|_{\max} &\leqslant 8.1 \times 2^{1-2^{i+1}} \|E_0\|_{\max} \frac{\varepsilon_0}{\kappa K} \\ \|F_i - F_\infty\|_{\max} &\leqslant 8.1 \times 2^{1-2^{i+1}} \|F_0\|_{\max} \frac{\varepsilon_0}{\kappa K}. \\ \|\Sigma_i - \Sigma_\infty\|_{\max} &\leqslant 1.85 \times 2^{1-2^i} \frac{\varepsilon_0}{\kappa^2 K}. \end{aligned}$$

Proof. Let us denote for each $i \geqslant 0$,

$$\begin{array}{ccccccc} \varepsilon & = & \varepsilon_0 & \qquad \varepsilon_i & = & \max(\kappa_i^2 K_i^2 \|Z_i\|_{\max}, \kappa_i^2 K_i \|\Delta_i\|_{\max}) \\ \kappa & = & \kappa_0 & \qquad \kappa_i & = & \max\left(1,\ \max_{1\leqslant j<k\leqslant n} \frac{1}{|\sigma_{i,k} - \sigma_{i,j}|}\right) \\ K & = & K_0 & \qquad K_i & = & \max_{1\leq k\leq n}\left(1,\ |\sigma_{i,k}|\right), \end{array}$$

where $\sigma_{i,1}, \ldots, \sigma_{i,n}$ denote the diagonal entries of Σ_i. Let us show by induction on i that

$$\varepsilon_i \leqslant 2^{1-2^i}\varepsilon \tag{5.17}$$

$$\|\Sigma_i - \Sigma_0\|_{\max} \leqslant (2 - 2^{2-2^i})\frac{2a}{\kappa}\varepsilon \tag{5.18}$$

with $a = \dfrac{1}{1-8\varepsilon}$. These inequalities clearly hold for $i = 0$. Assuming that the induction hypothesis holds for a given i and let us prove it for $i+1$. We first prove that $\|\Sigma_{i+1} - \Sigma_0\|_{\max} \leqslant (2 - 2^{2-2^{i+1}})\frac{2a}{\kappa}\varepsilon$ under the assumption $\|\Sigma_i - \Sigma_0\|_{\max} \leqslant (2 - 2^{2-2^i})\frac{2a}{\kappa}\varepsilon$. To do this, at the first step we show that this implies $K - \frac{4a}{\kappa}\varepsilon \leqslant K_i \leq K + \frac{4a}{\kappa}\varepsilon$ and $\frac{1}{1+8a\varepsilon}\kappa \leqslant \kappa_i \leq \frac{\kappa}{1-8a\varepsilon}$. Let us prove $K - \frac{4a}{\kappa}\varepsilon \leqslant K_i \leqslant K + \frac{4a}{\kappa}\varepsilon$. We have

$$\begin{aligned} K_i := \|\Sigma_i\|_{\max} &\leq \|\Sigma_0\|_{\max} + \|\Sigma_i - \Sigma_0\|_{\max} \\ &\leqslant K + (2 - 2^{2-2^i})\frac{2a}{\kappa}\varepsilon \\ &\leqslant K + \frac{4a}{\kappa}\varepsilon \leqslant K(1 + 4a\varepsilon). \end{aligned}$$

This implies simultaneously $K_i \geqslant K - |K - K_i| \geqslant K - \frac{4a}{\kappa}\varepsilon$ and $K_i \geqslant K(1 - 4a\varepsilon)$. Let us show that $\kappa_i \leq \frac{\kappa}{1-8a\varepsilon}$. In fact, if the $\sigma_{i,j}$'s are the diagonal values of Σ_i, the Weyl's bound [220] implies that

$$|\sigma_{i,j} - \sigma_{0,j}| \leqslant \|\Sigma_i - \Sigma_0\|_{\max} \leqslant \frac{4a}{\kappa}\varepsilon \qquad \text{for } 1 \leqslant j \leqslant n.$$

So that for $1 \leqslant j < k \leqslant n$, we obtain using $1 - 8a\varepsilon \geqslant 0$:

$$\begin{aligned} |\sigma_{i,k} - \sigma_{i,j}| &\geqslant |\sigma_{0,k} - \sigma_{0,j}| - |\sigma_{i,k} - \sigma_{0,k}| - |\sigma_{i,j} - \sigma_{0,j}| \\ &\geqslant |\sigma_{0,k} - \sigma_{0,j}|(1 - \kappa|\sigma_{i,k} - \sigma_{0,k}| - \kappa|\sigma_{i,j} - \sigma_{0,j}|) \\ &\geqslant |\sigma_{0,j} - \sigma_{0,k}|(1 - 8a\varepsilon) \geqslant 0. \end{aligned}$$

Finally, we get :

$$\kappa_i \leqslant \frac{\kappa}{1-8a\varepsilon}.$$

On the other hand the inequality

$$|\sigma_{i,k} - \sigma_{i,j}| \leqslant |\sigma_{0,k} - \sigma_{0,j}| + |\sigma_{i,k} - \sigma_{0,k}| + |\sigma_{i,j} - \sigma_{0,j}|$$

implies in the same way that above

$$\kappa_i \geqslant \frac{1}{1+8a\varepsilon}\kappa.$$

Next we prove (5.18) for $i+1$. We know $S_i = diag(\Delta_i - Z_i\Sigma_i)$. Since $\varepsilon_i = \max(\kappa_i^2 K_i^2 \|Z_i\|_{\max}, \kappa_i^2 K_i \|\Delta_i\|_{\max})$ and $\kappa_i, K_i \geqslant 1$ then $\|S_i\|_{\max} \leq \dfrac{2}{\kappa_i}\varepsilon_i \leqslant \dfrac{2(1+8a\varepsilon)}{\kappa}2^{1-2^i}\varepsilon$.

It follows :

$$\begin{aligned}\|\Sigma_{i+1}-\Sigma_0\|_{\max} &\leqslant \|S_i\|_{\max}+\|\Sigma_i-\Sigma_0\|_{\max}\\ &\leqslant \frac{2(1+8a\varepsilon)}{\kappa}2^{1-2^i}\varepsilon+(2-2^{2-2^i})\frac{2a}{\kappa}\varepsilon\\ &\leqslant \left(2-2^{1-2^i}\left(2-1\right)\right)\frac{2a}{\kappa}\varepsilon \qquad \text{since } 1+8a\varepsilon=a\\ &\leqslant \left(2-2^{1-2^i}\right)\frac{2a}{\kappa}\varepsilon\end{aligned}$$

But it is eay to see that $2^{1-2^i}\geqslant 2^{2-2^{i+1}}$. Finally we get

$$\|\Sigma_{i+1}-\Sigma_0\|_{\max}\leqslant\left(2-2^{2-2^{i+1}}\right)\frac{2a}{\kappa}\varepsilon.$$

Hence we can also write

$$K_i-\frac{2a}{\kappa_i}\varepsilon\leqslant\|\Sigma_i\|_{\max}-\|\Sigma_{i+1}-\Sigma_i\|_{\max}\leqslant K_{i+1}\leqslant\|\Sigma_i\|_{\max}+\|\Sigma_{i+1}-\Sigma_i\|_{\max}\leqslant K_i+\frac{2a}{\kappa_i}\varepsilon$$

Using more the Weyl's bound we can easily get that

$$\frac{\kappa_i}{1+4a\varepsilon}\leqslant\kappa_{i+1}\leqslant\frac{\kappa_i}{1-4a\varepsilon}.$$

Now we bound $\kappa_{i+1}^2K_{i+1}^2\|Z_{i+1}\|_{\max}$. We have

$$Z_{i+1} = Z_iX_i+Y_iZ_i+Y_i(Z_i+I_n)X_i.$$

Since $\|X_i\|_{\max},\|Y_i\|_{\max}\leq\kappa_i(\|\Delta_i\|_{\max}+K_i\|Z_i\|_{\max})\leqslant\frac{2}{\kappa_iK_i}\varepsilon_i$, we can write

$$\begin{aligned}\kappa_{i+1}^2K_{i+1}^2\|Z_{i+1}\|_{\max} &\leqslant \frac{\kappa_{i+1}^2K_{i+1}^2}{\kappa_i^3K_i^3}4\varepsilon_i^2+\frac{\kappa_{i+1}^2K_{i+1}^2}{\kappa_i^4K_i^4}4\varepsilon_i^3+\frac{\kappa_{i+1}^2K_{i+1}^2}{\kappa_i^2K_i^2}4\varepsilon_i^2\\ &\leqslant 4\left(2+\varepsilon_i\right)\left(\frac{\kappa_{i+1}K_{i+1}}{\kappa_iK_i}\right)^2\varepsilon_i^2\\ &\leqslant 4\left(2+\varepsilon_i\right)\left(\frac{1+2a\varepsilon}{1-4a\varepsilon}\right)^2\varepsilon_i^2\end{aligned}$$

On the other hand

$$\Delta_{i+1} = \Delta_iX_i+Y_i\Delta_i+Y_i(\Delta_i+\Sigma_i)X_i.$$

Hence

$$\begin{aligned}\kappa_{i+1}^2K_{i+1}\|\Delta_{i+1}\|_{\max} &\leqslant \frac{\kappa_{i+1}^2K_{i+1}}{\kappa_i^3K_i^2}4\varepsilon_i^2+\frac{\kappa_{i+1}^2K_{i+1}}{\kappa_i^4K_i^3}4\varepsilon_i^3+\frac{\kappa_{i+1}^2K_{i+1}}{\kappa_i^2K_i}4\varepsilon_i^2\\ &\leqslant 4\left(2+\varepsilon_i\right)\frac{\kappa_{i+1}^2K_{i+1}}{\kappa_i^2K_i}\varepsilon_i^2\\ &\leqslant 4\left(2+\varepsilon_i\right)\frac{1+2a\varepsilon}{(1-4a\varepsilon)^2}\varepsilon_i^2\end{aligned}$$

It follows

$$\begin{aligned}
\varepsilon_{i+1} &\leqslant 4\,(2+\varepsilon)\left(\frac{1+2a\varepsilon}{1-4a\varepsilon}\right)^2\varepsilon_i^2 \\
&\leqslant 8\,(2+\varepsilon)\left(\frac{1-6\varepsilon}{1-12\varepsilon}\right)^2\varepsilon\,2^{1-2^{i+1}}\varepsilon \\
&\leqslant 2^{1-2^{i+1}}\varepsilon \qquad \text{since } 8\,(2+\varepsilon)\left(\frac{1-6\varepsilon}{1-12\varepsilon}\right)^2\varepsilon \leqslant 1 \text{ for } \varepsilon \leqslant 0.033.
\end{aligned}$$

This completes the proof of the two induction hypothesis (5.17–5.18) at order $i+1$. Let $W_i = \prod_{k=0}^{i}(I_n + X_k)$. Since

$$\begin{aligned}
\|X_k\|_{\max} &\leqslant \frac{2}{\kappa_k K_k}\varepsilon_k \\
&\leqslant \frac{2(1+8a\varepsilon)}{\kappa K(1-4a\varepsilon)}\varepsilon 2^{1-2^k} \\
&\leqslant \frac{2}{\kappa K(1-12\varepsilon)}\varepsilon 2^{1-2^k}
\end{aligned}$$

Consequently,

$$\begin{aligned}
\|W_\infty - I_n\|_{\max} &\leqslant \prod_{i\geqslant 0}(1+\frac{2}{\kappa K(1-12\varepsilon)}\varepsilon 2^{1-2^i}) - 1 \\
&\leqslant \frac{4}{\kappa K(1-12\varepsilon)}\varepsilon \quad \text{from Lemma 5.1.1} \\
&\leqslant \frac{0.22}{\kappa K} \qquad \text{since } \varepsilon \leqslant 0.033..
\end{aligned}$$

Hence W_∞ is invertible and $E_0 = E_\infty W_\infty^{-1}$. This implies that E_0 is invertible. Moreover,

$$\begin{aligned}
\|W_i - W_\infty\|_{\max} &\leqslant \|W_i\|_{\max}\left\|1 - \prod_{k\geqslant i+1}(1+\|X_k\|_{\max})\right\|_{\max} \\
&\leqslant (1+\|W_i - I_n\|_{\max})\left\|\prod_{k\geqslant 0}(1+\frac{2}{\kappa K(1-12\varepsilon)}\varepsilon\times 2^{1-2^{k+i+1}}) - 1\right\|_{\max} \\
&\leqslant (1+0.22)\times\frac{4}{\kappa K(1-12\varepsilon)}\times 2^{1-2^{i+1}}\varepsilon \quad \text{from Lemma 5.1.1} \\
&\leqslant \frac{8.1}{\kappa K}\times 2^{1-2^{i+1}}\varepsilon.
\end{aligned}$$

We deduce that

$$\|E_i - E_\infty\|_{\max} \leqslant \frac{8.1}{\kappa K}\times 2^{1-2^{i+1}}\|E_0\|_{\max}\varepsilon.$$

In the same way we show that F_0 is invertible and

$$\|F_i - F_\infty\|_{\max} \leqslant \frac{8.1}{\kappa K}\times 2^{1-2^{i+1}}\|F_0\|_{\max}\varepsilon.$$

Finally

$$\begin{aligned}
\|\Sigma_i - \Sigma_\infty\|_{\max} &\leqslant \sum_{k\geqslant i} \|\Sigma_{k+1} - \Sigma_k\|_{\max} \\
&\leqslant \sum_{k\geqslant i} \frac{2}{\kappa_k^2 K_k}\varepsilon_k \\
&\leqslant \left(\sum_{k\geqslant 0} 2^{-2^k}\right) 2^{1-2^i} \frac{2}{\kappa^2 K(1-12\varepsilon)(1-8\varepsilon)}\varepsilon \\
&\leqslant 0.82 \times 2.25 \times 2^{1-2^i}\frac{\varepsilon}{\kappa K} \qquad \text{since } \sum_{k\geqslant 0} 2^{-2^k} \leqslant 0.82 \text{ and } \varepsilon \leqslant 0.033. \\
&\leqslant 1.85 \times 2^{1-2^i}\varepsilon_0.
\end{aligned}$$

The theorem is proved. □

Proposition 5.1.6. *The complexity of one Newton iteration in Theorem 5.1.5 is in* $\mathcal{O}(n^3)$.

Proof. The computation of all the entries $x_{i,j}$, $y_{i,j}$ of X_i and Y_i by the formulas (5.9–5.13) requires in total $\mathcal{O}(n^2)$ arithmetic operations. The computation of $Z_i, \Delta_i, S_i, E_{i+1}, F_{i+1}$, which requires 6 backward stable matrix multiplications and diagonal matrix operations, has a complexity in $\mathcal{O}(n^3)$. Consequently, the complexity of each iteration is in $\mathcal{O}(n^3)$. □

Remark 5.1.1 – It is possible to generalize this approach to the case where the diagonal matrices are replaced by Jordan matrices.

5.1.4 Newton-like method for two simultaneously diagonalizable matrices.

Let M_1, M_2 be two commuting matrices in $\mathcal{W}_n$, thus M_1 and M_2 are simultaneously diagonalizable. We aim to find $E, F \in \mathrm{GL}_n$ which diagonalize simultaneously M_1, M_2 so that : $FM_kE = \Sigma_k \mid k \in \{1,2\}$, and $\Sigma_1, \Sigma_2 \in \mathcal{D}'_n$. This equivalent to find the numerical solution of $f(E, F, \Sigma_1, \Sigma_2) = 0$ such that $f : (E, F, \Sigma_1, \Sigma_2) \mapsto (FM_1E - \Sigma_1, FM_1E - \Sigma_1)$

We consider as before the perturbations $E + EX$, $F + YF$ and $\Sigma_k + S_k$ of respectively E, F and Σ_k for $k\in \{1,2\}$. Letting $Z_k = \mathrm{FM}_k\, E - \Sigma_k$ for $k = 1, 2$, we have :

$$\begin{aligned}
&(F + \mathrm{YF})M_k(E + \mathrm{EX}) - (\Sigma_k + S_k) \\
&= Z_k - S_k + \Sigma_k X + Y\Sigma_k + Z_k X + \mathrm{YZ}_k + Y(Z_k + \Sigma_k)X
\end{aligned} \tag{5.19}$$

By assuming Z_1, Z_2 are of small norm, the linear system to solve from Equation (5.19) is the following

$$Z_k - S_k + \Sigma_k X + Y\Sigma_k \;=\; 0, \qquad k = 1, 2 \tag{5.20}$$

A solution of (5.20) is given by the following lemma.

Lemma 5.1.7. *Let* $\Sigma_k = \mathrm{diag}(\sigma_1^k, \cdots, \sigma_n^k)$, $Z_k = (z_{i,j}^k)_{1\leq i,j\leq n}$ *be given matrices in* $\mathbb{C}^{n\times n}$ *for* $k \in \{1,2\}$. *Assume that* $\begin{vmatrix} \sigma_j^1 & \sigma_j^2 \\ \sigma_i^1 & \sigma_i^2 \end{vmatrix} \neq 0$ *for* $i \neq j$. *Let* X, Y, *and* S_k *be the matrices defined by*

$$\begin{aligned} x_{i,i} &= 0 && (5.21)\\ x_{i,j} &= \frac{\begin{vmatrix} \sigma_j^1 & z_{i,j}^1 \\ \sigma_j^2 & z_{i,j}^2 \end{vmatrix}}{\begin{vmatrix} \sigma_i^1 & \sigma_j^1 \\ \sigma_i^2 & \sigma_j^2 \end{vmatrix}}, \qquad i \neq j && (5.22)\\ y_{i,i} &= 0 && (5.23)\\ y_{i,j} &= -\frac{\begin{vmatrix} \sigma_i^1 & z_{i,j}^1 \\ \sigma_i^2 & z_{i,j}^2 \end{vmatrix}}{\begin{vmatrix} \sigma_i^1 & \sigma_j^1 \\ \sigma_i^2 & \sigma_j^2 \end{vmatrix}}, \qquad i \neq j && (5.24)\\ S_k &= \mathrm{ddiag}(Z_k), \qquad k = 1,2. && (5.25) \end{aligned}$$

Then we have

$$Z_k - S_k + \Sigma_k X + Y\Sigma_k = 0, \qquad k = 1,2 \tag{5.26}$$

Moreover

$$\|X\|_{\max}, \|Y\|_{\max} \leqslant 2\kappa\varepsilon K \tag{5.27}$$

where $\varepsilon = \max(\|Z_1\|_{\max}, \|Z_2\|_{\max})$, $\kappa = \max\left(1, \max_{i\neq j} \frac{1}{\begin{vmatrix} \sigma_i^1 & \sigma_j^1 \\ \sigma_i^2 & \sigma_j^2 \end{vmatrix}}\right)$, $K = \max(1, \max_{i,k}|\sigma_i^k|)$.

Proof. It is easy to verify that the equation (5.26) implies that for $i \neq j$,

$$\sigma_i^k x_{i,j} + \sigma_j^k y_{i,j} + z_{i,j}^k = 0$$

and that the solution of these equations is given by the formula (5.22), (5.24). Choosing $x_{i,i} = y_{i,i}$=0, we take $S_k = \mathrm{ddiag}(Z_k + \Sigma_k X + Y\Sigma_k) = \mathrm{ddiag}(Z_k)$ since $\Sigma_k X + Y\Sigma_k$ is an off-matrix, to satisfy the equation (5.26). The bounds (5.27) follows easily from (5.22), (5.24). □

Theorem 5.1.8. *Let* E_0, $F_0 \in GL_n$ *and* $\Sigma_{0,k} = \mathrm{diag}(\sigma_{0,1}^k, \ldots, \sigma_{0,n}^k) \in \mathcal{D}'_n$, $k = 1,2$, *be given and let define the sequences for* $i \geqslant 0$ *and* $k = 1,2$ *by :*

$$\begin{aligned} Z_{i,k} &= F_i M_k E_i - \Sigma_{i,k}\\ S_{i,k} &= \mathrm{diag}(Z_{i,k})\\ E_{i+1} &= E_i(I_n + X_i)\\ F_{i+1} &= (I_n + Y_i)F_i\\ \Sigma_{i+1,k} &= \Sigma_{i,k} + S_{i,k}, \end{aligned}$$

where X_i, Y_i *are defined by the formulas (5.21–5.24). Let* $\varepsilon_0 = \max(\|Z_{0,1}\|_{\max}, \|Z_{0,2}\|_{\max})$, $\kappa_0 = \max\left(1, \max_{i\neq j} \frac{1}{\begin{vmatrix}\sigma_{0,i}^1 & \sigma_{0,j}^1\\ \sigma_{0,i}^2 & \sigma_{0,j}^2\end{vmatrix}}\right)$ *and* $K_0 = \max(1, \max_{j,k}|\sigma_{0,j}^k|)$. *Assume that*

$$u := 4\varepsilon_0\kappa_0^2 K_0^3 \leqslant 0.094. \tag{5.28}$$

Then the sequences $(\Sigma_{i,k}, E_i, F_i)_{i\geqslant 0}$*converge quadratically to the solution of* $FM_kE - \Sigma_k$ *for* $k = 1, 2$. *More precisely* E_0 *and* F_0 *are invertible and*

$$\begin{aligned}\|E_i - E_\infty\|_{\max} &\leqslant 1.46 \times 2^{1-2^{i+1}}\|E_0\|_{\max}u\\ \|F_i - F_\infty\|_{\max} &\leqslant 1.46 \times 2^{1-2^{i+1}}\|F_0\|_{\max}u.\end{aligned}$$

Proof. Let us denote for each $i \geqslant 0$,

$$\begin{array}{cccccc}\varepsilon &=& \varepsilon_0 & \varepsilon_i &=& \max(\|Z_{i,1}\|_{\max}, \|Z_{i,2}\|_{\max})\\ \kappa &=& \kappa_0 & \kappa_i &=& \max\left(1,\ \max_{1\leqslant j<k\leqslant n} \frac{1}{\begin{vmatrix}\sigma_{i,j}^1 & \sigma_{i,k}^1\\ \sigma_{i,j}^2 & \sigma_{i,k}^2\end{vmatrix}}\right)\\ K &=& K_0 & K_i &=& \max(1, \max_{j,k}(|\sigma_{i,j}^k|)),\end{array}$$

where $\sigma_{i,1}^k, \ldots, \sigma_{i,n}^k$ are the diagonal entries of $\Sigma_{i,k}$. Let us show by induction on i that

$$\varepsilon_i \leqslant 2^{1-2^i}\varepsilon \tag{5.29}$$

$$\|\Sigma_{i,k} - \Sigma_{0,k}\|_{\max} \leqslant (2 - 2^{2-2^i})\varepsilon \tag{5.30}$$

These inequalities clearly hold for $i = 0$. Assuming that the induction hypothesis holds for a given i and let us prove it for $i+1$. We can notice that $\varepsilon_i \leq 1$. In fact by induction hypothesis, we have $\varepsilon_i \leq 2^{1-2^i}\varepsilon_0$ and from (5.28) $\varepsilon_0 = \frac{u}{4\kappa_0^2 K_0^3} \leq 1$, since $u \leq 1$ and $\kappa_0, K_0 \geq 1$. As $2^{1-2^i} \leq 1,\ \forall i \geq 0$, we have $\varepsilon_i \leq 1$. We first prove that $\|\Sigma_{i+1,k} - \Sigma_{0,k}\|_{\max} \leqslant (2 - 2^{2-2^{i+1}})\varepsilon$ under the assumption $\|\Sigma_{i,k} - \Sigma_{0,k}\|_{\max} \leqslant (2 - 2^{2-2^i})\varepsilon$. To do this, at the first step we show that this implies $K_i \leq K + 2\varepsilon$ and $\kappa_i \leq \dfrac{\kappa}{1 - 8\kappa\varepsilon(K+\varepsilon)}$. Let us prove $K_i \leqslant K + 2\varepsilon$. We have

$$\begin{aligned}K_i := \|\Sigma_i\|_{\max} &\leq \|\Sigma_0\|_{\max} + \|\Sigma_i - \Sigma_0\|_{\max}\\ &\leqslant K + (2 - 2^{2-2^i})\varepsilon\\ &\leqslant K + 2\varepsilon.\end{aligned}$$

Let us show that $\kappa_i \leq \dfrac{\kappa}{1 - 8\kappa\varepsilon(K+\varepsilon)}$. In fact, if the σ_{i,j^k}'s are the diagonal values of Σ_i^k, we have $|\sigma_{i,j}^k - \sigma_{0,j}^k| \leqslant \|\Sigma_{i,k} - \Sigma_{0,k}\|_{\max} \leqslant 2\varepsilon$ for $1 \leqslant j \leqslant n$ and $k = 1, 2$. It follows :

$$\begin{aligned}|\sigma_{i,j}^1\sigma_{i,k}^2 - \sigma_{0,j}^1\sigma_{0,k}^2| &= |\sigma_{i,j}^1\sigma_{i,k}^2 - \sigma_{0,j}^1\sigma_{i,k}^2 + \sigma_{0,j}^1\sigma_{i,k}^2 - \sigma_{0,j}^1\sigma_{0,k}^2|\\ &= |\sigma_{i,k}^2(\sigma_{i,j}^1 - \sigma_{0,j}^1) + \sigma_{0,j}^1(\sigma_{i,k}^2 - \sigma_{0,k}^2)|\\ &\leqslant 2\varepsilon|\sigma_{i,k}^2| + 2\varepsilon|\sigma_{0,j}^1|\\ &\leqslant 2\varepsilon(K + 2\varepsilon) + 2\varepsilon K = 4\varepsilon(K+\varepsilon).\end{aligned}$$

Now,

$$\begin{aligned} |\sigma^1_{i,j}\sigma^2_{i,k} - \sigma^1_{i,k}\sigma^2_{i+1,j}| &\geqslant \\ |\sigma^1_{0,j}\sigma^2_{0,k} - \sigma^1_{0,k}\sigma^2_{0,j}| - |\sigma^1_{0,j}\sigma^2_{0,k} - \sigma^1_{i+1,j}\sigma^2_{i,k}| - |\sigma^1_{i,k}\sigma^2_{i,j} - \sigma^1_{0,k}\sigma^2_{0,j}| &\geqslant \\ |\sigma^1_{0,j}\sigma^2_{0,k} - \sigma^1_{0,k}\sigma^2_{0,j}|(1 - 8k\varepsilon(K+\varepsilon)). & \end{aligned}$$

Finally, we get :

$$\kappa_i \leqslant \frac{\kappa}{1 - 8\kappa\varepsilon(K+\varepsilon)}.$$

To prove (5.30) it is sufficient to write

$$\begin{aligned} \|\Sigma_{i+1,k} - \Sigma_{0,k}\|_{\max} &\leqslant \|S_{i,k}\|_{\max} + \|\Sigma_{i+1,k} - \Sigma_{0,k}\|_{\max} \\ &\leqslant \varepsilon_i + (2 - 2^{2-2^i})\varepsilon \\ &\leqslant (2^{1-2^i} + 2 - 2^{2-2^i})\varepsilon \leqslant (2 - 2^{2-2^{i+1}})\varepsilon. \end{aligned}$$

Let us prove (5.29). Since we have

$$Z_{i+1,k} = Z_{i,k}X_i + Y_iZ_{i,k} + Y_i(Z_{i,k} + \Sigma_{i,k})X_i.$$

we deduce

$$\begin{aligned} \|Z_{i+1,k}\|_{\max} &\leqslant 2\varepsilon_i^2\kappa_iK_i + 2\varepsilon_i^2\kappa_iK_i + 4\varepsilon_i^2\kappa_i^2K_i^2(\varepsilon_i + K_i) \\ &\leqslant 4\varepsilon_i^2\kappa_i^2K_i + 4\varepsilon_i^2\kappa_i^2K_i^2(1 + K_i) \quad \text{since } \varepsilon_i \leqslant 1 \text{and } \kappa_i \geqslant 1 \\ &\leqslant 3 \times 4\varepsilon_i^2\kappa_i^2K_i^3 = 12\varepsilon_i^2\kappa_i^2K_i^3 \qquad \text{since } K_i \geqslant 1. \end{aligned}$$

It follows

$$\begin{aligned} \varepsilon_{i+1} &\leqslant \frac{12\kappa^2(K+2\varepsilon)^3}{(1 - 8\kappa\varepsilon(K+\varepsilon))^2}\varepsilon_i^2 \leqslant \frac{12\varepsilon\kappa^2(K+2\varepsilon)^3}{(1 - 8\kappa\varepsilon(K+\varepsilon))^2}2^{2-2^{i+1}}\varepsilon \\ &\leqslant 3\frac{\left(1+\frac{u}{2}\right)^3}{\left(1 - 2u\left(1+\frac{u}{4}\right)\right)^2}u2^{2-2^{i+1}}\varepsilon \quad \text{since} \quad \frac{\varepsilon}{K} \leqslant \frac{u}{4}, \kappa\varepsilon \leqslant \frac{u}{4} \\ &\leqslant 2^{1-2^{i+1}}\varepsilon \quad \text{since} \quad 3\frac{\left(1+\frac{u}{2}\right)^3}{\left(1 - 2u\left(1+\frac{u}{4}\right)\right)^2} \leqslant 2^{-1} \text{ for } u \leqslant 0.094. \end{aligned}$$

Let $W_i = \prod_{k=0}^{i}(I_n + X_k)$. Since

$$\begin{aligned} \|X_l\|_{\max} &\leqslant 2\kappa_lK_l\varepsilon_l \\ &\leqslant 2\frac{\kappa}{1 - 8\kappa\varepsilon(K+\varepsilon)}(K+2\varepsilon)\varepsilon 2^{1-2^l} \\ &\leqslant \frac{\left(1+\frac{u}{2}\right)u}{2\left(1 - 2u\left(1+\frac{u}{4}\right)\right)}2^{1-2^l} \\ &\leqslant 0.65 \times 2^{1-2^l}u \quad \text{since } u \leqslant 0.094. \end{aligned}$$

Consequently,

$$\begin{aligned}\|W_\infty - I_n\|_{\max} &\leqslant \prod_{i\geqslant 0}(1+0.65\times 2^{1-2^i}u)-1\\ &\leqslant 1.3u \quad \text{from Lemma 5.1.1}\\ &\leqslant 1.3\times 0.094 = 0.1222\end{aligned}$$

Hence W_∞ is invertible and $E_0 = E_\infty W_\infty^{-1}$. This implies that E_0 is invertible. Moreover,

$$\begin{aligned}\|W_i - W_\infty\|_{\max} &\leqslant \|W_i\|_{\max}\left\|1-\prod_{k\geqslant i+1}(1+\|X_k\|_{\max})\right\|_{\max}\\ &\leqslant (1+\|W_i-I_n\|_{\max})\left\|\prod_{k\geqslant 0}(1+0.059\times 2^{1-2^{k+i+1}})-1\right\|_{\max}\\ &\leqslant (1+0.1222)\times 1.3\times 2^{1-2^{i+1}}u\\ &\leqslant 1.46\times 2^{1-2^{i+1}}u.\end{aligned}$$

We deduce that

$$\|E_i - E_\infty\|_{\max} \leqslant 1.46\times 2^{1-2^{i+1}}\|E_0\|_{\max}u.$$

In the same way we show that F_0 is invertible and

$$\|F_i - F_\infty\|_{\max} \leqslant 1.46\times 2^{1-2^{i+1}}\|F_0\|_{\max}u.$$

The theorem is proved. □

5.1.5 Convergence of a pencil of simultaneously diagonalizable matrices.

In this subsection we present two strategies to solve the system (4.3) of a pencil of commuting matrices $(M_i)_{1\leq i\leq s}$ in $\mathcal{W}_n$. The first strategy is trivial and consists of finding the common diagonalizers E and F of the pencil by numerically solving one of the systems $(FE-I_n, FM_1E-\Sigma_1) = 0$ or $(FM_1E-\Sigma_1, FM_2E-\Sigma_1) = 0$ using Theorem 5.1.5 or Theorem 5.1.8. Next we deduce the remaining diagonal matrices Σ_i using the formulas

$$\Sigma_{i,k} = \frac{E(:,k)^* M_i E(:,k)}{E(:,k)^* E(:,k)} \qquad 1\leqslant k\leqslant n, \quad 2 \text{ or } 3 \leqslant i \leqslant s,$$

where $E(:,k)$ is the *k-th* column in E.
In this strategy we use that a diagonalizer of one or two matrices of the pencil can diagonalize the other matrices of the pencil. We note that, in general, we don't have this property for simultaneously diagonalizable matrices, where, for instance, it is posssible to find a diagonalizer of M_1 which is not a common diagonalizer for the other matrices of the pencil. Nevertheless, this property holds here since we suppose that the matrices M_i have simple eigenvalues.

Another strategy is to find a "good" linear combination of the M_i's. This is based on Lemma 5.1.9 and Theorem 5.1.10.

Lemma 5.1.9. *Let us suppose that the M_i commute pairwise and they are linearly independent i.e. $\sum_{i=1}^{s} a_i M_i = 0 \Rightarrow a_i = 0, i = 1 : s$. Let $E \in \mathrm{GL}_n$ and $\Sigma_i \in \mathcal{D}'_n$ be such that*

$$E^{-1} M_i E - \Sigma_i \;=\; 0, \quad i = 1 : s.$$

Let $S \in \mathbb{C}^{n \times s}$ and the column i of S is the diagonal of Σ_i. Let $\sigma = (\sigma_1, \ldots, \sigma_n)$ and $\Sigma = \mathrm{diag}(\sigma)$. Then the matrix S has a full rank and $\alpha = (S^*S)^{-1}S^*\sigma$ satisfies

$$\sum_{i=1}^{s} \alpha_i E^{-1} M_i E - \Sigma \;=\; 0.$$

Proof. Since the matrices M_i are simultaneously diagonalizable there exists E be such that $E^{-1}M_iE - \Sigma_i = 0$. The condition

$$\sum_{i=1}^{s} \alpha_i \Sigma_i - \Sigma \;=\; 0$$

is written as $S\alpha = \sigma$ where $S \in \mathbb{C}^{n \times s}$. The assumption $\sum_{i=1}^{s} a_i M_i = 0 \Rightarrow a_i = 0, i = 1 : s$ implies that the matrix has a full rank. Consequently,

$$\alpha \;=\; (S^*S)^{-1}S^*\sigma.$$

The lemma follows. □

Theorem 5.1.10. *Let $M_1, \ldots, M_p \in \mathbb{C}^{n \times n}$ be p simultaneously diagonalizable matrices and verify the assumption of linearly independent. Let us consider matrices E_0, F_0 and $\Sigma_{0,i} = \mathrm{diag}(F_0 M E_0)$, $i = 1 : p$. Let us define the matrix $S \in \mathbb{C}^{n \times p}$ in which the column i is the diagonal of $\Sigma_{0,i}$. Let $\sigma = \left(1, e^{\frac{2i\pi}{n}}, \ldots, e^{\frac{2i(n-1)\pi}{n}}\right)$, $\Sigma = \mathrm{diag}(\sigma)$ and $\alpha = (S^*S)^{-1}S^*\sigma$. We consider the system*

$$\begin{pmatrix} EF - I_n \\ FME - \Sigma \end{pmatrix} \;=\; 0 \tag{5.31}$$

where $M = \sum_{i=1}^{p} \alpha_i M_i$. If

$$n^2 \max(\|Z_0\|_{\max}, \|\Delta_0\|_{\max}) \;\leqslant\; 16 \times 0.033$$

then (F_0, E_0, Σ) satisfies the condition (5.16) of Theorem 5.1.5.

Proof. In this case the quantity κ defined in the Theorem 5.1.5 is equal to

$$\begin{aligned} \kappa \;&=\; \frac{1}{2\,|\sin\left(\frac{\pi}{n}\right)|} \\ &\leqslant\; \frac{n}{4} \quad \text{since } |\sin\left(\frac{\pi}{n}\right)| \geqslant \frac{2}{n} \text{ for } n \geqslant 2. \end{aligned}$$

Since $K_0 = 1$ we get

$$\varepsilon_0 = \max(\kappa_0^2 K_0^2 \|Z_0\|_{\max}, \kappa_0^2 K_0 \|\Delta_0\|_{\max}) \leq \frac{n^2}{16} \max(\|Z_0\|_{\max}, \|\Delta_0\|_{\max}).$$

The condition

$$\max(\|Z_0\|_{\max}, \|\Delta_0\|_{\max}) \leq 0.033\, \frac{16}{n^2},$$

gives the result. □

5.1.6 Numerical illustration

We use a JULIA implementation of the Newton sequences in the numerical experiments. The experimentation has been done on a Dell Windows desktop with 8 GB memory and Intel 2.3 GHz CPU. We use the Julia package `ArbNumerics` for the computation in high precision.

5.1.7 Simulation

In this section we apply the Newton iterations presented in Theorem 5.1.5 (resp. Theorem 5.1.8) on examples of diagonalizable matrices (resp. of two simultaneously diagonalizable matrices). We validate experimentally the sufficiency of the condition established in Theorem 5.1.5 (resp. Theorem 5.1.8) to have a quadratic sequence (Tables 5.1, 5.2, 5.6, and 5.7). On the other hand, as this condition is sufficient but not necessary, we show through some other examples how this Newton sequence starting from an initial point which is not verifying this condition could converge quadratically (Tables 5.3, 5.4, 5.8, and 5.9). We note that the the computation in the aforementioned tables is done in high precision. Nevertheless, we test also the two Newton-type sequences using machine precision (Tables 5.5 and 5.10) and this to show that these sequences have the same numerical behavior of a classical Newton method, i.e., if the solution is in the neighborhood of the initial point the Newton-type iterations will converge towards this solution with a few number of iterations and the residual error obtained at the end is in double precision.

This allows us to have an heuristic estimation on the numerical dependency of the Newton sequences from this condition to converge. Furthermore, these examples reveal the possibility of achieving computation in such problem with high precision. For example, in the case of a diagonalizable matrix of simple eigenvalues, we can compute its eigenvalues using one of the solvers which works with a double precision. Then we take this point as an initial point for the Newton sequence of Theorem 5.1.5 in order to increase the precision. Hereafter, we give some details about the tests : *Test-1* for Theorem 5.1.5 and *Test-2* for Theorem 5.1.8, considered in this section.

Test-1. Let $\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$, $M = E\Sigma E^{-1} + 10^{-e}A$, where $e \in \{3, 6\}$. The matrices E, Σ, and $A \in \mathbb{K}^{n\times n}$ are chosen randomly following standard normal distributions such that E is invertible, Σ is diagonal with n different diagonal entries and A is a random square matrix obeying normal distribution of size n and Frobenius norm equal to 1. Since M is a small perturbation of $E\Sigma E^{-1}$, more precisely $\|M - E\Sigma E^{-1}\| = 10^{-e}$, M is a diagonalizable matrix of simple eigenvalues. Herein, we apply the Newton iteration of Theorem 5.1.5 on M with initial point $E_0 = E$, $F_0 = E^{-1}$ and $\Sigma_0 = \Sigma$. The residual error reported in this test at iteration k is given by :

$$\text{err}_{res} = \max(\|F_kE_k - I_n\|_{\max}, \|F_kME_k - \Sigma_k\|_{\max}).$$

Test-2. Let $\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$, $M_1 = F^{-1}\Sigma_1E^{-1}$, $M_2 = F^{-1}\Sigma_2E^{-1}$, where E, F, Σ_1 and $\Sigma_2 \in \mathbb{K}^{n\times n}$ are randomly sampled according to standard normal distributions, such that E and F are invertible, Σ_1 and Σ_2 are diagonal with n different diagonal entries. The Newton iteration in Theorem 5.1.8 is applied on M_1 and M_2 with initial point E_0, F_0, $\Sigma_{0,1}$ and $\Sigma_{0,2}$, such that these matrices are obtained by applying a small perturbation on respectively E, F, Σ_1 and Σ_2 as follows :
$E_0 = E + 10^{-e}A$, $F_0 = F + 10^{-e}B$, $\Sigma_{0,1} = \Sigma_1 + 10^{-e}C$, $\Sigma_{0,2} = \Sigma_2 + 10^{-e}D$, where $e \in \{3, 6\}$, A and B (resp. C and D) are random square matrices (resp. random diagonal matrices

with different diagonal entries) of size n and Frobenius norm equal to 1, with entries in $\mathbb{K}$ following standard normal distributions. The residual error reported in this test at iteration k is given by :

$$\text{err}_{res} = \max(\|F_k M_1 E_k - \Sigma_{k,1}\|_{\max}, \|F_k M_2 E_k - \Sigma_{k,2}\|_{\max}).$$

We notice that the condition established in Theorem 5.1.5 (resp. Theorem 5.1.8) is reached in *Test-1* (resp. *Test-2*) for matrices of size 10 with order of perturbation equal to 10^{-6}, and we can see in Tables 5.1, 5.2, 5.6, and 5.7 that the Newton sequences with initial point verifying the condition in the associated theorem converge quadratically. We can notice also that by increasing the perturbation up to 10^{-3} (the initial point does not verify the condition in the associated theorem), the Newton sequences converge quadratically for different sizes of matrices $n = 10,\ 50,\ 100$ (see Tables 5.3, 5.4, 5.8, and 5.9). Moreover, we can notice in Table 5.5 the Newton-type iteration of Theorem 5.1.5 applied in double precision converges with a few number of iterations ~ 5 and the final residual error measured with the Frobenius norm is of order machine precision $\sim 10^{-14}$ and it is of the same order obtained by the standard Julia method `eigen` to compute the eigen decomposition. The same remarks are valid for Table 5.10 where the Newton-type sequence of Theorem 5.1.8 needs, in double precision, a few iterations to converges towards the solution given by using the Frobenius norm a residual error of order machine precision.

Table 5.1 – The computational results throughout 7 iterations of an example of implementation of *Test-1* with $\mathbb{K} = \mathbb{R},\ n = 10$ and $e = 6$ in precision 1024.

Iteration	$\varepsilon := \max(\kappa_0^2 K_0^2 \|Z_0\|_{\max}, \kappa_0^2 K_0 \|\Delta_0\|_{\max}) \leq 0.033$	err_{res}
1	0.00131	$9.33e-6$
2	$2.39e-8$	$1.06e-10$
3	$1.68e-18$	$7.49e-21$
4	$2.93e-38$	$1.31e-40$
5	$4.21e-78$	$1.87e-80$
6	$1.17e-157$	$5.24e-160$
7	$4.16e-288$	$6.20e-293$

Table 5.2 – The computational results throughout 7 iterations of an example of implementation of *Test-1* with $\mathbb{K} = \mathbb{C},\ n = 10$ and $e = 6$ in precision 1024.

Iteration	$\varepsilon := \max(\kappa_0^2 K_0^2 \|Z_0\|_{\max}, \kappa_0^2 K_0 \|\Delta_0\|_{\max}) \leq 0.033$	err_{res}
1	0.02581	$2.76e-4$
2	$3.49e-6$	$2.33e-8$
3	$9.51e-14$	$6.34e-16$
4	$5.31e-29$	$3.54e-31$
5	$1.96e-59$	$1.31e-61$
6	$3.02e-120$	$2.01e-122$
7	$4.58e-242$	$3.05e-244$

Table 5.3 – The residual error throughout 7 iterations given by the implementation of *Test-1* with $\mathbb{K} = \mathbb{R}, e = 3$ and $n = 10, 50, 100$ in precision 1024.

Iteration	$n = 10$	$n = 50$	$n = 100$
1	$8.57e-3$	$7.93e-2$	$3.22e-2$
2	$1.91e-4$	$5.76e-2$	$1.38e-2$
3	$1.58e-8$	$6.19e-3$	$6.12e-4$
4	$4.79e-16$	$8.74e-5$	$5.42e-7$
5	$3.56e-31$	$1.31e-8$	$3.83e-13$
6	$1.39e-61$	$2.39e-16$	$1.80e-25$
7	$1.91e-122$	$7.03e-32$	$3.81e-50$

Table 5.4 – The residual error throughout 7 iterations given by the implementation of *Test-1* with $\mathbb{K} = \mathbb{C}, e = 3$ and $n = 10, 50, 100$ in precision 1024.

Iteration	$n = 10$	$n = 50$	$n = 100$
1	$8.84e-3$	$9.75e-2$	$1.61e-2$
2	$8.59e-6$	$6.39e-5$	$1.03e-4$
3	$3.91e-11$	$3.99e-9$	$4.68e-9$
4	$9.87e-22$	$1.87e-17$	$3.13e-17$
5	$7.60e-43$	$4.42e-34$	$8.84e-34$
6	$5.14e-85$	$2.50e-67$	$9.45e-67$
7	$2.64e-169$	$8.28e-134$	$1.05e-132$

Table 5.5 – The residual error throughout 5 iterations given by the implementation of *Test-1* with $\mathbb{K} = \mathbb{R}, e = 3$ and $n = 10, 20, 30,$ in double precision.

Iteration	$n = 10$	$n = 20$	$n = 30$
1	$4.78e-3$	$1.01e-2$	$1.01e-2$
2	$4.71e-3$	$2.55e-3$	$1.14e-3$
3	$2.29e-5$	$1.97e-5$	$4.08e-7$
4	$1.43e-9$	$2.36e-10$	$2.26e-13$
5	$4.06e-15$	$1.23e-14$	$5.04e-14$
$\|M - E_{\texttt{eigen}}\Sigma_{\texttt{eigen}}E_{\texttt{eigen}}^{-1}\|$	$9.49e-15$	$2.83e-14$	$7.45e-14$
$\|M - E_{\texttt{newton}}\Sigma_{\texttt{newton}}E_{\texttt{newton}}^{-1}\|$	$2.96e-15$	$1.01e-14$	$3.42e-14$

5.1.8 Cauchy matrix

In this section we present an example for a Cauchy matrix of size $n = 13$ of entries $a_{i,j} = \frac{1}{i+j}$, $\forall 1 \leq i, j \leq 13$, that illustrates how the Newton-type iteration can be used to increase the accuracy of the eigenvalues. We take the eigen decomposition given by the standard JULIA method `eigen` from the package `LinearAlgebra` as an initial point of Newton sequences in Theorem 5.1.5 with 5 iterations. The computation is done with the precision 1024 using `ArbNumerics` package. The initial point given by `eigen` is in double precision. It is converted to the precision 1024 using `ArbNumerics` package, in order to apply Newtons iterations with this precision of 1024 bits. In Table 5.11 we report the eigenvalues given by `eigen` ($\sigma_{\texttt{eigen}}$)

Table 5.6 – The computational results throughout 7 iterations of an example of implementation of *Test-2* with $\mathbb{K} = \mathbb{R},\ n = 10$ and $e = 6$ in precision 1024.

Iteration	$4\kappa^2 K^3 \varepsilon \leq 0.094$	err_{res}
1	$7.65e-2$	$6.72e-6$
2	$1.73e-7$	$1.52e-11$
3	$5.58e-18$	$4.90e-22$
4	$5.49e-39$	$4.82e-43$
5	$3.10e-81$	$2.73e-85$
6	$2.28e-165$	$2.01e-169$
7	$2.20e-279$	$1.94e-283$

Table 5.7 – The computational results throughout 7 iterations of an example of implementation of *Test-2* with $\mathbb{K} = \mathbb{C},\ n = 10$ and $e = 6$ in precision 1024.

Iteration	$4\kappa^2 K^3 \varepsilon \leq 0.094$	err_{res}
1	$6.86e-3$	$9.16e-6$
2	$7.14e-9$	$9.53e-12$
3	$9.51e-21$	$1.26e-23$
4	$6.69e-44$	$8.92e-47$
5	$3.77e-90$	$5.04e-93$
6	$2.59e-182$	$3.45e-185$
7	$1.65e-281$	$2.20e-284$

and the eigenvalues rounded to the double precision given by Newton-type sequence ($\sigma_{\texttt{newton}}$) initialized with `eigen`. We also report the relative error $\frac{|\sigma_{\texttt{newton}}-\sigma_{\texttt{eigen}}|}{\sigma_{\texttt{newton}}}$ in order to show the refinement amount realized by the Newton method. As we can see the matrix of this example is ill-conditioned (Cauchy matrices are in general ill-conditioned). There is a cluster of eigenvalues nearby zero. The accuracy enhancement obtained by applying Newton-type iterations can be clearly seen in Table 5.11, in particular for the first four smallest eigenvalues. For instance, the smallest eigenvalue returned by `eigen` is of order 10^{-17} close to the second smallest eigenvalues of order 10^{-16}. Newton-type method shows that the smallest eigenvalue of the order 10^{-19} yields a large relative error ~ 39.33. This also shows that all the eigenvalues are well-separated.

5.1.9 Sub-matrix iterations

It is possible to adapt the proposed method, taking into account the condition of the eigenvalue σ_i given by the quantity

$$\kappa(\sigma_i) = \max_{i \neq j} \left(1, \frac{1}{|\sigma_i - \sigma_j|}\right)$$

Theoretical results imply that the computation of clusters of eigenvalues is ill-conditioned. However, one can apply Theorem 3 on sub-matrices to improve the well-conditioned eigenvalues. We denote

$$\delta \ = \ \sqrt{\frac{K\|\Delta_0\|_{\max}}{0.033}}$$

Table 5.8 – The residual error throughout 7 iterations given by the implementation of *Test-2* with $\mathbb{K} = \mathbb{R}, e = 3$ and $n = 10, 50, 100$ in precision 1024.

Iteration	$n = 10$	$n = 50$	$n = 100$
1	$2.91e-2$	$4.57e-3$	$1.01e-2$
2	$7.97e-5$	$1.03e-6$	$1.31e-6$
3	$4.21e-9$	$1.69e-11$	$3.71e-11$
4	$1.07e-16$	$2.42e-23$	$1.23e-22$
5	$3.92e-33$	$1.18e-44$	$1.46e-43$
6	$2.63e-64$	$1.02e-89$	$1.67e-86$
7	$1.71e-128$	$3.20e-177$	$9.01e-172$

Table 5.9 – The residual error throughout 7 iterations given by the implementation of *Test-2* with $\mathbb{K} = \mathbb{C}, e = 3$ and $n = 10, 50, 100$ in precision 1024.

Iteration	$n = 10$	$n = 50$	$n = 100$
1	$7.33e-3$	$3.14e-3$	$5.52e-3$
2	$3.49e-6$	$7.48e-7$	$1.35e-6$
3	$2.91e-12$	$1.11e-13$	$1.19e-13$
4	$2.04e-24$	$2.54e-27$	$1.68e-27$
5	$8.23e-49$	$3.04e-54$	$2.19e-54$
6	$1.88e-97$	$3.41e-108$	$1.50e-108$
7	$1.31e-194$	$1.91e-215$	$4.53e-216$

Table 5.10 – The residual error throughout 5 iterations given by the implementation of *Test-2* with $\mathbb{K} = \mathbb{R}, e = 3$ and $n = 10, 20, 30$, in double precision.

Iteration	$n = 10$	$n = 20$	$n = 30$
1	$2.71e-3$	$1.21e-2$	$4.64e-3$
2	$1.36e-6$	$4.91e-6$	$2.24e-6$
3	$1.39e-12$	$2.57e-11$	$4.74e-11$
4	$6.16e-15$	$8.97e-14$	$1.55e-13$
5	$7.04e-15$	$8.09e-14$	$1.53e-13$
$\max(\|M_1 - E\Sigma_1 E^{-1}\|, \|M_2 - E\Sigma_2 E^{-1}\|)$	$3.74e-15$	$4.13e-14$	$8.21e-14$

and p the index such that $\Sigma = \begin{pmatrix} \Sigma_p & \\ & \Sigma_{n-p} \end{pmatrix}$, $\Sigma_p = \operatorname{diag}(\sigma_1, \ldots, \sigma_p)$, $\Sigma_{n-p} = \operatorname{diag}(\sigma_{p+1}, \ldots, \sigma_n)$ and $|\sigma_i - \sigma_j| > \delta$ for all $1 \leqslant i \leq p$ and $i < j \leqslant n$. We adapt Newton iteration to the block associated with the well-conditioned eigenvalues by defining the matrices

Table 5.11 – The relative error between $\sigma_{\texttt{eigen}}$ from the method `eigen` and $\sigma_{\texttt{newton}}$ from the Newton-type method for the Cauchy matrix $(\frac{1}{i+j})_{1\leq i,j\leq 13}$.

Eigenvalue	$\sigma_{\texttt{eigen}}$	$\sigma_{\texttt{newton}}$	$\frac{\lvert\sigma_{\texttt{newton}}-\sigma_{\texttt{eigen}}\rvert}{\sigma_{\texttt{newton}}}$
1	2.4030587641505818e-17	5.958203769841865e-19	39.33
2	1.8824087522342697e-16	1.7156976132548192e-16	0.09716
3	2.3152722725223998e-14	2.3178576801522747e-14	0.00111
4	1.9513972147589434e-12	1.951356013568409e-12	$2.11e-5$
5	1.1466969172503778e-10	1.1466967568738049e-10	$1.39e-7$
6	4.991788233415145e-9	4.991788235245136e-9	$3.66e-10$
7	1.6668681228080362e-7	1.666868122813953e-7	$3.54e-12$
8	4.360227301207107e-6	4.360227301206033e-6	$2.46e-13$
9	9.040674871074817e-5	9.040674871075823e-5	$1.11e-13$
10	0.0014925044272821445	0.0014925044272821172	$1.83e-14$
11	0.01955788569925287	0.01955788569925287	$4.81e-17$
12	0.19958813407010345	0.19958813407010337	$4.64e-16$
13	1.3693334145989837	1.3693334145989824	$9.98e-16$

X, Y and S as follows :

$$\begin{aligned} x_{i,i} &= 0 \\ x_{i,j} &= \begin{cases} \dfrac{-\delta_{i,j}+z_{i,j}\sigma_j}{\sigma_i-\sigma j} & \text{if} \quad |\sigma_i-\sigma_j|>\delta \\ 0 & \text{otherwise} \end{cases} \\ Y &= -Z-X \\ S &= \operatorname{diag}(-\Delta+Z\Sigma). \end{aligned}$$

Table 5.12 (resp. Table 5.13) shows the residual error err_{res} as in *Test-1* for the Cauchy matrix of size 200 (resp. the Rosser matrix of size 256 [177]) by applying the aforementioned sequences, the initial point is given by the Julia method `eigen`. The computation is done in precision 1024.

Table 5.12 – The residual error throughout 6 iterations with the Cauchy matrix of size 200.

Iteration	$p=12,\ \delta=4.51e-7$	$p=5,\ \delta=4.51e-7$
1	$2.45e-15$	$2.35e-15$
2	$9.63e-26$	$3.75e-29$
3	$1.56e-36$	$1.21e-53$
4	$1.54e-45$	$1.81e-83$
5	$1.15e-54$	$3.49e-110$
6	$5.08e-64$	$8.67e-137$

Table 5.13 – The residual error throughout 6 iterations with the Rosser matrix of size 256.

Iteration	$p = 11, \delta = 1.11e-3$	$p = 5, \delta = 1.11e-3$
1	$7.15e-12$	$1.65e-12$
2	$7.18e-20$	$7.18e-20$
3	$1.42e-40$	$1.81e-41$
4	$1.73e-53$	$1.56e-85$
5	$7.17e-66$	$1.75e-119$
6	$8.79e-79$	$8.11e-153$

Summary of this section. Taking a Newton approach towards systems of equations describing the simultaneous diagonalization problem of diagonalizable matrices, leads us to new algorithmic insights. We exhibit a Newton-type method without solving a linear system at each step as is the case of a classical Newton method. The numerical experiments corroborate the quadratic convergence predicted by the theoretical analysis.

5.2 Riemannian conjugate gradient algorithm for approximate simultaneous diagonalization of matrices

In the previous section, we studied the certification problem for the convergence of a pencil of simultaneously diagonalizable matrices. In this section, we consider the second part of the simultaneous matrix diagonalization problem, where the pencil of matrices is not necessarily simultaneously diagonalizable. Thus we aim to approximate it locally by a pencil of matrices which is simultaneously diagonalizable. We study this problem from an optimization point of view by taking into account the geometric constraints of the optimization problem and we present a Riemannian conjugate gradient algorithm. We consider the approximation problem over the real field $\mathbb{R}$. Given a general pencil $M = [M_1, \ldots, M_s]$ of real square matrices, such that $M_k \in \mathbb{R}^{n\times n}, \forall k \in \{1, \ldots, s\}$, we aim to compute a simultaneously diagonalizable pencil that approximates M i.e. to find two invertible matrices E and F such that FM_kE^t is the most diagonal, $\forall k \in \{1, \ldots, s\}$.
For shortness we refer to the Approximate Simultaneous Diagonalization of matrices problem as **ASD** problem.

Notation. In this section we consider the real field. We denote by $\mathcal{D}_n^*$ the group of $n \times n$ nonsingular diagonal matrices, and by $\mathcal{P}_n$ the group of $n \times n$ permutation matrices.

5.2.1 Cost function

The first cost function that comes in mind is the one which minimizes the Frobenius norm of the off-diagonal entries of each matrix in the pencil M :

$$\tilde{f}(E,F) = \frac{1}{2}\sum_{k=1}^{s} \|FM_kE^t - \operatorname{ddiag}(FM_kE^t)\|^2 = \frac{1}{2}\sum_{k=1}^{s} \|\operatorname{off}(FM_kE^t)\|^2,\ (E,F) \in \mathrm{GL}_n \times \mathrm{GL}_n.$$

However, as discussed in [25], this function is not invariant by diagonal scaling i.e. $\tilde{f}(\Sigma_1 E, \Sigma_2 F) \neq \tilde{f}(E,F)$ with $\Sigma_1, \Sigma_2 \in \mathcal{D}_n^*$, which might produce in practice undesirable

degenerate solutions, since ASD is an optimization problem on GL_n. For this reason, we choose to conduct the optimization method using the following cost function

$$f(E,F) = \frac{1}{2}\sum_{k=1}^{s} \|M_k - F^{-1}\,\mathrm{ddiag}(FM_kE^t)E^{-t}\|^2,\ (E,F) \in \mathrm{GL}_n \times \mathrm{GL}_n, \tag{5.32}$$

which generalizes the one proposed in [7] for a pencil of symmetric matrices, and it is invariant to the diagonal scaling.
Herein, approximate simultaneous diagonalization of matrices problem (ASD) consists of minimizing the cost function (5.32) :

$$\min_{(E,F)\in \mathrm{GL}_n\times\mathrm{GL}_n} f(E,F) = \min_{(E,F)\in \mathrm{GL}_n\times\mathrm{GL}_n} \frac{1}{2}\sum_{k=1}^{s} \|M_k - F^{-1}\,\mathrm{ddiag}(FM_kE^t)E^{-t}\|^2. \tag{5.33}$$

5.2.2 Oblique geometric constraints

It is easy to notice that if $(E,F) \in \mathrm{GL}_n \times \mathrm{GL}_n$ is a solution of (5.33), then $(E\Sigma_1P_1, P_2\Sigma_2F)$, $\forall P_1, P_2 \in \mathcal{P}_n,\ \forall(\Sigma_1,\Sigma_2) \in \mathcal{D}_n^* \times \mathcal{D}_n^*$ is also a solution. In practice, the permutation has no impact on the solution process. Contrarily, the diagonal scaling can conduct to degenerate solutions (we can construct sequences of non-singular diagonal matrices which converge towards singular matrices). Hence, to avoid degenerate solutions, some additional constraints must be added to the constraint set. In this regard, there exists in the literature, various approaches (see. [1, 25, 26, 205, 12]). From the existing possibilities, we choose to fix the norm of the rows of E and F with the oblique constraint i.e. the rows of E and F are of unit Frobenius norm [1, 25]. This means that we seek E and F in the oblique manifold $\mathcal{M}_n^o$ previously defined in Section 3.5.5.
Finally, we formulate the ASD problem as a Riemannian least-squares problem on $\mathcal{M}_n^o \times \mathcal{M}_n^o$ as follows :

$$\min_{(E,F)\in \mathcal{M}_n^o\times\mathcal{M}_n^o} f(E,F) = \min_{(E,F)\in \mathcal{M}_n^o\times\mathcal{M}_n^o} \frac{1}{2}\sum_{k=1}^{s} \|M_k - F^{-1}\,\mathrm{ddiag}(FM_kE^t)E^{-t}\|^2. \tag{ASD}$$

We choose to equip the oblique manifold $\mathcal{M}_n^o$ with the right-invariant metric inherited from GL_n (see Sections 3.5.4 and 3.5.5). In Definition 5.2.1 we define the metric that we equip with the Cartesian product of oblique manifolds $\mathcal{M}_n^o \times \mathcal{M}_n^o$.

Definition 5.2.1. For $(E,F) \in \mathcal{M}_n^o \times \mathcal{M}_n^o$, $\xi = (\xi^E, \xi^F)$, $\eta = (\eta^E, \eta^F) \in T_E\mathcal{M}_n^o \times T_F\mathcal{M}_n^o \simeq T_{(E,F)}\mathcal{M}_n^o \times \mathcal{M}_n^o$, then

$$\langle \xi, \eta\rangle^r_{(E,F)} = \langle (\xi^E, \xi^F), (\eta^E, \eta^F)\rangle^r_{(E,F)} = \langle \xi^E, \eta^E\rangle^r_E + \langle \xi^F, \eta^F\rangle^r_F,$$

where $\langle ., .\rangle^r_E$ (resp. $\langle ., .\rangle^r_F$) is as in Section 3.5.4.

5.2.3 The proposed Riemannian conjugate gradient method

We introduce a Riemannian conjugate gradient method with Armijo backtracking line-search step (RCG) (Section 3.4.1) to solve the (ASD). The algorithm (RCG) for ASD problem in pseudo-code is given as follows :

Require: initial iterate $X_1 := (E_1, F_1) \in \mathcal{M}_n^o \times \mathcal{M}_n^o$, tangent vector $\eta_0 := (\eta^{E_0}, \eta^{F_0}) = (0, 0)$.

1: **for** $i = 1, 2, \ldots$ **do**

2: Compute the gradient $\xi_i := \operatorname{grad}_{ob} f(X_i)$;

3: Compute a conjugate direction $\eta_i := -\xi_i + \beta_i \mathcal{T}_{X_{i-1} \to X_i}(\eta_{i-1})$;

4: Perform Armijo backtracking to find the smallest integer $m \geq 0$ such that

$$f(X_i) - f(R_{X_i}(0.5^m \eta_i)) \geq -0.1 \times 0.5^m \langle \xi_i, \eta_i \rangle_{X_i}^r;$$

5: Compute the next new point $X_{i+1} := R_{X_i}(0.5^m \eta_i)$;

6: **end for**

Algorithm 5.1 – Geometric conjugate gradient for ASD.

Hereafter, the different steps of the algorithm will be explained in more details.
In step 2 of Algorithm 5.1, the computation of the Riemannian gradient $\operatorname{grad}_{ob} f(X)$ of f at $X := (E, F) \in \mathcal{M}_n^o \times \mathcal{M}_n^o$ with respect to the right-invariant metric relies on some matrix trace properties that we recall in the following lemma [*] :

Lemma 5.2.1. *Let A, B, $C \in \mathbb{R}^{n \times n}$, then :*

$$\begin{aligned}
\operatorname{tr}(ABC) &= \operatorname{tr}(BCA) = \operatorname{tr}(CAB) \text{ (invariant by cyclic permutation)} \\
\operatorname{tr}(A^t) &= \operatorname{tr}(A) \\
\operatorname{tr}(aA) &= a \operatorname{tr}(A),\ a \in \mathbb{R} \\
\operatorname{tr}(A \operatorname{ddiag}(B)) &= \operatorname{tr}(\operatorname{ddiag}(A) B) = \operatorname{tr}(\operatorname{ddiag}(A) \operatorname{ddiag}(B)) \\
\langle ., . \rangle &\text{ is the canonical inner product in } \mathbb{R}^{n \times n} (\text{i.e.} \langle A, B \rangle = \operatorname{tr}(A^t B)).
\end{aligned}$$

To compute this gradient, we first write f as a trace function by using Lemma 5.2.1 :

$$f(E, F) = \sum_{k=1}^{s} f_k(E, F)$$

such that,

$$\begin{aligned}
f_k(E, F) &= \tfrac{1}{2} \| M_k - F^{-1} \operatorname{ddiag}(F M_k E^t) E^{-t} \|^2 \\
&= \tfrac{1}{2} \operatorname{tr}((M_k - F^{-1} \operatorname{ddiag}(F M_k E^t) E^{-t})^t (M_k - F^{-1} \operatorname{ddiag}(F M_k E^t) E^{-t})) \\
&= \tfrac{1}{2} \operatorname{tr}(M_k M_k^t - 2 E^{-1} \operatorname{ddiag}(F M_k E^t) F^{-t} F^{-1} \operatorname{ddiag}(F M_k E^t) E^{-t}).
\end{aligned}$$

Let $f^F := f(., F) : \mathcal{M}_n^o \to \mathbb{R},\ E \mapsto f(E, F)$; $f^E := f(E, .) : \mathcal{M}_n^o \to \mathbb{R},\ F \mapsto f(E, F)$.

We have,

$$\operatorname{grad}_{ob} f(E, F) = (\operatorname{grad}_{ob} f^F(E), \operatorname{grad}_{ob} f^E(F)) = \Big(\sum_{k=1}^{s} \operatorname{grad}_{ob} f_k^F(E), \sum_{k=1}^{s} \operatorname{grad}_{ob} f_k^E(F) \Big),$$

where $\operatorname{grad}_{ob} f_k^F(E)$ and $\operatorname{grad}_{ob} f_k^E(F)$ denote respectively the Riemannian gradient of $f_k^F = f_k(., F) : \mathcal{M}_n^o \to \mathbb{R},\ E \mapsto f_k(E, F)$ at $E \in \mathcal{M}_n^o$ and of $f_k^E = f_k(E, .) : \mathcal{M}_n^o \to \mathbb{R},\ F \mapsto$

∗. The computation of the gradient is derived by a technical analog of the gradient when the matrices are considered symmetric, see for instance [8, 1, 25].

$f_k(E,F)$ at $F \in \mathcal{M}_n^o$, equipped with the right-invariant metric.
We show how to compute $\operatorname{grad}_{ob} f_k^F(E)$ then the same procedure can be applied to compute $\operatorname{grad}_{ob} f_k^E(F)$.
To compute the Riemannian gradient of f_k^F at E in $\mathcal{M}_n^o$ endowed with the right-invariant metric, we recall from Table 3.1 that :

$$\operatorname{grad}_{ob} f_k^F(E) = P_{ob,r}^E(\operatorname{grad} f_k^F(E)),$$

where $P_{ob,r}^E$ is the orthogonal projection map into $T_E\mathcal{M}_n^o$ according to the right-invariant metric already been given in the same table 3.1, and $\operatorname{grad} f_k^F(E)$ is the Riemannian gradient of f_k^F at $E \in \mathrm{GL}_n$ equipped with the right-invariant metric given by (see Section 3.5.4) :

$$\operatorname{grad} f_k^F(E) = \operatorname{grad}_{\mathrm{Euc}} f_k^F(E) E^t E,$$

such that $\operatorname{grad}_{\mathrm{Euc}} f_k^F(E)$ is the Euclidean gradient of f_k^F at $E \in \mathrm{GL}_n$.
Next, to compute $\operatorname{grad}_{\mathrm{Euc}} f_k^F(E)$ we compute the time derivative of f_k^F ($\dot{f}_k^F$) in terms of the time derivative of E ($\dot{E} \in T_E\mathrm{GL}_n$) by using the trace identities in Lemma 5.2.1 in order to obtain

$$\dot{f}_k^F(E) = \operatorname{tr}(X\dot{E}),$$

on the other hand we have that $\dot{f}_k^F(E) = \operatorname{tr}((\operatorname{grad}_{\mathrm{Euc}} f_k^F(E))^t \dot{E})$, herein we deduce $\operatorname{grad}_{\mathrm{Euc}} f_k^F(E) = X^t$.
Finally, the formulas of $\operatorname{grad}_{\mathrm{Euc}} f_k^F(E)$ and $\operatorname{grad}_{\mathrm{Euc}} f_k^E(F)$ are given by :

$$\begin{aligned}
\operatorname{grad}_{\mathrm{Euc}} f_k^F(E) &= [Q_k(E,F)\operatorname{ddiag}(\mathrm{FM}_k E^t) - \operatorname{ddiag}(Q_k(E,F))\mathrm{FM}_k E^t]E^{-t},\\
\operatorname{grad}_{\mathrm{Euc}} f_k^E(F) &= [Q_k(E,F)^t\operatorname{ddiag}(\mathrm{FM}_k E^t) - \operatorname{ddiag}(Q_k(E,F))\mathrm{EM}_k^t F^t]F^{-t},
\end{aligned}$$

with $Q_k(E,F) = (EE^t)^{-1}(\mathrm{EM}_k^t F^t - \operatorname{ddiag}(\mathrm{FM}_k E^t))(FF^t)^{-1}$.

For shortness, ξ^{E_i} (resp. ξ^{Fi}) denotes hereafter $\operatorname{grad}_{ob} f^{F_i}(E_i)$ (resp. $\operatorname{grad}_{ob} f^{E_i}(F_i)$).
In step 3, $\eta_i := (\eta^{E_i}, \eta^{F_i}) \in T_{E_i}\mathcal{M}_n^o \times T_{F_i}\mathcal{M}_n^o$ such that :

$$\eta_i := -\xi_i + \beta_i \mathcal{T}_{X_{i-1}\to X_i}(\eta_{i-1}) = -(\xi^{E_i}, \xi^{F_i}) + \beta_i(\mathcal{T}^{ob,r}_{E_{i-1}\to E_i}(\eta^{E_{i-1}}), \mathcal{T}^{ob,r}_{F_{i-1}\to F_i}(\eta^{F_{i-1}})),$$

where $\mathcal{T}^{ob,r}_{E_{i-1}\to E_i}$ (resp. $\mathcal{T}^{ob,r}_{F_{i-1}\to F_i}$) is the vector transport on $\mathcal{M}_n^o$ equipped with the right-invariant metric from Table 3.1. We use the Polak-Ribière formula in (3.8) for β_i :

$$\beta_i = \frac{\langle \xi_i, \xi_i - \mathcal{T}_{X_{i-1}\to X_i}(\xi_{i-1})\rangle^r_{X_i}}{\langle \xi_i, \xi_i\rangle^r_{X_i}}$$

In step 5, we define the new point $X_{i+1} := (E_{i+1}, F_{i+1}) \in \mathcal{M}_n^o \times \mathcal{M}_n^o$ such that :

$$X_{i+1} := R_{X_i}(0.5^m \eta_i) = R_{(E_i,F_i)}(0.5^m\eta^{E_i}, 0.5^m\eta^{F_i}) = (R^{ob,r}_{E_i}(0.5^m\eta^{E_i}), R^{ob,r}_{F_i}(0.5^m\eta^{F_i})),$$

where $R^{ob,r}_{E_i}$ (resp. $R^{ob,r}_{F_i}$) is the retraction on $\mathcal{M}_n^o$ for $E_i \in \mathcal{M}_n^o$ (resp. for $F_i \in \mathcal{M}_n^o$) according to the right-invariant metric from Table 3.1.

The algorithm stops when $\|\operatorname{grad}_{ob} f(X_i)\| \leq \text{tolerence}$.

5.3 Simultaneous matrix diagonalization and tensor decomposition

In this section, we show the connection between simultaneous matrix diagonalization and tensor decomposition. For real tensor rank approximation of real multilinear tensors of dimension 3 and size (n_1, n_2, n_3) with approximation rank $r \geq \max(n_1, n_2)$, we develop an *alternate optimization algorithm*, based on approximate simultaneous diagonalization of matrices.

We denote by $\mathfrak{T}(n_1, n_2, n_3)$ the set of real multilinear tensors of dimension 3 and size (n_1, n_2, n_3) i.e. $\mathfrak{T}(n_1, n_2, n_3) := \mathbb{R}^{n_1} \otimes \mathbb{R}^{n_2} \otimes \mathbb{R}^{n_3}$.
A pencil $M = [M_1, \ldots, M_s]$ of matrices $M_i \in \mathbb{R}^{n \times n}$ can be seen as a tensor $\mathbf{M} \in \mathfrak{T}(n, n, s)$ where the slice $\mathbf{M}_{[:,:,i]}$ is the matrix M_i.
Let $M = [M_1, \ldots, M_s]$ be a pencil of simultaneously diagonalizable matrices $M_i \in \mathbb{R}^{n \times n}$, i.e. there exists matrices $E, F \in \mathrm{GL}_n$ with

$$M_k = F \operatorname{diag}(\Sigma_{[:,k]}) E^t = F \Sigma_{[k]} E^t \tag{5.34}$$

where $\Sigma = [\sigma_{i,j}] \in \mathbb{R}^{n \times s}$ and $\Sigma_{[:,k]}$ is the k^{th} column of Σ and $\Sigma_{[k]} = \operatorname{diag}(\Sigma_{[:,k]})$.

Proposition 5.3.1. *A pencil of simultaneously diagonalizable matrices $M = [M_1, \ldots, M_s]$ such that $M_i \in \mathbb{R}^{n \times n}$, $\forall 1 \leq i \leq s$, is in correspondence with a tensor $\mathbf{M} \in \mathfrak{T}(n, n, s)$ of rank $\leqslant n$.*

Proof. To a family $M = [M_1, \ldots, M_s]$ of $n \times n$ matrices, corresponds the tensor $\mathbf{M} \in \mathfrak{T}(n, n, s)$ such that $\mathbf{M}_{[:,:,i]} = M_i$. If M is simultaneously diagonalizable as in (5.34), then the corresponding tensor $\mathbf{M}$ can be written as

$$\mathbf{M} = \sum_{k=1}^{n} F_k \otimes E_k \otimes \Sigma_{[k,:]} \tag{5.35}$$

where F_k (resp. E_k) is the k^{th} column of F, (resp. E) and $\Sigma_{[k,:]} = [\sigma_{k,1}, \ldots, \sigma_{k,s}]$ is the k^{th} row of Σ. Conversely, if a tensor $\mathbf{M}$ can be written as (5.35), then the pencil $M = [M_{[:,:,i]}]$ is simultaneously diagonalizable with $M_k = F \Sigma_{[k]} E^t$ where $E = [E_1, \ldots, E_n]$, $F = [F_1, \ldots, F_n]$ and $\Sigma_{[k]} = \operatorname{diag}(\Sigma_{1,k}, \ldots, \Sigma_{n,k})$. □

The matrices E, F, Σ involved in (5.34) (resp. (5.35)) are called the decomposition factors of M (resp. $\mathbf{M}$).
Now, suppose that we have a tensor $\mathbf{M} \in \mathfrak{T}(n_1, n_2, n_3)$ that we aim to approximate to a tensor of rank $r \geq \max(n_1, n_2)$, i.e. find three factor matrices $A,\ B,\ C$ respectively in $\mathbb{R}^{n_1 \times r}$, $\mathbb{R}^{n_2 \times r}$ and $\mathbb{R}^{n_3 \times r}$, that minimize the following non-linear least-squares function :

$$f(A, B, C) = \tfrac{1}{2} \Big\| \mathbf{M} - \sum_{i=1}^{r} A[:, i] \otimes B[:, i] \otimes C[:, i] \Big\|^2.$$

Our approach, based on alternate optimization method, to find such factor matrices A, B and C can be summarized as follows :

1. Let M be the pencil of n_3 matrices of size $n_1 \times n_2$ associated to $\mathbf{M}$.

2. Extend the pencil M to a pencil $\tilde{M} = [\tilde{M}_1, \ldots, \tilde{M}_{n_3}]$ with $\tilde{M}_i$ of size $r \times r$ such that the block of the first n_1 rows with n_2 columns in $\tilde{M}_i$ is equal to M_i in M, and the remain entries of $\tilde{M}_i$ are sampled randomly, for instance, according to a Gaussian distribution.
3. Approximate the pencil $\tilde{M}$ locally by a pencil of simultaneously diagonalizable matrices $\tilde{M}'$ by using Algorithm 5.1 in Section 5.2, i.e. find $(E, F) \in \mathcal{M}_r^o \times \mathcal{M}_r^o$ that solves locally :

$$\min_{(E,F)\in\mathcal{M}_r^o\times\mathcal{M}_r^o} f_1(E,F) = \min_{(E,F)\in\mathcal{M}_r^o\times\mathcal{M}_r^o} \frac{1}{2}\sum_{k=1}^{n_3} \|\tilde{M}_k - F^{-1}\,\mathrm{ddiag}(F\tilde{M}_kE^t)E^{-t}\|^2.$$

4. Recall from (5.35) that the tensor associated to the pencil of simultaneously diagonalizable matrices $\tilde{M}'$ can be written as a tensor rank-r decomposition with the factors E, F and Σ. This means, that the tensor associated to $\tilde{M}$ is approximated locally by the tensor rank-r decomposition given by $\tilde{M}'$.
5. Since our goal is to find a rank-r approximation of the tensor $\mathbf{M}$, such that its slices $\mathbf{M}[:,:,k]$ are fixed blocks in the matrices $\tilde{M}_k$ of the pencil $\tilde{M}$, the distance function $\frac{1}{2}\sum_{k=1}^{n_3} \|\tilde{M}_k - F^{-1}\,\mathrm{ddiag}(F\tilde{M}_kE^t)E^{-t}\|^2$ can be reduced, by fixing the obtained matrices E, F and optimizing all the entries of the matrices $\tilde{M}_k$ except of those in the fixed blocks. This leads us to the following linear least-squares problem :

$$\begin{aligned} &\min_{\tilde{M}} f_2(\tilde{M}) = \min_{\tilde{M}} \frac{1}{2}\sum_{k=1}^{n_3} \|\tilde{M}_k - F^{-1}\,\mathrm{ddiag}(F\tilde{M}_kE^t)E^{-t}\|^2, \\ &\text{s.t. } \tilde{M}_k[1:n_1, 1:n_2] = M_k, \text{ for } k \in \{1,\ldots,n_3\}. \end{aligned} \tag{5.36}$$

   In summary, we minimize the function

$$\begin{aligned} f(E,F,\tilde{M}) = &\frac{1}{2}\sum_{k=1}^{n_3} \|\tilde{M}_k - F^{-1}\,\mathrm{ddiag}(F\tilde{M}_kE^t)E^{-t}\|^2 \\ &\text{s.t. } (E,F) \in \mathcal{M}_r^o \times \mathcal{M}_r^o, \\ &\quad \tilde{M}_k[1:n_1, 1:n_2] = M_k, \text{ for } k \in \{1,\ldots,n_3\}, \end{aligned} \tag{5.37}$$

   by *alternating* between two optimization minimization problems. The first updates E and F by minimizing f_1 in step 3 using the Riemannian conjugate gradient iteration from Section 5.2, and the second updates the entries which don't belong to the fixed blocks in $\tilde{M}_k$ of the pencil $\tilde{M}$ by solving the linear least-squares problem with f_2 in step 5.
6. Hence, we present an iterative algorithm such that at each iteration the alternate optimization method from step 5 is applied. This is an *alternate optimization algorithm* (AO). The algorithm stops when $f(E,F,\tilde{M}) \leq \mathrm{tolerance}$, or when the number of iterations reaches a maximum number $N_{\max}$.
7. At the end, when the algorithm stops, an approximated rank-r decomposition is obtained for the tensor associated to a pencil $\tilde{M}$, from which we can extract an approximated rank-r decomposition for the targeted tensor $\mathbf{M}$.

The Alternate Optimization algorithm (AO) is given in pseudo-code in Algorithm 5.2.
We detail the linear least-squares problem in (5.36) and step 6 of Algorithm 5.2. It can be solved

as follows :
For each k in $\{1, \ldots, n_3\}$, we consider the linear least-squares problem

$$\min_{\tilde{M}_k \in \mathbb{R}^{r\times r}} \frac{1}{2}\|\tilde{M}_k - F^{-1}\,\mathrm{ddiag}(F\tilde{M}_k E^t)E^{-t}\|^2, \text{ s.t. } \tilde{M}_k[1:n_1, 1:n_2] = M_k.$$

Let $H := \tilde{M}_k - F^{-1}\,\mathrm{ddiag}(F\tilde{M}_k E^t)E^{-t} \in \mathbb{R}^{r\times r}$, let $h_{i,j}$ be the entry of H in the *i-th* row and the *j-th* column. We have $h_{i,j} = \langle e_{i,j}, H\rangle$, where $(e_{i,j})_{1\leq i,j\leq r}$ is the canonical basis of $\mathbb{R}^{r\times r}$. By developing the calculus using the trace properties in Lemma 5.2.1 in order to isolate our targeted variable $\tilde{M}_k$, we find :

$$h_{i,j} = \langle e_{i,j} - F^t\,\mathrm{ddiag}(E^{-1}[j,:] \otimes F^{-1}[i,:])E, \tilde{M}_k\rangle.$$

Hence, minimizing the Frobenius norm of $h_{i,j}$ is equivalent to minimizing the Frobenius norm of $(\mathrm{vec}(e_{i,j} - F^t\,\mathrm{ddiag}(E^{-1}[j,:] \otimes F^{-1}[i,:])E))^t\,\mathrm{vec}(\tilde{M}_k)$. This leads us to the following system :

$$GX = 0,$$

where $G = [(\mathrm{vec}(e_{i,j} - F^t\,\mathrm{ddiag}(E^{-1}[j,:] \otimes F^{-1}[i,:])E))^t] \in \mathbb{R}^{r^2\times r^2}$, $X = \mathrm{vec}(\tilde{M}_k) \in \mathbb{R}^{r^2}$. Recall that we have n_1n_2 fixed entries in $\tilde{M}_k$ equal to the n_1n_2 entries of M_k, thus we write the system $GX = 0$ as $G = [G_1, G_2]$, $X = [X_1, X_2]$, such that $X_2 \in \mathbb{R}^{n_1n_2}$ represents the fixed known entries in $\tilde{M}_k$ and $G_2 \in \mathbb{R}^{r^2\times n_1n_2}$ contains the corresponding columns to X_2 in G. Similarly, $X_1 \in \mathbb{R}^{r^2-n_1n_2}$ represents the vector of the unknown entries of $\tilde{M}_k$ and $G_1 \in \mathbb{R}^{r^2\times(r^2-n_1n_2)}$ contains the corresponding columns to X_2 in G. Hence, the system to solve becomes :

$$G_1X_1 = Y, \tag{5.38}$$

where $Y := -G_2X_2$. Consequently, we can write

$$X_1 = G_1^\dagger X_2,$$

where $G_1^\dagger$ denotes the pseudo-inverse of G_1. Note that if G_1 is of full column rank then X_1 is the least-squares solution of the overdetermined system (5.38) with $G_1^\dagger = (G_1^tG_1)^{-1}G_1^t$.

For step 8 in Algorithm 5.2 the three factors A, B and C are computed by simply taking the block

$$\begin{aligned}&(\tilde{M}_k - F_{\mathrm{end}}^{-1}\,\mathrm{ddiag}(F_{\mathrm{end}}\tilde{M}_kE_{\mathrm{end}}^t)E_{\mathrm{end}}^{-t})[1:n_1, 1:n_2]\\ =&M_k - (F_{\mathrm{end}}^{-1}\,\mathrm{ddiag}(F_{\mathrm{end}}\tilde{M}_kE_{\mathrm{end}}^t)E_{\mathrm{end}}^{-t})[1:n_1, 1:n_2], \text{ since } \tilde{M}_k[1:n_1, 1:n_2] = M_k,\end{aligned}$$

where

$$\begin{aligned}&(F_{\mathrm{end}}^{-1}\,\mathrm{ddiag}(F_{\mathrm{end}}\tilde{M}_kE_{\mathrm{end}}^t)E_{\mathrm{end}}^{-t})[1:n_1, 1:n_2]\\ =&F_{\mathrm{end}}^{-1}[1:n_1, 1:r]\,\mathrm{ddiag}(F_{\mathrm{end}}\tilde{M}_kE_{\mathrm{end}}^t)E_{\mathrm{end}}^{-t}[1:r, 1:n_2].\end{aligned}$$

Thus the pencil M is approximated by the pencil of simultaneously diagonalizable matrices $[F_{\mathrm{end}}^{-1}[1:n_1, 1:r]\,\mathrm{ddiag}(F_{\mathrm{end}}\tilde{M}_kE_{\mathrm{end}}^t)E_{\mathrm{end}}^{-t}[1:r, 1:n_2]]_{1\leq k\leq n_3}$ which correspond to a rank-r decomposition with A, B and C given by (see Proposition 5.3.1) :

$$\begin{aligned}&A = F_{\mathrm{end}}^{-1}[1:n_1, 1:r];\\ &B = E_{\mathrm{end}}^{-t}[1:n_2, 1:r];\\ &C = \Sigma^t, \text{ where } \Sigma \text{ is of size } r\times n_3, \text{ such that } \Sigma[:,k] = \mathrm{diag}(F_{\mathrm{end}}\tilde{M}_{\mathrm{end},k}E_{\mathrm{end}}^t).\end{aligned}$$

Herein, the tensor associated to the pencil M is approximated by the rank-r decompostion given by the three factor matrices A, B and C.

Remark 5.3.1 – If the tensor to approximate is of size (n, n, s) and the approximation rank r is equal to n, there is no need to have the extension step in the Algorithm 5.2, and the algorithm is reduced simply to Algorithm 5.1 for approximate simultaneous diagonalization of matrices.

Require: $\mathbf{M} \in \mathfrak{T}(n_1, n_2, n_3)$, approximation rank $r \geq \max(n_1, n_2)$.

1: Compute M the pencil associated to $\mathbf{M}$;

2: Extend M to the pencil $\tilde{M}$ of n_3 matrices, such that $\tilde{M}_k[1 : n_1, 1 : n_2] = M_k$ for $k \in \{1, \ldots, n_3\}$;

3: Set initial iterate $(E_0, F_0, \tilde{M}_0 := \tilde{M})$, with $(E_0, F_0) \in \mathcal{M}_r^o \times \mathcal{M}_r^o$.

4: **for** $i = 1, 2, \ldots$ **do**

5: Obtain (E_i, F_i) by applying Algorithm 5.1 initialized by (E_{i-1}, F_{i-1}) to solve :

$$\min_{(E,F)\in\mathcal{M}_r^o\times\mathcal{M}_r^o} f_1(E, F) = \min_{(E,F)\in\mathcal{M}_r^o\times\mathcal{M}_r^o} \frac{1}{2}\sum_{k=1}^{n_3} \|\tilde{M}_{i-1,k} - F^{-1}\,\mathrm{ddiag}(F\tilde{M}_{i-1,k}E^t)E^{-t}\|^2.$$

6: Fix (E_i, F_i) and update $\tilde{M}_{i-1}$ to $\tilde{M}_i$ by solving :

$$\begin{aligned} &\min_{\tilde{M}} f_2(\tilde{M}) = \min_{\tilde{M}} \frac{1}{2}\sum_{k=1}^{n_3} \|\tilde{M}_k - F_i^{-1}\,\mathrm{ddiag}(F_i\tilde{M}_kE_i^t)E_i^{-t}\|^2, \\ &\text{s.t. } \tilde{M}_k[1 : n_1, 1 : n_2] = M_k, \text{ for } k \in \{1, \ldots, n_3\}. \end{aligned} \tag{5.39}$$

7: **end for**

8: Extract factor matrices $A \in \mathbb{R}^{n_1\times r}$, $B \in \mathbb{R}^{n_2\times r}$, $C \in \mathbb{R}^{n_3\times r}$, such that :

$$\begin{aligned} &A = F_{\mathrm{end}}^{-1}[1 : n_1, 1 : r]; \\ &B = E_{\mathrm{end}}^{-t}[1 : n_2, 1 : r]; \\ &C = \Sigma^t, \text{ where } \Sigma \text{ is of size } r \times n_3, \text{ such that } \Sigma[:, k] = \mathrm{diag}(F_{\mathrm{end}}\tilde{M}_{\mathrm{end},k}E_{\mathrm{end}}^t). \end{aligned}$$

Algorithm 5.2 – Alternate optimization algorithm (AO) for tensor rank approximation of tensors in $\mathfrak{T}(n_1, n_2, n_3)$ with $r \geq \max(n_1, n_2)$.

Example 5.3.1 – The operator of matrix multiplication of square matrices of size 2×2 can be represented by the homogeneous polynomial $T(x, y, z) = x_1y_1z_1 + x_1y_3z_3 + x_2y_1z_2 + x_2y_3z_4 + x_3y_2z_1 + x_3y_4z_3 + x_4y_2z_2 + x_4y_4z_4$ of degree 3 in three variables $x = (x_1, ..., x_4), y = (y_1, ..., y_4)$ and $z = (z_1, ..., z_4)$. Let $T \in \mathfrak{T}(4, 4, 4)$ be the associated tensor to the aforementioned polynomial. By taking all the monomials of degree three that we can construct with x_i, y_j and z_k, for $i, j, k \in \{1, \ldots, 4\}$ and considering the lexicographic ordering on the monomials indexing the rows, columns and slices of T, the tensor T is as follows :

$$T[1,:,:] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$T[2,:,:] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$T[3,:,:] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$T[4,:,:] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The tensor T is of rank 7 (see. [221]), whereas the generic rank is 6. Since its rank is higher than the generic rank, the tensor T has many rank-7 decompositions. We apply the alternate optimization algorithm (Algorithm 5.2) to find a rank-7 decomposition of T. Starting with an initial point that gives an initial error

$$\mathrm{err}_0 := \frac{1}{2}\|T - \sum_{i=1}^{7} A_0[:,i] \otimes B_0[:,i] \otimes C_0[:,i]\|^2 = 412.63,$$

we found a rank decomposition after running the algorithm for 800 iterations that took around 90 seconds. The error at the final iteration is

$$\mathrm{err}_{\mathrm{end}} := \frac{1}{2}\|T - \sum_{i=1}^{7} A_{\mathrm{end}}[:,i] \otimes B_{\mathrm{end}}[:,i] \otimes C_{\mathrm{end}}[:,i]\|^2 = 3.19e-12.$$

The three factor matrices $A,\ B,\ C \in \mathbb{R}^{4\times 7}$ given by the algorithm are as follows :

$$A = \begin{pmatrix} -3.29804 & -4.1045 & 3.19274 & -0.867293 & 0.22113 & 0.0994379 & -1.09832 \\ -2.23989 & -1.90246 & 0.718265 & -0.195113 & -0.408636 & -0.183755 & -0.745938 \\ 0.364624 & 1.03971 & -0.352981 & 1.5467 & -0.394356 & 0.112363 & -1.24109 \\ 0.247625 & -0.872426 & -0.0794047 & 0.347959 & 0.728746 & -0.207641 & -0.842896 \end{pmatrix},$$

$$B = \begin{pmatrix} -0.0807727 & -1.12551 & 1.16343 & 0.0966848 & -0.774754 & -0.846987 & 0.00372952 \\ 2.56926 & -0.723095 & -1.77348 & -0.147383 & -0.808044 & -0.883381 & -0.118618 \\ 0.0715004 & -0.74756 & 0.65238 & 0.874531 & 0.685632 & -0.474938 & 0.0337356 \\ -2.27374 & 3.29779 & -0.994464 & -1.3331 & 0.715093 & -0.495345 & -1.07291 \end{pmatrix},$$

$$C = \begin{pmatrix} -0.0537698 & 0.0635183 & 0.174347 & 0.737993 & -0.237901 & -1.15977 & 0.800897 \\ 0.239012 & 0.270613 & -0.256709 & 0.39936 & 1.05748 & 1.70766 & 0.4334 \\ 0.0515551 & 0.118177 & -0.167164 & 0.0232047 & -0.00748016 & -0.760829 & 0.5254 \\ -0.229165 & -0.162557 & 0.246134 & 0.012557 & 0.0332504 & 1.12025 & 0.284317 \end{pmatrix}.$$

We note that Algorithm 5.2 can give other rank-7 approximations depending on the initial point, as it is a local optimization method. Nevertheless, we choose to present in this example the case where the initial point conduct to an *exact* rank-7 decomposition (by exact we mean the $\mathrm{err}_{\mathrm{end}}$ is very reduced, for instance, $\mathrm{err}_{\mathrm{end}} \leq 1.1e^{-10}$), since we knows already that this tensor has an exact rank-7 decomposition and thus it is more interesting to find an exact rank-7 decomposition than an approximated rank-7 decomposition.

5.4 Practical session

We present a simple example of a random rank-6 three-dimensional tensor of size $(3, 3, 5)$ normally distributed in order to show the implementation of the method described in Section 5.3, available in the Julia package TensorDec.jl [†].

```
[55]: # Run the file that contains the functions.
      include("least_square.jl")
```

```
[55]: example (generic function with 1 method)
```

```
[54]: # Tensor of order three of size (n1, n2, n3), the approximation rank is r.
      n1=3; n2=3; n3=5; r=6;
```

```
[30]: # Generate a random tensor of size (n1, n2, n3) and rank r, the entries are␣
       ↪normally-distributed random numbers of type Float64.
      T1, A1, B1, C1 = tensor_rank_r(n1,n2,n3,r);
      # A1, B1, C1 are the three factor matrices used to build the tensor T1.
```

```
[31]: # The tensor T1:
      T1
```

```
[31]: 3×3×5 Array{Float64,3}:
      [:, :, 1] =
       -2.83325  -0.10325   1.11907
        1.23147   1.13445  -0.00525748
       -2.72041  -1.31434  -0.0792543

      [:, :, 2] =
       -1.13283  -0.712515   3.64971
        1.88834   1.23159   -1.5367
        5.25158  -2.28315   -3.16988

      [:, :, 3] =
       -4.95478     0.949384   5.70849
        4.04154     1.47325   -1.57491
       -0.00942829  0.767305  -1.02144

      [:, :, 4] =
        8.64859   0.876996   -6.18629
       -3.16332   0.0220391   2.21234
       -1.15905  -1.66203     2.09874

      [:, :, 5] =
       -8.08281   -2.0019     5.06823
        3.40111    0.955546  -2.83084
       -0.751114  -1.57149   -0.639213
```

```
[32]: # Apply a perturbation of order 1.e-3.
      T_perturb = randn(n1,n2,n3)
      T = T1 + 1.e-3*(T_perturb/norm(T_perturb));
```

†. https://gitlab.inria.fr/AlgebraicGeometricModeling/TensorDec.jl

[33]:
```
# Take the pencil of the frontal slices of T:
pencil=[T[:,:,i] for i in 1:5];
```

[34]:
```
# Take the matrix that contains the extension sizes
SIZE=[6 6 6 6 6 ; 6 6 6 6 6];

# Extend the pencil 'pencil' of 5 matrices of size (3, 3) to a pencil␣
↪'pencil_ext' of 5 matrices of size (6, 6).
pencil_ext = extend(pencil,SIZE);

# Show the first matrix in pencil:
pencil[1]
```

[34]:
```
3×3 Array{Float64,2}:
 -2.83342  -0.103067   1.11897
  1.23152   1.13458   -0.00520429
 -2.72041  -1.31432   -0.0791214
```

[35]:
```
# Show the first matrix in pencil_ext:
pencil_ext[1]
```

[35]:
```
6×6 Array{Float64,2}:
 -2.83342   -0.103067    1.11897     0.901066  0.0743145  0.980506
  1.23152    1.13458    -0.00520429  0.232798  0.484185   0.900788
 -2.72041   -1.31432    -0.0791214   0.184807  0.423952   0.266923
  0.159625   0.779979    0.0873955   0.409319  0.88752    0.356534
  0.754062   0.414404    0.516783    0.175066  0.327537   0.316228
  0.327431   0.00241824  0.0210144   0.534145  0.428405   0.592499
```

[47]:
```
# Take initial two matrices E and F:
E = randn(6,6); F=randn(6,6);

# The approximated decompostion of rank 6 given by pencil_ext, E and F is
A0, B0, C0, T0=factors(pencil_ext, E, F,n1,n2,n3,r);

# Error between tensor to approximate T and T0 the tensor of rank 6 with factor␣
↪matrices A0, B0 and C0:
print("Initial error: ", round(0.5*norm(T-T0)^2,digits=2))
```

```
Initial error: 10510.02
```

The error that we show for the algorithm alternate is:

$$\frac{1}{2}\sum_{k=1}^{n_3} \|\text{pencil\_ext}[k] - F^{-1}\text{ddiag}(F\text{pencil\_ext}[k]E^t)E^{-t}\|^2.$$

The Julia function that computes this error is *obj*.

[75]:
```
# The initial error of the algorithm alternate:
print("Initial error of the function alternate:",␣
 ↪round(obj(pencil_ext,E,inv(E),F,inv(F)), digits = 2))
```

```
Initial error of the function alternate:55879.53
```

[65]:
```
# Apply 20 iterations of the alternate algorithm and show the first three and␣
 ↪the last three error given by respectively the
# approximate simultaneous step 'err_simdiag_i' and the linear least-squares␣
 ↪step 'err_least_i'.
pencil_ext_end, E_end, F_end = alternate(pencil_ext,pencil,E,F,20, 1.e-2);
```

```
err_simdiag_1:6.675755
err_least_1:0.819885
err_simdiag_2:0.536113
err_least_2:0.419789
err_simdiag_3:0.356233
err_least_3:0.314931
⋮
err_simdiag_18:0.057583
err_least_18:0.055109
err_simdiag_19:0.05278
err_least_19:0.050682
err_simdiag_20:0.04864
err_least_20:0.04679
```

[66]:
```
# Take the three factor matrices given by pencil_end, E_end, F_end for rank 6␣
 ↪approximation of the tensor T:
A_end, B_end, C_end, T_end = factors(pencil_ext_end, E_end, F_end,n1,n2,n3,r);
```

[73]:
```
# Error between tensor to approximate T and T0 the tensor of rank 6 with factor␣
 ↪matrices A0, B0 and C0:
print("Initial error between T and T0: ", round(0.
 ↪5*norm(T-T0)^2,digits=2),"\n","\n")

# Error between tensor to approximate T and T_end the tensor of rank 6 with␣
 ↪factor matrices A_end, B_end and C_end:
print("Final error between T and T_end: ", round(0.5*norm(T-T_end)^2,digits=4))
```

```
Initial error between T and T0: 10510.02
Final error between T and T_end: 0.0461
```

5.5 Conclusion

In this chapter, we studied the simultaneous diagonalization of matrices problem. The considered matrices were general square matrices. A pencil of matrices $M = [M_1, \ldots, M_s]$ is simultaneously diagonalizable if there exists two matrices E and F such that FM_kE is a diagonal matrix, for $k \in \{1, \ldots, s\}$. In this regards, we addressed three topics. The first (Section 5.1) was about the certification problem of a pencil of simultaneously diagonalizable matrices, where we presented a Newton-type method associated to this problem and we proved, under a sufficient condition on the initial point, a certification of the convergence of the sequence quadratically towards the solution. In the second part (Section 5.2), we tackled the problem of approximate simultaneous diagonalization of matrices (ASD) and we developed a Riemannian conjugate gradient algorithm to locally approximate a pencil of matrices to a pencil of simultaneously diagonalizable matrices. In the last part (Section 5.3), we showed how to use this approach in tensor rank approximation problem,

where there exists a link between simultaneous diagonalization of matrices and CP decomposition of tensors. We used approximate simultaneous diagonalization, more precisely the Riemannian conjugate gradient algorithm 5.1, to implement an algorithm (Algorithm 5.2) that computes an approximated rank-r decomposition for three-dimensional tensors when r is higher than the size of each of the first two dimensions. We note that the results of Section 5.2 and Section 5.3 constitute raw results which we aimed to integrate into this manuscript. Indeed, the investigations are still pursued in this direction, and we are driving more numerical experiments to have a better understanding of the numerical behavior of the two algorithms (5.1 and 5.2), which can help us, for instance, in the analysis of their convergence.

CHAPTER 6

Tensor decomposition for learning Gaussian mixtures from moments

In data processing and machine learning, an important challenge is to recover and exploit models that can represent accurately the data. We consider the problem of recovering Gaussian mixture models from datasets. We investigate symmetric tensor decomposition methods for tackling this problem, where the tensor is built from empirical moments of the data distribution. We consider identifiable tensors, which have a unique decomposition, showing that moment tensors built from spherical Gaussian mixtures have this property. We prove that symmetric tensors with interpolation degree strictly less than half their order are identifiable and we present an algorithm, based on simple linear algebra operations, to compute their decomposition. Illustrative experimentations show the impact of the tensor decomposition method for recovering Gaussian mixtures, in comparison with other state-of-the-art approaches.
Keywords : *Method of moments, Gaussian mixtures, clustering, symmetric tensors.*

In this chapter we address the method of moments for recovering Gaussian mixtures. After reviewing Gaussian mixtures and moment methods in Section 6.1, we present in Section 6.2 an algebraic symmetric tensor decomposition method for identifiable tensors. In Section 6.3, we implement the method of moments to provide an initial point for the Expectation Maximization algorithm for recovering spherical Gaussian mixtures with some examples of synthetic and real datasets, in comparison with other state-of-the-art approaches.

6.1 Gaussian mixtures and high order moments

In this section, we review Gaussian mixture models and their applications to clustering.

6.1.1 Gaussian mixtures

Suppose that we wish to deal with some Euclidean data $x \in \mathbb{R}^m$, coming from a population composed of r homogeneous sub-populations (often called *clusters*). A reasonable assumption is then that each sub-population can be modelled using a simple probability distribution (e.g. Gaussian). This idea is at the heart of the notion of *mixture distribution*. The prime example of mixture is the *Gaussian mixture*, whose probability density over $\mathbb{R}^m$ is defined as

$$p_\theta(x) = \sum_{j=1}^{r} \omega_j \mathcal{N}(x|\mu_j, \Sigma_j), \tag{6.1}$$

where $\mathcal{N}(\cdot|\mu, \Sigma)$ denotes the Gaussian density with mean $\mu \in \mathbb{R}^m$ and definite positive covariance matrices $\Sigma \in \mathcal{S}_m^{++}$. The mixture is parametrized by a typically unknown $\theta = (\omega_1, ..., \omega_r, \mu_1, ..., \mu_r, \Sigma_1, ..., \Sigma_r)$, composed of

— $\omega = (\omega_1, ..., \omega_r)$, that belong to the r-simplex and correspond to the cluster proportions,

— μ_j and Σ_j, that correspond respectively to the mean and covariance of each cluster $j \in \{1, ..., r\}$.

Gaussian mixtures are ubiquitous objects in statistics and machine learning, and own their popularity to many reasons. Let us briefly mention a few of these.

Density estimation If r is allowed to be sufficiently large, it is possible to approximate any probability density using a Gaussian mixture (see e.g. [153]). This motivates the use of Gaussian mixtures as powerful density estimators that can be subsequently used for downstream tasks such as missing data imputation [71], supervised classification [95], or image classification [181] and denoising [106].

Clustering Perhaps the most common use of Gaussian mixtures is *clustering*, also called *unsupervised classification*. The task of clustering consists in uncovering homogeneous groups among the data at hand. Within the context of Gaussian mixtures, each group generally corresponds to a single Gaussian distribution, as in Equation (6.1). If the parameters of a mixture are known, then each point may be clustered using the posterior probabilities obtained via Bayes's rule :

$$\forall x \in \mathbb{R}^m, k \in \{1, ..., r\}, \ \Pr(x \text{ belongs to cluster } j) = \frac{\omega_j \mathcal{N}(x|\mu_j, \Sigma_j)}{p_\theta(x)}. \tag{6.2}$$

Detailed reviews on mixture models and their applications, notably to clustering, can be found in [83, 29, 144].

6.1.2 Learning mixture models

The main statistical question pertaining mixture models is to estimate the parameters $\theta = (\omega_1, ..., \omega_r, \mu_1, ..., \mu_r, \Sigma_1, ..., \Sigma_r)$ based on a data set $x_1, ..., x_n$. Typically, $X_1, ..., X_n$ are assumed to be independent and identically distributed random variables with common density p_{data}. The problem of statistical estimation is then to find some θ such that $p_\theta \approx p_{\text{data}}$. There are many approaches to this question, the most famous one being the *maximum likelihood method*. Maximum likelihood is based on the idea that maximising the *log-likelihood function*

$$\ell(\theta) = \sum_{i=1}^{n} \log p_\theta(x_i), \tag{6.3}$$

will lead to appropriate values of θ. One heuristic reason of the good behaviour of maximum likelihood is that $\ell(\theta)$ can be seen as a measure of how likely the observed data is, according to the mixture model p_θ. This means that the maximum likelihood estimate will be the value of θ that renders the observed data the likeliest. Another interesting interpretation of maximum likelihood in information-theoretic : when $n \longrightarrow \infty$, maximising the log-likelihood is equivalent to minimising the Kullback-Leibler divergence (an information-theoretic measure of distance between probability distributions) between p_θ and p_{data}, thus giving a precise sense to the statement $p_\theta \approx p_{\text{data}}$ (see e.g. [21, Section 1.6.1]). For more details on the properties of maximum likelihood, see e.g. [210, Section 5.5].

In the specific case of a mixture model, performing maximum-likelihood is however complex for several reasons. Firstly, as shown for instance by [130], finding a global maximum is actually often ill-posed in the sense that some problematic values of θ will lead to $\ell(\theta) = \infty$ while being very poor models of the data. While focusing on local rather global maxima will fix this first issue in a sense, iterative optimization algorithms are likely to pursue these unfortunate global maxima. Because of the peculiarities of mixture likelihoods, the most popular algorithm for maximising $\ell(\theta)$ is the *expectation maximization* (EM, [68]) algorithm, an iterative algorithm specialized for dealing with log-likelihoods of latent variable models. The EM algorithm is usually preferred to more generic gradient-based optimization algorithms [223]. In a nutshell, at each iteration, the EM algorithm clusters the data using Equation (6.2), and then computes the mean and covariance of each cluster. This iterative scheme is related to another popular clustering algorithm known as k-means (the close relationship between the two algorithms is detailed in [21, Section 9]). A key issue when using the EM algorithm for a Gaussian mixture is the choice of initialization. Indeed, a poor choice may lead to degenerate solutions, extremely slow convergence, or poor local optima (see [17] and references therein). We will see in this paper that good initial points can be obtained by using another estimation method called the *method of moments* (as was previously noted by [178] in a context of mixtures of multivariate Bernoulli distributions).

The *method of moments* is a general alternative to maximum likelihood. The idea is to choose several functions $g_1 : \mathbb{R}^m \longrightarrow \mathbb{R}^{q_1}, ..., g_d : \mathbb{R}^m \longrightarrow \mathbb{R}^{q_d}$ called *moments*, and to find θ by attempting to solve the system of equations

$$\begin{cases} \mathbb{E}_{x \sim p_{\text{data}}}[g_1(x)] = \mathbb{E}_{x \sim p_\theta}[g_1(x)] \\ \qquad\qquad ... \\ \mathbb{E}_{x \sim p_{\text{data}}}[g_d(x)] = \mathbb{E}_{x \sim p_\theta}[g_d(x)]. \end{cases} \tag{6.4}$$

Of course, since p_{data} is unknown, solving (6.4) is not feasible. However, one may replace the expected moments by empirical versions, and solve instead

$$\begin{cases} \frac{1}{n}\sum_{i=1}^n g_1(x_i) = \mathbb{E}_{x\sim p_\theta}[g_1(x)] \\ \qquad\qquad ... \\ \frac{1}{n}\sum_{i=1}^n g_d(x_i) = \mathbb{E}_{x\sim p_\theta}[g_d(x)]. \end{cases} \tag{6.5}$$

A very simple example of this, in the univariate $m = 1$ case, when $g_1(x) = x$, and $g_2(x) = x^2$. Then, solving (6.4) will ensure that the distributions of the model p_θ and the data p_{data} have the same mean and variance. However, many very different distributions have identical mean and variance ! A natural refinement of the previous idea is to consider also higher-order moments $g_3(x) = x^3, g_4(x) = x^4, ...$ This will considerably improve the estimates found using the method of moments. This approach was pioneered by [164] for learning univariate Gaussian mixtures. In the more general multivariate case $m > 1$, following [107], the moments chosen can be tensor products, as we detail in the next section in case of a Gaussian mixture with spherical covariances.

6.2 Learning structure from tensor decomposition

In this section, we describe the moment tensors revealing the structure of spherical Gaussian mixtures and how it can be decomposed using standard linear algebra operations. We note that tensor methods have been introduced for GMM before, see for instance [189].

Let $\mathbf{X} = (X_1, \dots, X_m)$ be a set of variables. The ring of polynomials in $\mathbf{X}$ with coefficients in $\mathbb{C}$ is denoted $\mathbb{C}[\mathbf{X}]$. The space of homogeneous polynomials of degree $d \in \mathbb{N}$ is denoted $\mathbb{C}[\mathbf{X}]_d$. We recall that a symmetric tensor T of order d (with real coefficients) can be represented by an homogeneous polynomial of degree d in the variables $\mathbf{X}$ of the form

$$T(\mathbf{X}) = \sum_{|\alpha|=d} T_\alpha \binom{d}{\alpha} \mathbf{X}^\alpha$$

where $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}^m$, $|\alpha| = \alpha_1 + \cdots + \alpha_m = d$, $T_\alpha \in \mathbb{R}$, $\binom{d}{\alpha} = \frac{d!}{\alpha_1! \cdots \alpha_m!}$, $\mathbf{X}^\alpha = X_1^{\alpha_1} \cdots X_m^{\alpha_m}$.

A decomposition of T as a sum of d^{th} power of linear forms is of the form

$$T(\mathbf{X}) = \sum_{i=1}^r \omega_i (\xi_i \cdot \mathbf{X})^d \tag{6.6}$$

where $\xi_i = (\xi_{i,1}, \dots, \xi_{i,m}) \in \mathbb{C}^m$ and $(\xi_i \cdot \mathbf{X}) = \sum_{j=1}^m \xi_{i,j} X_j$. When r is the minimal number of terms in such a decomposition, it is called the rank of T and the decomposition is called a rank decomposition (or a Waring decomposition) of $T(\mathbf{X})$.

We say that the decomposition is unique if the lines spanned by $\xi_1, \dots, \xi_r$ form a unique set of lines with no repetition. In this case, the decomposition of T is unique after normalization of the vectors ξ_i up to permutation (and sign change when d is even). A tensor T with a unique decomposition is called an *identifiable* tensor. Then the Waring decompositions of T are of the form $T(\mathbf{X}) = \sum_{i=1}^r \omega_i \lambda_i^{-d} (\lambda_i \, \xi_i \cdot \mathbf{X})^d$ for $\lambda_i \neq 0$, $i \in \{1, \dots, r\}$.

Given a random variable $x \in \mathbb{R}^m$, its moments are $T_\alpha = \mathbb{E}[x_1^{\alpha_1} \cdots x_m^{\alpha_m}]$ for $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbb{N}^m$. The symmetric tensor of all moments of order d of x is

$$\mathbb{E}[(x \cdot \mathbf{X})^d] = \sum_{|\alpha|=d} \mathbb{E}[x_1^{\alpha_1} \cdots x_m^{\alpha_m}] \binom{d}{\alpha} \mathbf{X}^\alpha.$$

6.2.1 The structure of the moment tensor

We aim at recovering the hidden structure a random variable, from the decomposition of its d^{th} order moment tensor. This is possible in some circumstances, that we detail hereafter.

Assumption 6.2.1. *The random variable $x \in \mathbb{R}^m$ is a mixture of spherical Gaussians of probability density* (6.1) *with parameters $\theta = (\omega_1, ..., \omega_r, \mu_1, ..., \mu_r, \sigma_1^2 I_m, , ..., \sigma_r^2 I_m)$ such that $r \leq m$.*

Theorem 6.2.2 ([107])**.** *Under the previous assumption, let*

- *$\tilde{\sigma}^2$ be the smallest eigenvalue of $\mathbb{E}[(x - \mathbb{E}[x]) \otimes (x - \mathbb{E}[x])]$ and v a corresponding unit eigenvector,*
- *$M_1(\mathbf{X}) = \mathbb{E}[(x \cdot \mathbf{X})(v \cdot (x - \mathbb{E}[x]))^2]$,*
- *$M_2(\mathbf{X}) = \mathbb{E}[(x \cdot \mathbf{X})^2] - \tilde{\sigma}^2 \|\mathbf{X}\|^2$,*
- *$M_3(\mathbf{X}) = \mathbb{E}[(x \cdot \mathbf{X})^3] - 3\,\|\mathbf{X}\|^2 M_1(\mathbf{X})$.*

Then $\tilde{\sigma}^2 = \sum_{i=1}^r \omega_i\, \sigma_i^2$ and

$$M_1(\mathbf{X}) = \sum_{i=1}^r \omega_i\, \sigma_i^2\, (\mu_i \cdot \mathbf{X}), \quad M_2(\mathbf{X}) = \sum_{i=1}^r \omega_i\, (\mu_i \cdot \mathbf{X})^2, \quad M_3(\mathbf{X}) = \sum_{i=1}^r \omega_i\, (\mu_i \cdot \mathbf{X})^3. \tag{6.7}$$

To analyse the properties of the decomposition (6.7), we use the apolar product on symmetric tensors from Definition 2.2.1.

For an homogeneous polynomial T of degree $d \in \mathbb{N}$ (or equivalently a symmetric tensor of order d), we define the *Hankel* operator of T in degree $k \leq d$ as the map

$$H_T^{k,d-k} : p \in \mathbb{C}[\mathbf{X}]_{d-k} \mapsto [\langle T, \mathbf{X}^\alpha\, p\rangle_d]_{|\alpha|=k} \in \mathbb{C}^{s_k}$$

where $s_k = \binom{m+k-1}{k} = \dim \mathbb{C}[\mathbf{X}]_k$ is the number of monomials of degree k in $\mathbf{X}$. The matrix of $H_T^{k,d-k}$ in the basis $(\mathbf{X}^\beta)_{|\beta|=d-k}$ is

$$H_T^{k,d-k} = (\langle T, \mathbf{X}^{\alpha+\beta}\rangle_d)_{|\alpha|=k,|\beta|=d-k}.$$

From the properties of the apolar product, we see that $H_T^{1,d-1} : p \mapsto \frac{1}{d}[\langle \partial_{X_i} T, p\rangle_{d-1}]_{1\leq i\leq m}$. For $\xi \in \mathbb{C}^m$ and $k \in \mathbb{N}$, let $\xi^{(k)} = (\xi^\alpha)_{|\alpha|=k}$. We also check that if $T = (\xi \cdot \mathbf{X})^d$ with $\xi \in \mathbb{C}^m$, then $H_{(\xi\cdot\mathbf{X})^d}^{k,d-k} = \bar{\xi}^{(k)} \otimes \bar{\xi}^{(d-k)}$ is of rank 1 and its image is spanned by the vector $\bar{\xi}^{(k)}$.

Proposition 6.2.3. *Assume that $r \leq m$, $w_i > 0$ for $i \in \{1, \ldots, r\}$ and $\mu_1, \ldots, \mu_r \in \mathbb{R}^m$ are linearly independent. The symmetric tensor $M_3(\mathbf{X})$ is identifiable, of rank r and has a unique Waring decomposition satisfying* (6.7)*.*

Proof. Assume that $M_3(\mathbf{X})$ has a decomposition of the form (6.7). Since the vector $\mu_1, \ldots, \mu_r$ are linearly independent, by a linear change of coordinates in Gl_m, we can further assume that $\mu_1 = e_1, \ldots, \mu_r = e_r$ are the first r vectors of the canonical basis of $\mathbb{R}^m$. In this coordinate system, $M_3(\mathbf{X}) = \sum_{i=1}^r X_i^3$ and the matrix $H_{M_3}^{1,2}$ in a convenient basis has a $r \times r$ identity block and zero elsewhere. Thus $H_{M_3}^{1,2}$ is of rank r. Its kernel of dimension $\frac{1}{2}\, m\, (m+1) - r$ is spanned by the polynomials $X_i X_j$ with $(i, j) \neq (k, k)$ for $k \in \{1, \ldots, r\}$. The kernel of $H_{M_3}^{1,2}$ is thus the space of homogeneous polynomials of degree 2, vanishing at $e_1, \ldots, e_r \in \mathbb{R}^n$.

If $M_3(\mathbf{X})$ can be decomposed as $M_3(\mathbf{X}) = \sum_{i=1}^{r'} \omega_i'\, (\mu_i' \cdot \mathbf{X})^3$ with $\omega_i' \in \mathbb{C}$, $\mu_i' \in \mathbb{C}^m$ and $r' < r$, then $H_{M_3}^{1,2}$, as a sum of $r' < r$ matrices $\omega_i' H_{(\mu_i'\cdot\mathbf{X})^3}^{1,2}$ of rank 1, would be of rank smaller

than $r' < r$, which is a contradiction. Thus a minimal decomposition of $M_3(\mathbf{X})$ is of length r and r is the rank of $M_3(\mathbf{X})$.

Let us show that the decomposition (6.7) of $M_3(\mathbf{X})$ is unique up to a scaling of the vector μ_i, i.e. that $M_3(\mathbf{X})$ is identifiable. For any Waring decomposition $M_3(\mathbf{X}) = \sum_{i=1}^r \omega_i' (\mu_i' \cdot \mathbf{X})^3$, the vectors $\mu_1', \ldots, \mu_r'$ are linear independant, since μ_i' spans $\mathrm{im} H^{1,2}_{(\mu_i' \cdot \mathbf{X})^3}$ and $H^{1,2}_{M_3} = \sum_{i=1}^r \omega_i' H^{1,2}_{(\mu_i' \cdot \mathbf{X})^3}$ is of rank r. As $\mu_1', \ldots, \mu_r'$ can be transformed into $e_1, \ldots, e_r$ by a linear change of variables, $\ker H^{1,2}_{M_3}$ is also the vector space of homogeneous polynomials of degree 2, vanishing at $\mu_1', \ldots, \mu_r' \in \mathbb{C}^m$. Therefore, the set of $\{\mu_1', \ldots, \mu_r'\}$ coincides, up to a scaling, with the set of points $\{\mu_1, \ldots, \mu_r\}$ of another Waring decomposition of $M_3(\mathbf{X}) = \sum_{i=1}^r \omega_i (\mu_i \cdot \mathbf{X})^3$. This shows that $M_3(\mathbf{X})$ is identifiable.

Therefore, a Waring decomposition of $M_3(\mathbf{X})$ is of the form $M_3(\mathbf{X}) = \sum_{i=1}^r \tilde{\omega}_i (\tilde{\mu}_i \cdot \mathbf{X})^3$ with $\tilde{\omega}_i = \lambda^{-3}\omega_i$, $\tilde{\mu}_i = \lambda_i \mu_i$ and $\lambda_i \neq 0$ for $i \in \{1, \ldots, r\}$. As $\tilde{\mu}_1, \ldots, \tilde{\mu}_r$ are linearly independent, the homogeneous polynomials $(\tilde{\mu}_1 \cdot \mathbf{X})^2, \ldots, (\tilde{\mu}_r \cdot \mathbf{X})^2$ are also linearly independant in $\mathbb{C}[\mathbf{X}]_2$ (by a linear change of variables, they are equivalent to $X_1^2, \ldots, X_r^2$). Consequently, the relation

$$M_2(\mathbf{X}) = \sum_{i=1}^r \omega_i (\mu_i \cdot \mathbf{X})^2 = \sum_{i=1}^r \lambda_i \tilde{\omega}_i (\tilde{\mu}_i \cdot \mathbf{X})^2$$

defines uniquely $\lambda_1, \ldots, \lambda_r$, and $M_3(\mathbf{X})$ has a unique Waring decomposition, which satisfies the relations (6.7). □

Under Assumption 6.2.1, the hidden structure of the random variable x can thus be recovered using Algorithm 6.1.

Input : The moment tensors $M_1(\mathbf{X}), M_2(\mathbf{X}), M_3(\mathbf{X})$.
- — Compute a Waring decomposition of $M_3(\mathbf{X})$ to get $\tilde{\omega}_i \in \mathbb{R}, \tilde{\mu}_i \in \mathbb{R}^m$, $i \in \{1, \ldots, r\}$ such that $M_3(\mathbf{X}) = \sum_{i=1}^r \tilde{\omega}_i (\tilde{\mu}_i \cdot \mathbf{X})^3$.
- — Solve the system $\sum_{i=1}^r \tilde{\omega}_i (\tilde{\mu}_i \cdot \mathbf{X})^2 \lambda_i = M_2(\mathbf{X})$ to get $\lambda_i \in \mathbb{R}$ and $\omega_i = \lambda_i^3 \tilde{\omega}_i \in \mathbb{R}_+$, $\mu_i = \lambda_i^{-1} \tilde{\mu}_i \in \mathbb{R}^m$ such that $M_3(\mathbf{X}) = \sum_{i=1}^r \omega_i (\mu_i \cdot \mathbf{X})^3$ and $M_2(\mathbf{X}) = \sum_{i=1}^r \omega_i (\mu_i \cdot \mathbf{X})^2$.
- — Solve the system $\sum_{i=1}^r \omega_i (\mu_i \cdot \mathbf{X}) \sigma_i^2 = M_1(\mathbf{X})$ to get $\sigma_i^2 \in \mathbb{R}_+$.

Output : $\omega_i \in \mathbb{R}_+, \mu_i \in \mathbb{R}^n, \sigma_i^2 \in \mathbb{R}_+$ for $i \in \{1, \ldots, r\}$.

Algorithm 6.1 – Recovering the hidden structure of a Gaussian mixture

This yields the parameters $\omega_i \in \mathbb{R}_+, \mu_i \in \mathbb{R}^m$, $\sigma_i \in \mathbb{R}_+$ for $i \in \{1, \ldots, r\}$ of the Gaussian mixture x.

In the experimentation, the moments involved in the tensors M_i will be approximated by empirical moments and we will compute an approximate decomposition of the empirical moment tensor $\hat{M}_3(\mathbf{X})$.

6.2.2 Decomposition of identifiable tensors

We describe now an important step of the approach, which is computing a Waring decomposition of a tensor. In this section, we consider a tensor $T \in \mathbb{C}[\mathbf{X}]_d$ of order $d \in \mathbb{N}$ with a Waring decomposition of the form $T = \sum_{i=1}^r \omega_i (\xi_i \cdot \mathbf{X})^d$ with $\omega_i \in \mathbb{C}, \xi_i \in \mathbb{C}^m$, that we recover by linear algebra techniques, under some hypotheses.

Definition 6.2.1. The interpolation degree $\iota(\Xi)$ of $\Xi = \{\xi_1, \ldots, \xi_r\} \subset \mathbb{C}^m$ is the smallest degree k of a family of homogenous interpolation polynomials $u_1, \ldots, u_r \in \mathbb{C}[\mathbf{X}]_k$ at the points Ξ ($u_i(\xi_j) = \delta_{i,j}$ for $i, j \in \{1, \ldots, r\}$).

For any $d \geq \iota(\Xi)$, there exists a family $(\tilde{u}_i)_{i\in\{1,\ldots,r\}}$ of interpolation polynomials of degree d, obtained from an interpolation family $(u_i)_{i\in\{1,\ldots,r\}}$ in degree $\iota(\Xi)$ as $\tilde{u}_i = \frac{(\lambda\cdot\mathbf{X})^{d-\iota(\Xi)}}{(\lambda\cdot\xi_i)^{d-\iota(\Xi)}} u_i$ for a generic $\lambda \in \mathbb{C}^m$ such that $\lambda \cdot \xi_i \neq 0$ for $i \in \{1, \ldots, r\}$.

Notice that if the points $\Xi = \{\xi_1, \ldots, \xi_r\}$ are linearly independent (and therefore $r \leq m$), then $\iota(\Xi) = 1$ since a family of linear forms interpolating Ξ can be constructed.

If $k \geq \iota(\Xi)$, then the evaluation map $\mathbf{e}_\Xi^{(k)} : p \in \mathbb{C}[\mathbf{X}]_k \mapsto (p(\xi_1), \ldots, p(\xi_r)) \in \mathbb{C}^r$ is surjective. Its kernel is the space of homogeneous polynomials of degree k vanishing at Ξ. Any supplementary space admits a basis $u_1, \ldots, u_r$, which is an interpolating family for Ξ in degree k. A property of the interpolation degree is the following :

Lemma 6.2.4. *For $k > \iota(\Xi)$, the common roots of $\ker \mathbf{e}_\Xi^{(k)}$ is the union $\cup_{i=1}^r \mathbb{C}\,\xi_i$ of lines spanned by $\xi_1, \ldots, \xi_r \in \mathbb{C}^m$.*

Proof. As $\iota(\Xi) + 1$ is the Castelnuovo-Mumford regularity of the vanishing ideal $I(\Xi) = \{p \in \mathbb{C}[\mathbf{X}] \mid p \text{ homogeneous}, p(\xi) = 0 \text{ for } \xi \in \Xi\}$ [76][Ch.4], it is generated in degree $k > \iota(\Xi)$ and the common roots of $\ker \mathbf{e}_\Xi^{(k)} = I(\Xi)_k$ is $\cup_{i=1}^r \mathbb{C}\,\xi_i$. □

Hereafter, we show that tensors T such that $\mathrm{rank} H_T^{k,d-k} = r$ for $k > \iota(\Xi) + 1$ are identifiable and we describe a numerically robust algorithm to compute their Waring decomposition.

Let $U = (U_{\alpha,j})_{|\alpha|=k, j\in\{1,\ldots,r\}} \in \mathbb{C}^{s_k \times r}$ be such that $\mathrm{im}\, U = \mathrm{im}\, H_T^{k,d-k}$ and $U_i = (U_{e_i+\alpha,j})_{|\alpha|=k-1, j\in\{1,\ldots,r\}}$ be the submatrices of U with the rows indexed by the monomials divisible by X_i for $i \in \{1, \ldots, m\}$.

Theorem 6.2.5. *Let $T \in \mathbb{C}[\mathbf{X}]_d$ with a decomposition $T = \sum_{i=1}^r \omega_i\, (\xi_i \cdot \mathbf{X})^d$ with $\omega_i \in \mathbb{C}$ and $\xi_i = (\xi_{i,1}, \ldots, \xi_{i,n}) \in \mathbb{C}^m$ such that $\mathrm{rank} H_T^{k,d-k} = r$ for some $k \in [\iota(\xi_1, \ldots, \xi_r) + 1, d]$. Then T is identifiable of rank r and there exist invertible matrices $E \in \mathbb{C}^{s_k \times s_k}$, $F \in \mathbb{C}^{r\times r}$ such that*

$$E^t\, U_i\, F = \begin{bmatrix} \Delta_i \\ 0 \end{bmatrix} \tag{6.8}$$

with $\Delta_i = \mathrm{diag}(\bar{\xi}_{1,i}, \ldots, \bar{\xi}_{r,i})$ for $i \in \{1, \ldots, m\}$. For any pair (E, F), which diagonalizes simultaneously $[U_1, \ldots, U_m]$ as in (6.8)*, there exist unique $\omega'_1, \ldots, \omega'_r \in \mathbb{C}$ such that $T = \sum_{i=1}^r \omega'_i\, (\xi'_i \cdot \mathbf{X})^d$ with $\bar{\xi}'_i = ((\Delta_1)_{i,i}, \ldots, (\Delta_m)_{i,i})$.*

Proof. From the decomposition of T, we have for $k \leq d$ that

$$H_T^{k,d-k} = \sum_{i=1}^r \omega_i\, \bar{\xi}_i^{(k)} \otimes \bar{\xi}_i^{(d-k)}$$

is a linear combination of r Hankel matrices $\bar{\xi}_i^{(k)} \otimes \bar{\xi}_i^{(d-k)}$ of rank 1. If T is of rank $r' < r$, then using its decomposition of rank r', $H_T^{k,d-k}$ would be of rank $\leq r' < r$, which is a contradiction. This shows that T is of rank r.

As $\mathrm{rank} H_T^{k,d-k} = r$, we deduce that the image of $H_T^{k,d-k}$ is spanned by $\bar{\xi}_1^{(k)}, \ldots, \bar{\xi}_r^{(k)}$ and there exists an invertible matrix $F \in \mathbb{C}^{r\times r}$ such that

$$U\,F = [\bar{\xi}_1^{(k)}, \ldots, \bar{\xi}_r^{(k)}]$$

For any polynomial $p \in \mathbb{C}[\mathbf{X}]_k$, which coefficient vector in the monomial basis $(\mathbf{X}^\alpha)_{|\alpha|=k}$ is denoted $[p]$, we have $[p]^t U F = [p(\bar{\xi}_1), \ldots, p(\bar{\xi}_r)]^t$. This shows that $U^\perp = \{p \in \mathbb{C}[\mathbf{X}] \mid [p]^t U = 0\}$ is $\ker \mathbf{e}_{\bar{\Xi}}^{(k)}$. By Lemma 6.2.4 since $k \geq \iota(\bar{\Xi})$, the common roots of the homogeneous polynomials in $\ker \mathbf{e}_{\bar{\Xi}}^{(k)}$ are the scalar multiples of $\bar{\Xi}$. Consequently, the set of lines spanned by the vectors Ξ of a Waring decomposition of T is uniquely determined as the conjugate of the zero locus of $U^\perp \subset \mathbb{C}[\mathbf{X}]_k$ and T is identifiable.

For any $p \in \mathbb{C}[\mathbf{X}]_{k-1}$ represented by its coefficient vector $[p]$ in the monomial basis $(\mathbf{X}^\alpha)_{|\alpha|=k-1}$, we have

$$[p]^t U_i F = [x_i p]^t U F = [\bar{\xi}_{1,i}\, p(\bar{\xi}_1), \ldots, \bar{\xi}_{r,i}\, p(\bar{\xi}_r)]^t. \tag{6.9}$$

Let E be the coefficient matrix of a basis $u_1, \ldots, u_r, v_{r+1}, \ldots, v_{s_{k-1}}$ of $\mathbb{C}[\mathbf{X}]_{k-1}$, such that $u_1, \ldots, u_r$ is an interpolating family for $\bar{\Xi} = \{\bar{\xi}_1, \ldots, \bar{\xi}_r\}$ and $v_{r+1}, \ldots, v_{s_{k-1}}$ is a basis of $\ker \mathbf{e}_{\bar{\Xi}}^{(k-1)}$. The matrix E is invertible by construction, and we deduce from (6.9) that

$$E^t U_i F = \begin{bmatrix} \mathrm{diag}(\bar{\xi}_{1,i}, \ldots, \bar{\xi}_{r,i}) \\ 0 \end{bmatrix}.$$

Let us show conversely that for any pair of matrices (E', F'), which diagonalizes simultaneously $[U_1, \ldots, U_m]$ as in (6.8) with $\Delta_i = \mathrm{diag}(\bar{\xi}'_{1,i}, \ldots, \bar{\xi}'_{r,i})$, there exist unique $\omega'_1, \ldots, \omega'_r \in \mathbb{C}$ such that $T = \sum_{i=1}^r \omega'_i\, (\xi'_i \cdot \mathbf{X})^d$.

Let $u'_1, \ldots, u'_r, v'_{r+1}, \ldots, v'_{s_{k-1}} \in \mathbb{C}[\mathbf{X}]$ be the polynomials corresponding to the columns of E'. Then for a generic $\lambda = (\lambda_1, \ldots, \lambda_r) \in \mathbb{C}^m$, we have

$$\begin{aligned} \mathrm{diag}((\lambda \cdot \bar{\xi}'_1), \ldots, (\lambda \cdot \bar{\xi}'_r)) &= \sum_{i=1}^m \lambda_i [u'_1, \ldots, u'_r]^t U_i F' = \sum_{i=1}^m \lambda_i [u'_1, \ldots, u'_r]^t U_i F (F^{-1} F') \\ &= [(\lambda \cdot \bar{\xi}_j)\, u'_i(\bar{\xi}_j)]_{i,j\in\{1,\ldots,r\}} F^{-1} F' \\ &= \mathrm{diag}((\lambda \cdot \bar{\xi}_1), \ldots, (\lambda \cdot \bar{\xi}_r))\, [u'_i(\bar{\xi}_j)]_{i,j\in\{1,\ldots,r\}} F^{-1} F'. \end{aligned}$$

As $\lambda \in \mathbb{C}^m$ is generic and $\lambda \cdot \bar{\xi}_i \neq 0$ for $i \in \{1, \ldots, r\}$, we deduce that $\Delta = [u'_i(\bar{\xi}_j)]_{i,j\in\{1,\ldots,r\}} F^{-1} F'$ is a diagonal and invertible matrix and that $\xi'_i = \bar{\Delta}_{i,i} \xi_i$ with $\Delta_{i,i} \neq 0$.

Then we have $(\xi'_i \cdot \mathbf{X})^d = \bar{\Delta}_{i,i}^d\, (\xi_i \cdot \mathbf{X})^d$ and $T = \sum_{i=1}^r \omega'_i\, (\xi'_i \cdot \mathbf{X})^d$ with $\omega'_i = \bar{\Delta}_{i,i}^{-d} \omega_i$, which concludes the proof of the theorem. □

This leads to Algorithm 6.2 to compute a Waring decomposition of an identifiable tensor T.

6.3 Numerical experimentations

The model used in this section is the Gaussian Mixture Model (GMM) with differing spherical covariance matrices. Recall that if $x = (x_1, \ldots, x_n)$ is a sample of n independent observations from r multivariate Gaussian mixture with differing spherical covariance matrices of dimension

Input : $T \in \mathbb{C}[\mathbf{X}]_d$, which admits a decomposition with r points $\Xi = \{\xi_1, \ldots, \xi_r\}$ and $k > \iota(\Xi)$.
- — Compute the Singular Value Decomposition of $H_T^{k,d-k} = U\,S\,V^t$;
- — Deduce the rank r of $H_T^{k,d-k}$, take the first r columns of U and build the submatrices U_i with rows indexed by the monomials $(X_i\mathbf{X}^\alpha)_{|\alpha|=k-1}$ for $i \in \{1, \ldots, n\}$;
- — Compute a simultaneous diagonalization of the pencil $[U_1 \ldots, U_m]$ as $E^t U_i F = \begin{bmatrix} \mathrm{diag}(\bar{\xi}_{1,i}, \ldots, \bar{\xi}_{r,i}) \\ 0 \end{bmatrix}$ and deduce the points $\xi_i = (\xi_{i,1}, \ldots, \xi_{i,m}) \in \mathbb{C}^m$ for $i \in \{1, \ldots, r\}$;
- — Compute the weights $\omega_1, \ldots, \omega_r$ by solving the linear system $T = \sum_{i=1}^r \omega_i\,(\xi_i \cdot \mathbf{X})^d$;

Output : $\omega_i \in \mathbb{C}$, $\xi_i \in \mathbb{C}^m$ s.t. $T = \sum_{i=1}^r \omega_i\,(\xi_i \cdot \mathbf{X})^d$.

Algorithm 6.2 – Decomposition of an identifiable tensor

m, and $h = (h_1, h_2, \ldots, h_n)$ is the latent variable that determine the component from which the observation originates, then :

$$x_i \mid (h_i = k) \sim \mathcal{N}_m(\mu_k, \sigma_k^2 I_m) \text{ where,}$$

$$\Pr(h_i = k) = \omega_k, \text{ for } k \in \{1, \ldots, r\}, \text{ such that } \sum_{k=1}^{r} \omega_k = 1.$$

The aim of statistical inference is to find the unknown parametrs μ_k, σ_k^2 and w_k, for $k \in \{1, \ldots, r\}$ from the data x. This can be done by finding the maximum likelihood estimation (MLE) i.e. finding the optimal maximum of the likelihood function associated to this model. The expectation maximization algorithm (EM) [68], usually used for finding MLEs, is an iterative algorithm in which the initialization i.e. the initial estimation of the latent parameters is crucial, since various initializations can lead to different local maxima of the likelihood function, consequently, yielding different clustering partition. Thus, in this section we compare the clustering results obtained by different initialization of the EM algorithm against the initialization by the method of moments through examples of simulated (subsection 6.3.1) and real (subsection 6.3.2) datasets. We fix a maximum of 100 iterations of the EM algorithm. The different initialization considered in this section are the following :

- — The k-means method [142] according to the following strategy :
  The best partition obtained out of 50 runs of the k-means algorithm.
- — The method of moments, where Algorithm 6.1 is applied to build the moments and Algorithm 6.2 is applied to the empirical moment tensor corresponding to $M_3(\mathbf{v}X)$ (see Algorithm 6.2.2), with less than 5 Riemannian Newton iterations (Chapter 4) to reduce the distance between the empirical moment tensor and its decomposition.
- — The Model-based hierarchical agglomerative clustering algorithm (MBHC) [209, 82].
- — The emEM strategy [20] as in [131] which makes 5 iterations for each of 50 short runs of EM, and follows the one which maximizes the log-likelihood function by a long run of EM.

The k-means, MBHC and emEM are common strategies for initialising the EM algorithm for GMMs. The comparison among the different EM initialization strategies is based on three measures : The Bayesian Information Criterion (BIC) [184, 81], the Adjusted Rand Index (ARI) [108], and the error rate (errorRate). The BIC is a penalized-likelihood criterion given by the following

formula

$$\mathrm{BIC} = -2\ell(\hat{\theta}) + \log(n)\nu,$$

where ℓ is the log-likelihood function , $\hat{\theta}$ is the MLE which maximizes the log-likelihood function and ν is the number of the estimated parameters. This criterion measures the quality of the model such that for comparing models the one with the largest BIC value among the other models is the most fitted to the studied dataset. The ARI criterion measures the similarity between the estimated clustering obtained by the applied model and the exact true clustering. Its value is bounded between 0 and 1. The more this measure is close to 1 the more the estimated clustering is accurate. The error rate measure can be viewed as an alternative of the ARI. In fact this criterion measures the minimum error between the predicted clustering and the true clustering, and thus low error rate means high agreement between the estimated and the true clustering. The former criteria as well as the EM algorithm are used from the tools of the package `mclust` [185] in R programming language.

6.3.1 Simulation

We performed 100 simulations from each of the two models described in examples 6.3.1 and 6.3.2. We counted the instances where each of the considered initialising strategies for the EM could find throughout the 100 simulated data and among the other initialization methods the largest BIC, the highest ARI, ARI≥ 0.99 (as in this case the clustering obtained is the most accurate) and the lowest errorRate. The values of the BIC, ARI, errorRate and consumed time of the different considered initialization strategies for one dataset sampled according to the model of Example 6.3.1 (resp. 6.3.2) are presented in Table 6.1 (resp. 6.3), and Figure 6.1 (resp. 6.2) shows a two-dimensional visualization of the observations according to the first four features, the observations in the upper panels are labeled according to the actual clustering, while they are labeled in the lower panels according to the clustering obtained by the EM algorithm initialized by the method of moments. In order to have an estimation about the numerical stability of the obtained results, we repeat the same numerical experiment for each example 20 times and we compute the means (Table 6.2, 6.4) and the variances (values in parentheses in Table 6.2, 6.4) of the 20 percentages obtained of each of the BIC, ARI, ARI≥ 0.99 and errorRate values for the different initialising strategies.

As we mentioned before the initialization strategies considered in this comparison against the method of moments are common and have, in general, good numerical behavior. Nevertheless, we cannot expect all the initialization strategies that exist for the EM algorithm to work well in all the cases [20, 145]. Hereafter, two examples are chosen in such a way to present some cases where the common initialization strategies k-means, MBHC and emEM have some difficulties to provide a good initialization to the EM algorithm for the GMMs with differing spherical covariance matrices, or in other words where the initialization by the method of moments outperforms the other considered initializations. For instance, we put in each of these two examples one cluster of small size (the blue cluster in Figure 6.1, the red cluster in Figure 6.2), we want to make the clusters overlap, since these initialization strategies could misscluster the dataset if the clusters are intersecting. We notice that this choice of the mean vectors and the different variances in each of the two examples yields a dataset with the expected clustering characteristic.

Example 6.3.1 – In the first simulation example, a multivariate dataset (m=6) of n=1000 observations generated with r=4 clusters according to the following parameters :

— The probability vector : $\omega = (0.2782, 0.0139, 0.3324, 0.3756)^t$.
— The mean vectors : $\mu_1 = (-5.0, -9.0, 8.0, 8.0, 2.0, 5.0)^t$, $\mu_2 = (-7.0, 6.0, -1.0, 6.0, -8.0, -10.0)^t$, $\mu_3 = (-4.0, -10.0, -5.0, 1.0, 5.0, 4.0)^t$, $\mu_4 = (-6.0, 6.0, 5.0, 4.0, -1.0, -1.0)^t$.
— The variances : $\sigma_1^2 = 1.5$, $\sigma_2^2 = 2.5$, $\sigma_3^2 = 5.0$, $\sigma_4^2 = 15.0$.

Table 6.1 – Numerical results of one data set of Example 6.3.1

Method	BIC	ARI	errorRate	time(s)
em_km	-29590.48	0.8281	0.168	**0.045**
em_mom	**-29492.11**	**1.0**	**0.0**	0.547
em_mbhc	-29594.97	0.8574	0.099	0.287
em_emEM	-29593.18	0.8366	0.132	0.171

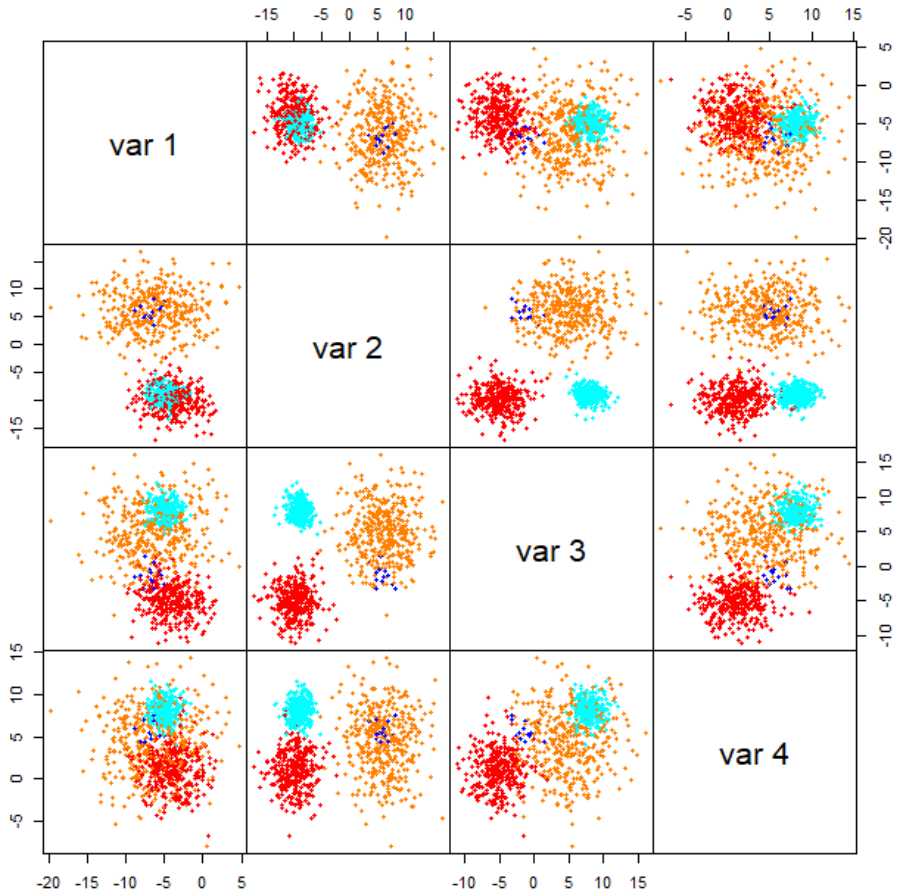


Figure 6.1 – Scatterplot matrix for the sampled dataset of Example 6.3.1 projected onto the first four variables (features) : upper panels show scatterplots for pairs of variables in the original clustering ; lower panels show the clustering obtained by applying the EM algorithm initialized by the method of moments.

Table 6.2 – Estimation of the stability of Example 6.3.1 results

Method	BIC	ARI	ARI ≥ 0.99	errorRate
em_km	38.35% (37.82)	47.6% (21.41)	48.85% (21.61)	47.6% (21.2)
em_mom	**74.8%** (41.01)	**88.75%** (15.36)	**83.4%** (18.36)	**88.60%** (14.46)
em_mbhc	10.75% (12.41)	15.9% (17.57)	15.55% (22.99)	15.9% (19.46)
em_emEM	7.3% (8.43)	14.5% (8.05)	12.6% (17.83)	14.95% (7.52)

Example 6.3.2 – In the second simulation example, a multivariate dataset (m=5) of n=1000 observations generated with r=3 clusters according to the following parameters :

— The probability vector : $\omega = (0.0930, 0.2151, 0.6918)^t$.

— The mean vectors : $\mu_1 = (7.0, -4.0, -4.0, -6.0, -4.0)^t$, $\mu_2 = (2.0, -4.0, -6.0, -10.0, -3.0)^t$, $\mu_3 = (4.0, -4.0, -5.0, 6.0, 1.0)^t$.

— The variances : $\sigma_1^2 = 5.0$, $\sigma_2^2 = 10.0$, $\sigma_3^2 = 15.0$.

Table 6.3 – Numerical results of one data set of Example 6.3.2

Method	BIC	ARI	errorRate	time(s)
em_km	-28360.30	0.4352	0.309	**0.051**
em_mom	**-28246.02**	**0.9498**	**0.03**	0.504
em_mbhc	-28358.67	0.3197	0.384	0.292
em_emEM	-28360.42	0.4408	0.296	0.141

Table 6.4 – Estimation of the stability of Example 6.3.2 results

Method	BIC	ARI	ARI ≥ 0.99	errorRate
em_km	0.45% (0.576)	0.05% (0.05)	0.0% (0.0)	0.1%(0.095)
em_mom	**50.0%** (18.63)	**92.35%** (9.82)	**0.0%** (0.0)	**92.1%** (7.46)
em_mbhc	49.35% (19.82)	2.45% (3.63)	0.0% (0.0)	2.45% (2.58)
em_emEM	0.3% (0.326)	5.2% (4.48)	0.0% (0.0)	5.9% (5.36)

The Table 6.2, 6.4 show that in Example 6.3.1, 6.3.2 the best results among the considered initialising strategies are for the method of moments. In fact, in the former two tables we see that the method of moments found throughout the 100 simulated datasets, in average (by runing the numerical experiment 20 times), the largest BIC, highest ARI, ARI≥ 0.99 and lowest errorRate among the other initialization strategies in more instances than all the other considered initialization method, implying in this context marked outperformance for the moments initialization method. Note that the consumed time (see. Table 6.1, 6.3) tends to be higher in the method of moments than in the other initialization strategies. This is expected since stochastic approaches (to which the methods k-means, MBHC and emEM belong) outperform the deterministic approaches (as the method of moments) in this term.

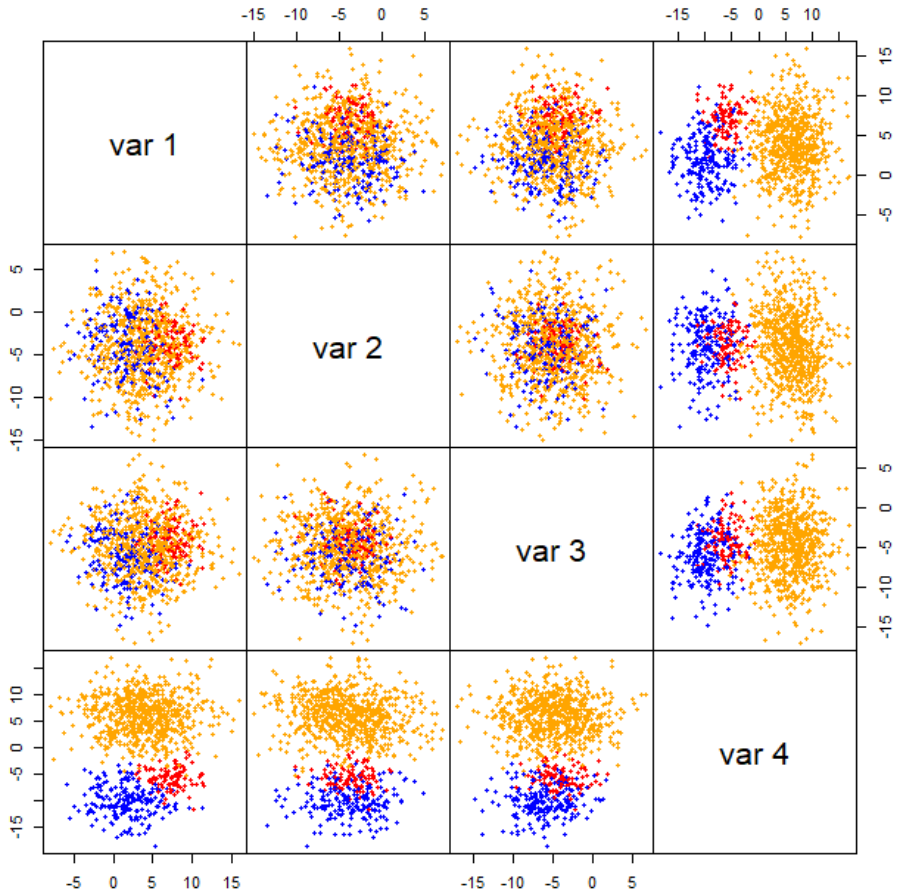


Figure 6.2 – Scatterplot matrix for the sampled dataset of Example 6.3.2 projected onto the first four variables (features) : upper panels show scatterplots for pairs of variables in the original clustering ; lower panels show the clustering obtained by applying the EM algorithm initialized by the method of moments.

6.3.2 Real data

In this subsection we present four examples of real datasets, for which we know already their number of clusters, and we report the different BIC, ARI and errorRate values as well as the consumed time attained by the EM algorithm initialized by the different considered initialization strategies and used with the GMM of different spherical covariance matrices. The explored real data are : The famous iris data [79, 70] widely used as an example of clustering to test the algorithms, Diabetes [173], olive oil [15], and MNIST [69].

Example 6.3.3 – The iris dataset contains four physical measurements (length and width of sepals and petals) for 50 samples of three species of iris (setosa, virginica and versicolor). The number of features is $m = 4$ and the number of clusters is $r = 3$.
The four initialization strategies yield the same BIC value. The ARI and the errorRate values are

Table 6.5 – Numerical results of Example 6.3.3

Method	BIC	ARI	errorRate	time(s)
em_km	-1227.6656	0.6199	0.167	**0.007**
em_mom	**-1227.6676**	**0.6410**	**0.153**	0.203
em_mbhc	-1227.6696	0.6199	0.167	**0.007**
em_emEM	-1227.6495	0.6302	0.160	0.045

slightly better with the moment initialization among the other considered initialization strategies. On the other hand, the consumed time is clear higher in the moment method initialization.

Example 6.3.4 – The Diabete dataset [173] contains three measurements : glucose, insulin and sspg ; made on 145 non-obese adult patients classified into three types of diabetes : Normal, Overt, and Chemical. Herein, in this example $m = r = 3$. We apply the different initialization strategies for the EM algorithm, the Table 6.6 shows the results.

Table 6.6 – Numerical results of Example 6.3.4

Method	BIC	ARI	errorRate	time(s)
em_km	-5363.06	0.3371	0.289	**0.007**
em_mom	-5222.11	**0.6355**	**0.144**	0.380
em_mbhc	**-5221.32**	**0.6355**	**0.144**	0.008
em_emEM	-5221.33	0.6207	0.151	0.049

Despite the fact that k-means method is the fastest method in this example, the ARI and the BIC are noticeably lower than in the other methods. Concerning the method of moments, it succeeds to have quite similar scores to the other methods in this example, but with a bigger computation time.

Example 6.3.5 – The olive oil data set contains the chemical composition (8 chemical properties) of 572 olive oils. They are derived from three different macro-areas in Italy (South, Sardinia and Centre North). The dataset contains nine regions from which the olive oils were taken in Italy. Thus we can cluster this dataset according to the macro-areas ($r = 3$) or the region ($r = 9$). As the number of features in this dataset is $m = 8$, we choose $r = 3$, so that the condition $r \leq m$ for

the method of moment is verified.
The results show that the MBHC initialization strategy yields the largest BIC, the highest ARI and

Table 6.7 – Numerical results of Example 6.3.5

Method	BIC	ARI	errorRate	time(s)
em_km	-10948.64	0.4018	0.262	**0.021**
em_mom	-10946.46	0.4532	0.210	0.508
em_mbhc	**-10625.59**	**0.5003**	**0.185**	0.080
em_emEM	-10948.72	0.4040	0.260	0.087

the lowest errorRate values among the other initialization strategies. Nevertheless, the initialization by the moment method comes in second position after the MBHC strategy in terms of the BIC, ARI and errorRate values, while the K-means and the emEM initialization strategies attain almost the same values of the previously mentionned criteria.

This shows that for these datasets which are not well fitted by the mixture of spherical Gaussians, the moment method can still give good initializations for the EM algorithm, in comparision with the common initialization strategies.

Example 6.3.6 – The MNIST digit image database [69] is a large database that contains images of 28×28 pixels for handwritten digits (0 to 9). Each pixel contains an integer between 0 and 255 that represents the grayscale levels. The number of features is $28 \times 28 = 784$. We choose the MNIST digit image dataset which contains 60000 images. We take a subset of this dataset that contains the images of label 0 or 1. The size of the subset is 12665 images. Since the number of features is quite large (784), and we aim to test a spherical Gaussian mixture model, a good practice in this case is to apply one of the dimensionality reduction strategies. Roughly speaking, the dimensionality reduction strategies aim to reduce the number of features such that a high percentage of the information within the dataset is conserved. In other words, the performance in term of accuracy of the clustering methods will not be noticeably affected by this reduction, and on the other hand this will reduce considerably the time of computation. For this purpose, we choose to apply the Principal Component Analysis transformation (PCA) [84, 115]. We conserve the first five variables given by this transformation (see Figure 6.3). The dataset that we consider in this example contains 12665 observations, the number of clusters is $r = 2$, and the number of features is $m = 5$. We apply the different initialization strategies and we report the results in Table 6.8.
As we can see, the results given by the method of moments in Table 6.8 are very satisfactory in

Table 6.8 – Numerical results of Example 6.3.6

Method	BIC	ARI	errorRate	time(s)
em_km	-384977.3	0.9304	**0.017**	**0.537**
em_mom	-384978.2	**0.9308**	**0.017**	1.87
em_mbhc	**-382746.2**	0.2445	0.252	543.4
em_emEM	-384977.6	0.9301	0.0177655	1.80

comparison with the other initialization strategies with ARI$= 0.9308$. In particular, the method of moments clearly outperforms MBHC method in this regard, in term of accuracy and the time of computation. In fact, the MBHC takes 543.4 seconds without reaching a *good* ARI score. This

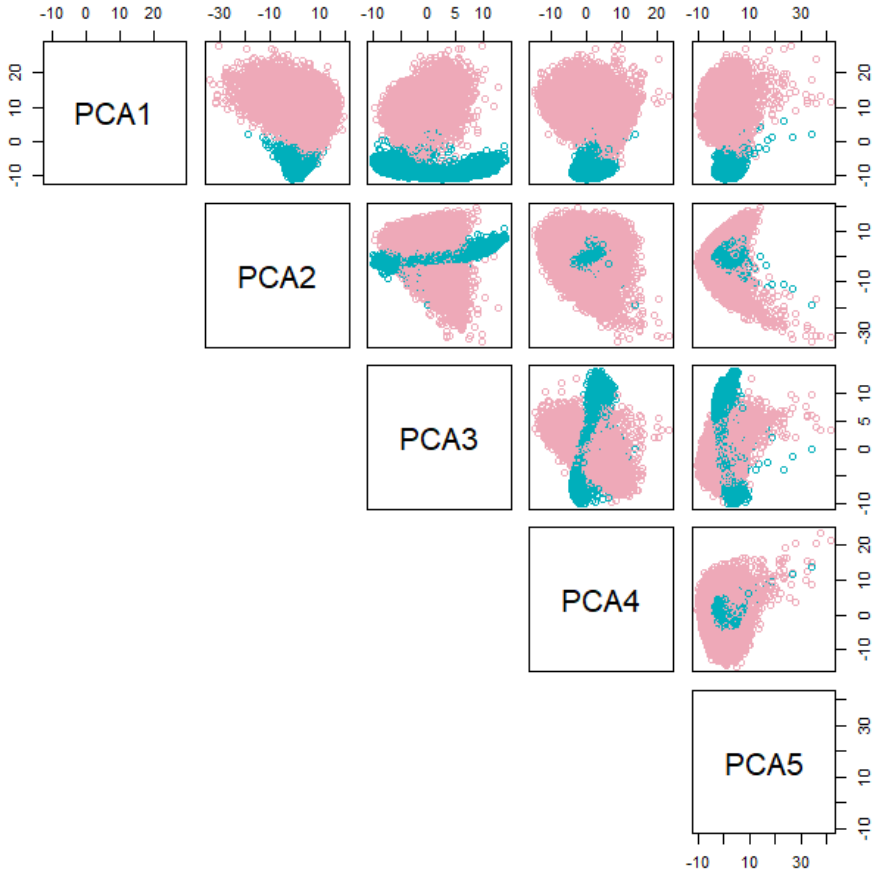


Figure 6.3 – Scatterplot for pairs of variables : upper panels show the first five features obtained by applying the PCA transformation on the dataset of Example 6.3.6. The graphs points marked according to the true two classes 0 and 1.

example sheds some light on the performance of the method of moments. The large number of samples (in this example equal to 12665) does not have a high impact on the computation time, which is not the case, for the MBHC method, where this factor increases significantly its computation time. Moreover, it is true that a large number of features could have a negative impact on the computation time of the method of moments, but it is not a sever limitation since as we saw in this example, this can be efficiently remedied by applying one of the dimensionality reduction techniques. In this regard, some recent work [165] studies how the computation complexity of the moment method can be reduced while conserving its desirable high accuracy property. Conducting more research in this direction, we believe that the method of moments will have more sophisticated and competitive (in term of computation time) developments in the future.

6.4 Conclusion

We present an algorithm that use our low rank symmetric tensor approximation algorithms (4) to implement the method of moments. We propose to use this method to initialize the EM algorithm, and we show concretely throughout synthetic and real datasets examples the good impact of this choice in comparison with other state-of-the-art approaches. We prove that symmetric tensors with interpolation degree strictly less than half their order are identifiable, and we present an algorithm, based on simple linear algebra operations, to compute their decomposition.

Conclusions and Perspectives

CHAPTER 7

Conclusions and Perspectives

We summarize the result obtained in this thesis, and we describe some research perspectives.

Low rank symmetric tensor approximation problem We addressed the low rank tensor approximation problem for symmetric tensors with complex coefficients. Using the basic link between symmetric tensors and homogeneous polynomials we described two Riemannian Newton type optimization algorithms with trust region steps, which are Riemannian Newton algorithm and Riemannian Gauss–Newton algorithm. In the first algorithm, we used the parametrization of the constraint set (the set of symmetric tensors of symmetric rank bounded by the approximation rank r strictly less than the generic rank) by using the standard weighted normalized factor matrices parameterization. We computed explicitly the Hessian matrix by exploiting the apolar identities and partial complex derivatives tools. We proved that under some regularity conditions on non-defective tensors in the neighborhood of the initial point, the iteration completed with a trust region scheme is converging to a local minimum. In the second algorithm, we used for the first time the Veronese manifold to parameterize the constraint set. We presented a suitable basis for the tangent space of Veronese manifold which allowed us computing the different ingredients of a Riemannian Gauss-Newton iteration. We proposed to choose the initial point for strictly subgeneric ranks approximation with interpolation degree less than $\frac{d-1}{2}$, with d is the order of the symmetric tensor, by the method that we called the SMD based on the computation of generalized eigenvectors and generalized eigenvalues of pencils of Hankel matrices, and we showed the good impact of this choice on the numerical performance of the algorithms. A series of numerical experiments were presented, through which we tested our algorithms versus other state-of-the-art algorithms. The algorithms that we presented are particularly well suited for the symmetric case. For instance, the Gauss-Newton algorithm is strictly connected with the geometry of varieties of rank-r symmetric tensors. Indeed, we showed throughout this work how taking into account the geometric structure of the constraint set where the target approximation lives can be relevant and can conduct to concrete efficient approximation algorithms. In this direction, we believe that a natural extension of this work, is to investigate new optimization algorithms for the low rank tensor approximation problem for other types of tensors by adopting the same methodology of this work, i.e. by interacting with the geometry of the problem while dealing with this approximation problem as a numerical optimization problem. For instance, for multisymmetric tensors this will lead to investigate the so-called Segre-Veronese manifold and to exploit the corresponding with multi-homogeneous polynomials. Also practical interesting questions interrogate the possibility of the use of some tensor's compression techniques such as the symmetric Tucker decomposition

[61, 62, 66]

$$\mathcal{T} = [\![\mathcal{S}; A, \ldots, A]\!] = \sum_{i_1=1}^{l} \ldots \sum_{i_d=1}^{l} s_{i_1 \ldots i_d} a_{i_1} \otimes \ldots \otimes a_{i_d},$$

for $\mathcal{T}$ is a symmetric tensor in $\mathcal{S}^d(\mathbb{R}^n)$, $\mathcal{S}$ is a symmetric tensor in $\mathcal{S}^d(\mathbb{R}^l)$ and $A \in \mathbb{R}^{n \times l}$ is an orthogonal matrix. For symmetric rank-r approximation, we can try to compute this symmetric Tucker decomposition for l bounded by r and then to apply our symmetric rank-r approximation algorithms on the core tensor in such a way that if

$$\mathcal{S} \approx \sum_{i=1}^{r} s_i \otimes \ldots \otimes s_i;$$

we can take $\mathcal{T} \approx \sum_{i=1}^{r} (As_i) \otimes \ldots \otimes (As_i)$, as a low rank approximation for $\mathcal{T}$. This approach could be interesting especially to reduce the computation complexity. This also allows us to combine the two formats of the approximation problem which are tensors and polynomials, where the symmetric tensor $\mathcal{T}$ is used to compute a Tucker decomposition and the homogeneous polynomial associated to the core tensor is used to compute a low rank approximation, and thus to take advantage from the reduction of the computation cost than can be realized using each of the two formats. Another interesting question is to construct a method based on simple linear algebra operations that can provide a good initialization for our algorithms for all strictly subgeneric symmetric rank without the limitation regarding the bound on the interpolation degree on which depends the method SMD that we use.

Simultaneous matrix diagonalization We addressed the simultaneous matrix diagonalization problem. We presented a Newton-type sequence that converges quadratically towards the solution of the system of equations associated to a pencil of simultaneously diagonalizable matrices. We considered the case of one diagonalizable matrix and two simultaneously diagonalizable matrices, then based on the resolution of the two aforementioned cases we concluded on the case of a family of simultaneously diagonalizable matrices. Moreover, we gave a certification on the quadratic convergence towards the solution when the initial point verify a sufficient condition that we established. This approach allows the computation in high precision, where this type of computation could be important in certain circumstances. Also, it is an iterative method that allows to obtain the numerical solution of a system of equations with the desirable property of local quadratic convergence without depending on computing the inverse of matrices such in a classical Newton iteration. We focused on the regular case. Some improvements and extensions can be considered, such as the treatment of clusters of eigenvalues.
Further, we described a Riemannian conjugate gradient algorithm that approximates a pencil of matrices no necessarily simultaneously diagonalizable by a pencil of simultaneously diagonalizable matrices based on solving a Riemannian optimization problem over the Cartesian product of two oblique manifolds. This approach can be considered as a generalization of the recent work in [25] where the matrices of the pencil are considered symmetric and the problem depends on one diagonalizer matrix E with its inverse. We employed this algorithm to construct an alternate optimization algorithm which for a tridimensinal tensor of size (n_1, n_2, n_3) and approximation rank r such that $r \geq \max(n_1, n_2)$ takes the pencil of matrices that correspond to the frontal slices of the tensor, then extends the pencil of $n_1 \times n_2$ matrices to $r \times r$ matrices. The algorithm then works on the extended pencil and alternate between two steps. The first step uses the conjugate

gradient algorithm to approximate the extended pencil by a pencil of simultaneously diagonalizable matrices, where this allows to compute a rank-r approximation for the tensor associated to the extended pencil. The second step solves a linear least-squares problem to update the entries that does not belong to the fixed blocks matrices that correspond to the matrices of the original pencil. Finally, the algorithm extract a rank-r approximation for the original tensor. This is a recent approach, that we continue to investigate and more analysis are needed to have a better understanding of the numerical performance. Indeed, the convergence of the method is not yet understood. Moreover, we observed that the method becomes slow for ill-conditioned cases. This point can be enhanced by developing a strategy to avoid ill-conditioned points, for instance by adding a regularization term to the ill-conditioned matrices. We note that because of this extension step, the applicability of the method could be limited to approximation ranks which exceed moderately the dimension of the two first modes. Finally, after dealing with these questions, it will be interesting to apply this algorithm to applications in data science for instance in image processing.

Tensor decomposition for learning Gaussian mixtures from moments In the context of unsupervised machine learning, the type of models to be recovered plays an important role. For Gaussian mixture models, where iterative methods such as Expectation Maximization algorithms are applied, the choice of the initialization is also crucial to recover an accurate model of a given dataset. We considered the method of moments to recover Gaussian mixtures, in particular spherical Gaussian mixtures, in order to achieve clustering tasks. We computed the first, second and third order moments in terms of the means and the covariances of the clusters and the vector of the cluster proportions. We used the estimation given by the method of moments as an initial point to the EM algorithm. We demonstrated in the experimentations that tensor decomposition techniques can provide a good initial point for the EM algorithm, and that the moment tensor method outperforms the other state-of-the-art initialization strategies in term of accuracy, when datasets are well represented by spherical Gaussian mixture models. For that purpose, we presented a new tensor decomposition algorithm adapted to the decomposition of identifiable tensors with low interpolation degree, which applies to a 3^{rd} order moment tensors associated to the data distribution as we have shown. One inconvenient side for the method of moments in comparison with the other approaches is that in general it consumes more computation time. This is not a limitation, since in general a successful method is the method that can make a trade-off between accuracy and complexity. Hence, since the method of moments works well in term of accuracy, a straightforward question is to try to reduce its complexity. For instance, we have seen that large number of features affects negatively the computation time of the method where we proposed to refine this using one of the dimensionality reduction strategies, in this regard, further approaches can be investigated. Another direction that can be explored, is the construction of moments method for non-spherical Gaussian mixtures. Finally, in term of tensor applications, we notice that low rank tensor approximation problems are recently widely explored for deep learning applications thus future works can be oriented in this direction.

Bibliography

[1] P.-A. Absil and K.A. Gallivan. Joint diagonalization on the oblique manifold for independent component analysis. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 5, pages V–V, 2006.

[2] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

[3] P. A. Absil, Robert Mahony, and Jochen Trumpf. An extrinsic look at the Riemannian Hessian. In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, pages 361–368, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[4] P.-A. Absil and Jérôme. Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1) :135–158, 2012.

[5] Evrim Acar, Daniel M. Dunlavy, Tamara G. Kolda, and Morten Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1) :41–56, 2011. Multiway and Multiset Data Analysis.

[6] Evrim Acar and Bülent Yener. Unsupervised multiway data analysis : A literature survey. *IEEE transactions on knowledge and data engineering*, 21(1) :6–20, 2008.

[7] B. Afsari. Sensitivity analysis for the problem of matrix joint diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 30 :1148–1171, 2008.

[8] Bijan Afsari and Perinkulam S. Krishnaprasad. Some gradient based joint diagonalization methods for ica. In Carlos G. Puntonet and Alberto Prieto, editors, *Independent Component Analysis and Blind Signal Separation*, pages 437–444, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[9] Riku AKEMA, Masao YAMAGISHI, and Isao YAMADA. Approximate simultaneous diagonalization of matrices via structured low-rank approximation. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E104.A(4) :680–690, 2021.

[10] J. Alexander and André Hirschowitz. Polynomial interpolation in several variables. volume 4, pages 201–222, 1995.

[11] Elizabeth S. Allman, Catherine Matias, and John A. Rhodes. Identifiability of parameters in latent structure models with many observed variables. *Annals of Statistics*, 37(6A) :3099–3132, 12 2009.

[12] Shun-ichi Amari, Tian-Ping Chen, and Andrzej Cichocki. Nonholonomic orthogonal learning algorithms for blind source separation. *Neural computation*, 12(6) :1463–1484, 2000.

[13] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15 :2773–2832, 2014.

[14] E. Andruchow, G. Larotonda, L. Recht, and A. Varela. The left invariant metric in the general linear group. *Journal of Geometry and Physics*, 86 :241–257, 2014.

[15] Adelchi Azzalini and Giovanna Menardi. Clustering via nonparametric density estimation : The R package pdfcluster. *Journal of Statistical Software, Articles*, 57(11) :1–26, 2014.

[16] Davide Bacciu and Danilo P. Mandic. Tensor decompositions in deep learning. *CoRR*, abs/2002.11835, 2020.

[17] Jean-Patrick Baudry and Gilles Celeux. Em for mixtures. *Statistics and computing*, 25(4) :713–726, 2015.

[18] A. Bernardi, J. Brachat, P. Comon, and B. Mourrain. General tensor decomposition, moment matrices and applications. *Journal of Symbolic Computation*, 52 :51–71, May 2013.

[19] Alessandra Bernardi, Alessandro Gimigliano, and Monica Idà. Computing symmetric rank for symmetric tensors. *Journal of Symbolic Computation*, 46(1) :34–53, 2011.

[20] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Choosing starting values for the em algorithm for getting the highest likelihood in multivariate gaussian mixture models. *Comput. Stat. Data Anal.*, 41(3–4) :561–575, January 2003.

[21] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[22] Ake. Bjorck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.

[23] Greg Blekherman and Zach Teitler. On maximum, typical, and generic ranks. *CoRR*, abs/1402.2371, 2014.

[24] Stéphane Bonhomme, Koen Jochmans, and Jean-Marc Robin. Nonparametric estimation of non-exchangeable latent-variable models. *Journal of Econometrics*, 201(2) :237–248, 2017.

[25] Florent Bouchard, Bijan Afsari, Jérôme Malick, and Marco Congedo. Approximate joint diagonalization with Riemannian optimization on the general linear group. *SIAM Journal on Matrix Analysis and Applications*, 41(1) :152–170, 2020.

[26] Florent Bouchard, Jérôme Malick, and Marco Congedo. Riemannian optimization and approximate joint diagonalization for blind source separation. *IEEE Transactions on Signal Processing*, 66(8) :2041–2054, 2018.

[27] A. Boudjellal, A. Mesloub, K. Abed-Meraim, and A. Belouchrani. Separation of dependent autoregressive sources using joint matrix diagonalization. *IEEE Signal Processing Letters*, 22(8) :1180–1183, 2015.

[28] Nicolas Bourbaki. *Algebra I : Chapters 1–3, Elements of Mathematics*. Springer-Verlag, Berlin, 1998.

[29] Charles Bouveyron, Gilles Celeux, T Brendan Murphy, and Adrian E Raftery. *Model-based clustering and classification for data science : with applications in R*, volume 50. Cambridge University Press, 2019.

[30] Jerome Brachat, Pierre Comon, Bernard Mourrain, and Elias Tsigaridas. Symmetric tensor decomposition. *Linear Algebra and its Applications*, 433(11-12) :1851–1872, December 2010.

[31] D. H. Brandwood. A complex gradient operator and its application in adaptive array theory. *IEE Proceedings F : Communications Radar and Signal Processing*, 130(1) :11–16, February 1983.

[32] Paul. Breiding and Nick. Vannieuwenhoven. The condition number of join decompositions. *SIAM Journal on Matrix Analysis and Applications*, 39(1) :287–309, 2018.

[33] Paul. Breiding and Nick. Vannieuwenhoven. A Riemannian trust region method for the canonical tensor rank approximation problem. *SIAM Journal on Optimization*, 28(3) :2435–2465, 2018.

[34] Rasmus Bro. Parafac tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 38(2) :149–171, 1997.

[35] Rasmus Bro. Multi-way analysis in the food industry - models, algorithms, and applications. Technical report, MRI, EPG and EMA," Proc ICSLP 2000, 1998.

[36] Angelika Bunse-Gerstner, Ralph Byers, and Volker Mehrmann. A chart of numerical methods for structured eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 13(2) :419–453, 1992.

[37] Angelika Bunse-Gerstner, Ralph Byers, and Volker Mehrmann. Numerical methods for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 14(4) :927–949, 1993.

[38] Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science & Business Media, , 2013.

[39] Cesar F. Caiafa and Andrzej Cichocki. Generalizing the column–row matrix decomposition to multi-way arrays. *Linear Algebra and its Applications*, 433(3) :557–573, 2010.

[40] Jean-François Cardoso and Antoine Souloumiac. Blind beamforming for non gaussian signals. *IEE Proceedings-F*, 140 :362–370, 1993.

[41] Jean-François Cardoso and Antoine Souloumiac. Jacobi angles for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 17(1) :161–164, 1996.

[42] J. Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition. *Psychometrika*, 35(3) :283–319, Sep 1970.

[43] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3) :283–319, 1970.

[44] Bilian Chen, Simai He, Zhening Li, and Shuzhong Zhang. Maximum block improvement and polynomial optimization. *SIAM Journal on Optimization*, 22(1) :87–107, 6 2012.

[45] Jann-Long Chern and Luca Dieci. Smoothness and periodicity of some matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 22(3) :772–792, 2001.

[46] Luca Chiantini, Giorgio Ottaviani, and Nick Vannieuwenhoven. An algorithm for generic and low-rank specific identifiability of complex tensors. *SIAM Journal on Matrix Analysis and Applications*, 35(4) :1265–1287, 2014.

[47] Luca Chiantini, Giorgio Ottaviani, and Nick Vannieuwenhoven. On generic identifiability of symmetric tensors of subgeneric rank. *Transactions of the American Mathematical Society*, 369(6) :4021–4042, Nov 2016.

[48] Luca Chiantini, Giorgio Ottaviani, and Nick Vannieuwenhoven. Effective criteria for specific identifiability of tensors and forms. *SIAM Journal on Matrix Analysis and Applications*, 38(2) :656–681, 2017.

[49] A. Cichocki, R. Zdunek, A.H. Phan, and S. Amari. *Nonnegative Matrix and Tensor Factorizations : Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley, 2009.

[50] Andrzej Cichocki. Era of big data processing : A new approach via tensor networks and tensor decompositions, 2014.

[51] Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan. Tensor decompositions for signal processing applications : From two-way to multiway component analysis. *IEEE signal processing magazine*, 32(2) :145–163, 2015.

[52] Pierre Comon. Tensor decompositions, state of the art and applications. In J. G. McWhirter and I. K. Proudler, editors, *Mathematics in Signal Processing V*, pages 1–24. Clarendon Press, Oxford, 2002.

[53] Pierre Comon. Tensors : A brief introduction. *IEEE Signal Processing Magazine*, 31(3) :44–53, 2014.

[54] Pierre. Comon, Gene. Golub, Lek-Heng. Lim, and Bernard. Mourrain. Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis and Applications*, 30(3) :1254–1279, 2008.

[55] Pierre Comon and Christian Jutten. *Handbook of Blind Source Separation : Independent Component Analysis and Applications*. Academic Press, Inc., USA, 1st edition, 2010.

[56] Pierre Comon and Myriam Rajih. Blind identification of under-determined mixtures based on the characteristic function. *Signal Processing*, 86(9) :2271 – 2281, 2006. Special Section : Signal Processing in UWB Communications.

[57] David A. Cox, John B. Little, and Donal O'Shea. *Using Algebraic Geometry*. Number 185 in Graduate Texts in Mathematics. Springer, New York, 2nd edition, 2005.

[58] G.R. Curbastro. *Résumé de quelques travaux sur les systèmes variables de fonctions associés a une forme différentielle quadratique*. Gauthier-Villars, 1892.

[59] L. de Lathauwer, B. de Moor, and J. Vandewalle. Independent component analysis and (simultaneous) third-order tensor diagonalization. *IEEE Transactions on Signal Processing*, 49(10) :2262–2271, 2001.

[60] Lieven De Lathauwer. A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. *SIAM journal on Matrix Analysis and Applications*, 28(3) :642–666, 2006.

[61] Lieven De Lathauwer. Decompositions of a higher-order tensor in block terms—part i : Lemmas for partitioned matrices. *SIAM Journal on Matrix Analysis and Applications*, 30(3) :1022–1032, 2008.

[62] Lieven De Lathauwer. Decompositions of a higher-order tensor in block terms—part ii : Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*, 30(3) :1033–1066, 2008.

[63] Lieven. De Lathauwer, Bart. De Moor, and Joos. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4) :1253–1278, 2000.

[64] Lieven. De Lathauwer, Bart. De Moor, and Joos. Vandewalle. On the best rank-1 and rank-(R_1, R_2, . . ., R_n) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4) :1324–1342, 2000.

[65] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. Computation of the canonical decomposition by means of a simultaneous generalized schur decomposition. *SIAM Journal on Matrix Analysis and Applications*, 26(2) :295–327, 2004.

[66] Lieven De Lathauwer and Dimitri Nion. Decompositions of a higher-order tensor in block terms—part iii : Alternating least squares algorithms. *SIAM Journal on Matrix Analysis and Applications*, 30(3) :1067–1083, 2008.

[67] Vin de Silva and Lek-Heng Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3) :1084–1127, Jan 2008.

[68] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1) :1–38, 1977.

[69] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6) :141–142, 2012.

[70] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository. `https://archive.ics.uci.edu/ml/index.php`, 2017.

[71] Marco Di Zio, Ugo Guarnera, and Orietta Luzi. Imputation through finite Gaussian mixture models. *Computational Statistics & Data Analysis*, 51(11) :5305–5316, 2007.

[72] Ignat Domanov and Lieven De Lathauwer. Canonical polyadic decomposition of third-order tensors : Relaxed uniqueness conditions and algebraic algorithm. *Linear Algebra and its Applications*, 513 :342–375, 2017.

[73] Ignat Domanov and Lieven De Lathauwer. Canonical polyadic decomposition of third-order tensors : Reduction to generalized eigenvalue decomposition. *SIAM Journal on Matrix Analysis and Applications*, 35(2) :636–660, 2014.

[74] S.C. Douglas. Self-stabilized gradient algorithms for blind source separation with orthogonality constraints. *IEEE Transactions on Neural Networks*, 11(6) :1490–1497, 2000.

[75] Petros Drineas and Michael W. Mahoney. A randomized algorithm for a tensor-based generalization of the singular value decomposition. *Linear Algebra and its Applications*, 420(2) :553–571, 2007.

[76] David Eisenbud. *The Geometry of Syzygies : A Second Course in Commutative Algebra and Algebraic Geometry*. Springer, 2005.

[77] Mohamed Elkadi and Bernard Mourrain. *Introduction à la résolution des systèmes polynomiaux*, volume 59 of *Mathèmatiques et Applications*. Springer, , 2007.

[78] Mike Espig, Wolfgang Hackbusch, and Aram Khachatryan. On the convergence of alternating least squares optimisation in tensor format representations. *arXiv preprint arXiv :1506.00062*, 2015.

[79] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2) :179–188, 1936.

[80] Bernhard N. Flury and Walter Gautschi. An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form. *SIAM Journal on Scientific and Statistical Computing*, 7(1) :169–184, 1986.

[81] C. Fraley and A. E. Raftery. How many clusters ? which clustering method ? answers via model-based cluster analysis. *The Computer Journal*, 41(8) :578–588, 1998.

[82] Chris Fraley. Algorithms for model-based gaussian hierarchical clustering. *SIAM Journal on Scientific Computing*, 20(1) :270–281, 1998.

[83] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458) :611–631, 2002.

[84] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11) :559–572, 1901.

[85] Tuo Fu and Xiqi Gao. Simultaneous diagonalization with similarity transformation for non-defective matrices. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 4, pages IV–IV, 2006.

[86] Luis David Garcia, Michael Stillman, and Bernd Sturmfels. Algebraic geometry of Bayesian networks. *Journal of Symbolic Computation*, 39(3-4) :331–355, Mar 2005.

[87] L. Godinho and J. Natário. *An Introduction to Riemannian Geometry : With Applications to Mechanics and Relativity*. Universitext. Springer International Publishing, 2014.

[88] Abhinav Goel, Caleb Tung, Yung-Hsiang Lu, and George K. Thiruvathukal. A survey of methods for low-power deep learning and computer vision. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6, 2020.

[89] M. Haardt and J.A. Nossek. Simultaneous schur decomposition of several nonsymmetric matrices to achieve automatic pairing in multidimensional harmonic retrieval problems. *IEEE Transactions on Signal Processing*, 46(1) :161–169, 1998.

[90] Martin Haardt, Florian Roemer, and Giovanni Del Galdo. Higher-order svd-based subspace estimation to improve the parameter estimation accuracy in multidimensional harmonic retrieval problems. *IEEE Transactions on Signal Processing*, 56(7) :3198–3213, 2008.

[91] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2012.

[92] Jouhayna Harmouch, Houssam Khalil, and Bernard Mourrain. Structured low rank decomposition of multivariate Hankel matrices. *Linear Algebra and Its Applications*, 542 :161–185, April 2018.

[93] Richard Harshman. Foundations of the PARAFAC procedure : Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16 :1–84, 1970.

[94] Richard A Harshman. FOUNDATIONS OF THE PARAFAC PROCEDURE : MODELS AND CONDITIONS FOR AN "EXPLANATORY" MULTIMODAL FACTOR ANALYSIS. page 84.

[95] Trevor Hastie and Robert Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society : Series B (Methodological)*, 58(1) :155–176, 1996.

[96] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[97] Chikio Hayashi and Fumi Hayashi. A new algorithm to solve Parafac-model. *Behaviormetrika*, 9(11) :49–60, Jan 1982.

[98] Chikio Hayashi and Fumi Hayashi. A new algorithm to solve parafac-model. *Behaviormetrika*, 9 :49–60, 1982.

[99] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[100] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6) :1–39, 2013.

[101] Frank L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4) :164–189, 1927.

[102] J. V. D. Hoeven and B. Mourrain. Efficient certification of numeric solutions to eigenproblems. In *MACIS*, 2017.

[103] Joris van der Hoeven and Jean-Claude Yakoubsohn. Certified singular value decomposition. Technical Report HAL 01941987, 2018.

[104] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Presss, Cambridge, 1991.

[105] Roger A Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, Cambridge, 2012.

[106] Antoine Houdard, Charles Bouveyron, and Julie Delon. High-dimensional mixture models for unsupervised image denoising (HDMI). *SIAM Journal on Imaging Sciences*, 11(4) :2815–2846, 2018.

[107] Daniel Hsu and Sham M. Kakade. Learning mixtures of spherical gaussians : Moment methods and spectral decompositions. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 11–20, New York, NY, USA, January 2013. Association for Computing Machinery.

[108] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1) :193–218, 1985.

[109] Johan Håstad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4) :644–654, 1990.

[110] R. Iferroudjene, K. Abed Meraim, and A. Belouchrani. A new jacobi-like method for joint diagonalization of arbitrary non-defective matrices. *Applied Mathematics and Computation*, 211(2) :363–373, 2009.

[111] Majid Janzamin, Rong Ge, Jean Kossaifi, and Anima Anandkumar. Spectral Learning on Matrices and Tensors. *Foundations and Trends® in Machine Learning*, 12(5-6) :393–536, 2019.

[112] J.Harris. *Algebraic Geometry : A First Course*. Graduate Texts in Mathematics, Springer-Verlag, New York, NY, 1998.

[113] M. Joho and K. Rahbar. Joint diagonalization of correlation matrices by using Newton methods with application to blind signal separation. *Sensor Array and Multichannel Signal Processing Workshop Proceedings, 2002*, pages 403–407, 2002.

[114] Ian Jolliffe. *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[115] Ian Jolliffe. *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[116] Joe Kileel and João M. Pereira. Subspace power method for symmetric tensor decomposition and generalized PCA. 2019.

[117] Tamara G. Kolda. Multilinear operators for higher-order decompositions. Technical Report SAND2006-2081, Sandia National Laboratories, April 2006.

[118] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3) :455–500, September 2009.

[119] Tamara G. Kolda and Jackson R. Mayo. An adaptive shifted power method for computing generalized tensor eigenpairs. *SIAM Journal on Matrix Analysis and Applications*, 35(4) :1563–1581, 2014.

[120] K. Konstantinides and K. Yao. Statistical analysis of effective singular values in matrix rank determination. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(5) :757–763, May 1988.

[121] Tomaž Košir and Bor Plestenjak. On stability of invariant subspaces of commuting matrices. *Linear Algebra and its Applications*, 342(1) :133–147, 2002.

[122] D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by Riemannian optimization. *BIT Numerical Mathematics*, 54(2) :447–468, 2014.

[123] D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by Riemannian optimization. *BIT Numerical Mathematics*, 54(2) :447–468, 2014.

[124] Wim P Krijnen, Theo K Dijkstra, and Alwin Stegeman. On the non-existence of optimal solutions and the occurrence of "degeneracy" in the candecomp/parafac model. *Psychometrika*, 73(3) :431–439, 2008.

[125] J. B. Kruskal. *Rank, Decomposition, and Uniqueness for 3-Way and n-Way Arrays*, page 7–18. North-Holland Publishing Co., NLD, 1989.

[126] JB Kruskal, RA Harshman, and ME Lundy. How 3-mfa data can cause degenerate parafac solutions, among other relationships. In *Multiway data analysis*, pages 115–122. 1989.

[127] Joseph B. Kruskal. Three-way arrays : rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications*, 18(2) :95–138, 1977.

[128] J. Landsberg. *Tensors : Geometry and Applications*, volume 128 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, Rhode Island, December 2011.

[129] J.M. Landsberg. *Tensors : Geometry and Applications*. Graduate studies in mathematics. American Mathematical Society, 2011.

[130] Lucien Le Cam. Maximum likelihood : an introduction. *International Statistical Review/-Revue Internationale de Statistique*, pages 153–171, 1990.

[131] Rémi Lebret, Serge Iovleff, Florent Langrognet, Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Rmixmod : The R package of the model-based unsupervised, supervised, and semi-supervised classification Mixmod library. *Journal of Statistical Software*, 67(6) :1–29, 2015.

[132] John Lee. *Introduction to Smooth Manifolds. 2nd revised ed*, volume 218. 01 2012.

[133] Seok Lee, Minseok Choi, Hyungmin Kim, and Frank Chongwoo Park. Geometric direct search algorithms for image registration. *IEEE Transactions on Image Processing*, 16(9) :2215–2224, 2007.

[134] S. E. Leurgans, R. T. Ross, and R. B. Abel. A Decomposition for Three-Way Arrays. *SIAM Journal on Matrix Analysis and Applications*, 14(4) :1064–1083, October 1993.

[135] Lek-Heng Lim and Pierre Comon. Nonnegative approximations of nonnegative tensors. *Journal of Chemometrics : A Journal of the Chemometrics Society*, 23(7-8) :432–441, 2009.

[136] Lek-Heng Lim and Pierre Comon. Blind multilinear identification. *IEEE Transactions on Information Theory*, 60(2) :1260–1280, 2014.

[137] Xiangqian Liu and N.D. Sidiropoulos. Cramer-rao lower bounds for low-rank decomposition of multidimensional arrays. *IEEE Transactions on Signal Processing*, 49(9) :2074–2086, 2001.

[138] Haiping Lu, Konstantinos N. Plataniotis, and Anastasios N. Venetsanopoulos. A survey of multilinear subspace learning for tensor data. *Pattern Recognition*, 44(7) :1540–1551, 2011.

[139] Xavier Luciani and Laurent Albera. Canonical polyadic decomposition based on joint eigenvalue decomposition. *Chemometrics and Intelligent Laboratory Systems*, 132 :152–167, 2014.

[140] Xavier Luciani and Laurent Albera. Canonical polyadic decomposition based on joint eigenvalue decomposition. *Chemometrics and Intelligent Laboratory Systems*, 132 :152–167, 2014.

[141] Xavier Luciani and Laurent Albera. Joint eigenvalue decomposition of non-defective matrices based on the lu factorization with application to ica. *IEEE Transactions on Signal Processing*, 63(17) :4594–4608, 2015.

[142] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[143] RE Mahony. The constrained newton method on a lie group and the symmetric eigenvalue problem. *Linear algebra and its applications*, 248 :67–89, 1996.

[144] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. Finite mixture models. *Annual review of statistics and its application*, 6 :355–378, 2019.

[145] Volodymyr Melnykov and Ranjan Maitra. Finite mixture models and model-based clustering. *Statistics Surveys*, 4(none) :80 – 116, 2010.

[146] Ammar Mesloub, Adel Belouchrani, and Karim Abed-Meraim. Efficient and stable joint eigenvalue decomposition based on generalized givens rotations. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1247–1251. IEEE, 2018.

[147] Mi Miller, A Trouvd, L Younes, and Cmla ens Cachan. The metric spaces, euler equations, and normal geodesic image motions of computational anatomy. In *Proceedings of the 2003 International Conference on Image Processing*, pages 635–638. IEEE.

[148] Martin J Mohlenkamp. Musings on multilinear fitting. *Linear Algebra and its Applications*, 438(2) :834–852, 2013.

[149] Morten Mørup. Applications of tensor (multiway array) factorizations and decompositions in data mining. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 1(1) :24–40, 2011.

[150] Bernard Mourrain. Polynomial-exponential decomposition from moments. *Foundations of Computational Mathematics*, 18(6) :1435–1492, December 2018.

[151] Kevin P Murphy. *Machine learning : a probabilistic perspective*. MIT press, 2012.

[152] Z. Nehari. *Introduction to Complex Analysis*. Allyn & Bacon, 1968.

[153] TrungTin Nguyen, Faicel Chamroukhi, Hien D Nguyen, and Geoffrey J McLachlan. Approximation of probability density functions via location-scale finite mixtures in Lebesgue spaces. *arXiv preprint arXiv :2008.09787*, 2020.

[154] Jiawang Nie. Low rank symmetric tensor approximations. *SIAM Journal on Matrix Analysis and Applications*, 38(4) :1517–1540, 2017.

[155] Jiawang. Nie and Li. Wang. Semidefinite relaxations for best rank-1 tensor approximations. *SIAM Journal on Matrix Analysis and Applications*, 35(3) :1155–1179, 2014.

[156] M. Nikpour, J. Manton, and G. Hori. Algorithms on the Stiefel manifold for joint diagonalisation. *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2 :II–1481–II–1484, 2002.

[157] Yasunori Nishimori and Shotaro Akaho. Learning algorithms utilizing quasi-geodesic flows on the Stiefel manifold. *Neurocomput.*, 67 :106–135, August 2005.

[158] Luke Oeding and Giorgio Ottaviani. Eigenvectors of tensors and algorithms for waring decomposition. *Journal of Symbolic Computation*, 54 :9–35, 2013.

[159] I. V. Oseledets, D. V. Savostianov, and E. E. Tyrtyshnikov. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM Journal on Matrix Analysis and Applications*, 30(3) :939–956, 2008.

[160] Samet Oymak and Mahdi Soltanolkotabi. Learning a deep convolutional neural network via tensor decomposition. *Information and Inference : A Journal of the IMA*, 10(3) :1031–1071, 2021.

[161] Pentti Paatero. A weighted non-negative least squares algorithm for three-way 'parafac' factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 38(2) :223–242, 1997.

[162] Pentti Paatero. The multilinear Engine–A Table-Driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model. *Journal of Computational and Graphical Statistics*, 8(4) :854–888, 1999.

[163] Pentti Paatero. Construction and analysis of degenerate parafac models. *Journal of Chemometrics*, 14, 2000.

[164] Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185 :71–110, 1894.

[165] João M Pereira, Joe Kileel, and Tamara G Kolda. Tensor moments of gaussian mixture models : Theory and applications. *arXiv preprint arXiv :2202.06930*, 2022.

[166] A. Phan, Petr Tichavský, and Andrzej Cichocki. Fast alternating ls algorithms for high order candecomp/parafac tensor factorizations. *IEEE Transactions on Signal Processing*, 61 :4834–4846, 2013.

[167] Anh Huy Phan and Andrzej Cichocki. Tensor decompositions for feature extraction and classification of high dimensional datasets. *Nonlinear theory and its applications, IEICE*, 1(1) :37–68, 2010.

[168] Anh-Huy Phan, Petr Tichavský, and Andrzej Cichocki. Candecomp/parafac decomposition of high-order tensors through tensor reshaping. *IEEE Transactions on Signal Processing*, 61(19) :4847–4860, 2013.

[169] Anh-Huy. Phan, Petr. Tichavský, and Andrzej. Cichocki. Low complexity damped Gauss–Newton algorithms for CANDECOMP/PARAFAC. *SIAM Journal on Matrix Analysis and Applications*, 34(1) :126–147, 2013.

[170] Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. Introduction to Tensor Decompositions and their Applications in Machine Learning. *arXiv :1711.10781 [cs, stat]*, November 2017. Comment : 13 pages, 12 figures.

[171] Kamran Rahbar and James P. Reilly. Geometric optimization methods for blind source separation of signals. In *in Proc. ICA*, pages 375–380, 2000.

[172] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2) :1126–1153, 2013.

[173] Gerald M. Reaven and Rachel G. Miller. An attempt to define the nature of chemical diabetes using a multidimensional analysis. *Diabetologia*, 16 :17–24, 1979.

[174] K. Reich. *Die Entwicklung des Tensorkalküls : Vom absoluten Differentialkalkül zur Relativitätstheorie*. Science Networks. Historical Studies. Birkhäuser Basel, 1994.

[175] R. Remmert and R.B. Burckel. *Theory of Complex Functions*. Graduate Texts in Mathematics. Springer New York, 1991.

[176] Florian Roemer, Carola Schroeter, and Martin Haardt. A semi-algebraic framework for approximate cp decompositions via joint matrix diagonalization and generalized unfoldings. In *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 2023–2027, 2012.

[177] B. Rosser, C. Lanczos, M.R. Hestenes, and W. Karush. Separation of close eigen-values of a real symmetric matrix. *Journal of Research of the National Bureauof Standards*, 47, 1950.

[178] Matteo Ruffini, Ricard Gavalda, and Esther Limón. Clustering patients with tensor decomposition. In *Machine Learning for Healthcare Conference*, pages 126–146. PMLR, 2017.

[179] E. Sanchez and B. Kowalski. Tensorial resolution : A direct trilinear decomposition. *undefined*, 1990.

[180] Eugenio Sanchez and Bruce R. Kowalski. Tensorial resolution : A direct trilinear decomposition. *Journal of Chemometrics*, 4(1) :29–45, 1990.

[181] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the Fisher vector : Theory and practice. *International journal of computer vision*, 105(3) :222–245, 2013.

[182] Berkant Savas and Lars Eldén. Handwritten digit classification using higher order singular value decomposition. *Pattern Recognition*, 40(3) :993–1003, 2007.

[183] Berkant. Savas and Lek-Heng. Lim. Quasi-Newton methods on Grassmannians and multilinear approximations of tensors. *SIAM Journal on Scientific Computing*, 32(6) :3352–3393, 2010.

[184] Gideon Schwarz. Estimating the Dimension of a Model. *Annals of Statistics*, 6(2) :461–464, July 1978.

[185] Luca Scrucca, Michael Fop, T. Brendan Murphy, and Adrian E. Raftery. mclust 5 : clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1) :289–317, 2016.

[186] Yaroslav Shitov. A counterexample to comon's conjecture. *SIAM Journal on Applied Algebra and Geometry*, 2(3) :428–443, 2018.

[187] N.D. Sidiropoulos, R. Bro, and G.B. Giannakis. Parallel factor analysis in sensor array processing. *IEEE Transactions on Signal Processing*, 48(8) :2377–2388, 2000.

[188] Nicholas D Sidiropoulos and Rasmus Bro. On the uniqueness of multilinear decomposition of n-way arrays. *Journal of chemometrics*, 14(3) :229–239, 2000.

[189] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13) :3551–3582, 2017.

[190] Age Smilde, Rasmus Bro, and Paul Geladi. *Multi-way Analysis with Applications in the Chemical Sciences*. John Wiley, West Sussex, UK, 2004.

[191] Laurent. Sorber, Marc Van. Barel, and Lieven De. Lathauwer. Unconstrained optimization of real functions in complex variables. *SIAM Journal on Optimization*, 22(3) :879–898, 2012.

[192] Laurent. Sorber, Marc. Van Barel, and Lieven. De Lathauwer. Optimization-based algorithms for tensor decompositions : Canonical polyadic decomposition, decomposition in rank-$(l_r, l_r, 1)$ terms, and a new generalization. *SIAM Journal on Optimization*, 23(2) :695–720, 2013.

[193] Mikael Sørensen and Lieven De Lathauwer. Multidimensional harmonic retrieval via coupled canonical polyadic decomposition—part i : Model and identifiability. *IEEE Transactions on Signal Processing*, 65(2) :517–527, 2016.

[194] Mikael Sørensen and Lieven De Lathauwer. Multidimensional harmonic retrieval via coupled canonical polyadic decomposition—part ii : Algorithm and multirate sampling. *IEEE Transactions on Signal Processing*, 65(2) :528–539, 2016.

[195] Mikael Sørensen, Ignat Domanov, and Lieven De Lathauwer. Coupled canonical polyadic decompositions and multiple shift invariance in array processing. *IEEE Transactions on Signal Processing*, 66(14) :3665–3680, 2018.

[196] Mikael Sørensen, Lieven De Lathauwer, Pierre Comon, Sylvie Icart, and Luc Deneire. Canonical polyadic decomposition with a columnwise orthonormal factor matrix. *SIAM Journal on Matrix Analysis and Applications*, 33(4) :1190–1213, 2012.

[197] Mikael Sørensen, Frederik Van Eeghem, and Lieven De Lathauwer. Blind multichannel deconvolution and convolutive extensions of canonical polyadic and block term decompositions. *IEEE Transactions on Signal Processing*, 65(15) :4132–4145, 2017.

[198] Alwin Stegeman. Degeneracy in candecomp/parafac explained for $p \times p \times 2$ arrays of rank $p + 1$ or higher. *Psychometrika*, 71 :483–501, 2006.

[199] Alwin Stegeman. Low-rank approximation of generic $p \times q \times 2$ arrays and diverging components in the candecomp/parafac model. *SIAM Journal on Matrix Analysis and Applications*, 30(3) :988–1007, 2008.

[200] Hans De Sterck and Killian Miller. An adaptive algebraic multigrid algorithm for low-rank canonical tensor decomposition. *SIAM Journal on Scientific Computing*, 35(1) :B1–B24, 2013.

[201] G. W. Stewart. Rank degeneracy. *SIAM Journal on Scientific and Statistical Computing*, 5(2) :403–413, 1984.

[202] G. W. Stewart. *Matrix Algorithms : Volume II : Eigensystems*. Society for Industrial and Applied Mathematics, 2001.

[203] V. Strassen. Rank and optimal computation of generic tensors. *Linear Algebra and its Applications*, 52-53 :645–685, July 1983.

[204] Jos M. F. ten Berge. Kruskal's polynomial for 2×2×2 arrays and a generalization to 2×n×n arrays. *Psychometrika*, 56 :631–636, 1991.

[205] Petr Tichavsky and Arie Yeredor. Fast approximate joint diagonalization incorporating weight matrices. *IEEE Transactions on Signal Processing*, 57(3) :878–891, 2009.

[206] Giorgio Tomasi and Rasmus Bro. A comparison of algorithms for fitting the PARAFAC model. *Computational Statistics & Data Analysis*, 50(7) :1700–1734, April 2006.

[207] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3) :279–311, 1966.

[208] André Uschmajew. Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM Journal on Matrix Analysis and Applications*, 33(2) :639–652, 2012.

[209] Shivakumar Vaithyanathan and Byron Dom. Model-based hierarchical clustering. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, page 599–608, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[210] Aad W Van der Vaart. *Asymptotic statistics*. Cambridge university press, 1998.

[211] A.-J. van der Veen and A. Paulraj. An analytical constant modulus algorithm. *IEEE Transactions on Signal Processing*, 44(5) :1136–1155, 1996.

[212] A.J. van der Veen, P.B. Ober, and E.F. Deprettere. Azimuth and elevation computation in high resolution doa estimation. *IEEE Transactions on Signal Processing*, 40(7) :1828–1832, 1992.

[213] Bart Vandereycken, P.-A. Absil, and Stefan Vandewalle. A riemannian geometry with complete geodesics for the set of positive semidefinite matrices of fixed rank, 2010.

[214] Nick Vannieuwenhoven. Condition numbers for the tensor rank decomposition. *Linear Algebra and its Applications*, 535 :35–86, 2017.

[215] Nick Vannieuwenhoven, Raf Vandebril, and Karl Meerbergen. A new truncation strategy for the higher-order singular value decomposition. *SIAM Journal on Scientific Computing*, 34(2) :A1027–A1052, 2012.

[216] M. A. O. Vasilescu and Demetri Terzopoulos. Multilinear analysis of image ensembles : Tensorfaces. In *Proceedings of the 7th European Conference on Computer Vision-Part I*, ECCV '02, page 447–460, Berlin, Heidelberg, 2002. Springer-Verlag.

[217] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. Tensorlab 3.0, Mar. 2016.

[218] R. Vollgraf and K. Obermayer. Quadratic optimization for simultaneous matrix diagonalization. *IEEE Transactions on Signal Processing*, 54(9) :3270–3278, 2006.

[219] Wenwu Wang, Saeid Sanei, and Jonathon Chambers. Penalty function-based joint diagonalization approach for convolutive blind separation of nonstationary sources, Jan 2005.

[220] H. Weyl. Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung). *Mathematische Annalen*, 71 :441–479, 1912.

[221] S. Winograd. On multiplication of 2×2 matrices. *Linear Algebra and its Applications*, 4(4) :381–388, 1971.

[222] Lei Xu and Michael I. Jordan. On Convergence Properties of the EM Algorithm for Gaussian Mixtures. *Neural Computation*, 8(1) :129–151, 01 1996.

[223] Lei Xu and Michael I Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural computation*, 8(1) :129–151, 1996.

[224] A. Yeredor. Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation. *IEEE Transactions on Signal Processing*, 50(7) :1545–1553, 2002.

[225] Ernesto Zacur, Matías Nicolás Bossa, and Salvador Olmos. Left-invariant riemannian geodesics on spatial transformation groups. *SIAM J. Imaging Sci.*, 7 :1503–1557, 2014.

[226] F.L. Zak. *Tangents and Secants of Algebraic Varieties*. Translations of Mathematical Monographs, AMS, Providence, RI, 1993.

[227] Xinzhen. Zhang, Chen. Ling, and Liqun. Qi. The best rank-1 approximation of a symmetric tensor and related spherical optimization problems. *SIAM Journal on Matrix Analysis and Applications*, 33(3) :806–821, 2012.

Algorithmes d'optimisation pour le problème d'approximation des décompositions en rang tensoriel: application au clustering en apprentissage automatique

Rima KHOUJA

Résumé

Les tenseurs sont une généralisation d'ordre supérieur des matrices. Ils apparaissent dans une myriade d'applications. La décomposition de rang de tenseur décompose le tenseur en une somme minimale de tenseurs simples de rang 1. En pratique, la présence de bruit dans les entrées du tenseur fait que le calcul d'une décomposition de petit rang approchée est plus pertinente que de son calcul exact. Ce problème est connu comme le problème d'approximation des décompositions en rang tensoriel. Dans cette thèse, nous étudions ce problème pour les tenseurs symétriques, c.à.d pour les tenseurs avec des entrées invariantes par les permutations d'indices. Nous considérons des tenseurs symétriques avec des valeurs complexes. Parsuite en utilisant le lien entre les tenseurs et les polynômes homogènes, ainsi que des techniques d'optimisation complexe, nous proposons une approche d'optimisation riemannienne et nous développons un algorithme de Newton riemannien et un algorithme de Gauss–Newton riemannien pour résoudre ce problème. Nous abordons également le problème de diagonalisation simultanée de matrices, qui est étroitement lié au problème de décomposition tensorielle. Nous considérons ce problème sous deux angles : la certification et l'approximation. Pour la première partie, nous développons une suite de type Newton à convergence quadratique locale, et nous proposons un teste de certification. Pour la deuxième partie, nous développons un algorithme de gradient conjugué riemannien qui calcule localement un faisceau de matrices simultanément diagonalisables approché. En combinant cet algorithme avec un problème linéaire de moindres carrés, nous introduisons un algorithme d'optimisation alterné qui calcule une approximation de la décomposition pour les tenseurs tridimensionnels, quand le rang d'approximation est supérieur à la dimension de deux premiers modes. Enfin, en se basant sur les deux approches : tenseurs symétriques et diagonalisation simultanée de matrices, nous abordons le problème de clustering en apprentissage automatique pour les modèles de mélanges de Gaussiènne sphériques. Nous utilisons ces méthodes pour implémenter la méthode des moments, afin de fournir un bon point initial pour l'algorithme de maximisation de vraissemblance.

Mots-clés : Tenseurs, algorithmes d'optimisation, apprentissage automatique, clustering, optimisation riemannienne, diagonalisation simultanée de matrices, mélanges Gaussiènnes, optimisation complex, variétés différentielles.