ROYAUME DU MAROC
Université Mohammed V - RABAT -
جامعة محمد الخامس – الرباط –
Faculté des sciences
كلية العلوم
CENTRE D'ETUDES DOCTORALES - SCIENCES ET TECHNOLOGIES

N° d'ordre 3507

THESE

En vue de l'obtention du : ***DOCTORAT***

Structure de Recherche : Laboratoire de Recherche Informatique et Télécommunications
Discipline : Sciences de l'ingénieur
Spécialité : Informatique et Télécommunications

Présentée et soutenue le 15-07-2021 par :

Marouane NAZIH

Les méthodes d'optimisation proximale pour la décomposition canonique polyadique : Conceptions et implémentations algorithmiques et applications aux traitements des signaux.

Mounir GHOGHO	*PES*	*Université Internationale de Rabat.*	*Président*
Abdellah ADIB	*PES*	*Faculté des sciences et techniques Mohammedia-Université Hassan II.*	*Rapporteur/Examinateur*
Benayad NSIRI	*PES*	*École Nationale Supérieure des Arts et Métiers-Université Mohammed V, Rabat.*	*Rapporteur/Examinateur*
Mohammed OUMSIS	*PES*	*École Supérieure de Technologie Salé Université Mohammed V de Rabat.*	*Rapporteur/Examinateur*
Pierre COMON	*Directeur de Recherche*	*Centre national de la recherche scientifique à Grenoble-Université Grenoble Alpes.*	*Examinateur*
Khalid MINAOUI	*PES*	*Faculté des Sciences Université Mohammed V,Rabat.*	*Directeur de thèse*

Année Universitaire : 2020/2021

À ma mère et mon père bien aimés Lhaja Fatima et Lhaj Abdellah,
Je dédie ce travail de thèse aux 2 personnes
qui me sont le plus chères au monde
pour avoir fait de moi ce que je suis.

À mon adorable belle-mère : Khalti Latifa

À ma chère fiancée Manal,
pour son soutien et ses encouragements
durant toutes ces longues années.

À mon cher frère : Hamid
À ma grande soeur : Siham
À ma petite soeur : Salma

Remerciements

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire de Recherche en Informatique et Télécommunications (LRIT) de la Faculté des Sciences de Rabat (FSR)-Université Mohammed V de Rabat au Maroc sous la direction de **Mr Khalid MINAOUI**, Professeur d'Enseignement Supérieur à la Faculté des Sciences de Rabat.

En premier lieu, je tiens à remercier mon directeur de thèse **Mr Khalid MINAOUI**, Professeur d'Enseignement Supérieur à la Faculté des Sciences de Rabat avec qui j'ai travaillé depuis mon stage de Master. Dans une ambiance toujours décontractée mais néanmoins studieuse, il a bien guidé mes premiers pas dans le monde de la Recherche. Et je réitère mes remerciement pour sa grande disponibilité, son aide précieuse, ainsi que pour ses efforts qu'il a prodigués pour l'accomplissement de ce travail de thèse.

Je tiens à remercier **Mr Mounir GHOGHO**, Professeur d'Enseignement Supérieur à l'Université Internationale de Rabat, d'avoir accepté de présider le jury de ma thèse.

Mers remerciements vont également à **Mr Abdellah ADIB**, Professeur d'Enseignement Supérieur à la faculté des sciences et techniques de Mohammedia, d'avoir accepté de rapporter ce mémoire de thèse.

Je tiens aussi à remercier **Mr Benayad NSIRI**, Professeur d'Enseignement Supérieur à l'École Nationale Supérieure d'Arts et Métiers de Rabat, d'avoir accepté de juger la qualité de mon travail en tant que rapporteur.

Je tiens aussi à remercier **Mr Mohammed OUMSIS**, Professeur d'Enseignement Supérieur à l'École Supérieure de Technologie de Salé, d'avoir accepté de juger la qualité de mon travail en tant que rapporteur.

Je voudrais également remercier **Mr Pierre COMON**, Directeur de Recherche CNRS à l'Université Grenoble, d'avoir accepté d'examiner mon travail.

Je remercie mes amis qui m'ont apporté leur soutien et leurs encouragements tout au long de cette aventure. Un merci spécial à Ayoub LAHMIDI et Abderrahim MORJANE pour être de vrais amis, merci pour vos beaux cœurs et votre soutien continu, merci pour les discussions inspirantes et tous les moments amusants que nous avons passés ensemble.

Merci à tous les membres du LRIT avec qui j'ai partagé des souvenirs inoubliables.

Je garde le meilleur pour la fin, ma famille qui a supporté toutes les difficultés morales et matérielles pour me soutenir au terme de mes études. J'adresse ma profonde gratitude et mon immense reconnaissance à mes raisons d'être, ma mère et mon père, qui m'ont éduqué et orienté. Merci de m'avoir encouragé et soutenu dans mes choix. Nul mot et nulles expressions refléteront le grand amour et la profonde gratitude que je porte pour vous.

On dit que les mots du cœur sont merveilleux, tels étaient les mots de mon unique frère Hamid, tu es le grand homme sur lequel nous pouvons compter tous, fier de toi, très reconnaissant de la pertinence de tes conseils, de ton soutien.

Résumé

Dans le cadre du travail réalisé, nous nous sommes intéressés à la conception et le développement d'algorithmes de décomposition des tableaux de données multidimensionnels particuliers appelés tenseurs. Parmi les décompositions de ces tenseurs, nous nous intéressons à la décomposition Canonique Polyadique (CP). Différentes méthodes développées dans la littérature souffrent de certaines lacunes selon la nature des données traitées, la précision, le temps de calcul...

L'objectif principal porte sur le développement de méthodes pour l'aide à l'estimation correcte d'un ensemble de matrices de la décomposition CP. Pour atteindre cet objectif, nous proposons cinq algorithmes pour le calculer de la décomposition CP dans le cas des données complexes et non-négative basé sur les algorithmes proximaux. Les résultats ont démontré le bon comportement de ces cinq algorithmes, ainsi qu'un meilleur compromis entre la précision et la vitesse de convergence par rapport aux algorithmes de la littérature.

Mots clés : Traitement du signal, Tenseur, Canonique polyadique, Algorithmes proximaux.

Abstract

In this work, we are interested in the design and development of algorithms for decomposing particular multidimensional data arrays called tensors. Among the decompositions of these tensors, we are interested in the Canonical Polyadic (CP) decomposition. Different methods have been developed in the literature suffer from certain shortcomings depending on the nature of the data processed, the accuracy, the computation time...

The main objective is to develop new methods for the correct estimation of a set of matrices for the CP decomposition. To achieve this objective, we propose five algorithms for computing the CP decomposition in the case of complex and non-negative data based on proximal algorithms. The results have demonstrated the good behavior of these five algorithms, as well as a better compromise between accuracy and convergence speed compared to the CP decomposition algorithms in the literature.

Keywords : Signal processing, Tensor, Canonical polyadic, Proximal algorithms.

Table des matières

Liste des abréviations

PCA	*Principal Component Analysis*
ICA	*Independent Component Analysis*
ALS	*Alternating Least Squares*
CG	*Conjugate Gradient*
CPD	*Canonical Polyadic Decomposition*
NCP	*Non-negative Canonical Polyadic*
EVD	*Eigenvalue Decomposition*
PGD	*Projected Gradient Descent*
LLR	*Logarithm Likelihood Ratio*
LM	*Levenberg-Marquardt*
LS	*Line Search*
NLCG	*Non Linear Conjugate Gradient*
PN	*Pseudo-Noise*
SNR	*Signal-to-Noise Ratio*
SVD	*Singular Values Decomposition*
HOSVD	*Higher-order singular value decomposition*
PARAFAC	*Parallel Factor Analysis*
PP	*Proximal Point*
FB	*Forward-Backward*
PG	*Proximal Gradient*
APG	*Accelerated Proximal Gradient*

Liste des figures

Liste des tableaux

Liste des algorithmes

Chapitre

1 INTRODUCTION GÉNÉRALE

1.1 Contexte général

Les tenseurs intéressent les mathématiciens et les physiciens depuis le XIXe siècle. En physique, ils offrent un langage pratique pour exprimer certaines lois naturelles. Un exemple célèbre est celui établi par la théorie de la relativité générale d'Eistein, dont les équations fondamentales sont exprimées en termes de tenseurs. À partir des années 60, un intérêt croissant pour les tenseurs a également été observé dans de nombreuses autres communautés scientifiques et d'ingénieurs, en partie sous la perspective des travaux initiaux en psychométrie ayant appliqué des techniques tensorielles à des fins d'analyse de données [Tucker, 1963, Tucker *et al.*, 1964, Tucker, 1966, Harshman *et al.*, 1970, Carroll et Chang, 1970]. Notamment, des techniques de séparation aveugle des sources ont été développées dans les années 90 en exploitant la structure tensorielle des cumulants d'ordre supérieur [Cardoso, 1990a, Cardoso, 1990b, Cardoso, 1991, Comon, 1994], tandis que de nombreux travaux utilisant des modèles tensoriels ont également fait surface dans la littérature chimiométrique [Appellof et Davidson, 1981, Bro *et al.*, 1997, Bro, 1998, Leurgans et Ross, 1992]. Aujourd'hui, les applications des tenseurs, dont la liste ne cesse de s'accroître, englobent des problèmes de télécommunications [Sidiropoulos *et al.*, 2000b, de Almeida *et al.*, 2007, Fernandes *et al.*, 2008, Fernandes *et al.*, 2011, Favier et de Almeida, 2014], de traitement du signal [De Lathauwer, 1997, Sidiropoulos *et al.*, 2000a, Muti et Bourennane, 2005, Ozerov *et al.*, 2011, Cichocki *et al.*, 2015], de vision par ordinateur [Shashua et Hazan, 2005, Liu *et al.*, 2012, Cheng *et al.*, 2014, Marmarelis *et al.*, 2013], de génie biomédical [Becker *et al.*, 2014, Roohi *et al.*, 2016, Ribeiro *et al.*, 2016, Mahyari *et al.*, 2016], de modélisation et d'identification de systèmes dynamiques [Kibangou et Favier, 2006, Favier et Kibangou, 2009, Favier *et al.*, 2012, Burt et Goulart, 2013] et d'exploration de données [Bader *et al.*, 2008, Kolda et Sun, 2008, Mørup, 2011, Papalexakis *et al.*, 2016].

Ce regain d'intérêt, relativement récent, pour les méthodes tensorielles s'explique principalement par leur capacité à exploiter une structure de problème supplémentaire par rapport aux méthodes matricielles plus traditionnelles. L'estimation des spectres d'excitation/émission à partir de données de fluorescence en chimiométrie au moyen de

techniques de décomposition tensorielle d'ordre élevé [Bro *et al.*, 1997, Bro, 1998] est un exemple stéréotypé d'une telle supériorité, qui découle dans ce cas de l'unicité de ces quantités sous des conditions beaucoup plus faibles que celles requises lors de l'utilisation des décompositions matricielles. Plus généralement, la même idée s'applique aux problèmes inverses dont les quantités recherchées constituent des modèles multilinéaires, comme, par exemple, dans l'estimation des directions d'arrivée dans le traitement des réseaux d'antennes [Sahnoun et Comon, 2015]. Les modèles tensoriels sont également utiles dans de nombreux autres problèmes car ils fournissent souvent des représentations précises et parcimonieuses des données multidimensionnelles du monde réel, un fait qui peut être exploité pour développer des techniques de stockage, de calcul et d'estimation efficaces. Cette ligne de pensée est adoptée, par exemple, pour la compression de données [Dauwels *et al.*, 2011, Ballester-Ripoll *et al.*, 2015], la modélisation des systèmes non linéaires [Burt et Goulart, 2013, Favier *et al.*, 2012] et la récupération des tenseurs à faible rang [Cheng *et al.*, 2016, Rauhut *et al.*, 2017].

Parallèlement au recours croissant aux méthodes tensorielles dans les sciences appliquées, de nouvelles directions de recherche passionnantes liées aux propriétés des tenseurs, aux représentations tensorielles et à l'algèbre multilinéaire ont fleuri. Des progrès significatifs ont été réalisés grâce à des efforts considérables ces derniers temps [Hackbusch, 2012, Domanov et De Lathauwer, 2013a, Domanov et De Lathauwer, 2013b, Oeding et Ottaviani, 2013, Friedland et Lim, 2018, Blekherman et Teitler, 2015, Chiantini *et al.*, 2017, Hackbusch *et al.*, 2017, Qi *et al.*, 2016], mais certaines propriétés fondamentales des tenseurs d'ordre supérieur sont encore très peu explorées par les chercheurs.

1.2 Problématiques et objectives

L'analyse de signaux multidimensionnels est devenue un outil important dans de nombreux problèmes de traitement du signal. Ce type d'analyse permet de profiter de différentes diversités d'un signal afin d'en extraire au mieux des informations utiles et inhérentes. Ces signaux peuvent être directement stockés après la mesure dans des tableaux à plusieurs dimensions lorsque le modèle physique le permet. Ces tableaux peuvent aussi être construits en appliquant différents opérateurs d'analyse tels que les statistiques d'ordre supérieur ou les représentations temps-fréquence.

Ainsi, les travaux présentés dans cette thèse portent sur la conception et le développement d'algorithmes de décomposition de tableaux de données multidimensionnels particuliers appelés tenseurs. La définition du terme tenseur dépendra généralement de domaine scientifique dans lequel il est utilisé. Dans le cas général, un tenseur est traité comme une entité mathématique qui jouit de la propriété de multilinéarité après changement du système de coordonnées [Comon, 2002].

Dans le contexte de notre thèse, un tenseur d'ordre N représente un tableau multidimensionnel dont chaque élément est accessible via N indices. Un tenseur d'ordre un

est un vecteur, un tenseur d'ordre deux est une matrice alors qu'un tenseur d'ordre zéro est un scalaire. L'analyse des tenseurs d'ordre supérieur à deux est dite algèbre multilinéaire. Tout comme l'algèbre linéaire qui est bâtie sur le concept de vecteur et développe la théorie des espaces vectoriels, l'algèbre multilinéaire est l'algèbre des tenseurs d'ordre supérieur à deux qui est bâtie sur le concept de tenseur et développe la théorie des espaces tensoriels.

Nous s'intéressons aux tenseurs d'ordre supérieur à deux car ils possèdent des propriétés qui ne sont pas valables pour les matrices et les vecteurs. La plupart des exemples proposés dans ce manuscrit concerneront les tenseurs d'ordre trois ce qui nous permet de bien appréhender les caractéristiques propres aux tenseurs tout en étant faciles à représenter sous forme de cube comme dans la figure 1.1.

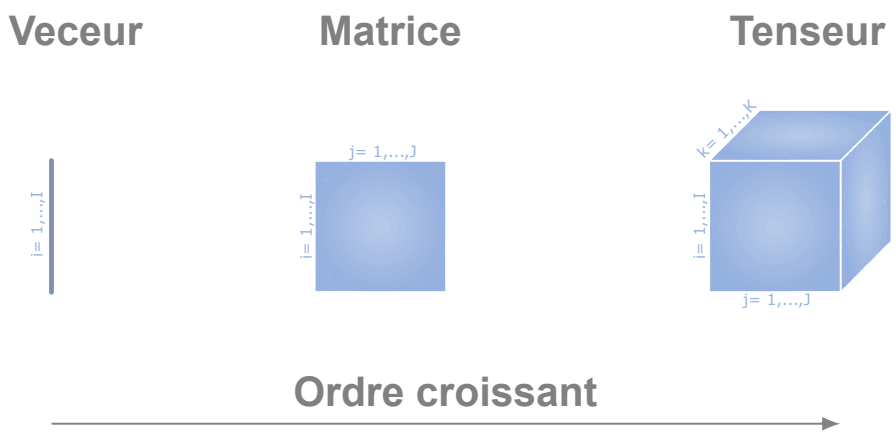


FIGURE 1.1 – Alors qu'un vecteur et une matrice représentent des ensembles de données à direction unique et à direction double, respectivement, un tenseur d'ordre supérieur représente un ensemble de données à direction multiple.

Les décompositions tensorielles jouent un rôle important dans plusieurs problèmes de traitement du signal comme la psychométrie [Harshman *et al.*, 1970], la chimiométrie [Appellof et Davidson, 1981], la vision par ordinateur [Vasilescu et Terzopoulos, 2002], la fouille de données [Acar *et al.*, 2005], les neurosciences [Beckmann et Smith, 2005] ou les télécommunications numériques [Sidiropoulos *et al.*, 2000b]. Un résumé des différentes décompositions tensorielles et de leurs domaines d'application peut être trouvé dans [Kolda et Bader, 2009, Cichocki *et al.*, 2015].

Les premiers travaux sur les décompositions tensorielles sont attribués à Hitchcock [Hitchcock, 1928] en 1927. Par la suite, ces travaux ont été popularisés par Tucker [Tucker, 1963, Tucker, 1966] en 1963, puis par Caroll et Chang [Carroll et Chang, 1970] et Harshman [Harshman *et al.*, 1970] en 1970. La décomposition tensorielle multilinéaire la plus générale est la décomposition de Tucker. Cette dernière considère qu'un tenseur peut se décomposer à l'aide d'un tenseur coeur et de plusieurs matrices facteurs de

manière multilinéaire. Dans le cas où les matrices facteurs sont orthogonales, cette décomposition prend le nom de la Décomposition en valeurs singulières d'ordre supérieur (HOSVD) [De Lathauwer *et al.*, 2000a]. Lorsque le tenseur coeur est diagonal, il s'agit alors de la décomposition Canonique Polyadique (CP). Celle-ci est aussi connue sous le nom de PARAFAC (en anglais : Parallel Factor Analysis). La HOSVD est généralement utilisée pour des problèmes de compression [Savas et Eldén, 2007]. La décomposition CP est quant à elle très utilisée dans les problèmes de séparation de sources et d'estimation de mélanges [Comon et Jutten, 2010]. Elle permet, en effet, de retrouver sur les colonnes de ses matrices facteurs une ou plusieurs estimations des signaux sources et/ou du mélange. L'intérêt d'utiliser la décomposition CP pour des problèmes de séparation de sources est qu'elle possède des conditions d'unicité souvent vérifiées en pratique et qu'elle permet l'estimation de mélanges sous-déterminés [Comon et Rajih, 2006].

Nous nous focalisons principalement sur : la décomposition Canonique Polyadique (CP) des tenseurs. Le but général des algorithmes de la décomposition tensorielle CP est d'estimer un ensemble de matrices à partir d'un tableau de données multidimensionnel en minimisant une fonction coût judicieusement choisie. Ces algorithmes peuvent être directement utilisés pour la séparation aveugle de sources issues d'un mélange linéaire. Dans ce cas, l'objectif est de trouver les signaux sources à partir d'observations caractérisant un mélange inconnu de sources. Le problème de séparation de sources modélise de nombreux systèmes multicapteurs comme des réseaux d'antennes, de microphones ou de capteurs chimiques. Il peut donc être appliqué aux domaines de l'astronomie [Delabrouille *et al.*, 2003, Moussaoui *et al.*, 2008], de l'audio [Chabriel et Barrère, 2005, Koldovsky et Tichavskỳ, 2010], du biomédical [Congedo *et al.*, 2008, De Lathauwer *et al.*, 1995b, Hérault *et al.*, 1985, He *et al.*, 2006, Albera *et al.*, 2012], de la sismique [Paulus et Mars, 2006, Thirion, 1995], des télécommunications numériques [Sidiropoulos *et al.*, 2000b, Nion et De Lathauwer, 2008, Ferréol *et al.*, 2004, Chevalier *et al.*, 2004, Castella et Moreau, 2009, Castella *et al.*, 2006a] ou encore de la spectroscopie de fluorescence [Smilde *et al.*, 2005, Bro, 1998, Bro *et al.*, 1997, Luciani, 2007, Luciani et Albera, 2014].

Il existe de nombreux algorithmes pour la décomposition CP. Le principe général de ces algorithmes consiste à minimiser l'erreur quadratique entre les données observées et les données estimées. Nous pouvons distinguer deux grandes familles d'algorithmes permettant de minimiser cette erreur : les algorithmes de type alterné (alternating way) et les algorithmes simultanés (all-at-once way).

La plupart de ces algorithmes souffrent généralement de problèmes de convergence (minima locaux, faible vitesse de convergence ou coût de calcul élevé). De plus, la majorité de ces algorithmes sont très sensibles à la surestimation des facteurs (lorsque le nombre de sources est inconnu) et aux facteurs corrélés, ce qui entraîne une convergence lente ou même une absence de convergence de ces algorithmes en raison des phénomènes de

dégénération du tenseur. Ces phénomènes ont été bien classés par "Richard Harshman" [Harshman, 2004] dans les trois cas suivants :

— Goulot d'étranglement (Bottleneck) : Un goulot d'étranglement survient lorsque deux ou plusieurs facteurs dans l'un des modes (directions ou dimensions) du tenseur sont presque colinéaires [Rajih *et al.*, 2008].

— Marécage (Swamp) : Le phénomène de marécage se produit lorsque tous les modes présentent au moins deux facteurs quasi colinéaires [Rajih *et al.*, 2008, Rayens et Mitchell, 1997], ce qui peut être considéré comme un cas général de goulot d'étranglement.

— Dégénérescences de la CP (CP-degeneracies) : Les dégénérescences de CP peuvent être considérées comme un cas particulier de phénomène de marécage, où certains des facteurs se rapprochent et, en même temps, ont tendance à s'annuler, ce qui conduit à une meilleure qualité de l'ajustement [Harshman, 2004, Paatero, 2000].

En addition, dans certaines applications comme la spectroscopie de fluorescence [Smilde *et al.*, 2005, Bro *et al.*, 1997] ou l'imagerie hyperspectrale [Zhang *et al.*, 2008], les matrices facteurs et par conséquent les termes du tenseur qu'elles forment doivent contenir des valeurs non-négatives. La décomposition CP est alors appelée décomposition CP non-négative (NCP). Cette caractéristique permet de contraindre la valeur des matrices facteurs estimées et a donné lieu à plusieurs autres méthodes de décomposition CP sous contrainte de non-négativité. De nombreuses méthodes pour la décomposition NCP ont été présentées dans [Cichocki *et al.*, 2009].

Pour palier ces problèmes, des travaux récents [Farias *et al.*, 2018, Sahnoun et Comon, 2015] ont démontré que l'introduction d'une contrainte sur les cohérences des matrices facteurs permet d'assurer l'existence d'une meilleure approximation de faible rang tout en empêchant les phénomènes décrits ci-dessus de se produire. En plus, prendre en compte les contraintes offertes par une décomposition CP non-négative permet de rendre les algorithmes plus robustes à la surestimation du nombre de facteurs ou d'améliorer leurs performances lorsque les facteurs sont corrélés.

Dans le but d'améliorer les performances des algorithmes de la décomposition CP au niveau de précision et de vitesse de convergence, ainsi pour surmonter les situations difficiles là où les phénomènes de marécage et de goulots d'étranglement apparaissent. Nous proposons dans cette thèse trois nouveaux algorithmes fonctionnant sur des données réelles et complexes basés sur les méthodes proximales [Bauschke *et al.*, 2011a, Parikh et Boyd, 2014], ces dernières étant aujourd'hui des outils d'optimisation fiables et puissants, conduisant à une variété d'algorithmes proximaux, tels que l'algorithme de

Point Proximal (PP), l'algorithme Avant-Arrière (FB), l'algorithme de Gradient Proximal (PG), l'algorithme de Gradient Proximal Accéléré (APG), et plusieurs autres incluant la linéarisation et/ou le fractionnement. Les stratégies proposées permettront ainsi d'obtenir de meilleures performances que celles fournies par les algorithmes existants.

Par ailleurs, il n'existe à ce jour, à notre meilleure connaissance, aucun algorithme proximal pour estimer simultanément les matrices facteurs de la décomposition CP non-négative. Ce constat nous a encouragé à réaliser deux nouveaux algorithmes fonctionnant cette fois sur des données non-négatives, en se basant sur la méthode du gradient proximal accéléré dans lesquelles nous imposerons la contrainte non-négative à travers une simple stratégie de projection.

1.3 Plan de la thèse

Ce manuscrit est donc organisé de la manière suivante : À la fin de ce chapitre, nous introduirons les notations et les définitions d'algèbre multilinéaire qui seront utilisées dans ce document. La définition mathématique d'un tenseur d'ordre supérieur, les propriétés propres à leur utilisation ainsi que les opérateurs tensoriels manipulés dans ce manuscrit, seront ensuite exposés.

Dans le chapitre 2, nous commencerons par introduire un ensemble de décompositions matricielles. Ensuite, nous présenterons quelques décompositions tensorielles. Une plus grande attention sera accordée aux décompositions d'un tenseur de troisième ordre, puisque ça permettra d'appréhender et de présenter les caractéristiques propres de ces décompositions. Dans certains cas, la généralisation au $N^{\text{ème}}$ ordre sera également donnée. À la fin de ce chapitre, une étude plus approfondie sera accordée à la décomposition CP puisqu'elle fait l'objective de toutes nos contributions, et dans laquelle nous introduirons une décomposition tensorielle basée sur la décomposition CP qui resolve le problème de l'indétermination de l'échelle, dont nous prouverons l'existence de cette décomposition et aussi nous étudierons son unicité.

Le chapitre 3 est consacré à une étude bibliographique sur les problèmes d'optimisation, ainsi que les différents types de méthodes d'optimisation. Ensuite, nous introduisons les différentes méthodes itératives qui ont été utilisés pour optimiser la décomposition tensorielle CP sans contraintes. Parmi ces méthodes, nous distinguons deux grands types d'algorithmes : "les algorithmes alternés" et "les algorithmes simultanés". Dans un premier temps, nous allons présenter les méthodes alternées dans lesquelles nous étudierons plus en détail l'algorithme des moindres carrés alternés (ALS), fréquemment utilisé dans la littérature pour optimiser la décomposition tensorielle CP, puis l'insertion d'une recherche linéaire dans cet algorithme sera montré. Ensuite, nous nous attacherons à une description plus spécifique des méthodes simultanées qui ont été implémentées pour la résolution du problème de la décomposition CP sans contrainte (aussi parfois

appelées méthodes d'optimisation directe). Cela comprend les méthodes basées sur le gradient, les méthodes de Newton et les méthodes utilisant des directions conjuguées. Avant de conclure avec un bilan sur le travail présenté dans ce chapitre, nous allons exposer également une étude sur les méthodes du choix du pas de recherche.

Le chapitre 4 est dédiée aux algorithmes pour la résolution du problème de la décomposition CP avec contraintes. Dans la première partie, nous introduisons des algorithmes basés sur la descente de gradient pour minimiser la fonction de coût sous une contrainte d'égalité, tout en indiquant les deux façons pour calculer le facteur d'échelle. Quant à la deuxième partie, elle consiste à la proposition d'algorithmes d'optimisation qui minimisent cette fois-ci notre fonction de coût sous deux contraintes, d'égalité et d'inégalité. Ceci donne lieu alors naturellement aux algorithmes de barrière et les algorithmes proximaux. Le début de cette deuxième partie est dédié à la formulation du problème de la décomposition CP sous les deux contraintes "de cohérence" et "de facteur d'échelle optimal". Ensuite, une étude plus détaillée sur les méthodes de barrière sera menée. Par la suite, nous allons présenter plus en détail le principe général des méthodes proximales ainsi que leur fiabilité et leur adéquation pour notre problème de la décomposition CP avec contraintes. Avant de conclure avec un bilan sur le travail présenté dans ce chapitre, nous allons exposer les différentes stratégies que nous avons mises en oeuvre et nous allons comparer les trois algorithmes proximaux proposés à ceux de la littérature.

Dans le chapitre 5, notre concentration sera sur la résolution de la décomposition Canonique Polyadique Non-négative (NCP). Le début de ce chapitre est dédié à la formulation du problème de la décomposition NCP. Ensuite, nous allons introduire les différentes méthodes qui ont été utilisés pour optimiser la décomposition CP sous la contrainte de non-négativité. Par la suite, nous allons exposer les deux différentes stratégies dérivées de la méthode du gradient proximal accéléré que nous avons mis en oeuvre pour la résolution du problème de la NCP. Enfin, avant de conclure avec un bilan sur le travail présenté dans ce chapitre, nous allons comparer les deux algorithmes proposés à ceux de la NCP dans la littérature.

Pour clôturer ce mémoire, une conclusion générale présente un bilan sur l'ensemble des travaux réalisés. Ensuite, nous allons dresser quelques perspectives et améliorations qui peuvent être une suite logique des travaux réalisés et présentés dans ce mémoire.

Enfin, les annexes A et B ont servi comme compléments des concepts présentés dans le chapitre 3 et chapitre 4 respectivement. Quant à l'annexe C, elle liste la production scientifique résultante de l'ensemble des travaux de thèse réalisés.

1.4 Notations et outils mathématiques

1.4.1 Notations

Commençons d'abord par l'introduction des notations essentielles qui seront utilisées dans ce document. $\mathbb{R}$ désigne l'ensemble des nombres réels, alors que $\mathbb{C}$ est l'ensemble des nombres complexes. Les tenseurs seront notés par des lettres calligraphiques $\mathcal{T}$, les matrices par des lettres majuscules grasses $\mathbf{A}$ les vecteurs par des minuscules grasses $\mathbf{a}$ et les scalaires par des minuscules en italiques a. De plus, la $p^{\text{ème}}$ colonne de la matrice $\mathbf{A}$ est notée $\mathbf{a}_p$ et le $p^{\text{ème}}$ élément d'un vecteur $\mathbf{a}$ est a_p. L'élément d'indice $\{i, j\}$ de la matrice $\mathbf{A}$ sera noté a_{ij}, alors que l'élément $\{i, j, k\}$ d'un tenseur d'ordre trois $\mathcal{T}$ se notera $t_{i,j,k}$. $\mathbf{1}$ représentera un vecteur contenant des uns, et $\mathbf{I}$ la matrice identité.

Le conjugué d'un élément a sera noté a^*, alors que son module $|a|$. Les opérateurs matriciels classiques seront notés de la manière suivante :

- $\mathbf{A}^{-1}$ pour l'opérateur inverse.
- $\mathbf{A}^*$ pour l'opérateur conjugué.
- $\mathbf{A}^T$ pour l'opérateur transposé.
- $\mathbf{A}^H$ pour l'opérateur transposé-conjugué ou transposé hermitien.
- $\mathbf{A}^\dagger$ pour la pseudo-inverse au sens de Moore-Penrose.

Dans ce document, nous utiliserons aussi l'opérateur *diag* qui permet de ranger les éléments d'un vecteur sur la diagonale d'une matrice :

$$diag\{\mathbf{a}\} = \begin{pmatrix} a_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_R \end{pmatrix} \tag{1.1}$$

1.4.2 Définitions et propriété mathématiques

1.4.2.1 Définition d'un tenseur

Les outils d'algèbre linéaire et plus particulièrement les vecteurs et les matrices, permettent d'indexer des données à l'aide d'un ou deux indices. Lorsque le nombre de dimensions d'un système est supérieur à deux, il est naturel d'introduire un nouvel outil qui est le tenseur. Le terme tenseur peut avoir différentes significations selon le domaine scientifique. Ici, un tenseur définit une application multilinéaire. Lorsque les bases des espaces vectoriels sont fixées, un tenseur correspond à un tableau multidimensionnel

[Kolda et Bader, 2009, McCullagh, 2018]. Ainsi dans la suite de cette thèse, un tenseur sera assimilé à un tableau de nombres multidimensionnel. La définition d'un tenseur est donnée en détails par [De Silva et Lim, 2008], qui rappelle que ce dernier ne doit pas être confondu avec le tenseur utilisé en physique et en ingénierie. Ces derniers sont souvent désignés par le tenseur champ (field tensor) en mathématique.

Généralement, Les tenseurs sont des objets mathématiques issus de l'algèbre multilinéaire permettant de généraliser les scalaires et les vecteurs. Nous considérons qu'un tenseur d'ordre N représente un tableau multidimensionnel dont chaque élément est accessible via N indices $\{I_1, ..., I_N\}$. Ce tenseur est noté par :

$$\mathcal{T} \in \mathbb{C}^{I_1 \times I_2 \times ... \times I_N}, \tag{1.2}$$

et ses éléments sont notés : $t_{I_1,...,I_N}$. Chaque dimension du tenseur est appelée «n-mode» en référence au $n^{\text{ème}}$ indice du tenseur [Smilde *et al.*, 2005, De Lathauwer *et al.*, 2000b, Kroonenberg, 1983]. Un tenseur d'ordre **zéro** est un scalaire, un tenseur d'ordre **un** est un vecteur, alors qu'un tenseur d'ordre **deux** est une matrice. Un tenseur d'ordre **trois** peut être resprésenté par une somme de produits tensoriels entre trois vecteurs, donc il peut être généré au moyen de trois vecteurs et représenté par un tableau à trois dimensions (figure 1.2). L'ordre d'un tenseur désigne alors le nombre de ces dimensions. La figure 1.3 illustre un tenseur d'ordre 3 dont les dimensions successives sont $I = 2$, $J = 2$ et $K = 2$.

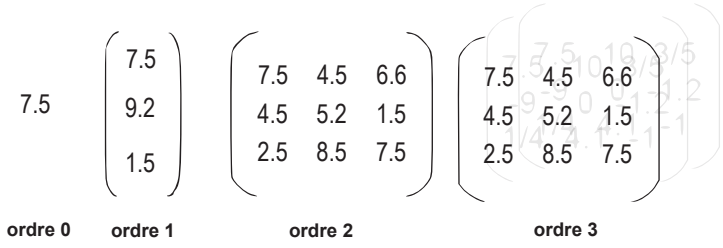


FIGURE 1.2 – les 4 premiers ordres d'un tenseur.

1.4.2.2 *Les tranches et les tubes*

Pour une matrice, on trouve les lignes et les colonnes. Ainsi, par exemple, la $i^{\text{ème}}$ ligne d'une matrice $\mathbf{A}$ est notée $\mathbf{a}_{i:}$ et la $j^{\text{ème}}$ colonne est notée $\mathbf{a}_{j:}$ cette dernière peut être aussi désignée par $\mathbf{a}_j$, la ponctuation « : » représente le *full rang* d'un indice donné. Cette notation est étendue aux tenseurs d'ordre supérieur.

Considérons un tenseur $\mathcal{T}$ d'ordre 3. En spécifiant un indice, on obtient une tranche (une matrice) et en spécifiant deux indices, on obtient une fibre (un vecteur). La figure 1.4 illustre les tranches horizontales $\mathcal{T}_{i,:,:}$, les tranches latérales $\mathcal{T}_{:,j,:}$ et les tranches frontales $\mathcal{T}_{:,:,k}$ (noté parfois simplement par T_k). De la même façon, La fibre d'un tenseur est similaire

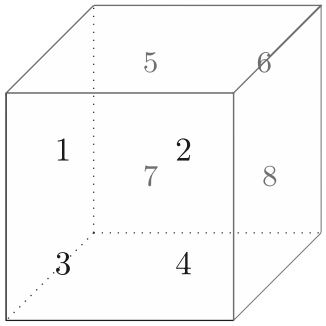


FIGURE 1.3 – Un tenseur d'ordre 3 et de dimension $2 \times 2 \times 2$.

à une tranche mais en fixant l'ensemble des indices sauf un seul. La figure 1.5 présente les fibres en colonnes $\mathcal{T}_{:,j,k}$, les fibres en lignes $\mathcal{T}_{i,:,k}$ et en tubes $\mathcal{T}_{i,j,:}$.

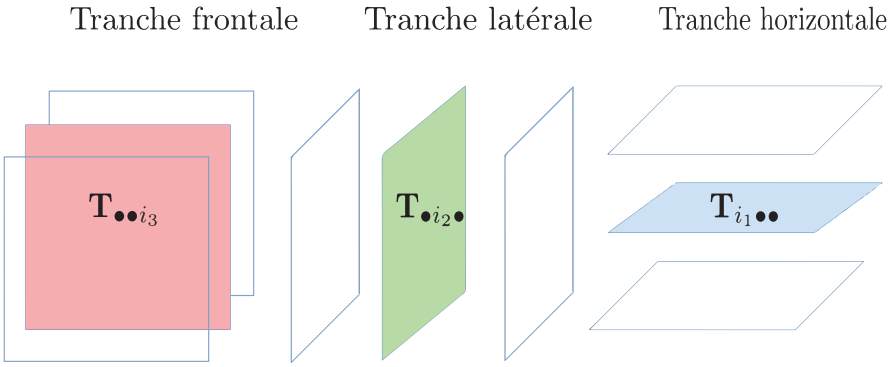


FIGURE 1.4 – Les tranches pour un tenseur d'ordre 3.

1.4.2.3 *Déploiement d'un tenseur*

Afin d'étudier les propriétés des données multidimensionnelles dans une direction n-modale particulière, définissons la matrice dépliante [Bader et Kolda, 2006, Kiers, 2000] dans le n-mode du tenseur $\mathcal{T}$, notée T_n. Typiquement, le déploiement selon le n-mode d'un tenseur revient à concaténer toutes les fibres associées à ce mode en colonnes de façon à obtenir une matrice. La matrice dépliante dans le n-mode d'un tenseur $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ est notée par $T_{(n)}$.

Il existe trois façons de déplier un tenseur en matrice selon trois modes. Pour mieux comprendre, considérons un tenseur $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$, le déploiement dans le premier mode mène à une matrice $\mathbf{T}_{(1)}^{I \times KJ}$ de taille $I \times JK$. Les déploiements dans le mode deux et trois mènent respectivement à une matrice $\mathbf{T}_{(2)}^{J \times KI}$ de taille $J \times KI$ et $\mathbf{T}_{(3)}^{K \times JI}$ de taille $K \times JI$,

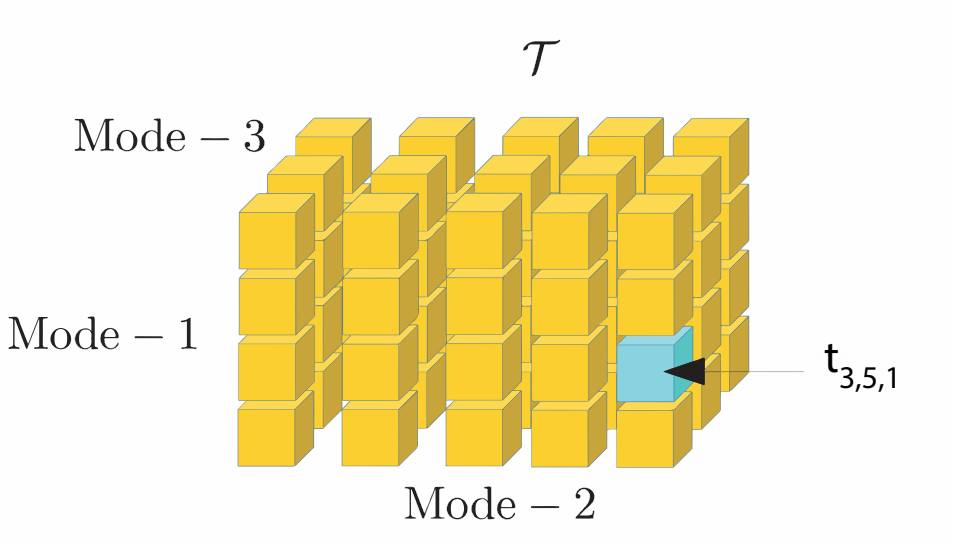


FIGURE 1.5 – Les fibres pour un tenseur d'ordre 3.

(la figure 1.6 illustre le déploiement du tenseur $\mathcal{T}$ selon les 3-modes) et qui sont défini par :

$$\mathbf{T}_{(1)}(i,p) = \mathcal{T}(i,j,k) \qquad \text{où p = j + (k - 1)(J)} \tag{1.3}$$

$$\mathbf{T}_{(2)}(j,p) = \mathcal{T}(i,j,k) \qquad \text{où p = i + (k - 1)(I)} \tag{1.4}$$

$$\mathbf{T}_{(3)}(k,p) = \mathcal{T}(i,j,k) \qquad \text{où p = i + (j - 1)(I)} \tag{1.5}$$

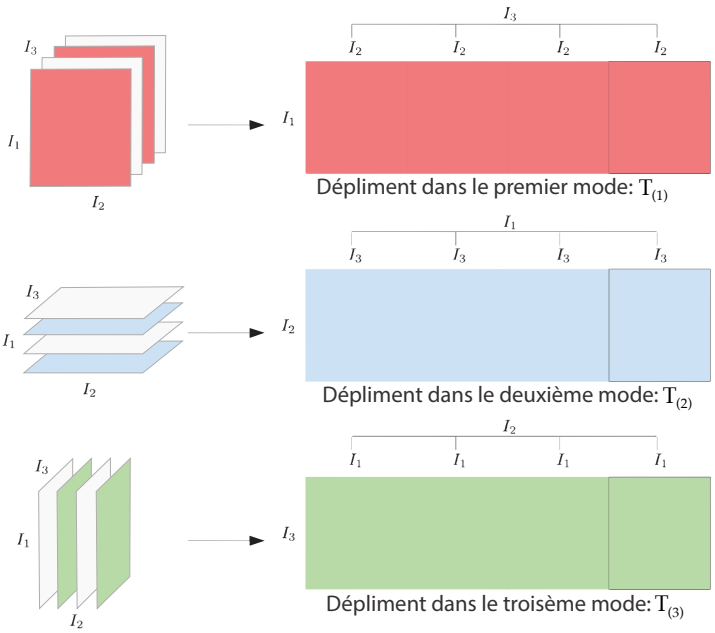


FIGURE 1.6 – Schéma explicatif du dépliement matriciel d'un tenseur d'odre 3.

Les colonnes peuvent être ordonné de different façon mais deux standard méthode qui ont été utilisé par [De Lathauwer *et al.*, 2000a] et [Kiers, 2000]. les deux sont cycliques, mais l'ordre est inversé.

▷ Pour [De Lathauwer *et al.*, 2000a], l'ordre des colonnes est cyclique vers l'arrière et donc pour notre tenseur d'ordre 3, l'ordre d'ordonnancement est donné d'abord par **K** et en suite par **J** pour **I** l'indice des lignes, et nous nous référons à cet ordre par cyclique vers l'arrière (en anglais : backward cyclic) ou 'bc' (figure 1.7).

▷ Pour [Kiers, 2000], l'ordre des colonnes est inverser par rapport a la méthode *backward cyclic* l'ordre d'ordonnancement est fait par **J** et en suite par **K** pour **I** l'indice des lignes, et nous nous référons à cet ordre par cyclique vers l'avant (en anglais :forward cyclic) ou 'fc' (figure 1.8).

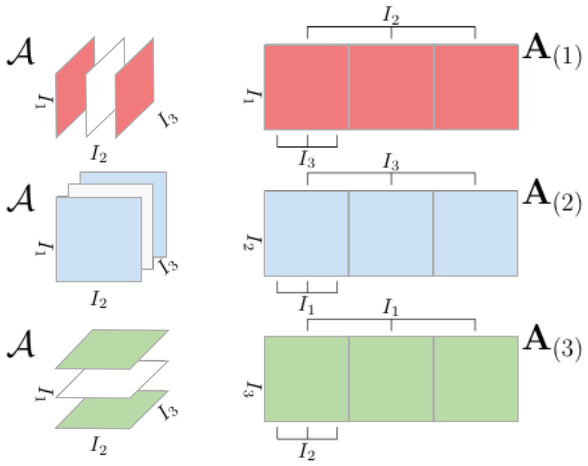


FIGURE 1.7 – L'ordre cyclique 'backward cyclic' pour un tenseur d'ordre 3.

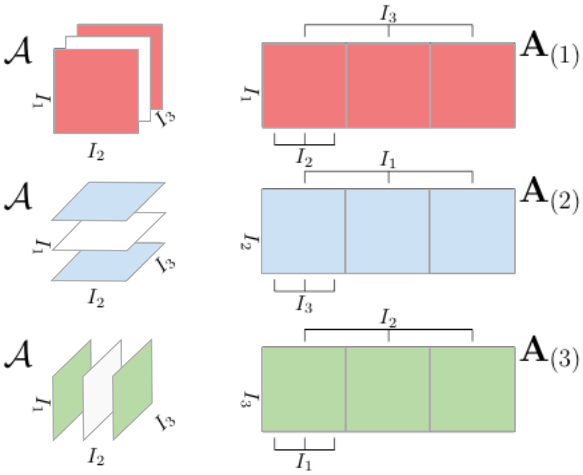


FIGURE 1.8 – L'ordre cyclique 'forward cyclic' pour un tenseur d'ordre 3.

Remarque : Pour une matrice **A**, la dépliante dans le 1-mode de la matrice A n'est autre que la matrice A elle-même et la dépliante dans le 2-mode est la matrice A^T.

Exemple : Soit les tranches frontales d'un tenseur $\mathcal{T} \in \mathbb{R}^{3\times4\times2}$:

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

Les trois représentations matricielles de ce tenseur sont les suivantes :

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

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

$$\mathbf{T}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & \dots & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & \dots & 21 & 22 & 23 & 24 \end{bmatrix}$$

1.4.2.4 *Tenseur symétrique, antisymétrique*

Un tenseur peut être symétrique, antisymétrique ou ni l'un ni l'autre. On dira alors qu'un tenseur $\mathcal{T}$ d'ordre N est **symétrique** s'il est inchangé par permutations des indices [Comon *et al.*, 2008, Comon *et al.*, 2009b]. Cela veut dire que les éléments de ce tenseur vérifient $t_{I_1,\dots,I_N} = t_{\mathbf{\Pi}(I_1 I_2 \dots I_N)}$ pour toute permutation $\mathbf{\Pi}$ des indices. Un tenseur est dit **antisymétrique** si, par une permutation quelconque des indices, il subit un changement de signe qui n'est d'autre que celui de la permutation. Pour un tenseur antisymétrique, les composantes dans lesquelles un indice se répète au moins deux fois sont toutes nulles. Par exemple, les k composantes t_{iik} du tenseur $\mathcal{T}$ sont nulles.

1.4.2.5 *Tenseur diagonal et tenseur creux*

♢ Un tenseur est diagonal (figure 1.9) si tous ces éléments qui ne sont pas désignés par le même indice dans toutes les dimensions sont nuls. On note $\mathcal{I} \in \mathbb{R}^{I\times I\times\dots\times I}$ le tenseur diagonal avec des 1 sur son super-diagonale et 0 sinon.

♢ Un tenseur creux (figure 1.9) avec une tranche frontale sur son diagonal peut être exprimé mathématiquement par :

$$\mathcal{I} \times_3 \mathcal{C}$$

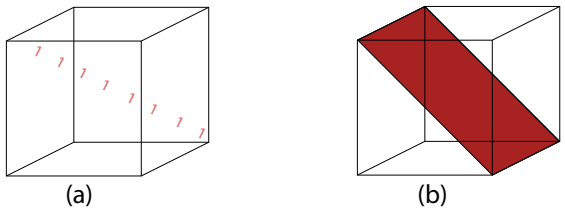


FIGURE 1.9 – (a) Tenseur diagonal , (b) Tenseur creux

1.4.3 Produits et operateurs matriciels

1.4.3.1 Produit de Kronecker

Le produit de kronecker [Brewer, 1978], noté $\otimes$, entre deux matrices : $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$ et $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_G] \in \mathbb{R}^{J \times G}$ est une matrice notée $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{IJ \times RG}$ et définie comme suit :

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \ldots & a_{1R}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \ldots & a_{2R}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \ldots & a_{IR}\mathbf{B} \end{pmatrix} \tag{1.6}$$

Propriétés Le produit de kronecker présente les propriétés suivantes, si l'on considère 3 matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ et $\mathbf{D}$, dont deux (les matrices $\mathbf{B}$ et $\mathbf{C}$) sont de la même taille, α désignant un scalaire :

- Associatif : $\mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}) = (\mathbf{A} \otimes \mathbf{B}) \otimes \mathbf{C}$;
- Distributif :
  - $\mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) = (\mathbf{A} \otimes \mathbf{B}) + (\mathbf{A} \otimes \mathbf{C})$;
  - $(\mathbf{B} + \mathbf{C}) \otimes \mathbf{A} = (\mathbf{B} \otimes \mathbf{A}) + (\mathbf{C} \otimes \mathbf{A})$;
- Non commutatif : $\mathbf{A} \otimes \mathbf{B} \neq \mathbf{B} \otimes \mathbf{A}$
- $(\mathbf{A} \otimes \mathbf{B})^T = \mathbf{A}^T \otimes \mathbf{B}^T$
- $(\mathbf{A} \otimes \mathbf{B})^\dagger = \mathbf{A}^\dagger \otimes \mathbf{B}^\dagger$.
- si $\mathbf{A}$ et $\mathbf{B}$ sont inversibles, $\mathbf{A} \otimes \mathbf{B}$ l'est aussi : $(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$
- $\alpha(\mathbf{A} \otimes \mathbf{B}) = (\alpha\mathbf{A}) \otimes \mathbf{B}) = \mathbf{A} \otimes (\alpha\mathbf{B})$
- La propriété suivante mélange les aspects liés au produit matriciel usuel et au produit de Kronecker : $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$
- Le produit de Kronecker n'est pas commutatif ; cependant pour toutes $\mathbf{A}$ et $\mathbf{B}$, il existe deux matrices de permutation $\mathbf{P}$ et $\mathbf{Q}$ telles que : $\mathbf{A} \otimes \mathbf{B} = \mathbf{P}(\mathbf{B} \otimes \mathbf{A})\mathbf{Q}$.

Nombre de multiplications du produit de Kronecker Le produit $a_{ij}\mathbf{B}$ coûte JG multiplications. Donc, selon l'expression (1.6), il est nécessaire de calculer $IRJG$ multiplications pour effectuer $\mathbf{A} \otimes \mathbf{B}$.

1.4.3.2 Produit de Khatri-Rao

Le produit de Khatri-Rao [Khatri et Rao, 1968], noté $\odot$, entre deux matrices possédant le même nombre de colonnes, $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$ et $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_R] \in \mathbb{R}^{J \times R}$, est définie comme le produit de kronecker selon les colonnes :

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2, \ldots, \mathbf{a}_R \otimes \mathbf{b}_R] \in \mathbb{R}^{IJ \times R} \tag{1.7}$$

Propriétés Il présente les propriétés suivantes si l'on considère deux matrices $\mathbf{A}$ et $\mathbf{B}$ possédant le même nombre de colonnes et une troisième matrice $\mathbf{C}$ de la même taille que $\mathbf{B}$ ainsi qu'un scalaire α :

- Associatif : $\mathbf{A} \odot (\mathbf{B} \odot \mathbf{C}) = (\mathbf{A} \odot \mathbf{B}) \odot \mathbf{C}$;
- Distributif : $\mathbf{A} \odot (\mathbf{B} + \mathbf{C}) = (\mathbf{A} \odot \mathbf{B}) + (\mathbf{A} \odot \mathbf{C})$;
- Non commutatif : $\mathbf{A} \odot \mathbf{B} \neq \mathbf{B} \odot \mathbf{A}$
- $\alpha(\mathbf{A} \odot \mathbf{B}) = (\alpha\mathbf{A}) \odot \mathbf{B} = \mathbf{A} \odot (\alpha\mathbf{B})$
- $(\mathbf{A} \odot \mathbf{B})^T = (\mathbf{A}^T\mathbf{A}) * (\mathbf{B}^T\mathbf{B})$
- $(\mathbf{A} \odot \mathbf{B})^\dagger = ((\mathbf{A}^T\mathbf{A}) * (\mathbf{B}^T\mathbf{B}))^\dagger(\mathbf{A} \odot \mathbf{B})^T$

Nombre de multiplications du produit de Khatri-Rao Pour réaliser le produit de Kronecker $a_j \odot b_j$, IJ multiplications sont nécessaires. Ainsi pour réaliser le produit de Khatri-Rao, IJR multiplications sont nécessaires.

Remarque Nous verrons plus tard que le produit Khatri-Rao est très important pour l'expression de la décomposition CP. Remarquons que les matrices dans un produit Khatri-Rao ont tous le même nombre de colonnes. En outre, si $\mathbf{a}$ et $\mathbf{b}$ sont les vecteurs, les produits Khatri-Rao et Kronecker sont identiques, et on aura : $\mathbf{a} \odot \mathbf{b} = \mathbf{a} \otimes \mathbf{b}$.

1.4.3.3 Produit de Hadamard

Le produit de Hadamard [Horn, 1990] entre deux matrices $\mathbf{A}$ et $\mathbf{B}$ de même taille $I \times J$ est le produit terme à terme entre chaque élément des matrices, noté par $\mathbf{A} \boxdot \mathbf{B}$ de taille $I \times J$:

$$\mathbf{A} \boxdot \mathbf{B} = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} & \ldots & a_{1J}b_{1J} \\ a_{21}a_{21} & a_{22}b_{22} & \ldots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \ldots & a_{IJ}b_{IJ} \end{pmatrix} \tag{1.8}$$

Propriétés Ce produit présente les propriétés suivantes si l'on considère trois matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ de la même taille, et un scalaire α :

— Associatif : $\mathbf{A} \boxdot (\mathbf{B} \boxdot \mathbf{C}) = (\mathbf{A} \boxdot \mathbf{B}) \boxdot \mathbf{C}$;

— Distributif : $\mathbf{A} \boxdot (\mathbf{B} + \mathbf{C}) = (\mathbf{A} \boxdot \mathbf{B}) + (\mathbf{A} \boxdot \mathbf{C})$;

— Commutatif : $\mathbf{A} \boxdot \mathbf{B} = \mathbf{B} \boxdot \mathbf{A}$

— $(\mathbf{A} \boxdot \mathbf{B})^T = \mathbf{A}^T \boxdot \mathbf{B}^T = \mathbf{B}^T \boxdot \mathbf{A}^T$

— $\alpha(\mathbf{A} \boxdot \mathbf{B}) = (\alpha\mathbf{A}) \boxdot \mathbf{B} = \mathbf{A} \boxdot (\alpha\mathbf{B})$

— $(\mathbf{A} \odot \mathbf{B})^T(\mathbf{A} \odot \mathbf{B}) = \mathbf{A}^T\mathbf{A} \boxdot \mathbf{B}^T\mathbf{B}$

Nombre de multiplications du produit de Hadamard Pour réaliser le produit $\mathbf{A} \boxdot \mathbf{B}$, il faut effectuer IJ multiplications.

Nous résumons la complexité numérique des différents produits matriciels dans le tableau 1.1.

Produits matriciels	taille de la 1^{ere} matrice	taille de la 2^{eme} matrice	complexité numérique
Produit matriciel classique	$\mathbf{I} \times \mathbf{R}$	$\mathbf{R} \times \mathbf{J}$	**IJR**
Produit de Hadamard	$\mathbf{I} \times \mathbf{J}$	$\mathbf{I} \times \mathbf{J}$	**IJ**
Produit de Kronecker	$\mathbf{I} \times \mathbf{J}$	$\mathbf{K} \times \mathbf{R}$	**IJKR**
Produit de Khatri-Rao	$\mathbf{I} \times \mathbf{J}$	$\mathbf{K} \times \mathbf{J}$	**IJK**

TABLE 1.1 – Tableau récapitulatif des complexités numériques des différents produits matriciels.

1.4.3.4 La trace d'une matrice

La trace d'une matrice est simplement la somme de ses éléments diagonaux. Pour une matrice $\mathbf{A}$ carrée de dimension $N \times N$:

$$\text{trace}\{\mathbf{A}\} = \sum_{i=1}^{N} a_{ii} \tag{1.9}$$

Propriétés Elle présente des propriétés qui seront exploitées par la suite. Considérons trois matrices carrées $\mathbf{D}^{(1)}, \mathbf{D}^{(2)}, \mathbf{D}^{(3)}$ de dimension $M \times M$, et quatre matrices rectangulaires $\mathbf{D}^{(4)}, \mathbf{D}^{(5)}, \mathbf{D}^{(6)}$ et $\mathbf{D}^{(7)}$ (de taille resp. $M \times N, N \times M, M \times N, M \times N$), on a alors les propriétés suivantes :

— $\text{trace}\{\mathbf{D}^{(1)} + \mathbf{D}^{(2)}\} = \text{trace}\{\mathbf{D}^{(1)}\} + \text{trace}\{\mathbf{D}^{(2)}\}$;

— $\text{trace}\{\alpha\mathbf{D}^{(1)}\} = \alpha\text{trace}\{\mathbf{D}^{(1)}\}$;

— $\text{trace}\{(\mathbf{D}^{(1)})^T\} = \text{trace}\{\mathbf{D}^{(1)}\}$;

— $\text{trace}\{\mathbf{D}^{(1)}\mathbf{D}^{(2)}\mathbf{D}^{(3)}\} = \text{trace}\{\mathbf{D}^{(3)}\mathbf{D}^{(1)}\mathbf{D}^{(2)}\} = \text{trace}\{\mathbf{D}^{(2)}\mathbf{D}^{(3)}\mathbf{D}^{(1)}\}$;

— $\text{trace}\{\mathbf{D}^{(4)}\mathbf{D}^{(5)}\} = \text{trace}\{\mathbf{D}^{(5)}\mathbf{D}^{(4)}\}$

— $\text{d}((\mathbf{D}^{(1)})^T) = (\text{d}(\mathbf{D}^{(1)}))^T$

— $\text{trace}\{(\mathbf{D}^{(4)})^T(\mathbf{D}^{(6)} * \mathbf{D}^{(7)})\} = \text{trace}\{((\mathbf{D}^{(4)})^T * (\mathbf{D}^{(6)})^T)\mathbf{D}^{(7)}\}$

1.4.3.5 Opérateur de vectorisation

L'opérateur vec(.) permet de transformer une matrice en un vecteur. La convention choisie ici est d'empiler les colonnes de la matrice. Si $\mathbf{A} \in \mathbb{R}^{M\times N}$:

$$\text{vec}(\mathbf{A}) = \begin{pmatrix} a_{11} \\ \dots \\ a_{M1} \\ — \\ a_{12} \\ \dots \\ a_{M2} \\ — \\ \dots \\ — \\ a_{MN} \\ a_{1N} \\ \dots \\ a_{MN} \end{pmatrix}$$

Les éléments du vecteur sont ainsi définis par :

$$(\text{vec}(\mathbf{A}))_{m+(n-1)M} = a_{mn}$$

Propriétés La vectorisation permet de simplifier certains calculs en offrant des relations aves le produit de Kronecker ou encore la trace. Si on considère 4 matrices $\mathbf{A} \in \mathbb{R}^{M\times N}, \mathbf{B} \in \mathbb{R}^{M\times N}, \mathbf{C} \in \mathbb{R}^{M\times M}, \mathbf{D} \in \mathbb{R}^{M\times M}, \alpha \in \mathbb{R}$ et deux vecteurs $\mathbf{a}$ et $\mathbf{b}$ de taille quelconque :

— $\text{vec}(\alpha\mathbf{A}) = \alpha\text{vec}(\mathbf{A})$;

— $\text{vec}(\mathbf{A} + \mathbf{B}) = \text{vec}(\mathbf{A}) + \text{vec}(\mathbf{B})$;

— $\text{vec}(\mathbf{A})^T\text{vec}(\mathbf{B}) = \text{trace}(\mathbf{A}^T\mathbf{B})$;

— $\text{vec}(\mathbf{ab}^T) = \mathbf{b} \otimes \mathbf{a}$;

— $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B})$; (Cette relation reste vraie pour n'importe quelles matrices **A**,**B**,**C**, dès lors que le produit **ABC** est défini et carré)

— $\text{trace}\{\mathbf{ABCD}\} = \text{vec}(\mathbf{D}^T)^T(\mathbf{A} \otimes \mathbf{C}^T)\text{vec}(\mathbf{B}^T)$; (Cette relation reste vraie pour n'importe quelles matrices **A**, **B**, **C**, dès lors que le produit **ABCD** est défini et carré)

1.4.3.6 *Opérateur de projection sur* $\mathbb{R}$

Soit $\mathbf{A} \in \mathbb{R}^{I\times J}$, l'opérateur de projection sur $\mathbb{R}^+$ est alors défini par :

$$([\mathbf{A}]_+)_{i,j} = \begin{cases} \mathbf{a}_{i,j} & \text{si } \mathbf{a}_{i,j} \geq 0 \\ 0 & \text{sinon.} \end{cases}$$

1.4.4 Opérateurs tensoriels

1.4.4.1 *Le produit n-modal*

Le produit n-mode, noté $\times_n$, généralise le produit matriciel au produit entre un tenseur et les vecteurs n-modaux dans un n-mode particulier [De Lathauwer *et al.*, 2000a]. Considèrons un tenseur d'ordre trois $\mathcal{T} \in \mathbb{C}^{I\times J\times K}$ et une matrice $\mathcal{A} \in \mathbb{C}^{L\times I}$. Le produit en 1-mode entre le tenseur $\mathcal{T}$ et la matrice **A** est un tenseur de taille $L \times J \times K$ dont les éléments selon le premier mode sont définis par :

$$(\mathcal{T}\times_1 \mathbf{A})_{ljk} = \sum_{i=1}^{I} t_{ijk} \times a_{li}$$

De même, le produit en 2-mode de $\mathcal{T}$ par une matrice $\mathbf{B} \in \mathbb{C}^{L\times J}$ est un tenseur de taille $I \times L \times K$ tel que ses éléments sont définis par :

$$(\mathcal{T}\times_2 \mathbf{B})_{ilk} = \sum_{i=1}^{J} t_{ijk} \times b_{lj}$$

ainsi que le produit en 3-mode de $\mathcal{T}$ par une matrice $\mathbf{C} \in \mathbb{C}^{L\times K}$ est un tenseur de taille $I \times J \times L$ tel que ses éléments sont définis par :

$$(\mathcal{T}\times_3 \mathbf{C})_{ilk} = \sum_{i=1}^{K} t_{ijk} \times c_{lk}$$

Nous pouvons généraliser l'écriture de ces trois dernières équations aux produits n-modaux entre le tenseur $\mathcal{A} \in \mathbb{C}^{I_1\times I_2\times\ldots\times I_N}$ et les matrices $\mathbf{X}^n \in \mathbb{C}^{J_n\times I_n}$, avec n=1,⋯,N. Le résultat de ce produit est un tenseur $\mathcal{T} \in \mathbb{C}^{J_1\times J_2\times\ldots\times J_N}$:

$$\boxed{\mathcal{T} = \mathcal{A} \times_1 \mathbf{X}^1 \times_2 \ldots \times_N \mathbf{X}^N} \tag{1.10}$$

Exemple Soit $\mathcal{T} \in \mathbb{R}^{3\times4\times2}$:

$$\mathbf{T}_{::1} = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \mathbf{T}_{::2} = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix} \tag{1.11}$$

Et la matrice $\mathbf{A} \in \mathbb{R}^{2\times3}$:

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Le nombre de colonnes de $\mathbf{A}$ est égale à la taille de mode-1 de $\mathcal{T}$. Donc, on peut calculer $\mathcal{T} \times_1 \mathbf{A}$, qui est de taille $2 \times 4 \times 2$:

$$(\mathcal{T} \times_1 \mathbf{A})_{::1} = \begin{bmatrix} 22 & 49 & 76 & 103 \\ 28 & 64 & 100 & 136 \end{bmatrix},$$

Propriété :

— Quel que soit le tenseur $\mathcal{T} \in \mathbb{R}^{J_1\times J_2\times\ldots\times J_N}$, et les matrices $\mathbf{A} \in \mathbb{R}^{I_m\times J_m}$, $\mathbf{B} \in \mathbb{R}^{I_n\times J_n}$:

$$\mathcal{T} \times_m \mathbf{A} \times_n \mathbf{B} = (\mathcal{T} \times_m \mathbf{A}) \times_n \mathbf{B} = (\mathcal{T} \times_n \mathbf{B}) \times_m \mathbf{A} (m \neq n).$$

— Quel que soit le tenseur $\mathcal{T} \in \mathbb{R}^{J_1\times J_2\times\ldots\times J_N}$, et les matrices $\mathbf{A} \in \mathbb{R}^{I\times J_n}$, $\mathbf{B} \in \mathbb{R}^{K\times I}$:

$$\mathcal{T} \times_n \mathbf{A} \times_n \mathbf{B} = \mathcal{T} \times_n (\mathbf{BA})$$

— Quel que soit le tenseur $\mathcal{T} \in \mathbb{R}^{J_1\times J_2\times\ldots\times J_N}$, et la matrice $\mathbf{A} \in \mathbb{R}^{I\times J_n}$ avec des colonnes de rang plein :

$$\mathcal{X} = \mathcal{T} \times_n \mathbf{A} \Rightarrow \mathcal{T} = \mathcal{X} \times_n \mathbf{A}^\dagger$$

— Enfin, quel que soit le tenseur $\mathcal{T} \in \mathbb{R}^{J_1\times J_2\times\ldots\times J_N}$, et la matrice $\mathbf{A} \in \mathbb{R}^{I\times J_n}$ orthogonale :

$$\mathcal{X} = \mathcal{T} \times_n \mathbf{A} \Rightarrow \mathcal{T} = \mathcal{X} \times_n \mathbf{A}^T$$

$\diamondsuit$ Le produit n-mode d'un tenseur $\mathcal{T} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_N}$ par un vecteur $\mathbf{a} \in \mathbb{R}^{I_n}$, noté par $\mathcal{T}\bar{\times}_n \mathbf{a}$, est un tenseur d'ordre N-1, i.e, la taille est $I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N$ dont les éléments sont définis par :

$$(\mathcal{T}\bar{\times}_n\mathbf{a})_{i_1}\ldots i_{n-1}i_{n+1}i_N = \sum_{i_n=1}^{I_N} t_{i_1\ldots i_N} a_{i_n}$$

Exemple Soit $\mathcal{T} \in \mathbb{R}^{3\times4\times2}$:

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

et le vecteur $\mathbf{v} \in \mathbb{R}^{4\times1}$:

$$\mathbf{v} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Donc $\mathcal{T}\bar{\times}_2\mathbf{v}$ est défini comme :

$$\mathcal{T}\bar{\times}_2\mathbf{v} = \begin{bmatrix} 70 & 190 \\ 80 & 200 \\ 90 & 210 \end{bmatrix}$$

1.4.4.2 Le produit extérieur

Le produit extérieur d'un tenseur d'ordre P noté $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_p}$ et d'un tenseur d'ordre Q noté $\mathcal{B} \in \mathbb{R}^{J_1\times J_2\times\ldots\times J_q}$, est un tenseur d'ordre P+Q défini par (pour toutes les valeurs des indices) :

$$\boxed{(\mathcal{A}\circ\mathcal{B})_{i_1i_2\ldots i_pj_1j_2\ldots j_q} = a_{i_1i_2\ldots i_p}b_{j_1j_2\ldots j_q}} \tag{1.12}$$

dont les éléments sont définis par le produit :

$$t_{I_1,\ldots,I_N} = \mathbf{a}^{(1)}_{i_1}\ \mathbf{a}^{(2)}_{i_2}\ \ldots\ \mathbf{a}^{(N)}_{i_N}$$

$\diamondsuit$ le produit extérieur $(\mathbf{a}\circ\mathbf{b})$ des vecteurs $\mathbf{a} \in \mathbb{R}^{I\times1}$, $\mathbf{b} \in \mathbb{R}^{J\times1}$, est une matrice de rang 1 :

$$\mathbf{A} = \mathbf{a}\circ\mathbf{b} = \mathbf{a}\mathbf{b}^T \in \mathbb{R}^{I\times J}$$

$\diamondsuit$ le produit extérieur $(\mathbf{a}\circ\mathbf{b}\circ\mathbf{c})$ des vecteurs $\mathbf{a} \in \mathbb{R}^{I\times1}$, $\mathbf{b} \in \mathbb{R}^{J\times1}$ et $\mathbf{c} \in \mathbb{R}^{K\times1}$, est un tenseur $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ d'ordre 3 de rang 1 dont les éléments sont définies par :

$$t_{ijk} = a_ib_jc_k.$$

1.4.4.3 Produit scalaire tensoriel

Le produit scalaire tensoriel, noté $\langle , \rangle$, est défini de la façon suivante :

Considérons deux tenseurs $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ de même ordre N, Le produit scalaire entre ces deux tenseurs est donné par :

$$\boxed{\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1, i_2, \ldots, i_N} a_{i_1, i_2, \ldots, i_N} b_{i_1, i_2, \ldots, i_N}} \tag{1.13}$$

Une propriété importante du produit tensoriel est que :

$$\langle \mathcal{A}, \mathcal{B} \rangle = \text{trace}\{\mathbf{A}_{(n)} \mathbf{B}_{(n)}^T\}, \forall n = 1, \ldots, N$$

dans laquelle $\mathbf{A}_{(n)}$ et $\mathbf{B}_{(n)}^T$ sont les matrices dépliantes de $\mathcal{A}$ et $\mathcal{B}$ dans le n-mode.

1.4.4.4 Norme Frobenius

La norme de Frobenius $\|.\|_F$ d'un tenseur $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$ d'ordre trois est définie par :

$$\|\mathcal{T}\|_F = \sqrt{\sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} |t_{ijk}|^2} = \sqrt{\langle \mathcal{T}, \mathcal{T} \rangle} = \sqrt{\text{trace}\{\mathbf{T}_{(n)} \mathbf{T}_{(n)}^T\}}, \forall n = 1, \ldots, 3$$

$\diamondsuit$ La distance quadratique entre deux tenseurs $\mathcal{X}$ et $\mathcal{Y} \in \mathbb{C}^{I_1 \times I_2 \times \ldots \times I_N}$ peut être définie par la quantité $\|\mathcal{X} - \mathcal{Y}\|_F^2$

1.4.4.5 Rang d'un tenseur

Il n'est pas évident de généraliser la définition du rang d'une matrice pour un tenseur d'ordre N (N>2). En effet, le rang d'une matrice quelconque $\mathbf{A}$ vérifie deux définitions :

Définition 1 : Il correspond à la dimension du sous-espace vectoriel engendrée par les vecteurs (lignes et colonnes) de $\mathbf{A}$.

Définition 2 : il est lié au produit extérieur étant le nombre minimum de matrices de rang 1 qui génèrent la matrice $\mathbf{A}$.

Pour un tenseur d'ordre N (avec N>2), l'égalité entre ces deux définitions n'est plus vérifiée [De Silva et Lim, 2008], et conduit à deux définitions de rang distinctes : Un rang multilinéaire R_n et un rang tensoriel R. Au début de 19$^{\text{ème}}$ siècle, Hitchcock a présenté pour la première fois le concept d'un rang multiple [Hitchcock, 1928], qui généralise directement le rang des vecteurs colonnes et des vecteurs lignes d'une matrice pour un tenseur d'ordre supérieur. Plus tard, Kruskal introduit l'idée du rang n-modal [Kruskal, 1989], popularisé par De Lathauwer [De Lathauwer *et al.*, 2000a]. La différence entre le rang n-modal et le rang multiple est que le premier se restreint aux matrices dépliantes du

tenseur $\mathcal{T}$, alors que le deuxième est défini pour un déploiement arbitraire. Dans les années 70, le concept de rang a été indépendamment proposé dans des applications par Carroll et Chang [Carroll et Chang, 1970], Harshman [Harshman *et al.*, 1970] et Kruskal [Kruskal, 1977, Kruskal, 1989]. Le nouveau concept de rang lié au produit extérieur est connu sous le nom de rang tensoriel R.

Rang tensoriel R : Le concept de rang lié au produit extérieur d'une matrice peut être étendu aux tenseurs d'ordre supérieur. La définition d'un tenseur de rang R généralise la deuxième définition matricielle. En effet, $\mathcal{T}$ est un tenseur de rang R, si R est le nombre minimum de tenseurs de rang 1 qui génèrent $\mathcal{T}$ par combinaison linéaire. Nous notons le rang du tenseur $\mathcal{T}$ par :

$$\boxed{Rang(\mathcal{T}) = R} \tag{1.14}$$

Remarque : Le rang d'une matrice $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ est toujours inférieur ou égal au $min(I_1, I_2)$. Cette propriété n'est plus vérifiée dans le cas tensoriel [Kolda, 2001, De Lathauwer *et al.*, 2000a]. En effet le rang d'un tenseur $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ peut être tel que :

$$\boxed{R > min(I_1, I_2, ..., I_N)} \tag{1.15}$$

Le rang d'un tenseur peut être typique ou générique. Une propriété est dite typique si elle est vrai sur un ensemble de volumes non nulle, alors qu'elle est dite générique si elle est vrai presque partout. En d'autres termes, une propriété générique est typique, mais l'inverse n'est pas vrai [Comon et ten Berge, 2008]. De ce fait, on peut dire qu'un rang est typique si sa probabilité non nulle. Le rang générique d'un tenseur est celui obtenu en choisissant ses éléments aléatoirement selon une loi continue. S'il y a un seul rang typique, il peut être appelé le rang générique.

Rang n-modal, rang multilinéaire R_n : Cette définition du rang généralise directement le rang des vecteurs colonnes et des vecteurs lignes d'une matrice pour un tenseur d'ordre supérieur. Ce concept a été présenter pour la première fois par Hitchcock dans [Hitchcock, 1928] sous le nom de rang multiple.

Soit un tenseur $\mathcal{T}$ d'ordre N, le rang n-modal, R_n est la dimension de l'espace vectoriel généré par les fibres n-modales. En d'autres termes, le rang n-modal est le rang de la matrice dépliante $\mathbf{T}_{(n)}$ dans le n-mode :

$$\boxed{Rang(\mathbf{T}_{(n)}) = Rang_{(n)}(\mathbf{T}) = R_n} \tag{1.16}$$

En conséquence le rang multilinéaire est le $Rang(R_1, ..., R_N)$.

Si on considère un tenseur $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$, son rang multilinéaire est défini par le triplet (R_1, R_2, R_3) ; avec : $R_i = Rang(\mathbf{T}_{(i)})$

Remarque : Pour une matrice $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ le rang colonnes est toujours égal au rang lignes ainsi qu'au rang du produit extérieur. Cette égalité n'est plus vérifié dans le cas tensoriel. Par contre on observe toujours une valeur du rang n-modal inférieur ou égal au rang du tenseur $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ telle que [De Lathauwer *et al.*, 2000a] :

$$\boxed{\mathbf{R}_n \leq min(R, I_n)} \tag{1.17}$$

Tenseur de rang 1 : Un tenseur $\mathcal{T}$ d'ordre N est de rang 1 s'il peut s'écrire comme le produit extérieur de N vecteurs, $\mathcal{T} = a_1 \circ a_2 \circ \ldots \circ a_N$. Ainsi, un tenseur d'ordre trois $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$ est de rang 1 si ces éléments peuvent s'écrire comme $\mathbf{t}_{ijk} = \mathbf{a}_i \mathbf{b}_j \mathbf{c}_k$, ou : $\mathbf{a} \in \mathbb{R}^{I \times 1}$,$\mathbf{b} \in \mathbb{R}^{J \times 1}$ et $\mathbf{c} \in \mathbb{R}^{K \times 1}$ (La figure 1.10).

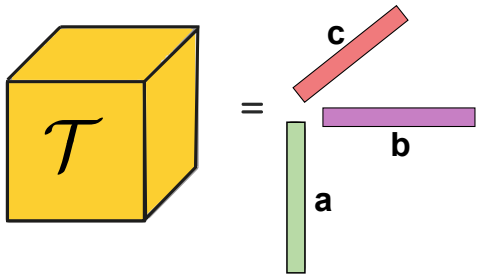


FIGURE 1.10 – Un tenseur d'ordre 3 et de rang 1.

♢ Cette définition généralise la définition d'une matrice de rang 1 : $\mathbf{M} \in \mathbb{R}^{I \times J}$ est de rang 1 si : $\mathbf{M} = \mathbf{a}.\mathbf{b}^T$.

1.5 Bilan du chapitre

Ce chapitre fournit les éléments fondamentaux de l'algèbre multilinéaire qui seront utilités par la suite. Dans un premier temps, nous avons donné quelques définitions du tenseur, ainsi que quelques propriétés mathématiques. et nous avons terminé le chapitre par définir les représentations et les opérations les plus importantes concernant les tenseurs, ainsi que leurs propriétés.

Nous avons mis en évidence le fait que la généralisation des outils bilinéaires aux outils multilinéaires n'est pas triviale. La majorité des opérations entre les matrices se généralisent facilement pour les tenseurs. c'est le cas par exemple de la somme et de la multiplication d'un tenseur avec un réel. Pour le produit scalaire et la norme, la définition

s'étend aussi naturellement. Cependant, il en existe d'autres où les matrices et les tenseurs se distinguent. L'une des principales distinctions est la non-extension du théorème d'ECKART-YOUNG [Eckart et Young, 1936] au cas tensoriel (tenseur d'ordre > 2). Il en résulte deux définitions de rangs distincts pour un tenseur :

Définition 1 Le rang n-modal R_n est le rang multilinéaire du tenseur $\mathcal{T}$ d'ordre N. Ce rang généralise le rang colonnes et lignes d'une matrice pour chaque dépliante n-modale [Hitchcock, 1927, Hitchcock, 1928].

Définition 2 Le rang tensoriel R [Carroll et Chang, 1970, Harshman *et al.*, 1970, Kruskal, 1977, Kruskal, 1989]. Ce rang généralise le rang associé au produit extérieur d'une matrice.

Le rang n-modal R_n et le rang du tenseur R mènent donc à deux définitions des décompositions tensorielles, toutes les deux issues de la généralisation de la décomposition en valeurs singulières d'une matrice, notée SVD (*Singular Values Decomposition*). Ainsi, le rang multilinéaire est lié à la décomposition tensorielle qui généralise la modélisation matricielle de la SVD, appelée la décomposition de **TUCKER3**, alors que le rang tensoriel R, associé à la décomposition tensorielle, généralise la modélisation canonique de la SVD, appelée la décomposition de **PARAFAC/CANDECOMP**. Nous présenterons dans le chapitre suivant ces deux décompositions tensorielles ainsi que d'autres.

Chapitre

2 DÉCOMPOSITIONS TENSORIELLES

Sommaire

2.1 Introduction

Dans un nombre croissant de domaines tels que le traitement de signal, l'analyse multivariée ou la programmation scientifique, il est nécessaire de manipuler des quantités à plusieurs indices [Chevalier *et al.*, 2005, Sidiropoulos *et al.*, 2000b]. Ces quantités sont appelées des tenseurs. L'analyse des tenseurs d'ordre supérieur à trois est dite algèbre multilinéaire. On s'intéresse aux tenseurs d'ordre supérieur à trois, car ils possèdent des propriétés qui ne sont pas valables pour les matrices ou les vecteurs. De même qu'une matrice qui peut être décomposée de plusieurs façons (SVD, QR, etc), un tenseur peut se décomposer de différentes manières.

La décomposition tensorielle est un domaine de l'algèbre multilinéaire qui caractérise un tenseur comme une combinaison linéaire de produits extérieurs des vecteurs. Parmi les différentes décompositions tensorielles, deux types qui ont été largement étudiés dans la littérature, tout en faisant l'objectif de nombreuses applications dans des domaines différents. Ces différentes décompositions sont la décomposition **PARAFAC** (Parallel Factor Analysis) [Harshman *et al.*, 1970, Harshman, 1972], la décomposition de **TUCKER3**, qui peut être interprétée comme une généralisation de l'analyse en composantes principales (ACP) à des ordres supérieurs [Tucker, 1966]. Cette décomposition est connue aussi sous le nom de la décomposition en valeurs singulières d'ordre supérieur (HOSVD) [Kolda, 2001], qui est une généralisation de la SVD matricielle. La décomposition Tucker-3 a été appliquée avec succès dans différents domaines tels que la Chromatographie [Bro, 1998] et l'analyse de la perception des personnes [Kroonenberg, 1983].

Nous nous intéressons ici à la décomposition PARAFAC des tenseurs. Cette décomposition a été introduite de manière indépendante sous le nom de CANDECOMP (Canonical Decomposition) en psychométrie [Carroll et Chang, 1970] et de PARAFAC (Parallel Factor Analysis) en phonétique [Harshman *et al.*, 1970], mais la terminologie la plus répandue est CP (Canonical Polyadic Decomposition) [Comon *et al.*, 2009a]. Elle a ensuite été utilisée dans des domaines variés, comme la chimiométrie [Bro *et al.*, 1997], ou les télécommunications [Sidiropoulos *et al.*, 2000b, Sidiropoulos *et al.*, 2000a].

La caractéristique la plus intéressante de la décomposition CP est son originalité intrinsèque. Car, et contrairement à la décomposition matricielle, où le problème de liberté de rotation se manifeste, la décomposition PARARAC des tenseurs est essentiellement unique, à l'indétermination de l'échelle et de permutation [Kruskal, 1977, Stegeman et Sidiropoulos, 2007]. La première preuve d'unicité a été fournie par Kruskal dans [Kruskal, 1977]. Récemment, cette preuve a été reformulée en utilisant l'algèbre linéaire de base [Stegeman et Sidiropoulos, 2007]. Une preuve concise valable pour les tenseurs complexes a été donnée dans [Sidiropoulos *et al.*, 2000b]. Contrairement à la décomposition CP, la décomposition de Tucker-3 ne possède pas une unicité intrinsèque et n'est pas intéressante d'un point de vue d'estimation des paramètres. Cependant, on peut parfois résoudre ce problème en utilisant des versions limitées du modèle de Tucker.

Le reste de ce chapitre est organisé de la manière suivante : dans la section 2.2, nous introduisons quelques décompositions matricielles. Dans la section 2.3, quelques décompositions tensorielles seront présentés. Une plus grande attention sera accordée aux décompositions d'un tenseur de troisième ordre, puisque ça permettra d'appréhender et de présenter les caractéristiques propres de ces décompositions. Dans certains cas, la généralisation au $N^{\text{ème}}$ ordre sera également donnée. Une étude plus approfondie sera accordée à la décomposition CP dans la section 2.4 puisqu'elle fait l'objective de toutes nos contributions, et dans laquelle nous introduisons une décomposition tensorielle basée sur la décomposition CP qui resolve le problème de l'indétermination de l'échelle. Dont nous prouverons l'existence de cette décomposition et aussi nous étudierons son unicité.

2.2 Les différentes techniques de la décomposition matricielle

2.2.1 La décomposition LU

Toute matrice $\mathbf{A} \in \mathbf{C}^{R \times R}$ peut être décomposée comme :

$$\mathbf{A} = \mathbf{DPLU}, \tag{2.1}$$

avec $\mathbf{D}$ une matrice diagonale, $\mathbf{P}$ une matrice de permutation, $\mathbf{L}$ une matrice triangulaire inférieure ne contenant que des $\mathbf{1}$ sur sa diagonale et $\mathbf{U}$ une matrice triangulaire supérieure ne contenant que des $\mathbf{1}$ sur sa diagonale.

2.2.2 La décomposition QR

Toute matrice $\mathbf{A} \in \mathbf{C}^{R \times R}$ peut s'écrire sous la forme :

$$\mathbf{A} = \mathbf{DQR}, \tag{2.2}$$

avec $\mathbf{D}$ une matrice diagonale, $\mathbf{R}$ une matrice triangulaire ne contenant que des $\mathbf{1}$ sur sa diagonale et $\mathbf{Q}$ une matrice orthogonale. Si $\mathbf{A} \in \mathbf{C}^{R \times R}$ alors $\mathbf{Q}$ est une matrice unitaire.

2.2.3 La décomposition polaire

Toute matrice $\mathbf{A} \in \mathbf{C}^{R \times R}$ peut s'écrire sous la forme :

$$\mathbf{A} = \mathbf{QH}, \tag{2.3}$$

avec $\mathbf{H}$ une matrice symétrique et $\mathbf{Q}$ une matrice orthogonale. Dans le cas complexe $\mathbf{H}$ est une matrice symétrique hermitienne et $\mathbf{Q}$ est une matrice unitaire.

2.2.4 La décomposition polaire algébrique

Nous appelons matrice symétrique complexe toute matrice $\mathbf{H} \in \mathbf{C}^{R\times R}$ vérifiant $\mathbf{H} = \mathbf{H}^T$. De même, nous appelons matrice orthogonale complexe toute matrice $\mathbf{Q} \in \mathbf{C}^{R\times R}$ vérifiant $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$.

Soit $\mathbf{A} \in \mathbf{C}^{R\times R}$ une matrice inversible. Il est montré dans [Kaplansky, 1990] que $\mathbf{A}$ admet une décomposition polaire algébrique

$$\mathbf{A} = \mathbf{Q}\mathbf{H}, \tag{2.4}$$

avec $\mathbf{H}$ une matrice symétrique complexe et $\mathbf{Q}$ une matrice orthogonale complexe.

2.2.5 La décomposition de Schur

Toute matrice carrée $\mathbf{M} \in \mathbf{C}^{R\times R}$ peut s'écrire :

$$\mathbf{M} = \mathbf{Q}\mathbf{T}\mathbf{Q}^H, \tag{2.5}$$

où $\mathbf{Q} \in \mathbf{C}^{R\times R}$ est une matrice unitaire et $\mathbf{T}$ est une matrice triangulaire.

Il est intéressant de noter que la matrice $\mathbf{T}$ contient sur sa diagonale les valeurs propres de la matrice $\mathbf{M}$.

2.2.6 La décomposition en valeur singulière

Toute matrice $\mathbf{A} \in \mathbf{C}^{M\times N}$ peut être décomposée de la manière suivante :

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^H \tag{2.6}$$

- $\mathbf{U} \in \mathbf{C}^{M\times M}$ une matrice unitaire,
- $\mathbf{S} \in \mathbf{C}^{M\times N}$ est une matrice ne contenant que des zéros, excepté sur sa diagonale,
- $\mathbf{V} \in \mathbf{C}^{N\times N}$ une matrice unitaire.

Cette décomposition en valeur singulière (figure 2.1) est notée SVD (*Singular Value Decomposition*). Une SVD peut être tronquée à l'ordre Q, nous aurons alors

- $\mathbf{U} \in \mathbf{C}^{M\times Q}$ telle que $\mathbf{U}^H\mathbf{U} = \mathbf{I}_Q$,
- $\mathbf{S} \in \mathbf{C}^{Q\times Q}$ est une matrice diagonale,
- $\mathbf{V} \in \mathbf{C}^{N\times Q}$ telle que $\mathbf{V}^H\mathbf{V} = \mathbf{I}_Q$.

L'égalité 2.6 reste vraie si et seulement si Q est supérieur ou égal au rang de la matrice $\mathbf{A}$.

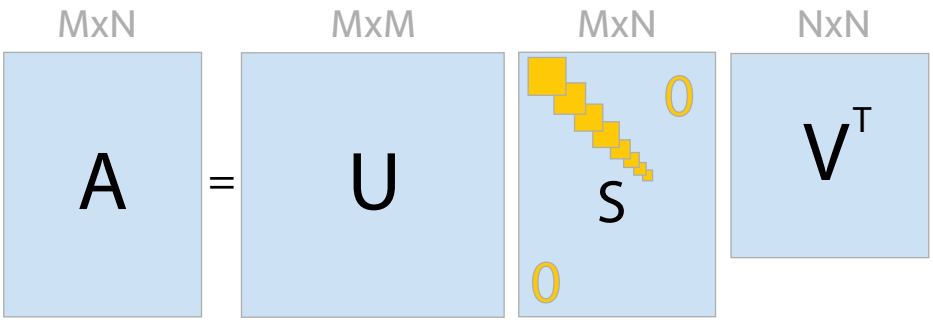


FIGURE 2.1 – Schéma de la décomposition en valeur singulière pour une matrice de taille $M \times N$.

Théorème d'Eckart-Young Soient $\mathbf{A} \in \mathbf{C}^{M \times N}$ et $\mathbf{AQ} = \mathbf{USV}$ la matrice correspondant à la SVD de $\mathbf{A}$ tronquée au rang Q avec $\mathbf{S}$ contenant les Q valeurs singulières de plus grand module de $\mathbf{A}$. Nous avons alors

$$\mathbf{A}_Q = \underset{\mathbf{X}}{\operatorname{argmax}} \|A - X\|_F \tag{2.7}$$

où $\mathbf{X}$ est une matrice de rang Q. Ainsi, $\mathbf{A}_Q$ est la matrice de rang Q approximant le mieux la matrice $\mathbf{A}$ au sens de la norme de Fobenius.

2.3 Les différentes techniques de la décomposition tensorielle

La présente section est dediée à la décomposition des tenseurs d'ordre supérieur. La décomposition tensorielle, aussi appelée analyse factorielle à voies multiples (Multi-way factor analysis), est un domaine d'algèbre multilinéaire, qui caractérise un tenseur comme une combinaison linéaire de produits extérieurs des vecteurs. Selon l'approche considérée, les décompositions de tenseurs peuvent être considérées comme des généralisations de l'analyse en composantes principales (ACP) ou de la décomposition en valeurs singulières (SVD) pour les ordres supérieures à deux. L'analyse d'un tenseur en terme de ses facteurs décomposés est utile dans les problèmes où différents facteurs d'un mélange multilinéaire doivent être identifiés à partir des données mesurées. Dans ce qui suit, quelques décompositions tensorielles seront présentées. Nous détaillerons la décomposition de TUCKER3 qui généralise l'analyse en composantes principales. Par la suite, nous présenterons la décomposition tensorielle de PARAFAC/CANDECOMP, dont l'acronyme utilisé dans ce manuscrit sera la décomposition CP. Une plus grande attention sera accordée aux décompositions tensorielles de troisième ordre, puisque ça permet de bien appréhender et de présenter les caractéristiques propres de ces décompositions.

2.3.1 La décomposition TUCKER3

La décomposition de TUCKER3 a été introduite pour la première fois par TUCKER en 1963 [Tucker, 1963]. Plusieurs noms ont été attribués dans la littérature pour désigner cette décomposition, le tableau 2.1 illustre les noms les plus connus pour cette décomposition.

Nom	Proposé par
Three-mode factor analysis (3MFA/TUCKER3)	[Tucker, 1966]
Three-mode principal components analysis (3MPCA)	[Kroonenberg et De Leeuw, 1980]
N-mode principal components analysis	[Kapteyn *et al.*, 1986]
Higher-order SVD (HOSVD)	[De Lathauwer *et al.*, 2000a]
N-mode SVD	[Vasilescu et Terzopoulos, 2002]

TABLE 2.1 – Quelque noms pour la décomposition Tucker3

La décomposition TUCKER3 est générale, dans le sens où elle incorpore la plupart des autres décompositions tensorielles de troisième ordre comme des cas particuliers. Le modèle de TUCKER3 a été initialement élaboré pour effectuer une décomposition en valeurs singulières sur tous les modes du tenseur, indépendamment les uns des autres.

2.3.1.1 Domaines d'application

La décomposition du Tucker3 trouve des applications dans des domaines très variés, la décomposition Tucker3 trouve sa première utilité en analyse chimique, initiée par HERION [Henrion, 1994]. De nombreux exemples d'applications en psychométrie ont été réalisés par KIERS et VAN MECHELEN [Kiers, 2000]. Ensuite, [De Lathauwer *et al.*, 2000a] ont utilisé la décomposition de TUCKER3 en traitement du signal. MUTI et BOURENNANE [Muti, 2004] l'ont appliqué pour étendre le filtre deWiener pour le débruitage d'images couleurs et des signaux sismiques. Cette technique est utilisée pour modéliser les expressions du visages et la compression d'images [Wang *et al.*, 2003], la décomposition de TUCKER3 trouve aussi un intérêt dans l'extraction des connaissances, ou SAVAS [Savas, 2003] l'exploitent pour la reconnaissance de l'écriture manuscrite. Finalement, cette décomposition a démontré la puissance de l'analyse sous-espace multilinéaire dans le contexte de la reconnaissance faciale [Vasilescu et Terzopoulos, 2002, Vasilescu et Terzopoulos, 2003, Vasilescu et Terzopoulos, 2005, Vasilescu et Terzopoulos, 2007], a travers une représentation multilinéaire, appelée TensorFaces, et qui donne des taux de reconnaissance faciale supérieurs aux approches standard et linéaires tel que le PCA (Eigenfaces).

2.3.1.2 Principe

La décomposition de Tucker3 est une forme d'analyse en composantes principales (ACP). Elle factorise un tenseur $\mathcal{T}$ d'ordre N en un tenseur noyau multiplié par une matrice sur chacun de ses N modes. Cette décomposition d'un tenseur $\mathcal{T}$ d'ordre N s'exprime alors :

$$\mathcal{T} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \ldots \times_N \mathbf{A}^{(N)}$$

Où $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times K_n}$ est une matrice orthogonale telle que :

$$(\mathbf{A}^{(n)})^T \mathbf{A}^{(n)} = \mathbf{I}^{K_n \times K_n}$$

Les matrices $\mathbf{A}^{(1)}, \forall n = 1, \ldots, N$ sont appelées les matrices facteurs, et $\mathcal{G} \in \mathbb{R}^{K_1 \times \ldots \times K_N}$ est le tenseur noyau.

Soit un tenseur $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$ d'ordre 3. La décomposition Tucker3 de ce tenseur consiste à trouver un tenseur noyau $\mathcal{G} = [g_{pqr}] \in \mathbb{R}^{P \times Q \times R}$, et les trois matrices facteurs $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_P] \in \mathbb{R}^{I \times P}$, $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_Q] \in \mathbb{R}^{J \times Q}$, et $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_R] \in \mathbb{R}^{K \times R}$, qui vérifie la relation suivante :

$$\boxed{\mathcal{T} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} (\mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r)} \tag{2.8}$$

Comme il peut s'écrire sous forme d'élément par élément (forme scalaire) :

$$t_{ijk} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr} \tag{2.9}$$

avec a_{ip}, b_{jq} et c_{kr} sont les éléments des trois matrices $\mathbf{A} \in \mathbb{R}^{I \times P}$, $\mathbf{B} \in \mathbb{R}^{J \times Q}$ et $\mathbf{C} \in \mathbb{R}^{K \times R}$ respectivement, et g_{pqr} est l'élément du tenseur noyau $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ dont chaque indice indique le niveau d'interaction entre les différents composants.

À partir de l'equation 2.9, nous pouvons dire que la décomposition du tenseur selon TUCKER3 est une combinaison linéaire des produits extérieurs, où le coefficient de chaque terme de produit extérieur représente l'élément scalaire correspondant du tenseur noyau. Une illustration de la décomposition TUCKER3 est donnée dans la figure 2.2.

Il existe plusieurs descriptions mathématiques équivalentes pour le modèle Tucker (voir le tableau 2.2). Il peut être exprimé en utilisant le produit n-mode

$$\mathcal{T} = \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$$

On peut aussi le mettre sous forme matricielle, ce qui donne pour le premier mode :

$$\mathbf{T}_{(1)} = \mathbf{A}\mathbf{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T$$

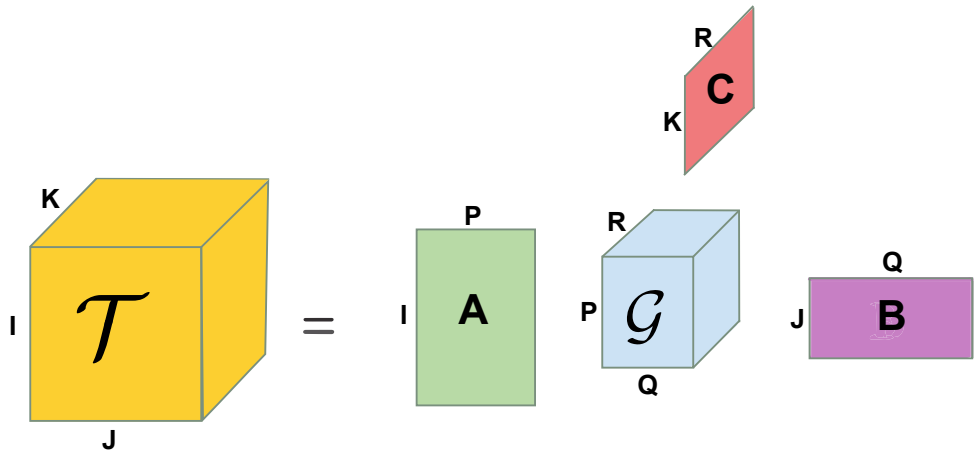


FIGURE 2.2 – Schéma de la décomposition de Tucker3 pour un tenseur d'ordre 3.

On fait alors intervenir le produit de Kronecker, et on déplie le tenseur noyau $\mathcal{G}$ dans le premier mode. Il est bien sûr possible de faire la même chose pour les deux autres modes :

$$\mathbf{T}_{(2)} = \mathbf{B}\mathbf{G}_{(2)}(\mathbf{C} \otimes \mathbf{A})^T$$

$$\mathbf{T}_{(3)} = \mathbf{C}\mathbf{G}_{(3)}(\mathbf{B} \otimes \mathbf{A})^T$$

Il est souvent pratique de les représenter sous formes vectorisés

$$\text{vec}(\mathbf{T}_{(1)}) = \text{vec}(\mathbf{A}\mathbf{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T) = (\mathbf{C} \otimes \mathbf{B}) \otimes \mathbf{A}\text{vec}(\mathbf{G}_{(1)})$$

$$\text{vec}(\mathbf{T}_{(2)}) = \text{vec}(\mathbf{B}\mathbf{G}_{(2)}(\mathbf{C} \otimes \mathbf{A})^T) = (\mathbf{C} \otimes \mathbf{A}) \otimes \mathbf{B}\text{vec}(\mathbf{G}_{(2)})$$

$$\text{vec}(\mathbf{T}_{(3)}) = \text{vec}(\mathbf{C}\mathbf{G}_{(3)}(\mathbf{B} \otimes \mathbf{A})^T) = (\mathbf{B} \otimes \mathbf{A}) \otimes \mathbf{C}\text{vec}(\mathbf{G}_{(3)})$$

2.3.1.3 Le manque d'unicité

La décomposition de Tucker3 n'est pas unique, car il y'a une infinité de solution pour les matrices facteurs et pour le tenseur noyau menant au même tenseur $\mathcal{T}$. En d'autres termes, la décomposition de Tucker3 permet des transformations linéaires arbitraires sur les trois matrices facteurs (prévus que l'inverse de ces transformations est appliquée au tenseur noyau) sans affecter le tenseur reconstruit $\mathcal{T}$. pour voir cela, nous définissons les matrices non singulières $\mathbf{T}_a \in \mathbb{R}^{P\times P}$, $\mathbf{T}_b \in \mathbb{R}^{Q\times Q}$ et $\mathbf{T}_c \in \mathbb{R}^{R\times R}$. Compte tenu de la matrice dépliée $\mathbf{T}_{(1)}$, nous avons ce qui suit :

$$\mathbf{T}_{(1)} = \mathbf{A}\mathbf{T}_a\mathbf{T}_a^{-1}\mathbf{G}_{(1)}(\mathbf{C}\mathbf{T}_c\mathbf{T}_c^{-1} \otimes \mathbf{B}\mathbf{T}_b\mathbf{T}_b^{-1})^T$$

Opérateur	Formule mathématique
Le produit extérieur	$\mathcal{T} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr}(\mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r)$
Scalaire	$t_{ijk} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr}$
Le n-mode produit	$\mathcal{T} = \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$
La représentation par tranche	$\mathbf{T}_k = \mathbf{A}\mathbf{H}_k\mathbf{B}^T (k = 1, \ldots, K)$ avec : $\mathbf{H}_k = \sum_{r=1}^{R} c_{kr}\mathbf{G}_r$
Vecteur	$\text{vec}(\mathcal{T}) = \text{vec}(\mathbf{T}_{(1)}) = (\mathbf{C} \otimes \mathbf{B} \otimes \mathbf{A})\text{vec}(\mathbf{G}_1)$
Le produit de Kronecker	$\mathbf{T}_{(1)} = \mathbf{A}\mathbf{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T$ $\mathbf{T}_{(2)} = \mathbf{B}\mathbf{G}_{(2)}(\mathbf{C} \otimes \mathbf{A})^T$ $\mathbf{T}_{(3)} = \mathbf{C}\mathbf{G}_{(3)}(\mathbf{B} \otimes \mathbf{A})^T$

TABLE 2.2 – Les formulations mathématiques de la décomposition Tucker3 pour un tenseur d'ordre 3

En utilisant la propriété du Produit de Kronecker suivante :
Soit $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times S}$, $\mathbf{C} \in \mathbb{R}^{R \times P}$ et $\mathbf{D} \in \mathbb{R}^{S \times Q}$ on a :

$$\mathbf{AC} \otimes \mathbf{BD} = (\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D})$$

On trouve :

$$\begin{aligned} \mathbf{T}_{(1)} &= \mathbf{A}\mathbf{T}_a\mathbf{T}_a^{-1}\mathbf{G}_{(1)}((\mathbf{C}\mathbf{T}_c \otimes \mathbf{B}\mathbf{T}_b)(\mathbf{T}_c^{-1} \otimes \mathbf{T}_b^{-1}))^T \\ &= (\mathbf{A}\mathbf{T}_a)[\mathbf{T}_a^{-1}\mathbf{G}_{(1)}(\mathbf{T}_c^{-1} \otimes \mathbf{T}_b^{-1})^T](\mathbf{C}\mathbf{T}_c \otimes \mathbf{B}\mathbf{T}_b)^T \end{aligned}$$

$$\mathbf{T}_{(1)} = \tilde{\mathbf{A}}\tilde{\mathbf{G}}_{(1)}(\tilde{\mathbf{C}} \otimes \tilde{\mathbf{B}})^T \tag{2.10}$$

Où $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{T}_a$, $\tilde{\mathbf{B}} = \mathbf{B}\mathbf{T}_b$ et $\tilde{\mathbf{C}} = \mathbf{C}\mathbf{T}_c$ sont des matrices facteurs transformés et $\tilde{\mathbf{G}}_{(1)} = \mathbf{T}_a^{-1}\mathbf{G}_{(1)}(\mathbf{T}_c^{-1} \otimes \mathbf{T}_b^{-1})^T$ est un tenseur noyau transformé. L'équation (2.10) signifie que nous avons un nombre infini de matrices $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{C}}$ et $\tilde{\mathbf{G}}_{(1)}$ donnant lieu à la même matrice $\mathbf{T}_{(1)}$. Ce fait indique clairement la non-unicité de la décomposition de Tucker3.

L'unicité complète des matrices facteurs et du tenseur noyau de la décomposition TUCKER3 n'est possible que dans certains cas spéciaux, où au moins deux matrices facteurs ont une structure spéciale qui permet une détermination unique des matrices de transformation. De plus, il a été montré que l'unicité partielle peut exister dans les cas où le tenseur noyau de la décomposition TUCKER3 est sous la contrainte d'avoir plusieurs éléments égaux à zéro [ten Berge et Smilde, 2002].

2.3.2 Cas spéciaux : Tucker2 et Tucker1

Le modèle Tucker décrit ci-dessus est souvent appelé le modèle Tucker3 car un tenseur du troisième ordre est décomposé en trois matrices facteurs (par exemple, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$) et un tenseur noyau $\mathcal{G}$. Dans les applications où nous avons deux matrices facteurs ou même une seule, le modèle Tucker3 pour un tenseur d'ordre 3 se simplifie dans les modèles Tucker2 ou Tucker1 (voir le tableau 2.3).

Modèle	Description
Tucker2	$t_{ijk} = \sum_{p=1}^{P} \sum_{q=1}^{Q} g_{pqk} a_{ip} b_{jq}$
	$\mathcal{T} = \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B}$
	$\mathcal{T} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \mathbf{a}_p \circ \mathbf{b}_q \circ g_{pq}$
	$\mathbf{T}_{(3)} = \mathbf{G}_{(3)} (\mathbf{B} \otimes \mathbf{A})^T$
Tucker1	$t_{ijk} = \sum_{p=1}^{P} g_{pjk} a_{ip}$
	$\mathcal{T} = \mathcal{G} \times_1 \mathbf{A}$,$\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$
	$\mathbf{T}_k = \mathbf{A}\mathbf{G}_k$, $k = 1, \ldots, K$
	$\mathbf{T}_{(1)} = \mathbf{A}\mathbf{G}_{(1)}$

TABLE 2.3 – Les modèles de Tucker pour un tenseur d'ordre 3

♢ Le modèle Tucker2 peut être obtenue à partir du modèle Tucker3 par l'absorption d'un seul facteur par le tenseur noyau (voir la figure 2.3), c'est-à-dire,

$$\mathcal{T} = \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B}$$

Comme on peut l'écrire par cette forme :

$$t_{ijk} = \sum_{p=1}^{P} \sum_{q=1}^{Q} g_{pqk} a_{ip} b_{jq}$$

♢ Le modèle Tucker1 n'a qu'une seule matrice de facteur (tandis que les deux autres sont absorbés par le tenseur noyau) qui est décrit comme (voir aussi la figure 2.4)

$$\mathcal{T} = \mathcal{G} \times_1 \mathbf{A}$$

Et sous forme élément par élément, il s'écrit par :

$$t_{ijk} = \sum_{p=1}^{P} g_{pjk} a_{ip}$$

Il est intéressant de noter que l'approximation d'un tenseur par des matrices facteurs et d'un tenseur noyau permet souvent de simplifier les opérations mathématiques et de réduire le coût de calcul de certaines opérations en algèbre multilinéaire.

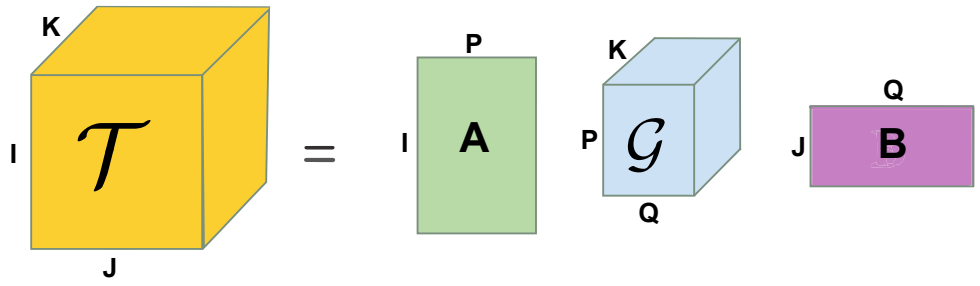


FIGURE 2.3 – Schéma de la décomposition de Tucker2 pour un tenseur d'ordre 3.

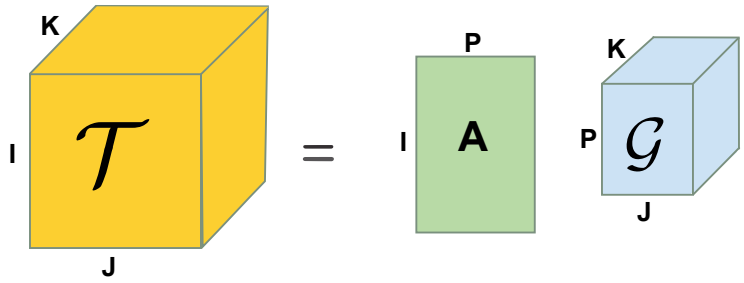


FIGURE 2.4 – Schéma de la décomposition de Tucker1 pour un tenseur d'ordre 3.

2.3.3 La décomposition CP

Comme pour le modèle de TUCKER3, la décomposition CP est connue sous plusieurs noms dans la littérature. On la retrouve en 1927 sous le nom de la décomposition polyadique d'un tenseur (en anglais : Polyadic Decomposition of a Tensor), instauré par HITCHCOCK [Hitchcock, 1927]. Puis c'est en 1944 que CATTELL [Cattell, 1944] a décrit le principe de Parrallel proportional analysis et le concept d'axes multiples pour l'analyse, autrement dit PARAFAC qui est l'abréviation de Parallel Factor Analysis. Cette idée est devenue populaire en 1970, en psychométrie grâce à HARSHMAN [Harshman *et al.*, 1970]. Elle a été ensuite reprise indépendamment par CARROLL et CHANG sous la terminologie Canonical Decomposition(CanDecomp) [Carroll et Chang, 1970]. Cette décomposition est alors répandue sous le nom de Canonical Polyadic Decomposition (CP)[Comon *et al.*, 2009a].

Alors que le modèle de TUCKER3 généralise la décomposition matricielle de la SVD, la décomposition CP généralise cette représentation sous sa forme canonique. le tableau 2.4 résume les différents noms pour la décomposition CP.

Nom	Proposé par
Polyadic Form of a Tensor	[Hitchcock, 1927]
Parallel Factor Analysis (PARAFAC)	[Harshman *et al.*, 1970]
Canonical Decomposition (CanDecomp)	[Carroll et Chang, 1970]
Topographic components model	[Mocks, 1988]
CP (CANDECOMP/PARAFAC)	[Kiers, 2000]

TABLE 2.4 – Quelque noms pour la décomposition CP

La décomposition CP décrit également la structure de base des cumulants d'ordre supérieur des données multivariées sur laquelle toutes les méthodes algébriques pour l'analyse en composantes indépendantes sont fondées [Henrion, 1994, Comon *et al.*, 2008].

2.3.3.1 *Domaines d'application*

La décomposition CP a trouvé plusieurs applications dans des domaines aussi variés que la psychométrie [Carroll et Chang, 1970] et la phonétique [Harshman *et al.*, 1970]. Elle connait également un grand intérêt dans la séparation de sources, l'identification aveugle et dans l'analyse en composantes indépendantes (ACI) [Comon, 1994, De Lathauwer *et al.*, 1995a, Comon et Jutten, 2010, Sørensen et De Lathauwer, 2013] ainsi qu'en télécommunications pour les signaux CDMA (Code Division Multiple Access) et les technologies MIMO (Multi-Input Multi-Output) [Castella *et al.*, 2006b, Nion, 2007, Sidiropoulos *et al.*, 2000a, de Almeida *et al.*, 2007, Sørensen et De Lathauwer, 2013]. Aussi, cette décomposition trouve un intérêt dans les grands enjeux actuels en matière d'informatique, qui sont l'apprentissage automatique (machine learning) et le big data [Zhang *et al.*, 2017, Zhang *et al.*, 2018, Gao *et al.*, 2019, Bu, 2018].

2.3.3.2 *Principe*

La décomposition CP d'un tenseur $\mathcal{T}$ d'ordre N de taille $I_1 \times I_2 \times \ldots \times I_N$ est la décomposition de ce tenseur sous forme de la somme d'un nombre minimum de tenseurs de rang 1 :

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \ldots \circ \mathbf{a}_r^{(N)}$$

Où $\mathbf{a}_r^{(n)} \in \mathbb{R}^{I_n}, r \in [1:R], n \in [1:N]$
Ce nombre minimum de termes correspond au rang du tenseur. On notera par ailleurs $\mathbf{A}^{(n)}$ la matrice de taille $I_n \times R$ obtenue en concaténant les vecteurs $\mathbf{a}_1^{(n)}, \mathbf{a}_2^{(n)}, \ldots, \mathbf{a}_R^{(n)}, n \in [1:N]$

Si on considère un tenseur $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$ d'ordre 3, alors il admet une décomposition CP sous forme d'une somme de tenseurs de rang 1 :

$$\boxed{\mathcal{T} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r} \tag{2.11}$$

Avec $\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$, $r \in [1 : R]$, sont des tenseurs de rang 1, et où les trois matrices $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}, \mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_R] \in \mathbb{R}^{J \times R}, \mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_R] \in \mathbb{R}^{K \times R}$ sont les matrices facteurs dont les colonnes sont appelées les facteurs. Lorsque le nombre entier R est minimal, qui est le nombre de tenseurs de rang 1 nécessaires au maintien de l'égalité 2.11, on parle alors de la décomposition canonique polyadique (CP), où R représente le rang du tenseur $\mathcal{T}$. Un schéma de décomposition CP d'un tenseur d'ordre 3 est présenté dans la figure 2.5.

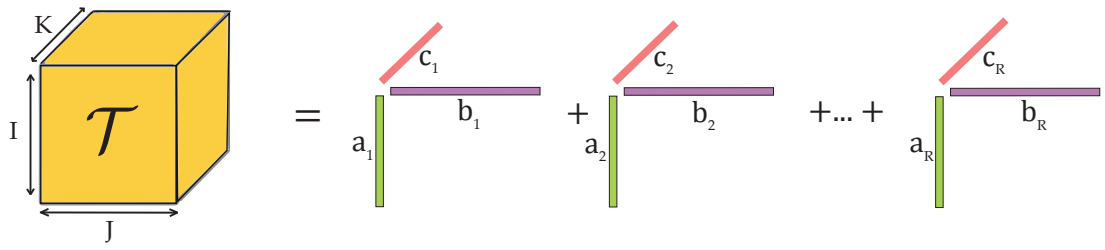


FIGURE 2.5 – Schéma de la décomposition CP pour un tenseur d'ordre 3.

Cette décomposition peut s'écrire selon Kolda [Kolda et Bader, 2009] (voir aussi Kruskal [Kruskal, 1989]) par une notation plus concise comme suit :

$$\mathcal{T} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

On peut réécrire cette relation en intervenant les composantes des matrices facteurs :

$$t_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$$

En utilisant, le produit n-mode on trouve :

$$\mathcal{T} = \mathcal{I} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$$

Une écriture matricielle, couramment utilisée, considère les tranches du tenseur représentées dans la figure 1.4. Notamment, la décomposition CP pour un tenseur $\mathcal{T}$ d'ordre 3 est aussi parfois écrite pour les k tranches frontales :

$$\mathbf{T}_k = \mathbf{A}\mathbf{D}_C^{(k)}\mathbf{B}^T \tag{2.12}$$

où $\mathbf{D}_C^{(k)} \equiv diag(\mathbf{c}_{k:})$, $k = 1, \ldots, K$ est la matrice diagonale qui contient la $k^{\text{ième}}$ ligne de $\mathbf{C}$ sur sa diagonale.
En termes de représentations matricielles du tenseur $\mathcal{T}$, la décomposition CP peut s'écrire :

$$\mathbf{T}_{(1)}^{I\times KJ} = \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T,$$
$$\mathbf{T}_{(2)}^{J\times KI} = \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T,$$
$$\mathbf{T}_{(3)}^{K\times JI} = \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T.$$

Opérateur	Formulation
Le produit extérieur	$\mathcal{T} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$
Scalaire	$t_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$
Le n-mode produit	$\mathcal{T} = \mathcal{I} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$
La représentation par tranche	$\mathbf{T}_k = \mathbf{A}\mathbf{D}_C^{(k)}\mathbf{B}^T$ avec : $\mathbf{D}_C^{(k)} \equiv diag(\mathbf{c}_{k:})$
Le produit de Khatri-Rao	$\mathbf{T}_{(1)}^{I\times KJ} = \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T$ $\mathbf{T}_{(2)}^{J\times KI} = \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T$ $\mathbf{T}_{(3)}^{K\times JI} = \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T$

TABLE 2.5 – Les formules mathématiques associés à la décomposition CP pour un tenseur d'ordre 3

En pratique, on préfère souvent s'adapter à un modèle multilinéaire de rang inférieur, R $<$ rang($\mathcal{T}$), fixé à l'avance, afin de faire face à un problème d'approximation. Plus précisément, il vise à minimiser une fonction objectif de la forme :

$$\boxed{\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) = \|\mathcal{T} - \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\|_F^2} \tag{2.13}$$

Malheureusement, le problème de l'approximation n'est pas toujours bien posé. En fait, comme l'a souligné R.Harshman [Harshman, 2004], la borne inférieure ne peut jamais être atteinte, car cette approximation peut être de rang plus élevé que R. Selon la classification proposée par Richard Harshman [Kruskal *et al.*, 1989, Comon *et al.*, 2009a], ce phénomène est appelé CP-degeneracies. Il peut se produire lorsqu'on tente d'approximer un tenseur par un autre de rang inférieur. Plusieurs exemples qui expliquent ce

phénomène sont donnés dans [Comon *et al.*, 2008, De Silva et Lim, 2008].

Il existe de nombreuses situations où les algorithmes pour le calcule de la décomposition CP (2.13) peuvent souffrir de l'absence ou de la lenteur de la convergence, et qui peuvent être dues à des dégénérescences du tenseur. Ces situations ont été bien classées par Richard Harshman [Harshman, 2004] dans les trois cas suivants :

— Goulot d'étranglement (Bottleneck) : Un goulot d'étranglement survient lorsque deux ou plusieurs facteurs dans l'un des modes sont presque colinéaires [Rajih *et al.*, 2008].

— Marécage (Swamp) : Le phénomène de marécage se produit lorsque tous les modes présentent au moins deux facteurs quasi colinéaires [Rajih *et al.*, 2008, Rayens et Mitchell, 1997], ce qui peut être considéré comme un cas général de goulot d'étranglement.

— Dégénérescences de la CP (CP-degeneracies) : Les dégénérescences de CP peuvent être considérées comme un cas particulier de phénomène de marécage, où certains des facteurs se rapprochent et, en même temps, ont tendance à s'annuler, ce qui conduit à une meilleure qualité de l'ajustement [Harshman, 2004, Paatero, 2000].

2.3.3.3 Unicité

Une des propriétés intéressantes propres aux tenseurs d'ordre supérieur à deux, est que leur décomposition tensorielle CP est *souvent unique*, ce qui n'est pas le cas des décompositions matricielles [Sidiropoulos et Bro, 2000, Harshman *et al.*, 1970, Kolda et Bader, 2009]. La décomposition d'une matrice en une somme de matrices de rang un existe elle aussi, mais elle n'est pas unique, sauf si l'on impose certaines contraintes fortes sur les matrices de la décomposition, par exemple une condition d'orthogonalité.
L'unicité signifie qu'il n'y a qu'une seule combinaison de tenseurs de rang un qui par sommation génèrent le tenseur $\mathcal{T}$, à l'exception de *l'indétermination d'échelle et de permutation*. Lorsque la décomposition est unique à ces deux indéterminations près, on dit qu'elle est essentiellement unique.

Le résultat le plus reconnu d'unicité est présenté par KRUSKAL [Kruskal, 1977]. L'étude de la condition d'unicité de la décomposition CP est basée sur un concept fondamental, qui est le concept de **k-rang** *(Kruskal-rank)*, plus restreinte que la notion usuelle du rang matriciel. Le concept k-rang a été largement utilisé comme un concept clé indiquant l'unicité de CP.

♢ **Rang de Kruskal d'une matrice :**
Le rang de kruskal d'une matrice $\mathbf{A}$, ou k-rang de $\mathbf{A}$, noté $K_{\mathbf{A}}$, est défini comme le nombre maximum tel que tout ensemble composé de k colonne de $\mathbf{A}$ soit linéairement indépendant.
★ Le K-rang d'une matrice est toujours inférieur ou égal à son rang, et nous avons :

$$K_{\mathbf{A}} \leq r_{\mathbf{A}} \leq min(I, J)$$

.

Exemple 1 : Soit la matrice $\mathbf{A}$ de taille 3×3 définie par :

$$\mathbf{A} = \begin{bmatrix} 5 & 2 & 1 \\ 7 & 4 & 2 \\ 5 & 6 & 3 \end{bmatrix}$$

$\mathbf{A}$ est de rang 2 mais de K-rang 1 car tous les ensembles de 2 colonnes ne sont pas linéairement indépendants(la $2^{\text{ème}}$ et $3^{\text{ème}}$ la colonne sont colinéaires).Il n'y a que les ensembles de 1 colonne qui sont tous linéairement indépendants.

Exemple 2 : Soit la matrice $\mathbf{A}$ de taille 3×3 définie par :

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 5 & 2 & 3 \\ 8 & 3 & 5 \end{bmatrix}$$

$\mathbf{A}$ est de rang 2 (la $1^{\text{ère}}$ colonne est la somme de la $2^{\text{ème}}$ et de la $3^{\text{ème}}$ la colonne). Il n'y a qu'un ensemble de 3 colonnes (la matrice elle-même, et elle n'est pas de rang plein). Par contre, tous les ensembles de 2 colonnes sont linéairement indépendants. Dons son K-rang est 2.

Condition de kruskal : Kruskal a établi en 1977 [Kruskal, 1977] une condition suffisante qui garantit l'unicité de la décomposition CP pour les tenseurs d'ordre 3 laquelle s'écrit :

$$2R + 2 \leq min(I, R) + min(J, R) + min(K, R) \tag{2.14}$$

En effet, pour des matrices tirées aléatoirement selon un distribution continue, le K-rang et le rang coïncident avec une probabilité égale à 1, et le rang de A par exemple vaut min(I,R). Le résultat présenté ci-dessus (2.14), est fortement lié au concept du rang tensoriel. Elle impose une condition suffisante qui implique l'unicité de la décomposition CP pour un tenseur d'ordre 3. Cette condition est donnée par le théorème :

Théorème 1 :
Soit un tenseur d'ordre 3 et de rang R, qui se décompose sous la forme 2.11. Si

$$2R + 2 \leq K_{\mathbf{A}} + K_{\mathbf{B}} + K_{\mathbf{C}} \tag{2.15}$$

alors cette décomposition est essentiellement unique. $K_{\mathbf{A}}, K_{\mathbf{B}}$ et $K_{\mathbf{C}}$ représentent les K-rangs des matrices facteurs.

Cette preuve de l'unicité de la solution a par la suite été étendue à des tenseurs d'ordre supérieur à 3 par Sidiropoulos et Bro [Sidiropoulos et Bro, 2000] qui ont imposé une condition suffisante qui implique l'unicité de la décomposition CP pour un tenseur d'ordre N :

$$\boxed{2R + (N+1) = \sum_{n=1}^{N} K_{\mathbf{A}(n)}} \tag{2.16}$$

On remarque bien que pour le N^{eme} ordre, les conditions nécessaires découlent directement de celles du troisième ordre données dans l'équation 2.15.

De façon intuitive, on peut démontrer que la décomposition CP ne souffre pas d'ambiguïté. Considérons le modèle CP écrit en (2.12). En prenant les matrices $\mathbf{H}$ et $\mathbf{P}$, toutes deux de taille $R \times R$ et non singulières, on peut écrire la relation suivante sans changer la matrice $\mathbf{T}$:

$$\mathbf{T}_k = \mathbf{A}\mathbf{H}\mathbf{H}^{-1}\mathbf{D}_C^{(k)}\mathbf{P}\mathbf{P}^{-1}\mathbf{B}^T$$

On se retrouverait donc avec les matrices facteurs $\mathbf{AH}$, $\mathbf{H}^{-1}\mathbf{D}_C^{(k)}\mathbf{P}$, $\mathbf{P}^{-1}\mathbf{B}^T$.
Or, $\mathbf{H}^{-1}\mathbf{D}_C^{(k)}\mathbf{P}$ doit être diagonale, ce qui implique que $\mathbf{H}$ et $\mathbf{P}$ doivent être choisies comme matrices de permutations ou de facteurs d'échelle. De cette manière on ne peut donc être confronté qu'à des indéterminations de permutations ou d'échelle, qui, comme dit précédemment, ne sont pas obstacle à la détermination d'une solution unique.

2.3.3.4 *Estimation des matrices facteurs*

Le problème ici posé consiste donc à estimer les 3 matrices facteurs $\mathbf{A}$,$\mathbf{B}$ et $\mathbf{C}$, en supposant que le rang R est connu. Un moyen classique de modéliser ce problème consiste à se ramener à un problème d'optimisation en cherchant alors à minimiser une fonction de coût judicieusement choisie $\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C})$. On écrit généralement ce problème comme un problème d'ajustement de modèle en choisissant alors de minimiser l'erreur quadratique :

$$\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \|\mathcal{T} - \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\|_F^2 \tag{2.17}$$

Cette fonction de coût peut également s'écrire sous les trois formes équivalentes :

$$\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \|\mathbf{T}_{(1)}^{I \times KJ} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T\|_F^2$$

$$\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \|\mathbf{T}_{(2)}^{J \times KI} - \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T\|_F^2$$

$$\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \|\mathbf{T}_{(3)}^{K \times JI} - \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T\|_F^2$$

Dans la littérature, on peut trouver plusieurs solutions pour résoudre ce problème d'optimisation. L'approche la plus populaire est d'appliquer la technique de moindres carrés alternés *(Alternating Least Squares ALS)* [Carroll et Chang, 1970, Harshman *et al.*, 1970, Bro, 1998].
Pour pouvoir utiliser d'autres types d'algorithmes d'optimisation, la différentielle $d\Upsilon$ de Υ doit être calculée, et finalement, les gradients matriciels (la matrice $\nabla_{\mathbf{A}}\Upsilon$ de taille $I \times R$, la matrice $\nabla_{\mathbf{B}}\Upsilon$ de taille $J \times R$, et la matrice $\nabla_{\mathbf{C}}\Upsilon$ de taille $K \times R$) peuvent être évalués. (le détail de calcul des gradients matriciels est donné dans l'annexe A.1).

$$\begin{aligned}\nabla_{\mathbf{A}}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{\partial\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{A}} \\ &= \frac{\partial}{\partial\mathbf{A}}(\|\mathbf{T}_{(1)}^{I\times KJ} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T\|_F^2) \\ &= 2[-\mathbf{T}_{(1)}^{I\times KJ} + \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T](\mathbf{C}\odot\mathbf{B}),\end{aligned}$$

$$\begin{aligned}\nabla_{\mathbf{B}}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{\partial\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{B}} \\ &\frac{\partial}{\partial\mathbf{B}}(\|\mathbf{T}_{(2)}^{J\times KI} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T\|_F^2) \\ &= 2[-\mathbf{T}_{(2)}^{J\times KI} + \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T](\mathbf{C}\odot\mathbf{A}),\end{aligned}$$

$$\begin{aligned}\nabla_{\mathbf{C}}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{\partial\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{C}} \\ &\frac{\partial}{\partial\mathbf{C}}(\|\mathbf{T}_{(3)}^{K\times JI} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T\|_F^2) \\ &= 2[-\mathbf{T}_{(3)}^{K\times JI} + \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T](\mathbf{B}\odot\mathbf{A}),\end{aligned}$$

En égalisant les composantes du gradient à 0, on obtient la solution suivante :

$$\begin{aligned}\hat{\mathbf{A}} &= \mathbf{T}_{(1)}^{I\times KJ}((\mathbf{C}\odot\mathbf{B})^T)^\dagger \\ \hat{\mathbf{B}} &= \mathbf{T}_{(2)}^{J\times KI}((\mathbf{C}\odot\mathbf{A})^T)^\dagger \\ \hat{\mathbf{C}} &= \mathbf{T}_{(3)}^{K\times JI}((\mathbf{B}\odot\mathbf{A})^T)^\dagger\end{aligned}$$

2.4 La décomposition CP et les indéterminations d'échelle et de permutation

De nombreuses décompositions tensorielles sont présentes dans la littérature. Elles divergent en fonction des données pour lesquelles elles sont prédestinées. Dans notre cas, la décomposition tensorielle qui nous intéressera est la CP. Comme présenté dans la section

2.3.3, l'unicité de la décomposition CP peut etre prouvée, mais qu'à deux indéterminations près.

Dans cette section, nous présenterons les deux indéterminations de la décomposition tensorielle CP, ainsi que les solutions existantes dans la littérature jusqu'à nos jours. En suite, nous introduirons la décomposition tensorielle CP avec isolation du facteur d'échelle. Ce facteur sera calculé de deux manières différentes : Dans un premier temps, nous calculerons le facteur d'échelle comme étant le produit des normes des matrices facteurs. La deuxième solution qui nous sera très utile dans nos contributions consiste à trouver le facteur d'échelle optimal qui minimise la fonction objectif (équation 2.13). Après le nouveau conditionnement du problème, nous prouverons l'existence et l'unicité de la décomposition CP avec le facteur d'échelle optimal.

2.4.1 Indéterminations d'échelle et de permutation

La décomposition CP (2.11) n'est unique qu'à deux indéterminations près : l'indétermination d'échelle et l'indétermination de permutation. En effet, la décomposition (2.11) peut aussi s'écrire :

$$\boxed{\mathcal{T} = \sum_{r=1}^{R} \left(\frac{1}{\lambda_r(\mathbf{A})\lambda_r(\mathbf{B})\lambda_r(\mathbf{C})} \right) \lambda_r(\mathbf{A})\mathbf{a}_r \circ \lambda_r(\mathbf{B})\mathbf{b}_r \circ \lambda_r(\mathbf{C})\mathbf{c}_r} \tag{2.18}$$

Si $\lambda_r(\mathbf{A})\lambda_r(\mathbf{B})\lambda_r(\mathbf{C}) = 1$, les vecteurs $\lambda_r(\mathbf{A})\mathbf{a}_r$, $\lambda_r(\mathbf{B})\mathbf{b}_r$ et $\lambda_r(\mathbf{C})\mathbf{c}_r$ sont donc des solutions au même titre que les vecteurs $\mathbf{a}_r$, $\mathbf{b}_r$ et $\mathbf{c}_r$. Il s'agit de l'ambiguité d'échelle.

De plus, l'ordre des termes $\mathbf{a}_r$, $\mathbf{b}_r$ et $\mathbf{c}_r$ est arbitraire. Rien ne garantit que cet ordre sera respecté pendant la reconstruction de celles-ci. Il s'agit de l'ambiguité de permutation.

En d'autres termes, les matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ sont uniques à une permutation près, de même qu'à un facteur d'échelle près des colonnes des matrices. Cela signifie que toutes matrices $\widetilde{\boldsymbol{A}}$, $\widetilde{\boldsymbol{B}}$ et $\widetilde{\boldsymbol{C}}$ satisfaisant l'équation 2.12 sont liées à $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ par :

$$\widetilde{\boldsymbol{A}} = \mathbf{A}\mathbf{\Pi}\mathbf{\Lambda}_1, \quad \widetilde{\boldsymbol{B}} = \mathbf{B}\mathbf{\Pi}\mathbf{\Lambda}_2, \quad \widetilde{\boldsymbol{C}} = \mathbf{C}\mathbf{\Pi}\mathbf{\Lambda}_3,$$

où $\mathbf{\Pi}$ est la matrice des permutation et $\mathbf{\Lambda}_1$, $\mathbf{\Lambda}_2$ et $\mathbf{\Lambda}_3$ sont les matrices d'échelle diagonales, tel que $\mathbf{\Lambda} = Diag\{\lambda_1, ..., \lambda_R\}$ satisfaisant la condition :

$$\mathbf{\Lambda}_1\mathbf{\Lambda}_2\mathbf{\Lambda}_3 = \mathbf{I},$$

avec $\mathbf{I}$ la matrice d'identité de taille $R \times R$. Dans la littérature des tenseurs, l'optimisation de la décomposition CP (2.13) est effectuée sans isoler le facteur d'échelle $\mathbf{\Lambda}$, qui est généralement compris dans l'une des matrices facteurs, de telle sorte à avoir $\mathbf{\Lambda} = \mathbf{I}$.

Dans un premier temps, nous mettons les facteurs λ_r à l'extérieur du produit. Les indéterminations d'échelle sont alors nettement réduites au norme unité, mais ne sont pas complètement fixées, d'où la difficulté d'estimer l'erreur d'identification des matrices facteurs $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$. Une deuxième démarche plus récente qui sera présentée en détaille dans la prochaine section, consiste à calculer la valeur du facteur d'échelle optimal. Cela nous permet de contrôler le conditionnement du problème.

2.4.2 La décomposition CP en isolant le facteur d'échelle

L'objectif est d'identifier tous les paramètres du côté droit de l'équation (2.11), étant donné le tenseur $\mathcal{T}$. Selon les résultats existants [Ten Berge et Sidiropoulos, 2002, Sidiropoulos *et al.*, 2000a, Kruskal, 1977], un tenseur du troisième ordre et de rang R peut être représenté de façon unique comme somme de R tenseurs de rang 1 sous certaines conditions. Comme mentionné dans la section 2.3.3.3, Kruskal a démontré que la condition $k_A + k_B + k_C \geq 2R+2$ est suffisante pour garantir l'unicité de la décomposition CP [Kruskal, 1977]. Cela signifie que sous la condition de Kruskal, les matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ sont uniques à une permutation près et à un facteur d'échelle de ses colonnes. Pour l'unicité, Harshman a montré qu'il est suffisant d'avoir $\mathbf{A}$ et $\mathbf{B}$ de rang plein, et $\mathbf{C}$ de k-rang 2 [Harshman *et al.*, 1970]. Lorsque $1 < R \leq 2$, les conditions de Kruskal et de Harshman sont équivalentes. Si $R > 2$, la condition de Kruskal peut être satisfaite même si celle de Harshman ne l'est pas, et cette condition est prétendue être juste suffisante pour $R > 3$ [Sidiropoulos *et al.*, 2000a]. Cependant, les observations sont effectivement corrompues par le bruit, de sorte que l'équation (2.11) ne tient pas exactement.

2.4.2.1 *Approximation de rang inférieur*

Calculer la décomposition CP de $\mathcal{T}$ consiste à estimer les matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$, ce qui peut être fait par la minimisation de la fonction de coût suivante :

$$\boxed{\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C};\mathbf{\Lambda}) = \|\mathcal{T} - (\mathbf{A},\mathbf{B},\mathbf{C}.\mathbf{\Lambda})\|_F^2} \tag{2.19}$$

tel que $\mathbf{\Lambda}$ denote un tenseur diagonal de taille $R \times R \times R$, dont les éléments diagonaux λ_r, et $(\mathbf{A},\mathbf{B},\mathbf{C}).\mathbf{\Lambda}$ est un ersatz d'un tenseur de trois dimensions de coordonnées : $\Sigma_r a_{ir} b_{jr} c_{kr}$.

Minimiser l'erreur quadratique (2.19) revient à trouver le meilleur rang R approximatif de $\mathcal{T}$ et sa décomposition CP [Comon et Lim, 2011]. La fonction de coût (2.19) peut aussi s'écrire sous trois formes compactes équivalentes relativement aux trois matrices facteurs :

$$\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C};\mathbf{\Lambda}) = \|\mathbf{T}_{(1)}^{I\times KJ} - \mathbf{A}\mathbf{\Lambda}(\mathbf{C}\odot\mathbf{B})^T\|_F^2 \tag{2.20}$$

$$\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C};\mathbf{\Lambda}) = \|\mathbf{T}_{(2)}^{J\times KI} - \mathbf{B}\mathbf{\Lambda}(\mathbf{C}\odot\mathbf{A})^T\|_F^2 \tag{2.21}$$

$$\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C};\mathbf{\Lambda}) = \|\mathbf{T}_{(3)}^{K\times JI} - \mathbf{C}\mathbf{\Lambda}(\mathbf{B}\odot\mathbf{A})^T\|_F^2 \tag{2.22}$$

2.4.2.2 *Conditionnement du problème*

Dans les algorithmes numériques, il est utile de fixer les indéterminations. Dans [Kolda et Bader, 2009], Kolda et Bader ont proposé de réduire les indéterminations en

normalisant les vecteurs et stocker les normes en $\mathbf{\Lambda}$. Une autre approche décrite dans [Rouijel *et al.*, 2014] ont proposé de tirer les facteurs λ_r à l'extérieur du produit, et calculer la valeur optimale du facteur d'échelle. Nous expliquons plus en détaille cette dernière approche car elle nous permet de bien contrôler le conditionnement du problème.

Supposant que les matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ sont données, nous allons calculer la valeur optimale du facteur d'échelle $\mathbf{\Lambda}$. Cela peut être fait en développant la norme de Frobenius dans l'équation (2.19), qui est une forme quadratique des entrées λ_r :

$$\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}; \mathbf{\Lambda}) = \|\mathcal{T}\| - \Sigma_p \lambda_p f_p^* - \Sigma_q \lambda_q^* f_q + \Sigma_{pq} \lambda_p \lambda_q^* f G_{pq}$$

et en annulant le gradient par rapport à λ_r (voir détails dans l'annexe A.2). Ensuite, le système linéaire est obtenu :

$$\mathbf{G}\lambda = \mathbf{f}, \tag{2.23}$$

où $\mathbf{f}$ est un vecteur de dimension $1 \times R$ défini par la contraction : $f_r = \Sigma_{ijk} t_{ijk} a_{ir} b_{jr} c_{kr}$, et $\mathbf{G}$ représente la matrice de Gram de dimension $R \times R$ définie par :

$$G_{pq} = (a_p \otimes b_p \otimes c_p)^H (a_q \otimes b_q \otimes c_q) \tag{2.24}$$

Rappelons la définition d'une matrice de Gram.

2.4.2.3 *Matrice de Gram*

Soit E un espace vectoriel euclidien et $u = (u_1, ..., u_n)$ une famille de n vecteurs, la matrice de Gram de cette famille est la matrice formée par tous les produits scalaires $\langle u_i, u_j \rangle$. Son déterminant noté $Gram(u)$ est appelé le déterminant de Gram de u.

Introduisons aussi le concept de cohérence que nous utiliserons dans la suite de conditionnement de problème. La notion de cohérence a reçu différents noms dans la littérature : l'incohérence mutuelle de deux dictionnaires [Donoho et Elad, 2003], la cohérence mutuelle de deux dictionnaires [Candes et Romberg, 2007], la cohérence d'une projection de sous-espace [Candès et Tao, 2010], etc. La version ici suit celle de [Gribonval et Nielsen, 2003].

2.4.2.4 *Cohérence*

La cohérence d'un ensemble de vecteurs de normes unitaires est définie comme la valeur maximale du module de produits scalaires croisées :

$$\mu_{\mathbf{A}} = \sup_{p \neq q} |a_p^H a_q| \tag{2.25}$$

Nous définissons de cette façon les cohérences $\mu_{\mathbf{A}}$, $\mu_{\mathbf{B}}$ et $\mu_{\mathbf{C}}$ associées aux matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$, respectivement, avec a_r, b_r et c_r leurs colonnes. Au vue de la matrice $\mathbf{G}$, nous pouvons voir que les cohérences jouent un rôle dans le conditionnement du problème. À partir des équations (2.23) et (2.24), et puisque les entrées diagonales de $\mathbf{G}$

sont égales à 1, il est clair que le fait d'imposer aux produits scalaires croisés $a_p^H a_q$ d'avoir un module strictement inférieur à 1, conduira à un conditionnement acceptable [Comon *et al.*, 2013, Lim et Comon, 2010]. Il est à noter également que les produits scalaires n'apparaissent pas individuellement (2.24), mais à travers leurs produits, puisque les entrées de $\mathbf{G}$ peuvent également être écrites comme : $G_{pq} = a_p^H a_q b_p^H b_q c_p^H c_q$. Cela a des implications très importantes, en particulier dans l'existence et l'unicité de la solution du problème (2.19), que nous élaborerons par la suite.

2.4.2.5 *Existence*

La fonction objectif (2.19) n'est pas coercive, ce qui explique pourquoi le minimum ne peut pas exister. En effet, ce phénomène où un tenseur ne parvient pas à avoir une meilleure approximation de rang R est beaucoup plus répandu qu'on ne l'imagine. Il se produit sur un large éventail de dimensions, des ordres et des rangs, et indépendamment du choix de la norme utilisée. Mais avec une condition supplémentaire sur la cohérence, nous serons en mesure de prouver l'existence grâce à la coercivité. Le théorème suivant montre que la solution au problème (2.19) existe toujours en ajoutant la condition sur la cohérence.

Théorème 2

Définissons trois cohérences $\mu_{\mathbf{A}}$, $\mu_{\mathbf{B}}$ et $\mu_{\mathbf{C}}$ associées aux matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$.

$$\mu_{\mathbf{A}}\mu_{\mathbf{B}}\mu_{\mathbf{C}} \leq \frac{1}{R-1} \tag{2.26}$$

alors, la borne inférieure de l'équation 2.19 est atteinte.

Avec cette condition, l'existence d'une meilleure approximation de rang inférieur est prouvée (Voir la démonstration dans l'annexe A.3). On peut constater que cette condition donne déjà une limite quantitative pour le conditionnement de l'équation (2.23), en raison des cohérences qui limitent les entrées extra diagonales de la matrice $\mathbf{G}$, possèdant des 1 sur sa diagonale [Comon *et al.*, 2013].

2.4.2.6 *Unicité*

Afin de relier l'unicité et la minimalité des décompositions multilinéaires à la cohérence, nous avons besoin d'une simple observation sur la notion de *rang de Kruskal* introduite dans la section 2.3.3.3. Selon le lemme proposé dans [Lim et Comon, 2010], on peut observer que $k_{\mathbf{A}} \geq \frac{1}{\mu_{\mathbf{A}}}$, aussi longtemps que $k_{\mathbf{A}}$ est strictement inférieur au rang des colonnes de $\mathbf{A}$. Inclure cette inégalité dans l'équation (2.15), conduit à la condition unique et suffisante suivante pour un tenseur d'ordre 3 :

$$\mu_{\mathbf{A}}^{-1} + \mu_{\mathbf{B}}^{-1} + \mu_{\mathbf{C}}^{-1} \geq 2(R+1) \tag{2.27}$$

Une généralisation de cette condition pour des tenseurs d'ordre d sera comme suit :

$$\Sigma_{k=1}^{d} \mu_k^{-1} \geq 2R + d - 1 \tag{2.28}$$

2.5 Bilan du chapitre

Dans ce chapitre, nous avons fourni quelques éléments fondamentaux de l'algèbre multilinéaires et des décompositions tensorielles. Ces outils mathématiques offrent une meilleure capacité de modélisation pour des données multimensionnelles.

Dans la première partie de ce chapitre, nous avons aligné plusieurs décompositions matriciels, en suite, nous avons présenté plusieurs décompositions tensorielles qui sont importantes dans le contexte de cette thèse. Ces décompositions ont été formulées à la fois en scalaire (multi-indexé) et en tenseur formes (produit extérieur ou un produit mode-n). La propriété d'unicité de ces décompositions a été étudiée, et ses conditions ont été présentées pour les différentes décompositions.

Une grande attention a été accordée dans la dernière partie à la décomposition CP avec isolement du facteur d'échelle. Le principe de cette décomposition, comme décrit dans ce chapitre, consiste à calculer la valeur optimal du facteur d'échelle que nous isolons, et qui est mis en dehors des matrices facteurs dans la décomposition CP. Nous avons montré que dans la décomposition tensorielle CP, le conditionnement du calcul de la matrice de facteur d'échelle optimale Λ dépend de la cohérence via une matrice de Gram. Cela a des implications très importantes sur l'existence et l'unicité d'une décomposition CP approximative du tenseur $\mathcal{T}$. En effet, en introduisant une condition sur la cohérence, nous garantissons l'existence d'une décomposition approximative de rang inférieur. De plus, nous avons donné une condition suffisante qui prouve l'unicité de la décomposition tensorielle. L'isolement de la matrice de facteur d'échelle permet de réduire les indéterminations d'échelle, cela permettra de contrôler le conditionnement du problème.

Dans les chapitres suivants, nous présentons les différentes méthodes (ou algorithmes) itératives de calcul d'une décomposition CP de rang donné. Nous distinguons deux principaux types d'algorithmes : **les algorithmes alternés**, où on va introduire plus en détail l'algorithme des moindres carrés alternés ALS (Alternating Least Squares), fréquemment utilisé dans la littérature pour optimiser la décomposition tensorielle CP et que nous avons déjâ évoqué lors de la section 2.3.3.4, et **les algorithmes de descente** qui sont des algorithmes d'optimisations à direction de descente qui nécessite le calcul du gradient de la fonction de coût par rapport aux paramètres à estimer tel que : les méthodes basées sur le gradient, les méthodes de Newton et les méthodes utilisant des directions conjuguées pour le cas sans contrainte, et tel que le gradient projeté et les algorithmes proximaux qui représentent la plupart de nos contributions pour le cas avec contrainte.

Chapitre

3 L'OPTIMISATION CP-DÉCOMPOSITION SANS CONTRAINTE

Sommaire

3.1 Introduction

L'optimisation mathématique a plusieurs aspects interagissants : la formulation des problèmes eux-mêmes, la conception d'algorithmes pour les résoudre, l'étude de leur propriété théorique, leur implémentation et leur mise en oeuvre pratique. Chacun de ces points fait l'objet de livres entiers et de nombreux articles de recherche ; ici, nous n'allons qu'effleurer ces questions dans la perspective de l'analyse tensorielle. Nous allons insister sur la structure particulière des problèmes d'optimisation de la décomposition CP.

Ce chapitre consiste à estimer les matrices de facteur $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ dans le cas de la décomposition CP, en résolvant le problème de minimisation suivant :

$$\boxed{\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) = \|\mathcal{T} - \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\|_F^2} \tag{3.1}$$

Dans ce chapitre, nous présentons les différentes méthodes (ou algorithmes) itératives qui ont été utilisés pour optimiser la décomposition tensorielle (3.1) sans contrainte. Nous distinguons deux principaux types d'algorithmes : **les algorithmes alternés** et **les algorithmes de descente**.

Les algorithmes originaux présentés dans cette thèse permettent d'améliorer les résultats fournis par la deuxième catégorie, c'est pourquoi nous lui accorderons plus d'importance ici. Cependant, il est aussi utile de rappeler ici le principe des méthodes de type alterné les plus connues, car nous serons amenés à reprendre plusieurs résultats et procédures algorithmiques établis à cette occasion. Ces deux types de méthodes formulent le problème de décomposition CP comme un problème d'optimisation classique que nous présentons dans la section suivante.

Avant cela, il convient de se donner des critères pour juger de la qualité des différentes méthodes :

- Le taux de convergence de l'algorithme lié à la méthode.
- Le nombre d'itérations nécessaires pour atteindre la convergence.
- La précision d'estimation des matrices facteurs à partir d'un tenseur bruité.
- La complexité numérique de la méthode définie ici comme le nombre de multiplications effectuées à chaque itération.
- La résistance au mauvais conditionnement des matrices facteurs i.e. lorsque les colonnes d'une ou de plusieurs matrices facteurs sont quasi-colinéaires.

Dans la section 3.2, une étude bibliographique sur les problèmes d'optimisation sera faite, ainsi que les différents types des méthodes d'optimisation seront étudiés. Ensuite dans la section 3.3, nous introduisons les méthodes alternées dans lequel nous étudierons plus en détail l'algorithme des moindres carrés alternés ALS (Alternating Least Squares), fréquemment utilisé dans la littérature pour optimiser la décomposition tensorielle CP

[Bro *et al.*, 1997, Sidiropoulos *et al.*, 2000a, Comon *et al.*, 2009a]. Au travers du présent chapitre, on s'attachera à une description plus spécifique des algorithmes itératifs (ou méthodes itératives) qui ont été implémentés et qui permettent la résolution des problèmes d'optimisation. Il convient de souligner que la plupart des algorithmes d'optimisation, contrainte ou non, fonctionnent selon un schéma général consistant, à chaque itération, à se rapprocher du minimum par la résolution d'un sous-problème de minimisation. Dans la section 3.4, nous considérerons les méthodes permettant de résoudre un problème d'optimisation sans contraintes (appelées aussi parfois méthodes d'optimisation directe), que nous décrierons. Parmi lesquelles :

1. Les méthodes basées sur le gradient
2. Les méthodes de Newton
3. Les méthodes utilisant des directions conjuguées.

Ces méthodes utilisent des dérivées, à l'exception des méthodes de directions conjuguées (sauf dans le cas particulier de la méthode du gradient conjugué). Finalement, la section 3.5 est dédiée aux méthodes permettant de calculer le pas exact ou approximatif pour chacune des directions choisie par ces méthodes de descente.

3.2 Étude bibliographique

L'optimisation est une discipline des mathématiques appliquées, qui se retrouve au coeur des méthodes de traitement de données multimodales. Par exemple, en apprentissage statistique, des algorithmes d'optimisation passent sur de grands volumes de données pour calculer les paramètres des modèles utilisés ensuite pour prédire, classifier ou décider.

La résolution de problèmes d'optimisation est devenue un sujet central en recherche opérationnelle, le nombre de problèmes d'aide à la décision pouvant être formalisés sous la forme d'un problème d'optimisation étant en forte croissance.

Les problèmes d'apprentissage de réseaux de neurones, de planification des tâches ou encore d'identification sont, par exemple, des problèmes communs d'optimisation.

En outre, l'Homme cherche à améliorer sa vie quotidienne, l'Homme aime la perfection ! Et sans qu'il se rend compte, il essaye à minimiser ses charges, son loyer ou la consommation de sa voiture... Il tente toujours à vrai dire à optimiser que se soit minimiser ses dépenses où maximiser ses biens.

Le scientifique vient pour concrétiser le voeux de l'Homme en modélisant les problèmes de la vie sous des fonctions coûts en utilisant les différents types d'optimisation.

De nos jours l'optimisation est devenue un domaine indispensable pour résoudre plusieurs problèmes que se soit dans l'industrie ou d'autres secteurs.

En effet nous avons assisté ces dernières années à une croissance très rapide des travaux utilisant les méthodes d'optimisation. Cette tendance peut être observée dans tous les domaines de la science.

Mathématiquement, dans le cas d'une minimisation, un problème d'optimisation se présente sous la forme suivante :

$$\begin{cases} \min_{\mathbf{x}} f(\mathbf{x}) & \\ tel\ que, & \\ g_i(\mathbf{x}) \leq 0 & i = 1, ..., m \\ h_j(\mathbf{x}) = 0 & j = 1, ..., p \\ \mathbf{x} \in \mathcal{S} \subset \mathbb{R}^n & \end{cases} \tag{3.2}$$

où f est la fonction (scalaire) à minimiser, appelée fonction coût ou fonction objectif, $\mathbf{x}$ représente le vecteur des variables d'optimisation, g_i sont les contraintes d'inégalité et h_j les contraintes d'égalité, et $\mathcal{S}$ est l'espace des variables (appelé aussi espace de recherche). $\mathcal{S}$ indique le type de variables considérées : réelles, entières, mixtes (réelles et entières dans un même problème), discrètes, continues, bornées, etc.

Un point x_A est appelé un point admissible si $x_A \in \mathcal{S}$ et si les contraintes d'optimisation sont satisfaites : $g_i(x_A) \leq 0, i = 1, ..., m$ et $h_j(x_A) = 0, j = 1, ..., p$. La solution de 3.2 est l'ensemble des optima $\{x^*\}$. Dans ce mémoire, on s'intéressera plutôt aux problèmes d'optimisation en variables réelles et complexes, avec des contraintes d'égalité et d'inégalité.

x^* est un minimum global de f si et seulement si $f(x^*) \leq f(x)\ \forall x \in \mathcal{S}$, et x^* est un minimum local de f si et seulement si $\forall x \in \mathcal{S} / \|x - x^*\| \leq \epsilon, \epsilon > 0$. La Figure 3.1 présente un exemple d'une fonction à une variable, avec des minima locaux et un minimum global. Parmi les minima locaux, celui qui possède la plus petite valeur de f est le minimum global.

Une fonction multimodale présente plusieurs minima (locaux et globaux), alors qu'une fonction unimodale n'a qu'un minimum, le minimum global. La Figure 3.2 illustre une fonction multimodale à deux variables.

On appelle méthode (ou algorithme ou recherche) locale celle qui converge vers un minimum local. Les recherches locales partent usuellement d'un point initial x_0 avec un pas initial ρ. Ces paramètres vont conditionner la descente d'une des vallées de la fonction.

La méthode d'optimisation est conditionnée par des paramètres de contrôle et des conditions initiales (valeurs initiales des variables de conception et des paramètres de contrôle). Elle peut être caractérisée selon le modèle illustré en figure 3.3.

L'efficacité d'une méthode d'optimisation est liée à la sensibilité et à la robustesse par rapport aux paramètres de contrôle et aux conditions initiales. Lorsque les variables de conception doivent prendre une valeur bien précise pour que la méthode de résolution converge vers l'optimum d'une fonction donnée, la méthode est dite sensible aux conditions initiales. Une méthode d'optimisation est robuste si pour une même valeur des paramètres de contrôle et des conditions initiales, elle est capable de trouver l'optimum de différentes fonctions. Une méthode parfaite devrait être totalement insensible aux conditions initiales et aux variables de conception, et converger vers l'optimum quelles que soient la fonction objectif et les contraintes.

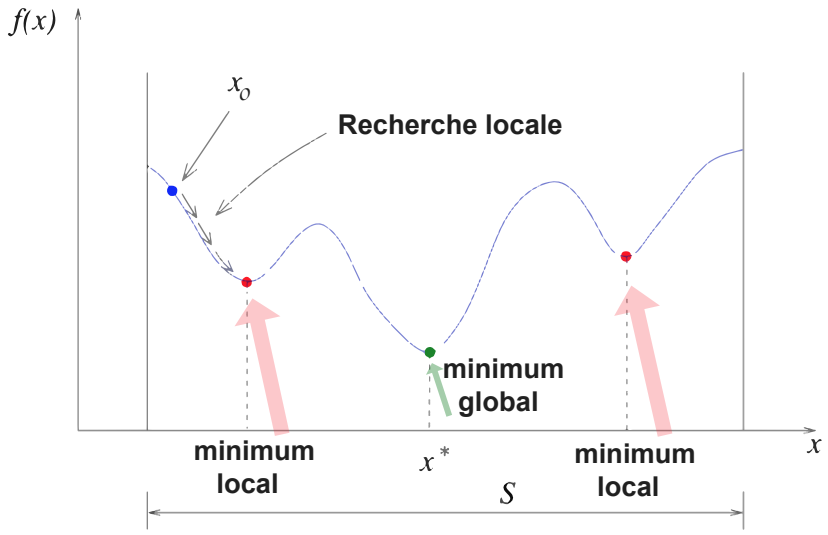


FIGURE 3.1 – Minima locaux et minimum global d'une fonction à une variable.

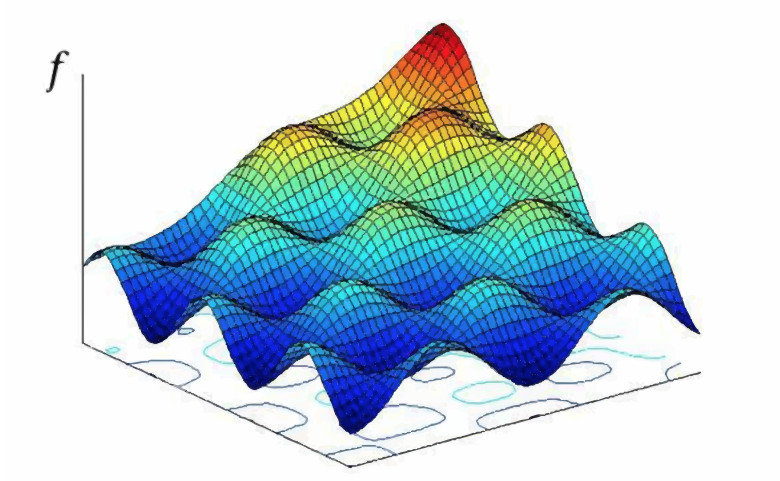


FIGURE 3.2 – Exemple de fonction multimodale à deux variables.

3.2.1 Opérateurs de recherche fondamentaux

En général, la recherche de l'optimum d'une fonction est réalisée à l'aide de deux opérateurs fondamentaux : **l'exploration** et **l'exploitation**. Le premier opérateur, qui est l'exploration, permet une localisation imprécise de l'optimum global, alors que le deuxième opérateur, l'exploitation, affine cette solution en augmentant la précision de l'optimum. Le succès et l'efficacité d'une technique de résolution dépendent la plupart du temps d'un compromis entre l'exploration et l'exploitation. Certaines méthodes toutefois n'utilisent qu'un seul de ces opérateurs pour parvenir à l'optimum. Ainsi, les méthodes déterministes, exploitant les dérivées de la fonction objectif et des contraintes pour atteindre rapidement et précisément le minimum local le plus proche du point de départ,

privilégient l'exploitation au détriment de l'exploration.

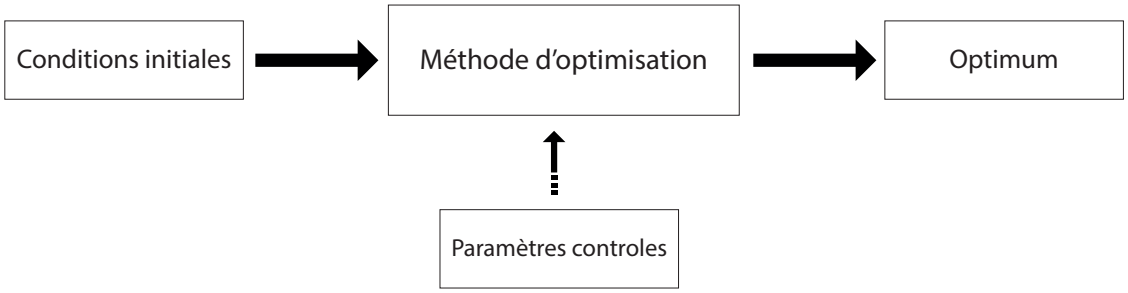


FIGURE 3.3 – Modèle des méthodes d'optimisation.

Tout algorithme d'optimisation doit utiliser ces deux stratégies pour trouver l'optimum global : l'exploration pour la recherche de régions inexplorées de l'espace de recherche, et l'exploitation pour exploiter la connaissance acquise aux points déjà visités et ainsi trouver des points meilleurs. Ces deux exigence peuvent paraître contradictoires, mais un bon algorithme de recherche doit trouver le bon compromis entre les deux. Une recherche purement aléatoire est bonne pour l'exploration, mais pas pour l'exploitation, alors que la recherche dans le voisinage est une bonne méthode d'exploitation mais pas d'exploration.

3.2.2 Ordre d'une méthode d'optimisation

Les méthodes d'optimisation peuvent être classées à partir de leur ordre selon leurs nécessitées ou non du calcul des dérivées de la fonction objectif et des contraintes par rapport aux paramètres.

Une méthode est dite d'ordre zéro si elle utilise uniquement la connaissance de la fonction elle-même. Elle est d'ordre un si elle requiert le calcul des dérivées premières et d'ordre deux s'il lui faut aussi accéder aux dérivées secondes. Les méthodes d'ordre zéro sont en général peu précises et convergent plus lentement vers l'optimum. En revanche, elles offrent l'avantage d'éviter le calcul du gradient, ce qui est intéressant lorsque la fonction n'est pas différentiable, ou si le calcul de son gradient représente un coût important. C'est notamment le cas des modèles à éléments finis.

Les méthodes d'ordre un permettent d'accélérer la localisation de l'optimum, puisque le gradient donne l'information sur la direction de l'amélioration (direction de recherche). Par contre, elles sont applicables seulement aux problèmes où la fonction objectif et les contraintes sont différentiables.

3.2.3 Classifications des méthodes d'optimisation

Les méthodes d'optimisations sont classées, selon le mode de recherche de l'optimum, en deux grands groupes : *les méthodes déterministes* et *les méthodes non déterministes*.

3.2.3.1 Les méthodes déterministes

Les méthodes déterministes sont généralement efficaces quand l'évaluation de la fonction est très rapide, où quand la forme de la fonction est connue a priori. Les cas les plus complexes (temps de calcul important, nombreux optima locaux, fonctions non-dérivables, fonctions fractales, fonctions bruitées) seront souvent traités plus efficacement par des méthodes non-déterministes.

Ces méthodes peuvent être subdivisées en plusieurs sous classes : les méthodes heuristiques, les méthodes statistiques, les méthodes Branch et Bound, les méthodes mathématiques, et les méthodes d'apprentissage automatique. Cette classification est illustrée en table 3.1. La classe qui nous intéresse dans ce manuscrit est celle des méthodes mathématiques.

Classe	Algorithme	Ordre	Globale où Locale
Méthodes heuristiques	Simplex	0	Locale
Plans d'expériences		0	Globale
Méthode Branch et Bound		0	Globale
Méthode d'apprentissage automatique	Réseaux de neurones	0	Locale
Méthodes mathématiques	Direction conjuguée	0	Locale
	Plus grande pente	≥ 1	Locale
	Gradient conjuguée	≥ 1	Locale
	Quasi Newton	≥ 1	Locale
	Proximal gradient	≥ 1	Locale
	Forward Backward	≥ 1	Locale
	Proximal gradient accéléré	≥ 1	Locale

TABLE 3.1 – Méthodes d'optimisation déterministes.

La recherche des extremas d'une fonction f à n variables $(x_1, x_2, ..., x_n)$ revient à résoudre un système de n équations à n inconnus, linéaires ou non linéaires :

$$\frac{\partial f}{\partial x_i}(x_1, x_2, ..., x_n) = 0 \tag{3.3}$$

tel que i=1,...,n. En général, les techniques classiques de résolution sont des méthodes simples à comprendre et faciles à mettre en oeuvre, mais leur utilisation nécessite comme étape préliminaire la localisation des extremas. La fonction objectif dans ce cas doit être continue et dérivable. Cette exigence n'est pas toujours vérifiée dans le cas de problèmes réels. La nature dynamique et discontinue des problèmes rend l'utilisation des méthodes classiques limitée. Parmi les méthodes classiques, on peut citer la méthode du gradient, les méthodes Quasi- Newton, la méthode du gradient conjugué...etc, que nous détaillerons dans la suite de ce chapitre.

3.2.3.2 Les méthodes non déterministes

Les méthodes d'optimisation non déterministes, appelées aussi méthodes stochastiques, s'appuient sur des mécanismes de transition probabilistes et aléatoires. Cette caractéristique indique que plusieurs exécutions successives de ces méthodes peuvent conduire à des résultats différents, pour une même configuration initiale d'un problème d'optimisation. Ces méthodes ont une grande capacité à trouver l'optimum global du problème. Contrairement à la plupart des méthodes déterministes, elles ne nécessitent ni point de départ, ni la connaissance du gradient de la fonction objectif pour atteindre la solution optimale. Elles sont d'ordre zéro. Cependant, elles demandent un nombre important d'évaluations de la fonction objectif. La figure 3.4 présente les méthodes stochastiques les plus utilisées.

Dans les problèmes d'optimisation, on cherche à minimiser une fonction qui peut être complexe, coûteuse à estimer, et dont les dérivées ne sont pas toujours disponibles. A ces difficultés s'ajoutent la prise en compte des contraintes non linéaires, et parfois l'aspect multi-objectifs, d'où l'existence de deux types de méthode ou algorithme d'optimisation :

— Algorithmes d'optimisation sans contraintes
— Algorithmes d'optimisation avec contraintes

Dans ce manuscrit, nous nous intéresserons aux deux types d'algorithmes pour optimiser le calcul de la décomposition tensorielle CP.

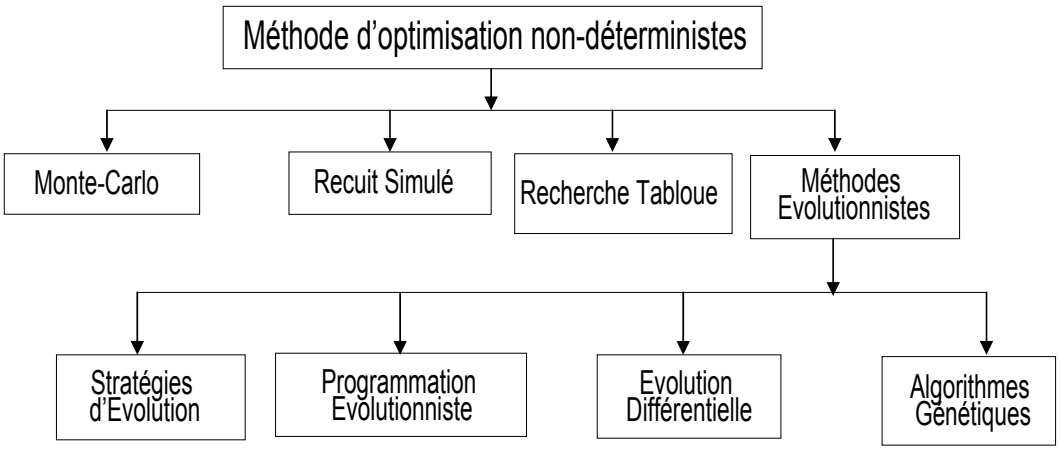


FIGURE 3.4 – Les méthodes d'optimisation non-déterministes.

3.3 Les méthodes alternées

Ces méthodes, comme leur nom l'indique, estiment les matrices facteurs de manière alternée à chaque itération. Le principe est de considérer, à une itération donnée, toutes

les matrices facteurs comme connues excepté une que l'on va estimer en fonction de toutes les autres. La structure de ces algorithmes est résumée dans la table algorithmique 3.1.

Algorithme 3.1 Forme générique des algorithmes de type alterné

Soit S_C un critère d'arrêt préalablement défini et It_{max} le nombre d'itérations maximal ;
Initialisation des matrices facteurs $\hat{\mathbf{A}}^{(2)}$ à $\hat{\mathbf{A}}^{(N)}$
$it \leftarrow 1$
while S_C est faux et it $< It_{max}$ **do**
for n = 1 à N **do**
Estimation des paramètres de la matrice facteurs $\hat{\mathbf{A}}^{(n)}_{(it+1)}$ en fonction des autres matrices facteurs précédemment estimées ;
end for
Mettre à jour S_C ;
$it \leftarrow it + 1$;
end while

3.3.1 Algorithme des moindres carrés alternés (ALS)

3.3.1.1 Principe

Le principe général de cet algorithme est assez simple. Soit $\mathcal{T}$ un tenseur d'ordre N suivant une décomposition donnée, dont les N matrices de facteurs inconnues sont $\mathbf{A}^{(1)}$, . . . , $\mathbf{A}^{(N)}$. Notons $\mathbf{T}^{(n)}$, $n = 1, \ldots, N$, les représentations matricielles du tenseur $\mathcal{T}$. Pour chaque $\mathbf{T}^{(n)}$, il existe une matrice $\mathbf{Z}^{(n)}$, construite à partir de $n-1$ matrices de l'ensemble $S = \{\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}\} - \{\mathbf{A}^{(n)}\}$, qui vérifie :

$$\mathbf{T}^{(n)} = \mathbf{Z}^{(n)}\mathbf{A}^{(n)} \tag{3.4}$$

Étant donné que seul le tenseur $\mathcal{T}$ est connu, l'algorithme ALS consiste à exploiter les N équations du type (3.4) pour estimer les matrices $\mathbf{A}^{(n)}$ d'une manière alternée. Ainsi, une fois l'estimation de toutes les matrices de l'ensemble S faite, l'estimation de $\mathbf{A}^{(n)}$ au sens des moindres carrés est obtenue par :

$$\hat{\mathbf{A}}^{(n)} = (\mathbf{Z}^{(n)})^{\dagger}\mathbf{T}^{(n)} \tag{3.5}$$

Dans le cas d'un tenseur d'ordre 3, l'algorithme ALS [Bro *et al.*, 1997, Sidiropoulos *et al.*, 2000a, Smilde *et al.*, 2005] est la solution classique pour minimiser cette fonction de coût (3.1). Il s'agit d'un algorithme itératif qui alterne entre l'estimation de $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$.

Le principe de l'algorithme **ALS** est de mettre à jour de manière alternée chacune des matrices en gardant les deux autres fixées. Si deux matrices parmi les trois sont

fixées, le système à résoudre devient alors un problème simple des moindres carrés (LS).

L'ALS est fait en trois étapes principales :

Étape 1 : On fixe les deux matrices $\mathbf{B}$ et $\mathbf{C}$ pour trouver $\mathbf{A}$ qui vérifie :

$$\min_{\mathbf{A}} \|\mathbf{T}_{(1)}^{I\times KJ} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T\|_F^2$$

Ce qui revient que la solution de $\mathbf{A}$ s'écrit :

$$\hat{\mathbf{A}} = \mathbf{T}_{(1)}^{I\times KJ}((\mathbf{C}\odot\mathbf{B})^T)^\dagger$$

Et parce que le pseudo-inverse du produit Khatri-Rao a cette forme spéciale

$$(\mathbf{C}\odot\mathbf{B})^\dagger = ((\mathbf{C}^T\mathbf{C})*(\mathbf{B}^T\mathbf{B}))^\dagger(\mathbf{C}\odot\mathbf{B})^T \tag{3.6}$$

Il est commune de réécrire la solution de $\mathbf{A}$ par

$$\hat{\mathbf{A}} = \mathbf{T}_{(1)}^{I\times KJ}(\mathbf{C}\odot\mathbf{B})((\mathbf{C}^T\mathbf{C})*(\mathbf{B}^T\mathbf{B}))^\dagger$$

♢L'avantage de cette version de l'équation, c'est que nous n'avons besoin que de calculer la pseudo-inverse d'une matrice de taille $R\times R$ plutôt qu'une matrice $JK\times R$.

Pour les deux autres étapes, on va utiliser directement la forme (3.6).

Étape 2 : Cette fois on va fixer $\mathbf{A}$ et $\mathbf{C}$ pour trouver $\mathbf{B}$ vérifiant :

$$\min_{\mathbf{B}} \|\mathbf{T}_{(2)}^{J\times KI} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T\|_F^2$$

Donc la solution de $\mathbf{B}$ s'écrit en utilisant la forme (3.6) par :

$$\hat{\mathbf{B}} = \mathbf{T}_{(2)}^{J\times KI}(\mathbf{C}\odot\mathbf{A})((\mathbf{C}^T\mathbf{C})*(\mathbf{A}^T\mathbf{A}))^\dagger$$

Étape 3 : Finalement, on fixe $\mathbf{A}$ et $\mathbf{B}$ pour trouver $\mathbf{C}$ qui vérifie :

$$\min_{\mathbf{C}} \|\mathbf{T}_{(3)}^{K\times JI} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T\|_F^2$$

Ce qui revient que la solution de $\mathbf{C}$ s'écrit :

$$\hat{\mathbf{C}} = \mathbf{T}_{(3)}^{K\times JI}(\mathbf{B}\odot\mathbf{A})((\mathbf{B}^T\mathbf{B})*(\mathbf{A}^T\mathbf{A}))^\dagger$$

L'algorithme ALS utilise les factorisations *Khatri-Rao* des matrices dépliées $\mathbf{T}^{I,KJ}$, $\mathbf{T}^{J,KI}$ et $\mathbf{T}^{K,JI}$ données dans la section 1.4.2.3, et représentées dans la figure 1.4.

Chaque représentation permet d'estimer l'une des trois matrices au sens des moindres carrées. Ainsi, chaque mise à jour des matrices est obtenue par une simple inversion matricielle. Un synopsis de l'algorithme ALS est donné dans l'algorithme 3.2.

Algorithme 3.2 CP-ALS

Entrées: $\mathcal{T}$ et le nombre de composants R

Sorties: $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ de taille respective $(I \times R)$, $(J \times R)$ et $(K \times R)$

$M = 1000$ (le nombre maximal d'itérations)

$\epsilon \leftarrow 10^{-6}$ (la précision de l'algorithme)

$k \leftarrow 0$

Répéter

$\mathbf{A} = \mathbf{T}_{(1)}^{I \times KJ}(\mathbf{C} \odot \mathbf{B})((\mathbf{C}^T\mathbf{C}) * (\mathbf{B}^T\mathbf{B}))^{\dagger}$

$\mathbf{B} = \mathbf{T}_{(2)}^{J \times KI}(\mathbf{C} \odot \mathbf{A})((\mathbf{C}^T\mathbf{C}) * (\mathbf{A}^T\mathbf{A}))^{\dagger}$

$\mathbf{C} = \mathbf{T}_{(3)}^{K \times JI}(\mathbf{B} \odot \mathbf{A})((\mathbf{B}^T\mathbf{B}) * (\mathbf{A}^T\mathbf{A}))^{\dagger}$

Jusqu'à ($\|\mathcal{T} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2 < \epsilon$ ou k>M)

Retourner $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$

La convergence à l'itération k est déclarée lorsque l'erreur entre le tenseur d'origine et sa version reconstruite, à partir des matrices de facteurs estimées, ne change pas de manière significative entre les itérations k et $k + 1$. L'erreur de reconstruction à l'itération $k^{\text{ème}}$ peut être calculée à partir de la formule suivante :

$$E(k) = \|\mathbf{T}_{(1)}^{I,KJ} - (\mathbf{C}^{(k)} \odot \mathbf{B}^{(k)})(\mathbf{A}^{(k)})^T\|_F \tag{3.7}$$

On peut dire que l'algorithme converge à l'itération k lorsque $\|E(k+1) - E(k)\| < \epsilon$ tel que ϵ et un seuil dont sa valeur est fixée d'avance. La mise à jour conditionnelle d'une matrice donnée peut être soit améliorée ou maintenue, mais ne peut pas aggraver la forme actuelle. L'algorithme converge toujours de manière monotone, au moins, vers un minimum local. Cependant, l'algorithme ALS est fortement dépendant de l'initialisation, et sa convergence vers le minimum global peut parfois être lente. De plus, la convergence de l'algorithme peut, dans certains cas, tomber dans les régions de marais (swamps), au cours desquelles la vitesse de convergence est très faible et l'erreur entre deux itérations consécutives ne diminue pas. Dans ce cas, pour éviter un arrêt anticipé de l'algorithme, une pratique courante est d'imposer une valeur minimale acceptable de l'erreur $E(k)$, au-dessus de laquelle la convergence globale n'est pas supposée encore atteinte.

Plusieurs variantes de l'algorithme ALS ont été proposées dans la littérature. Afin d'atténuer les problèmes de convergence lente causés par une initialisation aléatoire de l'algorithme, une solution d'analyse propre peut être utilisée [Sanchez et Kowalski, 1990, Sidiropoulos *et al.*, 2000b, Leurgans *et al.*, 1993]. Cette solution est également connue comme la décomposition trilinéaire directe [Sanchez et Kowalski, 1990]. Elle consiste à obtenir une première estimation des matrices de facteurs de la décomposition, par la construction d'un problème de valeur propre généralisée (ou un problème de diagonalisation conjointe) à partir de deux tranches de tenseur. L'initialisation à base de l'analyse propre, en plus de se limiter à des tenseurs avec seulement deux tranches dans l'un des modes, exige que les deux matrices de facteurs sont de rang colonne plein,

et que la troisième ne contient pas des éléments zéro. Dans [De Lathauwer, 2006], une généralisation de la solution d'analyse propre pour des tenseurs avec plus de deux tranches a été proposée, en liant l'estimation des matrices de facteurs de la décomposition CP au problème de diagonalisation matricielle simultané. Un autre moyen d'améliorer la vitesse de l'algorithme ALS est basé sur la méthode de compression TUCKER3 [Andersson et Bro, 1998, Bro et Andersson, 1998]. Cette méthode est utile lorsque les dimensions du tenseur sont grandes. Dans [Sidiropoulos *et al.*, 2000b], un algorithme qui accélère la convergence de ALS est proposée. Cet algorithme applique la méthode de compression TUCKER3 suivie d'une initialisation basée sur l'analyse propre.

3.3.2 Algorithme de la ligne de recherche améliorée (ELS)

La convergence de l'algorithme ALS peut également être améliorée par le biais de ce qu'on appelle la méthode de la ligne de recherche améliorée (ELS) [Rajih et Comon, 2005]. Cette méthode a montré son utilité lorsque la décomposition du tenseur est affectée par des facteurs dégénérescences (Factor degeneracies). La méthode ELS a également été utilisée pour estimer les facteurs des décompositions tensorielles en blocs [Rajih *et al.*, 2008]. L'idée de la méthode ELS consiste à rechercher le facteur de relaxation optimal R_{LS}, qui conduit à la solution finale d'un cycle donné en une seule étape. Par l'itération (k), on définit $\mathbf{L}_A^{(k)} = A^{(k-1)} - A^{(k-2)}$ comme étant la direction d'estimation de la matrice A. De même, nous définissons les deux directions $\mathbf{L}_B^{(k)}$ et $\mathbf{L}_C^{(k)}$. Au lieu de fixer une seule valeur de R_{LS} pour les trois modes comme utilisé dans la méthode LS (Line search), nous pouvons chercher le triplet optimal (R_A, R_B, R_C) qui minimise :

$$\Upsilon_{ELS} = \|\mathbf{T}_{(1)}^{I,KJ} - (\mathbf{A}^{(k-2)} + R_A\mathbf{L}_A^{(k)})((\mathbf{B}^{(k-2)} + R_B\mathbf{L}_B^{(k)}) \odot (\mathbf{C}^{(k-2)} + R_C\mathbf{L}_C^{(k)}))^T\|_F^2 \quad (3.8)$$

la méthode ELS est exécutée au début de l'algorithme ALS comme représentée dans la figure 3.5, où l'étape 1 correspond à la partie ELS. Les facteurs de relaxation appliqués aux matrices de facteurs sont calculées à l'étape 1 de la figure 3.5, en tant que des valeurs optimales qui fournissent la plus petite erreur Υ_{ELS}. À l'étape 3, et après estimation des matrices de facteurs de l'itération (k), nous mettons à jour celles de l'itération $(k-1)$ pour avoir les nouvelles matrices $\mathbf{A}^{(new)}$, $\mathbf{B}^{(new)}$ et $\mathbf{C}^{(new)}$. Les matrices de facteurs des deux itérations $(k-1)$ et (k) sont ensuite utilisées dans l'itération suivante si l'algorithme n'atteint pas la convergence.

3.3.3 Simulations

Dans cette section, nous évaluerons les performances de l'algorithme *ALS* sous différents scénarios. On notera par $ALS + ELS$ l'algorithme *ALS* avec recherche de ligne améliorée décrit dans la section 3.3.2 et représenté dans la figure 3.5. Chaque résultat obtenu est une moyenne de 100 réalisations indépendantes de Monte-Carlo. La tolérance est fixée à $\epsilon_{ALS} = 10^{-10}$. Les éléments des matrices de facteurs $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ suivent une loi gaussienne de moyenne nulle et de variance un.

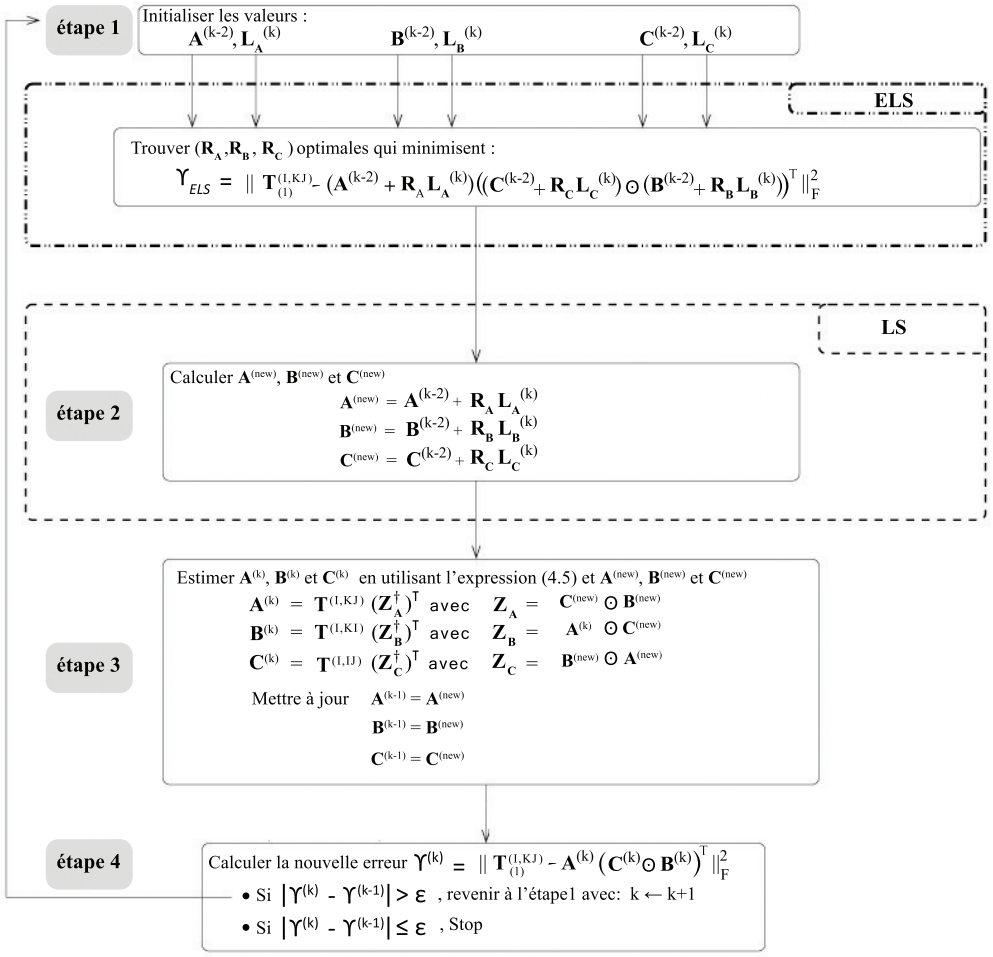


FIGURE 3.5 – Le résumé de l'algorithme ELS avec ELS.

Dans le premier scénario, nous considérons un tenseur d'ordre 3 et de taille $I = 6$, $J = 5$, $K = 4$ et de rang $R = 5$. Nous évaluons l'erreur de la reconstruction du tenseur $\mathcal{T}$ par rapport au nombre d'itérations demandés en utilisant la formule suivante : $\frac{\|\mathcal{T} - \hat{\mathcal{T}}\|^2}{\|\mathcal{T}\|^2}$, où $\mathcal{T}$ est le tenseur original et $\hat{\mathcal{T}}$ est le tenseur reconstruit à partir des matrices de facteurs estimées. La figure 3.6 représente la convergence de l'erreur de reconstruction du tenseur par rapport au nombre d'itérations pour l'algorithme ALS avec/sans la méthode ELS. Nous remarquons sur la figure 3.6 que la méthode ELS réduit le nombre d'itérations nécessaires pour atteindre le critère de convergence. Ainsi, le nombre d'itérations diminue de 1000 itérations à environ 150. De plus, même si le ALS atteint une valeur d'erreur de 10^{-10}, cela nécessite plus d'itérations et donc plus de temps. En revanche, le $ALS + ELS$ converge vers des valeurs d'erreur inférieures à 10^{-30} pour 500 itérations.

Dans la figure 3.7, nous avons tracé l'erreur d'estimation d'une matrice de facteurs, telle que l'erreur estimée est égale au nombre d'éléments de $\mathbf{A} - \hat{\mathbf{A}}$ différents de 0, divisé par le nombre total des éléments de $\mathbf{A}$ (c'est-à-dire $I \times R$), où $\hat{\mathbf{A}}$ désigne l'estimée de la matrice $\mathbf{A}$ après décision. La figure 3.7 montre que l'ELS est très utile pour réduire le

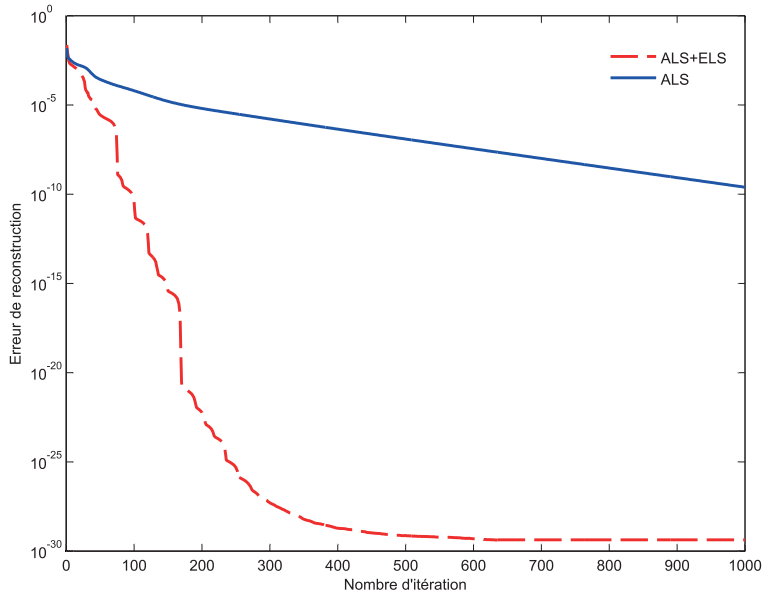


FIGURE 3.6 – Performance de l'algorithme ALS et $ALS + ELS$ en fonction de nombre d'itérations, pour un tenseur de rang 5 et de dimensions $6 \times 5 \times 4$.

nombre d'itérations nécessaires afin d'estimer la matrice de facteurs $\mathbf{A}$. En utilisant l'ELS, le nombre d'itérations diminue de 1000 à 500 pour atteindre une erreur d'estimation de 10^{-10}.

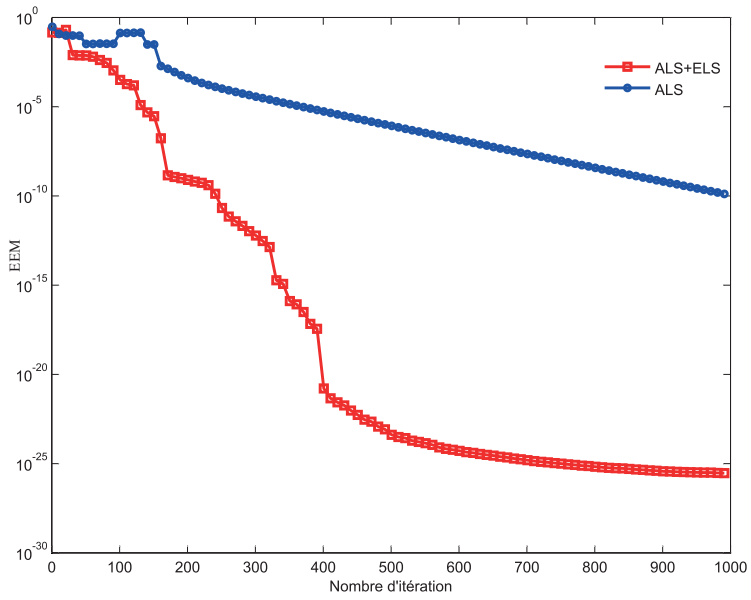


FIGURE 3.7 – Erreur d'estimation de la matrice de facteurs en utilisant les algorithmes ALS et $ALS + ELS$.

Pour le deuxième scénario, nous considérons un tenseur d'ordre 3, de taille $I = J = K = 10$ et de rang $R = 4$. Dans ce scénario, nous comparons la convergence de l'algorithme

ALS avec/sans ELS par rapport à la valeur du rapport signal sur bruit SNR (Signal Noise Ratio). Sur la figure 3.8, nous avons tracé la convergence de l'erreur de reconstruction du tenseur $\mathcal{T}$ par rapport au SNR. À partir de cette figure, nous remarquons bien que l'allure des deux courbes est presque similaire surtout pour des faibles valeurs de SNR. Une petite différence entre les deux courbes (ALS, $ALS + ELS$), est observée lorsque les valeurs de SNR deviennent de plus en plus grandes.

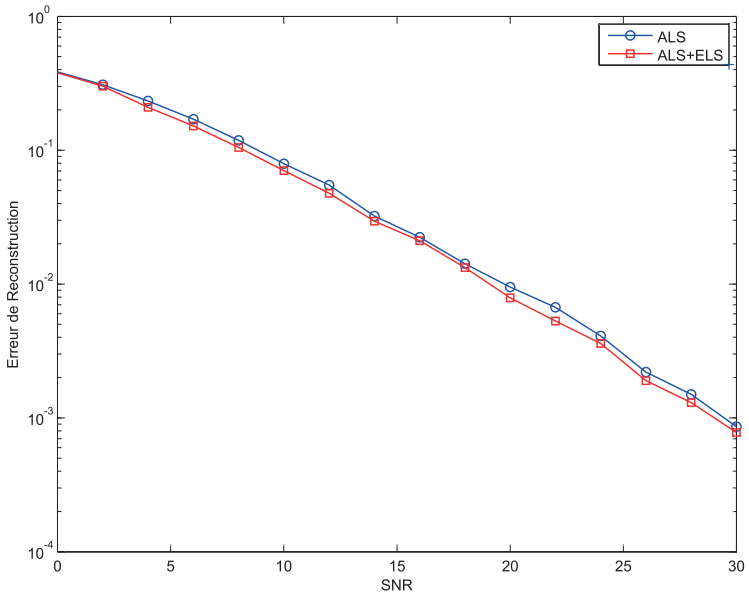


FIGURE 3.8 – Performance de l'algorithme ALS et $ALS + ELS$ en fonction de SNR, pour un tenseur de rang 4 et de dimensions $10 \times 10 \times 10$.

3.4 Les méthodes de descente

Dans cette section, on va introduire une classe importante d'algorithmes de résolution des problèmes d'optimisation sans contraintes : les algorithmes à direction de descente (appelées aussi algorithmes d'optimisation directe). En comparaison avec les algorithmes alternés, toutes les variables sont ici estimées simultanément à chaque itération. De plus ces algorithmes ont obligatoirement besoin de l'information du gradient des fonctions coût pour chercher l'optimum du problème. leur utilisation nécessite que la fonction coût soit au moins une fois différentiable par rapport aux paramètres à optimiser.

Une direction de descente est une direction le long de laquelle la fonction à minimiser à une dérivée directionnelle strictement négative. c'est le long de ces directions qu'un déplacement est effectué afin de trouver l'itéré suivant, en lequel la fonction à minimiser prend une valeur inférieure à celle qu'elle a en l'itéré courant. Des directions de descente peuvent être calculées par de nombreuses méthodes permettant de calculer une direction

de descente. Les principes sont présentées par la suite ; chacune avec ses propres caractéristiques. Un algorithmes qui utilise une telle direction hérite d'elle son nom. Ainsi l'algorithme du gradient est l'algorithme à directions de descente qui utilise la direction du gradient ; l'algorithme du gradient conjugué utilise la direction du gradient conjugué, etc.

On considère le problème d'optimisation sans contrainte :

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}).$$

On note : $\mathbf{G}{=}\nabla f$ le gradient et $\mathbf{H} = \nabla^2 f$ la hessienne de la fonction f au point courant $\mathbf{x}$.

Partant d'un point $x^{(0)}$ arbitrairement choisi, un algorithme de descente cherche à générer une suite d'itérés $(x^{(k)})_{k\in\mathbb{N}}$ telle que :

$$\forall k \in \mathbb{N}, f^{(k+1)} \leq f^{(k)}. \tag{3.9}$$

Dans le cas de notre fonction objectif 3.1, l'équation 3.9 deviendra comme suit :

$$\forall k \in \mathbb{N}, \Upsilon(\mathbf{A}^{(k+1)}, \mathbf{B}^{(k+1)}, \mathbf{C}^{(k+1)}) \leq \Upsilon(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)}). \tag{3.10}$$

Commençons par définir plus précisément la notion de descente. Le gradient joue un rôle essentiel en optimisation. Dans le cadre des méthodes d'optimisation, il sera également important d'analyser le comportement de la fonction objectif dans certaines directions.

Définition Soient $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ et $x \in \mathbb{R}^n$ Le vecteur $d \in \mathbb{R}^n$ est une direction de descente pour f à partir du point x si $t \longmapsto f(x^{(k)} + \rho d^{(k)})$ est décroissante en $\rho = 0$, c'est-à-dire s'il existe $\eta > 0$ tel que :

$$\forall \rho \in]0, \eta], f(x^{(k)} + \rho d^{(k)}) \leq f(x^{(k)}). \tag{3.11}$$

Proposition Soient $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ différentiable et $x^{(k)} \in \mathbb{R}^n$ tel que : $\nabla f(x^{(k)}) \neq 0$. Le vecteur $d^{(k)} \in \mathbb{R}^n$ est une direction de descente pour f à partir du point $x^{(k)}$ si et seulement si la dérivée directionnelle de f en $x^{(k)}$ dans la direction $d^{(k)}$ vérifie :

$$df(x^{(k)}; d^{(k)}) = \nabla f(x^{(k)})^T d^{(k)} < 0. \tag{3.12}$$

De plus pour tout $\alpha < 1$, il existe $\tilde{\eta} > 0$ tel que :

$$\forall \rho \in]0, \tilde{\eta}], f(x^{(k)} + \rho d^{(k)}) \leq f(x^{(k)}) + \rho\alpha\nabla f(x^{(k)})^T d^{(k)}. \tag{3.13}$$

Autrement dit, la relation 3.13 affirme que la décroissance de la fonction objectif f en faisant un pas de taille ρ dans la direction $d^{(k)}$ est égale au moins au pas multiplié par une fraction β de la pente.

Parmi toutes les directions de descente existantes en un point $x^{(k)}$ donné, une des plus remarquables est celle où la pente est la plus forte, c'est-à-dire que $d^{(k)}$ fait avec $-\nabla f(x^{(k)})$ un angle $\theta^{(k)}$ strictement plus petit que 90 degrés, comme l'illustre la figure 3.9. Pour le démontrer, il suffit de comparer les dérivées directionnelles :

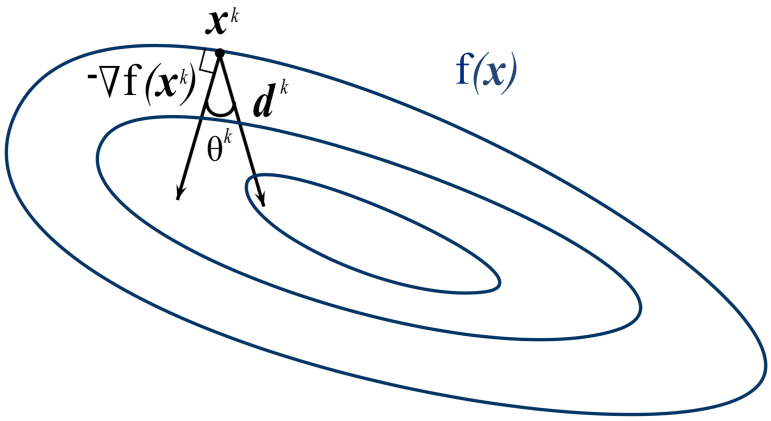


FIGURE 3.9 – La direction $d^{(k)}$ forme un angle $\theta^{(k)} < 90$ avec la direction de plus grande pente $-\nabla f(x^{(k)})$.

Théorème (Direction de plus forte descente). Soient $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction différentiable. Soit $x^{(k)} \in \mathbb{R}^n$. Alors pour toute direction $d^{(k)}$ de norme constante égale à $\|d^{(k)}\| = \|\nabla f(x^{(k)})\|$, on a :

$$(-\nabla f(x^{(k)}))^T \nabla f(x^{(k)}) = (d^{(k)})^T \nabla f(x^{(k)}). \tag{3.14}$$

La direction $(d^*)^{(k)} = \nabla f(x^{(k)})$ est appelée direction de plus forte descente.

L'algorithme de descente utilise cette propriété pour minimiser la fonction f. La suite $x^{(k)}{}_{k\geq 0}$ est construite par la récurrence :

$$x^{(k+1)} = x^{(k)} + \rho^{(k)}(d^{(k)}). \tag{3.15}$$

où $\rho^{(k)} > 0$ est le pas et $d^{(k)}$ est une direction de descente (figure 3.10). Une telle méthode consiste à alterner la construction de $d^{(k)}$ et la détermination de $\rho^{(k)}$. L'arrêt de la récurrence est contrôlé par un test portant généralement sur les faibles valeurs du gradient $\nabla f(x^{(k)})$. C'est en effet ce que suggère la condition d'optimalité $\nabla f(x^{(*)}) = 0$. Bien que l'annulation du gradient puisse intervenir en un maximum ou en un point selle de f, le choix de $(\rho^{(k)}, d^{(k)})$ va empêcher l'algorithme de converger vers un tel point.

3.4.1 Principe général

Le cadre est le suivant. On cherche un point $\mathbf{X}^{(*)}$ qui minimise la fonction de coût Υ. Les algorithmes à directions de descente cherchent un minimum de Υ en générant une suite de points $\{\mathbf{X}^{(k)}\}_{k\geqslant 1}$, appelés itérés, tel que : $\Upsilon(\mathbf{X}^{(k+1)}) < \Upsilon(\mathbf{X}^{(k)})$ qui approchent de mieux en mieux un minimiseur $\mathbf{X}^{(*)}$ du critère Υ. Cette suite est construite en se fondant sur deux constructions :

1. Calcul d'une direction de descente $\mathbf{D}^{(k)}$; la manière de procéder donne le nom à l'algorithme.

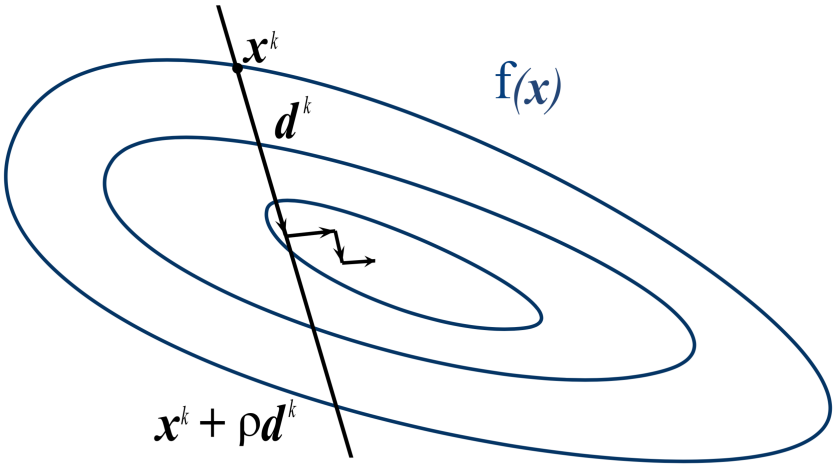


FIGURE 3.10 – La méthode de descente itérative alterne entre la détermination d'une direction de descente $d^{(k)}$ et la minimisation approchée de f le long de $d^{(k)}$ par une stratégie de recherche linéaire.

2. Détermination d'un pas $\rho^{(k)}$, qui est un nombre réel strictement positif, le long de la direction de descente de telle sorte que le nouvel itéré donne au critère une valeur inférieure à celle qu'il a en l'itéré courant ; le nouvel itéré est de la forme suivante :
$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \rho^{(k)}\mathbf{D}^{(k)},$$
cette opération de détermination du pas s'appelle la recherche linéaire.

Revenons à notre fonction objectif sous sa forme matricielle (équation 2.22), pour pouvoir utiliser d'autre types d'algorithmes d'optimisation que le ALS, la différentielle $d\Upsilon$ de la fonction Υ doit être calculée. De cette façon, les matrices $\nabla_A\Upsilon$, $\nabla_B\Upsilon$ et $\nabla_C\Upsilon$ de taille $I \times R$, $J \times R$ et $K \times R$ respectivement, peuvent être évaluées. Dans les calculs suivants, nous considérons le cas où $\Lambda = \mathbf{I}$ est une matrice identité de taille $R \times R$, comme été abondamment traité dans la littérature [Royer *et al.*, 2011], et $\mathbf{A} \in \mathbb{R}^{I\times R}$, $\mathbf{B} \in \mathbb{R}^{J\times R}$ et $\mathbf{C} \in \mathbb{R}^{K\times R}$. on a :

Le point à estimer est une matrice de taille (I+J+K)×R, définie par :

$$\mathbf{X}^{(k)} = \begin{pmatrix} \mathbf{A}^{(k)} \\ \mathbf{B}^{(k)} \\ \mathbf{C}^{(k)} \end{pmatrix}$$

Où un vecteur de taille (I+J+K)R×1 :

$$\mathbf{x}^{(k)} = \begin{pmatrix} vec(\mathbf{A}^{(k)}) \\ vec(\mathbf{B}^{(k)}) \\ vec(\mathbf{C}^{(k)}) \end{pmatrix}$$

Le calcul de gradient par rapport aux matrices à estimer est présenté dans le paragraphe 2.3.3.4, dont lequel on a :

$$\begin{aligned}\nabla_{\mathbf{A}}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{\partial\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{A}}\\ &= \frac{\partial}{\partial\mathbf{A}}(\|\mathbf{T}_{(1)}^{I\times KJ} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T\|_F^2)\\ &= 2[-\mathbf{T}_{(1)}^{I\times KJ} + \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T](\mathbf{C}\odot\mathbf{B})\\ &= 2(-\mathbf{T}_{(1)}^{I\times KJ}(\mathbf{C}\odot\mathbf{B}) + \mathbf{A}(\mathbf{C}^T\mathbf{C})\boxdot(\mathbf{B}^T\mathbf{B})),\end{aligned}$$

$$\begin{aligned}\nabla_{\mathbf{B}}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{\partial\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{B}}\\ &\frac{\partial}{\partial\mathbf{B}}(\|\mathbf{T}_{(2)}^{J\times KI} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T\|_F^2)\\ &= 2[-\mathbf{T}_{(2)}^{J\times KI} + \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T](\mathbf{C}\odot\mathbf{A})\\ &= 2(-\mathbf{T}_{(2)}^{J\times KI}(\mathbf{C}\odot\mathbf{A}) + \mathbf{B}(\mathbf{C}^T\mathbf{C})\boxdot(\mathbf{A}^T\mathbf{A})),\end{aligned}$$

$$\begin{aligned}\nabla_{\mathbf{C}}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{\partial\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{C}}\\ &\frac{\partial}{\partial\mathbf{C}}(\|\mathbf{T}_{(3)}^{K\times JI} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T\|_F^2)\\ &= 2[-\mathbf{T}_{(3)}^{K\times JI} + \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T](\mathbf{B}\odot\mathbf{A})\\ &= 2(-\mathbf{T}_{(3)}^{K\times JI}(\mathbf{B}\odot\mathbf{A}) + \mathbf{C}(\mathbf{B}^T\mathbf{B})\boxdot(\mathbf{A}^T\mathbf{A})),\end{aligned}$$

Donc le gradient dans ce cas est une matrice de taille (I+J+K)×R s'écrit :

$$\mathbf{G}^{(k)} = \begin{pmatrix}\nabla_{\mathbf{A}}F(\mathbf{A},\mathbf{B},\mathbf{C})\\ \nabla_{\mathbf{B}}F(\mathbf{A},\mathbf{B},\mathbf{C})\\ \nabla_{\mathbf{C}}F(\mathbf{A},\mathbf{B},\mathbf{C})\end{pmatrix}$$

Où sous forme d'un vecteur de taille (I+J+K)R×1 :

$$\mathbf{g}^{(k)} = \begin{pmatrix}vec(\nabla_{\mathbf{A}}F(\mathbf{A},\mathbf{B},\mathbf{C}))\\ vec(\nabla_{\mathbf{B}}F(\mathbf{A},\mathbf{B},\mathbf{C}))\\ vec(\nabla_{\mathbf{C}}F(\mathbf{A},\mathbf{B},\mathbf{C}))\end{pmatrix}$$

3.4.2 Algorithme de gradient

La méthode du gradient fait partie des classes de méthodes dites de descente. Considérons un point de départ $x^{(0)}$, et cherchons à minimiser la fonction de coût Υ. Puisqu'on veut atteindre x^*, nous cherchons à avoir : $\Upsilon(x^{(1)}) < \Upsilon(x^{(0)})$. Une forme particulièrement

simple est de chercher $x^{(1)}$ tel que le vecteur $x^{(1)} - x^{(0)}$ soit colinéaire à une direction de descente $d^{(0)} \neq 0$. Nous le noterons : $x^{(1)} - x^{(0)} = \rho^{(0)} d^{(1)}$, où $\rho^{(0)}$ est le pas de descente de la méthode et $d^{(0)}$ est la direction de descente. La direction et le pas de descente peuvent être fixes ou changer à chaque itération. On retrouve alors la règle d'adaptation du gradient simple, à savoir, pour le cas matriciel :

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \rho^{(k)} \mathbf{D}^{(k)}, \tag{3.16}$$

ou encore pour le cas vectoriel :

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \rho^{(k)} \mathbf{d}^{(k)}. \tag{3.17}$$

Cette méthode utilise pour direction de descente l'opposé du gradient de la fonction Υ au point courant $\mathbf{X}$. soit

$$\mathbf{D}^{(k)} = -\mathbf{G}^{(k)},$$

ou sous forme vectorisé :

$$\mathbf{d}^{(k)} = -\mathbf{g}^{(k)}.$$

Lorsque l'on travaille sur une résolution numérique d'un problème, on se donne en général un critère d'arrêt de la forme :

$$\|\Upsilon(x^{(k+1)}) - \Upsilon(x^{(k)})\| < \epsilon \tag{3.18}$$

De plus, puisque la convergence n'est pas toujours assurée, une règle de base est de fixer un nombre maximum d'itérations k_{max}. On obtient alors l'algorithme présenté dans la table 3.3 et dit du gradient.

Algorithme 3.3 Résumé de l'algorithme Gradient

Set $k = 0$, Initialize $(A^{(0)}, B^{(0)}, C^{(0)})$.
Do :
Compute the descent direction as the gradient w.r.t. $\mathbf{X}$:
$\mathbf{D}^{(k)} = -\mathbf{G}^{(k)} = \nabla\Upsilon(\mathbf{X}^{(k)})$
Compute the step size $\rho^{(k)}$
Update $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \rho^{(k)}\mathbf{D}^{(k)}$
Update k : $k = k + 1$
While $k \geq 1$ and subject to a stopping criterion : $\|\Upsilon(x^{(k+1)}) - \Upsilon(x^{(k)})\| < \epsilon$

Même si ces méthodes sont conceptuellement très simples et qu'elles peuvent être programmées directement, elles sont souvent lentes dans la pratique. Elles convergent mais sous des conditions de convergence souvent complexes. Par conséquent, on utilise plutôt la direction de gradient conjugué (CG).

3.4.3 Algorithme de gradient conjugué

L'algorithme du gradient conjugué peut être vu comme une légère modification de l'algorithme de la plus forte pente, pour lequel on garde en mémoire la direction de descente de l'itération précédente. Il construit sa direction de descente en $\mathbf{X}^{(k)}$ en ajoutant à l'opposé du gradient, la direction $\mathbf{D}^{(k-1)}$ calculée à l'itéré précédent $\mathbf{X}^{(k-1)}$, multipliée par un scalaire $\beta \in \mathbb{R}$:

La direction de descente est donnée sous forme matricielle par :

$$\mathbf{D}^{(k)} = \begin{cases} -\mathbf{G}^{(k)} & \text{si } k = 1 \\ -\mathbf{G}^{(k)} + \beta^{(k)}\mathbf{D}^{(k-1)} & \text{si } k \geq 1 \end{cases}$$

Ou sous forme vectorielle par :

$$\mathbf{d}^{(k)} = \begin{cases} -\mathbf{g}^{(k)} & \text{si } k = 1 \\ -\mathbf{g}^{(k)} + \beta^{(k)}\mathbf{d}^{(k-1)} & \text{si } k \geq 1 \end{cases}$$

Cette direction est appelée direction du gradient conjugué. À la première itération, il n'y a pas de direction précédente et l'algorithme prend alors simplement l'opposé du gradient. Le scalaire $\beta^{(k)}$ peut prendre différentes valeurs, ce qui donne à l'algorithme des propriétés différentes. Nous présentons ici les deux méthodes qui sont très fréquemment citées dans la littérature. Elles calculent $\beta^{(k)}$ par les formules suivantes :

Méthode de Fletcher-Reeves :
La méthode de Fletcher-Reeves consiste à calculer $\beta_{FR}^{(k+1)}$ de la manière suivante :

$$\beta_{FR}^{(k+1)} = \frac{\langle \mathbf{G}^{(k+1)}, \mathbf{G}^{(k+1)} \rangle}{\langle \mathbf{G}^{(k)}, \mathbf{G}^{(k)} \rangle} = \frac{\|\mathbf{G}^{(k+1)}\|_F^2}{\|\mathbf{G}^{(k)}\|_F^2}$$

Méthode de Polak-Ribière :
La méthode de Polack-Ribière consiste à calculer $\beta_{PR}^{(k+1)}$ de la manière suivante :

$$\beta_{PR}^{(k+1)} = \frac{\langle \mathbf{G}^{(k+1)}, \mathbf{G}^{(k+1)} - \mathbf{G}^{(k)} \rangle}{\langle \mathbf{G}^{(k)}, \mathbf{G}^{(k)} \rangle} = \frac{\langle \mathbf{G}^{(k+1)}, \mathbf{G}^{(k+1)} - \mathbf{G}^{(k)} \rangle}{\|\mathbf{G}^{(k)}\|_F^2}$$

3.4.4 Algorithme Quasi Newton

Les méthodes Quasi-Newton consistent à imiter la méthode de Newton, où l'optimisation d'une fonction est obtenue à partir des minimisations successives de son approximation au second ordre. En effet, l'inconvénient de la direction de Newton est qu'elle nécessite la connaissance du Hessien de la fonction objectif. Le calcul de cette matrice peut être compliqué et coûteux. De plus, pour certains problèmes, le critère n'est pas deux fois différentiable. Ceci a motivé l'apparition des méthodes de Quasi-Newton qui définissent la direction par :

$$\mathbf{d}^{(k)} = -(\mathbf{B}^{(k)})^{-1}\mathbf{g}^{(k)}$$

où $\mathbf{B}^{(k)}$ est une approximation du Hessien générée itérativement par une formule de mise à jour (Dennis et Moré, 1977). La première méthode de quasi-Newton a été suggérée par Davidon (Davidon, 1991) en 1959, puis améliorée par Fletcher et Powell (Fletcher et Powell, 1963) en 1963, connue sous la dénomination de formule DFP. La formule de quasi-Newton la plus robuste est celle de BFGS, indépendamment suggérée par Broyden (BROYDEN, 1970), Fletcher (Fletcher, 1970), Goldfarb (Goldfarb, 1970) et Shanno (Shanno, 1970) en 1970. La formule de mise à jour s'écrit comme :

$$\mathbf{B}^{(k+1)} = \mathbf{B}^{(k)} - \frac{\mathbf{B}^{(k)}\mathbf{z}^{(k)}(\mathbf{z}^T)^{(k)}\mathbf{B}^{(k)}}{(\mathbf{z}^T)^{(k)}\mathbf{B}^{(k)}\mathbf{z}^{(k)}} + \frac{\delta^{(k)}(\delta^T)^{(k)}}{(\delta^T)^{(k)}\mathbf{z}^{(k)}}$$

avec :

$$\mathbf{z}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \ , \ \delta^{(k)} = \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}$$

La direction de Quasi-Newton nécessite de calculer la matrice $\mathbf{M}^{(k)} = (\mathbf{B}^{(-1)})^{(k)}$. Cette matrice s'obtient simplement par récurrence :

$$\mathbf{M}^{(k+1)} = \mathbf{M}^{(k)} + \varrho[1 + \varrho(\delta^T)^{(k)})\mathbf{M}^{(k)}\delta^{(k)}]\mathbf{z}^{(k)}(\mathbf{z}^T)^{(k)} - \varrho\mathbf{z}^{(k)}(\delta^T)^{(k)})\mathbf{M}^{(k)} - \varrho\mathbf{M}^{(k)}\delta^{(k)}(\mathbf{z}^T)^{(k)}$$

La règle d'adaptation de l'algorithme BFGS sera alors :

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \rho^{(k)}\mathbf{M}^{(k)}\mathbf{g}^{(k)}. \tag{3.19}$$

3.4.5 Algorithme de Levenberg-Marquardt

Cette méthode est connue depuis déjà de nombreuses années, proposée pour la première fois par Levenberg en 1944 (Levenberg, 1944). Il a fallu ensuite attendre 20 ans pour voir l'amélioration de Marquardt (Marquardt, 1963). Comme pour la méthode de Gauss-Newton (voir la sous-section précédente), la méthode de Levenberg-Marquardt est une méthode itérative fondée sur un schéma newtonien qui prend en compte l'approximation du hessien.

Dans certaines applications, par exemple lorsque l'approximation du hessien n'est pas définie positivement, la direction $\mathbf{d}^{(k)}$ trouvée par l'algorithme de Gauss-Newton ne produit pas une réduction suffisante de la fonction objectif. Dans ce cas, la méthode de Levenberg-Marquardt permet de forcer la convergence en rapprochant la direction de descente $\mathbf{d}^{(k)}$ de la direction $-\mathbf{g}^{(k)}$. Cette méthode consiste à approcher le hessien de la fonction objectif en ajoutant un multiple de la matrice identité à la matrice $\mathbf{B}$ avant son inversion. L'équation 3.19 deviendra alors :

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \rho^{(k)}(\mathbf{B}^{(k)} + \alpha\mathbf{I})^{-1}\mathbf{g}^{(k)}. \tag{3.20}$$

avec α est le coeffcient de relaxation et $\mathbf{I}$ la matrice identité de taille $(I + J + K) \times R$. Si on pose $\alpha = 0$ dans l'équation 3.20, on retombe sur l'algorithme BFGS (équation 3.19).

De même, en posant $\mathbf{M} = \mathbf{I}$ ou encore, en considérant α suffisamment grand par rapport aux valeurs de $\mathbf{M}$, l'algorithme du Gradient est retrouvé (équation 3.17).

3.4.6 Algorithme Newton

Dans l'algorithme de Newton pour l'optimisation sans contrainte, on détermine une direction de descente $\mathbf{d}^{(k)}$ par la formule suivante :

$$\mathbf{d}^{(k)} = -(\mathbf{H}^{(k)})^{-1}\mathbf{g}^{(k)}$$

Cette direction est appelée direction de Newton. Il faut évidemment que la hessienne de Υ en l'itéré courant soit inversible pour que cette définition ait un sens.

3.4.7 Simulations

Dans cette section, nous allons présenter plusieurs simulations pour les différents algorithmes d'optimisation de la décomposition CP d'un tenseur d'ordre 3. Nous allons illustrer le comportement des différents algorithmes présentés auparavant, afin de mettre en évidence l'impact de plusieurs facteurs sur leurs performances.

Pour la première simulation, nous considérons un tenseur $\mathcal{T}_1$ non bruité d'ordre 3, de taille $15 \times 14 \times 16$ et de rang 5. Donc, nous somme menés à estimer $(14 + 15 + 16) \times 5$ variables. Le tenseur estimé sera noté $\mathcal{T}_1$ avec $\mathcal{T}_1 = \Sigma_{r=1}^{R} \hat{a}_r \circ \hat{b}_r \circ \hat{c}_r$ et $\hat{a}_r$, $\hat{b}_r$ et $\hat{c}_r$ les estimés de la $r^{\text{ème}}$ colonne des différentes matrices de facteurs pour $r = 1, ..., R$.

Pour avoir la possibilité de comparer entre les différents algorithmes, nous avons besoin d'un indice de performance. Pour ce premier scénario, nous choisissons d'utiliser un indice d'erreur, défini comme :

$$E = \|\mathcal{T}_1 - \hat{\mathcal{T}}_1\|_F^2. \tag{3.21}$$

Lorsque l'indice de performance E est proche de 0, les résultats obtenus sont dites meilleurs. Donc nous devons donner un seuil à partir duquel on considère que le minimum global de notre fonction est atteint, par exemple $\epsilon = 10^{-10}$. De ce fait, le critère d'arrêt pour les algorithmes itératifs sera $E < \epsilon$. Dans le cas des données non bruitées et si la décomposition est unique, la norme des résidus est nulle lorsque le minimum global est atteint, ce qui interprète bien le critère (3.21). Toutefois, lorsqu'un palier est rencontré, l'indice de performance E stagne de telle sorte à ce que le critère (3.21) ne peut pas être satisfait dans un délai de temps. Pour cette raison, le critère (3.21) sera couplé à un autre critère qui permet d'éviter les cas extrêmes, qui est un nombre d'itérations maximum fixe.

Dans le cas des données bruitées, la norme des résidus est non nulle lorsque le minimum global de notre fonction est atteint. De ce fait, le critère (3.21) ne peut pas être utilisé, mais nous utiliserons plutôt le critère suivant :

$$\|E^{(k)} - E^{(k+1)}\| < \epsilon. \tag{3.22}$$

Ce critère (3.21) sera utilisé en plus des deux autres mentionnés précédemment.

Après avoir générer les trois matrices de facteurs **A**, **B** et **C** à partir des distributions gaussiennes, nous avons testé chaque algorithme d'optimisation étudié dans les sections précédentes. Les figures 3.11, 3.12 et 3.13 illustrent l'erreur de reconstruction du tenseur $\mathcal{T}_1$ par rapport au nombre d'itérations réalisées pour les algorithmes Gradient, Gradient Conjugé, Levenberg Marquart et BFGS respectivement.

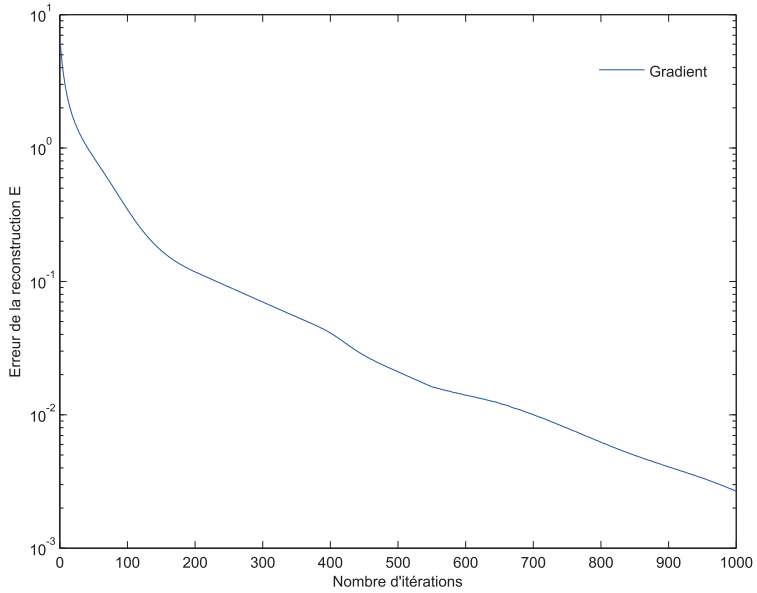


FIGURE 3.11 – Erreur de reconstruction du tenseur $\mathcal{T}_1$ par rapport au nombre d'itération demandé pour l'algorithme Gradient.

Dans la figure 3.14, une comparaison entre les différents algorithmes d'optimisation est faite. À partir des résultats obtenus dans cette figure, on remarque bien que l'algorithme Levenberg Marquardt converge vers la valeur de 10^{-10} pour un nombre d'itérations petit, 15 itérations, alors que c'est l'algorithme du Gradient qui demande plus d'itérations pour converger. Toutefois, cela n'est pas suffisant pour conclure que l'algorithme Levenberg Marquardt est le meilleur.

Pour faire une bonne comparaison entre les algorithmes d'optimisation étudiés, nous avons évalué la complexité algorithmique de ces derniers, ainsi que le temps de calcul. Le tableau 3.2 représente le temps pris par chaque algorithme pour converger vers le minimum de la fonction de coût (équation 3.1). D'après le tableau 3.2, on constate que l'algorithme de Gradient nécessite moins de temps pour converger vers le minimum en le comparant avec les deux autres algorithmes. Par contre, l'algorithme de Livenberg Marquardt, qui demande un nombre petit d'itérations (3.14), prend plus de temps pour converger vers le minimum.

Le troisième indice de performance évalué dans cette section est la complexité algorithmique, qui représente un bon moyen de comparaison du temps de calcul des différents algorithmes indépendamment, du processeur de calcul utilisé.

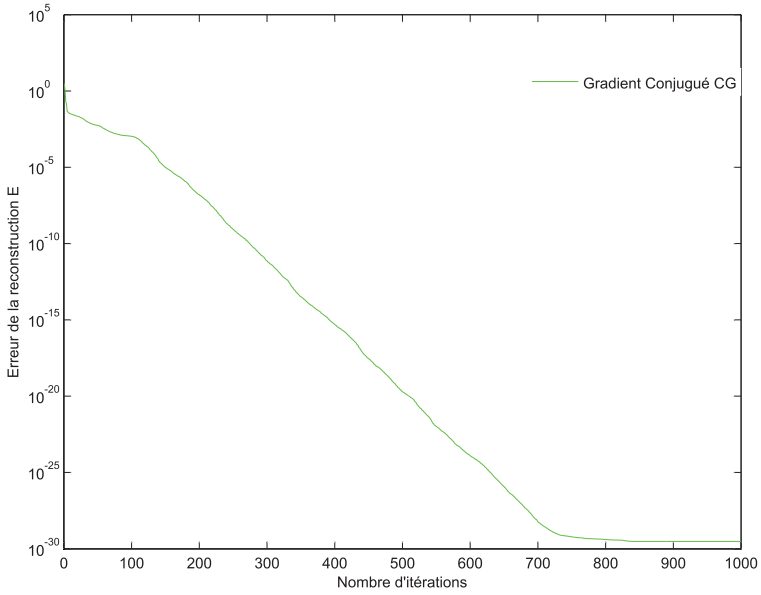


FIGURE 3.12 – Erreur de reconstruction du tenseur $\mathcal{T}_1$ par rapport au nombre d'itération demandé pour l'algorithme Gradient Conjugué.

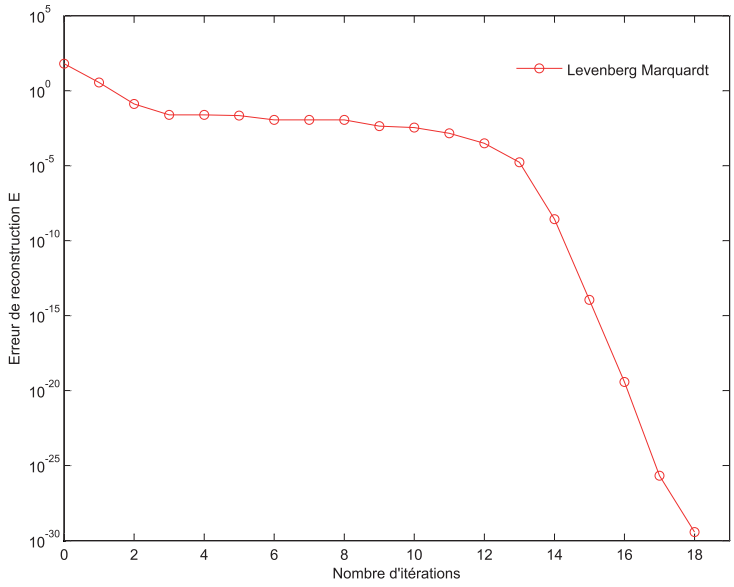


FIGURE 3.13 – Erreur de reconstruction du tenseur $\mathcal{T}_1$ par rapport au nombre d'itération demandé pour l'algorithme Levenberg Marquardt.

Commençant d'abord par l'algorithme *ALS*. Pour une seule itération, et pour estimer la matrice de facteurs $\mathbf{A}$, on a un produit de Khatri Rao de complexité $O(JKR)$, le calcul de pseudo inverse de complexité $O(7JKR^2 + \frac{11}{3}R^3)$, ainsi que l'expression de l'estimation de la matrice $\mathbf{A}$ de complexité $O(IJKR + IR^2 + IR)$.

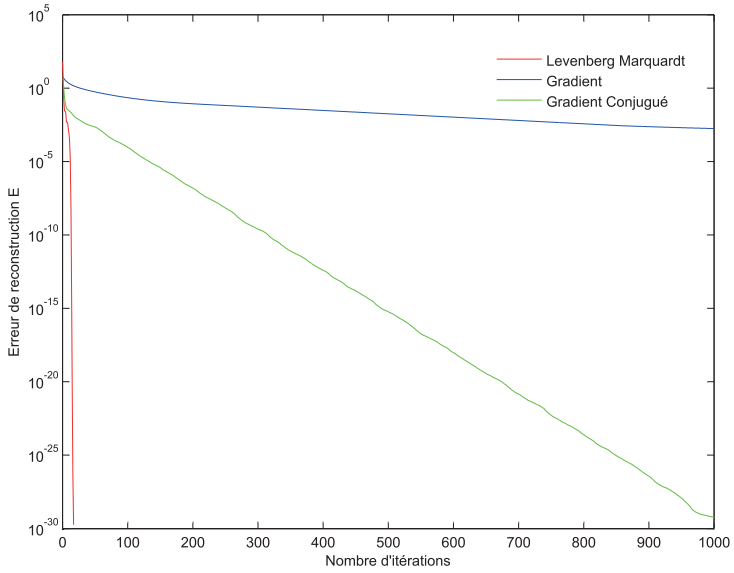


FIGURE 3.14 – Erreur de reconstruction par rapport au nombre d'itération demandé pour le tenseur $\mathcal{T}_1$ d'ordre 3 et de taille $I = 15$, $J = 14$ et $K = 16$.

Algorithme	Temps de Calcul (s)
Gradient	40.62
Gradient Conjugué (CG)	48.95
Levemberg Marquardt (LM)	80.61

TABLE 3.2 – Le temps de calcul pour les trois algorithmes : Gradient, Gradient Conjugué, Levemberg Marquardt.

Donc, la complexité de l'algorithme ALS pour une seule itération est de : $O(3IJKR + 11R^3 + (R + R^2)(I + J + K) + (R + 7R^2)(IJ + IK + JK))$.

Concernant l'algorithme Gradient, il y a trois composantes de gradient à calculer. Dans chaque composante, on a un produit de Khatri Rao, et 4 produits matriciel. Pour la première composante (4.15), la complexité du produit de Khatri Rao est $O(JKR)$ et celle des 4 produits matriciel (Le produit entre le produit de Khatri Rao et la matrice dépliante du tenseur, le produit entre $\mathbf{C}^T$ et $\mathbf{C}$, le produit entre $\mathbf{B}^T$ et $\mathbf{B}$ et le produit entre la matrice $\mathbf{A}$ et le produit de Hadamard) est $O(IJKR + IR^2 + JR^2 + KR^2)$. De même pour les deux autres composantes du gradient en respectant les matrices de facteurs. La complexité algorithmique du Gradient, pour une seule itération, est de : $O(3IJKR + 3R^2(I + J + K) + R(IJ + IK + JK))$.

Pour l'algorithme du Gradient Conjugué GC, on trouve le calcul des composantes de Gradient ainsi que le coefficient (4.28) pour trouver la direction de descente. La

complexité de cet algorithme sera alors la somme des complexités de l'algorithme Gradient et celle du calcul du coefficient. Dans le calcul de ce dernier, on a deux produits matriciels entre les matrices $\mathbf{G}$, de taille $(I+J+K)R$, qui contiennent les composantes du gradient. Par conséquence, la complexité de l'algorithme Gradient Conjugué CG est de : $O(3IJKR + 5R^2(I+J+K) + R(IJ+IK+JK))$.

En ce qui concerne l'algorithme Quasi Newton BFGS (4.34), en plus du calcul des composantes du gradient, nous aurons besoin de calculer l'approximé de la matrice Hessienne. Cette dernière demande 5 produits matriciels et une inversion matricielle. De ce fait, la complexité algorithmique par itération est de l'ordre de : $O((I+J+K)^3R^3 + 4(I+J+K)^2R^2 + 3IJKR + 3R^2(I+J+K) + R(IJ+IK+JK))$.

Pour l'algorithme de Levenberg Marquardt, il a la même complexité algorithmique que BFGS, vu que la seule différence de calcul est l'ajout d'un multiple de la matrice identité à la matrice $\mathbf{B}$ avant son inversion. Le tableau 3.3 représente un résumé de complexité algorithmique des différents algorithmes étudiés dans cette section.

Algorithme	Complexité par itération
ALS	$O(3IJKR + 11R^3 + (R+R^2)(I+J+K) + (R+7R^2)(IJ+IK+JK))$
Gradient	$O(3IJKR + 3R^2(I+J+K) + R(IJ+IK+JK))$
CG	$O(3IJKR + 5R^2(I+J+K) + R(IJ+IK+JK))$
BFGS	$O((I+J+K)^3R^3 + 4(I+J+K)^2R^2 + 3IJKR + 3R^2(I+J+K) + R(IJ+IK+JK))$
LM	$O((I+J+K)^3R^3 + 4(I+J+K)^2R^2 + 3IJKR + 3R^2(I+J+K) + R(IJ+IK+JK))$

TABLE 3.3 – La complexité algorithmique des différentes méthodes.

3.5 Les méthodes pour le choix du pas de recherche

Dans les algorithmes de descente itératifs, le schéma d'optimisation se formule comme la détermination, à chaque itération, d'un pas minimisant le critère dans un sous-espace généré par une ou plusieurs directions de recherche. Dans cette section, nous étudierons les techniques employées pour déterminer le pas, appelée recherche ou stratégie de pas. Avoir une bonne stratégie de pas est fondamental car cela influe de façon importante sur les propriétés de convergence de l'algorithme.

Nous présentons dans cette section, les stratégies usuelles pour le calcul du pas scalaire, également appelées stratégies de recherche linéaire, en se basant principalement sur [Bertsekas *et al.*, 1999, Boyd *et al.*, 2004].

Les algorithmes de descente présentés ci-dessus requièrent tous l'estimation d'un pas qui varie à chaque itération. La recherche linéaire consiste donc a déterminer le pas $\rho^{(k)}$ a effectuer le long d'une direction de descente $\mathbf{d}^{(k)}$. Il reste bien sûr toujours possible d'utiliser un pas fixe tout au long des itérations, mais cela risque d'être synonyme d'une faible vitesse de convergence si le pas est choisi trop petit. Dans le cas contraire, c'est à dire si on choisi un pas trop grand, l'algorithme risque de diverger, et on se trouvera alors dans l'incapacité d'estimer correctement les matrices de facteurs. Lorsque le sous-espace se limite à une unique direction, la recherche du pas consiste en la minimisation approchée d'une fonction scalaire suivante :

$$\rho^{(k)} = \underset{\rho}{argmin}\{\Upsilon(\mathbf{x}^{(k)} + \rho\mathbf{d}^{(k)})\} \tag{3.23}$$

Le pas $\rho^{(k)}$ est souvent choisi afin de vérifier les deux conditions suivantes :

— La fonction Υ doit décroitre suffisamment le long de la direction de descente ;
— Le pas $\rho^{(k)}$ ne doit pas être trop petit.

Le premier point est habituellement réalisé en forçant l'inégalité :
$\Upsilon(x + \alpha^{(k)}\mathbf{d}) \leq \Upsilon(x)$ en rajoutant un terme négatif $\beta^{(k)}$ au second membre :

$$\Upsilon(x + \rho^{(k)}\mathbf{d}) \leq \Upsilon(x) + \beta^{(k)}$$

Cette condition est habituellement vérifiée par des pas $\rho^{(k)}$ petits. Or de trop petits pas peuvent entrainer une fausse convergence : le second point doit donc également être satisfait.

3.5.1 Pas exactes

La détermination du pas exact d'une fonction f correspond à *la règle de Cauchy* 3.24. Il permet, comme son nom l'indique, de déterminer le pas de recherche qui minimse au maximum la fonction f à l'étape k, en fonction de la direction de descente $\mathbf{d}^{(k)}$ calculée. Lorsque la fonction f est multimodale, il peut être plus simple de rechercher le minimum dans un intervalle restreint (*règle de Cauchy limitée* 3.25) ou bien de déterminer le plus petit point stationnaire de f (*règle de Curry* 3.26) [Bertsekas *et al.*, 1999]. Ces règles, qualifiées d'exactes, ne sont utilisées que lorsque le minimum s'exprime de façon analytique.

1. Règle de Cauchy :

$$\rho^{(k)} = \underset{\rho\in\mathbb{R}^+}{argmin}\{f(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)})\} \tag{3.24}$$

2. Règle de Cauchy limitée :

$$\rho^{(k)} = \underset{\rho\in]0,s]}{argmin}\{f(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)})\} \tag{3.25}$$

3. Règle de Curry :

$$\rho^{(k)} = \underset{\rho \geq 0}{min}\{\rho > 0 | \nabla f(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)})^T \mathbf{d}^{(k)} = 0\} \tag{3.26}$$

La recherche linéaire exacte consiste donc à déterminer le pas optimal, c'est à dire le pas qui minimise la fonction f en fonction de la direction de descente $\mathbf{d}^{(k)}$ calculée. Le pas $\rho^{(k)}$ est donc solution du problème :

$$\rho^{(k)} = \underset{\rho \in \mathbb{R}^+}{argmin}\{f(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)})\}.$$

Revenons à notre fonction objectif 3.1, qui s'agit de minimiser la fonction dans le cas de la décomposition CP pour un tenseur du 3$^{\text{ème}}$ ordre suivant la *Règle de Cauchy* :

$$\Upsilon(\mathbf{A}^{(k+1)}, \mathbf{B}^{(k+1)}, \mathbf{C}^{(k+1)}) = \Upsilon[(\mathbf{A}^{(k)} + \rho\mathbf{D}_{\mathbf{A}}^{(k)}), (\mathbf{B}^{(k)} + \rho\mathbf{D}_{\mathbf{B}}^{(k)}), (\mathbf{C}^{(k)} + \rho\mathbf{D}_{\mathbf{C}}^{(k)})]$$

Le détail des calculs est donné au niveau de l'annexe B. Il s'agit en fait d'un polynôme de degré 6, dont l'expression est donnée par

$$\Upsilon(.) = \sum_{i=0}^{6} a_i \rho^i$$

Où les 7 coefficients a_i, pour $i = 0, \ldots, 6$ valent (la définition de k_i, où i varie de 0 à 6 est donnée à l'annexe B.1)

$$\begin{aligned}
a_0 &= \text{trace}[\mathbf{K}_0\mathbf{K}_0^T] \\
a_1 &= \text{trace}[2\mathbf{K}_1\mathbf{K}_0^T] \\
a_2 &= \text{trace}[2\mathbf{K}_2\mathbf{K}_0^T + \mathbf{K}_1\mathbf{K}_1^T] \\
a_3 &= \text{trace}[2(\mathbf{K}_3\mathbf{K}_0^T + \mathbf{K}_2\mathbf{K}_1^T)] \\
a_4 &= \text{trace}[2\mathbf{K}_3\mathbf{K}_1 + \mathbf{K}_2\mathbf{K}_2^T] \\
a_5 &= \text{trace}[2\mathbf{K}_3\mathbf{K}_2^T] \\
a_6 &= \text{trace}[\mathbf{K}_3\mathbf{K}_3^T]
\end{aligned}$$

En dérivant l'expression Υ par rapport à ρ, on obtient un polynôme de degré 5 donné par l'équation :

$$\mathrm{d}\Upsilon(.) = \sum_{i=0}^{5} (i+1) a_{i+1} \rho^i$$

Le pas optimal $\rho_{opt}^{(k)}$ correspond alors à la racine réelle et positive du polynôme menant au minimum du critère donné en 3.24. Ces racines sont estimées numériquement car pour

des degrés aussi élevés, il n'existe généralement pas de formule algébrique pour le calcul des racines.

Cependant, la résolution exacte peut demander un temps de calcul trop grand, et ne peut de toute façon pas être faite avec une précision infinie. De plus, l'efficacité supplémentaire, éventuellement apportée à l'algorithme par un pas exact permet rarement de compenser le temps consacré à déterminer un tel pas. Par conséquent, les stratégies usuelles pour le calcul du pas sont basées sur la vérification de conditions moins restrictives, qui permettent toutefois de garantir la convergence des algorithmes dans un temps raisonnable.

3.5.2 Pas approchés

Le calcul du pas exacte n'implique pas forcément des performances optimales de l'algorithme. De plus, elle n'est pas nécessaire pour assurer la convergence, dès lors que des conditions sont assurées par le pas utilisé. De plus, le calcul du pas exacte (ou pas optimal) est une solution coûteuse en terme du temps de calcul, qui consiste à calculer les racines d'un polynôme de degré élevé dans le cas de notre fonction objectif (équation 3.1). Dans des problèmes de grandes dimensions (des matrices de facteurs de grandes tailles), à chaque itération, le calcul du pas exacte représente la plus grande partie de la complexité algorithmique totale. Une condition naturelle est de demander au pas d'entraîner une décroissance suffisante de la fonction f. Cela se traduit le plus souvent par une inégalité de la forme :

$$f(\mathbf{x} + \rho^{(k)}\mathbf{d}) \leq f(\mathbf{x}) + \mathbf{C}, \tag{3.27}$$

avec $\mathbf{C}$ un terme inférieur strictement à zero. Cette forme est appelée condition de descente suffsante. Parmi les différentes méthodes de recherche du pas inexacte ou approché, on retrouve une qui est très simple et très efficace, appelée recherche du pas par marche arrière (*Backtracking*). Cette méthode permet à moindre coût d'obtenir une bonne approximation du pas $\rho^{(k)}$. Elle dépend de deux constantes α et β , avec $0 < \alpha < 0,5$ et $0 < \beta < 1$.

Cette recherche de pas est appelée marche arrière, car il commence avec un pas ρ suffisamment grand au début, par exemple un pas unité, puis cette valeur est réduite par le facteur β tel que $\rho = \rho\beta$, jusqu'à ce que la condition suivante sera vérifiée :

$$f(\mathbf{x} + \rho\mathbf{d}) \leq f(\mathbf{x}) + \alpha\rho\nabla f(\mathbf{x})^T\mathbf{d}. \tag{3.28}$$

La figure 3.15 illustre le principe de la méthode marche arrière (Backtracking).

3.5.2.1 Règle d'Armijo

La condition 3.28 est appelée la condition d'Armijo [Boyd *et al.*, 2004, Luenberger *et al.*, 1984]. $\mathbf{d}$ est la direction de descente et $\nabla f(\mathbf{x})$ est le gradient vu dans les sections précédentes. Dans le cas de l'algorithme du gradient $\mathbf{d} = -\mathbf{g}$, alors que $\mathbf{d} = -\mathbf{Mg}$ pour les algorithmes newtoniens (BFGS par exemple). La constante α peut être

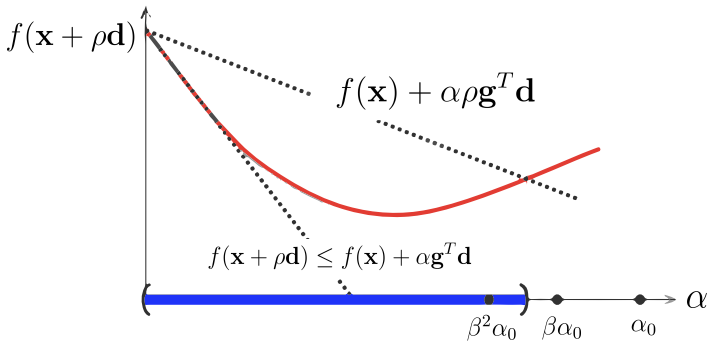


FIGURE 3.15 – Le principe de la méthode marche arrière.

interprétée comme la fraction de la diminution de la fonction f prédite par extrapolation linéaire acceptée. Le paramètre α est typiquement choisi entre $0,01$ et $0,3$, ce qui signifie que l'on accepte une diminution de f comprise entre 10% et 30% de la prédiction par extrapolation linéaire. Le paramètre β est souvent choisi entre $0,1$ et $0,8$ [Boyd *et al.*, 2004].

En reprenant nos notations, et en considérant notre fonction de coût $\Upsilon(.)$ (3.1), la condition d'Armijo s'écrit :

$$f(\mathbf{x}+\rho\mathbf{d}) \leq f(\mathbf{x}) + \alpha\rho\mathbf{g}^T\mathbf{d}. \tag{3.29}$$

Un résumé de l'algorithme marche arrière est donné dans la table 3.4.

Algorithme 3.4 Résumé de l'algorithme marche arrière

Given a descent direction $\mathbf{d}^{(k)}$ for Υ, $\alpha \in [0, 0.5]$ et $\beta \in [0, 1]$.
Initialize : $\rho = 1$.
Do :
$\rho = \beta\rho$
While : $\Upsilon(\mathbf{x}+\rho\mathbf{d}) < \Upsilon(\mathbf{x}) + \alpha\rho\mathbf{g}^T\mathbf{d}$

3.5.2.2 *Règle Goldstein*

Goldstein propose en 1967 une méthode basée sur le choix cette fois-ci de deux paramètres $0 < \alpha_1 < \alpha_1^{'} < 1$ et détermine les valeurs approchées de $\rho^{(k)}$ par deux conditions :

$$\begin{cases} f(\mathbf{x}^{(k)}+\rho^{(k)}\mathbf{d}^{(k)}) & \leq f(\mathbf{x}^{(k)}) + \alpha_1\rho^{(k)}(\mathbf{g}^{(k)})^T\mathbf{d}^{(k)}, \\ f(\mathbf{x}^{(k)}+\rho^{(k)}\mathbf{d}^{(k)}) & \geq f(\mathbf{x}^{(k)}) + \alpha_1^{'}\rho^{(k)}(\mathbf{g}^{(k)})^T\mathbf{d}^{(k)}. \end{cases}$$

où $\alpha_1^{'}$ est une constante choisie dans l'intervalle $]\alpha_1, 1[$ (par exemple $\alpha_1^{'} = 0.99$). C'est cette inégalité qui empêche le pas d'être trop petit. Le pas déterminé par cette règle est appelé pas de Goldstein.

3.5.2.3 Règle de Wolfe

Wolfe propose en 1969 une méthode basée sur le choix de deux paramètres $0 < \alpha_1 < \alpha_2 < 1$, cette méthode a pour but de déterminer un pas $\rho^{(k)}$ vérifiant les deux conditions suivantes :

— la fonction f doit décroître de manière significative :

$$f(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) \leq f(\mathbf{x}^{(k)}) + \alpha_1\rho^{(k)}(\mathbf{g}^{(k)})^T\mathbf{d}^{(k)} \tag{3.30}$$

— Afin d'empêcher le pas $\rho^{(k)}$ d'être trop petit, il lui est imposé de vérifier la condition supplémentaire suivante :

$$\nabla f(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)})^T\mathbf{d}^{(k)} \geq \alpha_2(\mathbf{g}^{(k)})^T\mathbf{d}^{(k)} \tag{3.31}$$

Deux valeurs usuelles des paramètres sont $\alpha_1 = 0{,}1$ et $\alpha_2 = 0{,}9$.

3.6 Bilan du chapitre

Dans ce chapitre, nous avons présenté les différents algorithmes d'optimisation qui minimisent la décomposition approximative du tenseur $\mathcal{T}$.

En premier lieu, un état de l'art des méthodes d'optimisation mathématiques a été dressé. Ces méthodes peuvent être réunies en deux différents groupes : les méthodes déterministes et les méthodes stochastiques. Les méthodes déterministes peuvent trouver le minimum global de la fonction sous certaines hypothèses, comme la convexité et la différentiabilité. En d'autres termes, si la fonction objectif remplit ces hypothèses dans une région locale contenant le minimum désiré, et si la configuration initiale est quelque part à l'intérieur de cette région, les méthodes déterministes convergent très rapidement vers ce minimum.

Une étude a porté sur les aspects fondamentaux des algorithmes déterministes standard. Plus précisément, les méthodes mathématiques (Gradient, Gradient conjugué, Levenberg Marquardt, Quasi Newton, BFGS ...), et des améliorations ont été proposées pour les rendre plus performants.

Cependant, et comme indiqué dans le chapitre précédent, le fait d'incorporer la contrainte sur le facteur d'échelle et celle sur la cohérence nous permettra de bien estimer les matrices de facteurs de la décomposition CP tout en surmontant les cas difficiles des marécages et des goulots d'étranglement. L'étude des algorithmes mathématiques de ce chapitre nous a montré leurs non-fonctionnement sur des problèmes d'optimisation avec contraintes. Cette particularité nous a conduit à faire appel à de nouveaux algorithmes d'optimisation capables de prendre en compte deux contraintes : une contrainte d'égalité sur les colonnes des matrices facteurs, et une autre d'inégalité sur les cohérences. Cet aspect sera l'objectif du prochain chapitre.

Chapitre

4 L'OPTIMISATION CP-DÉCOMPOSITION AVEC CONTRAINTE

Sommaire

4.1 Introduction

Il existe de nombreuses situations où les algorithmes de la CP décomposition peuvent souffrir de l'absence ou de la lenteur de la convergence, et qui peuvent être dues aux phénomènes de Goulot (Bottleneck) ou de Marécage (Swamp). De plus, nous avons présenté dans le chapitre 2 que l'introduction d'une condition supplémentaire sur les cohérences, l'existence d'une meilleure approximation de rang inférieur sera prouvée, ce qui montre que la solution au problème (3.1) existe toujours en ajoutant cette condition sur la cohérence, et où nous avons deja constater que cette condition donne déjà une limite quantitative pour le conditionnement de l'équation (2.23), en raison des cohérences qui limitent les entrées extra diagonales de la matrice $\mathbf{G}$, possèdant des 1 sur sa diagonale [Comon *et al.*, 2013].

Il s'est avéré utile de trouver des algorithmes d'optimisation qui minimisent le problème (3.1), tout en prenant en compte le facteur d'échelle optimal $\mathbf{\Lambda}$. En effet, comme présenté dans la section 2.4.2, l'isolement du facteur d'échelle dans la décomposition CP a plusieurs implications, spécialement sur l'existence et l'unicité de la solution de (3.1). Donc, il faut trouver des algorithmes d'optimisation qui prennent en compte deux contraintes : une contrainte d'égalité sur les colonnes des matrices facteurs, et une autre d'inégalité sur les cohérences.

Ce chapitre est dédiée aux algorithmes que nous proposerons pour minimiser la fonction (3.1). Dans un premier temps, nous introduisons des algorithmes basés sur la descente de gradient pour minimiser la fonction de coût sous une contrainte d'égalité, tout en indiquant les deux façons pour calculer le facteur d'échelle comme décrit dans la section 2.4.2. Par la suite, nous proposerons des algorithmes d'optimisation qui minimisera cette fois-ci notre fonction de coût sous deux contraintes, d'égalité et d'inégalité. Ceci donne lieu alors naturellement à des algorithmes barrières et des algorithmes proximaux. Ces derniers sont désormais des outils d'optimisation fiables et puissants, conduisant à une variété d'algorithmes, tels que l'algorithme de point proximal (en anglais : Proximal Point), l'algorithme avant-arrière (en anglais : Forward-Backward), l'algorithme de gradient proximal (en anglais : Proximal Gradient), l'algorithme de gradient proximal accéléré (en anglais : Accelerated Proximal Gradient), et plusieurs autres incluant la linéarisation et/ou la splitting.

4.2 Algorithmes avec contrainte d'égalité

4.2.1 Introduction

Notre problème d'optimisation consiste à minimiser l'erreur quadratique (3.1) sous une contrainte d'égalité. C'est une contrainte qui est toujours active dans le problème d'optimisation. Elle se définie comme un ensemble de vecteur unitaire qui appartient à un espace de Hilbert $\mathcal{H}$. Le problème d'optimisation que nous allons traité dans cette

première partie est le suivant :

$$\boxed{\begin{aligned} \textbf{min } \Upsilon(\mathbf{\Lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C}) &= \|\mathcal{T} - \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\|_F^2 \\ \textbf{tel que :} \quad \|\mathbf{a}_r\| &= \|\mathbf{b}_r\| = \|\mathbf{c}_r\| = 1. \end{aligned}} \tag{4.1}$$

La difficulté qui se pose dans cette optimisation est de savoir comment se déplacer dans l'ensemble des contraintes. Donc, les algorithmes d'optimisation vus aux sections 3.3 et 3.4 ne sont plus valables pour le problème (4.1). Une élaboration des méthodes d'optimisation permettant la prise en compte des contraintes d'égalité sera faite dans cette section.

Généralement, un problème d'optimisation avec contrainte d'égalité est de cette forme :

$$\begin{cases} \underset{x}{min}\ f(x) \\ tel\ que : h(x) = 0 \\ x \in \mathcal{C} \end{cases} \tag{4.2}$$

La première question est celle de l'existence du minimum global de la fonction f sur $\mathcal{C}$. Il existe principalement deux théorèmes qui permettent de répondre à cette question. Le premier affirme l'existence d'un point de minimum lorsque l'ensemble des contraintes est fermé borné. Le second est son équivalent pour les ensembles de contraintes fermés mais non bornés. Étant donnée une direction de recherche, comment garantir que l'on reste dans l'ensemble des contraintes $\mathcal{C}$. Pour cela, on introduit la notion de direction admissible.

Définition (Direction admissible) : Soit $\mathbf{x} \in \mathbb{R}^n$ un point admissible du problème (4.2). Une direction $\mathbf{d} \in \mathbb{R}^n$ sera dite admissible en $\mathbf{x}$ s'il existe $\eta > 0$ tel que $\mathbf{x} + \rho\mathbf{d}$ soit admissible quel que soit $\rho \in]0; \eta]$.

Rappelons que, dans le cas sans contrainte, les algorithmes de descente s'écrivent sous la forme générique :

$$\begin{cases} \mathbf{x}^{(0)} donne, \\ \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \rho^{(k)} \mathbf{d}^{(k)} \end{cases} \tag{4.3}$$

Où $\rho^{(k)}$ et $\mathbf{d}^{(k)}$ sont choisis de telle sorte que $f(\mathbf{x}^{(x+1)}) \leq f(\mathbf{x}^{(x)})$. Lorsqu'on minimise sous un ensemble de contraintes $\mathcal{C}$, il n'est pas sûr que $\mathbf{x}^{(k+1)}$ reste sur $\mathcal{C}$. Il est donc nécessaire de se ramener sur $\mathcal{C}$. On réalise cette dernière opération grâce à une projection sur $\mathcal{C}$. Ceci donne lieu alors naturellement aux algorithmes de descente projetés, tel que le Gradient projeté et le Gradient Conjugué projeté.

Construisons maintenant un algorithme de descente projeté qui résout le problème (4.1). Cet algorithme doit trouver trois matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ de colonnes de norme 1 qui minimisent la fonction (4.1). En d'autre terme, trouver les trois matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ sur une sphère de rayon 1.

La méthode du gradient projeté s'inspire des méthodes de gradient décrites dans la section 3.4. L'idée de base consiste à suivre la direction de plus profonde descente, comme dans le cas sans contrainte :

$$\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} - \rho^{(k)} \nabla \Upsilon_{\mathbf{A}}$$

$$\mathbf{B}^{(k+1)} = \mathbf{B}^{(k)} - \rho^{(k)} \nabla \Upsilon_{\mathbf{B}}$$

$$\mathbf{C}^{(k+1)} = \mathbf{C}^{(k)} - \rho^{(k)} \nabla \Upsilon_{\mathbf{C}}$$

où $\rho^{(k)} > 0$ est choisi de sorte que : $\Upsilon(\lambda^{(k+1)}, \mathbf{A}^{(k+1)}, \mathbf{B}^{(k+1)}, \mathbf{C}^{(k+1)}) \leq \Upsilon(\lambda^{(k)}, \mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)})$. Toutefois, si $(\lambda^{(k)}, \mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)}) \in \mathcal{C}$, rien ne garantit que $(\lambda^{(k+1)}, \mathbf{A}^{(k+1)}, \mathbf{B}^{(k+1)}, \mathbf{C}^{(k+1)})$ appartienne également à $\mathcal{C}$. Dès que l'on obtient un point non admissible, on projette celui-ci sur l'ensemble de contrainte $\mathcal{C}$. Le principe de l'algorithme est le suivant : Soit $\mathbf{x}^{(k)}$ l'itéré courant. On génère à l'itération suivante le point admissible :

$$\mathbf{y}^{(k)} = p_{\mathcal{C}}(\mathbf{x}^{(k)} - \rho^{(k)} \mathbf{g}^{(k)}), \tag{4.4}$$

où $p_{\mathcal{C}}$ désigne l'opérateur de projection sur l'ensemble $\mathcal{C}$, $\mathbf{x}^{(k)}$ est le vecteur qui contient les éléments des matrices facteurs et $\mathbf{g}^{(k)}$ est le vecteur qui contient les composantes du nouveau gradient calculées comme suivant :

$$\frac{\Upsilon(\mathbf{\Lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C})}{\partial a_{\alpha\beta}} = -2 \sum_{ijk} (T_{ijk} - \Sigma_{r=1}^{R} \lambda_r a_{ir} b_{jr} c_{kr}) \lambda_\beta b_{j\beta} c_{k\beta} \tag{4.5}$$

$$\frac{\Upsilon(\mathbf{\Lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C})}{\partial b_{\alpha\beta}} = -2 \sum_{ijk} (T_{ijk} - \Sigma_{r=1}^{R} \lambda_r a_{ir} b_{jr} c_{kr}) \lambda_\beta a_{i\beta} c_{k\beta} \tag{4.6}$$

$$\frac{\Upsilon(\mathbf{\Lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C})}{\partial c_{\alpha\beta}} = -2 \sum_{ijk} (T_{ijk} - \Sigma_{r=1}^{R} \lambda_r a_{ir} b_{jr} c_{kr}) \lambda_\beta a_{i\beta} b_{j\beta} \tag{4.7}$$

La forme matricielle des composantes du gradient $\nabla_{\mathbf{A}} \Upsilon$, $\nabla_{\mathbf{B}} \Upsilon$ et $\nabla_{\mathbf{C}} \Upsilon$ de taille $I \times R$, $J \times R$ et $K \times R$ respectivement sont :

$$\begin{aligned} \nabla_{\mathbf{A}} \Upsilon(\mathbf{\Lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C}) &= \frac{\partial \Upsilon(\mathbf{\Lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C})}{\partial \mathbf{A}} \\ &= \frac{\partial}{\partial \mathbf{A}} (\|\mathbf{T}_{(1)}^{I \times KJ} - \mathbf{A}\mathbf{\Lambda}(\mathbf{C} \odot \mathbf{B})^T\|_F^2) \\ &= 2[-\mathbf{T}_{(1)}^{I \times KJ} + \mathbf{A}\mathbf{\Lambda}(\mathbf{C} \odot \mathbf{B})^T](\mathbf{C} \odot \mathbf{B})\mathbf{\Lambda} \\ &= 2(-\mathbf{T}_{(1)}^{I \times KJ}(\mathbf{C} \odot \mathbf{B})\mathbf{\Lambda} + \mathbf{A}\mathbf{\Lambda}(\mathbf{C}^T\mathbf{C}) \boxdot (\mathbf{B}^T\mathbf{B})\mathbf{\Lambda}), \end{aligned} \tag{4.8}$$

$$\begin{aligned}\nabla_{\mathbf{B}}\Upsilon(\mathbf{\Lambda},\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{\partial\Upsilon(\mathbf{\Lambda},\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{B}} \\ &\frac{\partial}{\partial\mathbf{B}}(\|\mathbf{T}_{(2)}^{J\times KI} - \mathbf{B}\mathbf{\Lambda}(\mathbf{C}\odot\mathbf{A})^T\|_F^2) \\ &= 2[-\mathbf{T}_{(2)}^{J\times KI} + \mathbf{B}\mathbf{\Lambda}(\mathbf{C}\odot\mathbf{A})^T](\mathbf{C}\odot\mathbf{A})\mathbf{\Lambda} \\ &= 2(-\mathbf{T}_{(2)}^{J\times KI}(\mathbf{C}\odot\mathbf{A})\mathbf{\Lambda} + \mathbf{B}\mathbf{\Lambda}(\mathbf{C}^T\mathbf{C})\boxdot(\mathbf{A}^T\mathbf{A})\mathbf{\Lambda}),\end{aligned} \tag{4.9}$$

$$\begin{aligned}\nabla_{\mathbf{C}}\Upsilon(\mathbf{\Lambda},\mathbf{A},\mathbf{B},\mathbf{C}) &= \frac{\partial\Upsilon(\mathbf{\Lambda},\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{C}} \\ &\frac{\partial}{\partial\mathbf{C}}(\|\mathbf{T}_{(3)}^{K\times JI} - \mathbf{C}\mathbf{\Lambda}(\mathbf{B}\odot\mathbf{A})^T\|_F^2) \\ &= 2[-\mathbf{T}_{(3)}^{K\times JI} + \mathbf{C}\mathbf{\Lambda}(\mathbf{B}\odot\mathbf{A})^T](\mathbf{B}\odot\mathbf{A})\mathbf{\Lambda} \\ &= 2(-\mathbf{T}_{(3)}^{K\times JI}(\mathbf{B}\odot\mathbf{A})\mathbf{\Lambda} + \mathbf{C}\mathbf{\Lambda}(\mathbf{B}^T\mathbf{B})\boxdot(\mathbf{A}^T\mathbf{A})\mathbf{\Lambda}).\end{aligned} \tag{4.10}$$

Si la direction $\mathbf{d}^{(k)} = \mathbf{y}^{(k)} - \mathbf{x}^{(k)}$ est non nulle, alors c'est une direction de descente de Υ en $\mathbf{x}^{(k)}$. La direction $\mathbf{d}^{(k)}$ possède les propriétés suivantes :

1. Si $\mathbf{d}^{(k)} = 0$, alors : $p_{\mathcal{C}}(\mathbf{x}^{(k)} - \rho^{(k)}\mathbf{g}^{(k)}) = \mathbf{x}^{(k)}$. Cela signifie que la direction choisie par l'algorithme de gradient est orthogonale à l'ensemble des contraintes $\mathcal{C}$ en $\mathbf{x}^{(k)}$;
2. Supposons $\mathbf{d}^{(k)} \neq 0$. Alors $\mathbf{x}^{(k)}$ et $p_{\mathcal{C}}(\mathbf{x}^{(k)} - \rho^{(k)}\mathbf{g}^{(k)})$ sont des points admissibles du problème (4.1).

Dans les sous sections suivantes, nous allons présenter deux versions de l'algorithme Gradient projeté pour notre problème d'optimisation (4.1) dans lequel le facteur d'échelle est calculé comme étant le produit des normes des matrices facteurs, ensuite ce facteur d'échelle sera calculé de la façon optimale comme évoquée dans la section 2.4, cela permettra de bien affirmer l'importance de l'isolement de ce facteur d'échelle $\mathbf{\Lambda}$.

4.2.2 Facteur d'échelle sous optimal

Dans les derniers travaux sur la décomposition tensorielle CP, l'optimisation de la fonction objectif (3.1) était faite sans prendre en considération le facteur $\mathbf{\Lambda}$. Alors qu'on trouve que le facteur d'échelle $\mathbf{\Lambda}$ est une matrice d'identité dans les travaux [Comon *et al.*, 2009a, Rajih *et al.*, 2008], il est mis dans l'une des trois matrices dans [Royer *et al.*, 2011], ce qui rend ces dernières non-libres. Cette première solution consiste à appliquer l'algorithme de Gradient projeté tout en calculant le $\mathbf{\Lambda}$ comme produit des normes des colonnes des trois matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ après leurs normalisations (4.11).

$$\mathbf{\Lambda}^{(k+1)} = \mathbf{\Lambda}^{(k)}\mathbf{\Lambda}_{\mathbf{A}}\mathbf{\Lambda}_{\mathbf{B}}\mathbf{\Lambda}_{\mathbf{C}}, \tag{4.11}$$

avec $\mathbf{\Lambda}_{\mathbf{A}} = diag\{[\|\mathbf{a}_1\|, ..., \|\mathbf{a}_R\|]^T\}$. De même pour $\mathbf{\Lambda}_{\mathbf{B}}$ et $\mathbf{\Lambda}_{\mathbf{C}}$. Cette approche que nous nommerons "Algorithme1" a le schéma présenté dans la table 4.1.

Algorithme 4.1 Résumé de l'algorithme 1 basé sur le gradient projeté

Initialisation : $(\mathbf{A}^{(0)}, \mathbf{B}^{(0)}, \mathbf{C}^{(0)})$ point initial arbitrairement choisi, $\epsilon > 0$, $\mathbf{\Lambda}^{(0)} = \mathbf{I}$.
$k := 0$
Normaliser les colonnes des matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$
Tant que critère d'arrêt non satisfait,

1. Direction de descente : $\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)})$
2. Recherche d'un pas $\rho^{(k)}$ tel que :
   $\Upsilon(\mathbf{A}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{A}}^{(k)}, \mathbf{B}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{B}}^{(k)}, \mathbf{C}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{C}}^{(k)}) < \Upsilon(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)})$
3. Mise à jour :
   — $\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{A}}^{(k)}$,
   — $\mathbf{B}^{(k+1)} = \mathbf{B}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{B}}^{(k)}$,
   — $\mathbf{C}^{(k+1)} = \mathbf{C}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{C}}^{(k)}$,
   — $k = k + 1$,
   — Normaliser les nouveaux $\mathbf{A}^{(k)}$, $\mathbf{B}^{(k)}$ et $\mathbf{C}^{(k)}$,
   — Mettre à jour le $\mathbf{\Lambda}$: $\mathbf{\Lambda}^{(k+1)} = \mathbf{\Lambda}^{(k)}\mathbf{\Lambda}_{\mathbf{A}}\mathbf{\Lambda}_{\mathbf{B}}\mathbf{\Lambda}_{\mathbf{C}}$,

Retourner $\mathbf{A}^{(k)}$, $\mathbf{B}^{(k)}$ et $\mathbf{C}^{(k)}$.

4.2.3 Facteur d'échelle optimal

Dans cette deuxième solution, nous appliquons l'algorithme du gradient projeté sur le problème 4.1, mais cette fois en calculant le $\mathbf{\Lambda}$ optimal. Comme présenté dans la section 2.4, le fait d'isoler le facteur d'échelle et de calculer sa valeur optimale, permet de réduire les indéterminations d'échelle à des colonnes de norme unitaire. Trouver la valeur optimale de $\mathbf{\Lambda}$ revient à résoudre le système des équations (2.23), de tel sorte à avoir :

$$\mathbf{\Lambda} = \mathbf{G}^{-1}\mathbf{f}, \tag{4.12}$$

Le principe de ce deuxième algorithme, que nous nommerons "Algorithme 2", consiste à trouver à chaque itération un point admissible que nous projetterons après sur l'ensemble des contraintes. En d'autre termes, à chaque itération, et après avoir trouvé les trois matrices facteurs $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ qui minimisent la fonction de coût 4.1, nous calculons la valeur optimale du facteur d'échelle $\mathbf{\Lambda}$, puis nous normalisons les trois matrices facteurs. Cette deuxième approche que nous nommerons "Algorithme2" a le schéma présenté dans la table 4.2.

4.2.4 Simulations

Dans cette section, nous allons présenter les simulations réalisées, pour illustrer le comportement et le fonctionnement des trois algorithmes présentés antérieurement :

— Algorithme de Gradient Projeté avec $\mathbf{\Lambda} = \mathbf{I}$, que nous l'appelons Algorithme 0,
— Algorithme 1 avec $\mathbf{\Lambda}$ pris comme produit des normes des trois matrices facteurs,

Algorithme 4.2 Résumé de l'algorithme 2 basé sur le gradient projeté

Initialisation : $(\mathbf{A}^{(0)}, \mathbf{B}^{(0)}, \mathbf{C}^{(0)})$ point initial arbitrairement choisi, $\epsilon > 0$, $\mathbf{\Lambda}^{(0)} = \mathbf{I}$.
$k := 0$
Normaliser les colonnes des matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$
Calculer le $\mathbf{\Lambda}^{(0)}$: $\mathbf{\Lambda}^{(0)} = (\mathbf{G}^{(0)})^{-1}\mathbf{f}^{(0)}$,
Tant que critère d'arrêt non satisfait,

1. Direction de descente : $\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)})$
2. Recherche d'un pas $\rho^{(k)}$ tel que :
   $\Upsilon(\mathbf{A}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{A}}^{(k)}, \mathbf{B}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{B}}^{(k)}, \mathbf{C}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{C}}^{(k)}) < \Upsilon(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)})$
3. Mise à jour :
   — $\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{A}}^{(k)}$,
   — $\mathbf{B}^{(k+1)} = \mathbf{B}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{B}}^{(k)}$,
   — $\mathbf{C}^{(k+1)} = \mathbf{C}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{C}}^{(k)}$,
   — $k = k + 1$,
   — Normaliser les nouveaux $\mathbf{A}^{(k)}$, $\mathbf{B}^{(k)}$ et $\mathbf{C}^{(k)}$,
   — Mettre à jour le $\mathbf{\Lambda}$: $\mathbf{\Lambda}^{(k)} = (\mathbf{G}^{(k)})^{-1}\mathbf{f}^{(k)}$,

Retourner $\mathbf{A}^{(k)}$, $\mathbf{B}^{(k)}$ et $\mathbf{C}^{(k)}$.

— Algorithme 2 avec $\mathbf{\Lambda}$ optimal.

Dans cette première simulation, nous considérons un tenseur $\mathcal{T}_2$ de rang $R = 4$ et de taille $3 \times 3 \times 7$. et nous comparons la convergence des trois algorithmes par rapport à un seuil d'itérations précis. La figure 4.1 présente la convergence de ces algorithmes par rapport au seuil qui est $Seuil_{Iter} = 200$ itérations. En comparant les trois courbes obtenues dans cette figure, on remarque bien que pour 200 itérations, l'algorithme 0 tend vers une erreur de reconstruction d'un peu près 10^{-4}, l'algorithme 1 tend vers 10^{-5}, alors que l'algorithme 2 atteint une erreur de 10^{-6}. Cela nous laisse conclure que l'algorithme 2 avec le $\mathbf{\Lambda}$ optimal est plus performant que les deux autres algorithmes.

Les simulations faites jusqu'à maintenant étaient sur des données non bruitées. Considérons maintenant un tenseur $\mathcal{T}_3$ de dimensions $I = 4$, $J = 7$ et $K = 10$. Les matrices facteurs sont initialisées aléatoirement avec 4 colonnes. Les résultats sont obtenus après 100 itérations de Monte Carlo. à chaque itération, et pour chaque valeur de SNR, un nouveau bruit est ajouté à notre tenseur $\mathcal{T}_3$.

Dans la figure 4.2, une comparaison entre les 3 algorithmes en terme de l'erreur d'estimation des trois matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ est faite. Nous avons tracé l'erreur d'estimation des matrices facteurs en fonction de valeur de SNR. À partir de la figures 4.2, il est bien clair que la performance de l'algorithme 0 est très faible en comparaison avec l'algorithme 1 et 2. Cela aussi confirme l'idée que les algorithmes d'isolement de la matrice d'échelle

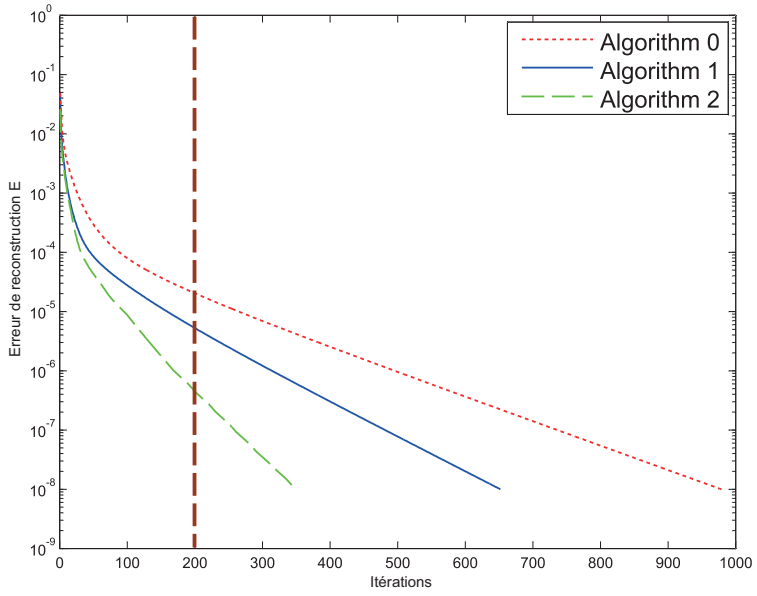


FIGURE 4.1 – Erreur de la reconstruction du tenseur $\mathcal{T}_2$ en fonction du nombre d'itérations.

sont attractifs.

4.3 Algorithmes avec contrainte d'inégalité

4.3.1 Introduction

L'isolation de la matrice d'échelle dans la décomposition CP a plusieurs implications, parmi lesquelles l'existance et l'unicité de la décomposition CP. Comme présenté dans le chapitre 2, on a bien montré que le fait d'imposer la contrainte (4.13), permet de garantir l'existence de la décomposition CP.

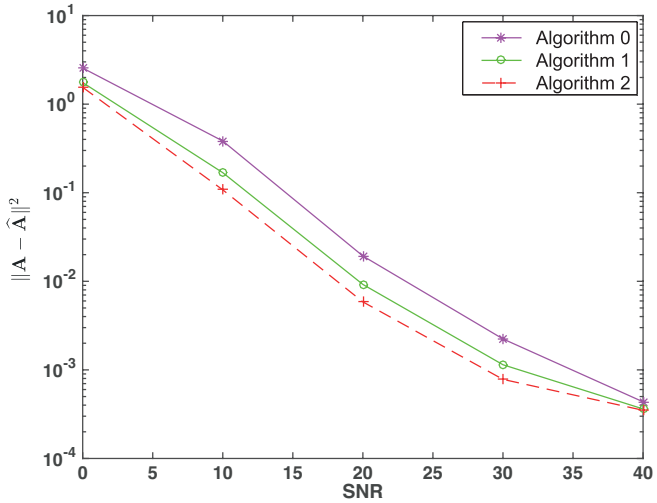
$$\mu_{\mathbf{A}}\mu_{\mathbf{B}}\mu_{\mathbf{C}} \leq \frac{1}{R-1} \tag{4.13}$$

Malgré l'utilisation de cette contrainte, la minimisation de la fonction 3.1 déclenche d'autres problèmes, comme l'absence de la différentiabilité de (4.13).

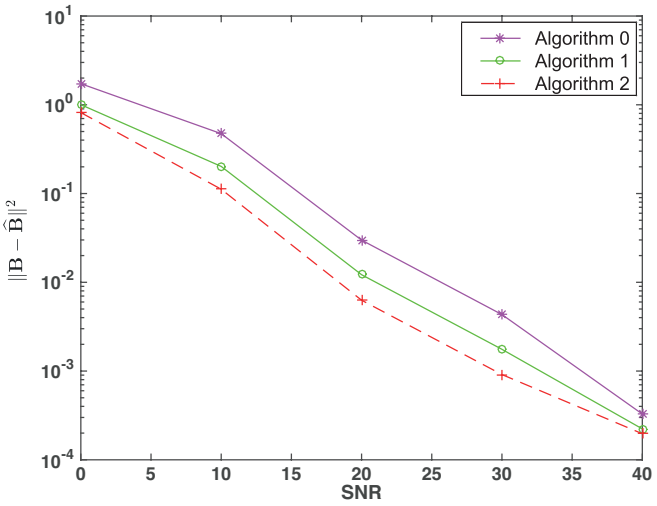
La contrainte (4.13) nécessite une certaine attention car elle implique des opérateurs max, qui ne sont pas différentiables. Pour cette raison, nous proposons d'utiliser le fait que la norme L_∞ peut être bornée par les normes L_{2p}, et approximée pour des grandes valeurs de p :

$$\|\mathbf{z}\|_\infty \overset{def}{=} \max_k z_k \leq \|\mathbf{z}\|_{2p} \overset{def}{=} (\sum_k z_k^{2p})^{1/2p} \ , \ \forall p \geq 1, \tag{4.14}$$

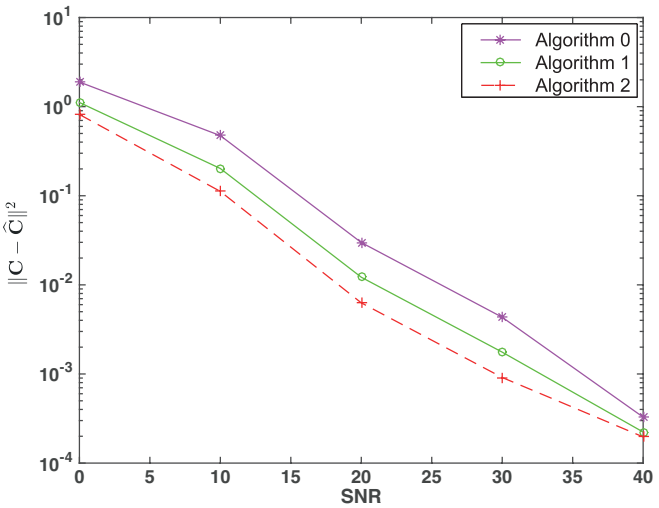
pour $z_k \in \mathbb{R}^+$. En appliquant cette inégalité à $z_k \equiv \|a_p^k a_q\|$, on peut relier les cohérences ci-dessus par une quantité différentiable, afin qu'une autre condition suffisante (un peu



(a) Matrice $\mathbf{A}$



(b) Matrice $\mathbf{B}$



(c) Matrice $\mathbf{C}$

FIGURE 4.2 – Erreur d'estimation des trois matrices pour le tenseur $\mathcal{T}_3$ en fonction de SNR.

plus contraignante) puisse être obtenue. Plus précisément :

$$\mu_A \overset{def}{=} \sup_{p\neq q}|a_p^H a_q| \leq \mu(\mathbf{A},p) \overset{def}{=} \left(\sum_{p<q}\|a_p^k a_q\|^{2p}\right)^{1/2p}. \tag{4.15}$$

Nous appelons ensuite $\mathcal{C}_p$ la contrainte obtenue en remplaçant les opérateurs max par les normes L_{2p} dans la contrainte (4.13) :

$$\mathcal{C}_p = (1-R) + \mu(\mathbf{A},p)^{-1}\mu(\mathbf{B},p)^{-1}\mu(\mathbf{C},p)^{-1} \geq 0. \tag{4.16}$$

Il est clair que si (4.16) est satisfait, alors (4.13) l'est aussi.

Notre objectif sera alors d'optimiser la décomposition tensorielle CP sous deux contraintes, une d'égalité et une autre d'inégalité. Ce qui veut dire, résoudre le problème d'optimisation suivant :

$$\begin{aligned} \textbf{min } \Upsilon(\mathbf{\Lambda},\mathbf{A},\mathbf{B},\mathbf{C}) &= \|\mathcal{T} - \sum_{r=1}^{R}\lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\|_F^2 \\ \textbf{tel que :} \quad \|\mathbf{a}_r\| &= \|\mathbf{b}_r\| = \|\mathbf{c}_r\| = 1. \\ \sum_{i=1}^{3}\mu(i,p)^{-1} &\leq R-1 \end{aligned} \tag{4.17}$$

Les algorithmes qu'on a proposé jusqu'à maintenant traitent l'optimisation d'une décomposition tensorielle sous une contrainte d'égalité, donc ils ne peuvent pas être appliqués sur le problème (4.17). Ceci donne lieu à des nouveaux algorithmes tels que les algorithmes de barrière et les algorithmes de proximaux que nous introduisons plus en détail dans les sections suivantes.

4.3.2 Formulation du problème

Dans cette section, nous allons décrire le principe des approches de pénalité qui permettent la formulation de notre problème d'optimisation (4.17) à travers des méthodes de pénalités intérieures, également appelées méthodes de points intérieurs ou méthodes de barrières. Le principe de ces méthodes réside dans la transformation d'un problème contraint en une séquence de problèmes sans contraintes, en ajoutant au coût une pénalité en cas de violation de celles-ci. Un tel sous-problème est résolu à chaque itération d'une méthode de pénalité.

L'appellation "pénalité intérieure" ou point intérieur est employée car le minimum est approché depuis l'intérieur de $\mathcal{C}$. Les méthodes de barrière s'appliquent aux problèmes dont l'ensemble admissible $\mathcal{C}$ est défini uniquement par une collection d'inégalités :

$$\begin{cases} \min\limits_{x} f(x) \\ tel\ que : g_i(x) \leq 0, \quad i = 1,...,n \\ x \in \mathcal{C} \end{cases} \tag{4.18}$$

En effet, ces méthodes utilisent des fonctions dites de barrière, définies uniquement à l'intérieur de $\mathcal{C}$. Si des contraintes sous forme d'égalités étaient introduites, l'intérieur de cet ensemble, c'est-à-dire son sous-ensemble tel qu'en chacun de ses points aucune contrainte n'est active, serait clairement vide. C'est donc le domaine de définition de la fonction de barrière qui serait vide, rendant l'utilisation de la méthode impossible. L'intérieur de l'ensemble $\mathcal{C}_I$ défini par les g_i est le suivant :

$$\mathcal{C}_I = \{x \mid g_i(x) < 0, \forall i \in \{1, ..., n\}\}$$

. La fonction de barrière, notée $B(x)$, est ajoutée au coût $f(x)$ [Nesterov et Nemirovskii, 1994, Roos *et al.*, 1997] ; elle est continue sur $\mathcal{C}_I$ et sa valeur tend vers l'infini lorsque la frontière de $\mathcal{C}$ est approchée par l'intérieur, c'est-à-dire lorsque l'un des $g_i(x)$ approche zéro par les valeurs négatives. Une itération de la méthode consiste ensuite à minimiser la fonction $f(x) + \eta B(x)$ (où η est un paramètre réel strictement positif) à l'aide d'algorithmes de minimisation directe, comme par exemple ceux décrits au section 3.4. Une fonction $B(x)$ et un η convenablement choisis assurent que cette minimisation ne puisse nous mener à des points situés hors de $\mathcal{C}_I$. La suite du processus consiste à réduire progressivement η afin de diminuer la pénalité et autoriser les algorithmes de minimisation directe à se rapprocher peu à peu de la frontière de $\mathcal{C}$.

Le tableau 4.1 et la figure 4.3 présentent quelques exemples de fonctions $B(x)$ qui sont des barrières associées au domaine $\mathcal{C}_B = [0, +\infty]$. Les barrières logarithmique et inverse [Fiacco et McCormick, 1990] sont des barrières strictes. En effet, elles tendent vers l'infini en $x = 0$. à l'inverse, les barrières entropique et hyperbolique sont définies en 0 et valent 0.

Nom	Barrière $B(x)$
Barrière logarithmique	$-\log(x)$
Barrière inverse	$\frac{1}{x}$
Barrière entropique	$x \log(x)$
Barrière hyperbolique	$-\sqrt{x}$

TABLE 4.1 – Exemples des fonctions barrières.

Il est important de noter que, si tous les g_i sont convexes, ces deux fonctions de barrière le sont également.
Remarquons qu'une nécessité, pour pouvoir appliquer une telle méthode, est de disposer d'un point initial situé à l'intérieur de $\mathcal{C}$. La méthode de barrière est définie en introduisant la séquence de paramètres $\eta^k; k = 0, 1, ...$ avec $0 < \eta^{(k+1)} < \eta^k$ et $\eta^k \to 0$ lorsque $k \to \infty$.
Le fait que $B(x)$ ne soit défini que dans l'intérieur $\mathcal{C}_I$ et tende vers l'infini au fur et à mesure que l'on se rapproche des bords de $\mathcal{C}$ assure que, même avec un algorithme de minimisation directe, le point obtenu à chaque itération appartient lui aussi à $\mathcal{C}_I$. Le fait

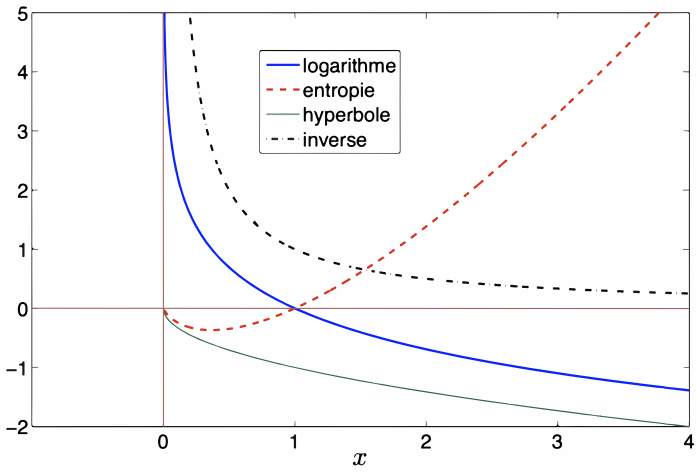


FIGURE 4.3 – Exemples des fonctions barrières.

que t^k tend vers zéro implique que le terme $f(x) + \eta B(x)$ tend vers $f(x)$ lorsque k tend vers l'infini.
Le comportement de la méthode dépend fortement du choix du paramètre initial η^0 et du facteur, disons t, satisfaisant $0 < t < 2$, utilisé pour décroître η^k à chaque itération par la formule $\eta^{(k+1)} = t\eta^{(k)}$. Il n'existe pas de règle universelle permettant d'obtenir un bon choix de η^0 et t. Cela dépend fortement du problème à résoudre, ainsi que du point initial x^0. L'utilisateur d'une méthode de barrière sera souvent condamné à exécuter la méthode plusieurs fois avec différentes valeurs de ces paramètres jusqu'à obtenir une convergence satisfaisante.
Les faits suivants peuvent néanmoins être relevés : si η^k est trop petit, et à plus forte raison si x^k est proche des bords du domaine, le terme de barrière peut se révéler trop faible, ne parvenant pas à empêcher une sortie de $\mathcal{C}$ (il faut donc prendre garde, durant l'exécution, et vérifier qu'on ne sort pas de cet ensemble, et, si cela est tout de même le cas, recommencer avec un η^0 ou un t plus grand). Dans le cas contraire, si η^k est trop grand, l'algorithme de minimisation directe ne pourra s'approcher suffisamment des bords du domaine, conduisant à une convergence globale lente. Ces paramètre doivent donc être soigneusement choisis selon le problème et le point initial.

Revenons maintenant à notre problème d'optimisation (4.17). L'idée est de modifier la fonction objectif Υ (4.17) à l'intérieur de l'ensemble des contraintes en ajoutant la fonction $log(\mathcal{C}_p)$, tel que $\mathcal{C}_p$ représente la contrainte sur la cohérence, qui a la forme suivante :

$$\mathcal{C}_p = (1 - R) + \mu(\mathbf{A}, p)^{-1}\mu(\mathbf{B}, p)^{-1}\mu(\mathbf{C}, p)^{-1} \geq 0. \tag{4.19}$$

La fonction du coût du problème (4.17) prendra alors la forme :

$$\Upsilon_b(\mathbf{\Lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C}) = \Upsilon - \eta \; log(\mathcal{C}_p). \tag{4.20}$$

Dans le cas présent, nous devons résoudre notre nouveau problème d'optimisation qui a la forme suivante :

$$\boxed{\begin{array}{l} \textbf{min } \Upsilon_b(\mathbf{\Lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C}) = \Upsilon - \eta \; log(\mathcal{C}_p) \\ \textbf{tel que :} \quad \|\mathbf{a}_r\| = \|\mathbf{b}_r\| = \|\mathbf{c}_r\| = 1. \end{array}} \tag{4.21}$$

4.3.3 Méthode barrière

Dans la littérature, les problèmes d'optimisation avec des contraintes d'inégalité sont résolus en utilisant la méthode barrière avec l'algorithme de Newton pour la recherche du point optimal de chaque itération. Dans cette partie, nous proposons comme solution au problème (4.21) la méthode Barrière avec l'algorithme du Gradient Projeté au lieu de l'algorithme Newton, dans l'objectif de diminuer la complexité des calculs.
En vue de la fonction de coût de notre nouveau problème d'optimisation, il est nécessaire de calculer le gradient de la nouvelle fonction avec Barrière.

4.3.3.1 Cas reél

On veut calculer le gradient de la fonction de coût (4.21) dans le cas réel, c'est à dire, dans le cas où les trois matrices facteurs $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ appartiennent à $\mathbb{R}^{I\times R}$, $\mathbb{R}^{J\times R}$ et $\mathbb{R}^{K\times R}$ respectivement. Calculons le gradient de Υ_b par rapport aux éléments $a_{\alpha\beta}$, $b_{\alpha\beta}$ et $c_{\alpha\beta}$ de $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ respectivement. On a :

$$\frac{\partial \Upsilon_b}{\partial a_{\alpha\beta}} = \frac{\partial \Upsilon}{\partial a_{\alpha\beta}} - \eta \frac{\partial log(\mathcal{C}_p)}{\partial a_{\alpha\beta}} \tag{4.22}$$

$$\frac{\partial \Upsilon_b}{\partial b_{\alpha\beta}} = \frac{\partial \Upsilon}{\partial b_{\alpha\beta}} - \eta \frac{\partial log(\mathcal{C}_p)}{\partial b_{\alpha\beta}} \tag{4.23}$$

$$\frac{\partial \Upsilon_b}{\partial c_{\alpha\beta}} = \frac{\partial \Upsilon}{\partial c_{\alpha\beta}} - \eta \frac{\partial log(\mathcal{C}_p)}{\partial c_{\alpha\beta}}, \tag{4.24}$$

Calculons d'abord les composantes du gradient de Υ, ensuite ceux du gradient de $log(\mathcal{C}_p)$. Dans la sous-section 4.2.1, nous avons calculé les composantes du gradient de Υ par rapport aux éléments $a_{\alpha\beta}$, $b_{\alpha\beta}$ et $c_{\alpha\beta}$ (4.5, 4.6, 4.7), ainsi que par rapport aux matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$ (4.8, 4.9, 4.10). Ce qui nous reste à calculer dans cette section sont les

composantes du gradient de $log(\mathcal{C}_p)$. Celles-ci seront déterminés par les formules suivantes :

$$\frac{\partial log(\mathcal{C}_p)}{\partial \mathbf{a}_\beta} = \frac{2\Sigma_{\beta\neq q}(\mathbf{a}_q\mathbf{a}_q^T)\mathbf{a}_\beta}{\mathcal{C}_p} \tag{4.25}$$

$$\frac{\partial log(\mathcal{C}_p)}{\partial \mathbf{b}_\beta} = \frac{2\Sigma_{\beta\neq m}(\mathbf{b}_m\mathbf{b}_m^T)\mathbf{b}_\beta}{\mathcal{C}_p} \tag{4.26}$$

$$\frac{\partial log(\mathcal{C}_p)}{\partial \mathbf{c}_\beta} = \frac{2\Sigma_{\beta\neq s}(\mathbf{c}_s\mathbf{c}_s^T)\mathbf{c}_\beta}{\mathcal{C}_p}, \tag{4.27}$$

4.3.3.2 *Cas complexe*

Dans le cas complexe, le calcul des composantes du gradient de Υ_b se fait par rapport à des matrices complexes. Cela nous pousse à réécrire les deux parties de la fonction (4.20) sous l'une des formes présentées dans [Comon, 1986, Hjorungnes et Gesbert, 2007]. Commençons d'abord par la première partie, qui est Υ. On a :

$$\Upsilon(\mathbf{\Lambda},\mathbf{A},\mathbf{B},\mathbf{C}) = \sum_{ijk}(T_{ijk} - \Sigma_{r=1}^{R}\lambda_r a_{ir}b_{jr}c_{kr})(T^*_{ijk} - \Sigma_{q=1}^{R}\lambda^*_q a^*_{iq}b^*_{jq}c^*_{kq}) \tag{4.28}$$

$$\begin{aligned}\Upsilon(\mathbf{\Lambda},\mathbf{A},\mathbf{B},\mathbf{C}) &= \|T_{ijk}\|^2 + \sum_{ijk}\sum_{r=1}^{R}\sum_{q=1}^{R}\lambda_r\lambda^*_q a_{ir}a^*_{iq}b_{jr}b^*_{jq}c_{kr}c^*_{kq} \\ &= -2\sum_{ijk}\Re\{T^*_{ijk}\sum_{r=1}^{R}a_{ir}b_{jr}c_{kr}\}\end{aligned}$$

Posons maintenant $M_{rq} = \sum_{jk}\lambda_r\lambda^*_q b_{jr}b^*_{jq}c_{kr}c^*_{kq}$ et $N_{ir} = \sum_{jk}T_{ijk}\lambda^*_r b^*_{jr}c^*_{kr}$. La nouvelle espression de Υ sera :

$$\Upsilon(\mathbf{\Lambda},\mathbf{A},\mathbf{B},\mathbf{C}) = \|T_{ijk}\|^2 + \sum_{i}\sum_{r=1}^{R}\sum_{q=1}^{R}a_{ir}M_{rq}a^*_{iq} - \sum_i(\sum_{q=1}^{R}N_{iq}a^*_{iq} + \sum_{r=1}^{R}a_{ir}N^*_{ir}).$$

Le gradient de Υ par rapport à $\mathbf{A}$ sera :

$$\frac{\partial\Upsilon}{\partial\mathbf{A}} = 2\mathbf{AM} - 2\mathbf{N} \tag{4.29}$$

De même, nous allons réecrire la deuxième partie de l'equation (4.20), c.-à-d. $\log(\mathcal{C}_p)$, pour pouvoir calculer son gradient par rapport à $\mathbf{A}$.

$$\mathcal{C}_p = (1-R) + (\sum_{j<k}|\mathbf{a}_j^H\mathbf{a}_k|^2)^{-1}(\sum_{l<m}|\mathbf{b}_l^H\mathbf{b}_m|^2)^{-1}(\sum_{n<q}|\mathbf{c}_n^H\mathbf{c}_q|^2)^{-1} \geq 0. \tag{4.30}$$

D'où le gradient par rapport à chaque colonne $\mathbf{a}_l$ de $\mathbf{A}$:

$$\frac{\partial\mathcal{C}_p}{\partial\mathbf{a}_l} = -2\sum_{q\neq l}(\mathbf{a}_q\mathbf{a}_q^H)\mathbf{a}_l = -2(\mathbf{A}\mathbf{A}^H - \|\mathbf{a}_l\|^2\mathbf{I})\mathbf{a}_l,$$

et celui de $\log(\mathcal{C}_p)$ par rapport à la matrice $\mathbf{A}$, si ses colonnes sont de norme unitaire :

$$\frac{\partial log(\mathcal{C}_p)}{\partial \mathbf{A}} = -\frac{2}{\mathcal{C}_p}(\mathbf{A}\mathbf{A}^H - \mathbf{I})\mathbf{A} \tag{4.31}$$

Le gradient de la fonction (4.20) par rapport à $\mathbf{A}$ est finalement :

$$\frac{\partial \Upsilon_b}{\partial \mathbf{A}} = 2\mathbf{A}\mathbf{M} - 2\mathbf{N} - \frac{2\eta}{\mathcal{C}_p}(\mathbf{A}\mathbf{A}^H - \mathbf{I})\mathbf{A} \tag{4.32}$$

Le calcul du gradient de Υ_b par rapport à $\mathbf{B}$ et $\mathbf{C}$ est similaire, en tenant compte du fait que les matrices $\mathbf{M}$ et $\mathbf{N}$ sont différentes.

Le principe de cet algorithme d'optimisation pour le problème (4.21) est donné dans la table 4.3. Cet algorithme est basé sur l'algorithme de point intérieur et Gradient projeté, et nous le nommerons "Algorithme Barrière-GP".

Algorithme 4.3 Résumé de l'algorithme barrière utilisant le gradient projeté

Initialisation : $(\mathbf{A}^{(0)}, \mathbf{B}^{(0)}, \mathbf{C}^{(0)})$ point initial arbitrairement choisi, $\epsilon > 0$, $\eta^0 > 0$ et $t = 1.5$ paramètre de la barrière.
Normaliser les colonnes des matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$
Calculer le $\mathbf{\Lambda}^{(0)}$: $\mathbf{\Lambda}^{(0)} = (\mathbf{G}^{(0)})^{-1}\mathbf{f}^{(0)}$,
Test sur la contrainte $\mathcal{C}_p$:

1. Si $\mathcal{C}_p \leq 0$, retourner à l'initialisation
2. Sinon : Tant que critère d'arrêt non satisfait, faire :
   - Direction de descente : $\mathbf{d}^{(k)} = -\nabla\Upsilon_b(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)})$
   - Recherche d'un pas $\rho^{(k)}$ tel que :
     $\Upsilon_b(\mathbf{A}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{A}}^{(k)}, \mathbf{B}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{B}}^{(k)}, \mathbf{C}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{C}}^{(k)}) < \Upsilon_b(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)})$
   - Mise à jour :
     - $\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{A}}^{(k)}$,
     - $\mathbf{B}^{(k+1)} = \mathbf{B}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{B}}^{(k)}$,
     - $\mathbf{C}^{(k+1)} = \mathbf{C}^{(k)} + \rho^{(k)}\mathbf{d}_{\mathbf{C}}^{(k)}$,
     - $k = k + 1$,
     - Normaliser les nouveaux $\mathbf{A}^{(k)}$, $\mathbf{B}^{(k)}$ et $\mathbf{C}^{(k)}$,
     - mettre à jour le $\mathbf{\Lambda}$: $\mathbf{\Lambda}^{(k)} = (\mathbf{G}^{(k)})^{-1}\mathbf{f}^{(k)}$,
   - Mettre à jour le $\eta^{(k+1)}$: $\eta^{(k+1)} = \eta^{(k)}/t$

Retourner $\mathbf{A}^{(k)}$, $\mathbf{B}^{(k)}$ et $\mathbf{C}^{(k)}$.

4.3.4 Méthodes proximales

4.3.4.1 Introduction

Nous nous intéresserons dans ce qui suit à une classe d'algorithmes, appelés algorithmes proximaux. Tout comme la méthode de Newton qui constitue un outil standard pour résoudre des problèmes de minimisation lisses, sans contraintes et de taille modeste, les algorithmes proximaux peuvent être considérés comme un outil analogue pour les versions non lisses, contraintes, à grande échelle ou distribuées de ces problèmes. Ils sont applicables de manière très générale, mais ils sont particulièrement bien adaptés aux problèmes d'intérêt récent et généralisé impliquant des ensembles de données de grande taille ou à haute dimension. Les méthodes proximales se situent à un niveau d'abstraction plus élevé que les algorithmes d'optimisation classiques tels que la méthode de Newton. Dans ces derniers, les opérations de base sont de bas niveau, consistant en des opérations d'algèbre linéaire et le calcul des gradients et des hessiens. Dans les algorithmes proximaux, l'opération de base consiste à évaluer un opérateur proximal d'une fonction, ce qui implique la résolution d'un petit sous-problème d'optimisation. Ces sous-problèmes peuvent être résolus avec des méthodes standard, mais peuvent être évaluées efficacement, c'est-à-dire que leur opérateur proximal admet souvent des solutions de forme analytiquement fermée, ou peuvent être résolus très rapidement avec des méthodes spécialisées simples exacte ou inexacte. De même, les algorithmes proximaux ont un certain nombre d'interprétations intéressantes et sont liés à de nombreux sujets différents en optimisation et en mathématiques appliquées. Une description détaillée de ces méthodes peut être trouvée dans [Parikh et Boyd, 2014, Combettes et Pesquet, 2011, Combettes et Pesquet, 2008].

4.3.4.2 Operateur proximal

L'opérateur proximal en un point $x \in \mathcal{H}$ a été défini dans [Moreau, 1965] pour une fonction $f \in \Gamma_0(\mathcal{H})$ [1] comme étant l'unique minimiseur de la fonction $f + \frac{1}{2}\|. - \mathbf{x}\|^2$. On a ainsi, pour $\gamma > 0$:

$$\begin{aligned} &\mathbf{prox}_{\gamma f} : \mathcal{H} \to \mathcal{H} \\ &\qquad \mathbf{x} \to \mathbf{prox}_{\gamma f}(\mathbf{x}) := \underset{\mathbf{y}\in\mathcal{H}}{\mathbf{argmin}}\ f(\mathbf{y}) + \frac{1}{2\gamma}\|\mathbf{y}-\mathbf{x}\|^2. \end{aligned} \tag{4.33}$$

La définition ci-dessus peut être étendue aux fonctions non convexes [Attouch et Bolte, 2009]. De plus, elle peut être rendue plus générale en utilisant une norme pondérée au lieu de la norme euclidienne [Becker et Fadili, 2012, Combettes et Vũ, 2014, Chouzenoux *et al.*, 2014].

Cet opérateur a été largement étudié pour des fonctions convexes usuelles du traitement du signal, par exemple dans [Chaux *et al.*, 2007, Combettes et Pesquet, 2008, Combettes et Pesquet, 2011, Combettes et Wajs, 2005, Parikh et Boyd, 2014]. Cette définition a été généralisée pour des fonctions $f : \mathcal{H} \to]-\infty, +\infty]$ non nécessairement

1. est la classe des fonctions convexes semi-continues inférieures de $\mathbb{R}^N$ à $]-\infty, +\infty]$ tel que dom $f \neq \emptyset$.

convexes dans le cas de normes pondérées dans [Hiriart-Urruty et Lemaréchal, 1993, Attouch *et al.*, 2013, Chouzenoux *et al.*, 2014, Combettes et Vũ, 2013]). Il y a encore plusieurs exemples où l'algorithme proximal a été étendu à l'optimisation non convexe. Parmi les exemples, on peut citer le splitting proximal non convexe inexact (NIPS) [Sra, 2012], l'algorithme de gradient proximal inertiel pour l'optimisation non convexe (IPiano) [Ochs *et al.*, 2014], le traitement unifié du gradient accéléré (UAG) [Ghadimi et Lan, 2016], le retrait itératif général et le seuillage (GIST) [Gong *et al.*, 2013], l'algorithme inertiel avant-arrière (IFB) [Boţ *et al.*, 2016] et l'algorithme de gradient proximal accéléré non monotone (nmAPG) [Li et Lin, 2015]. En particulier, NIPS [Sra, 2012], IPiano [Ochs *et al.*, 2014] et UAG [Ghadimi et Lan, 2016] permettent à f d'être non convexe, tandis que g doit toujours être convexe. GIST [Gong *et al.*, 2013], IFB [Boţ *et al.*, 2016] et nmAPG [Li et Lin, 2015] suppriment encore cette restriction et permettent à g d'être non convexe. Il est souhaitable que l'étape proximale ait une solution de forme fermée. Cela est vrai pour de nombreux régularisateurs convexes tels que le régulateur de lasso [Tibshirani, 1996], le régularisateur de lasso à structure arborescente [Liu et Ye, 2010, Jenatton *et al.*, 2011] et le régularisateur de lasso à groupes épars [Jacob *et al.*, 2009]. Cependant, lorsque g est non convexe, une solution de forme fermée n'existe que lorsque g est simple (par exemple, le régularisateur de lasso non convexe [Gong *et al.*, 2013]), mais pas pour les cas plus généraux (par exemple, le régularisateur de lasso non convexe à structure arborescente [Zhong et Kwok, 2014]).

4.3.4.3 *Opérateur de projection*

Définition : Soit E un sous-ensemble de $\mathcal{H}$. La projection de $x \in \mathcal{H}$ sur E est définie par :

$$p_E(\mathbf{x}) = \underset{\mathbf{y}\in E}{\mathbf{argmin}} \|\mathbf{y} - \mathbf{x}\| \tag{4.34}$$

En d'autres termes, un élément de $p_E(\mathbf{x})$ est un point de E tel que sa distance à $\mathbf{x}$ soit minimale.
Remarquons que l'opérateur proximal peut être vu comme une généralisation de l'opérateur de projection au sens où, pour tout sous-ensemble E de $\mathcal{H}$,

$$\begin{aligned} \mathbf{prox}_{\iota_E}(\mathbf{x}) &= \underset{\mathbf{y}\in\mathbb{R}^N}{\mathbf{argmin}}\ \iota_E(\mathbf{y}) + \frac{1}{2\gamma}\|\mathbf{y} - \mathbf{x}\|^2 \\ &= \underset{\mathbf{y}\in E}{\mathbf{argmin}}\ \frac{1}{2\gamma}\|\mathbf{y} - \mathbf{x}\|^2 \\ &= p_E(\mathbf{x}), \end{aligned} \tag{4.35}$$

où ι_E désigne la fonction indicatrice de E.

Proposition *(Bauschke et al., 2011b), Thm 3.14* : Si C est un sous-ensemble convexe fermé non vide de $\mathcal{H}$, alors la projection $p_E(\mathbf{x})$ de $\mathbf{x} \in \mathcal{H}$ sur C existe et elle est unique.

4.3.4.4 Inégalité de Kurdyka-Lojasiewicz

L'inégalité de Kurdyka-Lojasiewicz (KL) apparaît ainsi comme un outil fondamental en optimisation non convexe qui permet de démontrer la convergence de suites minimisant des fonctions non nécessairement convexes. Cette inégalité a été proposée en premier dans [Lojasiewicz, 1963], et il a été démontré qu'elle est satisfaite par toute fonction analytique réelle :

Propriété *(Inégalité de Kurdyka-Lojasiewicz) :* Soient $f : \mathbb{R}^N \to \mathbb{R}$ une fonction analytique réelle, et $\hat{\mathbf{x}}$ un point critique de cette fonction. Il existe $\theta \in [\frac{1}{2}, 1[$ et $\kappa \in]0, +\infty[$ tels que f satisfait l'inégalité de Lojasiewicz dans un voisinage $\mathcal{V}(\hat{\mathbf{x}})$ de $\hat{\mathbf{x}}$, c.-à-d. :

$$(\forall \mathbf{x} \in \mathcal{V}(\hat{\mathbf{x}})) \quad |f(\mathbf{x}) - f(\hat{\mathbf{x}})|^\theta \leq \kappa \|\nabla f(\mathbf{x})\|. \tag{4.36}$$

Cette inégalité a été utilisée pour démontrer la convergence d'algorithmes de descente de gradient pour la minimisation de fonctions différentiables non nécessairement convexe [Absil *et al.*, 2005], elle permet d'assurer que la courbe de gradient générée par ce type d'algorithme est de longueur finie [Absil *et al.*, 2005] (voir aussi [Bolte *et al.*, 2010]).

La proposition suivante nous permet aussi d'assurer l'existence de minimiseurs d'une somme de fonctions de $\Gamma_0(\mathcal{H})$ grâce à la coercivité.

Proposition *(Bauschke et al., 2011b) , Cor. 11.15 :* Soient f et g des fonctions de $\Gamma_0(\mathcal{H})$. Supposons que $dom\ f \cap domg \neq \emptyset$, que f est coercive et que g est bornée inférieurement. La somme $f + g$ est alors coercive et admet un minimiseur sur $\Gamma_0(\mathcal{H})$.

4.3.4.5 Problème général

Nous allons maintenant nous intéresser au cas où la fonction à minimiser est éclatée en une somme de deux fonctions, qui a la forme suivante :

$$\boxed{trouver\ \hat{\mathbf{x}} \in \underset{\mathbf{x}\in\mathcal{H}}{\mathbf{Argmin}} \left\{ f(\mathbf{x}) = h(\mathbf{x}) + g(\mathbf{x}) \right\}} \tag{4.37}$$

où h est différentiable (non nécessairement convexe) et g peut être à la fois non convexe et non différentiable.

Parmi les exemples de problèmes (4.37), on peut citer l'apprentissage parcimonieux et de faible rang avec des régularisateurs non convexes, par exemple la norme-l_p [Foucart et Lai, 2009], la pénalité Capped-l_1 [Zhang, 2010], la pénalité de somme logarithmique [Candes *et al.*, 2008], la pénalité concave minimale [Zhang *et al.*, 2010], la pénalité gémanique [Geman et Yang, 1995], la déviation absolue à coupure lisse [Fan et Li, 2001] et la norme Schatten-p [Mohan et Fazel, 2012].

Les méthodes les plus courantes pour résoudre le problème (4.37) comprennent le rétrécissement itératif général et le seuillage (GIST) [Gong *et al.*, 2013], l'inertie avant-arrière [Boţ *et al.*, 2016], iPiano [Ochs *et al.*, 2014], et le gradient proximal avec momentum [Li *et al.*, 2017], où GIST est la méthode du gradient proximal général (PG) et les trois dernières sont des extensions de PG avec momentum. La méthode PG accélérée a été étudiée dans [Ghadimi et Lan, 2016, Li et Lin, 2015, Nesterov *et al.*, 2020], où la convergence vers le point critique est garantie pour les méthodes non convexes et leurs versions accélérées sont prouvées pour les méthodes convexes.

4.3.5 Contribution 1

Pour comprendre l'idée derrière notre première méthode proposée pour la résolution de (4.21), nous commençons par revoir encore une fois la méthode du gradient projeté pour la résolution (4.37) dans le cas où $g = \delta_{\mathcal{C}}$ avec $\mathcal{C}$ étant un ensemble fermé et convexe non vide. Dans ce cas, le problème prend la forme :

$$\mathbf{min}\Big\{h(\mathbf{x}) : \mathbf{x} \in \mathcal{C}\Big\} \tag{4.38}$$

L'étape générale de mise à jour de la méthode du gradient projeté pour résoudre (4.38) prend la forme :

$$\mathbf{x}^{k+1} = p_{\mathcal{C}}(\mathbf{x}^k - \rho_k \nabla h(\mathbf{x}^k)), \tag{4.39}$$

il est facile de vérifier que l'étape de mise à jour peut également être écrite comme :

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in\mathcal{C}}{\mathbf{argmin}}\Big\{h(\mathbf{x}^k) + \langle \nabla h(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k\rangle + \frac{1}{2\rho_k}\|\mathbf{x} - \mathbf{x}^k\|^2\Big\} \tag{4.40}$$

Cela signifie que l'itération suivante correspond à la minimisation sur $\mathcal{C}$ de la somme de la linéarisation de la partie differentiable autour de l'itération actuelle plus un terme "prox" quadratique.

Pour revenir au modèle plus général (4.37), il est naturel de généraliser l'idée ci-dessus et de définir l'itération suivante comme le minimiseur de la somme de la linéarisation de h autour de $\mathbf{x}^k$, de la fonction non necessairement differentiable g et d'un terme approximatif quadratique :

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in\mathbb{E}}{\mathbf{argmin}}\Big\{h(\mathbf{x}^k) + \langle \nabla h(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k\rangle + g(\mathbf{x}) + \frac{1}{2\rho_k}\|\mathbf{x} - \mathbf{x}^k\|^2\Big\}. \tag{4.41}$$

Après une simple manipulation algébrique et l'annulation des termes constants, nous réalisons que (4.41) peut être réécrit comme :

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in\mathbb{E}}{\mathbf{argmin}}\Big\{g(\mathbf{x}) + \frac{1}{2\rho_k}\|\mathbf{x} - (\mathbf{x}^k - \rho_k \nabla h(\mathbf{x}^k))\|^2\Big\}, \tag{4.42}$$

qui, selon la définition de l'opérateur proximal, est le même que :

$$\mathbf{x}^{k+1} = \mathbf{prox}_{\rho_k g}(\mathbf{x}^k - \rho_k \nabla h(\mathbf{x}^k)), \tag{4.43}$$

La méthode ci-dessus est appelée méthode du *gradient proximal*, qui consiste en une étape de gradient suivie d'une étape proximale. Cette méthode est également appelée descente de gradient composite, ou descente de gradient généralisée. la raison de ces dénominations et qu'il résulte de plusieurs cas particuliers, lorsqu'on minimise $f = h + g$:

— $g = 0$: Descente de gradient,
— $g = p_{\mathcal{C}}$: Descente de gradient projetée,
— $h = 0$: Algorithme de minimisation proximale.

Il peut semblerait que nous ayons échangé un problème de minimisation contre un autre, néanmoins, le point clé de cette méthode est que $\mathbf{prox}_{\gamma g}(.)$ a une forme fermée pour de nombreuses fonctions importantes h. Note aussi que :

— La représentation de $\mathbf{prox}_{\gamma g}(.)$ ne dépend pas du tout de h, mais seulement de g,
— La partie différentiable h peut être compliquée, il suffit de calculer ses gradients.

Les méthodes de gradient proximal sont populaires pour résoudre divers problèmes d'optimisation avec une régularisation non différentiable. L'étape essentielle de la méthode du gradient proximal consiste à résoudre l'opérateur proximal $\mathbf{prox}_{\gamma g}(.)$. Si la fonction $g(x)$ est suffisamment simple, nous pouvons obtenir la solution de l'opérateur proximal de manière analytique. Par exemple, si $g(x) = \|x\|_1$, la solution de l'opérateur proximal peut être obtenue par un opérateur de seuillage du rétrécissement (shrinkage thresholding operator) [Beck et Teboulle, 2009b]. Si la fonction $g(x)$ est complexe de telle sorte que l'opérateur proximal correspondant n'a pas de solution analytique, un algorithme spécifique doit être conçu pour résoudre l'opérateur proximal. Par exemple, si $g(x) = \|x\|_1 + c\Sigma_{i<j} \max\{|x_i|, |x_j|\}$ tel qu'utilisé dans l'OSCAR [Zhong et Kwok, 2012] pour la régression parcimonieuse avec regroupement automatique des caractéristiques, [Zhong et Kwok, 2012] a proposé un algorithme itératif de fusion de groupes pour résoudre exactement l'opérateur proximal. Cependant, il serait coûteux de résoudre les opérateurs proximaux lorsque la fonction $g(x)$ est complexe. Encore une fois, prenez l'exemple d'OSCAR, lorsque les données sont à haute dimensionnalité (empiriquement supérieures à 1000), l'algorithme itératif de fusion de groupes deviendrait très inefficace. Encore pire, il n'y aurait pas de solveur pour résoudre exactement les opérateurs proximaux lorsque la fonction $g(x)$ est trop complexe. Par exemple, [Grave *et al.*, 2011] a proposé la norme Lasso trace pour prendre en compte la corrélation de la matrice de conception afin de stabiliser l'estimation en régression. Cependant, en raison de la complexité de la norme Lasso trace, il n'existe toujours pas de solveur pour résoudre exactement l'opérateur proximal correspondant.

Pour remédier aux problèmes ci-dessus, [Schmidt *et al.*, 2011] a d'abord proposé les méthodes de gradient proximal inexact (y compris la version de base (IPG) et la version accélérée de Nesterov (AIPG), qui résout l'opérateur proximal de manière approximative (c'est-à-dire en tolérant une erreur dans le calcul de l'opérateur proximal). Ils ont prouvé que les méthodes de gradient proximal inexactes peuvent avoir les mêmes taux de convergence que les méthodes de gradient proximal exactes, à condition que les erreurs de calcul de l'opérateur proximal diminuent à des taux appropriés. Indépendamment,

[Villa *et al.*, 2013] ont proposé l'algorithme AIPG et ont prouvé le taux de convergence correspondant. Dans l'article de [Villa *et al.*, 2013], ils ont appelé l'AIPG en référence à la méthode de séparation avant-arrière inexacte (AIFB). Ensuite, [Gu *et al.*, 2018] ont proposé récemment trois algorithmes de gradient proximal inexacts, dont la version de base et la version accélérée de Nesterov. Nous résumons ces travaux dans le tableau 4.2.

Algorithme	**Proximal**	**accéléré**	**h(x)**	**g(x)**	**Réference**
PG+APG	Exact	Oui	C	C	[Beck et Teboulle, 2009b]
APG	Exact	Oui	C+NC	C	[Ghadimi et Lan, 2016]
PG	Exact	Non	NC	NC	[Boţ *et al.*, 2016]
APG	Exact	Oui	C+NC	C+NC	[Li et Lin, 2015]
IPG+AIPG	Inexact	Oui	C	C	[Schmidt *et al.*, 2011]
AIFB	Inexact	Oui	C	C	[Villa *et al.*, 2013]
IPG+AIPG	Inexact	Oui	C+NC	C+NC	[Gu *et al.*, 2018]

TABLE 4.2 – Représentatifs des algorithmes de gradient proximal exacts et inexacts où C et NC signifient respectivement Convexe et Non Convexe.

En suivant le principe de l'IPG [Gu *et al.*, 2018], nous présentons l'algorithme IPG pour notre problème d'optimisation non convexe (4.37).

Plus précisément, l'algorithme IPG est présenté dans l'algorithme 4.4. Comme pour l'algorithme du gradient proximal exact, l'étape essentielle de l'algorithme IPG consiste à calculer un opérateur proximal inexact $\mathbf{x} \in \mathbf{prox}^{\epsilon}_{\gamma g}(\mathbf{y})$ comme suit :

$$\mathbf{x} \in \mathbf{prox}^{\epsilon}_{\gamma g}(\mathbf{y}) = \Big\{ \mathbf{z} \in \mathbb{R}^N : \frac{1}{2\gamma}\|\mathbf{z}-\mathbf{y}\|^2 + g(\mathbf{z}) \tag{4.44}$$

$$< \epsilon + \min_{x} \frac{1}{2\gamma}\|\mathbf{x}-\mathbf{y}\|^2 + g(\mathbf{x}) \Big\} \tag{4.45}$$

où ϵ indique une erreur dans le calcul de l'opérateur proximal. Comme indiqué dans [Tappenden *et al.*, 2016], il existe plusieurs méthodes pour calculer l'opérateur proximal inexact. La méthode la plus populaire est l'utilisation d'un algorithme dual primaire pour contrôler le double écart [Bach *et al.*, 2011]. Sur la base de ce double écart, nous pouvons contrôler strictement l'erreur dans le calcul de l'opérateur proximal.

Revenons maintenant à notre fonction de coût à minimiser (4.21) qui est composée d'une somme de deux termes, dont le premier concerne la fidélité des données - dénotée par Υ - et le second se réfère aux informations a priori sur les paramètres cibles, qui est dans notre cas la contrainte sur les cohérences définie au point (4.19) - dénotée et définie par $\mathcal{G} = -\eta\ log(\mathcal{C}_p)$. Cela conduit à la forme générale suivante :

$$\mathcal{F}(\mathbf{x}) = \underbrace{\Upsilon(\mathbf{x})}_{data\,fidelity} + \underbrace{\mathcal{G}(\mathbf{x})}_{constraint} \tag{4.46}$$

Algorithme 4.4 Méthode de base du gradient proximal inexact (IPG)

Entrée : m, error $\epsilon_k(k = 1, ..., m)$, stepsize $\gamma < \frac{1}{L}$.
Initialiser $\mathbf{x}_0 \in \mathbb{R}^d$.
Pour $k = 1, ..., m$, faire :
Calculer $\mathbf{x}_{k+1} \in \mathbf{prox}_{\gamma g}^{\epsilon_k}(\mathbf{x}_k - \gamma \nabla h(\mathbf{x}_k))$
Fin Pour
Retourner : $\mathbf{x}_m$.

Principalement, notre fonction objective est explicitement écrite comme suit :

$$\underset{\widehat{\mathbf{x}} \in \mathbb{R}^N}{\text{minimize}}\ \mathcal{F}(\widehat{\mathbf{x}}) = \|\mathbf{x} - \widehat{\mathbf{x}}\|_F^2 - \eta\ ln(C_p(\widehat{\mathbf{x}})) \tag{4.47}$$

Parmi les méthodes de séparation non convexes, nous avons les résultats décrits dans la section précédente de [Attouch *et al.*, 2013, Attouch et Bolte, 2009] qui introduisent une méthode générique pour les problèmes non convexes et non différentiable basée sur la théorie de Kurdyka-Lojasiewicz, ces résultats nous seront utiles par la suite quand nous traiterons particulièrement le phénomène de Marécage (Swamp). En revanche nous suivons les travaux plus récents de [Li et Lin, 2015, Nesterov, 2013] où une descente monotone de la valeur objective est imposée pour assurer la convergence.

Dans notre cas, l'hypothèse de la coercivité vérifiée à la section 2.4.2.5 avec un simple contrôle qui impose une descente monotone de la valeur objective serait suffisante pour assurer la convergence de notre algorithme. Il a été démontré dans [Li et Lin, 2015] que le problème posé (4.46) a au moins une solution et que, pour toute $\gamma \in]0, +\infty[$, ses solutions sont déterminées par la formule récursive suivante :

$$\mathbf{x}^{(k+1)} \leftarrow \underbrace{\mathbf{prox}_{\gamma^{(k)}\mathcal{G}}}_{Etape\ \ Gradient}\ \underbrace{(\mathbf{x}^{(k)} - \gamma^{(k)} \nabla \Upsilon(\mathbf{x}^{(k)}))}_{Etape\ \ Proximale} \tag{4.48}$$

Il apparaît clairement à partir de (4.48) que la méthode gradient proximal consiste à résoudre deux sous-problèmes principaux, qui peuvent être réécrits comme suit :

$$(sp_1)\quad \mathbf{y}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \gamma^{(k)} \nabla \Upsilon(\mathbf{x}^{(k)})$$
$$(sp_2)\quad \mathbf{x}^{(k+1)} \leftarrow \underset{\mathbf{x}}{\mathbf{argmin}}\ (\eta^k \mathcal{G}(\mathbf{x}) + \frac{1}{2\gamma^{(k)}} \|\mathbf{x} - \mathbf{y}^{(k+1)}\|^2).$$

Notre méthode gradient proximal est donc définie comme étant la composition d'une étape de gradient sur la fonction Υ qui est l'étape explicite, et d'une étape proximale sur la fonction $\mathcal{G}$ qui est l'étape implicite.

Nous allons maintenant décrire plus en détail chacun de ces étapes décrit ci-dessus.

Le principe général de l'approche proposée est détaillé dans les paragraphes suivants et résumé dans les Algorithme 4.5.

4.3.5.1 *Étape gradient*

Cette étape permet d'obtenir une solution approximative, en ajustant uniquement sur la fidélité des données, c.-à-d. la fonction Υ - indépendamment de la contrainte. Cette étape comprend également deux autres sous-étapes :

(i) Direction de la descente : Cette premiere sous-étape consiste à calculer la direction de descente $\mathbf{d}^{(k)}$ de Υ, en assurant une diminution convenable. Dans notre cas, nous avons proposer d'utiliser la méthode de descente du gradient afin que la minimisation soit aussi simple que possible. Il convient de noter que, bien que la méthode du gradient puisse souffrir d'une lente convergence, en particulier pour les grands ensembles de données [Lee et Waziruddin, 1970], ces inconvénients seront atténués par la prochaine étape du proximale qui ajuste l'orientation de la recherche en fonction de l'opérateur proximal de la fonction de cohérence $\mathcal{G}$. La direction est définie comme suit :

$$\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)}) = -\nabla\Upsilon(\mathbf{x}^{(k)}) \tag{4.49}$$

Où les expressions de gradient nécessaires pour déterminer la direction de la descente $\mathbf{d}^{(k)}$ sont de la forme :

$$\frac{\partial\Upsilon}{\partial\mathbf{A}} = 2\mathbf{A}\mathbf{M}^A - 2\mathbf{N}^A \tag{4.50}$$

avec

$$\begin{aligned} M^A_{pq} &\overset{def}{=} \Sigma_{jk}\lambda_p B_{jp} C_{kp} C^*_{kq} B^*_{jq} \lambda^*_q \\ N^A_{ip} &\overset{def}{=} \Sigma_{jk} T_{ijk} B^*_{jp} C^*_{kp} \lambda^*_p \end{aligned} \tag{4.51}$$

Les expressions du gradients par rapport à $\mathbf{B}$ and $\mathbf{C}$ sont calculées de la même manière.

(ii) Calcule du pas : La deuxième sous-étape concerne la détermination du pas de progression $\rho^{(k)}$ le long de la direction choisie $\mathbf{d}^{(k)}$. Parmi les nombreuses méthodes de recherche linéaire pour le choix d'un pas, la méthode *Backtracking* est largement utilisée. qui dépend de deux paramètres α et β, avec $0 < \alpha < 0,5$ et $0 < \beta < 1$.

L'idée est de commencer avec un pas suffisamment grande $\rho^{(k)}$ (par exemple $\rho = 1$) au début, puis le réduire en $\rho \leftarrow \rho * \beta$, jusqu'à ce que la condition Armijo suivante soit vérifiée :

$$\Upsilon(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) < \Upsilon(\mathbf{x}^{(k)}) + \alpha\rho^{(k)}\nabla\Upsilon(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}. \tag{4.52}$$

Pour conclure, l'étape du gradient peut être résumée comme suit :

$$\mathbf{y}^{(k)} = \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}. \tag{4.53}$$

4.3.5.2 *Étape proximale*

Étant donné que l'étape précédente ne concerne que le terme de fidélité des données Υ, l'étape du proximale devrait ajuster l'orientation de la recherche en fonction de la

contrainte sur les cohérences $\mathcal{G}$. Et pour ce faire, nous appliquons l'operateur proximal au point précédent résultant de l'étape du gradient, c'est-à-dire $\mathbf{y}^{(k)}$, comme suit :

$$\begin{aligned}\mathbf{y}^{(k+1)} &= \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) = \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{y}^{(k)}) \\ &= \arg\min_{\mathbf{x}} \underbrace{(\mathcal{G}(\mathbf{x}) + \frac{1}{2\rho^{(k)}}\|\mathbf{x} - \mathbf{y}^{(k)}\|_2^2)}_{\mathcal{H}(\mathbf{x})}\end{aligned} \tag{4.54}$$

Cette étape indique que $\mathbf{prox}_{\mathcal{G}}(\mathbf{y}^{(k)})$ est un point permettant de faire un compromis entre la minimisation de $\mathcal{G}$ et un rapprochement de $\mathbf{y}^{(k)}$.

Dans le cas général, la fonction décrivant les contraintes peut ne pas être différentiable, mais il est recommandé qu'elle soit simple dans le sens où elle admet une forme proximale fermée.

Dans notre cas présent, la contrainte sur les cohérences est de nature beaucoup plus complexe pour avoir une solution fermée pour notre opérateur proximal. Cette particularité nous a conduit à utiliser sa version différentiable (4.16) au lieu de sa version initiale non-différentiable (4.13). Cela nous permettra de calculer une approximation de l'opérateur proximal de $\mathcal{G}$ en exploitant la différentiabilité de cette dernière à travers deux étapes principales suivantes :

(i) Gradient de $\mathcal{H}$: Le gradient de $\mathcal{H}$ prend la forme :

$$\nabla\mathcal{H}(\mathbf{x}) = \eta\nabla\mathcal{G}(\mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{y}^{(k+1)}), \tag{4.55}$$

où le gradient $\nabla\mathcal{G}(\mathbf{x})$ est calculé comme suit :

$$\frac{\partial g}{\partial \mathbf{A}} = \frac{-1}{\mathcal{C}_p}\mathcal{L}_p^A\mathbf{A}[(\mathbf{A}^H\mathbf{A}) \boxdot \Omega^A - I], \tag{4.56}$$

où $\boxdot$ désigne le produit de Hadamard.

$$\mathcal{L}_p^A \overset{def}{=} (\Sigma_{p<q}|\mathbf{a}_p^H\mathbf{a}_q|^{2p})^{\frac{-1}{2p}-1}\mu(\mathbf{B},p)^{-1}\mu(\mathbf{C},p)^{-1},$$

et $\Omega_{pq}^A \overset{def}{=} |\mathbf{a}_q^H\mathbf{a}_p|^{2p-2}$. Les expressions sont similaires pour les composantes de $\mathbf{B}$ et $\mathbf{C}$.

(ii) Test de descente : La minimisation dans l'étape du proximale est arrêtée dès que nous rencontrons un bon candidat qui améliore $\mathbf{y}^{(k)}$, à cet effet, nous introduisons un Contrôleur qui assure la propriété de descente $\mathcal{F}(\mathbf{y}^{(k+1)}) < \mathcal{F}(\mathbf{y}^{(k)})$; qui est défini comme suit :

$$\mathbf{x}_{k+1} = \begin{cases}\mathbf{y}_{k+1} & \text{if } \mathcal{F}(\mathbf{y}_{k+1}) < \mathcal{F}(\mathbf{y}_k) \\ \mathbf{y}_k & \text{otherwise.}\end{cases} \tag{4.57}$$

Cela signifie que le nouveau itéré $\mathbf{x}^{(k+1)}$ assure la minimisation de la fonction $\mathcal{G}$ tout en ne s'écartant pas trop de l'itéré $\mathbf{y}^{(k)}$ basé sur la fidélité des données Υ.

Noter bien aussi que cette étape décrit ci-dessus permet une migration facile vers l'algorithme de gradient en cas d'échec de l'étape du proximale pour trouver un meilleur candidat qui améliore la direction générale de la minimisation.

Algorithme 4.5 Résumé de l'algorithme gradient proximal pour la décomposition CP (CP-PG)

Initialisation : $(\mathbf{A}^{(0)}, \mathbf{B}^{(0)}, \mathbf{C}^{(0)})$ point initial arbitrairement.
Normaliser les colonnes des matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$
Calculer le $\boldsymbol{\lambda}^{(0)}$: $\boldsymbol{\lambda}^{(0)} = (\mathbf{G}^{(0)})^{-1}\mathbf{f}^{(0)}$,
Pour $k \geq 1$ **et tant que critère d'arrêt non satisfait, faire :**

1. **Étape gradient**
   - Calculer la direction de la descente $\mathbf{d}^{(k)}$ selon (4.49) par rapport à $\mathbf{x}_k$: $\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{x}_k)$
   - Calculer le pas de progression ρ_k en utilisant la méthode backtracking telle que : $\Upsilon(\mathbf{x}_k + \rho_k\mathbf{d}^{(k)}) < \Upsilon(\mathbf{x}_k) + \alpha\rho^{(k)}\nabla\Upsilon(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}$
   - Mise à jour $\mathbf{y}_{k+1} = \mathbf{x}_k + \rho_k\mathbf{d}^{(k)}$
2. **Étape proximale**
   - Calculer l'opérateur proximal approximatif de $\mathcal{G}$ à $\mathbf{y}_{k+1}$ en utilisant (4.55) tel que : $\mathbf{y}_{k+1} = \mathbf{prox}_{\rho_k\mathcal{G}}(\mathbf{y}_{k+1})$
   - Test de descente $\mathbf{x}_{k+1} = \begin{cases} \mathbf{y}_{k+1} & \text{Si } \mathcal{F}(\mathbf{y}_{k+1}) < \mathcal{F}(\mathbf{y}_k) \\ \mathbf{y}_k & \text{sinon.} \end{cases}$
3. Extraire les trois blocs de $\mathbf{X}_{k+1}$: $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ et $\mathbf{C}_{k+1}$
4. Normaliser les colonnes de $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ et $\mathbf{C}_{k+1}$
5. Calculer le facteur d'échelle optimal $\boldsymbol{\lambda}^*$ en utilisant (2.23) tels que : $\mathbf{G}\boldsymbol{\lambda}^* = \mathbf{f}$

Fin Pour
Retourner $\mathbf{A}^{(k+1)}$, $\mathbf{B}^{(k+1)}$ et $\mathbf{C}^{(k+1)}$.

4.3.6 Contribution 2

Les méthodes de gradient accéléré ont été au coeur de la recherche dans le domaine de l'optimisation convexe. Dans une série de travaux célèbres [Nesterov, 1983, Nesterov, 2005, Nesterov, 2013, Beck et Teboulle, 2009b, Beck et Teboulle, 2009a, Tseng, 2008], plusieurs méthodes de gradient accéléré sont proposées pour le problème (4.37) avec h et g convexes. Dans ces méthodes, k itérations sont suffisantes pour trouver une solution à une erreur de $O(\frac{1}{k^2})$ par rapport à la valeur objective optimale. Récemment, Ghadimi et Lan [Ghadimi et Lan, 2016] ont présenté un traitement unifié de la méthode du gradient accéléré (UAG) pour l'optimisation

convexe, non convexe et stochastique. Ils ont prouvé que leur algorithme converge dans la programmation non convexe avec h non convexe mais g convexe et s'accélère avec un taux de convergence $O(\frac{1}{k^2})$ dans la programmation convexe pour le problème (4.37).

Attouch et al. [Attouch *et al.*, 2013] ont proposé un cadre unifié pour prouver la convergence d'une classe générale de méthodes de descendance en utilisant l'inégalité de Kurdyka-Lojasiewicz (KL) pour le problème (4.37) et Frankel et al. [Frankel *et al.*, 2015] ont étudié les taux de convergence des méthodes de descendance générale en supposant que la fonction de désingularisation φ dans la propriété KL a la forme de $\frac{C}{\theta}t^{\theta}$. Un exemple typique dans leur travail de cadre est la méthode du gradient proximal. Cependant, il n'existe pas de littérature montrant qu'il existe une méthode de gradient accéléré satisfaisant les conditions de leur cadre.

Parmi les autres méthodes typiques du problème (4.37), on peut citer l'IFB (Inertial Forward-Backward) [Boţ *et al.*, 2016], l'iPiano [Ochs *et al.*, 2014], la GIST (General Iterative Shrinkage and Thresholding) [Gong *et al.*, 2013], la GDPA (Gradient Descent with Proximal Average) [Zhong et Kwok, 2014] et les IR (Iteratively Reweighted Algorithms) [Mohan et Fazel, 2012, Ochs *et al.*, 2015]. Le tableau 4.3 montre que les méthodes existantes ne sont pas idéales. La GD et l'IFB ne peuvent pas accélérer la convergence pour les problèmes convexes. La GIST et la GDPA exigent que f soit explicitement écrit comme une différence entre deux fonctions convexes. iPiano exige la convexité de g et l'IR est adapté à certains cas particuliers de problèmes (4.37). La GDPA peut accélérer la convergence pour les problèmes convexes, cependant, il n'est pas clair si la GDPA peut converger vers des points critiques pour les problèmes non convexes. Le GAP peut assurer la convergence pour la programmation non convexe, cependant, il exige que g soit convexe. Cela limite les applications de l'UAG à la résolution de problèmes régularisés non convexes, tels que l'apprentissage parcimonieux et le problème du faible rang.

Notre deuxième contribution consiste à accélérer l'algorithme du gradient proximal 4.5 tout en étendant l'algorithme du gradient proximal accéléré (APG) de Beck et Teboulle [Beck et Teboulle, 2009b, Beck et Teboulle, 2009a] pour résoudre le problème général de la non convexité (4.37). Dans ce but, l'APG commence par extrapoler un point $\mathbf{y}_k$ en combinant le point actuel et le point précédent, puis procède à la méthode du gradient proximal.

Dans le cas convexe. Bech et Teboulle [Beck et Teboulle, 2009a] étendent la méthode du gradient accéléré de Nesterov au cas non différentiable. Elle est appelée méthode du gradient proximal accéléré et comprend les étapes suivantes :

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1} - 1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}) \tag{4.58}$$

$$\mathbf{x}_{k+1} = \mathbf{prox}_{\rho_k g}(\mathbf{y}_k - \rho_k \nabla h(\mathbf{y}_k)) \tag{4.59}$$

$$t_{k+1} = \frac{\sqrt{4(t_k)^2 + 1} + 1}{2} \tag{4.60}$$

Méthodes	**Hypothèse**	**Accéléré (CP)**	**Accéléré (NCP)**
GD	$h+g$: KL	Non	Oui
iPiano	convexe g, non convexe h	Non	Oui
GIST	non convexe h, $g=g_1-g_2$, g_1, g_2 convexe	Non	Oui
GDPA	non convexe h, $g=g_1-g_2$, g_1, g_2 convexe	Non	Oui
IR	spécial h et g	Non	Oui
IFB	non convexe h, non convexe g	Non	Oui
APG	convexe h, convexe g	Oui	Pas clair
UAG	non convexe h, convexe g	Oui	Oui
nmAPG	non convexe h, non convexe g	Oui	Oui

TABLE 4.3 – Comparaisons des méthodes pour la résolution du problème (4.37). Les mesures comprennent l'hypothèse selon laquelle les méthodes accélèrent pour les programmes convexes (CP) et convergent pour les programmes non convexes (NCP).

Où l'operateur proximale est définie comme $\mathbf{prox}_{\rho g}(\mathbf{x}) = \mathbf{argmin}\ h(\mathbf{u}) + \frac{1}{2\rho}\|\mathbf{x}-\mathbf{u}\|^2$. L'APG n'est pas un algorithme monotone, ce qui signifie que $f(\mathbf{x}_{k+1})$ ne peut être inférieur à $f(\mathbf{x}_k)$. Beck et Teboulle [Beck et Teboulle, 2009b] ont donc proposé un algorithme APG monotone, qui comprend les étapes suivantes :

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}) \tag{4.61}$$

$$\mathbf{z}_{k+1} = \mathbf{prox}_{\rho_k g}(\mathbf{y}_k - \rho_k \nabla h(\mathbf{y}_k)) \tag{4.62}$$

$$t_{k+1} = \frac{\sqrt{4(t_k)^2+1}+1}{2} \tag{4.63}$$

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{Si } f(\mathbf{z}_{k+1}) < f(\mathbf{x}_k) \\ \mathbf{x}_k & \text{sinon.} \end{cases} \tag{4.64}$$

Lorsque l'on étend l'APG à la programmation non convexe, la principale difficulté réside dans le point $\mathbf{y}_k$ extrapolé. Nous avons peu de restrictions sur $f(\mathbf{y}_k)$ lorsque la convexité est absente. En fait, $f(\mathbf{y}_k)$ peut être arbitrairement plus grand que $f(\mathbf{x}_k)$ lorsque $\mathbf{y}_k$ est une mauvaise extrapolation, surtout lorsque f est oscillatoire. Lorsque $\mathbf{x}_{k+1}$ est calculé par un opérateur proximal à un mauvais $\mathbf{y}_k$, $f(\mathbf{x}_{k+1})$ peut également être arbitrairement plus grand que $f(\mathbf{x}_k)$. L'APG monotone de Beck et Teboulle [Beck et Teboulle, 2009b] assure $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$. Cependant, cela ne suffit pas pour assurer la convergence vers les points critiques. Pour remédier à ce problème, nous introduisons un moniteur satisfaisant à la propriété de descente suffisante pour éviter une mauvaise extrapolation de $\mathbf{y}_k$ et la corriger ensuite par ce moniteur.

Comme le montre le tableau 4.2, pour les problèmes d'optimisation convexe, [Beck et Teboulle, 2009b] ont proposé une méthode de Nesterov de gradient proximal accéléré (APG) et [Schmidt *et al.*, 2011] ont proposé une méthode de Nesterov de gradient proximal inexact accéléré (AIPG). La méthode APG et la méthode AIPG sont toutes deux accélérées par un terme de momentum. Cependant, comme mentionné précédemment, la méthode accélérée traditionnelle de Nesterov peut rencontrer un mauvais terme de momentum pour l'optimisation de notre problème non convexe (4.37). Pour résoudre ce problème, [Li et Lin, 2015] a ajouté un autre opérateur proximal comme moniteur pour rendre la fonction objectif suffisamment descendante. Pour que les fonctions objectives générées à partir de l'AIPG strictement descendante, nous suivons le cadre de l'APG en [Li et Lin, 2015]. Cela conduit au calcul de deux opérateurs proximaux inexacts $\mathbf{z}_{k+1} \in \mathbf{prox}_{\gamma g}^{\epsilon_k}(\mathbf{y}_k - \gamma \nabla h(\mathbf{y}_k))$ et $\mathbf{v}_{k+1} \in \mathbf{prox}_{\gamma g}^{\epsilon_k}(\mathbf{x}_k - \gamma \nabla h(\mathbf{x}_k))$, où $\mathbf{v}_{k+1}$ est un moniteur permet de rendre la fonction objectif strictement descendante. Dans cette optique, un algorithme de gradient proximal inexact accéléré (AIPG) sera construit selon l'algorithme 4.6.

Algorithme 4.6 Méthode accélérée de gradient proximal inexact (AIPG)

Entrée : m, error $\epsilon_k (k = 1, ..., m)$, $t_0 = 0$, $t_1 = 1$, stepsize $\gamma < \frac{1}{L}$.
Initialiser $\mathbf{x}_0 \in \mathbb{R}^d$ et $x_1 = z_1 = x_0$.
Pour $k = 1, ..., m$, faire :
$y_k = x_k + \frac{t_{k-1}}{t_k}(z_k - x_k) + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$.
Calculer $\mathbf{z}_{k+1}$ comme étant $\mathbf{z}_{k+1} \in \mathbf{prox}_{\gamma g}^{\epsilon_k}(\mathbf{y}_k - \gamma \nabla h(\mathbf{y}_k))$
Calculer $\mathbf{v}_{k+1}$ comme étant $\mathbf{v}_{k+1} \in \mathbf{prox}_{\gamma g}^{\epsilon_k}(\mathbf{x}_k - \gamma \nabla h(\mathbf{x}_k))$
$t_{k+1} = \frac{\sqrt{4t_k^2+1}+1}{2}$.
$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{Si } f(\mathbf{z}_{k+1}) < f(\mathbf{v}_{k+1}) \\ \mathbf{v}_{k+1} & \text{sinon.} \end{cases}$
Fin Pour
Retourner : $\mathbf{x}_{m+1}$.

Pour traiter la mauvaise extrapolation, c.a.d, le terme de momentum, dans le cadre de notre problème non convexe (4.37), l'AIPG dans l'algorithme 4.6 utilise une paire d'opérateurs proximaux inexacts pour assurer la descente stricte des fonctions objectives. Ainsi, l'AIPG est un algorithme monotone. En fait, l'utilisation de deux opérateurs proximaux est une stratégie assez conservatrice. Comme mentionné dans [Li et Lin, 2015], nous pouvons accepter $\mathbf{z}_{k+1}$ comme $\mathbf{x}_{k+1}$ directement s'il satisfait au critère $f(\mathbf{z}_{k+1}) \leq f(\mathbf{z}_k) - \frac{\delta}{2}\|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2$ qui montre que $\mathbf{y}_k$ est une bonne extrapolation. $\mathbf{v}_{k+1}$ n'est calculé que lorsque ce critère n'est pas rempli. On peut ainsi réduire le nombre moyen d'opérateurs proximaux. En suivant cette idée, un algorithme non monotone de gradient proximal inexact accéléré (nmAIPG) sera obtenu comme dans l'algorithme 4.7. Empiriquement, nous

constatons que le nmAIPG avec la valeur de $\delta \in [0.5, 1]$ fonctionne bien.

Algorithme 4.7 Méthode non monotone de gradient proximal inexact accéléré (nmAIPG)

Entrée : m, error $\epsilon_k(k = 1, ..., m)$, $t_0 = 0$, $t_1 = 1$, stepsize $\gamma < \frac{1}{L}$, $\delta > 0$.
Initialiser $\mathbf{x}_0 \in \mathbb{R}^d$ et $\mathbf{x}_1 = \mathbf{z}_1 = \mathbf{x}_0$.
Pour $k = 1, ..., m$, faire :
$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1})$.
Calculer $\mathbf{z}_{k+1}$ comme étant $\mathbf{z}_{k+1} \in \mathbf{prox}^{\epsilon_k}_{\gamma g}(\mathbf{y}_k - \gamma \nabla h(\mathbf{y}_k))$
Si $f(\mathbf{z}_{k+1}) \leq f(\mathbf{x}_k) - \frac{\delta}{2}\|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2$
Alors
$\mathbf{x}_{x+1} = \mathbf{z}_{x+1}$
Sinon
Calculer $\mathbf{v}_{k+1}$ comme étant $\mathbf{v}_{k+1} \in \mathbf{prox}^{\epsilon_k}_{\gamma g}(\mathbf{x}_k - \gamma \nabla h(\mathbf{x}_k))$

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{Si } f(\mathbf{z}_{k+1}) < f(\mathbf{v}_{k+1}) \\ \mathbf{v}_{k+1} & \text{sinon.} \end{cases}$$

Fin Si
$t_{k+1} = \frac{\sqrt{4t_k^2+1}+1}{2}$.
Fin Pour
Retourner : $\mathbf{x}_{m+1}$.

Ces résultats de convergence pour ces deux algorithmes seront en particulier développés dans le chapitre suivant où des algorithmes de gradient proximal accélérés exacts vont être proposés. Pour cette proposition actuelle, Nous exploitons les résultats de convergence de [Li et Lin, 2015] pour résoudre notre problème (4.47) en proposant une nouvelle stratégie efficace et simple capable d'assurer la propriété de descente suffisante. Cette dernière se traduit par la mise en place d'un contrôleur d'extrapolation permettant à chaque itération de garantir le choix correct de l'étape d'extrapolation.

Notre deuxième méthode proposée a pour but d'accélérer notre algorithme de gradient proximal 4.5. Cette accélération exige deux étapes de traitement supplémentaires, à savoir l'étape d'extrapolation et l'étape du contrôleur d'extrapolation. À cette fin, nous décrivons ces deux étapes complémentaires dans ce qui suit, puis nous présentons un résumé de notre deuxième algorithme dans 4.8.

Nous avons montré précédemment que l'algorithme du gradient proximal accéléré (APG) extrapole d'abord un point $\mathbf{y}_k$ par une combinaison du point actuel et du point précédent comme :

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1} - 1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}), \tag{4.65}$$

Ensuite résoudre le problème par les deux étapes principales 4.3.5.1 et 4.3.5.2 de la méthode du gradient proximal par rapport au point $\mathbf{y}_k$. Cependant, pour une mauvaise

extrapolation $\mathbf{y}_k$, $\mathcal{F}(\mathbf{y}_k)$ peut être arbitrairement plus grand que $\mathcal{F}(\mathbf{x}_k)$. Pour résoudre ce problème, nous avons introduit un contrôleur d'extrapolation garantissant la décroissante de la fonction $\mathcal{F}$. Ce dernier est traduit par le critère suivant :

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{Si } \mathcal{F}(\mathbf{z}_{k+1}) < \mathcal{F}(\mathbf{x}_k) \\ \mathbf{x}_k & \text{sinon.} \end{cases}$$

Nous pouvons bien remarquer que grâce à ce contrôleur, seules les bonnes extrapolations sont permises tandis que dans le cas contraire d'une mauvaise extrapolation, la formule récursive de la mise à jour de l'extrapolation $t_{k+1} = \frac{\sqrt{4(t_k)^2+1}+1}{2}$ permet de la corriger à l'itération suivante.

4.3.7 Simulations

Dans cette section, nous testons les performances des algorithmes proposés sur trois scénarios : (i) dans le premier scénario, le point limite se situe dans la région admissible, c'est-à-dire lorsque la contrainte n'est pas active ($C_p \geq 0$) à la convergence, (ii) le deuxième scénario est un exemple concret dans lequel il est nécessaire de faire face au problème du marais, c'est-à-dire. lorsque les matrices facteurs sont fortement corrélées dans les trois modes et que la contrainte est active ($C_p \leq 0$) à la convergence, et (iii) le dernier scénario traite le problème du goulot d'étranglement, dans lequel les facteurs du premier mode sont choisis d'être colinéaires.

Pour montrer l'intérêt de nos deux algorithmes, nous les comparons à deux autres algorithmes de la CP décomposition : i) leur homologue sans contrainte et sans calcul de la valeur optimale de $\mathbf{\Lambda}$ 3.4.2, ii) l'algorithme Barrière-Gradient projeté 4.3.

Les performances sont évaluées selon trois critères : l'erreur entre les matrices facteurs reconstruites et les matrices originales, la vitesse d'exécution et la meilleure somme des "congruences" [Tomasi et Bro, 2006].

La meilleure somme de congruences nécessite de trouver la meilleure permutation σ parmi les colonnes des matrices facteurs ; elle est définie comme :

$$\max_{\sigma} = \sum_{r=1}^{R} \frac{|a_r^H \widehat{a}_{\sigma(r)}|}{\|a_r\| \|\widehat{a}_{\sigma(r)}\|} \frac{|b_r^H \widehat{b}_{\sigma(r)}|}{\|b_r\| \|\widehat{b}_{\sigma(r)}\|} \frac{|c_r^H \widehat{c}_{\sigma(r)}|}{\|c_r\| \|\widehat{c}_{\sigma(r)}\|} \tag{4.66}$$

Afin d'obtenir des résultats comparables et de visualiser le comportement de chaque algorithme, les points initiaux de toutes les méthodes – à savoir l'algorithme non contraint, désigné par "Algo Unconstrained", l'algorithme barrière résolu par le gradient, désigné par "Algo Constrained Gradient", l'algorithme du gradient proximal désigné par ("Algo PG") et le gradient proximal accéléré, désigné par ("Algo APG") – sont initialisés en utilisant les mêmes points initiaux et partagent les mêmes critères d'arrêt :

— La tolérance sur la norme de Frobenius du gradient divisée par le nombre d'entrées du gradient est fixée à 10^{-8}.

Algorithme 4.8 Résumé de l'algorithme gradient proximal accélérée pour la décomposition CP (CP-APG)

Initialisation : $(\mathbf{A}^{(0)}, \mathbf{B}^{(0)}, \mathbf{C}^{(0)})$ point initial arbitrairement, $t_0 = 0$ et $t_1 = 1$.
Normaliser les colonnes des matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$
Calculer le $\boldsymbol{\lambda}^{(0)}$: $\boldsymbol{\lambda}^{(0)} = (\mathbf{G}^{(0)})^{-1}\mathbf{f}^{(0)}$,
Pour $k \geq 1$ **et tant que critère d'arrêt non satisfait, faire :**

1. **Extrapolation**
   - $\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1})$.
   - $t_{k+1} = \frac{\sqrt{4(t_k)^2+1}+1}{2}$.
2. **Étape gradient**
   - Calculer la direction de la descente $\mathbf{d}^{(k)}$, selon (4.49), par rapport à $\mathbf{y}_k$: $\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{y}_k)$
   - Calculer le pas de progression ρ_k en utilisant la méthode backtracking telle que : $\Upsilon(\mathbf{y}_k + \rho_k\mathbf{d}^{(k)}) < \Upsilon(\mathbf{y}_k) + \alpha\rho^{(k)}\nabla\Upsilon(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}$
   - Mise à jour $\mathbf{z}_k = \mathbf{y}_k + \rho_k\mathbf{d}^{(k)}$
3. **Étape proximale**
   - Calculer l'opérateur proximal approximatif de $\mathcal{G}$ par rapport à $\mathbf{z}_k$ en utilisant (4.55) tel que : $\mathbf{z}_{k+1} = \mathbf{prox}_{\rho_k\mathcal{G}}(\mathbf{z}_k)$
   - Contrôleur d'extrapolation : $\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{Si } \mathcal{F}(\mathbf{z}_{k+1}) < \mathcal{F}(\mathbf{x}_k) \\ \mathbf{x}_k & \text{sinon.} \end{cases}$
4. Extraire les trois blocs de $\mathbf{X}_{k+1}$: $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ et $\mathbf{C}_{k+1}$
5. Normaliser les colonnes de $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ et $\mathbf{C}_{k+1}$
6. Calculer le facteur d'échelle optimal $\boldsymbol{\lambda}^*$ en utilisant (2.23) tels que : $\mathbf{G}\boldsymbol{\lambda}^* = \mathbf{f}$

Fin Pour
Retourner $\mathbf{A}^{(k+1)}$, $\mathbf{B}^{(k+1)}$ et $\mathbf{C}^{(k+1)}$.

— Le nombre maximum d'itération est fixé à 10^3.

Dans toutes les expériences, les résultats sont obtenus à partir de 100 exécutions de Monte Carlo pour tous les scénarios. A chaque itération et pour chaque valeur SNR, une nouvelle réalisation du bruit est tirée.

4.3.7.1 Simulation 1

Dans cette expérience, nous générons un model CP aléatoire de taille $4 \times 3 \times 6$ avec un rang 3 et avec des cohérences $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.99$, ce qui implique que $\mu(\mathbf{A})\mu(\mathbf{B})\mu(\mathbf{C}) = 0.97$ est plus grand que $\frac{1}{R-1} = \frac{1}{2}$. Cette configuration décrit le cas où la contrainte est active ($C_p \leq 0$) à la convergence.

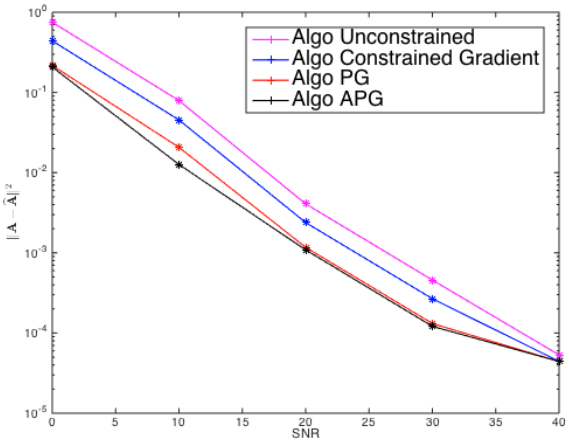
On fait varier η par itérations. Plus précisément, dans cette première expérience, η est initialisé à 1, et est divisé par 10 lorsque $\Upsilon(x)$ est réduit de moins de 10^{-4}. La figure 4.8 illustre les erreurs d'estimation des matrices en fonction du SNR. D'après ces résultats obtenus avec 100 points initiaux différents, nous pouvons constater que l'algorithme sans contrainte produit de mauvais résultats par rapport aux autres algorithmes. D'autre part, nous pouvons également observer que les deux algorithmes proximaux proposés sont plus précis que l'algorithme du gradient contraint.

La figure 4.5 représente la meilleure somme de congruences en fonction du SNR. Nous pouvons visualiser qu'avec les mêmes initialisations, les résultats sont beaucoup plus favorables, surtout à des SNR élevés, ce qui confirme la précision des algorithmes proximaux proposés.

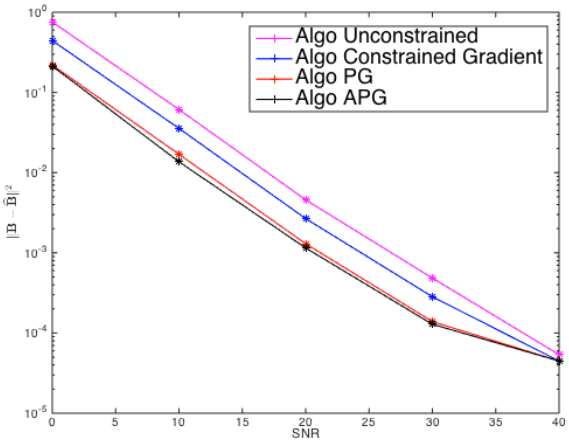
Pour illustrer une différence plus profonde entre les algorithmes, nous allons examiner la vitesse de convergence de la reconstruction du tenseur en fonction du nombre d'itérations, et nous allons également explorer le temps CPU pour mettre en évidence le temps d'exécution de chaque algorithme pour atteindre une précision de 10^{-6}. La figure 4.6 indique que l'algorithme du gradient proximal accéléré converge plus rapidement que les autres algorithmes en termes de nombre d'itérations, suivi par l'algorithme du gradient proximal, puis par l'algorithme du gradient contraint et enfin l'algorithme non contraint qui nécessite beaucoup d'itérations pour atteindre une précision de 10^{-6}. Ces résultats de vitesse de convergence sont également bien illustré dans le Tableau 4.4, qui indique le temps machine nécessaire pour chaque algorithme afin d'atteindre une précision de 10^{-6}, dans lequel il est clairement observé que le gradient proximal accéléré donne des résultats supérieurs pour faire face au phénomène de marais.

4.3.7.2 Simulation 2

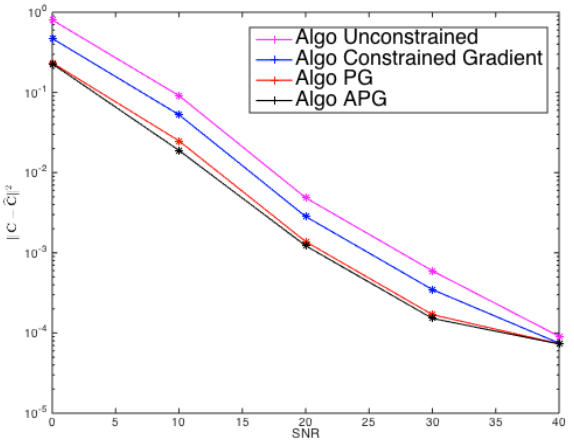
Dans cette deuxième expérience, nous générons un model CP aléatoire de taille $4\times3\times6$ avec un rang 3 et avec des cohérences $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.6$, ce qui implique que $\mu(\mathbf{A})\mu(\mathbf{B})\mu(\mathbf{C}) = 0,216$ est inférieur à $\frac{1}{R-1} = \frac{1}{2}$. Cette configuration décrit le cas où la contrainte n'est pas active ($C_p \geq 0$) à la convergence. Les points initiaux de toutes les méthodes sont générés aléatoirement, suivis de 5 itérations de la méthode des moindres



(a) Matrice **A**



(b) Matrice **B**



(c) Matrice **C**

FIGURE 4.4 – Erreur d'estimation des trois matrices en fonction de SNR, avec $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.99$.

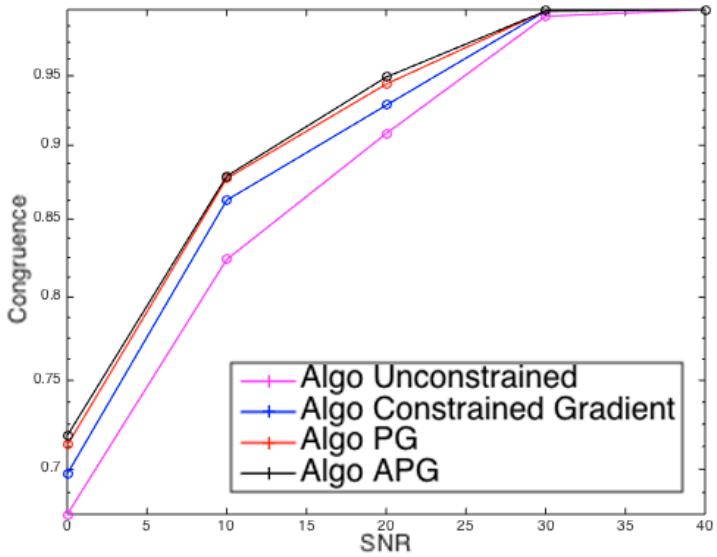


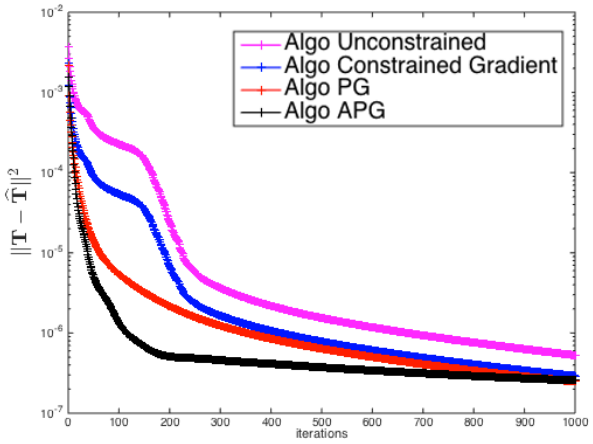
FIGURE 4.5 – Congruence par rapport aux SNRs avec une précision de 10^{-6} pour 100 initialisations aléatoires à chaque SNR.

SNR / **Algorithmes**	0	10	20	30	40
Unconstrained	2.10	1.67	1.44	0.80	0.61
Constrained Gradient	1.83	1.22	1.20	0.64	0.45
PG	1.71	1.13	1.11	0.59	0.44
APG	**1.62**	**0.83**	**0.76**	**0.46**	**0.43**

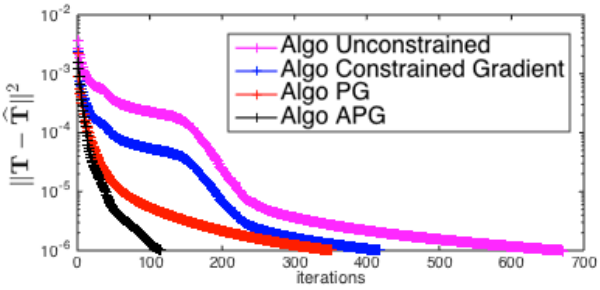
TABLE 4.4 – Temps CPU (en secondes) en fonction des SNRs avec une précision de 10^{-6} pour 100 initialisations aléatoires à chaque SNR.

carrés alternatifs (ALS). Dans cette expérience, η est initialisé à 0,1, et est divisé par 100 lorsque $\Upsilon(x)$ est réduit de moins de 10^{-4}.

La figure 4.7 présente l'erreur de reconstruction en fonction du nombre d'itérations. On peut donc observer que l'algorithme du gradient proximal accéléré converge plus rapidement que les autres algorithmes en termes de nombre d'itérations, suivi par l'algorithme du gradient proximal, puis par l'algorithme du gradient contraint, et finalement l'algorithme sans contrainte. Une inspection plus approfondie révèle également que l'algorithme proximal accéléré, l'algorithme du gradient proximal et l'algorithme du gradient contraint donnent 99% d'estimations correctes alors que l'algorithme sans contrainte en donné 97%.



(a) Up to 1000 iteration



(b) Up to a precision of 10^{-6}

FIGURE 4.6 – Erreur de reconstruction en fonction du nombre d'itérations, pour un tenseur de rang 3 et de taille $4 \times 3 \times 6$ avec $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.99$.

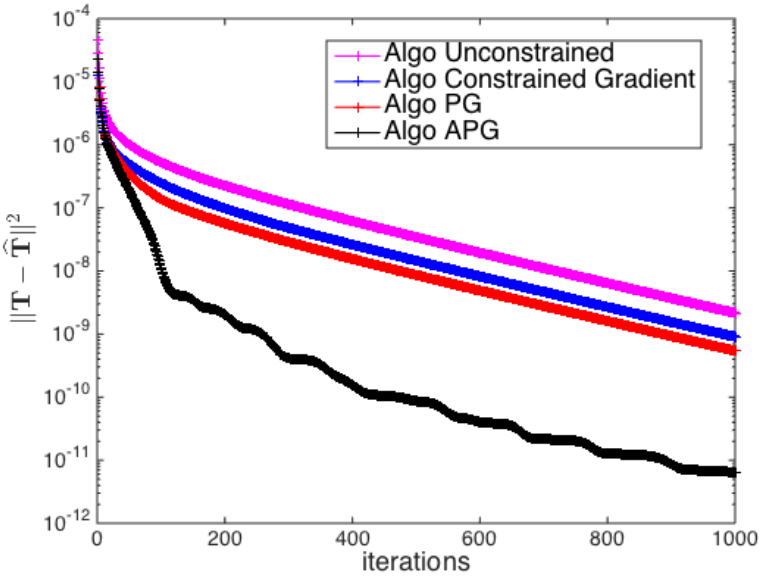


FIGURE 4.7 – Erreur d'estimation des trois matrices en fonction de SNR, pour un tenseur de rang 3 et de taille $4 \times 3 \times 6$ avec $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.6$.

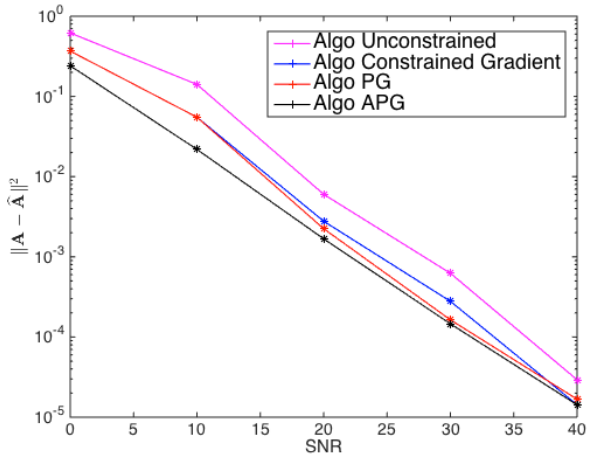
4.3.7.3 Simulation 3

Dans cette dernière expérience, nous générons un modèle CP aléatoire de taille $4 \times 3 \times 6$ avec un rang 3 et avec des cohérences $\mu(\mathbf{A}) = 0.99$, $\mu(\mathbf{B}) = 0.6$ et $\mu(\mathbf{C}) = 0.6$. Cette configuration adresse le problème du goulot d'étranglement, où nous avons choisi les facteurs du premier mode (c-a-d., la matrice $\mathbf{A}$) à être presque co-linéaires. Dans cette expérience, η est initialisé à 0,1, et est divisé par 100 lorsque $\Upsilon(x)$ est réduit de moins de 10^{-4}.

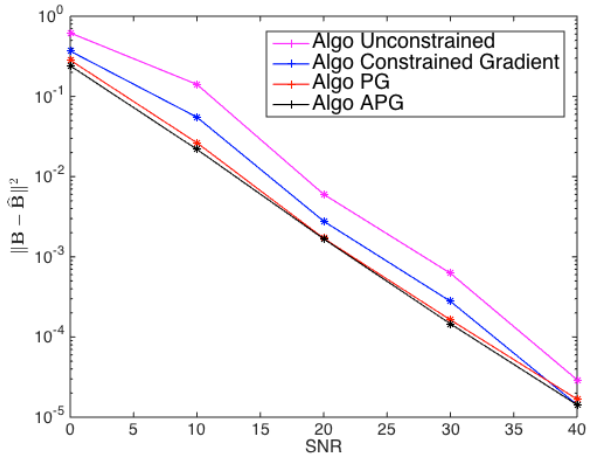
D'après les figures 4.8 et 4.9, nous pouvons encore observer que l'algorithme non contraint produit des résultats médiocres par rapport aux autres algorithmes. En revanche, l'algorithme du gradient proximal produit des résultats similaires à ceux de l'algorithme du gradient contraint pour des valeurs de SNR plus élevées, notamment pour les facteurs de la matrice corrélée $\mathbf{A}$. Cela peut s'expliquer par le fait que dans des SNR plus élevés, l'algorithme du gradient proximal est sensible aux points de départ. Cependant, l'algorithme proximal accéléré reste insensible aux points de départ et fournit plus de précision que les autres algorithmes.

La figure 4.10 montre que l'algorithme du gradient proximal accéléré converge plus rapidement que les autres algorithmes en termes de nombre d'itérations, tandis que les algorithmes du gradient proximal et du gradient contraint donnent des résultats similaires, suivis par l'algorithme non contraint qui nécessite beaucoup d'itérations.

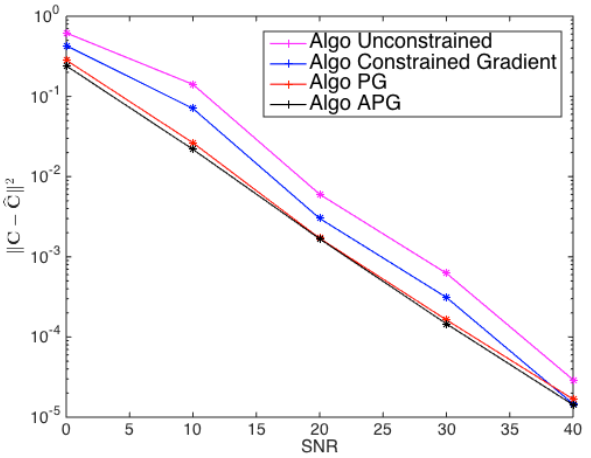
Ce résultat de vitesse de convergence est également bien illustré dans le Tableau 4.5, qui indique le temps CPU nécessaire à chaque algorithme pour atteindre une précision de 10^{-7}, et il convient de noter que l'algorithme du gradient proximal est un peu plus ra-



(a) Matrice **A**



(b) Matrice **B**



(c) Matrice **C**

FIGURE 4.8 – Erreur d'estimation des trois matrices en fonction de SNR, avec $\mu(\mathbf{A}) = 0.99$, $\mu(\mathbf{B}) = 0.6$, $\mu(\mathbf{C}) = 0.6$.

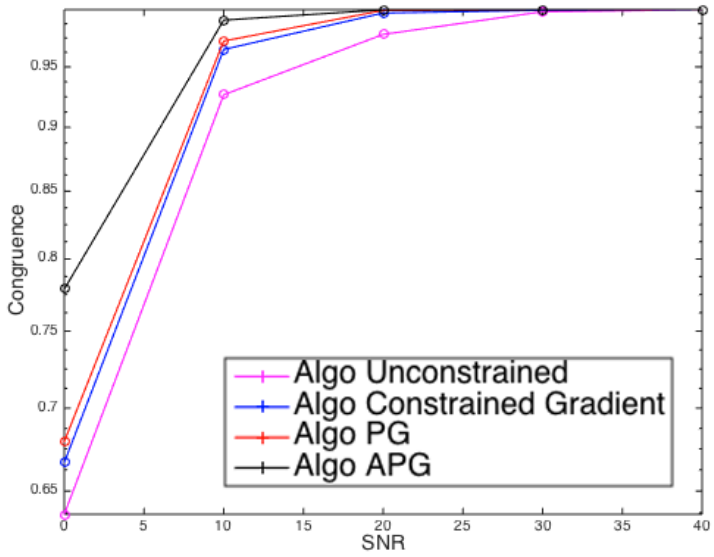


FIGURE 4.9 – Congruence par rapport aux SNRs avec une précision de 10^{-8} pour 100 initialisations aléatoires à chaque SNR.

pide que l'algorithme du gradient contraint, tandis que l'algorithme du gradient proximal accéléré reste le plus rapide pour atteindre une précision de 10^{-7}.

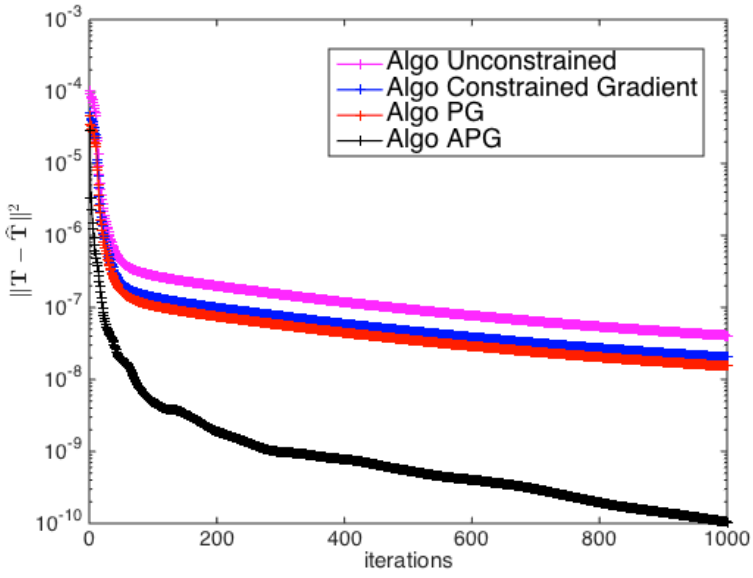


FIGURE 4.10 – Erreur d'estimation des trois matrices en fonction de SNR, pour un tenseur de rang 3 et de taille $4 \times 3 \times 6$ avec $\mu(\mathbf{A}) = 0.99$, $\mu(\mathbf{B}) = 0.6$ et $\mu(\mathbf{C}) = 0.6$.

SNR / **Algorithmes**	0	10	20	30	40
Unconstrained	1.85	1.66	1.19	0.95	0.62
Constrained Gradient	1.38	1.16	0.79	0.57	0.35
PG	1.37	1.15	0.77	0.52	0.36
APG	**1.35**	**1.12**	**0.74**	**0.47**	**0.30**

TABLE 4.5 – Temps CPU (en secondes) en fonction des SNRs avec une précision de 10^{-6} pour 100 initialisations aléatoires à chaque SNR.

4.3.8 Contribution 3

Dans cette section nous nous intéressons à un autre type des algorithmes proximaux, appelé algorithme Explicite-Implicite (Forward-Backward en anglais). cet algorithme a été initialement conçu pour le cas particulier où la somme d'un terme quadratique et d'une norme l_1 est considérée, cet algorithme est aussi connu sous le nom de Landweber seuillé. Il a été introduit sous une première forme dans [Bect *et al.*, 2004, Figueiredo et Nowak, 2003, Daubechies *et al.*, 2004] pour résoudre des problèmes inverses en présence de bruit gaussien. Cet algorithme a été généralisé par Combettes et Wajs dans [Combettes et Wajs, 2005] à une somme de deux fonctions appartenant à une classe plus large. Notre attention portera dans un premier temps sur cette dernière version nommée algorithme explicite-implicite et aux propriétés de convergences associées pour le cas général du problème (4.37) puis les adapter par la suite à notre problème de la décomposition CP par simple vérification des hypothèses requises.

Principe général de l'algorithme explicite-implicite *(Combettes et Wajs, 2005)* Soit $u_0 \in \mathcal{H}$ la valeur initiale. L'algorithme construit la suite $(u_n)_{k\in\mathbb{N}}$ en prenant, pour tout $n \in \mathbb{N}$,

$$u_{n+1} = u_n + \lambda_n(\mathbf{prox}_{\gamma g}(u_n - \gamma_n \nabla h(u_n) + b_n) + a_n - u_n), \tag{4.67}$$

où $\gamma_n > 0$ désigne le « pas » de l'algorithme et $\lambda_n > 0$ le paramètre de relaxation. Les suites $(a_n)_{n>0}$ et $(b_n)_{n>0}$ correspondent aux erreurs potentiellement introduites dans le calcul de l'opérateur proximal et du gradient. Il faut remarquer que ces erreurs ne sont pas fixées par l'utilisateur lors de l'implémentation de l'algorithme.

Proposition *(Combettes et Wajs, 2005), Prop. 3.1 :* Supposons que $h \in \Gamma_0(H)$ est différentiable de gradient $\beta - Lipschitz$, pour $\beta > 0$, $g \in \Gamma_0(H)$, et $\mathcal{H}$ est un espace de Hilbert réel, et soient $\hat{\mathbf{x}} \in \mathcal{H}$ et $\gamma \in]0, +\infty[$. Les assertions suivantes sont équivalentes :

i. $\hat{\mathbf{x}}$ est une solution du problème (4.37).

ii. $\hat{\mathbf{x}} = \mathbf{prox}_{\gamma g}(\hat{\mathbf{x}} - \gamma \nabla h(\hat{\mathbf{x}}))$.

D'après cette caractérisation de point fixe, une idée intuitive pour définir un algorithme itératif permettant de trouver une solution du problème (4.37) est :

$$(\forall k \in \mathbb{N}) \quad \mathbf{x}_{k+1} = \mathbf{prox}_{\gamma g}(\mathbf{x}_k - \gamma \nabla h(\mathbf{x}_k)), \tag{4.68}$$

pour un certain $\gamma > 0$ bien choisi. À chaque itération $k \in \mathbb{N}^*$, la suite $(\mathbf{x}_k)_{k\in\mathbb{N}}$ est donc définie comme étant la composition d'une étape de gradient sur la fonction h (appelée étape explicite), et d'une étape proximale sur la fonction g (appelée étape implicite).

De manière plus rigoureuse, un algorithme permettant de résoudre le problème (4.37) est l'algorithme Explicite-Implicite générant une suite dans $\mathcal{H}$, notée $(\mathbf{x}_k)_{k\in\mathbb{N}}$, grâce aux itérations suivantes [Chen et Rockafellar, 1997, Tseng, 2000] :

Algorithme 4.9 Algorithme Explicite-Implicite

Initialisation : Soient $x_0 \in \mathcal{H}$ et, pour tout $k \in \mathbb{N}$, $(\gamma_k, \lambda_k) \in]0, +\infty[^2$.
Itérations :
Pour $k = 0, 1, ...$
— $\mathbf{y}_k = \mathbf{prox}_{\gamma g}(\mathbf{x}_k - \gamma_k \nabla h(\mathbf{x}_k))$
— $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k(\mathbf{y}_k + \mathbf{x}_k)$

La suite des itérées $(\mathbf{x}_k)_{k\in\mathbb{N}}$ générée par l'algorithme 4.9 converge vers une solution du problème (4.37).

Proposition *(Combettes et Wajs, 2005), Thm. 3.4 :* Supposons que $h \in \Gamma_0(\mathcal{H})$ est différentiable et de gradient $\beta - Lipschitz$, pour $\beta > 0$, et $g \in \Gamma_0(\mathcal{H})$. Supposons de plus que $\mathbf{Argmin}(h + g) \neq \emptyset$. Soit $(\mathbf{x}_k)_{k\in\mathbb{N}}$ une suite générée par l'algorithme 4.9. Si
— $0 < inf_{(l\in\mathbb{N})}\gamma_l \leq sup_{(l\in\mathbb{N})}\gamma_l < 2\beta^{-1}$
— $(\forall k \in \mathbb{N}) \quad 0 < inf_{(l\in\mathbb{N})}\lambda_l \leq \lambda_k \leq 1.$
alors, $(\mathbf{x}_k)_{k\in\mathbb{N}}$ converge vers une solution du problème (4.37).

Récemment, dans le cas où $\mathcal{H} = \mathbb{R}^N$, les propriétés de convergence de l'algorithme Explicite-Implicite ont été étendues au cas où les fonctions h et g ne sont pas nécessairement convexes [Attouch et Bolte, 2009, Attouch *et al.*, 2013]. Ces résultats de convergence reposent principalement sur l'hypothèse que la fonction f satisfait l'inégalité de KL.

Théorème *(Attouch et al., 2013), Thm. 5 :* Supposons que f est une fonction propre, semi-continue inférieurement, bornée inférieurement, satisfaisant l'inégalité de KL. Supposons de plus que $h : \mathbb{R}^N \to \mathbb{R}$ est différentiable, de gradient $\beta - Lipschitz$, pour $\beta > 0$, et que $g : \mathbb{R}^N \to]-\infty, +\infty[$ est continue sur son domaine.

Si $(\mathbf{x}_k)_{k\in\mathbb{N}}$ est une suite bornée générée par l'algorithme 4.9, $\lambda_k = 0$, et $(\gamma_k)_{k\in\mathbb{N}}$ vérifie $0 < inf_{(l\in\mathbb{N})}\gamma_l \leq \Sigma_{k\in\mathbb{N}}\gamma_l < \beta^{-1}$, alors

i. la suite $(\mathbf{x}_k)_{k\in\mathbb{N}}$ converge vers un point critique $\hat{\mathbf{x}}$ de f ;

ii. la suite $(\mathbf{x}_k)_{k\in\mathbb{N}}$ est de longeur finie, c.-à-d.

$$\Sigma_{k\in\mathbb{N}}\|\mathbf{x}_{k+1}-\mathbf{x}_k\|<+\infty.$$

4.3.8.1 *Versions inexactes de l'algorithme Explicite-Implicite*

En pratique, l'opérateur proximal de la fonction g n'a pas nécessairement une forme explicite et doit être calculé de façon itérative. Il existe plusieurs méthodes permettant de prendre en compte des termes d'erreurs éventuels. L'une d'elles, donnée dans l'algorithme 4.10, consiste à considérer des termes d'erreurs sommables.

Algorithme 4.10 Algorithme Explicite-Implicite inexact (erreurs additives)

Initialisation : Soient $x_0\in\mathcal{H}$ et, pour tout $k\in\mathbb{N}$, $(\gamma_k,\lambda_k)\in]0,+\infty[^2$, $\mathbf{a}_0\in\mathcal{H}$ et $\mathbf{b}_0\in\mathcal{H}$.

Itérations :

Pour $k=0,1,...$

— $\mathbf{y}_k=\mathbf{x}_k+\lambda_k(\mathbf{prox}_{\gamma_g}(\mathbf{x}_k-\gamma_k\nabla h(\mathbf{x}_k)+\mathbf{b}_k)+\mathbf{a}_k-\mathbf{x}_k)$

— $\mathbf{x}_{k+1}=\mathbf{x}_k+\lambda_k(\mathbf{y}_k+\mathbf{x}_k)$

Dans cet algorithme, pour tout $k\in\mathbb{N}$, $\mathbf{a}_k\in\mathcal{H}$ est une terme d'erreur pouvant apparaitre lors du calcul de l'opérateur proximal, et $\mathbf{b}_k\in\mathcal{H}$ est un terme d'erreur pouvant apparaitre lors du calcul du gradient. Il a été démontré dans [Combettes et Wajs, 2005] que, lorsque h et g sont des fonctions convexes, si $\Sigma_{k\in\mathbb{N}}\|\mathbf{a}_k\|<+\infty$ et $\Sigma_{k\in\mathbb{N}}\|\mathbf{b}_k\|<+\infty$, alors le théorème 4.3.8 reste valide.

Une autre méthode modélisant un calcul approché de l'opérateur proximal de la fonction g est donnée dans l'algorithme 4.11. Dans cet algorithme, la première inégalité est appelée *condition de décroissance suffisante*, et la seconde est appelée *condition d'optimalité*. Il a été démontré dans [Attouch *et al.*, 2013] que le théorème 4.3.8 reste valide pour l'algorithme 4.11, lorsque l'on considère des fonctions h et g non nécessairement convexes.

Algorithme 4.11 Algorithme Explicite-Implicite inexact (erreurs relatives)

Initialisation : Soient $\mathbf{x}_0\in\mathbb{R}^N$, $\mathbf{a}\in]\beta,+\infty[$ et $\mathbf{b}\in]0,+\infty[$.

Itérations :

Pour $k=0,1,...$

— trouver $\mathbf{x}_{k+1}\in\mathbb{R}^N$ et $\mathbf{r}_{k+1}\in\partial g(\mathbf{x}_{k+1})$ tels que

— $g(\mathbf{x}_{k+1})+\langle\mathbf{x}_{k+1}-\mathbf{x}_k,\nabla h(\mathbf{x}_k)\rangle+\frac{\mathbf{a}}{2}\|\mathbf{x}_{k+1}-\mathbf{x}_k\|^2\leq g(\mathbf{x}_k)$,

— $\|\mathbf{r}_{k+1}+\nabla h(\mathbf{x}_k)\|\leq\mathbf{b}\|\mathbf{x}_{k+1}-\mathbf{x}_k\|$.

Parmi les méthodes de séparation non convexes, nous avons les résultats décrits dans la section précédente de [Attouch *et al.*, 2013, Attouch et Bolte, 2009] qui introduisent une méthode générique pour les problèmes non convexes et non différentiable basée sur la

théorie de Kurdyka-Lojasiewicz, ces résultats nous seront utiles par la suite quand nous traiterons particulièrement le phénomène de Marécage (Swamp). En revanche nous suivons les travaux plus récents de [Li et Lin, 2015, Nesterov, 2013] où une descente monotone de la valeur objective est imposée pour assurer la convergence.

Revenons maintenant à notre fonction de coût à minimiser (4.21) qui est composée d'une somme de deux termes, dont le premier concerne la fidélité des données - dénotée par Υ - et le second se réfère aux informations a priori sur les paramètres cibles, qui est dans notre cas la contrainte sur les cohérences définie au point (4.19) - dénotée et définie par $\mathcal{G} = -\eta\ log(\mathcal{C}_p)$. Dans ce cas, notre fonction objectif est définie comme suit :

$$\underset{\widehat{\mathbf{x}}\in\mathbb{R}^N}{\text{minimize}}\ \mathcal{F}(\widehat{\mathbf{x}}) = \|\mathbf{x} - \widehat{\mathbf{x}}\|_F^2 - \eta\ ln(C_p(\widehat{\mathbf{x}})) \tag{4.69}$$

Nous proposons ici d'utiliser la méthode Explicite-Implicite en suivant les travaux de [Li et Lin, 2015, Nesterov, 2013] pour résoudre le problème (4.69). Le principe général de notre méthode est décrit dans le paragraphe suivant et est résumé dans l'Algorithme 4.12.

Dans notre cas, l'hypothèse de la coercivité vérifiée à la section 2.4.2.5 avec un simple contrôle qui impose une descente monotone de la valeur objective serait suffisante pour assurer la convergence de notre algorithme. Il a été démontré dans [Li et Lin, 2015] que le problème posé (4.69) a au moins une solution et que, pour toute $\gamma \in]0, +\infty[$, ses solutions sont déterminées par les deux équations suivantes :

$$\mathbf{z}^{(k+1)} \leftarrow \underbrace{\mathbf{prox}_{\gamma^{(k)}\mathcal{G}}}_{Etape\ explicite}\ \underbrace{(\mathbf{x}^{(k)} - \gamma^{(k)}\nabla\Upsilon(\mathbf{x}^{(k)}))}_{Etape\ implicite} \tag{4.70}$$

$$\mathbf{x}^{(k+1)} \leftarrow \underbrace{\mathbf{x}^{(k)} + \tau^{(k)}(\mathbf{z}^{(k+1)} - \mathbf{x}^{(k)})}_{Relaxation} \tag{4.71}$$

Il apparaît clairement à partir de (4.71) que la méthode Explicite-Implicite consiste à résoudre trois sous-problèmes principaux, qui peuvent être réécrits comme suit :

$$\begin{aligned}
(sp_1)\ \ & \mathbf{y}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \gamma^{(k)}\nabla\Upsilon(\mathbf{x}^{(k)}) \\
(sp_2)\ \ & \mathbf{z}^{(k+1)} \leftarrow \underset{\mathbf{x}}{\mathbf{argmin}}\ (\eta^k\mathcal{G}(\mathbf{x}) + \frac{1}{2\gamma^{(k)}}\|\mathbf{x} - \mathbf{y}^{(k+1)}\|^2) \\
(sp_3)\ \ & \mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \tau^{(k)}(\mathbf{z}^{(k+1)} - \mathbf{x}^{(k)}).
\end{aligned}$$

Notre méthode Explicite-Implicite est donc définie comme étant la composition d'une étape de gradient sur la fonction Υ qui est l'étape explicite (appelée aussi étape du pas en avant), une étape proximale sur la fonction $\mathcal{G}$ qui est l'étape implicite (appelée aussi étape du pas en arrière) et une étape de relaxation.

Nous allons maintenant décrire plus en détail chacun de ces étapes décrit ci-dessus.

4.3.8.2 *Étape explicite*

Cette étape permet d'obtenir une solution approximative, en ajustant uniquement sur la fidélité des données, c.-à-d. la fonction Υ - indépendamment de la contrainte. Cette

étape comprend également deux autres sous-étapes.

(i) Direction de la descente : Cette premiere sous-étape consiste à calculer la direction de descente $\mathbf{d}^{(k)}$ de Υ, en assurant une diminution convenable. Dans notre cas, nous avons proposer d'utiliser la méthode de descente du gradient afin que la minimisation soit aussi simple que possible. Il convient de noter que, bien que la méthode du gradient puisse souffrir d'une lente convergence, en particulier pour les grands ensembles de données [Lee et Waziruddin, 1970], ces inconvénients seront atténués par la prochaine étape du "pas en arrière" qui ajuste l'orientation de la recherche en fonction de l'opérateur proximal de la fonction de cohérence $\mathcal{G}$. La direction est définie comme suit :

$$\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)}) = -\nabla\Upsilon(\mathbf{x}^{(k)}) \tag{4.72}$$

Où les expressions de gradient nécessaires pour déterminer la direction de la descente $\mathbf{d}^{(k)}$ sont de la forme :

$$\frac{\partial\Upsilon}{\partial\mathbf{A}} = 2\mathbf{A}\mathbf{M}^A - 2\mathbf{N}^A \tag{4.73}$$

avec

$$\begin{aligned} M^A_{pq} &\overset{def}{=} \Sigma_{jk}\lambda_p B_{jp} C_{kp} C^*_{kq} B^*_{jq} \lambda^*_q \\ N^A_{ip} &\overset{def}{=} \Sigma_{jk} T_{ijk} B^*_{jp} C^*_{kp} \lambda^*_p \end{aligned} \tag{4.74}$$

Expressions pour les gradients par rapport à $\mathbf{B}$ and $\mathbf{C}$ sont calculées de la même manière.

(ii) Calcule du pas : La deuxième sous-étape concerne la détermination du pas de progression $\rho^{(k)}$ le long de la direction choisie $\mathbf{d}^{(k)}$. Parmi les nombreuses méthodes de recherche linéaire pour le choix d'un pas, la méthode *Backtracking* est largement utilisée. qui dépend de deux paramètres α et β, avec $0 < \alpha < 0,5$ et $0 < \beta < 1$.

L'idée est de commencer avec un pas suffisamment grande $\rho^{(k)}$ (par exemple $\rho = 1$) au début, puis le réduire en $\rho \leftarrow \rho * \beta$, jusqu'à ce que la condition Armijo suivante soit vérifiée :

$$\Upsilon(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) < \Upsilon(\mathbf{x}^{(k)}) + \alpha\rho^{(k)}\nabla\Upsilon(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}. \tag{4.75}$$

Pour conclure, l'étape du *"pas en avant"* peut être résumée comme suit :

$$\mathbf{y}^{(k)} = \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}. \tag{4.76}$$

4.3.8.3 Étape implicite

Étant donné que l'étape précédente ne concerne que le terme de fidélité des données Υ, l'étape du *"pas en arrière"* devrait ajuster l'orientation de la recherche en fonction de la contrainte sur les cohérences $\mathcal{G}$. Et pour ce faire, nous appliquons l'algorithme proximal au point précédent résultant de l'étape précédente du gradient, c'est-à-dire $\mathbf{z}^{(k)}$, comme

suit :

$$\begin{aligned}\mathbf{y}^{(k+1)} &= \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) = \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{y}^{(k)}) \\ &= \arg\min_{\mathbf{x}} \underbrace{(\mathcal{G}(\mathbf{x}) + \frac{1}{2\rho^{(k)}}\|\mathbf{x} - \mathbf{y}^{(k)}\|_2^2)}_{\mathcal{H}(\mathbf{x})}\end{aligned} \tag{4.77}$$

Cette étape indique que $\mathbf{prox}_{\mathcal{G}}(\mathbf{y}^{(k)})$ est un point permettant de faire un compromis entre la minimisation de $\mathcal{G}$ et un rapprochement de $\mathbf{y}^{(k)}$.

Dans le cas général, la fonction décrivant les contraintes peut ne pas être différentiable, mais il est recommandé qu'elle soit simple dans le sens où elle admet une forme proximale fermée.

Dans notre cas, la fonction contrainte $\mathcal{G}$ est différentiable. Cela nous permettra de calculer une approximation de l'opérateur proximal de $\mathcal{G}$:

Gradient de $\mathcal{H}$: Le gradient de $\mathcal{H}$ prend la forme :

$$\nabla\mathcal{H}(\mathbf{x}) = \nabla\ \eta\ g(\mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{y}^{(k+1)}) \tag{4.78}$$

où le gradient $\nabla\mathcal{G}(\mathbf{x})$ est calculé comme suit :

$$\frac{\partial g}{\partial \mathbf{A}} = \frac{-1}{\mathcal{C}_p}\mathcal{L}_p^A\mathbf{A}[(\mathbf{A}^H\mathbf{A}) \boxdot \Omega^A - I] \tag{4.79}$$

où $\boxdot$ désigne le produit de Hadamard.

$$\mathcal{L}_p^A \overset{def}{=} (\Sigma_{p<q}|\mathbf{a}_p^H\mathbf{a}_q|^{2p})^{\frac{-1}{2p}-1}\mu(\mathbf{B},p)^{-1}\mu(\mathbf{C},p)^{-1},$$

et $\Omega_{pq}^A \overset{def}{=} |\mathbf{a}_q^H\mathbf{a}_p|^{2p-2}$. Les expressions sont similaires pour les composantes de $\mathbf{B}$ et $\mathbf{C}$.

La minimisation dans *l'étape implicite* est arrêtée dès que nous rencontrons un bon candidat qui améliore $\mathbf{y}^{(k+1)}$, ce qui signifie que le nouvel itérat $\mathbf{x}^{(k+1)}$ assure la minimisation de la fonction $\mathcal{G}$ ainsi qu'il ne s'écarte pas trop de l'itérat $\mathbf{y}^{(k+1)}$ basé sur la fidélité des données Υ.

4.3.8.4 Étape de relaxation

La méthode Explicite-Implicite offre une certaine flexibilité dans le choix de $\mathbf{x}^{(k+1)}$ grâce à un paramètre de relaxation $\tau^{(k)} \in [\epsilon, 1]$, qui agit comme une interpolation du point actuel $\mathbf{z}^{(k+1)}$ et du point précédent $\mathbf{x}^{(k)}$ comme :

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \tau^{(k)}(\mathbf{prox}_{\rho^{(k)}\eta\ g}(\mathbf{y}^{(k+1)}) - \mathbf{x}_k) \\ &= \mathbf{x}_k + \tau^{(k)}(\mathbf{z}^{(k+1)} - \mathbf{x}_k)\end{aligned} \tag{4.80}$$

Le paramètre de relaxation $\tau^{(k)}$ est fixé à $0,3$ pour toutes les itérations, cela vaut dire qu'une tolérance de 30% étant admise pour cette relaxation, il convient également

de noter que ce choix est effectué par toute une série de simulations pour ce paramètre.

Contrôleur de relaxation : L'algorithme Explicite-Implicite n'est pas un algorithme monotone [Beck et Teboulle, 2009b], cela peut avoir un impact sur le problème général de minimisation (4.69) qui peut impliquer un $\mathcal{F}(x_k)$ arbitrairement supérieur à $\mathcal{F}(x_{k+1})$. À cette fin, nous introduisons un moniteur qui assure la propriété de descente $\mathcal{F}(x^{(k+1)}) < \mathcal{F}(x^{(k)})$; qui est défini comme suit :

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_{k+1} & \text{if } \mathcal{F}(\mathbf{x}_{k+1}) < \mathcal{F}(\mathbf{x}_k) \\ \mathbf{y}_{k+1} & \text{otherwise.} \end{cases} \tag{4.81}$$

Noter bien que cette étape décrit ci-dessus permet une migration facile vers l'algorithme de gradient proximal lorsque le paramètre de relaxation dépasse la limite de tolérance prédéfinie.

4.3.9 Simulations

Dans cette section, nous comparons divers algorithmes sur la base des expérimentations réalisées dans deux scénarios : (i) le premier, qui est un exemple concret dans lequel il est nécessaire de faire face au phénomène de marais, c'est-à-dire lorsque les matrices facteurs sont fortement corrélées dans les trois modes et que la contrainte n'est pas satisfaite ($C_p \leq 0$) à la convergence, et (ii) le second scénario traite le cas normal de la décomposition CP, où le point limite se trouve dans la région admissible, c'est-à-dire lorsque la contrainte est satisfaite ($C_p \geq 0$) à la convergence, notons également que dans ce deuxième scénario nous avons fourni trois exemples de tenseurs de tailles différentes.

Pour voir l'intérêt de ce troisième algorithme que nous proposons, nous le comparons à d'autres algorithmes de la décomposition CP : i) L'algorithme de descente du gradient non contraint et avec calcul de la valeur optimale de Λ (UG) ii) L'algorithme de descente du gradient contraint (CG), iii) L'algorithme du gradient proximal (PG), iv) L'algorithme du gradient proximal accéléré (APG).

Pour illustrer le comportement de notre algorithme explicite-implicite proposé (FB), nous évaluons les performances de chaque algorithme en fonction de trois critères, à savoir la précision, le temps CPU et la meilleure somme de congruence.

Notez aussi que tous les algorithmes sont initialisés à chaque simulation avec les mêmes points initiaux, où les ensembles de données sont générés par des nombres aléatoires suivant une distribution normale de moyenne nulle et de variance unitaire, ainsi que tous les algorithmes partagent les critères d'arrêt communs suivants :

(i) Le nombre maximal d'itérations est fixé à 10^3.

(ii) Le taux d'erreur de reconstruction, qui est la distance entre le tenseur actuel et le tenseur original, pour la meilleure permutation σ.

Les résultats de ces simulations sont obtenus en réalisant la moyenne des résultats de 100 exécutions pour les trois exemples.

Algorithme 4.12 Résumé de l'algorithme Explicite-Implicite pour la décomposition CP (CP-FB)

Initialisation : $(\mathbf{A}^{(0)}, \mathbf{B}^{(0)}, \mathbf{C}^{(0)})$ point initial arbitrairement, $\tau^{(k)} = 0.3$.
Normaliser les colonnes des matrices $\mathbf{A}$, $\mathbf{B}$ et $\mathbf{C}$
Calculer le $\boldsymbol{\lambda}^{(0)}$: $\boldsymbol{\lambda}^{(0)} = (\mathbf{G}^{(0)})^{-1}\mathbf{f}^{(0)}$,
Pour $k \geq 1$ **et tant que critère d'arrêt non satisfait, faire :**

i. **Étape explicite**
   - Calculer la direction de la descente $\mathbf{d}^{(k)}$ selon (4.72) par rapport à : $\mathbf{x}_k$:
     $\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{x}_k)$
   - Calculer le pas de progression ρ_k en utilisant la méthode backtracking telle que :
     $\Upsilon(\mathbf{x}_k + \rho_k\mathbf{d}^{(k)}) < \Upsilon(\mathbf{x}_k)$
   - Mise à jour
     $\mathbf{y}_{k+1} = \mathbf{x}_k + \rho_k\mathbf{d}^{(k)}$

ii. **Étape implicite**
   - Calculer l'opérateur proximal approximatif de $\mathcal{G}$ à $\mathbf{y}_{k+1}$ en utilisant (4.78) tel que : $\mathbf{z}_{k+1} = \mathbf{prox}_{\rho_k\mathcal{G}}(\mathbf{y}_{k+1})$
   - Relaxation
     $\mathbf{x}_{k+1} = \mathbf{x}_k + \tau^{(k)}(\mathbf{z}_{k+1} - \mathbf{x}_k)$
   - Contrôle de la descente
     $$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_{k+1} & \text{Si } \mathcal{F}(\mathbf{x}_{k+1}) < \mathcal{F}(\mathbf{x}_k) \\ \mathbf{y}_{k+1} & \text{sinon.} \end{cases}$$

iii. Extraire les trois blocs de $\mathbf{X}_{k+1}$: $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ et $\mathbf{C}_{k+1}$

iv. Normaliser les colonnes de $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ et $\mathbf{C}_{k+1}$

v. Calculer le facteur d'échelle optimal $\boldsymbol{\lambda}^*$ en utilisant (2.23) tels que : $\mathbf{G}\boldsymbol{\lambda}^* = \mathbf{f}$

Fin Pour
Retourner $\mathbf{A}^{(k+1)}$, $\mathbf{B}^{(k+1)}$ et $\mathbf{C}^{(k+1)}$.

4.3.9.1 Scénario 1

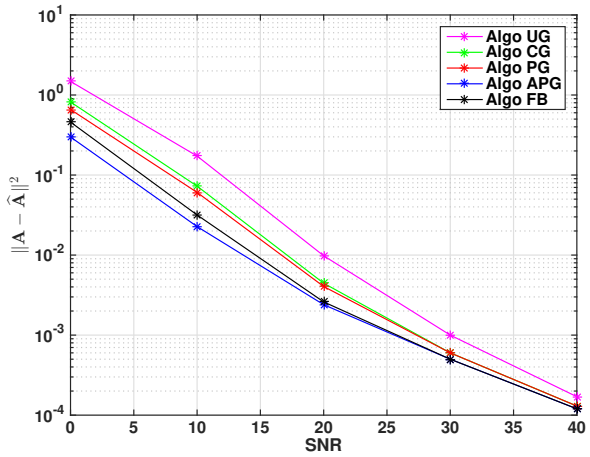
Dans ce premier scénario, nous générons un model CP aléatoire de taille $4 \times 3 \times 10$ de rang 5 et avec des cohérences $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.97$, ce qui implique que $\mu(\mathbf{A})\mu(\mathbf{B})\mu(\mathbf{C}) = 0.91$ est plus grand que $1/(R-1) = 1/4$. Cette configuration permet de résoudre le problème du marais, où toutes les matrices facteurs sont presque colinéaires. Pour cette expérience, η est initialisé à 1, et est divisé par 10 lorsque $\Upsilon(x)$ est réduit de moins de 10^{-2} et le paramètre de relaxation τ est fixé à 0,3 permettant ainsi une tolérance de 30% pour cette relaxation.

La figure 4.11 présente les erreurs d'estimation des matrices en fonction du SNR. A partir de ces résultats, on peut clairement observer que l'algorithme du gradient non contraint (UG) produit des résultats moins précis dans toutes les matrices facteurs par rapport aux autres algorithmes, ceci explique l'impact de la contrainte de cohérence sur la précision des algorithmes. D'autre part, on peut également observer que l'algorithme FB proposé est plus précis que les algorithmes du gradient contraint (CG) et du gradient proximal (PG), tandis que l'algorithme du gradient proximal accéléré (APG) reste le plus précis de tous les algorithmes, en particulier à un niveau de bruit élevé.

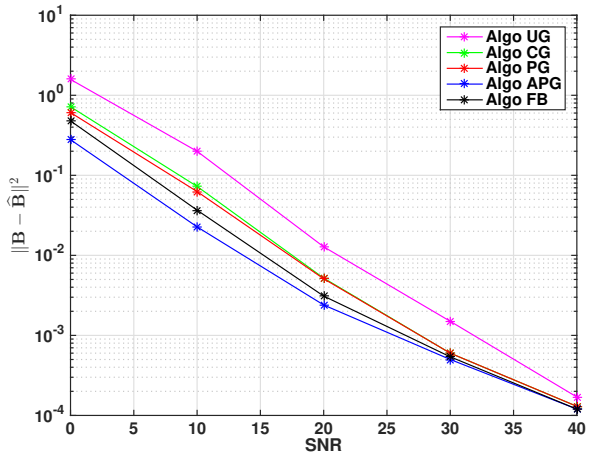
La figure 4.12 illustre la meilleure somme de congruences en fonction du SNR. On peut constater qu'avec les mêmes points de départ, les résultats de l'algorithme FB proposé et de l'algorithme APG sont beaucoup plus convaincants, cependant, l'algorithme APG est légèrement plus performant que l'algorithme FB, en particulier pour un niveau de bruit élevé. En outre, les algorithmes CG et PG donnent des résultats comparables, tandis que l'algorithme UG est moins performant que tous les autres algorithmes pour surmonter le phénomène de marais.

La figure 4.13 indique que l'algorithme UG produit des résultats moyens dans lesquels il n'atteint qu'une précision de 10^{-8}, alors que l'algorithme FB proposé et l'algorithme APG ont encore obtenu des résultats bien meilleurs que les algorithmes CG et PG du fait qu'ils convergent plus rapidement en termes de nombre d'itérations (notamment pour atteindre une précision de 10^{-9}), sauf que l'APG atteint cette précision en quelques itérations par rapport à l'algorithme FB. Par conséquent, l'algorithme APG reste toujours le choix préféré pour le cas difficile de la décomposition CP.

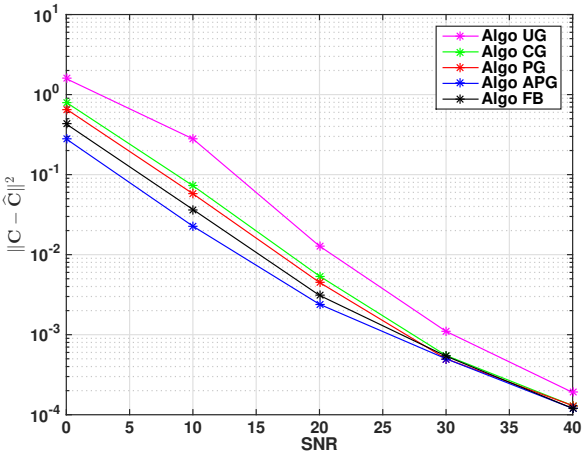
Ce résultat de vitesse de convergence est encore clairement présenté dans le tableau 4.6, qui liste le temps CPU nécessaire à chaque algorithme pour atteindre une précision de 10^{-8} et où nous pouvons remarquer un point significatif par rapport aux résultats précédents indiquant que l'algorithme PG est plus rapide que l'algorithme UG en termes de temps machine, une inspection approfondie révèle également que les algorithmes proximaux présentent une vitesse de convergence élevée, notamment pour traiter le problème des marécages.



(a) Matrice $\mathbf{A}$



(b) Matrice $\mathbf{B}$



(c) Matrice $\mathbf{C}$

FIGURE 4.11 – Erreur d'estimation des trois matrices en fonction de SNR, avec $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.97$.

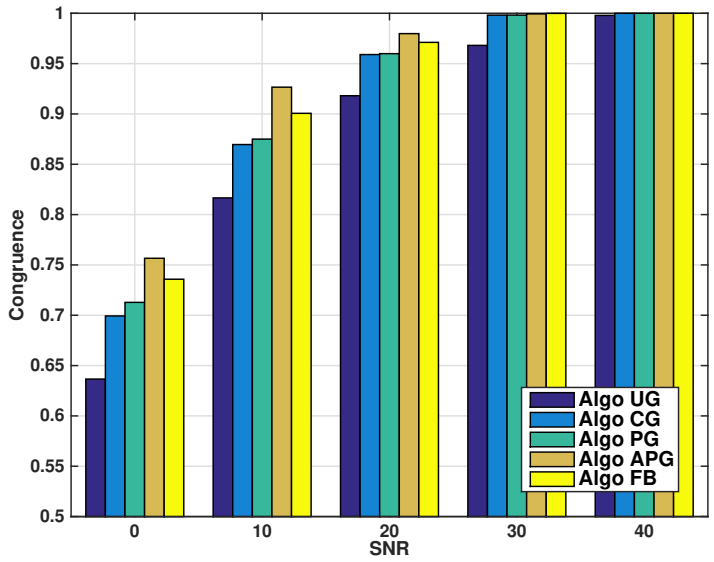


FIGURE 4.12 – Congruence par rapport aux SNRs avec une précision de 10^{-8} pour 100 initialisations aléatoires à chaque SNR.

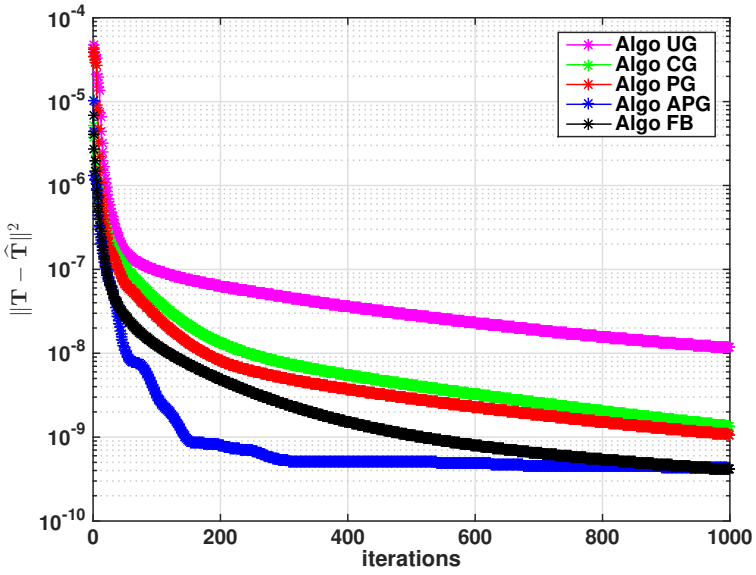


FIGURE 4.13 – Erreur d'estimation des trois matrices en fonction de SNR, pour un tenseur de rang 4 et de taille $4 \times 3 \times 10$ avec $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.97$.

4.3.9.2 Scénario 2

Dans ce deuxième scénario, nous abordons le cas normal de la décomposition CP à travers trois différentes dimensions de tenseurs, pour lesquels les points limites se situent dans la région admissible, c'est-à-dire lorsque la contrainte est satisfaite ($C_p \geq 0$) à la

SNR / **Algorithmes**	0	10	20	30	40
UG	2.45	1.78	1.66	0.96	0.62
CG	2.04	1.52	1.23	0.72	0.54
PG	1.80	1.21	1.10	0.68	0.50
APG	**1.59**	**0.95**	**0.84**	**0.47**	**0.44**
FB	1.75	1.09	0.91	0.52	0.48

TABLE 4.6 – Temps CPU (en secondes) en fonction des SNRs avec une précision de 10^{-8} pour 100 initialisations aléatoires à chaque SNR.

convergence. Il est à noter que le paramètre de relaxation τ est fixé à 0,7 pour tous les exemples de ce scénario, permettant ainsi une tolérance de 70% pour cette relaxation, Il est également à noter que ce choix est fait à travers une série de simulations pour ce paramètre.

Exemple 1 Dans ce premier exemple, nous générons un modèle CP aléatoire de taille $2 \times 2 \times 2$ et de rang 2, η est initialisé à 0,1, et divisé par 100 lorsque $\Upsilon(\mathbf{x})$ est réduit de moins de 10^{-4}. La figure 4.14 représente l'erreur de reconstruction en fonction du nombre d'itérations. A partir de ces résultats, nous observons clairement l'impact de la contrainte sur les performances des algorithmes dans lesquels nous comprenons que l'UG prend beaucoup d'itération pour atteindre seulement une précision de 10^{-11}. De plus, nous pouvons également remarquer que les trois algorithmes proximaux sont plus efficaces que le CG avec l'avantage de l'algorithme FB proposé qui atteint une précision de 10^{-14} en 120 itérations par rapport à l'APG qui nécessite plus d'itérations pour atteindre la même précision.

Afin de faire une comparaison plus précise, nous avons évalué le temps CPU de ces algorithmes. En analysant les résultats du Tableau 4.7, nous pouvons clairement voir que l'algorithme FB proposé prouve son efficacité en termes de précision et de temps CPU par rapport aux autres algorithmes.

UG	CG	PG	APG	FB
2.07	1.88	1.65	1.58	**1.47**

TABLE 4.7 – Temps CPU (en secondes) en fonction des SNRs avec une précision de 10^{-8} pour 100 initialisations aléatoires à chaque SNR.

Exemple 2 Dans ce deuxième exemple, nous générons un model CP aléatoire de taille $4 \times 3 \times 10$ et de rang 4, η est initialisé à 0,1, et divisé par 100 lorsque $\Upsilon(\mathbf{x})$ est réduit de moins

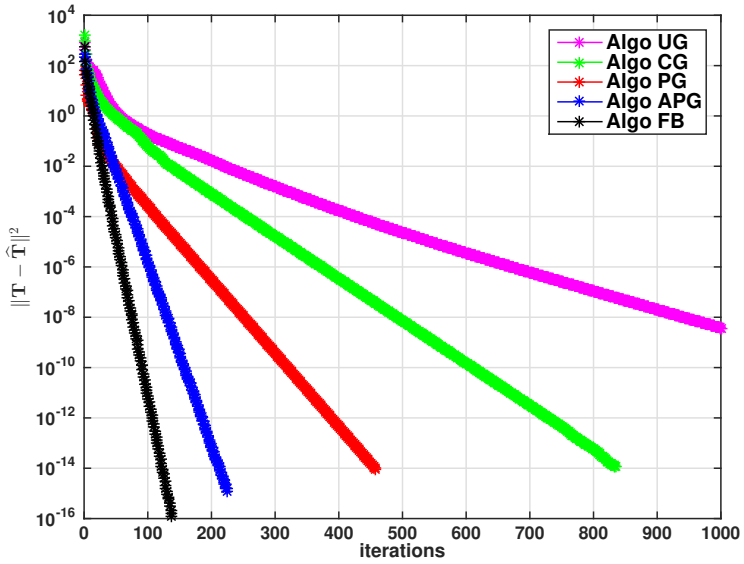


FIGURE 4.14 – Erreur d'estimation des trois matrices en fonction de SNR, pour un tenseur de rang 2 et de taille $2 \times 2 \times 2$.

de 10^{-4}. La figure 4.15 révèle que l'algorithme FB proposé nécessite moins de 230 itérations pour atteindre une précision de 10^{-14}, suivi par l'algorithme APG qui nécessite plus de 300 itérations pour atteindre la même précision. Nous pouvons également remarquer que, bien que l'algorithme CG produise de bons résultats par rapport à l'algorithme PG en termes de précision, cette performance est un peu coûteuse en termes de temps CPU pour ces deux algorithmes comme le montre le Tableau 4.8. D'autre part, les résultats du Tableau tab5xp prouvent toujours l'efficacité de l'algorithme proposé en termes de temps CPU par rapport aux autres algorithmes.

UG	CG	PG	APG	FB
2.17	1.78	1.73	1.64	**1.57**

TABLE 4.8 – Temps CPU (en secondes) en fonction des SNRs avec une précision de 10^{-12} pour 100 initialisations aléatoires à chaque SNR.

Exemple 3 Dans ce troisième exemple, nous générons un modèle CP aléatoire de taille $40 \times 40 \times 40$ et de rang 4, η est initialisé à 0,5, et divisé par 100 lorsque $\Upsilon(\mathbf{x})$ est réduit de moins de 10^{-4}. La figure 4.16 et le tableau 4.9 montrent qu'avec des tenseurs plus grands que les exemples précédents, l'algorithme non contraint nécessite toujours plus d'itérations et plus de temps CPU pour atteindre une précision de 10^{-11}. D'autre part, l'algorithme FB proposé dépasse significativement les algorithmes UG et PG en termes de

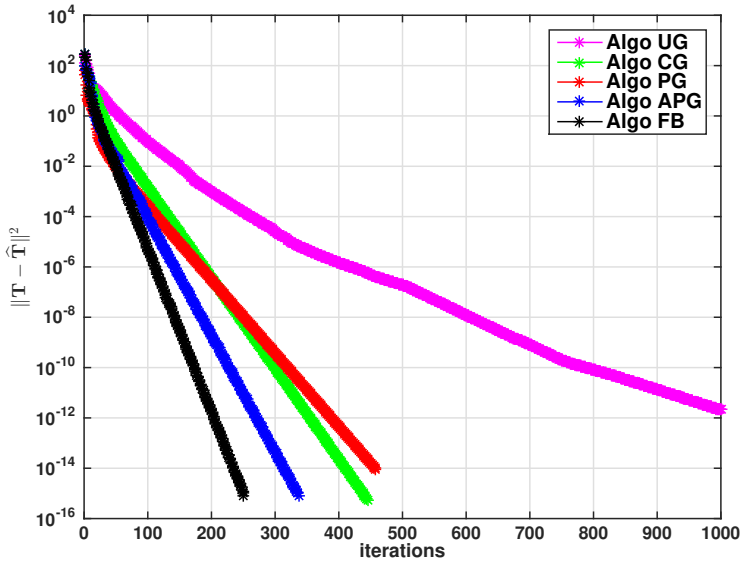


FIGURE 4.15 – Erreur d'estimation des trois matrices en fonction de SNR, pour un tenseur de rang 4 et de taille $4 \times 3 \times 10$.

nombre d'itérations et également en termes de temps machine. De plus, cette amélioration est légèrement significative par rapport à l'algorithme APG.

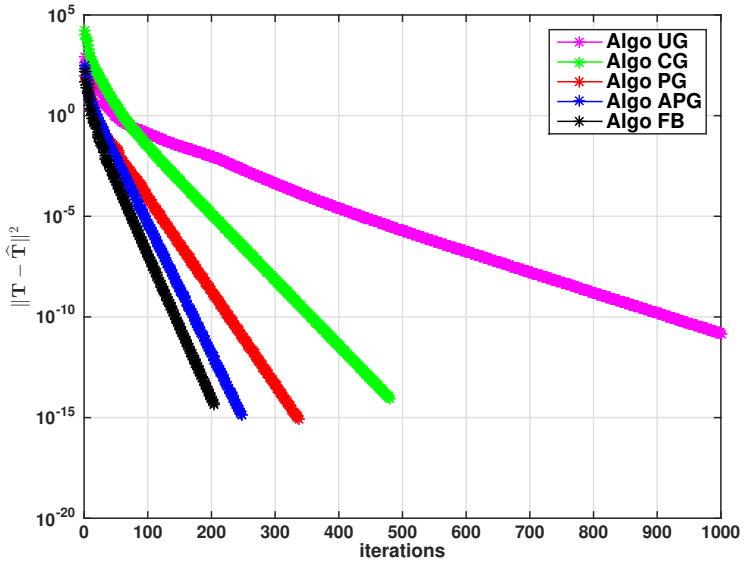


FIGURE 4.16 – Erreur d'estimation des trois matrices en fonction de SNR, pour un tenseur de rang 4 et de taille $40 \times 40 \times 40$.

Selon ces résultats de simulation, il est facile de voir l'impact de la contrainte de cohérence sur la précision des résultats, et nous pouvons également observer qu'avec différentes

UG	CG	PG	APG	FB
2.29	1.92	1.68	1.40	**1.46**

TABLE 4.9 – Temps CPU (en secondes) en fonction des SNRs avec une précision de 10^{-10} pour 100 initialisations aléatoires à chaque SNR.

tailles de tenseurs, l'efficacité de l'algorithme proposé est perçue. Cela fait de l'algorithme FB un meilleur choix pour assurer à la fois la précision et la vitesse de convergence de la décomposition CP dans le cas normal.

4.4 Bilan du chapitre

Dans ce chapitre, nous avons présenté deux types différents d'algorithmes d'optimisation permettant la résolution du problème de la décomposition CP avec contraintes.

En premier lieu, nous avons introduit le principe général de la méthode du gradient projeté, puis nous avons exposé deux méthodes pour la minimisation de la fonction de coût sous une contrainte d'égalité imposée sur les colonnes des matrices de facteurs. Deux algorithmes d'optimisation ont été élaborés : "Algorithme 1" et "Algorithme 2". Ces deux algorithmes reposent sur la méthode du gradient projeté tout en prenant en compte l'isolement du facteur d'échelle. Les résultats de l'analyse de ces deux algorithmes ont montré en effet qu'ils sont beaucoup moins sensibles à l'initialisation que l'ALS et qu'ils offrent de meilleures propriétés de convergence. Cependant, ils s'avèrent toujours incapables de surmonter les cas difficiles où les matrices facteurs d'un ou de tous les modes sont colinéaires.

Ensuite, lorsque nous avons également considéré la contrainte sur les cohérences afin de garantir l'existence et l'unicité de la solution de la nouvelle fonction de coût sous cette fois une contrainte d'inégalité. Trois nouveaux algorithmes d'optimisation ont été développés : "Algorithme gradient proximal", "Algorithme gradient proximal accéléré" et "Algorithme explicite implicite". Ces trois algorithmes proposés ont été conçus pour résoudre notre fonction de coût sous la contrainte d'inégalité. Les résultats de simulations ont démontré le bon comportement de ces trois algorithmes, ainsi qu'un meilleur compromis entre la précision et la vitesse de convergence par rapport aux autres algorithmes dans la littérature dans le cas normal et surtout dans les cas difficle où les phénomènes de goulot d'étranglement ou de marais se posent.

Dans le chapitre suivant, nous présenterons en outre deux nouvelles méthodes pour la décomposition CP non-négative qui sont toujours basées sur les méthodes proximales étudiées au cours de ce chapitre, en particulier sur la méthode du gradient proximal accéléré.

Chapitre

5

LA NONNÉGATIVE CP DECOMPOSITION

Sommaire

5.1 Introduction

Un fil conducteur dans diverses approches de réduction de modèles, de regroupement, d'extraction de caractéristiques, de classification et de séparation aveugle de sources (SAS) consiste à représenter les données originales par une approximation à faible dimension obtenue par factorisation ou décomposition de matrices ou de tenseurs. La notion de factorisations matricielles/tensorielles apparaît dans un large éventail d'applications importantes et chaque factorisation matricielle/tensorielle implique des différentes hypothèses concernant les matrices composantes (facteurs) et leurs structures sous-jacentes. Le choix de la méthode appropriée est donc essentiel dans chaque domaine d'application. Les factorisations approximatives de matrices et de tenseurs à faible rang jouent un rôle fondamental dans l'amélioration des données et l'extraction des composantes latentes (cachées).

Dans les approches de factorisation en matrices/tenseurs non-négatifs, les données à haute dimension telles que les séries temporelles non négatives ou les images sont factorisées pour trouver des composantes latentes non négatives significatives [Cichocki *et al.*, 2009]. Ces approches sont motivées par le fait que, outre la réduction de la dimensionnalité recherchée dans de nombreuses applications, l'ensemble des données sous-jacentes est non négatif et peut être mieux modélisé et interprété au moyen de composantes non négatives et, de préférence, parcimonieuses ou lisses, afin d'obtenir une représentation unique basée sur des parties additives (combinaisons non soustractives de bases non négatives) [Lee et Seung, 1999].

La factorisation en matrices non négatives (NMF) a été étudiée par de nombreux chercheurs, notamment Paatero et Tapper [Paatero et Tapper, 1994], pendant plus de 15 ans, mais elle a gagné en popularité grâce aux travaux de Lee et Seung publiés dans Nature [Lee et Seung, 1999]. En se basant sur l'argument selon lequel les composantes non négatives sont importantes. Dans la perception visuelle du cerveau humain, ils ont proposé des algorithmes simples de mise à jour multiplicative pour trouver des représentations additives localisées basées sur des parties significatives d'images de visages (comme les sourcils, la bouche et le nez). Par la suite, la NMF a fait l'objet d'études approfondies et l'idée a été étendue aux modèles multilinéaires qui effectuent des décompositions tensorielles, y compris la décomposition de Tucker non négatives (NT) et la décomposition canonique polyadique non négative (NCP) [Cichocki *et al.*, 2009].

Dans le domaine du traitement du signal et de l'analyse des données, les factorisations en matrices/tenseurs non négatifs sont des outils importants et omniprésents en raison de leurs propriétés uniques et de leurs nombreuses applications [Cichocki *et al.*, 2009]. Comme la théorie et les nouvelles applications des factorisations en matrices/tenseurs non négatifs sont toujours en cours de développement et font l'objet de recherches approfondies (y compris l'unicité, la performance et l'estimation du rang non négatif).

Notre objectif est de présenter des cadres algorithmiques et informatiques pour l'analyse et le développement d'algorithmes efficaces et robustes pour la décomposition NCP destinés à la représentation des signaux, en particulier, basés sur des approximations

à faible rang de données à haute dimension. Ces topiques sont des facteurs clés pour le développement de nombreuses applications émergentes dans la vie réelle, par exemple, les affichages tensoriels vidéo tridimensionnels (3-D) [Wetzstein *et al.*, 2012], l'exploration de textes, la classification et le regroupement [Cichocki *et al.*, 2009, Kumar *et al.*, 2013, Thurau *et al.*, 2009].

Dans ce chapitre, nous introduirons tout d'abord le principe général de la factorisation de matrices non négatives qui constitue le fondement de la plupart des algorithmes de la décomposition NCP, puis nous exposerons les algorithmes et les calculs de base de la décomposition tensorielle NCP dans la littérature. Ensuite, nous présenterons nos contributions au NCP, dont les modèles mathématiques proposés sont combinés avec des éléments de régularisation basés sur des algorithmes proximaux.

5.2 La factorisation matricielle non-négatives

Dans la factorisation matricielle non négative (NMF), une matrice de données mixtes (non négative) $\mathbf{X}$ de taille $m \times n$ est approximativement factorisée en un produit de deux matrices de rang-k non négatives, avec k petit par rapport à m et n, $\mathbf{X} \approx \mathbf{WH}$. Cette factorisation présente l'avantage que $\mathbf{W}$ et $\mathbf{H}$ peuvent fournir une représentation physiquement réalisable des données mixtes. La NMF est largement utilisée dans une variété d'applications, y compris le contrôle des émissions atmosphériques, le traitement des images et des données spectrales, la fouille de textes, l'analyse chimiométrique, les processus d'apprentissage par réseau de neurones, la reconnaissance sonore, la télédétection et la caractérisation des objets, voir, par exemple, [Berry *et al.*, 2007].

Problème NMF : Étant donné une matrice non négative $\mathbf{X} \in \mathbb{R}^{m \times n}$ et un entier positif $k \leq \min\{m, n\}$, l'objectif est de trouver des matrices non négatives $\mathbf{W} \in \mathbb{R}^{m \times k}$ et $\mathbf{H} \in \mathbb{R}^{k \times n}$ qui minimise la fonction $f(\mathbf{W}, \mathbf{H}) = \frac{1}{2}\|\mathbf{X} - \mathbf{WH}\|_2^F$, c'est-à-dire.

$$\min_{\mathbf{H}} f(\mathbf{H}) = \|\mathbf{X} - \sum_{i=1}^{k} \mathbf{W}^{(i)} \circ \mathbf{H}^{(i)}\| \quad tel\ que \quad \mathbf{W}, \mathbf{H} \geq 0, \tag{5.1}$$

où $\mathbf{W}^{(i)}$ est la i-ème colonne de $\mathbf{W}$ et $\mathbf{H}^{(i)}$ la i-ème colonne de $\mathbf{H}^T$.

Voir la figure 5.1 qui illustre l'approximation d'une matrice par une somme de matrices de rang un déterminées par $\mathbf{W}$ et $\mathbf{H}$. La somme est tronquée après k termes.

Un grand nombre d'algorithmes numériques ont été développés pour résoudre la NMF. Les méthodologies adaptées suivent plus ou moins les principes des itérations en sens alterné, du Newton projeté, de l'approximation quadratique réduite et de la recherche de descente. Les implémentations spécifiques peuvent généralement être classées en algorithmes de moindres carrés alternatifs [Paatero et Tapper, 1994], algorithmes de mise à jour multiplicative [Hoyer, 2002, Lee et Seung, 1999, Lee et Seung, 2001], algorithmes de descente de gradient et algorithmes hybrides [Pauca *et al.*, 2004, Piper *et al.*, 2004]. Des évaluations générales de ces méthodes peuvent être trouvées dans [Chu *et al.*, 2004,

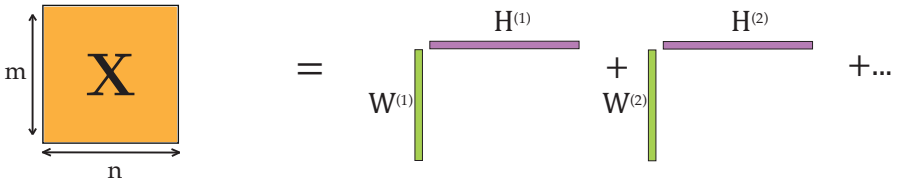


FIGURE 5.1 – Une illustration de la factorisation des matrices non négatives.

Liu et Yi, 2003]. Bien que les schémas et les approches soient différents, toute méthode numérique est essentiellement centrée sur la satisfaction des conditions d'optimalité du premier ordre dérivées de la théorie de Kuhn-Tucker. Notez que les facteurs calculés $\mathbf{W}$ et $\mathbf{H}$ peuvent seulement être des minimisateurs locaux de 5.1.

5.3 La décomposition CP non négative

Dans la plupart des cas, les composantes tensorielles sont non-négatives, de sorte qu'il est nécessaire d'incorporer une contrainte non négative dans le modèle de la décomposition pour pouvoir exploiter ces importantes composantes non négatives. Cela donne lieu à la décomposition CP non négative (NCP), qui est l'un des plus importants modèles de décomposition tensorielle avec contraintes. Cette dernière a connu un grand succès dans plusieurs applications pratiques, telles que la décomposition des données hyperspectrales [Rodriguez-Fernandez *et al.*, 2015], la décomposition des données d'électroencéphalographie (EEG) [Wang *et al.*, 2018], la décomposition des données de matrice d'excitation-émission de fluorescence (EEM) [Rodriguez-Fernandez *et al.*, 2015, Vu *et al.*, 2017], et également la décomposition des données neuronales [Williams *et al.*, 2018], ainsi que de nombreuses autres données tensorielles multicanaux [Acar et Yener, 2008, Mørup, 2011].

Formellement, étant donné un tenseur non négatif $\mathcal{X}$ d'ordre 3 avec une taille $I \times J \times K$, la décomposition CP non-négative (NCP) consiste à résoudre le problème de minimisation suivant :

$$\begin{aligned} \min_{\mathbf{A},\mathbf{B},\mathbf{C};\boldsymbol{\lambda}} \quad & \|\mathcal{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C};\boldsymbol{\lambda}]\!]\|_F^2 \\ \mathbf{s.t.} \quad & \mathbf{A} \geq 0, \mathbf{B} \geq 0, \mathbf{C} \geq 0 \end{aligned} \tag{5.2}$$

Comme indiqué auparavant, les colonnes des matrices de facteurs sont incluses dans un seul vecteur $\mathbf{x} = vec\{[\mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T]\}$, de sorte que l'objectif à minimiser a la forme suivante :

$$\mathcal{F}(\mathbf{x}) = \underbrace{f(\mathbf{x})}_{Fidelity} + \underbrace{g(\mathbf{x})}_{Regularization} . \tag{5.3}$$

Et qui peut être explicitement écrit comme :

$$\mathcal{F}(\mathbf{x}) = \|\mathbf{x} - \widehat{\mathbf{x}}_k\|_F^2 + \eta_n \|\tilde{\mathbf{x}}_{k-1} - \widehat{\mathbf{x}}_k\|_F^2 , \tag{5.4}$$

où $\tilde{\mathbf{x}}_{k-1}$ est la version prédécesseur de $\mathbf{x}_k$ dans l'itération précédente et η_n est le poids qui contrôle la rigueur de la pénalité, qui diminue au cours des itérations.

Selon la formulation de minimisation (5.2), la façon la plus naturelle d'effectuer cette minimisation est via les approches alternées, qui consistent à estimer alternativement les matrices facteurs en mettant à jour chaque matrice individuellement tout en gardant les autres fixes. De cette manière, le problème à résoudre est alors transformé en trois sous-problèmes convexes simples. Une telle approche est à la base des méthodes les plus couramment utilisées pour résoudre la décomposition NCP.

Dans notre démarche, nous avons décidé d'estimer les matrices facteurs d'une manière simultanée (appelée approche All-in-ones). Notez que l'une des conséquences majeures de cette approche est que le problème redevient un problème non convexe [Fu *et al.*, 2020]. A notre connaissance, nous sommes les premiers à fournir des algorithmes proximaux pour estimer simultanément les matrices facteurs de la décomposition polyadique canonique sous contraintes non négatives. Cette approche d'estimation simultanée a été traitée pour la décomposition CP mais sous une contrainte de cohérence des matrices facteurs afin d'assurer l'existence d'une meilleure approximation à faible rang [Nazih *et al.*, 2020] ; une telle approche a également été utilisée sous la contrainte non négative dans [Royer *et al.*, 2011], mais la minimisation dans ce dernier a été effectuée en utilisant la recherche en ligne améliorée (ELS) et le backtracking alternatif avec ELS pour les algorithmes de gradient et quasi-Newton. De plus, ce dernier incorpore explicitement la contrainte non négative des matrices facteurs dans le paramétrage du problème, contrairement à notre proposition dans laquelle nous imposons simplement des entrées positives par projection sur un orthant non négatif.

Le cadre principal pour la résolution de la NCP est le block coordinate descent (BCD) [Bertsekas et Scientific, , Xu et Yin, 2013, Kim *et al.*, 2014], qui est une méthode populaire pour résoudre le problème de décomposition tensorielle à grande échelle. Dans le cadre de la BCD, la décomposition tensorielle est optimisée en mettant à jour chaque matrice de facteurs de manière alternée en tant que sous-problème. Nous avons étudié quatre méthodes d'optimisation populaires et récentes pour résoudre le sous-problème de la décomposition tensorielle, que nous décrivons dans la section suivante.

5.4 Les méthodes d'optimisation pour la CP décomposition

De nombreuses méthodes d'optimisation ont été proposées pour résoudre le sous-problème à la fois dans le NCP et le NMF. [Lee, 1999, Lee *et al.*, 2001] ont proposé la méthode de mise à jour multiplicative (MU), qui est la méthode la plus populaire et la plus largement appliquée pour la NMF. [Cichocki et Phan, 2009, Cichocki *et al.*, 2009] ont proposé la méthode des moindres carrés alternatifs hiérarchiques (HALS) pour les problèmes à grande échelle. [Xu et Yin, 2013] ont proposé la méthode du gradient proximal alternatif (APG) avec des preuves de convergence mathématiques détaillées. L'idée similaire à APG a également été proposée pour NMF dans [Guan *et al.*, 2012] et pour

NCP dans [Zhang *et al.*, 2016]. Récemment, la méthode des multiplicateurs à direction alternée (ADMM) est devenue populaire [Boyd *et al.*, 2011, Huang *et al.*, 2016]. En outre, la méthode des moindres carrés non négatifs alternatifs (ANLS) a été analysée en profondeur dans l'article fondamental de Lin avec de fortes propriétés d'optimisation [Lin, 2007], qui a une influence significative sur la NMF. Une forme générale de la méthode ANLS est la programmation quadratique alternée non négative, que l'on peut abréger en ANQP.

Lorsque la méthode ANLS/ANQP est utilisée, les sous-problèmes apparaissent comme des problèmes de moindres carrés non négatifs (NNLS). De nombreuses méthodes efficaces ont été consacrées à la résolution des sous-problèmes NNLS, telles que la méthode du gradient projeté de Lin [Lin, 2007], la méthode quasi-Newton [Zdunek et Cichocki, 2006, Kim et Park, 2007], la méthode de l'ensemble actif [Kim et Park, 2008], la méthode du pivot principal par blocs [Kim et Park, 2011] et le réseau neuronal à projection inertielle [Dai *et al.*, 2019].

Les algorithmes les plus utilisés pour le calcul de la décomposition NCP consiste à résoudre le problème de minimisation suivant :

$$\begin{aligned} \min_{\mathbf{A}^{(n)}} F_{prox}(\mathbf{A}^{(n)}) = \quad & \frac{1}{2}\|\mathbf{X}_{(n)} - \mathbf{A}^{(n)}(\mathbf{B}^{(n)})^T\|_F^2 + \frac{\alpha_{(n)}}{2}\|\mathbf{A}^{(n)} - \hat{\mathbf{A}}^{(n)}\|_F^2 \\ \textbf{s.t.} \quad & \mathbf{A}^{(n)} \geq 0. \end{aligned} \tag{5.5}$$

Dans cette étude, nous utilisons quatre méthodes d'optimisation populaires pour résoudre le problème NCP, notamment MU, ANQP, HALS et APG. Toutes ces méthodes sont soigneusement adaptées au problème NCP.

Les règles de mise à jour de ces quatre algorithmes sont définies comme suit :

▷ **La règle de mise à jour de l'algorithme MU**

$$[\mathbf{A}^{(n)}]_{ir} = [\mathbf{A}^{(n)}]_{ir} \frac{[\mathbf{X}_{(n)}\mathbf{B}^{(n)}]_{(i,r)}}{[\mathbf{A}^{(n)}(\mathbf{B}^{(n)})^T\mathbf{B}^{(n)}]_{(i,r)}}$$

▷ **La règle de mise à jour de l'algorithme ANQP-BPP**

$$\begin{aligned} &\mathbf{X}_{(n)}\mathbf{B}^{(n)} \leftarrow \mathbf{X}_{(n)}\mathbf{B}^{(n)} + \alpha_n \hat{\mathbf{A}}^{(n)}; \\ &(\mathbf{B}^{(n)})^T\mathbf{B}^{(n)} \leftarrow (\mathbf{B}^{(n)})^T\mathbf{B}^{(n)} + +\alpha_n \mathbf{I}_R; \\ &\mathbf{A}^{(n)} = \min_{\mathbf{A}^{(n)} \geq 0} F_{prox}(\mathbf{A}^{(n)}) \\ &= NNQP\_BPP\Big(\mathbf{X}_{(n)}\mathbf{B}^{(n)}, (\mathbf{B}^{(n)})^T\mathbf{B}^{(n)}\Big) \end{aligned}$$

▷ **La règle de mise à jour de l'algorithme HALS**

$$[\mathbf{A}^{(n)}]_{(:,r)} = \left[[\mathbf{A}^{(n)}]_{(:,r)} + \frac{[\mathbf{X}_{(n)}\mathbf{B}^{(n)}]_{(:,r)} - \mathbf{A}^{(n)}[(\mathbf{B}^{(n)})^T\mathbf{B}^{(n)}]_{(:,r)}}{[(\mathbf{B}^{(n)})^T\mathbf{B}^{(n)}]_{(r,r)} + \alpha_n}\right]_+$$

▷ **La règle de mise à jour de l'algorithme APG**

$$\hat{\mathbf{A}}^{(n)} \textit{ est le point extrapole, } \hat{\mathbf{G}}^{(n)} \textit{ est le gradient, et}$$
$$\mathbf{L}^{(n)} \textit{ est une constante de Lipschitz.}$$
$$\mathbf{A}^{(n)} = \max\left(0, \hat{\mathbf{A}}^{(n)} - \frac{\hat{\mathbf{G}}^{(n)}}{\mathbf{L}^{(n)}}\right)$$

5.5 Méthode du gradient proximal accélérée

Dans cette deuxième étude, nous visons à étendre l'APG de Beck et Teboulle [Beck et Teboulle, 2009b, Beck et Teboulle, 2009a] pour résoudre le problème général de la non convexité (4.37). L'APG commence par extrapoler un point $\mathbf{y}_k$ en combinant le point actuel et le point précédent, puis résout un problème de cartographie proximale.

Lorsque l'on étend l'APG à la programmation non convexe, la principale difficulté réside dans le point $\mathbf{y}_k$ extrapolé. Nous avons peu de restrictions sur $f(\mathbf{y}_k)$ lorsque la convexité est absente. En fait, $f(\mathbf{y}_k)$ peut être arbitrairement plus grand que $f(\mathbf{x}_k)$ lorsque $\mathbf{y}_k$ est une mauvaise extrapolation, surtout lorsque f est oscillatoire. Lorsque $\mathbf{x}_{k+1}$ est calculé par une cartographie proximale à un mauvais $\mathbf{y}_k$, $f(\mathbf{x}_{k+1})$ peut également être arbitrairement plus grand que $f(\mathbf{x}_k)$. L'APG monotone de Beck et Teboulle [Beck et Teboulle, 2009b] assure $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$. Cependant, cela ne suffit pas pour assurer la convergence vers les points critiques. Pour remédier à ce problème, nous introduisons un moniteur satisfaisant à la propriété de descente suffisante pour éviter une mauvaise extrapolation de $\mathbf{y}_k$ et la corriger ensuite par ce moniteur.

5.5.1 APG dans le cas convexe

Nous analysons d'abord l'APG dans le cas convexe. Bech et Teboulle [Beck et Teboulle, 2009a] étendent la méthode du gradient accéléré de Nesterov au cas non différentiable. Elle est appelée méthode du gradient proximal accéléré et comprend les étapes suivantes :

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1} - 1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}) \tag{5.6}$$
$$\mathbf{x}_{k+1} = \mathbf{prox}_{\rho_k g}(\mathbf{y}_k - \rho_k \nabla h(\mathbf{y}_k)) \tag{5.7}$$
$$t_{k+1} = \frac{\sqrt{4(t_k)^2 + 1} + 1}{2} \tag{5.8}$$

Où l'operateur proximale est définie comme $\mathbf{prox}_{\rho g}(\mathbf{x}) = \mathbf{argmin}\ h(\mathbf{u}) + \frac{1}{2\rho}\|\mathbf{x} - \mathbf{u}\|^2$. L'APG n'est pas un algorithme monotone, ce qui signifie que $f(\mathbf{x}_{k+1})$ ne peut être inférieur à $f(\mathbf{x}_k)$. Beck et Teboulle [Beck et Teboulle, 2009b] ont donc proposé un algorithme APG

monotone, qui comprend les étapes suivantes :

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}) \tag{5.9}$$

$$\mathbf{z}_{k+1} = \mathbf{prox}_{\rho_k g}(\mathbf{y}_k - \rho_k \nabla h(\mathbf{y}_k)) \tag{5.10}$$

$$t_{k+1} = \frac{\sqrt{4(t_k)^2+1}+1}{2} \tag{5.11}$$

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{Si } f(\mathbf{z}_{k+1}) < f(\mathbf{x}_k) \\ \mathbf{x}_k & \text{sinon.} \end{cases} \tag{5.12}$$

5.6 APG dans le cas non convexe

Dans cette section, nous présentons deux algorithmes de type APG pour les problèmes généraux non convexes et non différentiables.

5.6.1 Contribution 4

Nous donnons deux raisons qui expliquent la difficulté de l'analyse de la convergence APG classique [Beck et Teboulle, 2009b, Beck et Teboulle, 2009a] pour les programmes non convexes : (1) $\mathbf{y}_k$ peut être une mauvaise extrapolation, (2) dans [Beck et Teboulle, 2009b], seule la propriété de descente, $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$, est assurée. Pour résoudre ces problèmes, nous devons surveiller et corriger $\mathbf{y}_k$ lorsqu'il a le potentiel d'échouer, et le moniteur doit jouir de la propriété de descente suffisante qui est essentielle pour assurer la convergence vers un point critique. Comme on le sait, les méthodes de gradient proximal peuvent assurer une descente suffisante [Attouch *et al.*, 2013] (cf. (15)). Nous utilisons donc une étape de gradient proximal comme moniteur. Plus particulièrement, notre algorithme se compose des étapes suivantes :

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}) \tag{5.13}$$

$$\mathbf{z}_{k+1} = \mathbf{prox}_{\rho_x g}(\mathbf{y}_k - \rho_k \nabla h(\mathbf{y}_k)) \tag{5.14}$$

$$\mathbf{v}_{k+1} = \mathbf{prox}_{\rho_z g}(\mathbf{x}_k - \rho_k \nabla h(\mathbf{x}_k)) \tag{5.15}$$

$$t_{k+1} = \frac{\sqrt{4(t_k)^2+1}+1}{2} \tag{5.16}$$

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{Si } f(\mathbf{z}_{k+1}) < f(\mathbf{v}_{k+1}) \\ \mathbf{v}_{k+1} & \text{sinon.} \end{cases} \tag{5.17}$$

où ρ_x et ρ_z peuvent être des constantes fixes satisfaisant $\rho_x < \frac{1}{L}$ et $\rho_z < \frac{1}{L}$, ou calculées dynamiquement par la méthode de recherche linéaire *Backtracking*. L est la constante de Lipschitz de ∇h.

Cet algorithme est une extension de l'APG monotone de Beck et Teboulle [Beck et Teboulle, 2009b]. La différence réside dans le $\mathbf{v}$ additionnel, qui joue le rôle de

moniteur, et l'étape de correction de x-update. Dans (5.12) $f(\mathbf{z}_{k+1})$ est comparé à $f(\mathbf{x}_k)$, tandis que dans (5.17) $f(\mathbf{z}_{k+1})$ est comparé à $f(\mathbf{v}_{k+1})$. Une autre différence est que l'algorithme de Beck et Teboulle assure seulement la descente alors que dans cette version l'algorithme assure une descente suffisante, ce qui signifie :

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - \delta\|\mathbf{v}_{k+1} - \mathbf{x}_k\|^2, \tag{5.18}$$

où $\delta > 0$ est une faible constante. Il n'est pas difficile de comprendre que seule la propriété de descente ne peut assurer la convergence vers un point critique dans une programmation non convexe. Nous présentons notre résultat de convergence dans le théorème suivant

Théorème *(Li et Lin, 2015), Thm. 1 :* Soient h une fonction non convexe avec des gradients continus Lipschitz et g une fonction semi-continue inférieure non convexe et non différentiable, supposons que $f(\mathbf{x})$ est coercitif. Alors $\{\mathbf{x}_k\}$ et $\{\mathbf{v}_k\}$ générés par (5.13)-(5.17) sont bornés. Si $\mathbf{x}^*$ est un point d'accumulation quelconque de $\{\mathbf{x}_k\}$, nous avons $0 \in \partial f(\mathbf{x}^*)$, c'est-à-dire que $\mathbf{x}^*$ est un point critique.

Un aspect remarquable de cet algorithme est que, bien que des modifications aient été apportées à l'algorithme de Beck et Teboulle, le taux de convergence $O(\frac{1}{k^2})$ dans le cas de la non convexité est toujours valable.

Le calcule du proximal opérateur : En vue de notre fonction contrainte dans (5.4), nous avons pu calculer l'opérateur proximal exact de g, qui se définit comme :

$$\mathbf{prox}_{\rho^{(k)}g}(\mathbf{z}^{(k)}) = \arg\min_{\mathbf{x}} \underbrace{(g(\mathbf{x}) + \frac{1}{2\rho^{(k)}}\|\mathbf{x} - \mathbf{z}^{(k)}\|_2^2)}_{\mathcal{H}(\mathbf{x})} \tag{5.19}$$

Pour cela, le gradient de la fonction $\mathcal{H}$ est mis à zéro :

$$\begin{aligned} \nabla\mathcal{H}(\mathbf{x}) = 0 &\implies \nabla g(\mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{z}^{(k)}) = 0 \\ &\implies -2 * \eta_n(\tilde{\mathbf{x}}_{k-1} - \mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{z}^{(k)}) = 0, \end{aligned} \tag{5.20}$$

et par un calcul plus simple, la forme analytique du proximal operateur pour notre fonction régularisée g serait de la forme fermée suivante :

$$\mathbf{prox}_{\rho^{(k)}g}(\mathbf{z}^{(k)}) = \frac{\frac{1}{\rho^{(k)}}\mathbf{z}^{(k)} + 2 * \eta_n\tilde{\mathbf{x}}_{k-1}}{\frac{1}{\rho^{(k)}} + 2 * \eta_n}. \tag{5.21}$$

Pour une meilleure référence, nous résumons l'algorithme APG monotone dans l'algorithme 5.1 pour minimiser (5.4).

Algorithme 5.1 Méthode monotone de gradient proximal accéléré (mnAPG) pour la décomposition NCP.

Initialiser : $\mathbf{z}_1 = \mathbf{x}_1 = \mathbf{x}_0$, $t_0 = 0$, $t_1 = 1$, $\rho_y > \frac{1}{L}$ et $\rho_x > \frac{1}{L}$.
Pour $k = 1, ..., m$, **faire** :
$y_k = x_k + \frac{t_{k-1}}{t_k}(z_k - x_k) + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$.
Calculer $\mathbf{z}_{k+1}$ comme étant $\mathbf{z}_{k+1} = \mathbf{prox}_{\gamma g}(\mathbf{y}_k - \rho_z \nabla h(\mathbf{y}_k))$
Calculer $\mathbf{v}_{k+1}$ comme étant $\mathbf{v}_{k+1} = \mathbf{prox}_{\gamma g}(\mathbf{x}_k - \rho_x \nabla h(\mathbf{x}_k))$
Projection $\mathbf{z}_{k+1} = \mathbf{max}(0, \mathbf{z}_{k+1})$ et $\mathbf{v}_{k+1} = \mathbf{max}(0, \mathbf{v}_{k+1})$
$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{Si } \mathcal{F}(\mathbf{z}_{k+1}) < \mathcal{F}(\mathbf{v}_{k+1}) \\ \mathbf{v}_{k+1} & \text{sinon.} \end{cases}$
$t_{k+1} = \frac{\sqrt{4t_k^2+1}+1}{2}$,
Extraire les trois blocs de $\mathbf{X}_{k+1}$: $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ et $\mathbf{C}_{k+1}$
Normaliser les colonnes de $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ et $\mathbf{C}_{k+1}$
Fin Pour
Retourner $\mathbf{A}^{(k+1)}$, $\mathbf{B}^{(k+1)}$ et $\mathbf{C}^{(k+1)}$.

5.6.2 Contribution 5

L'algorithme 5.1 est un algorithme monotone. Lorsque le problème est mal posé, un algorithme monotone doit se traîner au fond d'une étroite vallée incurvée pour que la valeur de la fonction objectif n'augmente pas, ce qui entraîne des pas courts ou même des zigzags et donc une convergence lente [Zhang et Hager, 2004]. La suppression de l'exigence de monotonie peut améliorer la vitesse de convergence car des pas plus grands peuvent être adoptés lors de la recherche linéaire.

D'autre part, dans l'algorithme 5.1, nous devons calculer $\mathbf{z}_{k+1}$ et $\mathbf{v}_{k+1}$ à chaque itération et utiliser $\mathbf{v}_{k+1}$ pour contrôler et corriger $\mathbf{z}_{k+1}$. Il s'agit d'une stratégie conservatrice et coûteuse. En fait, nous pouvons accepter $\mathbf{z}_{k+1}$ comme $\mathbf{x}_{k+1}$ directement s'il satisfait à un certain critère montrant que $\mathbf{z}_k$ est une bonne extrapolation. Dans ce cas, $\mathbf{v}_{k+1}$ n'est calculé que si ce critère n'est pas satisfait. Ainsi, nous pouvons réduire le nombre moyen des proximales opérateurs, en fonction du coût de calcul, à chaque itération. Dans cette sous-section, nous introduisons un APG non monotone pour accélérer la convergence.

Dans la APG monotone, (5.18) est assurée. Dans les APG non monotones, nous autorisons à $\mathbf{x}_{k+1}$ de faire une valeur de fonction objective plus grande que $f(\mathbf{x}_k)$. Plus précisément, nous permettons à $\mathbf{x}_{k+1}$ de produire une valeur de fonction objective inférieure à c_k, une relaxation de $f(\mathbf{x}_k)$. ck ne doit pas être trop éloigné de F(xk). Ainsi la moyenne de $f(\mathbf{x}_k)$,$f(\mathbf{x}_{k-1})$,...,$f(\mathbf{x}_1)$ est un bon choix. Nous suivons donc [Zhang et Hager, 2004] pour définir c_k comme une combinaison convexe de $f(\mathbf{x}_k)$,$f(\mathbf{x}_{k-1})$,...,$f(\mathbf{x}_1)$ avec des poids décroissants de façon exponentielle :

$$c_k = \frac{\sum_{j=1}^{k} \eta^{k-j} f(\mathbf{x}_j)}{\sum_{j=1}^{k} \eta^{k-j}}, \tag{5.22}$$

où $\eta \in [0, 1)$ contrôle le degré de non-monotonicité. En pratique, c_k peut être calculé efficacement par la récursion suivante :

$$q_{k+1} = \eta q_k + 1, \tag{5.23}$$

$$c_{k+1} = \frac{\eta q_k c_k + f(\mathbf{x}_{k+1})}{q_{k+1}}, \tag{5.24}$$

où $q_1 = 1$ et $c_1 = f(\mathbf{x}_1)$.

Selon (5.17), on peut diviser (5.18) en deux parties par les différents choix de $\mathbf{x}_{k+1}$. En conséquence, dans l'APG non monotone, nous considérons que les deux conditions suivantes doivent être remplacées (5.18) :

$$f(\mathbf{z}_{k+1}) \leq c_k - \delta \|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2, \tag{5.25}$$

$$f(\mathbf{v}_{k+1}) \leq c_k - \delta \|\mathbf{v}_{k+1} - \mathbf{y}_k\|^2, \tag{5.26}$$

Nous avons choisi (5.25) comme critères mentionnés précédemment. Lorsque le critère (5.25) est retenu, nous considérons que $\mathbf{y}_k$ est une bonne extrapolation et nous acceptons directement $\mathbf{z}_{k+1}$. Dans ce cas, nous ne calculons pas $\mathbf{v}_{k+1}$. Cependant, (5.25) ne tient pas tout le temps. Lorsqu'il échoue, nous considérons que $\mathbf{y}_k$ peut ne pas être une bonne extrapolation. Dans ce cas, nous calculons $\mathbf{v}_{k+1}$ par (5.15) satisfaisant (5.26), puis nous contrôlons et corrigeons $\mathbf{z}_{k+1}$ par (5.17). (5.25) est assuré lorsque $\rho_x \leq \frac{1}{L}$. Lorsqu'on utilise la méthode de recherche linéaire *backtracking*, un $\mathbf{v}_{k+1}$ satisfaisant (5.25) peut être trouvé par étapes finies.

En combinant (5.23), (5.24), (5.25) et $\mathbf{x}_{k+1} = \mathbf{z}_{k+1}$, on obtient :

$$c_{k+1} \leq c_k - \frac{\delta \|\mathbf{x}_{k+1} - \mathbf{y}_k\|^2}{q_{k+1}}, \tag{5.27}$$

De même, en remplaçant (5.25) et $\mathbf{x}_{k+1} = \mathbf{z}_{k+1}$ par (5.26) et $\mathbf{x}_{k+1} = \mathbf{v}_{k+1}$, respectivement, nous avons :

$$c_{k+1} \leq c_k - \frac{\delta \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2}{q_{k+1}}, \tag{5.28}$$

Cela signifie que nous remplaçons la condition de descente suffisante de $f(\mathbf{x}_k)$ dans (5.18) par la descente suffisante de c_k.

Nous résumons l'APG non monotone dans l'algorithme 5.2. Tout comme l'APG monotone, l'APG non monotone possède également la propriété de convergence dans le cas non convexe et le taux de convergence $O(\frac{1}{k^2})$ dans le cas convexe.

Nous présentons le résultat de la convergence dans le Théorème 5.6.2. Le théorème **??** est toujours valable pour l'algorithme 5.2 sans modification. Par conséquent, nous l'omettons ici.

On définit $\Omega_1 = \{k_1, k_2, ..., k_j, ...\}$ et $\Omega_2 = \{m_1, m_2, ..., m_j, ...\}$, comme dans l'algorithme 5.2, (5.25) tient et $\mathbf{x}_{k+1} = \mathbf{z}_{k+1}$ est réalisé pour tous $k = k_j \in \Omega_1$. Pour tous les $k = m_j \in \Omega_2$, (5.25) ne tient pas et (5.17) est réalisé. Ensuite, nous avons $\Omega_1 \cap \Omega_1 = \emptyset$, $\Omega_1 \cup \Omega_1 = \{1, 2, 3, ..., \}$ et le théorème suivant tient.

Théorème Soient h une fonction non convexe avec des gradients continus Lipschitz et g une fonction semi-continue inférieure non convexe et non différentiable, supposons que $f(\mathbf{x})$ est coercitif. Alors $\{\mathbf{x}_k\}$ et $\{\mathbf{v}_k\}$ où $k_j \in \omega_1$ générés par l'algorithme 5.2 sont bornés, et

— Si Ω_1 et Ω_2 est fini, alors pour tout point d'accumulation $\{\mathbf{x}^*\}$ de $\{\mathbf{x}_k\}$, nous avons $0 \in \partial f(\mathbf{x}^*)$.

— Si Ω_1 et Ω_2 sont tous deux infinis, alors pour tout point d'accumulation $\mathbf{x}^*$ de $\{\mathbf{x}_{k_j+1}\}$, $\mathbf{y}^*$ de $\{\mathbf{y}_{k_j}\}$ où $k_j \in \Omega_1$ et tout point d'accumulation $\mathbf{v}^*$ de $\{\mathbf{v}_{m_j+1}\}$, $\mathbf{x}^*$ de $\{\mathbf{x}_{m_j}\}$ où $m_j \in \Omega_2$, nous avons $0 \in \partial f(\mathbf{x}^*)$, $0 \in \partial f(\mathbf{y}^*)$ et $0 \in \partial f(\mathbf{v}^*)$.

Pour une meilleure référence, nous résumons l'algorithme APG non monotone dans l'algorithme 5.2 pour minimiser (5.4).

5.7 Simulations

Dans cette section, nous illustrons quelques résultats expérimentaux pour analyser les performances des algorithmes proposés avec les méthodes les plus populaires et les plus largement appliquées dans la littérature pour la décomposition CP non négative.

Une comparaison est effectuée avec les algorithmes conventionnels suivants : Multiplicative Updating (MU) [Gillis et Glineur, 2012], Hierarchical Alternative Least Squares (HALS) [Cichocki *et al.*, 2009], Nonnegative Alternative Least Squares with Frobenius norm regularization based on block principal pivoting (ANLS-BPP) [Kim et Park, 2008] et Alternating Proximal Gradient (PG alternating) [Wang *et al.*, 2019].

Par ailleurs, nous mesurons la performance de chaque algorithme en fonction de trois critères, à savoir la précision, le temps CPU et la meilleure somme de congruence, qui représente un critère significatif pour comparer deux tenseurs de rang $R > 1$ [Sahnoun et Comon, 2015].

Afin d'obtenir des résultats comparables, tous les algorithmes sont initialisés avec les

Algorithme 5.2 Méthode non monotone de gradient proximal accéléré (nmAPG) pour la décomposition NCP.

Initialiser : $\mathbf{z}_1 = \mathbf{x}_1 = \mathbf{x}_0$, $t_0 = 0$, $t_1 = 1$, $\eta \in [0, 1)$, $c_1 = f(\mathbf{x}_1)$, $q_1 = 1$, $\rho_y > \frac{1}{L}$ et $\rho_x > \frac{1}{L}$.
Pour $k = 1, 2, 3, ...,$ **faire** :
$y_k = x_k + \frac{t_{k-1}}{t_k}(z_k - x_k) + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$.
Calculer $\mathbf{z}_{k+1}$ comme étant $\mathbf{z}_{k+1} = \mathbf{prox}_{\gamma g}(\mathbf{y}_k - \rho_z \nabla h(\mathbf{y}_k))$
Projection $\mathbf{z}_{k+1} = \mathbf{max}(0, \mathbf{z}_{k+1})$
Si $f(z_{k+1}) \leq c_k - \delta \|z_{k+1} - y_k\|^2$
Alors
$\mathbf{x}_{x+1} = \mathbf{z}_{x+1}$
Sinon
Calculer $\mathbf{v}_{k+1}$ comme étant $\mathbf{v}_{k+1} = \mathbf{prox}_{\gamma g}(\mathbf{x}_k - \rho_x \nabla h(\mathbf{x}_k))$
Projection $\mathbf{v}_{k+1} = \mathbf{max}(0, \mathbf{v}_{k+1})$
$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{Si } \mathcal{F}(\mathbf{z}_{k+1}) < \mathcal{F}(\mathbf{v}_{k+1}) \\ \mathbf{v}_{k+1} & \text{sinon.} \end{cases}$$
Fin Si
$t_{k+1} = \frac{\sqrt{4t_k^2+1}+1}{2}$,
$q_{k+1} = \eta q_k + 1$,
$c_{k+1} = \frac{\eta q_k c_k + f(\mathbf{x}_{k+1})}{q_{k+1}}$,
Extraire les trois blocs de $\mathbf{X}_{k+1}$: $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ et $\mathbf{C}_{k+1}$
Normaliser les colonnes de $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ et $\mathbf{C}_{k+1}$
Fin Pour
Retourner $\mathbf{A}^{(k+1)}$, $\mathbf{B}^{(k+1)}$ et $\mathbf{C}^{(k+1)}$.

mêmes points initiaux qui sont générés avec des nombres aléatoires positifs uniformément distribués. De plus, et afin de visualiser clairement le comportement de chaque algorithme, nous ne considérons que le nombre maximum d'itérations, qui est fixé à 600.

Dans toutes les expériences, les résultats sont obtenus à partir de 50 exécutions de Monte Carlo.

5.7.1 Expérience 1

Dans cette expérience, nous générons un modèle NCP aléatoire de taille $3 \times 4 \times 6$ de rang 3 et avec des cohérences $0,4 \leq \mu(\mathbf{A}) \leq 0,6$, $0,4 \leq \mu(\mathbf{B}) \leq 0,6$ et $0,4 \leq \mu(\mathbf{C}) \leq 0,6$. On fait varier η par itérations. Plus précisément dans cette première expérience, η est initialisé à 1, et est divisé par 100 lorsque $f(x)$ est réduit de moins de 10^{-4}.

La figure 5.2 présente l'erreur de reconstruction en fonction du nombre d'itérations. On peut donc observer que les algorithmes de gradient proximal accéléré non monotone (Nm-APG) et de gradient proximal accéléré monotone (M-APG) sont plus précis que les autres algorithmes en termes de nombre d'itérations, suivis de l'algorithme de gradient proximal alternatif (PG alternatif), puis de l'algorithme ANLS-BPP, et enfin des algorithmes MU et HALS.

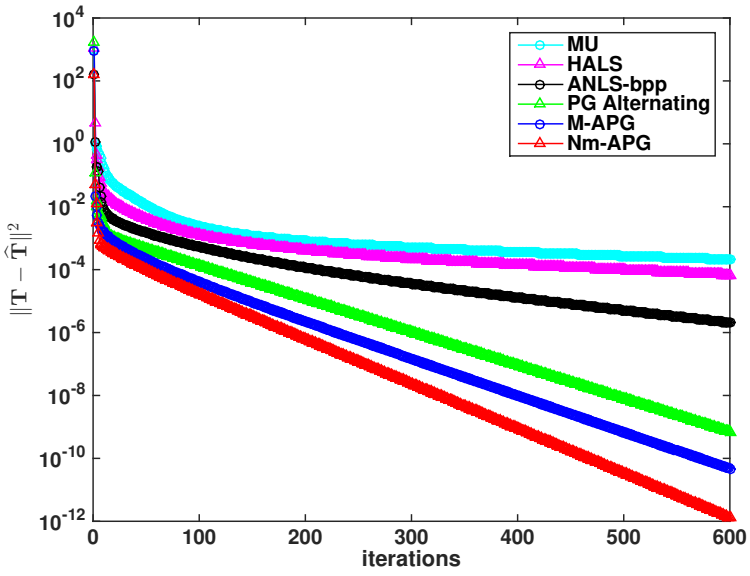


FIGURE 5.2 – Erreur de reconstruction en fonction du nombre d'itérations. Pour l'expérience 1.

Afin de présenter une comparaison plus approfondie entre les algorithmes, nous explorons le critère du temps CPU pour mettre en évidence le temps d'exécution de chaque algorithme. Le tableau 5.1 indique que l'algorithme M-APG nécessite plus de temps machine par rapport aux autres algorithmes, cet inconvénient se fait au détriment de la

précision et il est attendu en raison de la complexité élevée du contrôle utilisé dans l'algorithme M-APG qui demande le calcul d'un proximal supplémentaire à chaque itération. D'autre part, on peut clairement comprendre la réduction du temps dans sa version non monotone qui évite ce calcul du proximal supplémentaire à chaque itération.

TABLE 5.1 – Performances des algorithmes NCP pour l'expérience 1.

Algorithmes	Erreur absolue	Temps	Congruence
MU	2.1240e-04	2.66	0.95
HALS	6.6760e-05	2.60	0.95
ANLS-bpp	2.0701e-06	4.30	0.97
PG alternating	6.8699e-10	3.77	0.99
M-APG	4.6690e-11	7.04	0.99
Nm-APG	1.2813e-12	5.25	0.99

5.7.2 Expérience 2

Dans cette deuxième expérience, nous générons un modèle NCP aléatoire de taille $4 \times 3 \times 10$ de rang 4 et avec des cohérences $\mu(\mathbf{A}) = 0,9$, $\mu(\mathbf{B}) = 0,6$, $\mu(\mathbf{C}) = 0,6$. Cette configuration adresse le problème du goulot d'étranglement, où nous avons choisi les facteurs du premier mode (c.-à-d., la matrice $\mathbf{A}$) à être presque colinéaires. Dans cette expérience, η est initialisé à 1.5, et divisé par 10 lorsque $f(x)$ est réduit de moins de 10^{-2}.

La figure 5.3 illustre l'erreur de reconstruction en fonction du nombre d'itérations. A partir de ces résultats, nous pouvons encore observer que les algorithmes MU et HALS produisent des résultats médiocres par rapport aux autres algorithmes. D'autre part, l'algorithme Nm-APG proposé fournit des résultats supérieurs aux autres algorithmes dans le cas du goulot d'étranglement.

En analysant le temps d'exécution de chaque algorithme dans le tableau 5.2, nous pouvons effectivement remarquer que bien que l'algorithme Nm-APG nécessite moins de temps machine que l'algorithme M-APG, les deux algorithmes proposés nécessitent tout de même un peu plus de temps machine que les autres algorithmes. Cependant, il faut noter que notre façon simultanée d'estimer les matrices facteurs a permis d'obtenir une meilleure précision au détriment du temps machine. Il est également important de souligner que dans de nombreuses applications de la décomposition CP non négative, la précision est plus importante que le temps d'exécution , notamment pour la décomposition d'images hyperspectrales ou de données électroencéphalographiques.

D'après ces résultats, nous pouvons conclure que les algorithmes proximaux semblent être des choix appropriés pour le problème de la décomposition CP non négative.

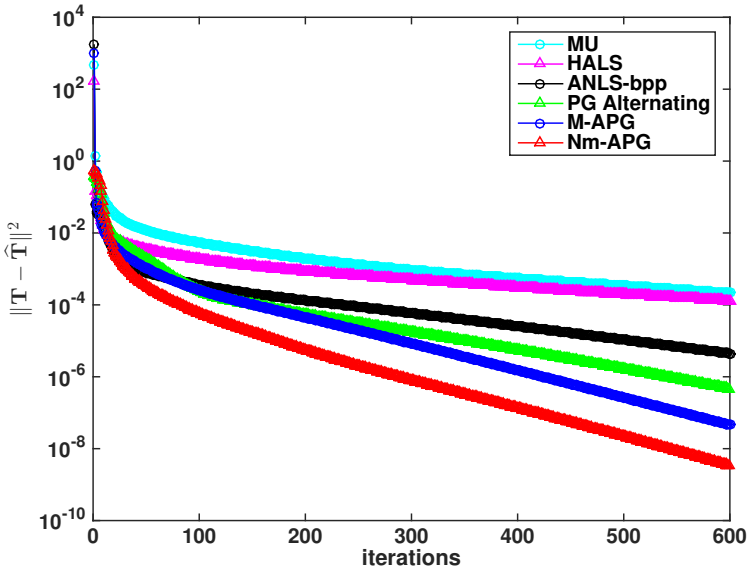


FIGURE 5.3 – Erreur de reconstruction en fonction du nombre d'itérations. Pour l'expérience 2.

TABLE 5.2 – Performances des algorithmes NCP pour l'expérience 2.

Algorithmes	Erreur absolue	Temps	Congruence
MU	2.1417e-04	2.67	0.95
HALS	1.2933e-04	2.55	0.95
ANLS-bpp	4.4461e-06	4.91	0.96
PG alternating	4.6119e-07	3.71	0.99
M-APG	4.5559e-08	8.11	0.99
Nm-APG	3.4007e-09	6.26	0.99

5.8 Bilan du chapitre

Dans ce dernier chapitre, nous avons présenté les approches de factorisation matricielle et tensorielle non négative. Une grande attention a été accordée à la décomposition CP non négative, dont nous avons présenté les cadres algorithmiques et informatiques pour l'analyse et le développement d'algorithmes pour cette décomposition. Nous avons ensuite proposé deux nouveaux algorithmes basés sur des méthodes proximales, plus précisément le gradient proximal accéléré monotone (M-APG) et le gradient proximal accéléré non monotone (Nm-APG), ces algorithmes proposés ont permis l'estimation des matrices facteurs de la décomposition CP non négative selon une approche d'estimation simultanée.

À la lumière des résultats expérimentaux, nous pouvons conclure que les algorithmes

proximaux semblent être des choix appropriés pour le problème de la décomposition CP non négative. De plus, notre choix d'estimer les matrices facteurs de manière simultanée a prouvé son efficacité en termes de précision par rapport à la manière alternée. Néanmoins, la voie alternée reste un bon choix lorsque le temps d'exécution devient une priorité pour certaines autres applications, telles que les données tensorielles multicanaux où l'estimation du multi-code CDMA doit être calculée en temps réel.

Conclusion générale et perspectives

Dans cette thèse, nous avons proposé plusieurs méthodes d'optimisation pour la décomposition tensorielle Canonical polyadic. Cette dernière se révèle très utile dans de nombreuses applications du monde réel en raison de son unicité et de la facilité d'interprétation de ses facteurs.

Dans un premier temps, nous avons fourni quelques éléments fondamentaux de l'algèbre multilinéaire. Ces outils mathématiques offrent une meilleure capacité de modélisation pour des données multi-dimensionnelles. Nous avons mis en évidence le fait que la généralisation des outils bilinéaires aux outils multilinéaires n'est pas triviale. La majorité des opérations entre les matrices se généralisent facilement pour les tenseurs. c'est le cas par exemple de la somme et de la multiplication d'un tenseur avec un réel. Pour le produit scalaire et la norme, la définition s'étend aussi naturellement. Cependant, il en existe d'autres où les matrices et les tenseurs se distinguent. L'une des principales distinctions est la non-extension du théorème d'ECKART-YOUNG au cas tensoriel (tenseur d'ordre > 2). Celle-ci donne lieu à deux définitions distinctes du rang pour un tenseur, nommées "rang multilinéaire" et "rang tensoriel". Ces deux résultats de rang ont donné naissance à deux décompositions tensorielles, lesquelles sont toutes deux dérivées de la généralisation de la décomposition en valeur singulière d'une matrice (SVD). En conséquence, le rang multilinéaire est relié à la décomposition tensorielle qui généralise la modélisation matricielle de la SVD, qui est appelée la décomposition **TUCKER3**, tandis que le rang tensoriel, relié à la décomposition tensorielle, généralise la modélisation canonique de la SVD et qui est appelée la décomposition **Canonical Polyadic** (CP).

Ensuite, nous avons présenté plusieurs décompositions matricielles et tensorielles qui sont importantes dans le contexte de cette thèse. En particulier, nous nous sommes principalement intéressés par les deux décompositions tensorielles les plus connues, à savoir la décomposition de Tucker3 et la décomposition CP. Une plus grande attention a été accordée aux décompositions d'un tenseur d'ordre trois, qui nous ont permis de mieux comprendre et présenter les caractéristiques propres de ces décompositions.

Ainsi, une étude plus approfondie a été consacrée à la décomposition CP, étant donné

qu'elle constitue le principal objectif de toutes nos contributions. Cette décomposition propose de décomposer un tenseur de rang R en une somme de R tenseurs de rang un. Nous avons notamment introduit une décomposition tensorielle basée sur la décomposition CP qui résout le problème de l'indétermination d'échelle. Cette décomposition consiste à calculer la valeur optimale du facteur d'échelle Λ que nous avons isolé et mis en dehors des matrices de facteurs dans la décomposition CP. Nous avons aussi montré que dans cette décomposition, le conditionnement du calcul de la matrice de facteur d'échelle optimale dépend de la cohérence via une matrice de Gram. Cela a des implications très importantes sur l'existence et l'unicité de la décomposition CP. De plus, nous avons donné une condition suffisante qui prouve l'unicité de cette décomposition tensorielle.

Après, nous avons développé plusieurs techniques de calcul de cette décomposition. Une étude détaillée a été portée sur les aspects fondamentaux des algorithmes déterministes standard. Plus précisément, les méthodes mathématiques (Gradient, Gradient conjugué, Levenberg Marquardt, Quasi Newton, BFGS ...), et des améliorations ont été proposées pour les rendre plus performants. L'étude de ces algorithmes mathématiques nous a montré leurs non-fonctionnement sur des problèmes d'optimisation avec des contraintes et la sensibilité de quelques-uns au conditionnement et à l'initialisation.

En vue de notre problème d'optimisation qui est sous une contrainte d'égalité imposée sur les colonnes des matrices de facteurs, il s'est avéré important de faire appel à des algorithmes d'optimisation pour minimiser la nouvelle fonction de coût sous des contraintes d'égalité. Deux algorithmes d'optimisation ont été élaborées : "Algorithme 1" et "Algorithme 2". Ces deux algorithmes sont basés sur la méthode du gradient projeté tout en prenant en compte l'isolement du facteur d'échelle. Ainsi, ces deux algorithmes de type descente de gradient sont beaucoup moins sensibles à l'initialisation que l'ALS et offrent de meilleures propriétés de convergence, cependant ils sont toujours incapables de surmonter les phénomènes de goulot d'étranglement (Bottleneck) ou de marais (Swamp), c'est-à-dire, lorsque les matrices de facteurs dans un ou tous les modes sont colinéaires respectivement. Ensuite, lorsque nous avons également considéré la contrainte sur les cohérences afin de garantir l'existence et l'unicité de la nouvelle fonction de coût sous cette fois une contrainte d'inégalité, nous avons développé trois nouveaux algorithmes qui traitent le problème d'optimisation sous contrainte d'inégalité que nous avons nommés : "Algorithme gradient proximal", "Algorithme gradient proximal accéléré" et "Algorithme explicite implicite" grâce à une formulation de barrière qui nous a permis de transformer le problème d'optimisation avec contrainte d'inégalité en un problème sans contrainte, ce qui nous aussi facilité la recherche de la solution. Les résultats de simulations ont démontré le bon comportement de ces trois algorithmes, ainsi qu'un meilleur compromis entre la précision et la vitesse de convergence par rapport aux autres algorithmes dans la littérature dans le cas normal et surtout dans les cas difficle où les phénomènes de goulot d'étranglement ou de marais se posent.

Dans notre deuxième étude, nous avons effectué l'estimation des matrices facteurs de la décomposition CP non-négative (NCP) en suivant une approche d'estimation simultanée. La conséquence majeure de cette approche est que le problème redevient un problème non convexe. Cet observation nous a encouragés à développer deux autre nouveaux algorithmes basés sur des méthodes proximales, plus précisément "le gradient proximal accéléré monotone (M-APG)" et "le gradient proximal accéléré non monotone (Nm-APG)" pour améliorer les performances des algorithmes de la décomposition NCP. A notre connaissance, aucun algorithme proximal n'a encore été proposé dans la littérature pour l'estimation simultanée des matrices facteurs de la décomposition CP non négative. Ainsi, pour ces algorithmes, nous avons introduit une fonction de régularisation qui pénalise la différence entre les facteurs actuels et précédents, en utilisant deux stratégies différentes capables de contrôler efficacement cette régularisation. Nous avons effectué une comparaison approfondie avec d'autres algorithmes NCP dans la littérature sur la base des simulations numériques, lesquelles ont montré la bonne performance des deux algorithmes proposés par rapport aux autres algorithmes NCP dans la littérature en termes de précision pour des cas normaux ainsi que pour le cas de goulot d'étranglement. Bien que les algorithmes proposés soient très performants en termes de précision, ils sont en revanche un peu plus coûteux en termes de temps machine.

Plusieurs voies peuvent être envisagées dans le sillage de ce travail. Nous en esquissons ici celles qui paraissent les plus pertinentes. Des améliorations importantes sont envisageables par rapport aux méthodes proposées dans le chapitre 4, tant sur le plan théorique que sur le plan applicatif, notamment au niveau de l'extension de ces algorithmes pour des tenseurs d'ordre plus élevé. De plus, les trois algorithmes proposés peuvent aussi servir de base aux algorithmes stochastiques proximaux pour résoudre des problèmes à grande échelle, à condition qu'ils soient traités avec plus de précaution en raison de l'apparition de nouveaux phénomènes indésirables. Un autre point d'intérêt concerne l'application de ces algorithmes pour la séparation aveugle des signaux ainsi que pour les données tensorielles multicanaux, plus particulièrement pour addresser le problème de l'estimation des paramètres dans les systèmes d'accès multiple par répartition en code à séquence directe (DS/CDMA).

Dans le sillon de la méthode Nm-APG présentée dans le chapitre 5, nous pensons paramétrer d'avantage ses degrés de liberté. Tout d'abord, il serait important d'étudier d'autres algorithmes de proximaux à l'instar de celles parallèles et distribués basés sur l'algorithme des directions alternées (ADMM), le principe consiste à diviser la fonction objectif (et les contraintes) en deux termes, dont au moins un est séparable. La séparabilité des termes nous donne la possibilité d'évaluer l'opérateur proximal en parallèle. Un aspect essentiel qui peut être traité serait la conception des algorithmes parallèles et distribués en exploitant les méthodes du proximal gradient ou du proximal gradient accéléré. Un

autre travail d'intérêt pourrait être de tester les deux méthodes proposées N-APG et Nm-APG pour la décomposition d'images hyperspectrales ou pour la décomposition de données électroencéphalographie qui priorisent la précision sur le temps d'exécution.

Dans le cadre d'un co-encadrement d'un nouveau thésard, nous menons actuellement une nouvelle réflexion qui consiste à mettre en oeuvre les méthodes proposées ainsi que la conception de nouveaux algorithmes stochastiques et distribués sur la base des méthodes de direction alternative (ADMM) en vue de la décomposition des données d'électroencéphalographie.

Annexes

ANNEXE

A

ANNEXE 1

A.1 Calcule des matrices de gradient

Les fonctions de coûts pour CP :

$$\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) = \|\mathbf{T}_{(1)}^{I\times KJ} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T\|_F^2$$

$$\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) = \|\mathbf{T}_{(2)}^{J\times KI} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T\|_F^2$$

$$\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) = \|\mathbf{T}_{(3)}^{K\times JI} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T\|_F^2$$

Pour simplifier, posant :

$$\delta_{(1)} = \mathbf{T}_{(1)}^{I\times KJ} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T$$

$$\delta_{(2)} = \mathbf{T}_{(2)}^{J\times KI} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T$$

$$\delta_{(3)} = \mathbf{T}_{(3)}^{K\times JI} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T$$

On a :

$$\|\delta_{(1)}\|_F^2 = \langle\delta_{(1)},\delta_{(1)}\rangle = \mathbf{trace}\{\delta_{(1)}^T\delta_{(1)}\}$$

Le but est d'obtenir :

$$\boxed{d\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) = \langle\tfrac{\partial\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{A}}, d\mathbf{A}\rangle + \langle\tfrac{\partial\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{B}}, d\mathbf{B}\rangle + \langle\tfrac{\partial\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{C}}, d\mathbf{C}\rangle} \quad (A.1)$$

Où $\frac{\partial\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C})}{\partial\mathbf{A}}$ est la dérivée partielle par rapport à $\mathbf{A}$.

En utilisant les propriétés de la fonction trace dans le chapitre 2, on aura :

$d\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) = d\mathbf{trace}\{\delta_{(1)}^T\delta_{(1)}\} = \mathbf{trace}\{d(\delta_{(1)}^T\delta_{(1)})\}$
$= \mathbf{trace}\{d(\delta_{(1)}^T)\delta_{(1)} + \delta_{(1)}^T d(\delta_{(1)}) = \mathbf{trace}\{d(\delta_{(1)}^T)\delta_{(1)}\} + \mathbf{trace}\{\delta_{(1)}^T d(\delta_{(1)})\}$
$= 2\mathbf{trace}\{\delta_{(1)}^T d(\delta_{(1)})\} = 2\mathbf{trace}\{\delta_{(2)}^T d(\delta_{(2)})\} = 2\mathbf{trace}\{\delta_{(3)}^T d(\delta_{(3)})\}$
$= 2\mathbf{trace}\{-\delta_{(1)}^T d\mathbf{A}(\mathbf{C}\odot\mathbf{B})^T - \delta_{(2)}^T d\mathbf{B}(\mathbf{C}\odot\mathbf{A})^T - \delta_{(3)}^T d\mathbf{C}(\mathbf{B}\odot\mathbf{A})^T\}$

$$= \mathbf{trace}\{-2(\mathbf{C}\odot\mathbf{B})^T\delta_{(1)}^T d\mathbf{A}\} + \mathbf{trace}\{-2(\mathbf{C}\odot\mathbf{A})^T\delta_{(2)}^T d\mathbf{B}\} + \mathbf{trace}\{-2(\mathbf{B}\odot\mathbf{A})^T\delta_{(3)}^T d\mathbf{C}\}$$
$$= \mathbf{trace}\{-2(\delta_{(1)}\mathbf{C}\odot\mathbf{B})^T d\mathbf{A}\} + \mathbf{trace}\{-2(\delta_{(2)}\mathbf{C}\odot\mathbf{A})^T d\mathbf{B}\} + \mathbf{trace}\{-2(\delta_{(3)}\mathbf{B}\odot\mathbf{A})^T d\mathbf{C}\}$$
$$= \langle -2(\delta_{(1)}\mathbf{C}\odot\mathbf{B}), d\mathbf{A}\rangle + \langle -2(\delta_{(2)}\mathbf{C}\odot\mathbf{A}), d\mathbf{B}\rangle + \langle -2(\delta_{(3)}\mathbf{B}\odot\mathbf{A}), d\mathbf{C}\rangle$$

Par identification avec A.1 on trouve :

$$\begin{aligned}\nabla_{\mathbf{A}}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= 2[-\mathbf{T}_{(1)}^{I\times KJ} + \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T](\mathbf{C}\odot\mathbf{B}) \\ &= 2(-\mathbf{T}_{(1)}^{I\times KJ}(\mathbf{C}\odot\mathbf{B}) + \mathbf{A}(\mathbf{C}^T\mathbf{C}) * (\mathbf{B}^T\mathbf{B}))\end{aligned}$$

$$\begin{aligned}\nabla_{\mathbf{B}}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= 2[-\mathbf{T}_{(2)}^{J\times KI} + \mathbf{B}(\mathbf{C}\odot\mathbf{A})^T](\mathbf{C}\odot\mathbf{A}) \\ &= 2(-\mathbf{T}_{(2)}^{J\times KI}(\mathbf{C}\odot\mathbf{A}) + \mathbf{B}(\mathbf{C}^T\mathbf{C}) * (\mathbf{A}^T\mathbf{A}))\end{aligned}$$

$$\begin{aligned}\nabla_{\mathbf{C}}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= 2[-\mathbf{T}_{(3)}^{K\times JI} + \mathbf{C}(\mathbf{B}\odot\mathbf{A})^T](\mathbf{B}\odot\mathbf{A}) \\ &= 2(-\mathbf{T}_{(3)}^{K\times JI}(\mathbf{B}\odot\mathbf{A})) + \mathbf{C}(\mathbf{B}^T\mathbf{B}) * (\mathbf{A}^T\mathbf{A}))\end{aligned}$$

A.2 Détail de calcul de Λ optimale

Notre but est de calculer la valeur optimale de Λ qui minimise l'expression suivante :

$$\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C};\mathbf{\Lambda}) = \|\mathcal{T} - (\mathbf{A},\mathbf{B},\mathbf{C}.\mathbf{\Lambda})\|_F^2 \tag{A.2}$$

En développant cette équation, elle conduit à :

$$\begin{aligned}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C};\mathbf{\Lambda}) &= \|\mathcal{T}\|^2 - \Sigma_{ijk}\Sigma_p t_{ijk}^*\lambda_p a_{ip}b_{jp}c_{kp} - \Sigma_{ijk}\Sigma_q t_{ijk}\lambda_q^* a_{iq}^* b_{jq}^* c_{kq}^* \\ &\quad + \Sigma_{pq}\Sigma_{ijk}\lambda_p a_{ip}b_{jp}c_{kp}\lambda_q^* a_{iq}^* b_{jq}^* c_{kq}^* \\ &= \|\mathcal{T}\| - \Sigma_p\lambda_p f_p^* - \Sigma_q\lambda_q^* f_q + \Sigma_{pq}\lambda_p\lambda_q^* f G_{pq}\end{aligned}$$

Tel que

$$G_{pq} = \Sigma_{ijk}a_{ip}b_{jp}c_{kp}a_{iq}^*b_{jq}^*c_{kq}^* \; et \; f_q = \Sigma_{ijk}t_{ijk}a_{iq}^*b_{jq}^*c_{kq}^*$$

Maintenant, et en annulant la dérivée de Υ par rapport à λ, nous pouvons trouver le système linéaire suivant :

$$\mathbf{G}\lambda = \mathbf{f}. \tag{A.3}$$

D'où :

$$\lambda = \mathbf{G}^{-1}\mathbf{f}, \tag{A.4}$$

A.3 Preuve du théorème 2

Soit $\mathcal{T} \in \mathbb{C}^{I\times J\times K}$, $\mathfrak{A} = \{\mathbf{A} \in \mathbb{C}^{I\times R} | \mu(\mathbf{A}) \leq \mu_{\mathbf{A}}\}$, $\mathfrak{B} = \{\mathbf{B} \in \mathbb{C}^{J\times R} | \mu(\mathbf{B}) \leq \mu_{\mathbf{B}}\}$, $\mathfrak{C} = \{\mathbf{C} \in \mathbb{C}^{K\times R} | \mu(\mathbf{C}) \leq \mu_{\mathbf{C}}\}$ et considérons la fonction du coût suivante :

$$\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{\Lambda}) = \|\mathcal{T} - \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\|_F^2 \tag{A.5}$$

Considérons aussi $\aleph = \mathbb{C}^R \times \mathfrak{A} \times \mathfrak{B} \times \mathfrak{C}$ un sous ensemble non compact de $\mathbb{C}^{R(1+I+J+K)}$, et soit la borne inférieure en question : $\eta := inf\{\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{\Lambda}) \mid (\mathbf{A}, \mathbf{B}, \mathbf{C}) \in \aleph\}$.
Nous allons montrer que le sous-niveau de l'ensemble Υ restreint à $\aleph_\alpha$, définie comme $\aleph_\alpha = \{(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{\Lambda}) \in \aleph \mid \Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{\Lambda}) \leq \alpha\}$, est compact pour tout $\alpha > \eta$ et donc la borne inférieure de Υ sur $\aleph$ est atteint. L'ensemble $\aleph_\alpha = \aleph \cap \Upsilon^{-1}(-\infty, \alpha]$ est fermé puisque $\aleph$ est fermé et Υ est continue (Par la continuité de la norme).

Il reste à montrer maintenant que $\aleph_\alpha$ est borné.

Supposons le contraire, alors ils existe une séquence $((\mathbf{A}, \mathbf{B}, \mathbf{C})_k)_{k=1}^{\infty} \subset \aleph$
avec $\|(\mathbf{A}, \mathbf{B}, \mathbf{C})_k\|_2 \to \infty$ mais $\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{\Lambda})_k \leq \alpha$ pour tout k.

Il est claire que $\|(\mathbf{A}, \mathbf{B}, \mathbf{C})_k\|_2 \to \infty$ implique que $\|\lambda^k\|_2 \to \infty$ Noter que :

$$\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{\Lambda}) \geq [\|\mathcal{T}\|_F - \|\mathcal{T} - \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\|_F]^2. \tag{A.6}$$

Nous avons :

$$\begin{aligned}
\|\sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\|_F^2 &= \sum_{p,q=1}^{R} \lambda_p \widehat{\lambda}_q \langle \mathbf{a}_p, \mathbf{a}_q\rangle \langle \mathbf{b}_p, \mathbf{b}_q\rangle \langle \mathbf{c}_p, \mathbf{c}_q\rangle \\
&\geq \sum_{p=1}^{R} |\lambda_p|^2 \|\mathbf{a}\|_2^2 \|\mathbf{b}\|_2^2 \|\mathbf{c}\|_2^2 - \sum_{p\neq q} |\lambda_p \widehat{\lambda}_q \langle \mathbf{a}_p, \mathbf{a}_q\rangle \langle \mathbf{b}_p, \mathbf{b}_q\rangle \langle \mathbf{c}_p, \mathbf{c}_q\rangle| \\
&\geq \sum_{p=1}^{R} |\lambda_p|^2 \| - \mu_{\mathbf{A}} \mu_{\mathbf{B}} \mu_{\mathbf{C}} (\sum_{p\neq q} |\lambda_p \widehat{\lambda}_q|) \\
&\geq \|\lambda_p\|_2^2 - \mu_{\mathbf{A}} \mu_{\mathbf{B}} \mu_{\mathbf{C}} \|\lambda_p\|_1^2 \geq (1 - R\mu_{\mathbf{A}} \mu_{\mathbf{B}} \mu_{\mathbf{C}}) \|\lambda_p\|_2^2
\end{aligned}$$

La dernière inégalité resulte du fait que $\|\lambda_p\|_1 \leq \sqrt{R}\|\lambda\|_2$ pour tout $\lambda \in \mathbb{C}^R$. À partir de l'hypotèse $1 - R\mu_{\mathbf{A}} \mu_{\mathbf{B}} \mu_{\mathbf{C}} > 0$ et $\|\lambda_p^{(k)}\|_2 \to \infty$, $\Upsilon((\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{\Lambda})_k) \to \infty$, ce qui contredit l'hypothèse $\Upsilon((\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{\Lambda})_k) \leq \alpha$ pour tout k.

ANNEXE

B

ANNEXE 2

B.1 Calcule du pas optimal

On souhaite minimiser l'expression suivante par rapport à ρ :

$$\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) = \|\mathbf{T}_{(1)}^{I\times KJ} - (\mathbf{A}+\rho\mathbf{D}_A)[(\mathbf{C}+\rho\mathbf{D}_C)\odot(\mathbf{B}+\rho\mathbf{D}_B)]^T\|_F^2$$

$$\begin{aligned}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= \|\mathbf{T}_{(1)}^{I\times KJ} - (\mathbf{A}+\rho\mathbf{D}_A)[(\mathbf{C}\odot\mathbf{B})+(\mathbf{C}\odot\rho\mathbf{D}_B)+(\rho\mathbf{D}_C\odot\mathbf{B})+(\rho\mathbf{D}_C\odot\rho\mathbf{D}_B)]^T\|_F^2\\ &= \|\mathbf{D}_A(\mathbf{D}_C\odot\mathbf{D}_B)^T\underline{\rho^3} + (\mathbf{D}_A(\mathbf{C}\odot\mathbf{D}_B)^T + \mathbf{D}_A(\mathbf{D}_C\odot\mathbf{B})^T + \mathbf{A}(\mathbf{D}_C\odot\mathbf{D}_B)^T)\underline{\rho^2}\\ &+ \mathbf{D}_A(\mathbf{C}\odot\mathbf{B})^T + \mathbf{A}(\mathbf{C}\odot\mathbf{D}_B)^T + \mathbf{A}(\mathbf{D}_C\odot\mathbf{B})^T)\underline{\rho} + (\mathbf{T}_{(1)}^{I\times KJ} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T)\|_F^2\end{aligned}$$

Définissons des variables intermédiaires :

$$\begin{aligned}\mathbf{K}_3 &= \mathbf{D}_A(\mathbf{D}_C\odot\mathbf{D}_B)^T\\ \mathbf{K}_2 &= \mathbf{D}_A(\mathbf{D}_C\odot\mathbf{B})^T + \mathbf{A}(\mathbf{D}_C\odot\mathbf{D}_B)^T\\ \mathbf{K}_1 &= \mathbf{D}_A(\mathbf{C}\odot\mathbf{B})^T + \mathbf{A}(\mathbf{C}\odot\mathbf{D}_B)^T\\ \mathbf{K}_0 &= \mathbf{T}_{(1)}^{I\times KJ} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T\end{aligned}$$

On obtient donc :

$$\begin{aligned}\Upsilon(\mathbf{A},\mathbf{B},\mathbf{C}) &= \mathbf{trace}\{(\mathbf{K}_3\rho^3+\mathbf{K}_2\rho^2+\mathbf{K}_1\rho+\mathbf{K}_0)(\mathbf{K}_3\rho^3+\mathbf{K}_2\rho^2+\mathbf{K}_1\rho+\mathbf{K}_0)^T\}\\ &= \mathbf{trace}\{\mathbf{K}_3\mathbf{K}_3^T\rho^6\\ &+ (2\mathbf{K}_3\mathbf{K}_2^T)\rho^5\\ &+ (2\mathbf{K}_3\mathbf{K}_1+\mathbf{K}_2\mathbf{K}_2^T)\rho^4\\ &+ (2(\mathbf{K}_3\mathbf{K}_{0^T}+\mathbf{K}_2\mathbf{K}_1^T))\rho^3\\ &+ (2\mathbf{K}_2\mathbf{K}_0^T+\mathbf{K}_1\mathbf{K}_1^T)\rho^2\\ &+ (2\mathbf{K}_1\mathbf{K}_0^T)\rho\\ &+ (2\mathbf{K}_0\mathbf{K}_0^T)\}\end{aligned}$$

Les sept coefficients $a_0 ... a_6$ sont obtenus par identification.

Annexe

C

Production scientifique

Les travaux réalisés et présentés dans ce mémoire ont été valorisés dans les publications indiquées ci-dessous :

Revues internationales (2)

1. **M. Nazih**, K. Minaoui, P. Comon, "Using the proximal gradient and the accelerated proximal gradient as a canonical polyadic tensor decomposition algorithms in difficult situations", Signal Processing (**IF=4.383**), vol. 171, no. 107472, 2020.
2. **M. Nazih**, K. Minaoui, E. Sobhani, P. Comon, "The forward-backward algorithm via a logarithmic regularized barrier to compute the canonical polyadic tensor decomposition under coherence constraint", Signal Processing (**IF=4.383**), vol. 188, no. 108178, 2021.

Papier soumis

— **M. Nazih**, K. Minaoui, E. Sobhani, P. Comon, "Monotone and non-monotone accelerated proximal gradient for nonnegative canonical polyadic tensor decomposition", Soumis à IEEE Transactions on Signal Processing (**IF=5.028**).

Conférences nationales et internationales (3)

3. **M. Nazih**, K. Minaoui, "A progression strategy of proximal algorithm for the unconstrained optimization", 4th International Conference on Optimization and Applications (ICOA), p. 1-6, IEEE, 26-27 avril 2018.
4. **M. Nazih**, K. Minaoui, A. Rouijel, "A progression strategy of proximal algorithm applied to the digital images inpainting", 9th International Symposium on Signal, Image, Video and Communications (ISIVC), IEEE, 27-30 Novembre 2018.
5. **M. Nazih**, K. Minaoui, P. Comon, "Computation of the regularized canonical polyadic decomposition of tensors using the accelerated proximal gradient with momentum", The Fifth International Conference on Big Data and Internet of Things (BDIoT'21), 17-18 Mars 2021.

Bibliographie

[Absil *et al.*, 2005] Absil, P.-A., Mahony, R. et Andrews, B. (2005). Convergence of the iterates of descent methods for analytic cost functions. *SIAM Journal on Optimization*, 16(2):531–547.

[Acar *et al.*, 2005] Acar, E., Camtepe, S. A., Krishnamoorthy, M. S. et Yener, B. (2005). Modeling and multiway analysis of chatroom tensors. *In International Conference on Intelligence and Security Informatics*, pages 256–268. Springer.

[Acar et Yener, 2008] Acar, E. et Yener, B. (2008). Unsupervised multiway data analysis : A literature survey. *IEEE transactions on knowledge and data engineering*, 21(1):6–20.

[Albera *et al.*, 2012] Albera, L., Kachenoura, A., Comon, P., Karfoul, A., Wendling, F., Senhadji, L. et Merlet, I. (2012). Ica-based eeg denoising : a comparative analysis of fifteen methods. *Bulletin of the Polish Academy of Sciences : Technical Sciences*, 60(3 Special issue on Data Mining in Bioengineering):407–418.

[Andersson et Bro, 1998] Andersson, C. A. et Bro, R. (1998). Improving the speed of multi-way algorithms : : Part i. tucker3. *Chemometrics and intelligent laboratory systems*, 42(1-2):93–103.

[Appellof et Davidson, 1981] Appellof, C. J. et Davidson, E. R. (1981). Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents. *Analytical Chemistry*, 53(13):2053–2056.

[Attouch et Bolte, 2009] Attouch, H. et Bolte, J. (2009). On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. *Mathematical Programming*, 116(1):5–16.

[Attouch *et al.*, 2013] Attouch, H., Bolte, J. et Svaiter, B. F. (2013). Convergence of descent methods for semi-algebraic and tame problems : proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, 137(1):91–129.

[Bach *et al.*, 2011] Bach, F., Jenatton, R., Mairal, J. et Obozinski, G. (2011). Optimization with sparsity-inducing penalties. *arXiv preprint arXiv :1108.0775*.

[Bader *et al.*, 2008] BADER, B. W., BERRY, M. W. et BROWNE, M. (2008). Discussion tracking in enron email using parafac. *In Survey of Text Mining II*, pages 147–163. Springer.

[Bader et Kolda, 2006] BADER, B. W. et KOLDA, T. G. (2006). Algorithm 862 : Matlab tensor classes for fast algorithm prototyping. *ACM Transactions on Mathematical Software (TOMS)*, 32(4):635–653.

[Ballester-Ripoll *et al.*, 2015] BALLESTER-RIPOLL, R., SUTER, S. K. et PAJAROLA, R. (2015). Analysis of tensor approximation for compression-domain volume visualization. *Computers & Graphics*, 47:34–47.

[Bauschke *et al.*, 2011a] BAUSCHKE, H. H., BURACHIK, R. S., COMBETTES, P. L., ELSER, V., LUKE, D. R. et WOLKOWICZ, H. (2011a). *Fixed-point algorithms for inverse problems in science and engineering*, volume 49. Springer Science & Business Media.

[Bauschke *et al.*, 2011b] BAUSCHKE, H. H., COMBETTES, P. L. *et al.* (2011b). *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer.

[Beck et Teboulle, 2009a] BECK, A. et TEBOULLE, M. (2009a). Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE transactions on image processing*, 18(11):2419–2434.

[Beck et Teboulle, 2009b] BECK, A. et TEBOULLE, M. (2009b). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202.

[Becker *et al.*, 2014] BECKER, H., ALBERA, L., COMON, P., HAARDT, M., BIROT, G., WENDLING, F., GAVARET, M., BÉNAR, C. G. et MERLET, I. (2014). Eeg extended source localization : tensor-based vs. conventional methods. *NeuroImage*, 96:143–157.

[Becker et Fadili, 2012] BECKER, S. et FADILI, M. J. (2012). A quasi-newton proximal splitting method. *arXiv preprint arXiv :1206.1156*.

[Beckmann et Smith, 2005] BECKMANN, C. F. et SMITH, S. M. (2005). Tensorial extensions of independent component analysis for multisubject fmri analysis. *Neuroimage*, 25(1):294–311.

[Bect *et al.*, 2004] BECT, J., BLANC-FÉRAUD, L., AUBERT, G. et CHAMBOLLE, A. (2004). A l 1-unified variational framework for image restoration. *In European Conference on Computer Vision*, pages 1–13. Springer.

[Berry *et al.*, 2007] BERRY, M. W., BROWNE, M., LANGVILLE, A. N., PAUCA, V. P. et PLEMMONS, R. J. (2007). Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173.

[Bertsekas *et al.*, 1999] BERTSEKAS, D. P., HAGER, W. et MANGASARIAN, O. (1999). Nonlinear programming. athena scientific belmont. *Massachusets, USA*.

[Bertsekas et Scientific,] BERTSEKAS, D. P. et SCIENTIFIC, A. Nonlinear programming 3rd edition theoretical solutions manual.

[Blekherman et Teitler, 2015] BLEKHERMAN, G. et TEITLER, Z. (2015). On maximum, typical and generic ranks. *Mathematische Annalen*, 362(3):1021–1031.

[Bolte *et al.*, 2010] BOLTE, J., DANIILIDIS, A., LEY, O. et MAZET, L. (2010). Characterizations of łojasiewicz inequalities : subgradient flows, talweg, convexity. *Transactions of the American Mathematical Society*, 362(6):3319–3363.

[Boţ *et al.*, 2016] BOŢ, R. I., CSETNEK, E. R. et LÁSZLÓ, S. C. (2016). An inertial forward–backward algorithm for the minimization of the sum of two nonconvex functions. *EURO Journal on Computational Optimization*, 4(1):3–25.

[Boyd *et al.*, 2004] BOYD, S., BOYD, S. P. et VANDENBERGHE, L. (2004). *Convex optimization*. Cambridge university press.

[Boyd *et al.*, 2011] BOYD, S., PARIKH, N. et CHU, E. (2011). *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc.

[Brewer, 1978] BREWER, J. (1978). Kronecker products and matrix calculus in system theory. *IEEE Transactions on circuits and systems*, 25(9):772–781.

[Bro, 1998] BRO, R. (1998). Multi-way analysis in the food industry-models, algorithms, and applications. *In MRI, EPG and EMA, Proc ICSLP 2000*. Citeseer.

[Bro *et al.*, 1997] BRO, R. *et al.* (1997). Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–172.

[Bro et Andersson, 1998] BRO, R. et ANDERSSON, C. A. (1998). Improving the speed of multiway algorithms : Part ii : Compression. *Chemometrics and intelligent laboratory systems*, 42(1-2):105–113.

[Bu, 2018] BU, F. (2018). An efficient fuzzy c-means approach based on canonical polyadic decomposition for clustering big data in iot. *Future Generation Computer Systems*, 88:675–682.

[Burt et Goulart, 2013] BURT, P. M. et GOULART, J. H. M. (2013). Evaluating the potential of volterra-parafac iir models. *In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5745–5749. IEEE.

[Candes et Romberg, 2007] CANDES, E. et ROMBERG, J. (2007). Sparsity and incoherence in compressive sampling. *Inverse problems*, 23(3):969.

[Candès et Tao, 2010] CANDÈS, E. J. et TAO, T. (2010). The power of convex relaxation : Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5): 2053–2080.

[Candes *et al.*, 2008] CANDES, E. J., WAKIN, M. B. et BOYD, S. P. (2008). Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905.

[Cardoso, 1990a] CARDOSO, J.-F. (1990a). Eigen-structure of the fourth-order cumulant tensor with application to the blind source separation problem. *In International Conference on Acoustics, Speech, and Signal Processing*, pages 2655–2658. IEEE.

[Cardoso, 1990b] CARDOSO, J.-F. (1990b). Tensor-based independent component analysis. *Signal Processing V, Theories and Applications*, pages 673–676.

[Cardoso, 1991] CARDOSO, J.-F. (1991). Super-symmetric decomposition of the fourth-order cumulant tensor. blind identification of more sources than sensors. *In Proc. ICASSP*, volume 91, pages 3109–3112. Citeseer.

[Carroll et Chang, 1970] CARROLL, J. D. et CHANG, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of 'eckart-young' decomposition. *Psychometrika*, 35(3):283–319.

[Castella *et al.*, 2006a] CASTELLA, M., BIANCHI, P., CHEVREUIL, A. et PESQUET, J.-C. (2006a). A blind source separation framework for detecting cpm sources mixed by a convolutive mimo filter. *Signal Processing*, 86(8):1950–1967.

[Castella et Moreau, 2009] CASTELLA, M. et MOREAU, E. (2009). Generalized identifiability conditions for blind convolutive mimo separation. *IEEE Transactions on signal processing*, 57(7):2846–2852.

[Castella *et al.*, 2006b] CASTELLA, M., RHIOUI, S., MOREAU, E. et PESQUET, J.-C. (2006b). Quadratic higher order criteria for iterative blind separation of a mimo convolutive mixture of sources. *IEEE Transactions on Signal Processing*, 55(1):218–232.

[Cattell, 1944] CATTELL, R. B. (1944). 'parallel proportional profiles' and other principles for determining the choice of factors by rotation. *Psychometrika*, 9(4):267–283.

[Chabriel et Barrère, 2005] CHABRIEL, G. et BARRÈRE, J. (2005). An instantaneous formulation of mixtures for blind separation of propagating waves. *IEEE transactions on signal processing*, 54(1):49–58.

[Chaux *et al.*, 2007] CHAUX, C., COMBETTES, P. L., PESQUET, J.-C. et WAJS, V. R. (2007). A variational formulation for frame-based inverse problems. *Inverse Problems*, 23(4):1495.

[Chen et Rockafellar, 1997] CHEN, G. H. et ROCKAFELLAR, R. T. (1997). Convergence rates in forward–backward splitting. *SIAM Journal on Optimization*, 7(2):421–444.

[Cheng *et al.*, 2014] CHENG, E., PANG, Y., ZHU, Y., YU, J. et LING, H. (2014). Curvilinear structure tracking by low rank tensor approximation with model propagation. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3057–3064.

[Cheng *et al.*, 2016] CHENG, H., YU, Y., ZHANG, X., XING, E. et SCHUURMANS, D. (2016). Scalable and sound low-rank tensor learning. *In Artificial Intelligence and Statistics*, pages 1114–1123. PMLR.

[Chevalier *et al.*, 2004] CHEVALIER, P., ALBERA, L., COMON, P. et FERRÉOL, A. (2004). Comparative performance analysis of eight blind source separation methods on radiocommunications signals. *In 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, volume 1, pages 273–278. IEEE.

[Chevalier *et al.*, 2005] CHEVALIER, P., ALBERA, L., FERRÉOL, A. et COMON, P. (2005). On the virtual array concept for higher order array processing. *IEEE Transactions on Signal Processing*, 53(4):1254–1271.

[Chiantini *et al.*, 2017] CHIANTINI, L., OTTAVIANI, G. et VANNIEUWENHOVEN, N. (2017). On generic identifiability of symmetric tensors of subgeneric rank. *Transactions of the American Mathematical Society*, 369(6):4021–4042.

[Chouzenoux *et al.*, 2014] CHOUZENOUX, E., PESQUET, J.-C. et REPETTI, A. (2014). Variable metric forward–backward algorithm for minimizing the sum of a differentiable function and a convex function. *Journal of Optimization Theory and Applications*, 162(1):107–132.

[Chu *et al.*, 2004] CHU, M., DIELE, F., PLEMMONS, R. et RAGNI, S. (2004). Optimality, computation, and interpretation of nonnegative matrix factorizations. *In SIAM Journal on Matrix Analysis*. Citeseer.

[Cichocki *et al.*, 2015] CICHOCKI, A., MANDIC, D., DE LATHAUWER, L., ZHOU, G., ZHAO, Q., CAIAFA, C. et PHAN, H. A. (2015). Tensor decompositions for signal processing applications : From two-way to multiway component analysis. *IEEE signal processing magazine*, 32(2):145–163.

[Cichocki et Phan, 2009] CICHOCKI, A. et PHAN, A.-H. (2009). Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 92(3):708–721.

[Cichocki *et al.*, 2009] CICHOCKI, A., ZDUNEK, R., PHAN, A. H. et AMARI, S.-i. (2009). *Nonnegative matrix and tensor factorizations : applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons.

[Combettes et Pesquet, 2008] COMBETTES, P. L. et PESQUET, J.-C. (2008). Proximal thresholding algorithm for minimization over orthonormal bases. *SIAM Journal on Optimization*, 18(4):1351–1376.

[Combettes et Pesquet, 2011] COMBETTES, P. L. et PESQUET, J.-C. (2011). Proximal splitting methods in signal processing. *In Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer.

[Combettes et Vũ, 2013] COMBETTES, P. L. et VŨ, B. C. (2013). Variable metric quasi-fejér monotonicity. *Nonlinear Analysis : Theory, Methods & Applications*, 78:17–31.

[Combettes et Vũ, 2014] COMBETTES, P. L. et VŨ, B. C. (2014). Variable metric forward–backward splitting with applications to monotone inclusions in duality. *Optimization*, 63(9):1289–1318.

[Combettes et Wajs, 2005] COMBETTES, P. L. et WAJS, V. R. (2005). Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200.

[Comon, 1986] COMON, P. (1986). Estimation multivariable complexe. *Traitement du Signal*, 3(2):97–101.

[Comon, 1994] COMON, P. (1994). Independent component analysis, a new concept ? *Signal processing*, 36(3):287–314.

[Comon, 2002] COMON, P. (2002). Tensor decompositions. *Mathematics in Signal Processing V*, pages 1–24.

[Comon *et al.*, 2008] COMON, P., GOLUB, G., LIM, L.-H. et MOURRAIN, B. (2008). Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1254–1279.

[Comon et Jutten, 2010] COMON, P. et JUTTEN, C. (2010). *Handbook of Blind Source Separation : Independent component analysis and applications*. Academic press.

[Comon et Lim, 2011] COMON, P. et LIM, L.-H. (2011). Sparse representations and low-rank tensor approximation.

[Comon *et al.*, 2009a] COMON, P., LUCIANI, X. et DE ALMEIDA, A. L. (2009a). Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics : A Journal of the Chemometrics Society*, 23(7-8):393–405.

[Comon *et al.*, 2013] COMON, P., MINAOUI, K., ROUIJEL, A. et ABOUTAJDINE, D. (2013). Performance index for tensor polyadic decompositions. *In 21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5. IEEE.

[Comon et Rajih, 2006] COMON, P. et RAJIH, M. (2006). Blind identification of under-determined mixtures based on the characteristic function. *Signal Processing*, 86(9): 2271–2281.

[Comon et ten Berge, 2008] COMON, P. et ten BERGE, J. M. (2008). Generic and typical ranks of three-way arrays. *In 2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3313–3316. IEEE.

[Comon *et al.*, 2009b] COMON, P., TEN BERGE, J. M., DE LATHAUWER, L. et CASTAING, J. (2009b). Generic and typical ranks of multi-way arrays. *Linear Algebra and its Applications*, 430(11-12):2997–3007.

[Congedo *et al.*, 2008] CONGEDO, M., GOUY-PAILLER, C. et JUTTEN, C. (2008). On the blind source separation of human electroencephalogram by approximate joint diagonalization of second order statistics. *Clinical Neurophysiology*, 119(12):2677–2686.

[Dai *et al.*, 2019] DAI, X., LI, C., HE, X. et LI, C. (2019). Nonnegative matrix factorization algorithms based on the inertial projection neural network. *Neural Computing and Applications*, 31(8):4215–4229.

[Daubechies *et al.*, 2004] DAUBECHIES, I., DEFRISE, M. et DE MOL, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics : A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457.

[Dauwels *et al.*, 2011] DAUWELS, J., SRINIVASAN, K., RAMASUBBA, R. M. et CICHOCKI, A. (2011). Multi-channel eeg compression based on matrix and tensor decompositions.

In 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 629–632. IEEE.

[de Almeida *et al.*, 2007] de ALMEIDA, A. L., FAVIER, G. et MOTA, J. C. M. (2007). Parafac-based unified tensor modeling for wireless communication systems with application to blind multiuser equalization. *Signal Processing*, 87(2):337–351.

[De Lathauwer, 1997] DE LATHAUWER, L. (1997). *Signal processing based on multilinear algebra*. Katholieke Universiteit Leuven Leuven.

[De Lathauwer, 2006] DE LATHAUWER, L. (2006). A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. *SIAM journal on Matrix Analysis and Applications*, 28(3):642–666.

[De Lathauwer *et al.*, 1995a] DE LATHAUWER, L., COMON, P., DE MOOR, B. et VANDEWALLE, J. (1995a). Higher-order power method. *Nonlinear Theory and its Applications, NOLTA95*, 1:4.

[De Lathauwer *et al.*, 1995b] DE LATHAUWER, L., DE MOOR, B. et VANDEWALLE, J. (1995b). Fetal electrocardiogram extraction by source subspace separation. *In IEEE SP/Athos Workshop on Higher-Order Statistics*, page 134.

[De Lathauwer *et al.*, 2000a] DE LATHAUWER, L., DE MOOR, B. et VANDEWALLE, J. (2000a). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278.

[De Lathauwer *et al.*, 2000b] DE LATHAUWER, L., DE MOOR, B. et VANDEWALLE, J. (2000b). On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM journal on Matrix Analysis and Applications*, 21(4):1324–1342.

[De Silva et Lim, 2008] DE SILVA, V. et LIM, L.-H. (2008). Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127.

[Delabrouille *et al.*, 2003] DELABROUILLE, J., CARDOSO, J.-F. et PATANCHON, G. (2003). Multidetector multicomponent spectral matching and applications for cosmic microwave background data analysis. *Monthly Notices of the Royal Astronomical Society*, 346(4):1089–1102.

[Domanov et De Lathauwer, 2013a] DOMANOV, I. et DE LATHAUWER, L. (2013a). On the uniqueness of the canonical polyadic decomposition of third-order tensors—part i : Basic results and uniqueness of one factor matrix. *SIAM Journal on Matrix Analysis and Applications*, 34(3):855–875.

[Domanov et De Lathauwer, 2013b] DOMANOV, I. et DE LATHAUWER, L. (2013b). On the uniqueness of the canonical polyadic decomposition of third-order tensors—part ii : Uniqueness of the overall decomposition. *SIAM Journal on Matrix Analysis and Applications*, 34(3):876–903.

[Donoho et Elad, 2003] DONOHO, D. L. et ELAD, M. (2003). Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202.

[Eckart et Young, 1936] ECKART, C. et YOUNG, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

[Fan et Li, 2001] FAN, J. et LI, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360.

[Farias *et al.*, 2018] FARIAS, R. C., de MORAIS GOULART, J. H. et COMON, P. (2018). Coherence constrained alternating least squares. *In 2018 26th European Signal Processing Conference (EUSIPCO)*, pages 613–617. IEEE.

[Favier et de Almeida, 2014] FAVIER, G. et de ALMEIDA, A. L. (2014). Tensor space-time-frequency coding with semi-blind receivers for mimo wireless communication systems. *IEEE Transactions on Signal Processing*, 62(22):5987–6002.

[Favier et Kibangou, 2009] FAVIER, G. et KIBANGOU, A. Y. (2009). Tensor-based methods for system identification. part 2 : Three examples of tensor-based system identification methods. *International Journal on Sciences and Techniques of Automatic Control (IJ-STA)*, 3(1):870–889.

[Favier *et al.*, 2012] FAVIER, G., KIBANGOU, A. Y. et BOUILLOC, T. (2012). Nonlinear system modeling and identification using volterra-parafac models. *International Journal of Adaptive Control and Signal Processing*, 26(1):30–53.

[Fernandes *et al.*, 2011] FERNANDES, C. A., FAVIER, G. et MOTA, J. C. (2011). Parafac-based channel estimation and data recovery in nonlinear mimo spread spectrum communication systems. *Signal Processing*, 91(2):311–322.

[Fernandes *et al.*, 2008] FERNANDES, C. E. R., FAVIER, G. et MOTA, J. C. M. (2008). Blind channel identification algorithms based on the parafac decomposition of cumulant tensors : the single and multiuser cases. *Signal Processing*, 88(6):1382–1401.

[Ferréol *et al.*, 2004] FERRÉOL, A., CHEVALIER, P. et ALBERA, L. (2004). Second-order blind separation of first-and second-order cyclostationary sources-application to am, fsk, cpfsk, and deterministic sources. *IEEE Transactions on Signal Processing*, 52(4):845–861.

[Fiacco et McCormick, 1990] FIACCO, A. V. et MCCORMICK, G. P. (1990). *Nonlinear programming : sequential unconstrained minimization techniques*. SIAM.

[Figueiredo et Nowak, 2003] FIGUEIREDO, M. A. et NOWAK, R. D. (2003). An em algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916.

[Foucart et Lai, 2009] FOUCART, S. et LAI, M.-J. (2009). Sparsest solutions of underdetermined linear systems via lq-minimization. *Applied and Computational Harmonic Analysis*, 26(3):395–407.

[Frankel *et al.*, 2015] FRANKEL, P., GARRIGOS, G. et PEYPOUQUET, J. (2015). Splitting methods with variable metric for kurdyka–łojasiewicz functions and general convergence rates. *Journal of Optimization Theory and Applications*, 165(3):874–900.

[Friedland et Lim, 2018] FRIEDLAND, S. et LIM, L.-H. (2018). Nuclear norm of higher-order tensors. *Mathematics of Computation*, 87(311):1255–1281.

[Fu *et al.*, 2020] FU, X., VERVLIET, N., DE LATHAUWER, L., HUANG, K. et GILLIS, N. (2020). Computing large-scale matrix and tensor decomposition with structured factors : A unified nonconvex optimization perspective. *IEEE Signal Processing Magazine*, 37(5):78–94.

[Gao *et al.*, 2019] GAO, J., LI, P. et CHEN, Z. (2019). A canonical polyadic deep convolutional computation model for big data feature learning in internet of things. *Future Generation Computer Systems*, 99:508–516.

[Geman et Yang, 1995] GEMAN, D. et YANG, C. (1995). Nonlinear image recovery with half-quadratic regularization. *IEEE transactions on Image Processing*, 4(7):932–946.

[Ghadimi et Lan, 2016] GHADIMI, S. et LAN, G. (2016). Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99.

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

[Gong *et al.*, 2013] GONG, P., ZHANG, C., LU, Z., HUANG, J. et YE, J. (2013). A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. *In international conference on machine learning*, pages 37–45. PMLR.

[Grave *et al.*, 2011] GRAVE, E., OBOZINSKI, G. et BACH, F. (2011). Trace lasso : a trace norm regularization for correlated designs. *arXiv preprint arXiv :1109.1990*.

[Gribonval et Nielsen, 2003] GRIBONVAL, R. et NIELSEN, M. (2003). Sparse representations in unions of bases. *IEEE transactions on Information theory*, 49(12):3320–3325.

[Gu *et al.*, 2018] GU, B., WANG, D., HUO, Z. et HUANG, H. (2018). Inexact proximal gradient methods for non-convex and non-smooth optimization. *In Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[Guan *et al.*, 2012] GUAN, N., TAO, D., LUO, Z. et YUAN, B. (2012). Nenmf : An optimal gradient method for nonnegative matrix factorization. *IEEE Transactions on Signal Processing*, 60(6):2882–2898.

[Hackbusch, 2012] HACKBUSCH, W. (2012). *Tensor spaces and numerical tensor calculus*, volume 42. Springer.

[Hackbusch *et al.*, 2017] HACKBUSCH, W., KRESSNER, D. et USCHMAJEW, A. (2017). Perturbation of higher-order singular values. *SIAM Journal on applied algebra and geometry*, 1(1):374–387.

[Harshman, 1972] HARSHMAN, R. A. (1972). Determination and proof of minimum uniqueness conditions for parafac1. *UCLA working papers in phonetics*, 22(111-117):3.

[Harshman, 2004] HARSHMAN, R. A. (2004). The problem and nature of degenerate solutions or decompositions of 3-way arrays. *In Talk at the Tensor Decompositions Workshop, Palo Alto, CA, American Institute of Mathematics*.

[Harshman *et al.*, 1970] HARSHMAN, R. A. *et al.* (1970). Foundations of the parafac procedure : Models and conditions for an" explanatory" multimodal factor analysis.

[He *et al.*, 2006] HE, T., CLIFFORD, G. et TARASSENKO, L. (2006). Application of independent component analysis in removing artefacts from the electrocardiogram. *Neural Computing and Applications*, 15(2):105–116.

[Henrion, 1994] HENRION, R. (1994). N-way principal component analysis theory, algorithms and applications. *Chemometrics and intelligent laboratory systems*, 25(1):1–23.

[Hérault *et al.*, 1985] HÉRAULT, J., JUTTEN, C. et ANS, B. (1985). Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé. *In 10 Colloque sur le traitement du signal et des images, FRA, 1985*. GRETSI, Groupe d'Etudes du Traitement du Signal et des Images.

[Hiriart-Urruty et Lemaréchal, 1993] HIRIART-URRUTY, J.-B. et LEMARÉCHAL, C. (1993). Convex analysis and minimization algorithms i, volume 305 of comprehensive study in mathematics.

[Hitchcock, 1927] HITCHCOCK, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189.

[Hitchcock, 1928] HITCHCOCK, F. L. (1928). Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7(1-4):39–79.

[Hjorungnes et Gesbert, 2007] HJORUNGNES, A. et GESBERT, D. (2007). Complex-valued matrix differentiation : Techniques and key results. *IEEE Transactions on Signal Processing*, 55(6):2740–2746.

[Horn, 1990] HORN, R. A. (1990). The hadamard product. *In Proc. Symp. Appl. Math*, volume 40, pages 87–169.

[Hoyer, 2002] HOYER, P. O. (2002). Non-negative sparse coding. *In Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565. IEEE.

[Huang *et al.*, 2016] HUANG, K., SIDIROPOULOS, N. D. et LIAVAS, A. P. (2016). A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *IEEE Transactions on Signal Processing*, 64(19):5052–5065.

[Jacob *et al.*, 2009] JACOB, L., OBOZINSKI, G. et VERT, J.-P. (2009). Group lasso with overlap and graph lasso. *In Proceedings of the 26th annual international conference on machine learning*, pages 433–440.

[Jenatton *et al.*, 2011] JENATTON, R., MAIRAL, J., OBOZINSKI, G. et BACH, F. (2011). Proximal methods for hierarchical sparse coding. *The Journal of Machine Learning Research*, 12:2297–2334.

[Kaplansky, 1990] KAPLANSKY, I. (1990). Algebraic polar decomposition. *SIAM Journal on Matrix Analysis and Applications*, 11(2):213–217.

[Kapteyn *et al.*, 1986] KAPTEYN, A., NEUDECKER, H. et WANSBEEK, T. (1986). An approach ton-mode components analysis. *Psychometrika*, 51(2):269–275.

[Khatri et Rao, 1968] KHATRI, C. et RAO, C. R. (1968). Solutions to some functional equations and their applications to characterization of probability distributions. *Sankhyā : The Indian Journal of Statistics, Series A*, pages 167–180.

[Kibangou et Favier, 2006] KIBANGOU, A. Y. et FAVIER, G. (2006). Wiener-hammerstein systems modeling using diagonal volterra kernels coefficients. *IEEE signal processing letters*, 13(6):381–384.

[Kiers, 2000] KIERS, H. A. (2000). Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics : A Journal of the Chemometrics Society*, 14(3):105–122.

[Kim et Park, 2007] KIM, H. et PARK, H. (2007). Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502.

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

[Kim *et al.*, 2014] KIM, J., HE, Y. et PARK, H. (2014). Algorithms for nonnegative matrix and tensor factorizations : A unified view based on block coordinate descent framework. *Journal of Global Optimization*, 58(2):285–319.

[Kim et Park, 2011] KIM, J. et PARK, H. (2011). Fast nonnegative matrix factorization : An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281.

[Kolda, 2001] KOLDA, T. G. (2001). Orthogonal tensor decompositions. *SIAM Journal on Matrix Analysis and Applications*, 23(1):243–255.

[Kolda et Bader, 2009] KOLDA, T. G. et BADER, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

[Kolda et Sun, 2008] KOLDA, T. G. et SUN, J. (2008). Scalable tensor decompositions for multi-aspect data mining. *In 2008 Eighth IEEE international conference on data mining*, pages 363–372. IEEE.

[Koldovsky et Tichavskỳ, 2010] KOLDOVSKY, Z. et TICHAVSKỲ, P. (2010). Time-domain blind separation of audio sources on the basis of a complete ica decomposition of an observation space. *IEEE transactions on audio, speech, and language processing*, 19(2): 406–416.

[Kroonenberg, 1983] KROONENBERG, P. M. (1983). *Three-mode principal component analysis : Theory and applications*, volume 2. DSWO press.

[Kroonenberg et De Leeuw, 1980] KROONENBERG, P. M. et DE LEEUW, J. (1980). Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97.

[Kruskal *et al.*, 1989] KRUSKAL, J., HARSHMAN, R. et LUNDY, M. (1989). How 3-mfa data can cause degenerate parafac solutions, among other relationships. *In Multiway data analysis*, pages 115–122.

[Kruskal, 1977] KRUSKAL, J. B. (1977). 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.

[Kruskal, 1989] KRUSKAL, J. B. (1989). Rank, decomposition, and uniqueness for 3-way and n-way arrays. *Multiway data analysis*, pages 7–18.

[Kumar *et al.*, 2013] KUMAR, A., SINDHWANI, V. et KAMBADUR, P. (2013). Fast conical hull algorithms for near-separable non-negative matrix factorization. *In International Conference on Machine Learning*, pages 231–239. PMLR.

[Lee, 1999] LEE, D. (1999). Seung, h. s.(1999). learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

[Lee et Seung, 2001] LEE, D. et SEUNG, H. (2001). Algorithms for non-negative matrix factorization advances in neural information processing 13 (proc. nips 2000).

[Lee et Seung, 1999] LEE, D. D. et SEUNG, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

[Lee et Waziruddin, 1970] LEE, E. et WAZIRUDDIN, S. (1970). Applying gradient projection and conjugate gradient to the optimum operation of reservoirs 1. *JAWRA Journal of the American Water Resources Association*, 6(5):713–724.

[Lee *et al.*, 2001] LEE, J. S., LEE, D. D., CHOI, S., PARK, K. S. et LEE, D. S. (2001). Non-negative matrix factorization of dynamic images in nuclear medicine. *In 2001 IEEE Nuclear Science Symposium Conference Record (Cat. No. 01CH37310)*, volume 4, pages 2027–2030. IEEE.

[Leurgans et Ross, 1992] LEURGANS, S. et ROSS, R. T. (1992). Multilinear models : applications in spectroscopy. *Statistical Science*, pages 289–310.

[Leurgans *et al.*, 1993] LEURGANS, S. E., ROSS, R. T. et ABEL, R. B. (1993). A decomposition for three-way arrays. *SIAM Journal on Matrix Analysis and Applications*, 14(4):1064–1083.

[Li et Lin, 2015] LI, H. et LIN, Z. (2015). Accelerated proximal gradient methods for nonconvex programming. *Advances in neural information processing systems*, 28:379–387.

[Li *et al.*, 2017] LI, Q., ZHOU, Y., LIANG, Y. et VARSHNEY, P. K. (2017). Convergence analysis of proximal gradient with momentum for nonconvex optimization. *In International Conference on Machine Learning*, pages 2111–2119. PMLR.

[Lim et Comon, 2010] LIM, L.-H. et COMON, P. (2010). Multiarray signal processing : Tensor decomposition meets compressed sensing. *Comptes Rendus Mecanique*, 338(6): 311–320.

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

[Liu *et al.*, 2012] LIU, J., MUSIALSKI, P., WONKA, P. et YE, J. (2012). Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):208–220.

[Liu et Ye, 2010] LIU, J. et YE, J. (2010). Moreau-yosida regularization for grouped tree structure learning. *Advances in neural information processing systems*, 23:1459–1467.

[Liu et Yi, 2003] LIU, W. et YI, J. (2003). Existing and new algorithms for nonnegative matrix factorization. *University of Texas at Austin*.

[Lojasiewicz, 1963] LOJASIEWICZ, S. (1963). Une propriété topologique des sous-ensembles analytiques réels. *Les équations aux dérivées partielles*, 117:87–89.

[Luciani, 2007] LUCIANI, X. (2007). *Analyse numerique des spectres de fluorescence 3D issus de melanges non lineaires*. Thèse de doctorat, Université du Sud Toulon Var.

[Luciani et Albera, 2014] LUCIANI, X. et ALBERA, L. (2014). Canonical polyadic decomposition based on joint eigenvalue decomposition. *Chemometrics and Intelligent Laboratory Systems*, 132:152–167.

[Luenberger *et al.*, 1984] LUENBERGER, D. G., YE, Y. *et al.* (1984). *Linear and nonlinear programming*, volume 2. Springer.

[Mahyari *et al.*, 2016] MAHYARI, A. G., ZOLTOWSKI, D. M., BERNAT, E. M. et AVIYENTE, S. (2016). A tensor decomposition-based approach for detecting dynamic network states from eeg. *IEEE Transactions on Biomedical Engineering*, 64(1):225–237.

[Marmarelis *et al.*, 2013] MARMARELIS, V. Z., SHIN, D. C., SONG, D., HAMPSON, R. E., DEADWYLER, S. A. et BERGER, T. W. (2013). Nonlinear modeling of dynamic interactions within neuronal ensembles using principal dynamic modes. *Journal of computational neuroscience*, 34(1):73–87.

[McCullagh, 2018] MCCULLAGH, P. (2018). *Tensor methods in statistics*. Courier Dover Publications.

[Mocks, 1988] MOCKS, J. (1988). Topographic components model for event-related potentials and some biophysical considerations. *IEEE transactions on biomedical engineering*, 35(6):482–484.

[Mohan et Fazel, 2012] MOHAN, K. et FAZEL, M. (2012). Iterative reweighted algorithms for matrix rank minimization. *The Journal of Machine Learning Research*, 13(1):3441–3473.

[Moreau, 1965] MOREAU, J.-J. (1965). Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299.

[Mørup, 2011] MØRUP, M. (2011). Applications of tensor (multiway array) factorizations and decompositions in data mining. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 1(1):24–40.

[Moussaoui *et al.*, 2008] MOUSSAOUI, S., HAUKSDOTTIR, H., SCHMIDT, F., JUTTEN, C., CHANUSSOT, J., BRIE, D., DOUTÉ, S. et BENEDIKTSSON, J. A. (2008). On the decomposition of mars hyperspectral data by ica and bayesian positive source separation. *Neurocomputing*, 71(10-12):2194–2208.

[Muti, 2004] MUTI, D. (2004). *Traitement du Signal Tensoriel, application aux images en couleurs et aux signaux sismiques bruitÃ©s*. Thèse de doctorat, Aix-Marseille 3.

[Muti et Bourennane, 2005] MUTI, D. et BOURENNANE, S. (2005). Multidimensional filtering based on a tensor approach. *Signal Processing*, 85(12):2338–2353.

[Nazih *et al.*, 2020] NAZIH, M., MINAOUI, K. et COMON, P. (2020). Using the proximal gradient and the accelerated proximal gradient as a canonical polyadic tensor decomposition algorithms in difficult situations. *Signal Processing*, page 107472.

[Nesterov, 1983] NESTEROV, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). *In Doklady an ussr*, volume 269, pages 543–547.

[Nesterov, 2005] NESTEROV, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152.

[Nesterov, 2013] NESTEROV, Y. (2013). Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161.

[Nesterov *et al.*, 2020] NESTEROV, Y., GASNIKOV, A., GUMINOV, S. et DVURECHENSKY, P. (2020). Primal–dual accelerated gradient methods with small-dimensional relaxation oracle. *Optimization Methods and Software*, pages 1–38.

[Nesterov et Nemirovskii, 1994] NESTEROV, Y. et NEMIROVSKII, A. (1994). *Interior-point polynomial algorithms in convex programming*. SIAM.

[Nion, 2007] NION, D. (2007). *MÃ©thodes PARAFAC gÃ©nÃ©ralisÃ©es pour l'extraction aveugle de sources : Application aux systÃ¨mes DS-CDMA*. Thèse de doctorat, Cergy-Pontoise.

[Nion et De Lathauwer, 2008] NION, D. et DE LATHAUWER, L. (2008). An enhanced line search scheme for complex-valued tensor decompositions. application in ds-cdma. *Signal Processing*, 88(3):749–755.

[Ochs *et al.*, 2014] OCHS, P., CHEN, Y., BROX, T. et POCK, T. (2014). ipiano : Inertial proximal algorithm for nonconvex optimization. *SIAM Journal on Imaging Sciences*, 7(2):1388–1419.

[Ochs *et al.*, 2015] OCHS, P., DOSOVITSKIY, A., BROX, T. et POCK, T. (2015). On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision. *SIAM Journal on Imaging Sciences*, 8(1):331–372.

[Oeding et Ottaviani, 2013] OEDING, L. et OTTAVIANI, G. (2013). Eigenvectors of tensors and algorithms for waring decomposition. *Journal of Symbolic Computation*, 54:9–35.

[Ozerov *et al.*, 2011] OZEROV, A., FÉVOTTE, C., BLOUET, R. et DURRIEU, J.-L. (2011). Multichannel nonnegative tensor factorization with structured constraints for user-guided audio source separation. *In 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 257–260. IEEE.

[Paatero, 2000] PAATERO, P. (2000). Construction and analysis of degenerate parafac models. *Journal of Chemometrics : A Journal of the Chemometrics Society*, 14(3):285–299.

[Paatero et Tapper, 1994] PAATERO, P. et TAPPER, U. (1994). Positive matrix factorization : A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126.

[Papalexakis *et al.*, 2016] PAPALEXAKIS, E. E., MITCHELL, T. M., SIDIROPOULOS, N. D., FALOUTSOS, C., TALUKDAR, P. P. et MURPHY, B. (2016). Turbo-smt : Parallel coupled sparse matrix-tensor factorizations and applications. *Statistical Analysis and Data Mining : The ASA Data Science Journal*, 9(4):269–290.

[Parikh et Boyd, 2014] PARIKH, N. et BOYD, S. (2014). Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239.

[Pauca *et al.*, 2004] PAUCA, V. P., SHAHNAZ, F., BERRY, M. W. et PLEMMONS, R. J. (2004). Text mining using non-negative matrix factorizations. *In Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 452–456. SIAM.

[Paulus et Mars, 2006] PAULUS, C. et MARS, J. I. (2006). New multicomponent filters for geophysical data processing. *IEEE transactions on geoscience and remote sensing*, 44(8):2260–2270.

[Piper *et al.*, 2004] PIPER, J., PAUCA, V. P., PLEMMONS, R. J. et GIFFIN, M. (2004). Object characterization from spectral data using nonnegative factorization and information theory. *In Proceedings of AMOS technical conference*.

[Qi *et al.*, 2016] QI, Y., COMON, P. et LIM, L.-H. (2016). Uniqueness of nonnegative tensor approximations. *IEEE Transactions on Information Theory*, 62(4):2170–2183.

[Rajih et Comon, 2005] RAJIH, M. et COMON, P. (2005). Enhanced line search applied to blind channel identification identifiability conditions. *In IEEE/SP 13th Workshop on Statistical Signal Processing, 2005*, pages 829–834. IEEE.

[Rajih *et al.*, 2008] RAJIH, M., COMON, P. et HARSHMAN, R. A. (2008). Enhanced line search : A novel method to accelerate parafac. *SIAM journal on matrix analysis and applications*, 30(3):1128–1147.

[Rauhut *et al.*, 2017] RAUHUT, H., SCHNEIDER, R. et STOJANAC, Ž. (2017). Low rank tensor recovery via iterative hard thresholding. *Linear Algebra and its Applications*, 523:220–262.

[Rayens et Mitchell, 1997] RAYENS, W. S. et MITCHELL, B. C. (1997). Two-factor degeneracies and a stabilization of parafac. *Chemometrics and Intelligent Laboratory Systems*, 38(2):173–181.

[Ribeiro *et al.*, 2016] RIBEIRO, L. N., de ALMEIDA, A. L. et ZARZOSO, V. (2016). Enhanced block term decomposition for atrial activity extraction in atrial fibrillation ecg. *In 2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pages 1–5. IEEE.

[Rodriguez-Fernandez *et al.*, 2015] RODRIGUEZ-FERNANDEZ, N. J., AIRES, F., RICHAUME, P., KERR, Y. H., PRIGENT, C., KOLASSA, J., CABOT, F., JIMENEZ, C., MAHMOODI, A. et DRUSCH, M. (2015). Soil moisture retrieval using neural networks : Application to smos. *IEEE Transactions on Geoscience and Remote Sensing*, 53(11):5991–6007.

[Roohi *et al.*, 2016] ROOHI, S. F., ZONOOBI, D., KASSIM, A. A. et JAREMKO, J. L. (2016). Dynamic mri reconstruction using low rank plus sparse tensor decomposition. *In 2016 IEEE International Conference on Image Processing (ICIP)*, pages 1769–1773. IEEE.

[Roos *et al.*, 1997] ROOS, C., TERLAKY, T. et VIAL, J.-P. (1997). *Theory and algorithms for linear optimization : an interior point approach*. Wiley Chichester.

[Rouijel *et al.*, 2014] ROUIJEL, A., MINAOUI, K., COMON, P. et ABOUTAJDINE, D. (2014). Cp decomposition approach to blind separation for ds-cdma system using a new performance index. *EURASIP Journal on Advances in Signal Processing*, 2014(1):128.

[Royer *et al.*, 2011] ROYER, J.-P., THIRION-MOREAU, N. et COMON, P. (2011). Computing the polyadic decomposition of nonnegative third order tensors. *Signal Processing*, 91(9):2159–2171.

[Sahnoun et Comon, 2015] SAHNOUN, S. et COMON, P. (2015). Joint source estimation and localization. *IEEE Transactions on Signal Processing*, 63(10):2485–2495.

[Sanchez et Kowalski, 1990] SANCHEZ, E. et KOWALSKI, B. R. (1990). Tensorial resolution : a direct trilinear decomposition. *Journal of Chemometrics*, 4(1):29–45.

[Savas, 2003] SAVAS, B. (2003). Analyses and tests of handwritten digit recognition algorithms. *LiTH-MAT-EX-2003-01, Link ping University, Department of Mathematics*.

[Savas et Eldén, 2007] SAVAS, B. et ELDÉN, L. (2007). Handwritten digit classification using higher order singular value decomposition. *Pattern recognition*, 40(3):993–1003.

[Schmidt *et al.*, 2011] SCHMIDT, M., ROUX, N. L. et BACH, F. (2011). Convergence rates of inexact proximal-gradient methods for convex optimization. *arXiv preprint arXiv :1109.2415*.

[Shashua et Hazan, 2005] SHASHUA, A. et HAZAN, T. (2005). Non-negative tensor factorization with applications to statistics and computer vision. *In Proceedings of the 22nd international conference on Machine learning*, pages 792–799.

[Sidiropoulos et Bro, 2000] SIDIROPOULOS, N. D. et BRO, R. (2000). On the uniqueness of multilinear decomposition of n-way arrays. *Journal of Chemometrics : A Journal of the Chemometrics Society*, 14(3):229–239.

[Sidiropoulos *et al.*, 2000a] SIDIROPOULOS, N. D., BRO, R. et GIANNAKIS, G. B. (2000a). Parallel factor analysis in sensor array processing. *IEEE transactions on Signal Processing*, 48(8):2377–2388.

[Sidiropoulos *et al.*, 2000b] SIDIROPOULOS, N. D., GIANNAKIS, G. B. et BRO, R. (2000b). Blind parafac receivers for ds-cdma systems. *IEEE Transactions on Signal Processing*, 48(3):810–823.

[Smilde *et al.*, 2005] SMILDE, A., BRO, R. et GELADI, P. (2005). *Multi-way analysis : applications in the chemical sciences*. John Wiley & Sons.

[Sørensen et De Lathauwer, 2013] SØRENSEN, M. et DE LATHAUWER, L. (2013). Blind signal separation via tensor decomposition with vandermonde factor : Canonical polyadic decomposition. *IEEE Transactions on Signal Processing*, 61(22):5507–5519.

[Sra, 2012] SRA, S. (2012). Scalable nonconvex inexact proximal splitting. *Advances in Neural Information Processing Systems*, 25:530–538.

[Stegeman et Sidiropoulos, 2007] STEGEMAN, A. et SIDIROPOULOS, N. D. (2007). On kruskal's uniqueness condition for the candecomp/parafac decomposition. *Linear Algebra and its applications*, 420(2-3):540–552.

[Tappenden *et al.*, 2016] TAPPENDEN, R., RICHTÁRIK, P. et GONDZIO, J. (2016). Inexact coordinate descent : complexity and preconditioning. *Journal of Optimization Theory and Applications*, 170(1):144–176.

[Ten Berge et Sidiropoulos, 2002] TEN BERGE, J. M. et SIDIROPOULOS, N. D. (2002). On uniqueness in candecomp/parafac. *Psychometrika*, 67(3):399–409.

[ten Berge et Smilde, 2002] ten BERGE, J. M. et SMILDE, A. K. (2002). Non-triviality and identification of a constrained tucker3 analysis. *Journal of Chemometrics : A Journal of the Chemometrics Society*, 16(12):609–612.

[Thirion, 1995] THIRION, N. (1995). *Separation d'ondes en prospection sismique*. Thèse de doctorat, Grenoble INPG.

[Thurau *et al.*, 2009] THURAU, C., KERSTING, K. et BAUCKHAGE, C. (2009). Convex non-negative matrix factorization in the wild. *In 2009 Ninth IEEE International Conference on Data Mining*, pages 523–532. IEEE.

[Tibshirani, 1996] TIBSHIRANI, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society : Series B (Methodological)*, 58(1):267–288.

[Tomasi et Bro, 2006] TOMASI, G. et BRO, R. (2006). A comparison of algorithms for fitting the parafac model. *Computational Statistics & Data Analysis*, 50(7):1700–1734.

[Tseng, 2000] TSENG, P. (2000). A modified forward-backward splitting method for maximal monotone mappings. *SIAM Journal on Control and Optimization*, 38(2):431–446.

[Tseng, 2008] TSENG, P. (2008). On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 1.

[Tucker, 1963] TUCKER, L. R. (1963). Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15:122–137.

[Tucker, 1966] TUCKER, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

[Tucker *et al.*, 1964] TUCKER, L. R. *et al.* (1964). The extension of factor analysis to three-dimensional matrices. *Contributions to mathematical psychology*, 110119.

[Vasilescu et Terzopoulos, 2002] VASILESCU, M. A. O. et TERZOPOULOS, D. (2002). Multilinear analysis of image ensembles : Tensorfaces. *In European conference on computer vision*, pages 447–460. Springer.

[Vasilescu et Terzopoulos, 2003] VASILESCU, M. A. O. et TERZOPOULOS, D. (2003). Multilinear subspace analysis of image ensembles. *In 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–93. IEEE.

[Vasilescu et Terzopoulos, 2005] VASILESCU, M. A. O. et TERZOPOULOS, D. (2005). Multilinear independent components analysis. *In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 547–553. IEEE.

[Vasilescu et Terzopoulos, 2007] VASILESCU, M. A. O. et TERZOPOULOS, D. (2007). Multilinear projection for appearance-based recognition in the tensor framework. *In 2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.

[Villa *et al.*, 2013] VILLA, S., SALZO, S., BALDASSARRE, L. et VERRI, A. (2013). Accelerated and inexact forward-backward algorithms. *SIAM Journal on Optimization*, 23(3):1607–1633.

[Vu *et al.*, 2017] VU, X., CHAUX, C., THIRION-MOREAU, N., MAIRE, S. et CARSTEA, E. M. (2017). A new penalized nonnegative third-order tensor decomposition using a block coordinate proximal gradient approach : Application to 3d fluorescence spectroscopy. *Journal of Chemometrics*, 31(4):e2859.

[Wang *et al.*, 2019] WANG, D., CONG, F. et RISTANIEMI, T. (2019). Higher-order nonnegative candecomp/parafac tensor decomposition using proximal algorithm. *In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3457–3461. IEEE.

[Wang *et al.*, 2018] WANG, D., ZHU, Y., RISTANIEMI, T. et CONG, F. (2018). Extracting multi-mode erp features using fifth-order nonnegative tensor decomposition. *Journal of neuroscience methods*, 308:240–247.

[Wang *et al.*, 2003] WANG, H. *et al.* (2003). Facial expression decomposition. *In Proceedings ninth IEEE international conference on computer vision*, pages 958–965. IEEE.

[Wetzstein *et al.*, 2012] WETZSTEIN, G., LANMAN, D. R., HIRSCH, M. W. et RASKAR, R. (2012). Tensor displays : compressive light field synthesis using multilayer displays with directional backlighting.

[Williams *et al.*, 2018] WILLIAMS, A. H., KIM, T. H., WANG, F., VYAS, S., RYU, S. I., SHENOY, K. V., SCHNITZER, M., KOLDA, T. G. et GANGULI, S. (2018). Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis. *Neuron*, 98(6):1099–1115.

[Xu et Yin, 2013] XU, Y. et YIN, W. (2013). A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789.

[Zdunek et Cichocki, 2006] ZDUNEK, R. et CICHOCKI, A. (2006). Non-negative matrix factorization with quasi-newton optimization. *In International conference on artificial intelligence and soft computing*, pages 870–879. Springer.

[Zhang *et al.*, 2010] ZHANG, C.-H. *et al.* (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942.

[Zhang et Hager, 2004] ZHANG, H. et HAGER, W. W. (2004). A nonmonotone line search technique and its application to unconstrained optimization. *SIAM journal on Optimization*, 14(4):1043–1056.

[Zhang *et al.*, 2008] ZHANG, Q., WANG, H., PLEMMONS, R. J. et PAUCA, V. P. (2008). Tensor methods for hyperspectral data analysis : a space object material identification study. *JOSA A*, 25(12):3001–3012.

[Zhang *et al.*, 2017] ZHANG, Q., YANG, L. T., CHEN, Z. et LI, P. (2017). An improved deep computation model based on canonical polyadic decomposition. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 48(10):1657–1666.

[Zhang *et al.*, 2018] ZHANG, Q., YANG, L. T., YAN, Z., CHEN, Z. et LI, P. (2018). An efficient deep learning model to predict cloud workload for industry informatics. *IEEE transactions on industrial informatics*, 14(7):3170–3178.

[Zhang, 2010] ZHANG, T. (2010). Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11(3).

[Zhang *et al.*, 2016] ZHANG, Y., ZHOU, G., ZHAO, Q., CICHOCKI, A. et WANG, X. (2016). Fast nonnegative tensor factorization based on accelerated proximal gradient and low-rank approximation. *Neurocomputing*, 198:148–154.

[Zhong et Kwok, 2012] ZHONG, L. W. et KWOK, J. T. (2012). Efficient sparse modeling with automatic feature grouping. *IEEE transactions on neural networks and learning systems*, 23(9):1436–1447.

[Zhong et Kwok, 2014] ZHONG, W. et KWOK, J. (2014). Gradient descent with proximal average for nonconvex and composite regularization. *In Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Résumé

Dans le cadre du travail réalisé, nous nous sommes intéressés à la conception et le développement d'algorithmes de décomposition des tableaux de données multidimensionnels particuliers appelés tenseurs. Parmi les décompositions de ces tenseurs, nous nous intéressons à la décomposition Canonique Polyadique (CP). Différentes méthodes développées dans la littérature souffrent de certaines lacunes selon la nature des données traitées, la précision, le temps de calcul... L'objectif principal porte sur le développement de méthodes pour l'aide à l'estimation correcte d'un ensemble de matrices de la décomposition CP. Pour atteindre cet objectif, nous proposons cinq algorithmes pour le calculer de la décomposition CP dans le cas des données complexes et non-négative basé sur les algorithmes proximaux. Les résultats ont démontré le bon comportement de ces cinq algorithmes, ainsi qu'un meilleur compromis entre la précision et la vitesse de convergence par rapport aux algorithmes de la littérature.

Mots clefs : Traitement du signal, Tenseur, Canonique polyadique, Algorithmes proximaux.

Abstarct

In this work, we are interested in the design and development of algorithms for decomposing particular multidimensional data arrays called tensors. Among the decompositions of these tensors, we are interested in the Canonical Polyadic (CP) decomposition. Different methods have been developed in the literature suffer from certain shortcomings depending on the nature of the data processed, the accuracy, the computation time... The main objective is to develop new methods for the correct estimation of a set of matrices for the CP decomposition. To achieve this objective, we propose five algorithms for computing the CP decomposition in the case of complex and non-negative data based on proximal algorithms. The results have demonstrated the good behavior of these five algorithms, as well as a better compromise between accuracy and convergence speed compared to the CP decomposition algorithms in the literature.

Keywords: Signal processing, Tensor, Canonical polyadic, Proximal algorithms.