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 : CT35

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

Centre de Recherche *:* Centre de Recherches Mathématiques et Application de Rabat
Structure de Recherche : Laboratoire de Mathématiques, Informatique et Applications - Sécurité de L'Information " LABMIA-SI"
Discipline *: Mathématiques*
Spécialité *: Mathématiques Appliquées*

Présentée et soutenue le 24/11/2021 par :

Alaa EL ICHI

Calcul tensoriel et applications

JURY

Souad EL BERNOUSSI	PES, Université Mohammed V, Faculté des Sciences-Rabat	Présidente/ Rapportrice
Froilán M. DOPICO	PES, Université Charles III de Madrid	Examinateur/Rapporteur
Brahim BENOUAHMANE	PES, Université Hassan II -Casablanca	Examinateur /Rapporteur
Michela REDIVO-ZAGLIA	PES, Université de Padoue, Italie	Examinateur /Rapporteur
Khalide JBILOU	PES, Université du Littoral Côte d'Opale-France	Co-Directeur de Thèse
Rachid SADAKA	PES, Université Mohammed V, Ecole Normale Supérieure-Rabat	Directeur de Thèse

Année Universitaire : 2020/2021

Dédicaces

A MES CHERS PARENTS

ABDELJALIL ELICHI & HOSNA SANBI

Source inépuisable de tendresse, de patience et de sacrifice. Vos prières et vos Bénédictions m'ont été d'un grand secourstout au long de ma vie.

Quoique je puisse dire et écrire, je ne pourrais exprimer ma grande affection et ma profonde reconnaissance. Je n'espère ne jamais te décevoir, ni trahir ta confiance et tes sacrifices. Puisse Dieu tout puissant, te préserver et t'accorder santé, longue vie et Bonheur.

Ce mémoire est dédié également à mon frère ***HAMZA*** *et ma sœur* ***MAROUA*** *ainsi que* ***MA FAMILLE****.*

Aussi je dédie ce travail à mes amis en souvenirde nos merveilleux moments

Remerciements

Les travaux présentés ont été réalisés au *Laboratoire de Mathématiques, Informatique et Applications - Sécurité de L'Information " LABMIA-SI", Université Mohammed V, Faculté des Sciences de Rabat* sous la direction du Professeur **Rachid Sadaka** et *Laboratoire de Mathématiques Pures et Appliquées Joseph Liouville (LMPA), Université du Littoral Côte d'Opale, France* sous la direction du Professeur **Khalide JBILOU**.

Je tiens à remercier sincèrement grandement mon cher superviseur, Monsieur **Rachid SADAKA**, Professeur à *l'Université Mohammed V de Rabat* Directeur de mes travaux de recherches pour toute son attention, sa disponibilité qu'elle n'a jamais cessé de me prodiguer, je lui dis tout simplement merci.

Je voudrais aussi remercier grandement mon cher superviseur, Monsieur **Khalide JBILOU**, Professeur à *l'Université Littoral Côte d'Opale, France*. Sans aucun doute, cette thèse serait impossible sans lui. Un grand merci pour ses conseils, encouragements et son soutien. Ses connaissances dans la recherche m'ont influencé à chaque étape de mon parcours, je ne peux pas oublier qu'il m'a appris à rédiger mon premier article universitaire et à mettre mes idées en mots. Honnêtement, ce petit paragraphe ne peut exprimer à quel point je suis reconnaissant envers mon cher superviseur, je n'oublierai jamais ce qu'il a fait pour moi.

J'exprime ma profonde reconnaissance à Madamme **Souad El Bernoussi**, Professeur à *Université Mohammed V, Faculté des Sciences de Rabat* d'avoir accepté de juger ce travail et de présider le jury de ma thèse.

J'exprime ma profonde reconnaissance à Madamme **Souad El Bernoussi**, Professeur à *Université Mohammed V, Faculté des Sciences de Rabat* d'avoir accepté d'être rapportrice de ma thèse.

J'adresse un remerciement particulier à Monsieur **Froilán M. Dopico**, Professeur à *l'Universidad Carlos III de Madrid* d'avoir accepté d'examiner/rapporter ce travail et d'enrichir nos résultats par ses remarques et commentaires.

Je remercie vivement Monsieur **Brahim BENOUAHMANE**, Professeur à *l'Université Hassan II de Casablanca* qui a accepté de travailler en tant que examinateur/rapporteur de cette thèse et ont dû consacrer du temps et de l'énergie à une lecture approfondie qui a donné lieu à des commentaires dont la richesse et la pertinence ont effectivement guidé la finalisation du manuscrit.

Je tiens également à remercier Madame **Michela Redivo-Zaglia,** Professeur à *l'Université de Padoue, Italie* pour avoir accepté d'examiner/rapporter cette thèse. Il a consacré du temps et apporté son expertise jugement à mon travail

J'adresse toute ma gratitude à tous mes ami(e)s et à toutes les personnes qui m'ont aidé dans la réalisation de ce travail.

Enfin, les mots les plus simples étant les plus forts, j'adresse toute mon affection à ma famille.

Avant-propos

Cette thèse à été préparée à:

Laboratoire de Mathématiques Pures et Appliquées

Centre Universitaire de la Mi-Voix

Maison de la Recherche Blaise Pascal

50, Rue Ferdinand Buisson

CS 80699

Calais Cedex, France.

Fax: (33) 03 21 46 55 86.

Site: www.lmpa.univ-littoral.fr

Laboratoire de Mathématiques, Informatique et Applications, Sécurité de l'Information

Centre d'Etudes Doctorales en Sciences et Technologies de Rabat

Faculté des Sciences Rabat

4 Avenue Ibn Batouta

B.P. 1014

Rabat, Maroc.

Fax: (212) 37 77 54 61.

Résumé

Dans ces dernières années, plusieurs axes de recherches liées aux tenseurs (matrices multidimensionnelles) ont été exploités, optimisation non linéaire, algèbre multilinéaire, analyse non linéaire, Statistiques, langage de programmation Vu que dans la modélisation de plusieurs phénomènes, la représentation tensorielle à prouver efficacité face à celle matricielle. En effet, les données réelles traitées à partir des différents problèmes entraînants souvent des demandes écrasantes sur les ressources informatiques (stockage, coût de calcul). Convaincus par l'efficacité de l'approche tensorielle, nous proposons dans cette thèse, une étude qui cible la généralisation des méthodes numériques matricielles via le formalisme tensoriel. En particulier, nous nous sommes concentrés sur l'extension des méthodes des sous espaces de Krylov et leurs applications. Plus précisément, nous avons introduit les méthodes de sous espace de Krylov tensorielles, type global et par blocs liés aux différents produits tensoriels existant déjà. Nous avons même réussi à développer une nouvelle catégorie de ces méthodes nommée méthodes des sous espaces de Krylov tubales. De plus, nous avons proposé des applications de ces méthodes dans la résolution des systèmes tensoriels linéaires et non linéaires, la restauration des images et vidéos, ainsi que la résolution des équations tensorielles de Sylvester. Nous traitons aussi les méthodes liées aux machines Learning et la reconnaissance faciale.

Mots-clefs (5) : Tenseurs, sous espace de Krylov, produit d'Einstein, produit à n mode, T-Product, produit cosine, traitement des images et des vidéos, équations de Sylvester, reconnaissance faciale, extrapolation.

Abstract

In recent years, several research areas related to tensors (multidimensional matrices) have been exploited, non linear optimization, multilinear algebra, non linear analysis, statistics, programming language. In the modeling of several phenomena, the tensor representation proves efficiency in front of the matrix one. Indeed, the real data processing from different problems often leading to overwhelming demands on the computer resources (storage, calculation cost. . .). Convinced of the efficiency of the tonsorial approach, we propose in this thesis, a study that targets the generalization of numerical matrix methods via tonsorial formalism. In particular, we have focused on the extension of the Krylov subspace methods and their applications. More precisely, we have introduced tonsorial global and block Krylov subspace methods related to the various tensor products already existing.

We have even succeeded in developing a new class of these methods named tubal Krylov subspace methods. Moreover, we propose applications of these methods in the resolution of linear and nonlinear tensor systems, image and video restoration, as well as the resolution of Sylvester tensor equations.

Key Words (5): Tensors, Krylov subspace, Einstein product, n mode product, T-product, Cosine product, image and video processing, Sylvester equations, facial recognition, Extrapolation.

Résumé

Dans ces dernières années, plusieurs axes de recherches liées aux tenseurs (matrices multidimensionnelles) ont été exploités, optimisation non linéaire, algèbre multilinéaire,analyse non linéaire, Statistiques, langage de programmation……Vu que dans la mod- élisation de plusieurs phénomènes, la représentation tensorielle à prouver efficacité face à celle matricielle. En effet, les données réelles traitées à partir des différents prob- lèmes entraînants souvent des demandes écrasantes sur les ressources informatiques (stockage, coût de calcul).
Convaincus par l'efficacité de l'approche tensorielle, nous proposons dans cette thèse, une étude qui cible la généralisation des méthodes numériques matricielles via le for- malisme tensoriel. En particulier, nous nous sommes concentrés sur l'extension des méthodes **des sous espaces de Krylov** et leurs applications. Plus précisément, nous avons introduit les méthodes de sous espace de Krylov tensorielles, type global et par blocs liés aux différents produits tensoriels existant déjà. Nous avons même réussi à développer une nouvelle catégorie de ces méthodes nommée méthodes des sous- espaces de Krylov tubales. De plus, nous avons proposé des applications de ces méth- odes dans la résolution des systèmes tensoriels linéaires et non linéaires, la restauration des images et vidéos, ainsi que la résolution des équations tensorielles de Sylvester. Nous traitons aussi les méthodes liées aux machines Learning et la reconnaissance fa- ciale. En d'autres termes, nous proposons une généralisation des méthodes numériques d'identification et reconnaissance faciale à savoir l'analyse linéaire discriminante et l'analyse multilinéaire discriminantes. via plusieurs produits tensoriels.

Nous développons aussi dans cette thèse un autre axe de l'analyse numérique, il s'agit

des méthodes d'accélération des suites " **L'extrapolation**". Nous proposons un ensemble de méthodes numériques basées sur le formalisme tensoriel qui généralisent les méthodes d'extrapolation vectorielles et matricielles.

- Une meilleure visualisation des données, non seulement réserve les informations initiales du modèle mathématique, mais permet la conservation de ses caractéristiques.
- Un bon traitement des données.
- Une meilleure exploitation des différentes techniques de stockage des données.
- L'efficacité des calculs (bonne estimation, meilleure précision ...).

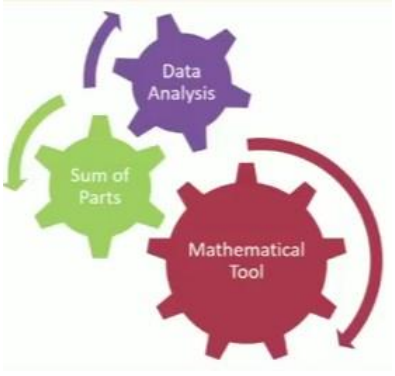


Figure 1: *Tenseur : outil mathématique pour résoudre les problèmes de big data*

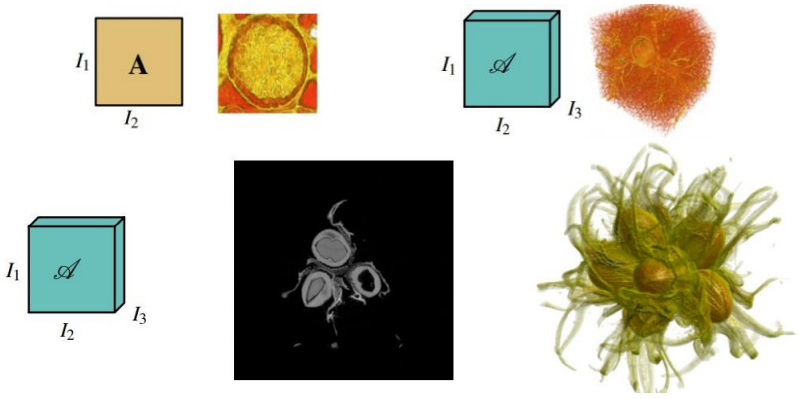


Figure 2: *Visualisation des données des tenseurs de 2ème et 3ème ordre respectivement*

En général, dans la majorité des problèmes, la représentation matricielle est insuffisante pour la représentation des données, elle crée des modèles mathématiques volumineux

et instables, et fournit des algorithmes très complexes, longs et moins précis, d'où la nécessité d'utiliser des tenseurs. Dans les représentations d'images des données acquises par exemple, la représentation des images couleur avec une matrice n'est pas pratique (qui serait de grandes dimensions), il est plus naturel de les représenter sous la forme d'une matrice multidimensionnelle (tenseurs d'ordre 3). Même chose pour les vidéos, qui peuvent être vues comme un tenseur d'ordre supérieur à 3. [49, 50].

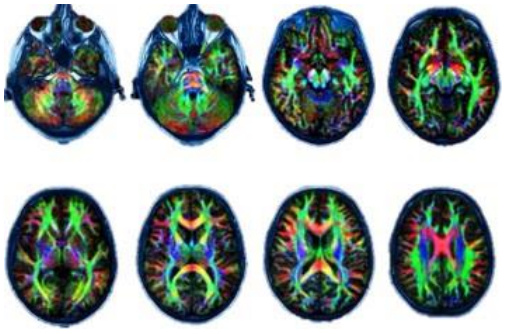


Figure 3: *IMagerie médicale : implique naturellement la 3D (spatio) et 4D (spatio-temporel) corrélations*

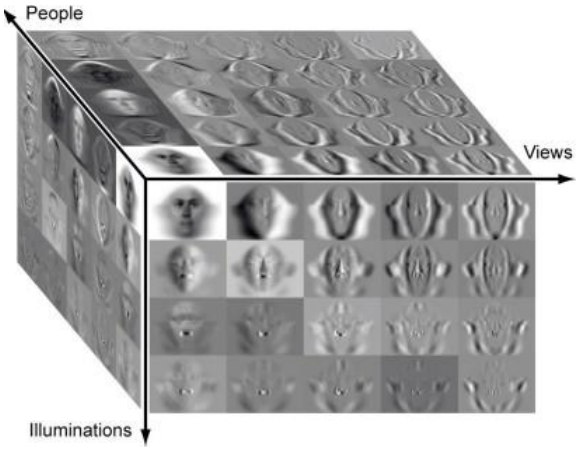


Figure 4: *Machine vision: comprendre le monde en 3D*

0.1 Les enjeux de la thèse

Certes, durant ces dernières années, plusieurs travaux ont été attribués à l'application des méthodes tensorielles aux sciences appliquées. Par exemple la résolution des équations différentielles [13, 109], la décomposition tensorielle [89, 90, 91]. Mais il est encore difficile de les considérer comme un moyen indispensable dans l'étude de certains problèmes scientifiques. La base théorique associée à ce type de raisonnement étant encore faible, il lui manque toute une théorie de base (produit, inverse . . .). D'où l'intérêt de notre travail de recherche, qui vise le développement de méthodes matricielles numériques pour la résolution de problèmes de grandes dimensions, ainsi que l'application de ces méthodes à des problèmes qui se posent dans la science et l'industrie. De manière générale, les principaux axes de recherche traités sont :

- **Méthodes de sous-espace de Krylov tensorielles** : L'objectif des processus de Krylov est de construire une base orthonormale d'un sous-espace de Krylov ou deux bases bi-orthonormales de deux sous-espaces de Krylov. Les trois types de processus de Krylov sont les processus de Krylov standard, les processus de Krylov globaux [79, 84] et par blocs qui s'appliquent respectivement aux sous-espaces de Krylov standard, global et par blocs. Récemment, Biek et al. [13, 14, 16, 109] ont réussi à développer la version tensorielle des méthodes de sous-espace de Krylov "**Méthodes de sous-espace de Krylov tensorielles** ". De même, Huang et al. [77] ont proposé une autre version tensorielle de ces méthodes.
  Dans cette dissertation, nous introduisons autres approches liées aux méthodes du sous-espace de Krylov tensorielles; voir le chapitre 3.

- **Méthodes tensorielles pour le traitement des images et des vidéos** : La restauration d'images est une étape très importante dans le processus de traitement des images. Cette dernière est nécessaire pour corriger les différents traitements que peut subir une image, car celle-ci, lors de sa capture peut être, dans la plupart des cas, sujette à des dégradations qui conduisent à l'absence des informations utiles qu'elle contenait et qui sont de grande importance. La représentation la plus utilisée d'une image numérique est celle d'un tableau bidimensionnel composé

d'un ensemble de lignes et de colonnes. Mais, cette représentation n'est plus pratique. Par exemple, la Figure 1.6 illustre qu'une image en niveaux de gris peut être stockée sous forme de tableau 2D, une image en couleurs RGB (Red (rouge), Green (vert), Blue (bleu)) peut être représentée sous forme de tableau 3D, composée de trois images en niveaux de gris pour chaque couleur. illustrées sous la forme d'un tableau 3D composé de centaines d'images en niveaux de gris couvrant une gamme continue de fréquences, voir la Figure 1.6.

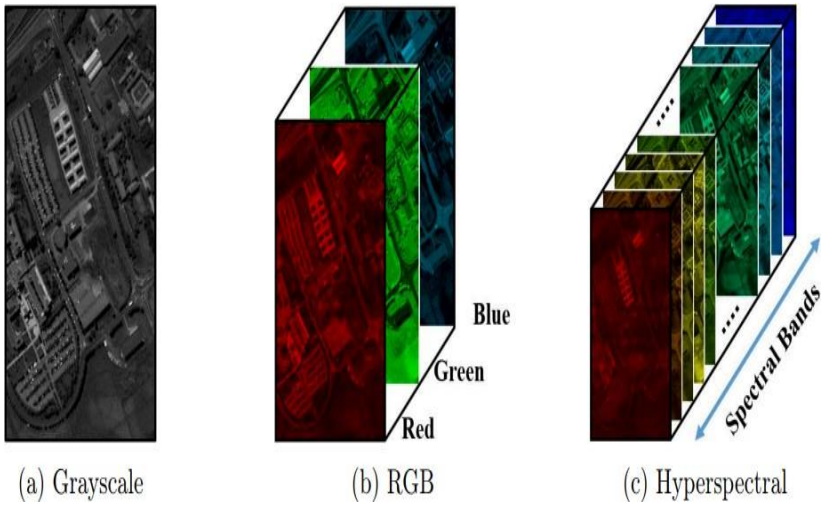


Figure 5: *Exemples de types d'images utilisant des échantillons en niveaux de gris*

Un autre exemple illustré par la Figure 1.7 montre que le remodelage d'une image 3D en représentation matricielle digitale est insuffisant, il entraîne une perte d'informations. De plus, la taille des matrices devient de plus en plus grande, ce qui posera un problème par la suite lors de la phase de traitement de l'image. Même chose pour les images multi spectrales, la représentation numérique en 2D de ces images détruit l'ordre naturel des données ; voir Figure 1.8. De plus, la Figure 1.9 montre qu' une vidéo peut être représentée comme un tenseur d'ordre 3, puisqu'une vidéo est juste une image (Red, Green, Blue) (RGB) qui se déplace dans le temps.

Plusieurs travaux traitent de différents problèmes liés à l'imagerie. Par exemple,

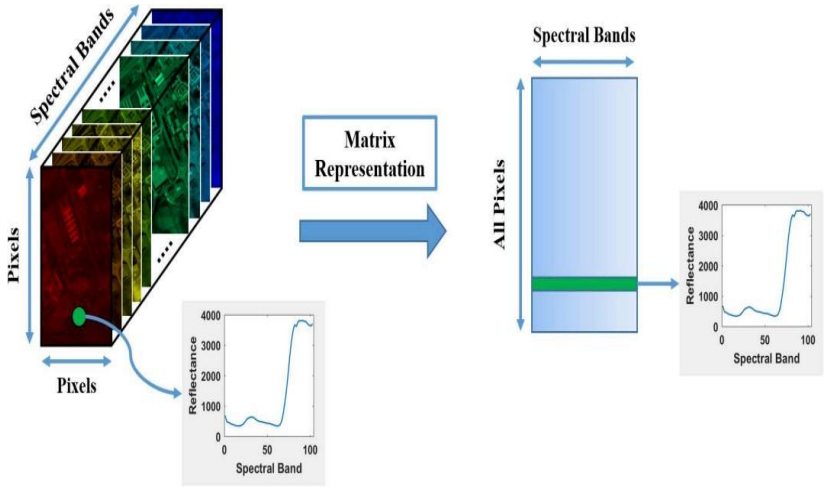


Figure 6: *Exemple de remodelage d'une image du type RGB en matrice 2D*

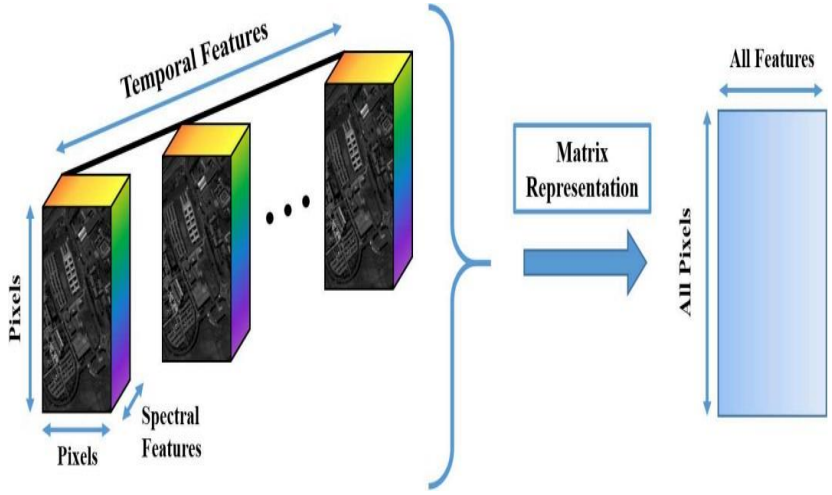


Figure 7: *Exemple de remodelage d'une image multi-spectrale en matrice 2D*

El Guide et al. [17, 18] proposent les **méthodes de sous-espace de Krylov matriciel** (GMRES, bidiagonalisation Golub Kahan) pour la résolution / restauration d'images. Par ailleurs, Biek et al. [13, 14] ont introduit la version tensorielle des **méthodes du sous-espace de Krylov** pour la restauration d'images.
En suivant la même idée, nous proposons dans les chapitres 4, 5 et 6 des approches basées sur le formalisme tensoriel pour la résolution des problèmes d'images et de vidéos.

- **Méthodes numériques pour résoudre les équations de Lyapunov et de Sylvester**

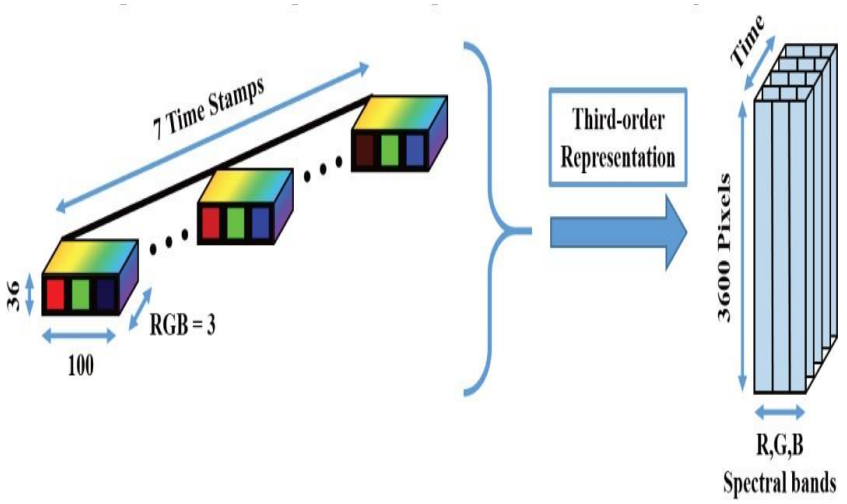


Figure 8: *Exemple de représentation d'une vidéo en tenseur de troisième ordre*

tensorielles : Ces dernières années, plusieurs travaux ont été développés pour la résolution des équations tensorielles de Sylvester, en tant que généralisation des méthodes matricielles de résolution de ces équations (Méthodes directes [8, 63], **méthodes du sous-espace de Krylov** pour la résolution des équations matricielles de Lyapunov/ Sylvester de grande dimension [47, 75]). De plus, des nouveaux travaux [13, 14, 109] qui se fonde sur le cadre tensoriel proposent de nouvelles méthodes basées sur le formalisme tensoriel pour la résolution des équations tensorielles de Sylvester. En suivant le même principe, nous visons la résolution de ces équations en introduisant de nouvelles approches tensorielles plus efficaces en termes de rapidité et de précision (chapitre 7).

- **Méthodes d'extrapolation tensorielles** : Des méthodes d'extrapolation scalaire, vectorielle ou matricielle ont été développées pour accélérer la convergence de certaines suites scalaires. En effet, l'extrapolation est une des techniques essen- tielles en analyse numérique, elle repose sur l'idée de construire une transforma- tion $(T_n)_{n\in \mathbb{N}}$ qui permet d'accélérer la convergence d'une suite donnée $(S_n)_{n\in \mathbb{N}}$. En pratique, les séquences $(S_n)_{n\in \mathbb{N}}$ qui décrivent les phénomènes (physiques, sociaux, etc.) se comportent mal, soit le temps de convergence est très long, ce qui implique l'augmentation des opérations élémentaires, soit la propagation des

erreurs et le temps d'exécution . L'une des premières méthodes d'extrapolation est le procédé Δ^2 d'Aitken [28]. Des généralisations ont ensuite été proposées au cours des dernières années. Parmi elles, la méthode Minimal Polynomial Extrapolation (MPE) [33, 104], la méthode Reduced Rank Extrapolation (RRE) [45], la Modified Minimal Polynomial Extrapolation (MMPE) [25, 115] et Topological *e*-algorithm (TEA) [25]. Pour plus de détails, voir [28, 79, 84, 86, 125, 126]. Jbilou et al. [84] ont introduit les **methodes d'extrapolation matricielles** par considération cette fois d'une séquence matricielle. Ils définissent la version matricielle (Block/Global) de MPE , RRE, MMPE et TEA. Inspirés par tous ces travaux, nous avons eu l'idée de considérer une suite tensorielle, ce qui donnera naissance aux **Méthodes d'extrapolation tensorielles**. (Chapitres 8 et 9).

Mots clés: Tenseurs, sous espace de Krylov , produit d'Einstein , produit à n mode , T-product, produit cosine , traitement des images et des vidéos, équations de Sylvester , reconnaissance faciale, extrapolation

Contents

Avant-propos **iii**
0.1 Les enjeux de la thèse ... viii
0.2 Thesis challenge ... xvi

List of Tables **xxv**

List of Figures **xxviii**

1 Introduction générale en français **1**
1.1 Aperçu général ... 1
1.2 Les enjeux de la thèse ... 4
1.3 Préliminaires ... 9
1.3.1 Produit à n modes (n-mode product) ... 11
1.3.2 Décompositions tensorielles ... 12
1.3.2.1 CANDECOMP decomposition ... 13
1.3.2.2 Notion de rang des tenseurs ... 14
1.4 Produit d'Einstein (Einstein product) ... 16
1.5 T-product ... 19
1.5.1 Transformation de Fourier discrète ... 19
1.6 Cosine product (c-product) ... 21
1.6.1 Discrete Cosine Transformation ... 21
1.7 Publications ... 22
1.8 Aperçu du manuscrit ... 22

2 Introduction **27**
2.1 Short overview ... 27
2.2 Thesis challenge ... 29
2.3 Preliminaries ... 34
2.3.1 n-mode product ... 36
2.3.2 Tensor decompositions ... 38
2.3.2.1 CANDECOMP decomposition ... 38
2.3.2.2 Notion of rank for tensors ... 39
2.4 Einstein product ... 42
2.5 T-product ... 45
2.5.1 Discrete Fourier Transformation ... 45
2.6 Cosine product (c-product) ... 46
2.6.1 Discrete Cosine Transformation ... 46
2.7 Publications ... 47

2.8 Manuscript Overview 48

3 Tensor tubal-Krylov subspace methods using the T-product 51
3.1 Introduction 51
3.2 Definitions and notations 52
3.2.1 Definitions and properties of the T-product 52
3.2.2 New tensor products 56
3.2.2.1 The T-Kronecker and the Tubal-inner products 57
3.2.2.2 T-Diamond product of third order tensor format 59
3.3 The tensor tubal global GMRES method 61
3.3.1 Tubal global QR factorization 61
3.3.2 The tensor tubal-global Arnoldi process 64
3.3.3 The tensor Tubal-Global GMRES method 69
3.3.4 Implementation of tensor Tubal-Global GMRES method 72
3.4 Tensor tubal-global Golub Kahan algorithm 74
3.5 Numerical experiments 77
3.5.1 Example1 79
3.5.2 Example 2 79
3.6 Conlusion 81

4 Tensor GMRES and Golub-Kahan Bidiagonalization methods via the Einstein product with applications to image and video processing 83
4.1 Introduction 84
4.2 Definitions and Notations 86
4.3 Krylov subspace methods via Einstein product and Tikhonov regularization 87
4.3.1 Tikhonov regularization 87
4.3.2 The Global GMRES method via Einstein product 88
4.3.3 The tensor Golub–Kahan method via the Einstein product 93
4.4 The tensor block GMRES process 98
4.4.1 The tensor block Arnoldi process via the Einstein product 98
4.4.2 The tensor Block GMRES method 101
4.5 Tensor Block Golub Kahan method 103
4.6 Numerical results 105
4.6.1 Example 1 108
4.6.2 Example 2 109
4.7 Conclusion 112

5 Tensor GMRES and Golub Kahan methods via the T-product for color image processing 113
5.1 Introduction 114
5.2 Definitions and Notations 115

5.3 Tensor T-global GMRES and tensor T-global Golub-Kahan algorithms 116
5.3.1 The tensor T-global GMRES 116
5.3.2 The Tensor T-global Golub Kahan algorithm 119
5.4 Application to discrete-ill posed tensor problems 121
5.5 Numerical results 128
5.5.1 Example 1 130
5.5.2 Example 2 134
5.6 Conclusion 135

6 Discrete cosine transform LSQR methods for multidimensional ill-posed problems 137
6.1 Introduction 137
6.2 Properties of the cosine product 138
6.3 Properties of the cosine product 140
6.4 Tensor discrete cosine global Krylov subspace methods 142
6.4.1 The tensor discrete cosine GMRES 143
6.4.2 The tensor discrete cosine Golub-Kahan method 146
6.4.3 The discrete cosine-LQSR method 148
6.5 Numerical results 151
6.5.1 Example 1 153
6.5.2 Example 2 155
6.6 Conclusion 157

7 Tensor Krylov subspace methods via the T-product for large Sylvester Tensor Equations 159
7.1 Introduction 160
7.2 Definitions and notations 161
7.3 Tensor tFOM and tGMRES algorithms 161
7.3.1 The tFOM method 161
7.3.2 The tGMRES method 163
7.4 t-Bartels-Stewart method for Sylvester tensor equations of small size 165
7.5 Tensor Krylov methods via T-product for solving large Sylvester tensor equations 168
7.5.1 Tubal block Arnoldi method 169
7.5.2 Tubal block Arnoldi method for solving large Sylvester tensor equations (TBAS) 172
7.6 Numerical experiments 175
7.6.1 Example 1 177
7.6.2 Example 2 178
7.7 Conclusion 179

8 Tensor extrapolation methods using the T-product 181
8.1 Introduction 181
8.2 Definitions and new tensor products 183
8.2.1 Definitions and properties 183
8.2.2 New tensor products 186

8.3 Extrapolation methods based on tensor formats 189
8.3.1 Tensor polynomial-type extrapolation methods 189
8.3.2 The tensor toplogical *e*- transformation 194
8.4 Application of tensor extrapolation methods to solve ill-posed tensor problems 195
8.4.1 TTSVD: Truncated Tensor Singular Value Decomposition 195
8.4.2 Tensor extrapolation methods applied to TTSVD sequences in ill-posed problems 201
8.5 Numerical experiments 208
8.6 Conclusion 209

9 Tensor global extrapolation methods using the n-mode and the Einstein products 211
9.1 Introduction 211
9.2 Preliminaries and notations 212
9.3 Tensor extrapolation methods 213
9.3.1 Tensor global-polynomial extrapolation methods 213
9.3.2 The tensor global topological *e*-transformation 219
9.4 Application to tensor linear/ non linear systems of equations 221
9.4.1 Application to tensor linear systems 221
9.4.2 Application to non linear tensor systems 223
9.5 The global-QR implementation of TG-MPE/TG-RRE 225
9.6 Conclusion 229

10 Conclusions and perspectives 231

Bibliography 233

List of Tables

3.1 Results for Example 1. $e = 10^{-6}$, $s = 5$ and $n_3 = 4$ 79
3.2 Results for Example 2. $n_3 = m_0^2$ $s = 3$ and $e = 10^{-6}$ 81

4.1 Results for Example 1. 109
4.2 Results for Example 2. 111

5.1 Results for Example 1. 132
5.2 Results for Example 2. 135

6.1 Results for Experiment 1 with papav256. Noise level 10^{-3} 154
6.2 Results for Example 1 with papav256. Noise level 10^{-2} 154
6.3 Results for Example 2 with noise level 10^{-3} 155
6.4 Results for Example 2 with noise level 10^{-2} 156
6.5 Results for DC-Golub-Kahan, T-Golub Kahan, T-LSQR and DC-LSQR, with noise level 10^{-2} with the image cat1024 157

7.1 Results for Example 2. $e = 10^{-6}$, and $n_3 = 2$ 178
7.2 Results for Example 2. $e = 10^{-6}$, $\mu = 1$ and $n_3 = 2$ 179
7.3 Results for Example 2. $e = 10^{-6}$, $\mu = 10$ and $n_3 = 2$ 179

8.1 The computed relative errors for TTSVD and TRRE-TTSVD ($v = 10^{-2}$) 209
8.2 The computed relative errors for TTSVD and TRRE-TTSVD ($v = 10^{-3}$) 209

List of Algorithms

1 Computing the T-product via FFT 54

2 Tensor T-trace 56

3 Tensor T-Kronecker 58

4 Tensor T-Diamond product 60

5 A normalization algorithm (Normalization(ffi)) 62

6 Tensor Tubal-Global QR decomposition 62

7 The Tensor Tubal-Global Arnoldi 65

8 Tensor T-QR decomposition 72

9 The Tensor Tubal-Global GMRES (m) 74

10 The Tensor Tubal-global Golub Kahan algorithm 75

11 Tensor Tubal-Global Golub Kahan (TTGK) 78

12 Tensor Global Arnoldi process via Einstein product 88

13 Global GMRES method via Einstein product 93

14 Global Golub–Kahan algorithm via Einstein product 93

15 GGKB and quadrature rules method for Tikhonov regularization via Einstein product 98

16 The tensor block Arnoldi process via the Einstein product 100

17 The tensor block Golub Kahan algorithm 104

18 Tensor T-global Arnoldi 116

19 The Tensor T-global Golub Kahan algorithm 120

20 Implementation of Tensor T-global GMRES(m) 123

21 Tensor Global Golub Kahan process (T-GGK) and quadrature rules method for Tikhonov regularization 127

22 Computing the c-product 141
23 Tensor discrete cosine Arnoldi 143
24 Restarted tensor discrete cosine GMRES (DC-GMRES(m)) method with Tikhonov regularization 146
25 The Tensor discrete cosine Bidiagonalisation Golub Kahan algorithm 146
26 The Tensor Discrete Cosine Golub-Kahan (DC-GK) method 148
27 The Discrete Cosine LSQR (DC-LSQR) algorithm 151
28 tFOM and tGMRES Algorithms 166
29 t-real Schur decomposition 167
30 t-Bartels-Stewart method 168
31 Tubal-QR Factorization (Tubal-QR) 170
32 The Tubal Block Arnoldi Algorithm (TBA) 170
33 The Tubal Block Arnoldi for solving large Sylvester tensor equation TBAS(m) 175
34 Matlab pseudocode for TSVD 201
35 Matlab pseudocode for TTSVD and the corresponding approximation of Moore–Penrose inverse 201
36 The TRRE-TTSVD Algorithm 207
37 TG-RRE, TG-MPE and TG-MMPE Algorithms 223
38 TG-RRE, TG-MPE and TG-MMPE for nonlinear systems 224
39 The global-QR decomposition 226
40 Implementation of TG-MPE/TG-RRE via the global-QR decomposition 228

List of Figures

1 Tenseur : outil mathématique pour résoudre les problèmes de big data vi
2 Visualisation des données des tenseurs de 2ème et 3ème ordre respectivement..... vi
3 **Imagerie médicale** : implique naturellement la 3D (spatio) et 4D (spatio-temporel) corrélations vii
4 **Machine vision**: comprendre le monde en 3D vii
5 Exemples de types d'images utilisant des échantillons en niveaux de gris ix
6 Exemple de remodelage d'une image du type RGB en matrice 2D x
7 Exemple de remodelage d'une image multi-spectrale en matrice 2D..... x
8 Exemple de représentation d'une vidéo en tenseur de troisième ordre..... xi
9 The "Geometry" of the Tensor Research Community..... xiv
10 Tensor: mathematical tool for solving big data problems xv
11 Data visualization of 2nd and 3rd order tensor respectively . xv
12 **Medical imaging**: naturally involves 3D (spatio) and 4D (spatio-temporal) correlations..... xvi
13 **Machine vision**: understanding the world in 3D..... xvi
14 Examples of image types using grayscale samples xviii
15 Example of reshaping a RGB image types into 2D matrix xviii
16 Example of reshaping a multi spectral image types into 2D matrix..... xix
17 Example of representation of a video in third order tensor . . xix

1.1 La "géométrie" de la communauté de recherche tensorielle.....2
1.2 Tenseur : outil mathématique pour résoudre les problèmes de big data.....2
1.3 Visualisation des données des tenseurs de 2ème et 3ème ordre respectivement.....3
1.4 **Imagerie médicale** : implique naturellement la 3D (spatio) et 4D (spatio-temporel) corrélations 3
1.5 **Machine vision**: comprendre le monde en 3D 4
1.6 Exemples de types d'images utilisant des échantillons en niveaux de gris.....6
1.7 Exemple de remodelage d'une image du type RGB en matrice 2D 6

1.8 Exemple de remodelage d'une image multi-spectrale en matrice 2D 7
1.9 Exemple de représentation d'une vidéo en tenseur de troisième ordre 7
1.10 Illustration des données à voies multiples : tenseur à voie nulle = scalaire, tenseur à voie unique = vecteur ligne ou colonne, tenseur à voie double = matrice, tenseur à *N* voie = tenseurs d'ordre supérieur. Les tenseurs à 4 et 5 voies sont représentés ici comme un ensemble de tenseurs à trois voies. 9
1.11 Un tableau à trois voies (tenseur du troisième ordre) 10
1.12 Fibres : pour les tenseurs du troisième ordre..... 10
1.13 Tranches pour un tenseur du troisième ordre..... 10
1.14 Illustration d'un tenseur de troisième ordre de rang un..... 12
1.15 Illustration du dépliage (aplatissement, matricialisation) en ligne et en colonne d'un tenseur d'ordre 3..... 12
1.16 Dépliage (matricialisation) d'un tenseur d'ordre 3. Le tenseur peut être déplié de trois façons pour obtenir des matrices comprenant ses vecteurs mode-1, mode-2 et mode-3..... 13
1.17 Exemple de dépliage du tenseur du troisième ordre en mode-1, mode-2 et mode-3. 14
1.18 llustration des multiplications en mode-n d'un tenseur du troisième ordre par des matrices. (a) multiplication mode- 1. (b) multiplication mode- 2. (c) multiplication mode- 3 multiplication. 15
1.19 Illustration of mode-n multiplication of a third-order tensor 15

2.1 The "Geometry" of the Tensor Research Community..... 28
2.2 Tensor: mathematical tool for solving big data problems..... 28
2.3 Data visualization of 2nd and 3rd order tensor respectively . 29
2.4 **Medical imaging**: naturally involves 3D (spatio) and 4D (spatio-temporal) correlations 29
2.5 **Machine vision**: understanding the world in 3D 30
2.6 Examples of image types using grayscale samples 31
2.7 Example of reshaping a RGB image types into 2D matrix..... 32
2.8 Example of reshaping a multi spectral image types into 2D matrix..... 32
2.9 Example of representation of a video in third order tensor..... 33
2.10 Illustration of multi-way data: zero-way tensor = scalar, 1-way tensor = row or column vector, 2-way tensor = matrix, *N*-way tensor = higher-order tensors. The 4-way and 5-way tensors are represented here as a set of the three-way tensors. 35
2.11 A three-way array (third-order tensor). 35
2.12 Fibers: for a third-order tensors..... 36
2.13 Slices for a third-order tensor..... 36
2.14 Illustration of rank-one third-order tensor..... 37
2.15 Illustration of row-wise and column-wise unfolding (flattening, matricizing) of a thirdorder tensor 38

2.16 Unfolding (matricizing) of a third-order tensor. The tensor can be unfolded in three ways to obtain matrices comprising its mode-1, mode-2 and mode-3 vectors. 39
2.17 Example of unfolding the third-order tensor in mode-1, mode-2 and mode-3. 40
2.18 Illustration of the mode-n multiplications of a third-order tensor by matrices. (a) mode- 1 multiplication. (b) mode- 2 multiplication. (c) mode- 3 multiplication. 41
2.19 Illustration of mode-n multiplication of a third-order tensor. . 41

4.1 Example 1: Original image (top), blurred and noisy image (bottom). 110
4.2 Example 1: Restored image by Algorithm 4 (top), and restored image by Algorithm 2 (bottom). 111
4.3 Frame no. 5: Original frame (left), blurred and noisy frame (right). 112
4.4 Frame no. 5: Restored frame by Algorithm 4 (left), and restored frame by Algorithm 2 (right). 112

5.1 Example 1: Original RGB images: peppers (left), papav256 (right). 133
5.2 Example 1: Blurred and noisy images, peppers (left), papav256 (right). 133
5.3 Example 1: Restored images by Algorithm 20, peppers (left), papav256 (right). 133
5.4 Example 1: Restored images by Algorithm 21, peppers (left), papav256 (right). 133
5.5 Example 2: Original frame no. 5 (left), blurred and noisy frame no. 5 (right). 135
5.6 Example 2: restored frame no. 5 by Algorithm 20 (left), and restored frame no. 5 by Algorithm 21 (right). 136

6.1 Original RGB images: papav256 (left) and cat1024 (right). . . 153
6.2 Test for Example 1, with DC-LSQR for papav256, and noise level 10^{-3}. Original (left), noisy-blurred (center) and restored (right) 155
6.3 Test for Example 2, with DC-LSQR for cat1024, and noise level 10^{-3}. Original (left), noisy-blurred (center) and restored (right). 156

List of Acronyms

RGB (Red, Green, Blue)

LDA Linear Discriminant Analysis

MPE Minimal Polynomial Extrapolation

RRE Reduced Rank Extrapolation

MMPE Modified Minimal Polynomial Extrapolation

TEA Topological *e*-algorithm

GMRES Generalized minimal residual method

LSQR Least-squares method

tFOM tensor Full orthogonalization method

tGMRES tensor GMRES

TBA Tubal Block Arnoldi

TBAS Tubal Block Arnoldi method to sole large Sylvester tensor equation

CP CANDECOMP decomposition

TD Tucker decomposition

DFT Discrete Fourier Transformation

FFT Fast Fourier Transform

DCT Discrete Cosine Transformation

TGA Tubal Global Arnoldi

TSVD Tensor Singular Value Decomposition

TTSVD Truncated TSVD

TMPE Tensor Minimal Polynomial Extrapolation

TRRE Tensor Reduced Rank Extrapolation

TMMPE Tensor Modified Minimal Polynomial Extrapolation

TTEA Tensor Topological *e*-algorithm

TGGS Tubal-Global Gram-Schmidt

TGQR Tubal-Global QR factorisation

TTGA Tensor Tubal-Global Arnoldi

TTGMRES The restarted tensor tubal global GMRES

T-GGMRES The restarted tensor global GMRES

T-GGK Tensor Global Golub Kahan process

TTGK Tensor Tubal-Global Golub Kahan for linear tensor equation

TGK Tensor Global Golub Kahan

TBAE The tensor block Arnoldi process via Einstein product

TBGKE The tensor block Golub Kahan process via Einstein product

TBGMRESE The tensor block Golub Kahan process via Einstein product

GGKB Global Golub–Kahan bidiagonalization

GKB Golub–Kahan bidiagonalization

DCA Tensor Discrete Cosine Arnoldi

DC-GMRES Restarted tensor Discrete Cosine GMRES

DC-GK Tensor Discrete Cosine Golub-Kahan

DC-LSQR Tensor Discrete LSQR

TG-RRE Tensor Global Reduced Rank Extrapolation

TG-MPE Tensor Global Minimal Polynomial Extrapolation

TG-MMPE Tensor Global Modified Minimal Polynomial Extrapolation

List of Symbols

$\mathcal{X}$	Tensor of order ≥ 3	②	Tensor Kronecker product
$\mathcal{X}$	Tensor of order 3	♦	Tensor Diamond product (T-product)
$\langle .,. \rangle$	Inner product	$\blacklozenge_c$	Tensor Cosine Diamond product
$\mathcal{X}^T, \mathcal{X}^T$	Transpose of tensor	$TK_m(.,.)$	Tensor Krylov global subspace (T-product)
$\mathcal{X}^{-1}, \mathcal{X}^{-1}$	Tensor inverse	$TK_{\hat{m}}(.,.)$	Tensor tubal Krylov global subspace
X, X^T	Matrix, matrix transpose	$\boxtimes^{(N)}$	Tensor product
u, u^T	Vector, vector transpose	$K_m(.,.)$	Tensor Krylov subspace (Einstein product)
$\|\|.\|\|_F$	Frobinus norm	$\langle .,. \rangle_{T_{I_2}}$	Tensor inner product
$\otimes$	Matrix Kronecker product	$\|\|.\|\|_{T_{I_2}}$	Tensor norm
$\diamond$	Diamond matrix product	$TK_{\hat{m}}(.,.)$	Tensor Krylov global subspace (c-product)
$*_N$	Einstein product	②	Tensor vector product
$\times_n$	n mode product	$\|\|.\|\|_2$	Euclidean norm
$\tilde{\times}_n$	n mode vector product	vec	Vectorization
$*$	T-product	$\overrightarrow{ffi}$	Lateral slice
$*_c$	Cosine product	$©^{(N)}$	Tensor Diamond product (Einstein product)
$\circ$	Outer product	$\mathbb{R}$	The set of real numbers
$\mathbf{a}$	Tube fiber	$\mathbb{C}$	The set of complex numbers
$\div$	Tube tensor product		

Chapter 1

Introduction générale en français

1.1 Aperçu général

Au début du XIXe siècle, les mathématiciens et les physiciens avaient beaucoup de mal à exprimer de nombreuses lois naturelles. D'où la nécessité de créer un nouvel outil mathématique qui permettrait d'introduire un langage robuste, simple et fiable, capable de décrire et d'étudier les différents phénomènes naturels. C'est ainsi qu'est né un nouvel "être mathématique" appelé "tenseurs", qui sera utilisé par la suite, non seulement dans les travaux exploités par des physiciens comme Einstein qui avait exprimé les fameuses équations fondamentales de la théorie de la relativité générale en fonction de tenseurs, mais aussi dans de nombreux domaines scientifiques, la psychométrie et l'analyse de données; [70, 131, 132, 133], chimiométrie [29, 96], traitement du signal [36, 95, 112, 129], ingénierie biomédicale [103, 116, 117], exploration de données [91, 106, 113] et vision par ordinateur [97, 98, 123]. Lorsqu'il s'agit de données de grandes dimensions, l'approche tensorielle est plus efficace que les méthodes matricielles traditionnelles. L'estimation de l'émission en chimiométrie basée sur des décompositions tensorielles [29, 30], la compression de données [40], le traitement de réseaux [5, 122] et la

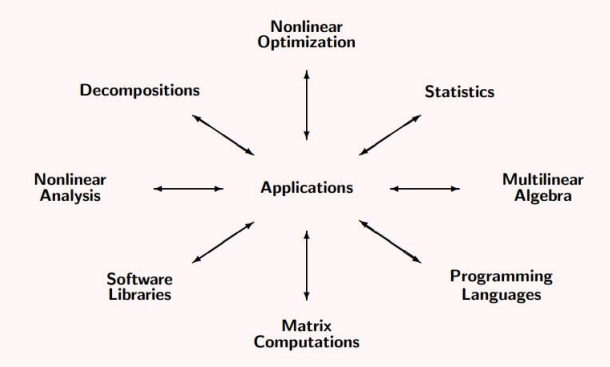


Figure 1.1: *La "géométrie" de la communauté de recherche tensorielle*

modélisation de systèmes [31, 54] sont tous des exemples célèbres qui illustrent l'avantage de l'approche tensorielle. En effet, la représentation multidimensionnelle (tensorielles) des données réelles permet :

* Une meilleure visualisation des données, non seulement réserve les informations initiales du modèle mathématique, mais permet la conservation de ses caractéristiques.
* Un bon traitement des données.
* Une meilleure exploitation des différentes techniques de stockage des données.
* L'efficacité des calculs (bonne estimation, meilleure précision ...).

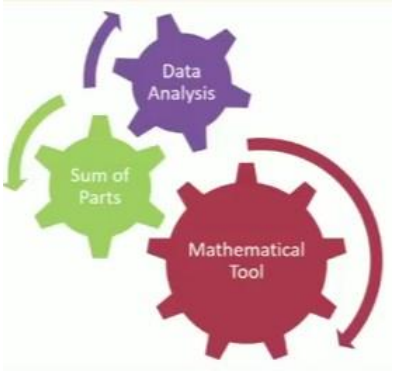


Figure 1.2: *Tenseur : outil mathématique pour résoudre les problèmes de big data*

En général, dans la majorité des problèmes, la représentation matricielle est insuffisante pour la représentation des données, elle crée des modèles

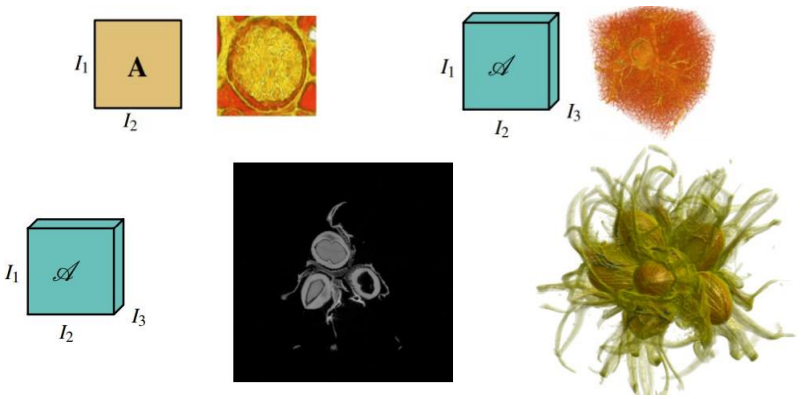


Figure 1.3: *Visualisation des données des tenseurs de 2ème et 3ème ordre respectivement*

mathématiques volumineux et instables, et fournit des algorithmes très complexes, longs et moins précis, d'où la nécessité d'utiliser des tenseurs. Dans les représentations d'images des données acquises par exemple, la représentation des images couleur avec une matrice n'est pas pratique (qui serait de grandes dimensions), il est plus naturel de les représenter sous la forme d'une matrice multidimensionnelle (tenseurs d'ordre 3). Même chose pour les vidéos, qui peuvent être vues comme un tenseur d'ordre supérieur à 3. [49, 50].

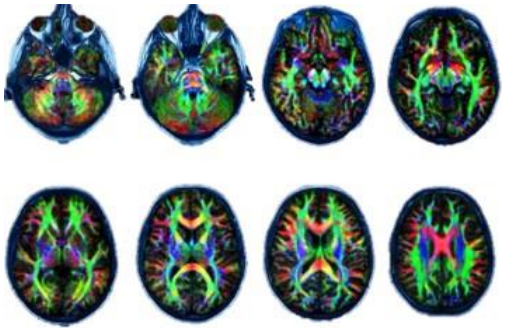


Figure 1.4: *IMagerie médicale : implique naturellement la 3D (spatio) et 4D (spatio-temporel) corrélations*

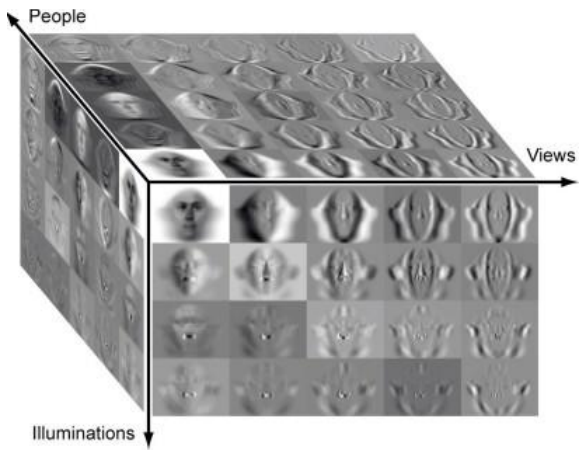


Figure 1.5: *Machine vision: comprendre le monde en 3D*

1.2 Les enjeux de la thèse

Certes, durant ces dernières années, plusieurs travaux ont été attribués à l'application des méthodes tensorielles aux sciences appliquées. Par exemple la résolution des équations différentielles [13, 109], la décomposition tensorielle [89, 90, 91]. Mais il est encore difficile de les considérer comme un moyen indispensable dans l'étude de certains problèmes scientifiques. La base théorique associée à ce type de raisonnement étant encore faible, il lui manque toute une théorie de base (produit, inverse . . .). D'où l'intérêt de notre travail de recherche, qui vise le développement de méthodes matricielles numériques pour la résolution de problèmes de grandes dimensions, ainsi que l'application de ces méthodes à des problèmes qui se posent dans la science et l'industrie. De manière générale, les principaux axes de recherche traités sont :

* **Méthodes de sous-espace de Krylov tensorielles** : L'objectif des processus de Krylov est de construire une base orthonormale d'un sous-espace de Krylov ou deux bases bi-orthonormales de deux sous-espaces de Krylov. Les trois types de processus de Krylov sont les processus de Krylov standard, les processus de Krylov globaux [79, 84] et par

blocs qui s'appliquent respectivement aux sous-espaces de Krylov standard, global et par blocs. Récemment, Biek et al. [13, 14, 16, 109] ont réussi à développer la version tensorielle des méthodes de sous-espace de Krylov "**Méthodes de sous-espace de Krylov tensorielles** ". De même, Huang et al. [77] ont proposé une autre version tensorielle de ces méthodes.

Dans cette dissertation, nous introduisons autres approches liées aux méthodes du sous-espace de Krylov tensorielles; voir le chapitre 3.

* **Méthodes tensorielles pour le traitement des images et des vidéos** : La restauration d'images est une étape très importante dans le processus de traitement des images. Cette dernière est nécessaire pour corriger les différents traitements que peut subir une image, car celle-ci, lors de sa capture peut être, dans la plupart des cas, sujette à des dégradations qui conduisent à l'absence des informations utiles qu'elle contenait et qui sont de grande importance. La représentation la plus utilisée d'une image numérique est celle d'un tableau bidimensionnel composé d'un ensemble de lignes et de colonnes. Mais, cette représentation n'est plus pratique. Par exemple, la Figure 1.6 illustre qu'une image en niveaux de gris peut être stockée sous forme de tableau 2D, une image en couleurs RGB (Red (rouge), Green (vert), Blue (bleu)) peut être représentée sous forme de tableau 3D, composée de trois images en niveaux de gris pour chaque couleur. illustrées sous la forme d'un tableau 3D composé de centaines d'images en niveaux de gris couvrant une gamme continue de fréquences, voir la Figure 1.6.

  Un autre exemple illustré par la Figure 1.7 montre que le remodelage d'une image 3D en représentation matricielle digitale est insuffisant, il entraîne une perte d'informations. De plus, la taille des matrices devient de plus en plus grande, ce qui posera un problème par la suite lors de la phase de traitement de l'image. Même chose pour les images multi spectrales, la représentation numérique en 2D de ces images détruit

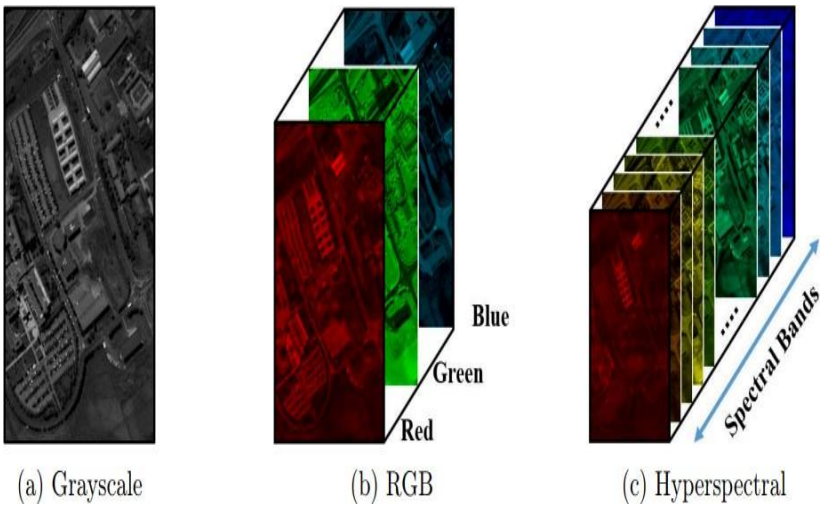


Figure 1.6: *Exemples de types d'images utilisant des échantillons en niveaux de gris*

l'ordre naturel des données ; voir Figure 1.8. De plus, la Figure 1.9 montre qu' une vidéo peut être représentée comme un tenseur d'ordre 3, puisqu'une vidéo est juste une image RGB qui se déplace dans le temps.

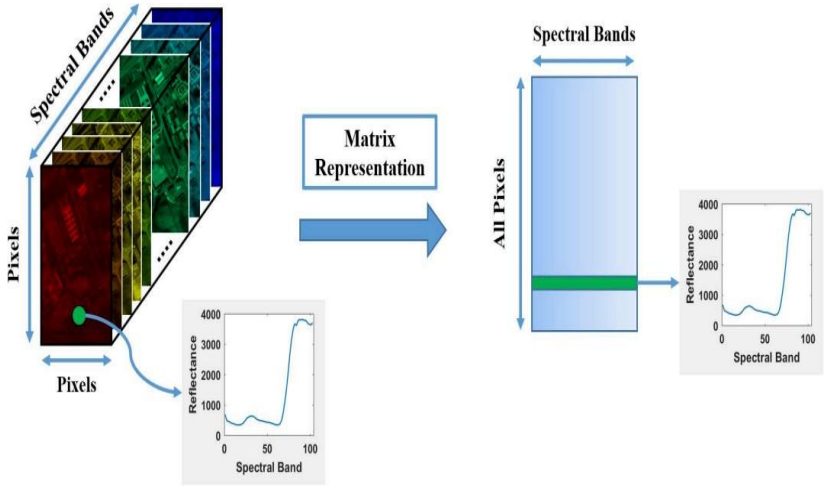


Figure 1.7: *Exemple de remodelage d'une image du type RGB en matrice 2D*

Plusieurs travaux traitent de différents problèmes liés à l'imagerie. Par exemple, El Guide et al. [17, 18] proposent les **méthodes de sous-espace de Krylov matriciel** (GMRES, bidiagonalisation Golub Kahan) pour la résolution / restauration d'images. Par ailleurs, Biek et al. [13, 14] ont

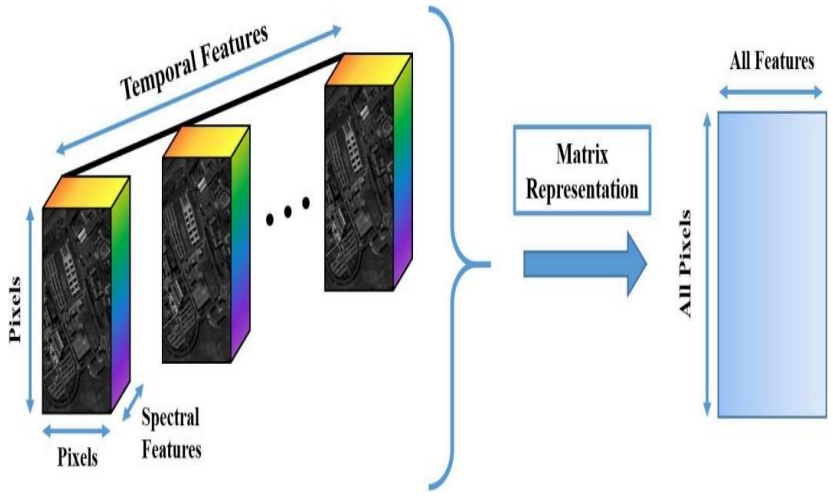


Figure 1.8: *Exemple de remodelage d'une image multi-spectrale en matrice 2D*

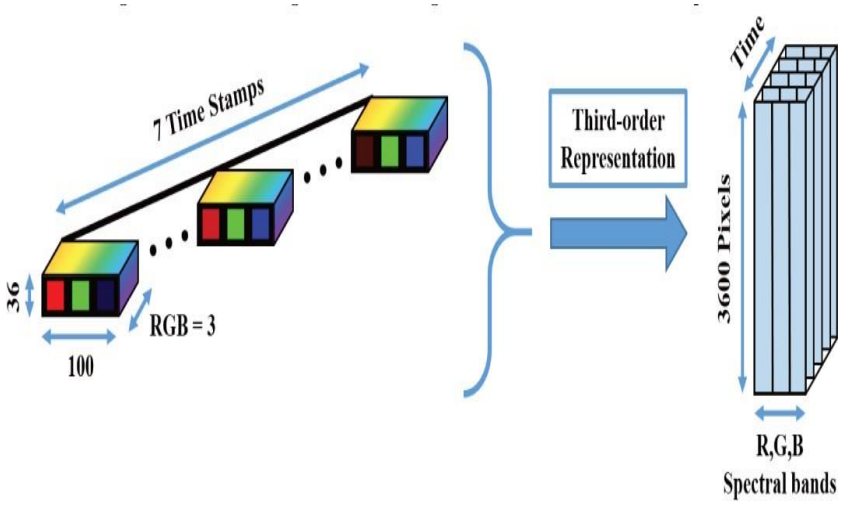


Figure 1.9: *Exemple de représentation d'une vidéo en tenseur de troisième ordre*

introduit la version tensorielle des **méthodes du sous-espace de Krylov** pour la restauration d'images.

En suivant la même idée, nous proposons dans les chapitres 4, 5 et 6 des approches basées sur le formalisme tensoriel pour la résolution des problèmes d'images et de vidéos.

* **Méthodes numériques pour résoudre les équations de Lyapunov et de Sylvester tensorielles** : Ces dernières années, plusieurs travaux ont été développés pour la résolution des équations tensorielles de Sylvester, en tant que généralisation des méthodes matricielles de résolution de

ces équations (Méthodes directes [8, 63], **méthodes du sous-espace de Krylov** pour la résolution des équations matricielles de Lyapunov/ Sylvester de grande dimension [47, 75]). De plus, des nouveaux travaux [13, 14, 109] qui se fonde sur le cadre tensoriel proposent de nouvelles méthodes basées sur le formalisme tensoriel pour la résolution des équations tensorielles de Sylvester. En suivant le même principe, nous visons la résolution de ces équations en introduisant de nouvelles approches tensorielles plus efficaces en termes de rapidité et de précision (chapitre 7).

* **Méthodes d'extrapolation tensorielles** : Des méthodes d'extrapolation scalaire, vectorielle ou matricielle ont été développées pour accélérer la convergence de certaines suites scalaires. En effet, l'extrapolation est une des techniques essentielles en analyse numérique, elle repose sur l'idée de construire une transformation $(T_n)_{n\in \mathrm{N}}$ qui permet d'accélérer la convergence d'une suite donnée $(S_n)_{n\in \mathrm{N}}$. En pratique, les séquences $(S_n)_{n\in \mathrm{N}}$ qui décrivent les phénomènes (physiques, sociaux, etc.) se comportent mal, soit le temps de convergence est très long, ce qui implique l'augmentation des opérations élémentaires, soit la propagation des erreurs et le temps d'exécution . L'une des premières méthodes d'extrapolation est le procédé Δ^2 d'Aitken [28]. Des généralisations ont ensuite été proposées au cours des dernières années. Parmi elles, la méthode MPE [33, 104], la méthode RRE [45], la MMPE [25, 115] et TEA [25]. Pour plus de détails, voir [28, 79, 84, 86, 125, 126]. Jbilou et al. [84] ont introduit les **methodes d'extrapolation matricielles** par considération cette fois d'une séquence matricielle. Ils définissent la version matricielle (Block-/Global) de MPE , RRE, MMPE et TEA. Inspirés par tous ces travaux, nous avons eu l'idée de considérer une suite tensorielle, ce qui donnera naissance aux **Méthodes d'extrapolation tensorielles**. (Chapitres 8 et 9).

1.3 Préliminaires

Un *tenseur* est un tableau multidimensionnel de données et une extension naturelle des scalaires, des vecteurs et des matrices à un ordre supérieur, un scalaire est un tenseur **d'ordre** 0, un vecteur est un tenseur **d'ordre** 1 et une matrice est un tenseur **d'ordre** 2 ; voir les Figures (1.10). **L' ordre** d'un tenseur est le nombre de ses indices, que l'on appelle *modes* ou *ways*. Pour un tenseur N-mode donné $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, La notation $x_{i_1,\ldots,i_N}$(avec $1 \leq i_j \leq I_j$, ; $j = 1, \ldots N$) représente l'élément $(i_1, \ldots, i_N)$ du tenseur $\mathcal{X}$. Correspondant à un tenseur donné $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N}$, la notation

$$\underbrace{\mathcal{X}_{:,:,\ldots,:,k}}_{(N-1)\text{- times}} \quad k = 1, 2, \ldots, I_N$$

désigne un tenseur dans $\mathbb{R}^{I_1 \times I_2 \times \ldots \times I_{N-1}}$ qui est obtenu en fixant le dernier indice et est appelé tranche frontale. Les fibres sont l'analogue d'ordre supérieur des lignes et des colonnes d'une matrice. Une fibre est définie en fixant tous les indices sauf un. Une colonne de matrice est une fibre mode-1 et une ligne de matrice est une fibre mode-2 ; voir les Figures (1.11, 1.12 et 1.13).

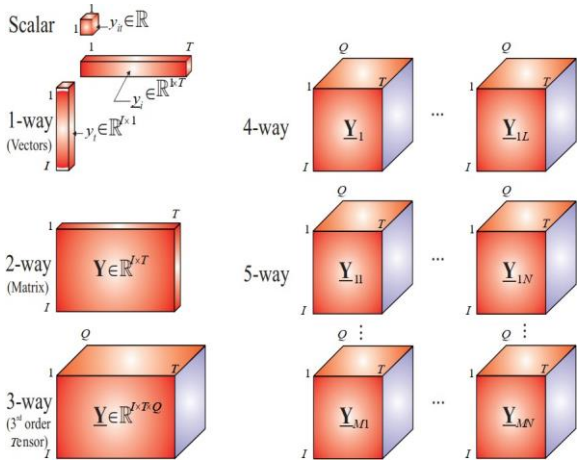


Figure 1.10: *Illustration des données à voies multiples : tenseur à voie nulle = scalaire, tenseur à voie unique = vecteur ligne ou colonne, tenseur à voie double = matrice, tenseur à N voie = tenseurs d'ordre supérieur. Les tenseurs à 4 et 5 voies sont représentés ici comme un ensemble de tenseurs à trois voies.*

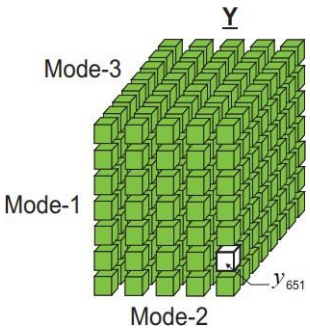


Figure 1.11: *Un tableau à trois voies (tenseur du troisième ordre) .*

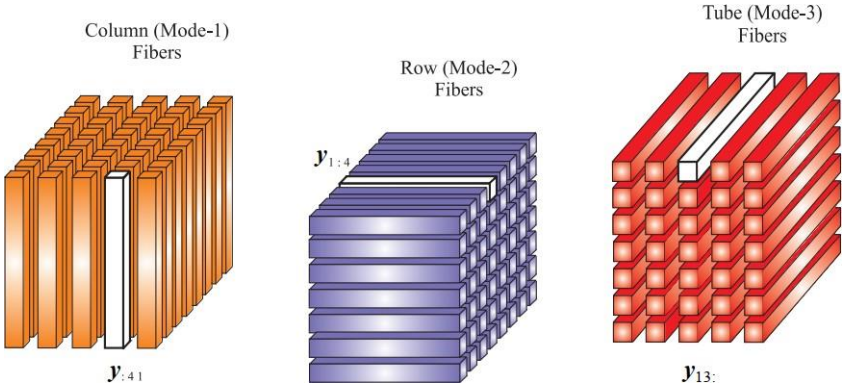


Figure 1.12: *Fibres : pour les tenseurs du troisième ordre.*

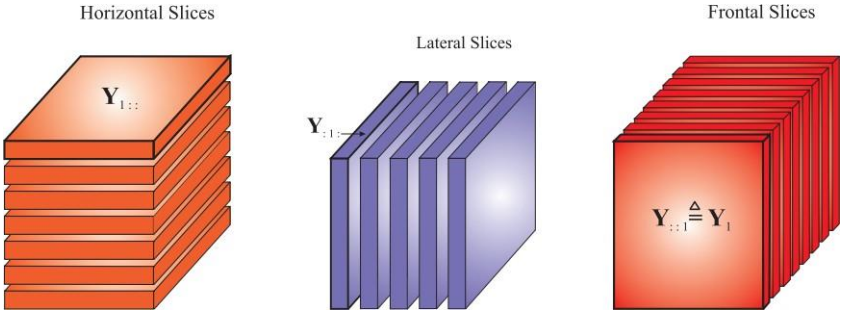


Figure 1.13: *Tranches pour un tenseur du troisième ordre.*

Dans cette thèse, nous nous basons sur plusieurs produits tensoriels. En particulier, nous avons utilisé les produits tensoriels suivants : **n-mode product** [91], **Einstein product** [46], **T-product** [89, 90] et **Cosine product** (c-produit) [88]. Dans ce qui suit, nous définissons des opérations algébriques importantes et énonçons des résultats fondamentaux liés à ces produits.

1.3.1 Produit à n modes (n-mode product)

La matrice à n modes d'un tenseur $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \ldots \times I_N}$ est désignée par $X_{(n)}$ et dispose les fibres à n modes comme étant les colonnes de la matrice résultante $X_{(n)} \in \mathbb{R}^{I_n \times (I_1 \ldots I_3 \ldots I_{n-1} I_{n+1} \ldots I_N)}$ avec $n = 1, \ldots, N$. nous avons:

$$X_{(n)}(i_n, j) = x_{i_1,\ldots,i_N}$$

où $j = 1 + \sum_{\substack{k=1 \\ n}}^{N} (i_k - 1) J_k$ avec $J_K = \prod_{\substack{m=1, \\ m=1}}^{k-1} I_m$. Une opération importante pour un tenseur est la *multiplication tenseur-matrice* [91], également connue sous le nom de produit à n modes d'un tenseur. $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ avec une matrice $U \in \mathbb{R}^{J \times I_n}$, Le produit à n modes est un tenseur de taille $I_1 \times I_2 \times \ldots I_{n-1} \times J \times I_{n+1} \times \ldots \times I_N$ dont les entrées sont données par :

$$(\mathcal{X} \times_n U)_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_N} u_{j i_n}$$

Le produit vectoriel à n modes d'un tenseur $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ avec un vecteur $v \in \mathbb{R}^{I_n}$ est dénoté par $(\mathcal{X} \tilde{\times}_n v)$ est un tenseur de taille $I_1 \times I_2 \times \ldots I_{n-1} \times I_{n+1} \times \ldots \times I_N$ dont les entrées sont données par:

$$(\mathcal{X} \tilde{\times}_n v)_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_N} v_{i_n}$$

.

Le produit externe (outer product) de deux vecteurs $u \in \mathbb{R}^I$ et $v \in \mathbb{R}^J$ est donné par

$$u \circ v = u v^T \in \mathbb{R}^{IJ} \tag{1.1}$$

et le produit externe de trois vecteurs $u \in \mathbb{R}^I$, $v \in \mathbb{R}^J$ and $w \in \mathbb{R}^Q$ donne un tenseur de rang un du troisième ordre :

$$\mathcal{Z} = u \circ v \circ w \in \mathbb{R}^{IJQ} \tag{1.2}$$

En général, le produit externe de deux tenseurs $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ et $\mathcal{Y} \in$

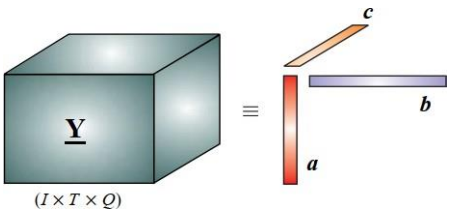


Figure 1.14: *Illustration d'un tenseur de troisième ordre de rang un*

$\mathbb{R}^{J_1 \times J_2 \times \ldots \times J_M}$ est donné par

$$\mathcal{Z} = \mathcal{X} \circ \mathcal{Y} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M} \tag{1.3}$$

où

$$\mathcal{Z}_{i_1,\ldots,i_N,j_1,\ldots,j_M} = \mathcal{X}_{i_1,\ldots,i_N}\, \mathcal{Y}_{j_1,\ldots,j_M} \tag{1.4}$$

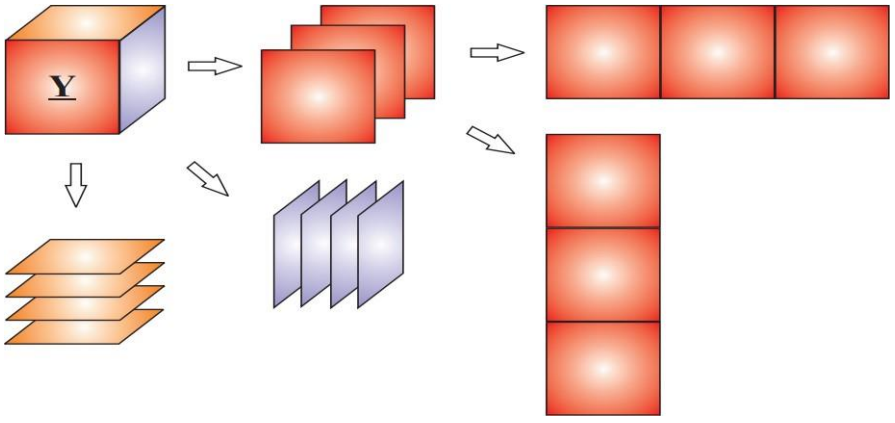


Figure 1.15: *Illustration du dépliage (aplatissement, matricialisation) en ligne et en colonne d'un tenseur d'ordre 3*

1.3.2 Décompositions tensorielles

Les décompositions et factorisations tensorielles ont été initiées par Hitchcock en 1927 [71] et plus tard développés par Cattelin en 1944 et par Tucker

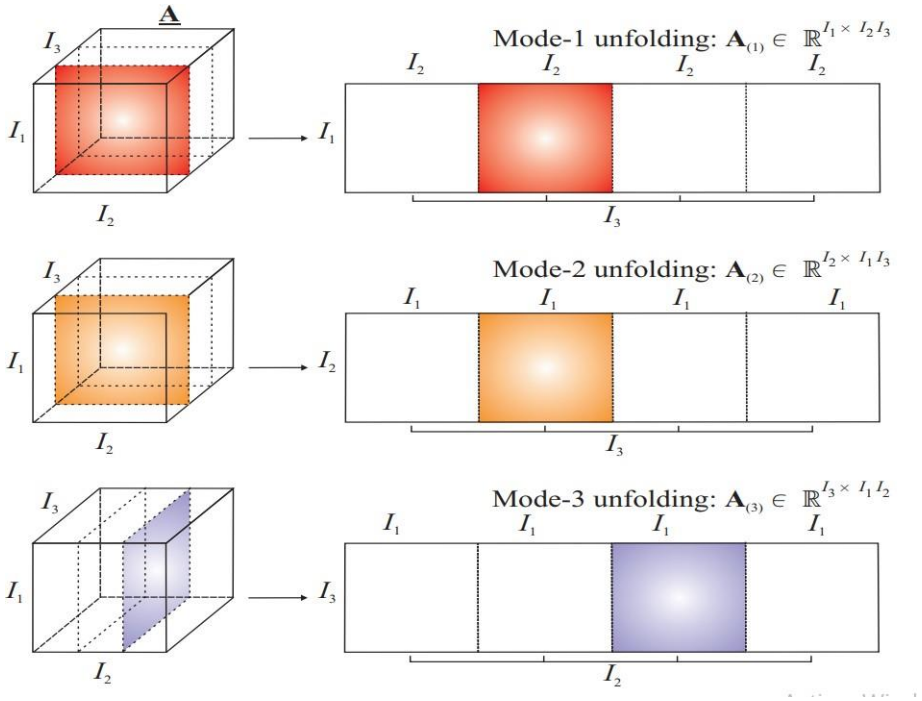


Figure 1.16: *Dépliage (matricialisation) d'un tenseur d'ordre 3. Le tenseur peut être déplié de trois façons pour obtenir des matrices comprenant ses vecteurs mode-1, mode-2 et mode-3.*

[132] en 1966. Dans ce qui suit, nous présentons quelques décompositions tensorielles classiques.

1.3.2.1 CANDECOMP decomposition

Definition 1.1. (CANDECOMP decomposition (CP)) Soit $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ un tensor d'ordre N . La CP de $\mathcal{X}$ est donnée par

$$\mathcal{X} = \sum_{r=1}^{R} a_r^1 \circ \ldots \circ a_r^N \tag{1.5}$$

avec R est un entier positif et a_r^k sont des vecteurs de taille I_k $1 \leq k \leq N$

La première question est de savoir comment choisir le nombre R. La plupart des procédures ajustent plusieurs décompositions CP avec différents nombres de composants jusqu'à ce que l'une d'entre elles soit bonne. Soit $\mathcal{X}$ un tensor d'ordre 3. L'objectif est de calculer CP avec R composantes qui se

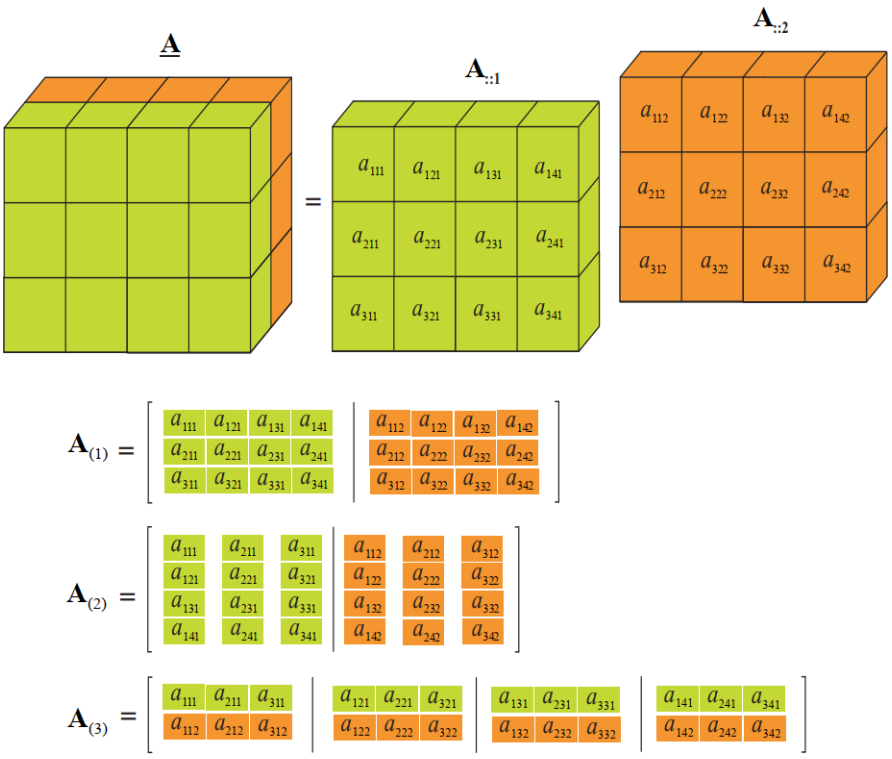


Figure 1.17: *Exemple de dépliage du tenseur du troisième ordre en mode-1, mode-2 et mode-3.*

rapproche le plus approximation de $\mathcal{X}$, i.e., Trouver:

$$\min_{\mathcal{Y}} ||\mathcal{X} - \mathcal{Y}|| \tag{1.6}$$

1.3.2.2 Notion de rang des tenseurs

Le rang d'une matrice est le nombre de vecteurs colonnes linéairement indépendants, ou, de manière équivalente, le nombre de valeurs singulières non nulles. Cependant, cette définition du rang d'une matrice n'est pas directement extensible aux tenseurs. Afin d'étendre la notion de rang aux tenseurs, il faut l'envisager sous un angle différent.

Definition 1.2. (tenseur de rang un) Soit $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ un tensor d'ordre N. $\mathcal{X}$ est appelé tenseur de rang 1 s'il peut être écrit comme le produit externe de N vecteurs

$$\mathcal{X} = a_1 \circ \ldots \circ a_N \tag{1.7}$$

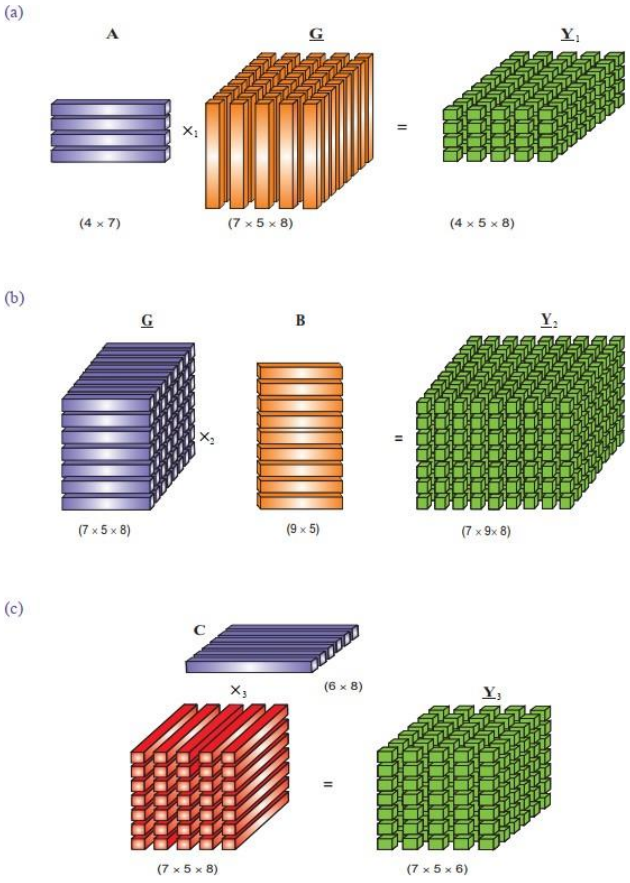


Figure 1.18: *llustration des multiplications en mode-n d'un tenseur du troisième ordre par des matrices. (a) multiplication mode- 1. (b) multiplication mode- 2. (c) multiplication mode- 3 multiplication.*

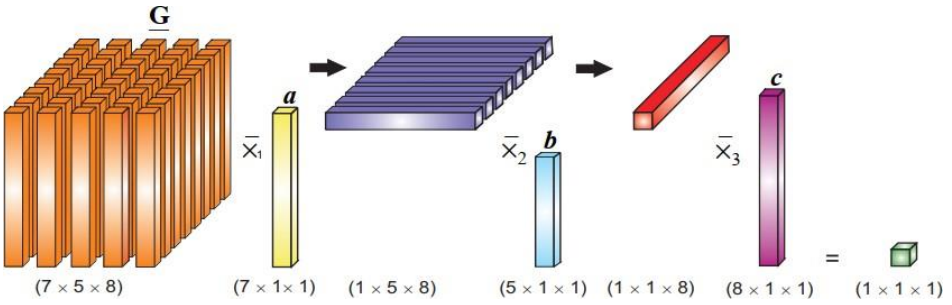


Figure 1.19: *Illustration of mode-n multiplication of a third-order tensor.*

avec R est un entier positif et a^k_rsont des vecteurs de taille I_k $1 \leq k \leq N$.

Le rang d'un tenseur est le nombre minimum de tenseurs de rang un qu'il faut additionner pour l'obtenir. La définition ci-dessus du rang des tenseurs est connue pour être NP difficile à calculer, ce qui la rend peu pratique pour

la plupart des applications. Nous définissons donc le n-rank et le rang multi-linéaire.

Definition 1.3. (Tenseur n-rank) Le n-rank est défini comme le nombre de fibres linéairement indépendantes à n modes d'un tenseur $\mathcal{X}$

$$rank_n(\mathcal{X}) = rank(\mathcal{X}_{(n)}) \tag{1.8}$$

Definition 1.4. (rang multi-linéaire) Le rang multi-linéaire d'un tenseur $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ est un vecteur, noté $rank_\wedge$ et est donné par

$$rank_\wedge = (rank_1(\mathcal{X}_{(1)}), \ldots, rank_N(\mathcal{X}_{(N)})) \tag{1.9}$$

avec $rank(\mathcal{X}_{(n)})$, $1 \leq n \leq N$ est le rang de la matrice de dépliage $\mathcal{X}_{(n)}$.

Dans ce qui suit, nous présentons la célèbre Tucker decomposition (TD), voir [91] pour plus de détails.

Definition 1.5. (TD) Soit $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$. LA TD de $\mathcal{X}$ est donnée par

$$\mathcal{X} = \mathcal{V} \times_1 U_1 \times_2 U_2 \ldots \times_N U_N \tag{1.10}$$

avec $\mathcal{V} \in \mathbb{R}^{R_1 \times R_2 \times \ldots \times R_N}$ est le tenseur central (tenseur noyau), $U_n \in \mathbb{R}^{I_n \times R_n}$, $1 \leq n \leq N$ sont des matrices de facteurs, et $(R_1, \ldots, R_N)$ est le rang multi-linéaire de la décomposition de Tucker.

1.4 Produit d'Einstein (Einstein product)

Definition 1.6. Soit $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_L \times K_1 \times \ldots \times K_N}$, $\mathcal{B} \in \mathbb{R}^{K_1 \times \ldots \times K_N \times J_1 \times \ldots \times J_M}$. Le produit d'Einstein des tenseurs $\mathcal{A}$ and $\mathcal{B}$ est le tenseur de taille $(I_1 \times \ldots \times I_L \times J_1 \times \ldots \times J_M)$ défini par :

$$(\mathcal{A} *_N \mathcal{B})_{i_1 \ldots i_L j_1 \ldots j_M} = \sum_{k_1=1}^{K_1} \sum_{k_2=1}^{K_2} \sum_{k_3=1}^{K_3} \ldots \sum_{k_N=1}^{K_N} a_{i_1 \ldots i_L k_1 \ldots k_N} \, b_{k_1 \ldots k_N \, j_1 \ldots j_M}$$

Definition 1.7. [77] Soit $\mathrm{A} = a_{i_1,\ldots,i_N,j_1,\ldots,j_M} \in \mathrm{R}^{I_1\times\ldots\times I_N\times J_1\times\ldots\times J_M}$, $\mathrm{B} = b_{i_1,\ldots,i_M,j_1,\ldots,j_N} \in \mathrm{R}^{J_1\times\ldots\times J_M\times I_1\times\ldots\times I_N}$, si $b_{i_1,\ldots,i_M,j_1,\ldots,j_N} = a_{j_1,\ldots,j_N,i_1,\ldots\ldots\ldots\ldots,i_M}$ alors B est appelé la transposée de A et désignée par A^T.

Un tenseur $\mathrm{D} = d_{i_1,\ldots,i_N,j_1,\ldots,j_N} \in \mathrm{C}^{I_1\times\ldots\times I_N\times I_1\times\ldots\times I_N}$ est diagonale si $d_{i_1,\ldots,i_N,j_1,\ldots,j_N} = 0$ au cas où les indices $i_1,\ldots,i_N$ sont différents de $j_1,\ldots,j_N$. Si toutes les entrées du tenseur sont nulles, on dit que ce tenseur est un tenseur nul, noté par O.

La trace d'un tenseur $\mathrm{A} \in \mathrm{C}^{I_1\times\ldots\times I_N\times I_1\times\ldots\times I_N}$ est donnée par $tr(\mathrm{A}) = \sum_{i_1\ldots i_N} a_{i_1\ldots i_N i_1\ldots\ldots\ldots\ldots i_N}$.

Le produit scalaire de deux tenseurs de même order $\mathrm{A},\mathrm{B} \in \mathrm{R}^{I_1\times\ldots\times I_N\times J_1\times\ldots\ldots\ldots\times J_m}$ est donné par

$$\langle \mathrm{A},\mathrm{B}\rangle = tr(\mathrm{B}^T *_N \mathrm{A}), \tag{1.11}$$

et la norme associée est définie par:

$$||\mathrm{A}||^2 = tr(\mathrm{A}^T *_N \mathrm{A}).$$

Definition 1.8. Un tenseur $\mathrm{X} \in \mathrm{R}^{I_1\times\ldots\times I_N\times I_1\times\ldots\times I_N}$ est appelé l'inverse du tenseur carré $\mathrm{A} \in \mathrm{R}^{I_1\times\ldots\times I_N\times I_1\times\ldots\times I_N}$ et désignée par A^{-1} s'il satisfait :

$$\mathrm{A} *_N \mathrm{X} = \mathrm{X} *_N \mathrm{A} = \mathrm{I}$$

avec I tenseur d'identité dont toutes les entrées sont nulles sauf les entrées diagonales. $\mathrm{I}_{i_1,\ldots,i_N,i_1,\ldots,i_N} = 1$.

Maintenant, nous rappelons quelques définitions et propriétés liées au produit d'Einstein.

Un tenseur déplié est une matrice obtenue en réorganisant systématiquement les entrées du tenseur dans un tableau bidimensionnel..

Definition 1.9. [27](Déplier un tenseur en une matrice) Soit la transformation Ψ_{IJ} de l'espace tensoriel $\mathrm{R}^{I_1\times\ldots\times I_N\times J_1\times\ldots\times J_M}$ dans l'espace matriciel $\mathrm{R}^{(I_1\ldots I_N)\times(J_1\ldots J_M)}$

avec $\Psi_{IJ}(\mathcal{X}) = X$ définie par composant comme

$$\mathcal{X}_{i_1 \ldots i_N j_1 \ldots j_M} = (X_{ivec(\mathbf{i},\mathrm{I})ivec(\mathbf{j},\mathrm{J})})$$

où $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$, $X \in \mathbb{R}^{(I_1 \ldots I_N) \times (J_1 \ldots J_M)}$, $ivec(\mathbf{i}, \mathrm{I})$ et $ivec(\mathbf{j}, \mathrm{J})$ sont respectivement deux fonctions d'indexation correspondant à l'ensemble d'indices $\mathbf{i} := \{i_1, \ldots, i_N\}$ et $\mathbf{j} := \{j_1, \ldots, j_M\}$, ie

$$ivec(\mathbf{i}, \mathrm{I}) = i_1 + \sum_{r=2}^{N} (i_r - 1) \prod_{u=1}^{r-1} I_u. \tag{1.12}$$

$$ivec(\mathbf{j}, \mathrm{J}) = j_1 + \sum_{s=2}^{M} (j_s - 1) \prod_{v=1}^{s-1} J_v. \tag{1.13}$$

and $\mathrm{I} := \{I_1, \ldots, I_N\}$, $\mathrm{J} := \{J_1, \ldots, J_M\}$ sont appelés le mode de ligne et le mode de colonne de $\mathcal{X}$, respectivement.

Definition 1.10. Soit $\mathcal{U}, \mathcal{R} \in \mathbb{C}^{I_1 \times \ldots \times I_N \times I_1 \times \ldots \times I_N}$ un tenseur non nul.

1. Le tenseur $\mathcal{U}$ est triangulaire supérieur si les entrées $u_{i_1,\ldots,i_N,j_1,\ldots,j_N} = 0$ lorsque $ivec(\mathbf{i}, \mathrm{I}) \geq ivec(\mathbf{i}, \mathrm{J})$.
2. Le tenseur $\mathcal{R}$ est triangulaire inférieur si les entrées $r_{i_1,\ldots,i_N,j_1,\ldots,j_N} = 0$ lorsque $ivec(\mathbf{i}, \mathrm{I}) \leq ivec(\mathbf{j}, \mathrm{J})$.

Definition 1.11. (Tenseur sous-colonne) Soit $\mathcal{A} = (a_{i_1,\ldots,i_N,j_1,\ldots,j_M}) \in \mathbb{C}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$.La j-ième sous-colonne de la $\mathcal{A}$ est le tenseur $\mathcal{A}_{(:,\ldots,:,j_1,\ldots,j_M)} = (a_{:,\ldots,:,j_1,\ldots,j_M}) \in \mathbb{C}^{I_1 \times \ldots \times I_N}$ où $j = j_M + \sum_{r=1}^{M-1} (j_r - 1) \prod_{L=r+1}^{M} J_L$.

Next, we define the row and column block tensors.

Definition 1.12. Soit $\mathcal{A} = (a_{i_1,\ldots,i_N,j_1,\ldots,j_M}) \in \mathbb{C}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$, $\mathcal{B} = (b_{i_1,\ldots,i_N,k_1,\ldots,k_M}) \in \mathbb{C}^{I_1 \times \ldots \times I_N \times K_1 \times \ldots \times K_M}$, $\mathcal{C} = (c_{j_1,\ldots,j_M,i_1,\ldots,i_N}) \in \mathbb{C}^{J_1 \times \ldots \times J_M \times I_1 \times \ldots \times I_N}$ et $\mathcal{D} = (d_{k_1,\ldots,k_M,i_1,\ldots,i_N}) \in \mathbb{C}^{K_1 \times \ldots \times K_M \times I_1 \times \ldots \times I_N}$. Alors

1. **Tenseur ligne par block** : Le tenseur ligne par block de $\mathcal{A}$ et $\mathcal{B}$ dénoté par $[\mathcal{A}\ \mathcal{B}]$ est le tenseur de taille $(\mathrm{I}_N \times \beta_1 \times \ldots \times \beta_M)$ où $\mathrm{I}_N = I_1 \times I_2 \times \ldots \times I_N$, $\beta_i = J_i + K_i, i = 1, \ldots, M$.

2. **Tenseur colonne par block** : Le tenseur colonne par block $\mathcal{C}, \mathcal{D}$ est défini comme suit

$$\mathcal{C} = \begin{bmatrix} \mathcal{C} \\ \mathcal{D} \end{bmatrix} = \begin{bmatrix} \mathcal{C}^T \mathcal{D}^T \end{bmatrix}^T \in \mathbb{C}^{\beta_1 \times \ldots \times \beta_M \times I_N}$$

3. Soit $\mathcal{T}_1 = [\mathcal{A}_1 \ \mathcal{B}_1]$, $\mathcal{T}_2 = [\mathcal{A}_2 \ \mathcal{B}_2]$ deux tenseurs ligne par block, avec $\mathcal{A}_1 \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$, $\mathcal{B}_1 \in \mathbb{R}^{I_1 \times \ldots \times I_N \times K_1 \times \ldots \times K_M}$, $\mathcal{A}_2 \in \mathbb{R}^{L_1 \times \ldots \times L_N \times J_1 \times \ldots \times J_M}$ et $\mathcal{B}_2 \in \mathbb{R}^{L_1 \times \ldots \times L_N \times K_1 \times \ldots \times K_M}$. Nous avons

$$\mathcal{T} = \begin{bmatrix} \mathcal{T}_1 \\ \mathcal{T}_2 \end{bmatrix} = \begin{bmatrix} \mathcal{A}_1 & \mathcal{B}_1 \\ \mathcal{A}_2 & \mathcal{B}_2 \end{bmatrix} \in \mathbb{C}^{\rho_1 \times \ldots \times \rho_N \times \beta_1 \times \ldots \times \beta_M}$$

avec $\rho_i = I_i + L_i, i = 1, \ldots, N$ et $\beta_j = J_j + K_j, j = 1, \ldots, M$.

Proposition 1.13. *Soit* $[\mathcal{A} \ \mathcal{B}], \begin{bmatrix} \mathcal{C} \\ \mathcal{D} \end{bmatrix}, \begin{bmatrix} \mathcal{A}_1 & \mathcal{B}_1 \\ \mathcal{A}_2 & \mathcal{B}_2 \end{bmatrix}$ *et* $\mathcal{F} \in \mathbb{C}^{I_N \times I_N}$. *selon la Définition 1.12 nous avons*

$$*_N [\mathcal{A} \ \mathcal{B}] = [\mathcal{F} *_N \mathcal{A} \ \mathcal{F} *_N \mathcal{B}] \in \mathbb{C}^{I_N \times \beta_1 \times \ldots \times \beta_M}$$

$$*_N \begin{bmatrix} \mathcal{C} \\ \mathcal{D} \end{bmatrix} = \begin{bmatrix} \mathcal{C} *_N \mathcal{F} \\ \mathcal{D} *_N \mathcal{F} \end{bmatrix} \in \mathbb{C}^{\beta_1 \times \ldots \times \beta_M \times I_N}$$

$$[\mathcal{A} \ \mathcal{B}] *_M \begin{bmatrix} \mathcal{C} \\ \mathcal{D} \end{bmatrix} = \mathcal{A} *_M \mathcal{C} + \mathcal{B} *_M \mathcal{D} \in \mathbb{C}^{I_N \times I_N}$$

$$\begin{bmatrix} \mathcal{A}_1 & \mathcal{B}_1 \\ \mathcal{A}_2 & \mathcal{B}_2 \end{bmatrix} *_M \begin{bmatrix} \mathcal{C} \\ \mathcal{D} \end{bmatrix} = \begin{bmatrix} \mathcal{A}_1 *_M \mathcal{C} + \mathcal{B}_1 *_M \mathcal{D} \\ \mathcal{A}_2 *_M \mathcal{C} + \mathcal{B}_2 *_M \mathcal{D} \end{bmatrix} \in \mathbb{C}^{\rho_1 \times \ldots \times \rho_M \times I_N}$$

1.5 T-product

1.5.1 Transformation de Fourier discrète

La Transformation de Fourier discrète joue un rôle très important dans la définition du T-product. On a

$$\tilde{v} = F_n v \in \mathbb{C}^n, \tag{1.14}$$

avec F_n est la matrice définie par

$$F_n = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ . & . & . & \dots & . \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)(n-1)} \end{bmatrix} \in \mathbb{C}^{n \times n}, \quad (1.15)$$

où $\omega = e^{\frac{-2\pi i}{n}}$ avec $i^2 = -1$. Il n'est pas difficile de montrer que (voir [64])

$$F_n^* = \overline{F}_n, \text{ and } F_n^* F_n = F_n F_n^* = nI_n. \quad (1.16)$$

Alors $F_n^{-1} = \frac{1}{n}\overline{F}_n$, ce qui montrent que $\frac{1}{\sqrt{n}}\overline{F}_n$ est une matrice unitaire.
Le coût du calcul du vecteur $\tilde{v}$ directement de (1.14) est $O(n^2)$. En utilisant Fast Fourier Transform (FFT) cela ne coûtera que $O(nlog(n))$ et cela rend FFT très rapide pour les problèmes de grandes dimension. Il est connu que

$$F_n \operatorname{circ}(v) F_n^{-1} = \operatorname{Diag}(\tilde{v}), \quad (1.17)$$

ce qui est équivalent à

$$F_n \operatorname{circ}(v) F_n^* = n \operatorname{Diag}(\tilde{v}), \quad (1.18)$$

avec

$$\operatorname{circ}(v) = \begin{bmatrix} v_1 & v_2 & \dots & v_n \\ v_2 & v_1 & \dots & v_{n-1} \\ . & & & . \\ . & : & \dots & . \\ v_n & v_{n-1} & \dots & v_1 \end{bmatrix},$$

et $\operatorname{Diag}(\tilde{v})$, est la matrice diagonale dont le i-ème élément diagonal est $\operatorname{Diag}(\tilde{v})_i$.

1.6 Cosine product (c-product)

1.6.1 Discrete Cosine Transformation

Nous rappelons quelques définitions et propriétés de la Discrete Cosine Transformation (DCT) et du c-produit. La DCT joue un rôle très important dans la définition du c-produit de tenseurs. On a

$$\tilde{v} = C_n v \in \mathrm{R}^n, \tag{1.19}$$

où C_n is the $n \times n$ DCT matrice avec des entrées

$$(C_n)_{ij} = \sqrt{\frac{2-\delta_{i1}}{n}} \cos\left[\frac{(i-1)(2j-1)\pi}{2n}\right] \qquad 1 < i, j < n$$

avec δ_{ij} est le delta de Kronecker. On sait que la matrice C_n est orthogonale, i.e, $C_n^T C_n = C_n C^T{}_n = I_n$; voir [111]. De plus, pour tout vecteur $v \in \mathrm{R}^n$, la multiplication $C_n v$ peut être calculée en $O(nlog(n))$ opérations. Également, Ng and al. [111] ont montré que les matrices qui peuvent être diagonalisées par C_n sont des matrices spéciales de Toeplitz-plus-Hankel. En d'autres termes, nous avons

$$C_n \,\mathrm{th}(v)\, C_n^{-1} = \mathrm{Diag}(\tilde{v}), \tag{1.20}$$

where

$$\mathrm{th}(v) = \underbrace{\begin{bmatrix} v_1 & v_2 & \dots & v_n \\ v_2 & v_1 & \dots & v_3 \\ \vdots & \vdots & \ddots & \vdots \\ v_n & v_{n-1} & \dots & v_1 \end{bmatrix}}_{\text{Toeplitz}} + \underbrace{\begin{bmatrix} v_2 & \dots & v_n & 0 \\ \vdots & & & v_n \\ v_n & 0 & \dots & \vdots \\ 0 & v_n & \dots & v_2 \end{bmatrix}}_{\text{Hankel}}$$

et $\mathrm{Diag}(\tilde{v})$ iest la matrice diagonale dont le i-ème élément diagonal est $(\tilde{v})_i$.

1.7 Publications

[52] A. El Ichi , K. Jbilou and R. Sadaka, Tensor Global Extrapolation Methods Using the n-Mode and the Einstein Products, Mathematics, 8(8) (2020), 1298.

[15] F. P. A. Beik, A. El Ichi , K. Jbilou and R. Sadaka, Tensor extrapolation methods with applications, Numerical Algorithms, Numerical Algorithms 87(2021) 1421–1444.

[49] M. El Guide, A. El Ichi, and K. Jbilou, Discrete cosine transform LSQR methods for multidimensional ill-posed problems, Journal of Mathematical Modeling, (2021), 1-17.

[50] M. El Guide, A. El Ichi, K. Jbilou and R. Sadaka, On tensor GMRES and Golub-Kahan methods via the T-product for color image processing, The Electronic Journal of Linear Algebra 37(2021), 524-543.

[51] A. El Ichi , K. Jbilou and R. Sadaka, On tensor tubal-Krylov subspace methods, Linear and Multilinear Algebra. (Accepted).

[16] F. P. A. Beik, M. El Guide, A. El Ichi, and K. Jbilou,Tensor GMRES and Golub-Kahan Bidiagonalization methods via the Einstein product with applications to image and video processing. Numerical linear algebra with applications (revised).

[53] A. El Ichi, Tensor Krylov subspace methods via the T-product for large Sylvester tensor equations. (submitted).

[44] F. Dufrenois, A. El Ichi, M. K. Jbilou and M. Hached, Cosine multidimensional linear discrete analysis methods for face recognition in large data. (In preparation).

1.8 Aperçu du manuscrit

La thèse peut être divisée en :

- **Chapitre** 3: Nous introduirons quelques nouvelles méthodes tubales de sous-espace Krylov pour résoudre des équations linéaires tensorielles. En utilisant le T-product, nous définirons en particulier la GMRES globale tubale tensorielle qui pourrait être considérée comme une généralisation de la GMRES globale. Nous donnons également une nouvelle version tubale de l'algorithme tensoriel de Golub-Kahan. À cette fin, nous présentons d'abord quelques nouveaux produits tensoriels et de nouvelles propriétés algébriques connexes. Les tests numériques présentés comparent les deux méthodes et montrent l'efficacité des procédures proposées.
- **Chapitre** 4: Dans ce chapitre, nous nous intéressons à certaines méthodes de sous-espace de Krylov tensoriel pour résoudre des équations linéaires tensorielles. Nous utilisons le produit d'Einstein bien connu pour deux tenseurs pour définir les algorithmes d'Arnoldi global tensoriel et de bidiagonalisation Golub Kahan tensoriel. En particulier, nous nous intéresserons aux problèmes linéaires mal posés et utiliserons certaines techniques de régularisation telles que la méthode de Tikhonov, pour obtenir une solution approximative acceptable. Nous commenterons le choix du paramètre apparaissant dans ces techniques de régularisation. Des applications au traitement vidéo sont données.
- **Chapitre** 5: Ce chapitre est consacré au développement de méthodes itératives tensorielles de sous-espace de Krylov pour résoudre les équations tensorielles multilinéaires de grande taille. Nous utilisons le T-product de deux tenseurs pour définir les algorithmes Tubal Global Arnoldi (TGA) et tenseur bidiagonalisation globale Golub-Kahan . En outre, nous illustrons comment les approches globales basées sur les tenseurs peuvent être exploitées pour résoudre des problèmes mal posés découlant de récupération d'images et de vidéos floues et multicanaux (couleur) flous, en utilisant la technique de régularisation dite de Tikhonov, pour

fournir des solutions régularisées approximatives calculables. Nous examinons également un critère de type validation croisée généralisée et principe de divergence pour la sélection du paramètre de régularisation dans la technique de Tikhonov. Des applications au traitement de séquences d'images sont données pour démontrer l'efficacité des algorithmes.

- **Chapitre** 6: Dans ce chapitre, nous proposons une nouvelle méthode de sous-espace de Krylov tensoriel pour des problèmes tensoriels linéaires mal posés comme dans la restauration d'images ou vidéos. Ces méthodes sont basées sur le DCT qui donne des calculs rapides de produit tenseur-tenseur. En particulier, nous nous concentrerons sur les versions tensorielles de Generalized minimal residual method (GMRES), la bidiagonalisation de Golub-Kahan et Least-squares method (LSQR). Les tests numériques présentés montrent que les méthodes sont très rapides et donnent de bonnes précisions lors de la résolution de certains problèmes linéaires tensoriels.
- **Chapitre** 7: Introduire de nouvelles méthodes de sous-espaces tensoriels de Krylov pour résoudre les équations tensorielles de Sylvester. La méthode proposée utilise le T-product pour les tenseurs et les sous-espaces tensoriels. Nous présentons quelques nouveaux produits tensoriels et les propriétés algébriques associées. Ces nouveaux produits nous permettront de développer les tensor Full orthogonalization method (tFOM), tensor GMRES (tGMRES), Tubal Block Arnoldi (TBA) et les Tubal Block Arnoldi method to sole large Sylvester tensor equation (TBAS). Nous donnons quelques propriétés liées à ces méthodes et proposons quelques expériences numériques.
- **Chapitre** 8: Nous développons principalement les méthodes d'extrapolation de vecteurs et de matrices polynomiales dans un cadre tensoriel. À cette fin, certains nouveaux produits entre tenseurs sont définis et le concept

de définition positive est étendu pour les tenseurs correspondant au T-product [89, 90].

En outre, nous discutons de la résolution du problème des moindres carrés associé à une équation tensorielle en utilisant Tensor Singular Value Decomposition (TSVD) . Nous décrivons comment une technique d'extrapolation peut également être mise en œuvre sur la séquence tensorielle produite par le Truncated TSVD (TTSVD) pour résoudre des équations tensorielles éventuellement mal posées. Nous développons essentiellement quelques méthodes d'extrapolation tensorielle, à savoir Tensor Reduced Rank Extrapolation (TRRE), Tensor Minimal Polynomial Extrapolation (TMPE), Tensor Modified Minimal Polynomial Extrapolation (TMMPE) et Tensor Topological *e*-algorithm (TTEA). Nous donnons quelques propriétés et montrons comment ces nouvelles méthodes d'extrapolation tensorielle peuvent être appliquées aux séquences obtenues par troncature du tenseur TSVD.

- **Chapitre** 9 : Nous présentons de nouvelles méthodes d'extrapolation tensorielle considérées comme des généralisations de méthodes d'extrapolation vectorielle, matricielle et par blocs telles que les méthodes d'extrapolation polynomiale ou les algorithmes de type *epsilon* correspondant au produit à n modes [91]. Nous définirons de nouveaux produits tensoriels qui seront utilisés pour introduire des méthodes d'extrapolation tensorielle globale. Nous discutons de l'application de ces méthodes à la résolution de systèmes d'équations linéaires et non linéaires et proposons une implémentation efficace de ces méthodes par la décomposition QR globale.

Une conclusion générale a été donnée avec quelques perspectives, suggestions et idées pour des travaux futurs.

Chapter 2

Introduction

2.1 Short overview

At the beginning of the nineteenth century, mathematicians and physicists found it very difficult to express many natural laws. Hence the need to create a new mathematical tool that would allow the introduction of a robust, simple and reliable language capable of describing and studying the various natural phenomena. This gave birth to a new "mathematical being" called "Tensors", which will be used thereafter, not only in the works exploited by physicists like Einstein who had expressed the famous fundamental equations of the theory of general relativity in function of tensors, but also in many scientific areas, psychometry and data analysis [70, 131, 132, 133], chemometrics [29, 96], signal processing [36, 95, 112, 129], biomedical engineering [103, 116, 117], data mining [91, 106, 113] and computer vision [97, 98, 123].

When it comes to high dimensional data, the tensor approach is more efficient than the traditional matrix methods. The estimation of emission in chemometrics based on tensor decompositions [29, 30], compression de données [40], network processing [5, 122] and system modeling [31, 54] are all famous examples that illustrate the advantage of the tensor approach. Indeed, the multidimensional (tensorial) representation of real data allows:

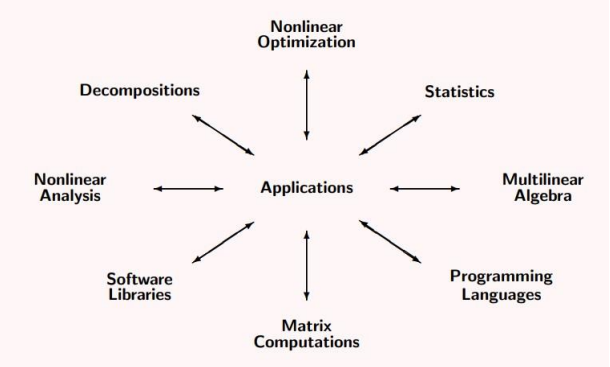


Figure 2.1: *The "Geometry" of the Tensor Research Community*

* Better visualization of the data, not only does it reserve the initial information of the mathematical model, but it allows the conservation of these characteristics.
* Good treatment of the data.
* Better exploitation of the various techniques of data storage.
* More efficiency of the calculations (good estimation, better precision ...).

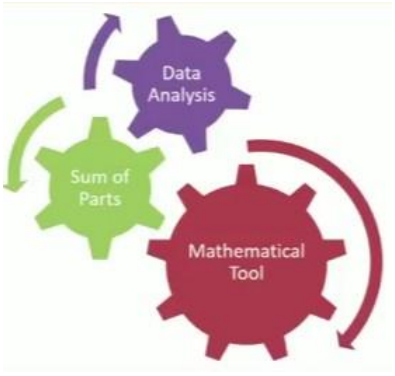


Figure 2.2: *Tensor: mathematical tool for solving big data problems*

In general, in the majority of problems, the matrix representation is insufficient for data representation, it creates large, unstable mathematical models, and provides very complex, long, less accurate algorithms, hence the need to use tensors. In image representations of the acquired data, for example, the representation of color images with a matrix is not practical (which would be of large dimensions), it is more natural to represent it in

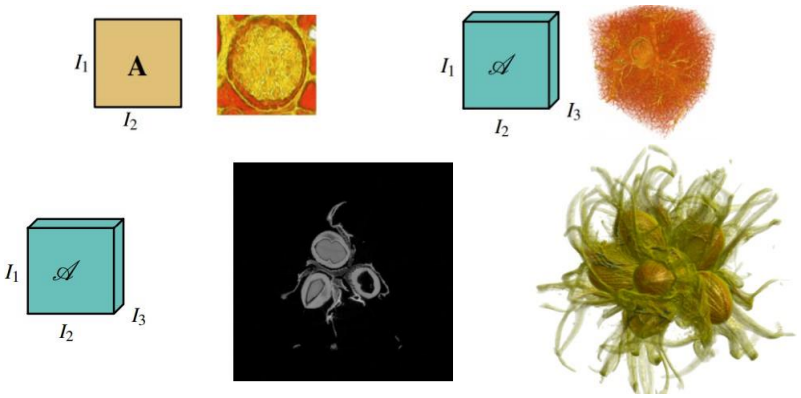


Figure 2.3: *Data visualization of 2nd and 3rd order tensor respectively*

the form of a multidimensional matrix (tensors of order 3). Same thing for the videos, which can be seen as a tensor of order higher than 3 [49, 50].

.

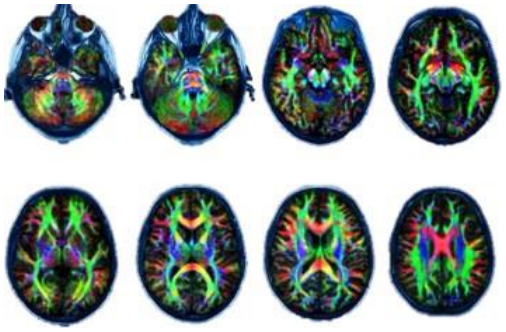


Figure 2.4: ***Medical iMaging****: naturally involves 3D (spatio) and 4D (spatio-temporal) correlations*

2.2 Thesis challenge

Certainly, in recent years, several works have been attributed to the application of tensorial methods to applied sciences. For example, solving differential equations [13, 109], tensor decomposition [89, 90, 91]. But it is still difficult to consider them as an indispensable means in the study of certain scientific problems. Since the theoretical basis associated with this kind of

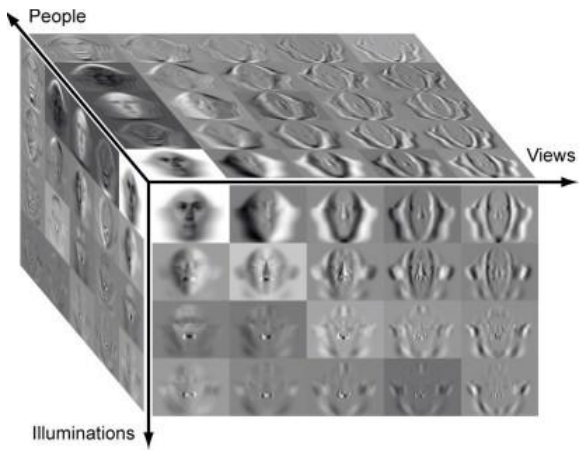


Figure 2.5: ***Machine vision****: understanding the world in 3D*

reasoning is still weak, it lacks a whole basic theory (product, inverse . . .). Hence the interest of our research work, which targets the development of numerical matrix methods for the solution of large scale problems, as well as the application of these methods to problems that occur in science and industry. In general, the main lines of research treated are :

* **Tensor Krylov subspace methods**: The objective of Krylov processes is to construct an orthonormal basis of a Krylov subspace or two bi-orthonormal bases of two Krylov subspace or two bi-orthonormal bases of two Krylov subspaces. Krylov subspaces. The three types of Krylov processes are the standard Krylov processes, global [79, 84]and block Krylov processes which apply respectively to standard, matrix and block subspaces of Krylov standard, matrix and block Krylov subspaces.

  Recently, Biek et al. [13, 14, 16, 109] have succeeded in developing the tensor version of Krylov subspace methods "**Tensor Krylov subspace methods** ". Also, Huang and al. [77] are proposed another tensor version of this method.

  In this dissertation, we introduce another version of tensor Krylov subspace methods; see Chapter 3.

* **Tensor methods for color image and video processing:**: Image restoration is a very important step in the image processing process. This last

one is necessary to correct the various treatments that can undergo an image, because it, during its capture can be, in most cases, subject to degradations that lead to the lack of useful information that is contained and which is of great importance. The most used representation of a digital image is that of a two-dimensional table composed of a set of rows and columns. But, this representation is not practical any more. For example, the Figure 2.6 illustrates that a grayscal image can be stored represented as a 2D array, color RGB image (Red, Green, Blue) as a 3D array, composing by three grayscale images for each color, Hyperspectral Images (HSI) can be illustrated as a 3D array composing by hundreds of grayscale images spanning a continuous range of frequencies see; Figure 2.6.

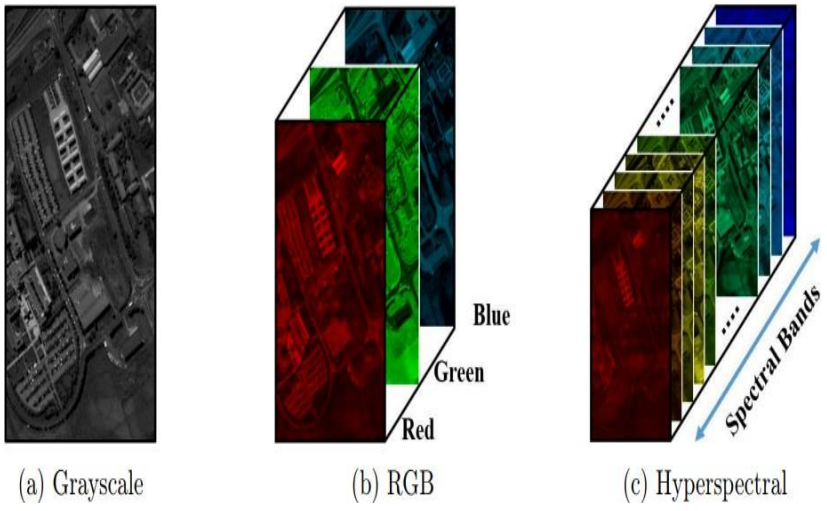


Figure 2.6: *Examples of image types using grayscale samples*

Another example illustrated by Figure 2.7 shows that reshaping a 3D image into matrix digital representation is insufficient, its leads to a loss of information. In addition, the size of the matrices becomes increasingly large, which will pose a problem later during the image processing phase. Same thing for the multi spectral images, the digital representation in 2D of these images destroys the natural ordering of the data; see Figure 2.8. Moreover, as shown in the Figure 2.9 a video can be

represented as a third order tensor, since a video is just an RGB image that moves in time.

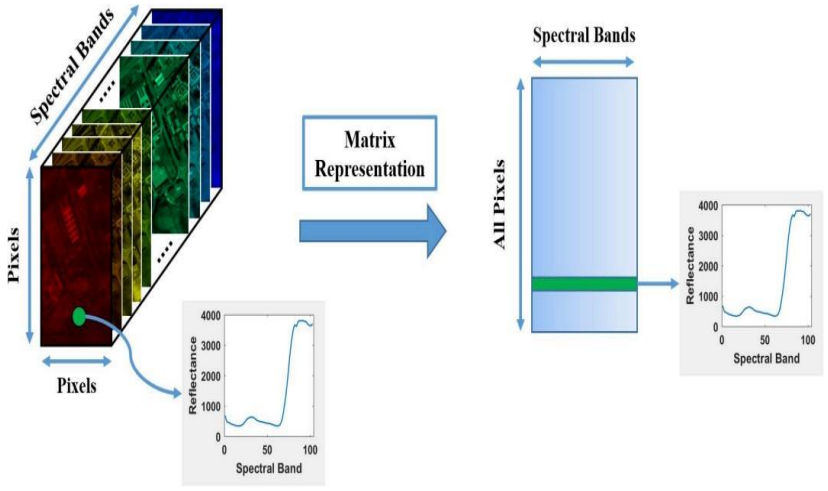


Figure 2.7: *Example of reshaping a RGB image types into 2D matrix*

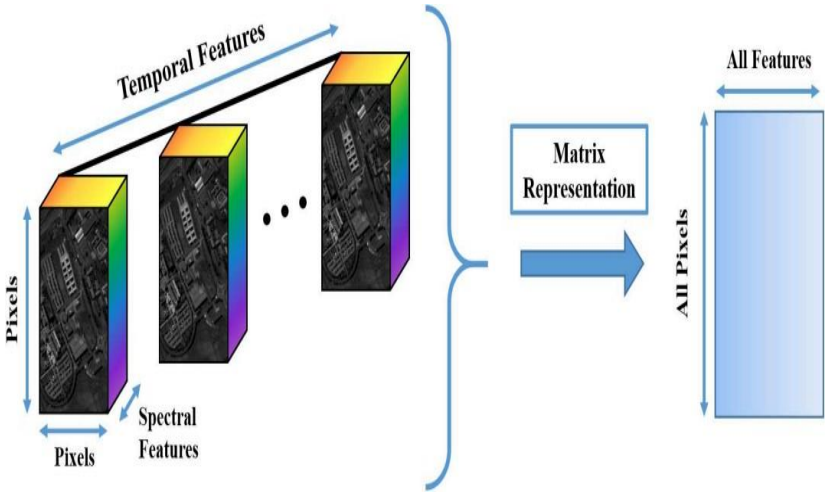


Figure 2.8: *Example of reshaping a multi spectral image types into 2D matrix*

Several works deal with different problems related to imaging. For example, El Guide and al. [17, 18] propose the **matrix Krylov subspace methods** (GMRES, Golub Kahan bidiagonalization) for the resolution / restoration of images. Furthermore, Biek and al. [13, 14] have introduced the tensor version of **Krylov subspace methods** for the image restoration.

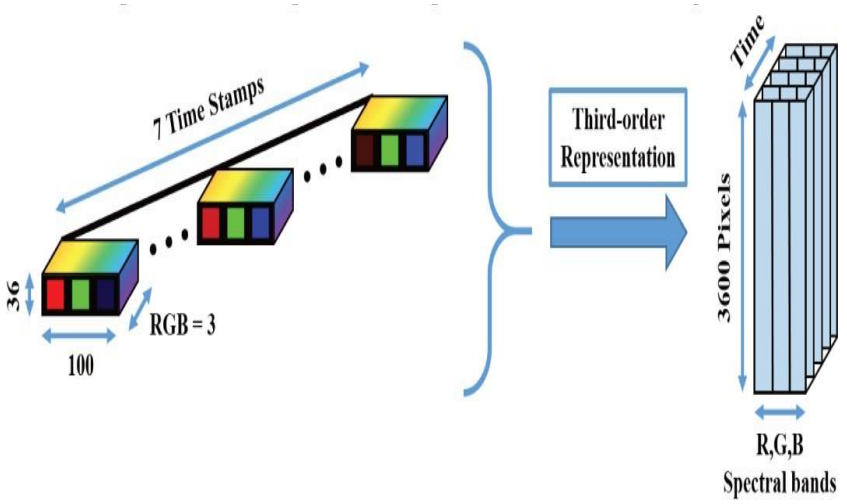


Figure 2.9: *Example of representation of a video in third order tensor*

By following the same idea, we propose in chapters 4,5 and 6 serval approaches based on tensor formalism for the resolution of color images and video problems.

* **Numerical methods for solving Lyapunov and Sylvester tensor equations**: In recent years, several works have been developed for the solution of tensor Sylvester equations, as a generalization of the matrix methods for solving Sylvester equations (Direct methods [8, 63], **Krylov subspace methods** for solving large-scale Lyapunov/ Sylvester matrix equations [47, 75]. These works [13, 14, 109] based on tensor framework propose new methods based on the tensor formalism for the solution of Sylvester tensor equations. Based on the same principle, we target the resolution of these equations by introducing new tensorial approaches that are more efficient in terms of speed and accuracy (Chapter 7).

* **Tensor Extrapolation methods** : Scalar, vector or matrix extrapolation methods have been developed to accelerate the convergence of some scalar sequences. Indeed, extrapolation is one of the essential techniques in numerical analysis, it is based on the idea of building a transformation $(T_n)_{n\in \mathbb{N}}$ which allows to accelerate the convergence of a given sequence

$(S_n)_{n\in \mathbb{N}}$. In practice, the sequences $(S_n)_{n\in \mathbb{N}}$ which describe the phenomena (physical, economic, chemical, etc.) behave badly, either the time of convergence is very long, which involves the increase of the elementary operations, the propagation of the errors and the time of execution . One of the earlier extrapolation method is the well known Aitken's Δ^2 process [28]. Generalisations have been then proposed the last decades. Among them the MPE [33, 104], the RRE method [45], the MMPE [25, 115]. For more details, see [28, 79, 84, 86, 125, 126]. A second class of vector sequence transformations contains the TEA [25]. Jbilou and al.[84] has been introduced the well know **matrix extrapolation methods** by considering this time some matrix sequence. They define the matrix version (Block/Global) of MPE , RRE, MMPE and TEA.
Inspired by all these works, we had the idea to consider a tensorial sequence, which will give birth to **Tensor Extrapolation methods**. (Chapters 8 and 9).

2.3 Preliminaries

A *tensor* is a multidimensional array of data and a natural extension of scalars, vectors and matrices to a higher order, a scalar is a 0^{th} **order** tensor, a vector is a 1^{th} **order** tensor and a matrix is 2^{th} **order** tensor; see Figures (2.10. The tensor **order** is the number of its indices, which is called *modes* or *ways*. For a given N-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times \ldots \times I_N}$, the notation $x_{i_1,\ldots,i_N}$ (with $1 \leq i_j \leq I_j, j = 1,\ldots N$) stands for element $(i_1,\ldots,i_N)$ of the tensor $\mathcal{X}$. Corresponding to a given tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\ldots \times I_N}$, the notation

$$\mathcal{X}\underbrace{:,:,\ldots,:}_{(N-1)\text{- times}},k \quad k = 1,2,\ldots,I_N$$

denotes a tensor in $\mathbb{R}^{I_1\times I_2\times \ldots \times I_{N-1}}$ which is obtained by fixing the last indexand is called frontal slice. Fibers are the higher-order analogue of matrix

rows and columns. A fiber is defined by fixing every index but one. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber; see Figures (2.11, 2.12 and 2.13).

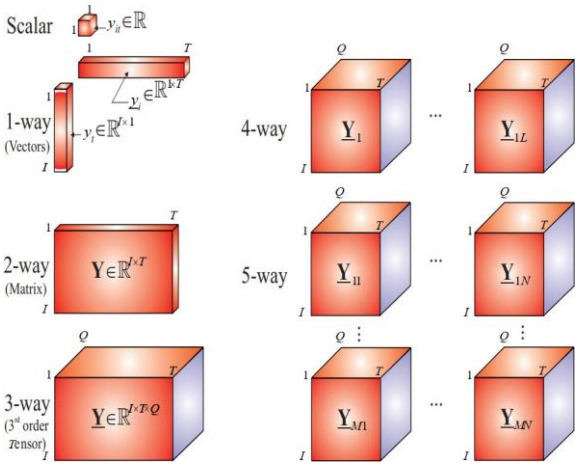


Figure 2.10: *Illustration of multi-way data: zero-way tensor = scalar, 1-way tensor = row or column vector, 2-way tensor = matrix, N-way tensor = higher-order tensors. The 4-way and 5-way tensors are represented here as a set of the three-way tensors.*

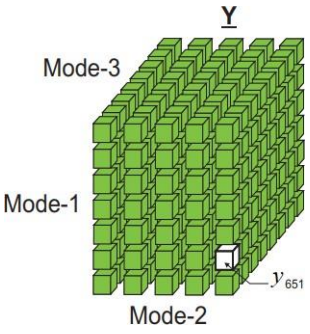


Figure 2.11: *A three-way array (third-order tensor).*

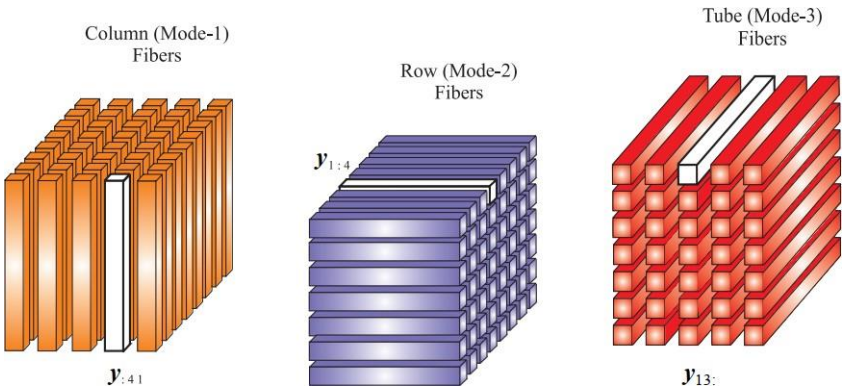


Figure 2.12: *Fibers: for a third-order tensors.*

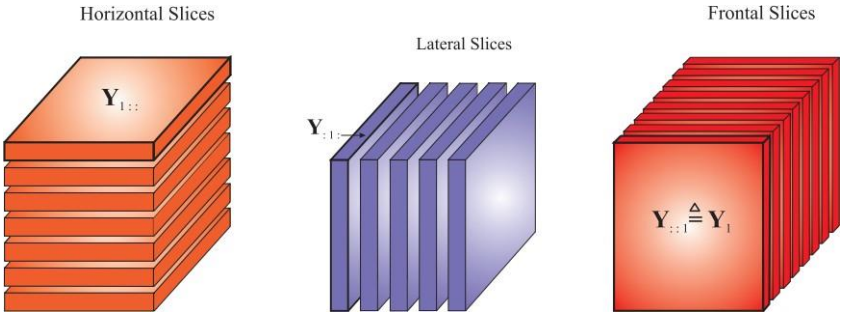


Figure 2.13: *Slices for a third-order tensor.*

In this thesis, we are based on several tensor products. In particular, we have used the **n-mode product**[91], **Einstein product**[46], **T-product** [89, 90] and **cosine product** (c-product) [88]. In the following, we define important algebraic operations and state fundamental results related to these products..

2.3.1 n-mode product

The n-mode matrix of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \ldots \times I_N}$ is denoted by $X_{(n)}$ and arranges the mode-n fibers to be the columns of the resulting matrix $X_{(n)} \in \mathbb{R}^{I_n \times (I_1 \ldots I_3 \ldots I_{n-1} I_{n+1} \ldots I_N)}$ with $n = 1, \ldots, N$. We have:

$$X_{(n)}(i_n, j) = x_{i_1, \ldots, i_N}$$

where $j = 1 + \sum_{k=1k/=}^{N} (i_k - 1)J_k$ with $J_K = \prod_{m=1,m/=}^{k-1} I_m$. An important operation for a tensor is the *tensor-matrix multiplication* [91], also known as n-mode product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ with a matrix $U \in \mathbb{R}^{J \times I_n}$, the n-mode product is a tensor of size $I_1 \times I_2 \times \ldots I_{n-1} \times J \times I_{n+1} \times \ldots \times I_N$ whose entries are given by:

$$(\mathcal{X} \times_n U)_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_N} u_{j i_n}$$

The n-mode vector product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ with a vector $v \in \mathbb{R}^{I_n}$ is denoted by $(\mathcal{X} \times_n v)$ is a tensor of size $I_1 \times I_2 \times \ldots I_{n-1} \times I_{n+1} \times \ldots \times I_N$ whose entries are given by:

$$(\mathcal{X} \times_n v)_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_N} v_{i_n}$$

.

The outer product of two vectors $u \in \mathbb{R}^I$ and $v \in \mathbb{R}^J$ is given by

$$u \circ v = uv^T \in \mathbb{R}^{IJ} \tag{2.1}$$

and the outer product of three vectors $u \in \mathbb{R}^I$, $v \in \mathbb{R}^J$ and $w \in \mathbb{R}^Q$ yields a third-order rank-one tensor:

$$\mathcal{Z} = u \circ v \circ w \in \mathbb{R}^{IJQ} \tag{2.2}$$

In general, the outer product of two tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ and $\mathcal{Y} \in$

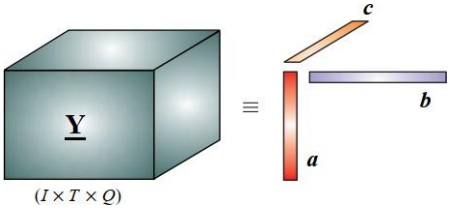


Figure 2.14: *Illustration of rank-one third-order tensor*

$\mathrm{R}^{J_1 \times J_2 \times \ldots \times J_M}$ is given by

$$\mathcal{Z} = \mathcal{X} \circ \mathcal{Y} \in \mathrm{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M} \tag{2.3}$$

where

$$\mathcal{Z}_{i_1,\ldots,i_N,j_1,\ldots,j_M} = \mathcal{X}_{i_1,\ldots,i_N} \mathcal{Y}_{j_1,\ldots,j_M} \tag{2.4}$$

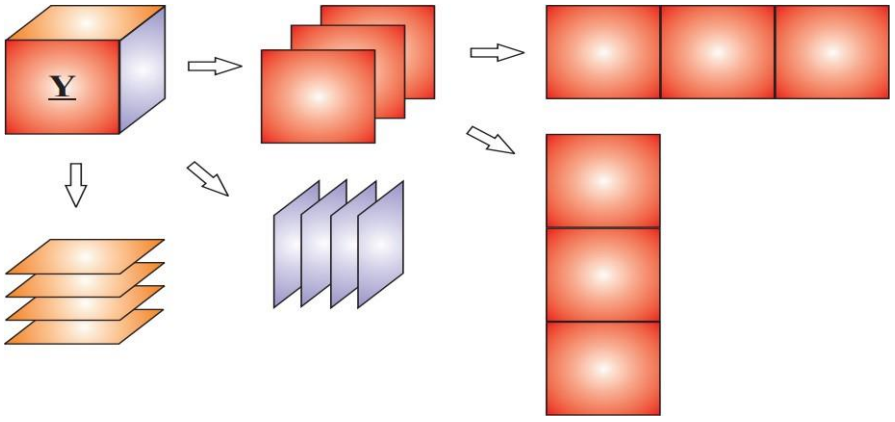


Figure 2.15: *Illustration of row-wise and column-wise unfolding (flattening, matricizing) of a thirdorder tensor*

2.3.2 Tensor decompositions

Tensor decompositions and factorizations were initiated by Hitchcock in 1927 [71] and later developed by Cattelin in 1944 and by Tucker [132] in 1966. In the following, we present some classical tensor decompositions.

2.3.2.1 CANDECOMP decomposition

Definition 2.1. (CP) Let $\mathcal{X} \in \mathrm{R}^{I_1 \times I_2 \times \ldots \times I_N}$ be an N-th order tensor. The CP of $\mathcal{X}$ is given by

$$\mathcal{X} = \sum_{r=1}^{R} a_r^1 \circ \ldots \circ a_r^N \tag{2.5}$$

where R is a positive integer and a_r^k are vectors of size I_k $1 \leq k \leq N$

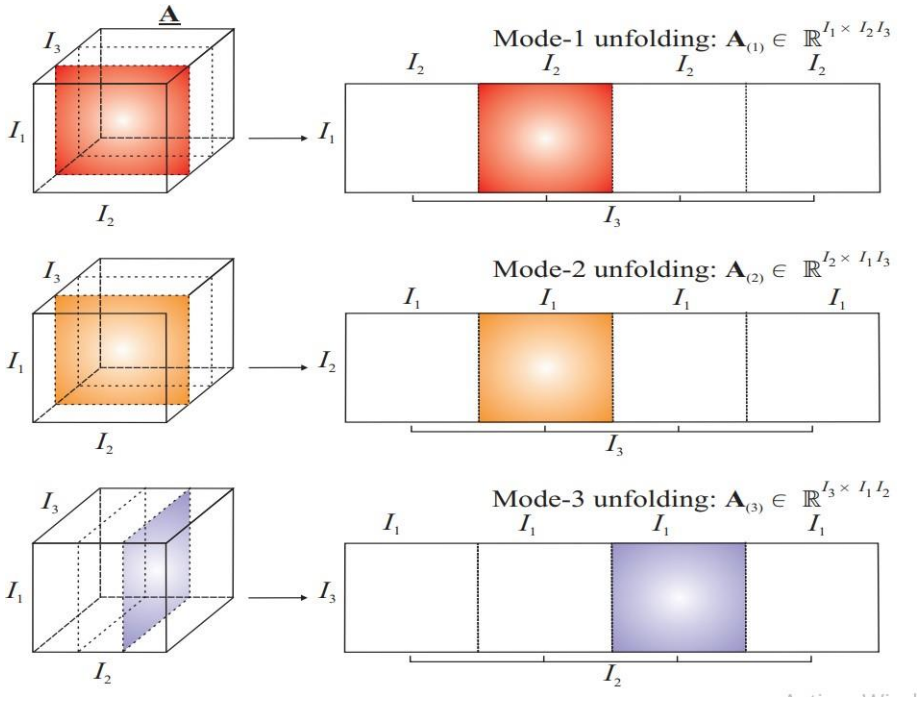


Figure 2.16: *Unfolding (matricizing) of a third-order tensor. The tensor can be unfolded in three ways to obtain matrices comprising its mode-1, mode-2 and mode-3 vectors.*

The first issue is how to choose the number R of components. Most procedures fit multiple CP decompositions with different numbers of components until one is good. Let $\mathcal{X}$ be a third-order tensor. The goal is to compute a CP with R components that best approximates $\mathcal{X}$, i.e., to find:

$$\min_{\mathcal{Y}} ||\mathcal{X} - \mathcal{Y}|| \tag{2.6}$$

2.3.2.2 Notion of rank for tensors

The rank of a matrix is the number of linearly independent column vectors, or, equivalently, the number of non-zero singular values. However, this definition of the matrix rank is not directly extensible to tensors. In order to extend the notion of rank to tensors, it has to be considered from a different perspective.

Definition 2.2. (rank one tensor) Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ be an N-th order

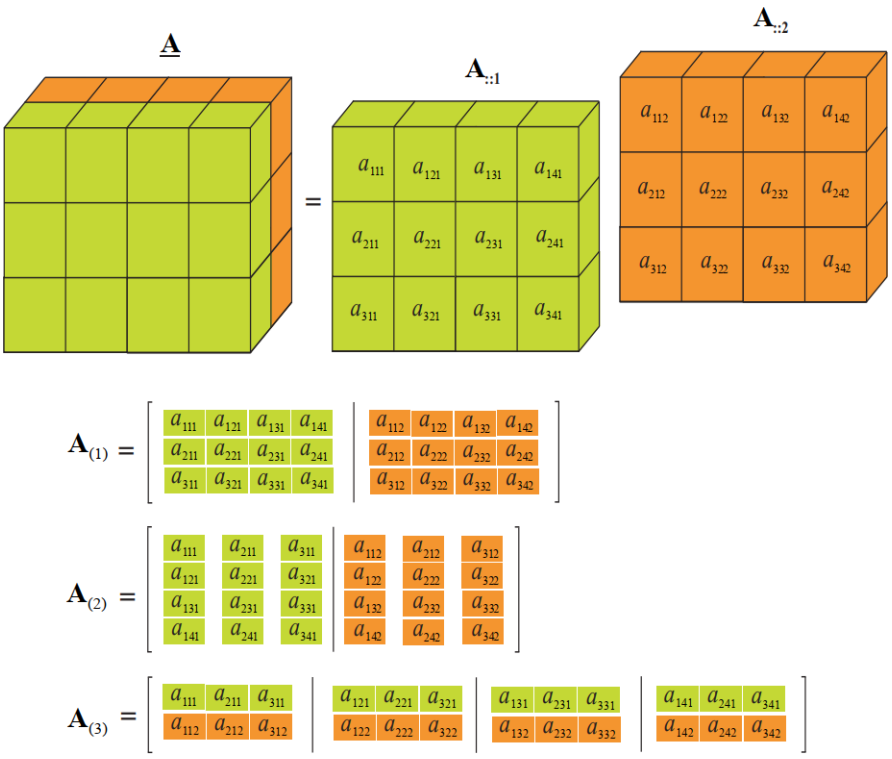


Figure 2.17: *Example of unfolding the third-order tensor in mode-1, mode-2 and mode-3.*

tensor. $\mathbb{X}$ is called rank 1 tensor if it can be written as the outer product of N vectors

$$\mathbb{X} = a_1 \circ \ldots \circ a_N \tag{2.7}$$

where R is a positive integer and a^k are vectors of size I_k $1 \leq k \leq N$

The rank of a tensor is the minimum number of rank one tensors we need to add up to get it. The above definition of tensor rank is known to be NP-hard to compute which makes it impractical for most applications. Thus, we define the n-rank and the multi-linear rank.

Definition 2.3. (Tensor n-rank) The n-rank is defined as the number of linearly independent n-mode fibres of a tensor $\mathbb{X}$

$$rank_n(\mathbb{X}) = rank(\mathbb{X}_{(n)}) \tag{2.8}$$

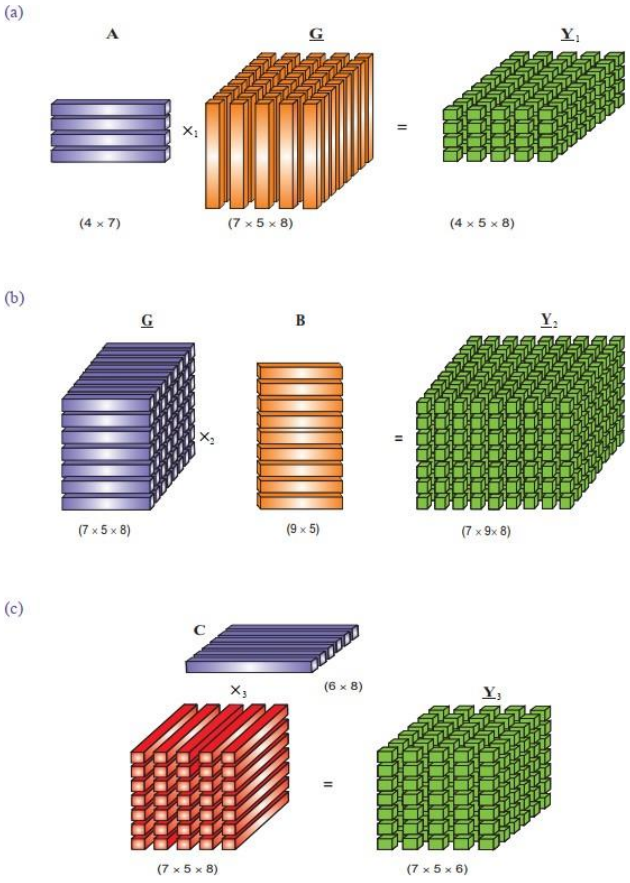


Figure 2.18: *Illustration of the mode-n multiplications of a third-order tensor by matrices. (a) mode- 1 multiplication. (b) mode- 2 multiplication. (c) mode- 3 multiplication.*

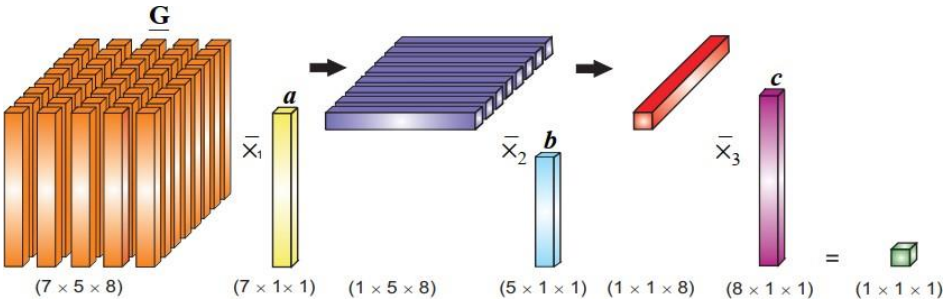


Figure 2.19: *Illustration of mode-n multiplication of a third-order tensor.*

Definition 2.4. (Multi-linear rank) The multi-linear rank of a tensor $\mathrm{X} \in \mathrm{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is a vector, noticed $rank_{\mathrm{A}}$ and is given by

$$rank_{\mathrm{A}} = (rank_1(\mathrm{X}_{(1)}), \ldots, rank_N(\mathrm{X}_{(N)})) \tag{2.9}$$

where $rank(\mathcal{X}_{(n)})$, $1 \leq n \leq N$ is the rank of the unfolding matrix $\mathcal{X}_{(n)}$.

In the following we introduce the well know TD, see [91] for more details.

Definition 2.5. (TD) Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$. the TD of $\mathcal{X}$ is given by

$$\mathcal{X} = \mathcal{V} \times_1 U_1 \times_2 U_2 \ldots \times_N U_N \tag{2.10}$$

where $\mathcal{V} \in \mathbb{R}^{R_1 \times R_2 \times \ldots \times R_N}$ is the core tensor, $U_n \in \mathbb{R}^{I_n \times R_n}$, $1 \leq n \leq N$ are factor matrices, and $(R_1, \ldots, R_N)$ is the multi-linear rank of the Tucker decomposition.

2.4 Einstein product

Definition 2.6. Let $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_L \times K_1 \times \ldots \times K_N}$, $\mathcal{B} \in \mathbb{R}^{K_1 \times \ldots \times K_N \times J_1 \times \ldots \times J_M}$, the Einstein product of the tensors $\mathcal{A}$ and $\mathcal{B}$ is the tensor of size $(I_1 \times \ldots \times I_L \times J_1 \times \ldots \times J_M)$ defined by :

$$(\mathcal{A} *_N \mathcal{B})_{i_1 \ldots i_L j_1 \ldots j_M} = \sum_{k_1=1}^{K_1} \sum_{k_2=1}^{K_2} \sum_{k_3=1}^{K_3} \ldots \sum_{k_N=1}^{K_N} a_{i_1 \ldots i_L k_1 \ldots k_N} \, b_{k_1 \ldots k_N \, j_1 \ldots j_M}$$

Definition 2.7. [77] Let $\mathcal{A} = a_{i_1,\ldots,i_N,j_1,\ldots,j_M} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$, $\mathcal{B} = b_{i_1,\ldots,i_M,j_1,\ldots,j_N} \in \mathbb{R}^{J_1 \times \ldots \times J_M \times I_1 \times \ldots \times I_N}$, if $b_{i_1,\ldots,i_M,j_1,\ldots,j_N} = a_{j_1,\ldots,j_N,i_1,\ldots,i_M}$ then $\mathcal{B}$ is called the transpose of $\mathcal{A}$ and denoted by $\mathcal{A}^T$.

A tensor $\mathcal{D} = d_{i_1,\ldots,i_N,j_1,\ldots,j_N} \in \mathbb{C}^{I_1 \times \ldots \times I_N \times I_1 \times \ldots \times I_N}$ is diagonal if $d_{i_1,\ldots,i_N,j_1,\ldots,j_N} = 0$ in the case that the indices $i_1, \ldots, i_N$ are different from $j_1, \ldots, j_N$. If all the entries of tensor are zero, we say this tensor is zero tensor, denoted by $\mathcal{O}$.

The trace of the tensor $\mathcal{A} \in \mathbb{C}^{I_1 \times \ldots \times I_N \times I_1 \times \ldots \times I_N}$ is given by $tr(\mathcal{A}) = \sum_{i_1 \ldots i_N} a_{i_1 \ldots i_N i_1 \ldots i_N}$.

The inner product of two tensors of the same order $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_m}$ is given by

$$\langle \mathcal{A}, \mathcal{B} \rangle = tr(\mathcal{B}^T *_N \mathcal{A}), \tag{2.11}$$

and the associated norm is defined by:

$$||\mathrm{A}||^2 = tr(\mathrm{A}^T *_N \mathrm{A}).$$

Definition 2.8. A tensor $\mathrm{X} \in \mathrm{R}^{I_1 \times \ldots \times I_N \times I_1 \times \ldots \times I_N}$ is called the inverse of the square tensor $\mathrm{A} \in \mathrm{R}^{I_1 \times \ldots \times I_N \times I_1 \times \ldots \times I_N}$ and denoted by A^{-1} if it satisfies :

$$\mathrm{A} *_N \mathrm{X} = \mathrm{X} *_N \mathrm{A} = \mathrm{I}$$

where I the identity tensor is called a unit tensor or identity tensor which all these entries are zero except for the diagonal entries $\mathrm{I}_{i_1,\ldots,i_N,i_1,\ldots,i_N} = 1$.

Now, we recall some definitions and proprieties related to Einstein product. An unfolded tensor is a matrix obtained by systematically reorganizing the tensor's entries into a two-dimensional array.

Definition 2.9. [27](Unfolding a tensor into a matrix) Define the transformation Ψ_{IJ} From the tensor space $\mathrm{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$ into the matrix space $\mathrm{R}^{(I_1 \ldots I_N) \times (J_1 \ldots J_M)}$ with $\Psi_{IJ}(\mathrm{X}) = X$ defined component-wise as

$$\mathrm{X}_{i_1 \ldots i_N j_1 \ldots j_M} = (X_{ivec(\mathbf{i},\mathrm{I})ivec(\mathbf{j},\mathrm{J})})$$

where $\mathrm{X} \in \mathrm{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$, $X \in \mathrm{R}^{(I_1 \ldots I_N) \times (J_1 \ldots J_M)}$, $ivec(\mathbf{i}, \mathrm{I})$ and $ivec(\mathbf{j}, \mathrm{J})$ are respectively two index functions corresponding to the subscript set $\mathbf{i} := \{i_1, \ldots, i_N\}$ and $\mathbf{j} := \{j_1, \ldots, j_M\}$, ie

$$ivec(\mathbf{i}, \mathrm{I}) = i_1 + \sum_{r=2}^{N} (i_r - 1) \prod_{u=1}^{r-1} I_u. \tag{2.12}$$

$$ivec(\mathbf{j}, \mathrm{J}) = j_1 + \sum_{s=2}^{M} (j_s - 1) \prod_{v=1}^{s-1} J_v. \tag{2.13}$$

and $\mathrm{I} := \{I_1, \ldots, I_N\}$, $\mathrm{J} := \{J_1, \ldots, J_M\}$ are referred to as the row mode and the column mode of X, respectively.

Definition 2.10. Let $\mathrm{U}, \mathrm{R} \in \mathrm{C}^{I_1 \times \ldots \times I_N \times I_1 \times \ldots \times I_N}$ non-zero tensors.

1. The tensor $\mathcal{U}$ is upper triangular if the entries $u_{i_1,\ldots,i_N j_1,\ldots,j_N} = 0$ when $ivec(\mathbf{i}, \mathcal{I}) \geq ivec(\mathbf{i}, \mathcal{J})$.
2. The tensor $\mathcal{R}$ is lower triangular if the entries $r_{i_1,\ldots,i_N j_1,\ldots,j_N} = 0$ when $ivec(\mathbf{i}, \mathcal{I}) \leq ivec(\mathbf{j}, \mathcal{J})$.

Definition 2.11. (Sub-Column tensor) Let $\mathcal{A} = (a_{i_1,\ldots,i_N j_1,\ldots,j_M}) \in \mathbb{C}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$. The j-th sub-column of $\mathcal{A}$ is the tensor $\mathcal{A}_{(:,\ldots,:,j_1,\ldots,j_M)} = (a_{:,\ldots,:,j_1,\ldots,j_M}) \in \mathbb{C}^{I_1 \times \ldots \times I_N}$ where $j = j_M + \sum_{r=1}^{M-1} (j_r - 1) \prod_{L=r+1}^{M} J_L$.

Next, we define the row and column block tensors.

Definition 2.12. Let $\mathcal{A} = (a_{i_1,\ldots,i_N j_1,\ldots,j_M}) \in \mathbb{C}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$, $\mathcal{B} = (b_{i_1,\ldots,i_N,k_1,\ldots,k_M}) \in \mathbb{C}^{I_1 \times \ldots \times I_N \times K_1 \times \ldots \times K_M}$, $\mathcal{C} = (c_{j_1,\ldots,j_M,i_1,\ldots,i_N}) \in \mathbb{C}^{J_1 \times \ldots \times J_M \times I_1 \times \ldots \times I_N}$ and $\mathcal{D} = (d_{k_1,\ldots,k_M,i_1,\ldots,i_N}) \in \mathbb{C}^{K_1 \times \ldots \times K_M \times I_1 \times \ldots \times I_N}$. Then

1. **Row block tensor** : the row block tensor of $\mathcal{A}$ and $\mathcal{B}$ denoted by $[\mathcal{A}\ \mathcal{B}]$ is the tensor of size $(\mathcal{I}_N \times \beta_1 \times \ldots \times \beta_M)$ where $\mathcal{I}_N = I_1 \times I_2 \times \ldots \times I_N, \beta_i = J_i + K_i, i = 1, \ldots, M$.
2. **Column block tensor** : the column block tensor of $\mathcal{C}, \mathcal{D}$ is defined as

$$\begin{bmatrix} \mathcal{C} \\ \mathcal{D} \end{bmatrix} = \left[C^T D^T \right]^T \in \mathbb{C}^{\beta_1 \times \ldots \times \beta_M \times \mathcal{I}_N}$$

3. Let $\mathcal{T}_1 = [\mathcal{A}_1\ \mathcal{B}_1]$, $\mathcal{T}_2 = [\mathcal{A}_2\ \mathcal{B}_2]$ be row block tensors, where $\mathcal{A}_1 \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$, $\mathcal{B}_1 \in \mathbb{R}^{I_1 \times \ldots \times I_N \times K_1 \times \ldots \times K_M}$, $\mathcal{A}_2 \in \mathbb{R}^{L_1 \times \ldots \times L_N \times J_1 \times \ldots \times J_M}$ and $\mathcal{B}_2 \in \mathbb{R}^{L_1 \times \ldots \times L_N \times K_1 \times \ldots \times K_M}$. We have

$$\begin{bmatrix} \mathcal{T}_1 \\ \mathcal{T}_2 \end{bmatrix} = \begin{bmatrix} \mathcal{A}_1 & \mathcal{B}_1 \\ \mathcal{A}_2 & \mathcal{B}_2 \end{bmatrix} \in \mathbb{C}^{\rho_1 \times \ldots \times \rho_N \times \beta_1 \times \ldots \times \beta_M}$$

where $\rho_i = I_i + L_i, i = 1, \ldots, N$ and $\beta_j = J_j + K_j, j = 1, \ldots, M$.

Proposition 2.13. *Let* $[\mathcal{A}\ \ \mathcal{B}], \begin{bmatrix} \mathcal{C} \\ \mathcal{D} \end{bmatrix}$ *as given by definition 2.12 and* $\mathcal{F} \in \mathbb{C}^{\mathcal{I}_N \times \mathcal{I}_N}$. *Then*

$$* \; F \; *_N [A \; B] = [F \; *_N \; A \; F \; *_N \; B] \in \; C^{I_N \times \beta_1 \times \; \ldots\ldots\ldots \times \beta_M}$$

$$*_{\square} \begin{matrix} C \\ D \end{matrix} * \; {}_N F = \begin{matrix} C *_N F \\ D *_N F \end{matrix} \in \; C^{\beta_1 \times \ldots \times \beta_M \times I_N}$$

$$* \; [A \; B] *_M \begin{matrix} C \\ D \end{matrix} = A * \; {}_M C + B * \; {}_M D \in \; C^{I_N \times I_N}$$

$$* \begin{matrix} A_1 & B_1 \\ A_2 & B_2 \end{matrix} *_M \begin{matrix} C \\ D \end{matrix} = \begin{matrix} A_1 *_M C + B_1 *_M D \\ A_2 *_M C + B_2 *_M D \end{matrix} \in \; C^{\rho_1 \times \ldots \times \rho_M \times I_N}$$

2.5 T-product

2.5.1 Discrete Fourier Transformation

The Discrete Fourier Transformation (DFT) plays a very important role in the definition of the T-product of tensors. The DFT on a vector $v \in \; R^n$ is defined by

$$\tilde{v} = F_n v \in \; C^n, \tag{2.14}$$

where F_n is the matrix defined as

$$F_n = \begin{matrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^{n-1} \\ . & . & . & \ldots & . \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \ldots & \omega^{(n-1)(n-1)} \end{matrix} \in \; C^{n \times n}, \tag{2.15}$$

where $\omega = e^{\frac{-2\pi i}{n}}$ with $i^2 = -1$. It is not difficult to show that (see [64])

$$F_n^* = \overline{F}_n, \text{ and } F_n^* F_n = F_n F_n^* = nI_n. \tag{2.16}$$

Then $F^{-1}_n = \frac{1}{n} \overline{F}_n$ which show that $\frac{1}{\sqrt{n}} F_n$ is a unitary matrix.
The cost of computing the vector $\tilde{v}$ directly from (2.14) is $O(n^2)$. Using the FFT , it will cost only $O(nlog(n))$ and this makes the FFT very fast for large

problems. It is known that

$$F_n \operatorname{circ}(v) F_n^{-1} = \operatorname{Diag}(\tilde{v}), \tag{2.17}$$

which is equivalent to

$$F_n \operatorname{circ}(v) F_n^{*} = n \operatorname{Diag}(\tilde{v}), \tag{2.18}$$

where

$$\operatorname{circ}(v) = \begin{bmatrix} v_1 & v_2 & \dots & v_n \\ v_2 & v_1 & \dots & v_{n-1} \\ \vdots & \vdots & \dots & \vdots \\ v_n & v_{n-1} & \dots & v_1 \end{bmatrix},$$

and $\operatorname{Diag}(\tilde{v})$, is the diagonal matrix whose i-th diagonal element is $\operatorname{Diag}(\tilde{v})_i$.

2.6 Cosine product (c-product)

2.6.1 Discrete Cosine Transformation

In this subsection, we recall some definitions and properties of the DCT and the c-product. The DCT plays a very important role in the definition of the c-product of tensors. The DCT on a vector $v \in \mathbb{R}^n$ is defined by

$$\tilde{v} = C_n v \in \mathbb{R}^n, \tag{2.19}$$

where C_n is the $n \times n$ DCT matrix with entries

$$(C_n)_{ij} = \sqrt{\frac{2 - \delta_{i1}}{n}} \cos\left(\frac{(i-1)(2j-1)\pi}{2n}\right) \qquad 1 < i, j < n$$

with δ_{ij} is the Kronecker delta. Its known that the matrix C_n is orthogonal, i.e, $C_n^T C_n = C_n C^T{}_n = I_n$; see [111]. Furthermore, for any vector $v \in \mathbb{R}^n$, the matrix vector multiplication $C_n v$ can be computed in $O(n\log(n))$ operations.

Also, Ng and al. [111] showed that matrices which can be diagonalized by C_n are some special Toeplitz-plus-Hankel matrices. In other words, we have

$$C_n \, \mathrm{th}(v) \, C_n^{-1} = \mathrm{Diag}(\tilde{v}), \tag{2.20}$$

where

$$\mathrm{th}(v) = \underbrace{\begin{bmatrix} v_1 & v_2 & \dots & v_n \\ v_2 & v_1 & \dots & v_3 \\ \vdots & \vdots & \ddots & \vdots \\ v_n & v_{n-1} & \dots & v_1 \end{bmatrix}}_{\text{Toeplitz}} + \underbrace{\begin{bmatrix} v_2 & \dots & v_n & 0 \\ \vdots & & & v_n \\ v_n & 0 & \dots & \vdots \\ 0 & v_n & \dots & v_2 \end{bmatrix}}_{\text{Hankel}}$$

and $\mathrm{Diag}(\tilde{v})$ is the diagonal matrix whose i-th diagonal element is $(\tilde{v})_i$.

2.7 Publications

[52] A. El Ichi , K. Jbilou and R. Sadaka, Tensor Global Extrapolation Methods Using the n-Mode and the Einstein Products, Mathematics, 8(8) (2020), 1298.

[15] F. P. A. Beik, A. El Ichi , K. Jbilou and R. Sadaka, Tensor extrapolation methods with applications, Numerical Algorithms, Numerical Algorithms 87(2021) 1421–1444.

[49] M. El Guide, A. El Ichi, and K. Jbilou, Discrete cosine transform LSQR methods for multidimensional ill-posed problems, Journal of Mathematical Modeling, (2021), 1-17.

[50] M. El Guide, A. El Ichi, K. Jbilou and R. Sadaka, On tensor GMRES and Golub-Kahan methods via the T-product for color image processing, The Electronic Journal of Linear Algebra 37(2021), 524-543.

[51] A. El Ichi , K. Jbilou and R. Sadaka, On tensor tubal-Krylov subspace methods, The Electronic Journal of Linear Algebra. (Accepted).

[16] F. P. A. Beik, M. El Guide, A. El Ichi, and K. Jbilou,Tensor GMRES and

Golub-Kahan Bidiagonalization methods via the Einstein product with applications to image and video processing. Numerical linear algebra with applications (revised).

[53] A. El Ichi, Tensor Krylov subspace methods via the T-product for large Sylvester tensor equations. (submitted).

[44] F. Dufrenois, A. El Ichi, M. Hached and K. Jbilou, Cosine multidimensional linear discrete analysis methods for face recognition in large data. (In preparation).

2.8 Manuscript Overview

The thesis can be divided into chapters :

- **Chapitre** 3: We will introduce some new tubal-Krylov subspace methods for solving linear tensor equations. Using the well known tensor T-product, we will in particular define the tensor tubal-global GMRES that could be seen as a generalization of the global GMRES. We also give a new tubal-version of the tensor Golub-Kahan algorithm. To this end, we first introduce some new tensor-tensor products and new related algebraic properties. The presented numerical tests compare the two methods and show the efficiency of the proposed procedures.
- **Chapitre** 4: In this chapter, we are interested in some tensor Krylov subspace methods to solve linear tensor equations. We use the well known Einstein product for two tensors to define the tensor global Arnoldi and the tensor Golub Kahan bidiagonalization algorithms. In particular, we will be interested in ill-posed linear problems and use some regularization techniques such as Tikhonov method, to obtain acceptable approximate solution. We will comment on the choice of the parameter appearing in these regularization techniques. Applications to video processing are given.

- **Chapitre** 5: This chapter is concerned with developing tensor iterative Krylov subspace methods to solve large multi-linear tensor equations. We use the T-product for two tensors to define tensor TGA and tensor tubal global Golub-Kahan bidiagonalization algorithms. Furthermore, we illustrate how tensor–based global approaches can be exploited to solve ill-posed problems arising from recovering blurry multichannel (color) images and videos, using the so-called Tikhonov regularization technique, to provide computable approximate regularized solutions. We also review a generalized cross-validation and discrepancy principle type of criterion for the selection of the regularization parameter in the Tikhonov regularization. Applications to image sequence processing are given to demonstrate the efficiency of the algorithms.
- **Chapitre** 6: In this chapter, we propose a new tensor Krylov subspace method for ill posed linear tensor problems such as in color or video image restoration. Those methods are based on the tensor-tensor DCT that gives fast tensor-tensor product computations. In particular, we will focus on the tensor discrete cosine versions of GMRES, Golub-Kahan bidiagonalisation and LSQR. The presented numerical tests show that the methods are very fast and give good accuracies when solving some linear tensor ill-posed problems.
- **Chapitre** 7: Introduce new tensor krylov subspace methods for solving large Sylvester tensor equations. The proposed method use the well known T-product for tensors and tensor subspaces. We introduce some new tensor products and the related algebraic properties. These new products will enable us to develop third-order the tFOM, tGMRES, TBA and the TBAS . We give some properties related to these method and propose some numerical experiments.
- **Chapitre** 8: We mainly develop the well known methods of extrapolation of vectors and polynomial matrices in a tensor framework. To

this end, some new products between tensors are defined and the concept of positive definition is extended for tensors corresponding to the T-product [89, 90]. In addition, we discuss the solution of the least squares problem associated with a tensor equation using TSVD. We describe how an extrapolation technique can also be implemented on the tensor sequence produced by the Ţ to solve possibly ill-posed tensor equations. We essentially develop some tensor extrapolation methods, namely TRRE, TMPE, TMMPE and TTEA. We give some properties and show how these new tensor extrapolation methods can be applied to sequences obtained by truncation of the tensor TSVD.

- **Chapitre** 9: We present new tensor extrapolation methods considered as generalizations of well known vector, matrix and block extrapolation methods such as polynomial extrapolation methods or *epsilon*-type algorithms corresponding to the n mode product [91]. We will define new tensor products that will be used to introduce global tensor extrapolation methods. We discuss the application of these methods to the solution of systems of linear and nonlinear equations and propose an efficient implementation of these methods by the global QR decomposition.

A general conclusion was given with some perspectives, suggestions and ideas for future work.

Chapter 3

Tensor tubal-Krylov subspace methods using the T-product

In this chapter, we will introduce some new tubal-Krylov subspace methods for solving some linear tensor equations. Using the well known tensor T-product, we will in particular define the tensor tubal-GMRES that could be seen as a generalization of the well known GMRES process. We also give a new tubal-version of the tensor Golub-Kahan algorithm. To this end, we first introduce some new tensor-tensor products and also some new definitions. The presented numerical tests compare the two methods and show the efficiency of the proposed procedures.

3.1 Introduction

We will develop some new tubal-Krylov subspace methods using projections onto special low dimensional tensor Krylov subspaces via the T-product. To this end, we introduce some new tensor products with some new related algebraic properties. Then we propose the The restarted tensor tubal global GMRES (TTGMRES) and the Tensor Tubal-Global Golub Kahan for linear tensor equation (TTGK) algorithms to solve linear tensor equations.
This chapter is organized as follows: In Section 3.2, we develop some new

tensor products and give some algebraic properties. After defining a Tubal-Global QR factorisation (TGQR) algorithm, we defined in Section 3.3, the tensor tubal-Global Arnoldi process that allows us to introduce the TTGMRES method. Section 3.4 is devoted to the TTGK method. Finally, some numerical tests are reported in Section 3.5.

3.2 Definitions and notations

3.2.1 Definitions and properties of the T-product

In this part, we briefly review some concepts and notations related to the T-Product, see [26, 89, 90] for more details. Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ be a third-order tensor, then the operations bcirc, unfold and fold are defined by

$$\text{bcirc}(\mathcal{A}) = \begin{bmatrix} A_1 & A_{n_3} & A_{n_3-1} & \dots & A_2 \\ A_2 & A_1 & A_{n_3} & \dots & A_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ A_n^{\,3} & A_{n_3-1} & \ddots & A_2 & A_1 \end{bmatrix} \in \mathbb{R}^{n_1 n_3 \times n_2 n_3},$$

$$\text{unfold}(\mathcal{A}) = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_{n_3} \end{bmatrix} \in \mathbb{R}^{n_1 n_3 \times n_2}, \qquad \text{fold}(\text{unfold}(\mathcal{A})) = \mathcal{A}.$$

Let $\tilde{\mathcal{A}}$ be the tensor obtained by applying the DFT on all the 3-mode tubes of the tensor $\mathcal{A}$. With the Matlab command **fft**, we have

$$\tilde{\mathcal{A}} = \mathbf{fft}(\mathcal{A}, [], 3), \text{ and } \mathbf{ifft}(\tilde{\mathcal{A}}, [], 3) = \mathcal{A},$$

where ifft denotes the Inverse Fast Fourier Transform.

The tensor $\widetilde{\mathcal{A}}$ can be obtained using the 3-mode product [89, 90, 91] as follows

$$\widetilde{\mathcal{A}} = \mathcal{A} \times_3 F_{n_3}, \qquad \mathcal{A} = \widetilde{\mathcal{A}} \times_3 F_n^{-1}{}_3 \tag{3.1}$$

where $\times_3$ is the 3-mode product defined in [91].Let

$\mathbf{A}$ be the matrix

$$\mathbf{A} = \text{BlockDiag}(\widetilde{\mathcal{A}}) = \begin{bmatrix} A^{(1)} & & & \\ & A^{(2)} & & \\ & & \ddots & \\ & & & A^{(n_3)} \end{bmatrix}, \tag{3.2}$$

and the matrices $A^{(i)}$'s are the frontal slices of the tensor $\widetilde{\mathcal{A}}$. The block circulant matrix bcirc($\mathcal{A}$) can also be block diagonalized by using the DFT and this gives

$$(F_{n_3} \otimes I_{n_1})\, \text{bcirc}(\mathcal{A})\, (F_{n_3}^{*} \otimes I_{n_2}) = \mathbf{A}, \tag{3.3}$$

As noticed in [90], the diagonal blocks of the matrix $\mathbf{A}$ satisfy the following property

$$\begin{aligned} &A^{(1)} \in \mathbb{R}^{n_1 \times n_2} \\ &conj(A^{(i)}) = A^{(n_3 - i + 2)}, \end{aligned} \tag{3.4}$$

where $conj(A^{(i)})$ is the complex conjugate of the matrix $A^{(i)}$. Next we recall the definition of the T-product.

Definition 3.1. The **T-product** (*) between two tensors $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times m \times n_3}$ is the $n_1 \times m \times n_3$ tensor given by:

$$\mathcal{A} * \mathcal{B} = \text{fold}(\text{bcirc}(\mathcal{A})\text{unfold}(\mathcal{B})).$$

Notice that from the relation (3.2), we can show that the product $\mathcal{C} = \mathcal{A} * \mathcal{B}$ is equivalent to $\mathbf{C} = \mathbf{AB}$. So, the efficient way to compute the T-product

is to use Fast Fourier Transform (FFT). Using the relation (3.4), the following algorithm allows us to compute in an efficient way the T-product of the tensors $\mathcal{A}$ and $\mathcal{B}$.

Algorithm 1 Computing the T-product via FFT

Inputs: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times m \times n_3}$

Output: $\mathcal{C} = \mathcal{A} * \mathcal{B} \in \mathbb{R}^{n_1 \times m \times n_3}$.

1. Compute $\tilde{\mathcal{A}} = \mathbf{fft}(\mathcal{A}, [], 3)$ and $\tilde{\mathcal{B}} = \mathbf{fft}(\mathcal{B}, [], 3)$.
2. Compute each frontal slices of $\tilde{\mathcal{C}}$ by

$$C^{(i)} = \begin{cases} A^{(i)} B^{(i)}, & i = 1, \ldots, \lfloor \frac{n_3 + 1}{2} \rfloor \\ conj(C^{(n_3 + i - 2)}), & i = \lfloor \frac{n_3 + 1}{2} \rfloor + 1, \ldots, n_3. \end{cases}$$

3. Compute $\mathcal{C} = \mathbf{ifft}(\tilde{C}, [], 3)$.

For the T-product, we have the following definitions

Definition 3.2. [90]

1. The identity tensor $\mathcal{I}_{n_1 n_1 n_3}$ is the tensor whose first frontal slice is the identity matrix $I_{n_1 n_1}$ and the other frontal slices are all zeros.
2. An $n_1 \times n_1 \times n_3$ tensor $\mathcal{A}$ is invertible, if there exists a tensor $\mathcal{B}$ of order $n_1 \times n_1 \times n_3$ such that

$$\mathcal{A} * \mathcal{B} = \mathcal{I}_{n_1 n_1 n_3} \quad \text{and} \quad \mathcal{B} * \mathcal{A} = \mathcal{I}_{n_1 n_1 n_3}.$$

   In that case, we set $\mathcal{B} = \mathcal{A}^{-1}$. It is clear that $\mathcal{A}$ is invertible if and only if bcirc($\mathcal{A}$) is invertible.
3. The transpose of $\mathcal{A}$ is obtained by transposing each of the frontal slices and then reversing the order of transposed frontal slices 2 through n_3.
4. If $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ are tensors of appropriate order, then

$$(\mathcal{A} * \mathcal{B}) * \mathcal{C} = \mathcal{A} * (\mathcal{B} * \mathcal{C}).$$

5. Suppose $\mathcal{A}$ and $\mathcal{B}$ are two tensors such $\mathcal{A} * \mathcal{B}$ and $\mathcal{B}^T * \mathcal{A}^T$ are defined. Then

$$(\mathcal{A} * \mathcal{B})^T = \mathcal{B}^T * \mathcal{A}^T.$$

Definition 3.3. Let $\mathcal{A}$ and $\mathcal{B}$ two tensors in $\mathbb{R}^{n_1 \times n_2 \times n_3}$. Then

1. The scalar inner product is defined by

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \sum_{i_3=1}^{n_3} a_{i_1 i_2 i_3} b_{i_1 i_2 i_3}.$$

2. The norm of $\mathcal{A}$ is defined by

$$\|\mathcal{A}\|_F = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}.$$

Remark 3.4. Another interesting way for computing the scalar product and the associated norm is as follows:

$$\langle \mathcal{A}, \mathcal{B} \rangle = \frac{1}{n_3} \langle \mathbf{A}, \mathbf{B} \rangle; \quad \|\mathcal{A}\|_F = \sqrt{\frac{1}{n_3}} \|\mathbf{A}\|_F$$

where the block diagonal matrix $\mathbf{A}$ is defined by (3.2).

Definition 3.5. [90]

1. An $n_1 \times n_1 \times n_3$ tensor $\mathcal{Q}$ is orthogonal if

$$\mathcal{Q}^T * \mathcal{Q} = \mathcal{Q} * \mathcal{Q}^T = \mathcal{I}_{n_1 n_1 n_3}.$$

2. A tensor is called f-diagonal if its frontal slices are orthogonal matrices. It is called upper triangular if all its frontal slices are upper triangular.

Definition 3.6. [107](Block tensor based on T-product) Suppose $\mathcal{A} \in \mathbb{R}^{n_1 \times m_1 \times n_3}$, $\mathcal{B} \in \mathbb{R}^{n_1 \times m_2 \times n_3}$, $\mathcal{C} \in \mathbb{R}^{n_2 \times m_1 \times n_3}$ and $\mathcal{D} \in \mathbb{R}^{n_2 \times m_2 \times n_3}$ are four tensors. The block tensor

$$\begin{bmatrix} \mathcal{A} & \mathcal{B} \\ \mathcal{C} & \mathcal{D} \end{bmatrix} \in \mathbb{R}^{(n_1+n_2) \times (m_1+m_2) \times n_3}$$

is defined by compositing the frontal slices of the four tensors.

Proposition 3.7. *Let* $\mathcal{A}, \mathcal{A}_1 \in \mathbb{R}^{n\times s\times n_3}$, $\mathcal{B}, \mathcal{B}_1 \in \mathbb{R}^{n\times p\times n_3}$, $\mathcal{A}_2 \in \mathbb{R}^{l\times s\times n_3}$, $\mathcal{B}_2 \in \mathbb{R}^{l\times p\times n_3}$, $\mathcal{C} \in \mathbb{R}^{s\times n\times n_3}$, $\mathcal{D} \in \mathbb{R}^{p\times n\times n_3}$ *and* $\mathcal{F} \in \mathbb{R}^{n\times n\times n_3}$ *. Then*

1. $\mathcal{F} * [\mathcal{A}\ \mathcal{B}] = [\mathcal{F} * \mathcal{A}\ \ \mathcal{F} * \mathcal{B}] \in \mathbb{R}^{n\times(s+p)\times n_3}$
2. $\begin{bmatrix}\mathcal{C}\\ \mathcal{D}\end{bmatrix} * \mathcal{F} = \begin{bmatrix}\mathcal{C} * \mathcal{F}\\ \mathcal{D} * \mathcal{F}\end{bmatrix} \in \mathbb{R}^{(s+p)\times n\times n_3}$
3. $[\mathcal{A}\ \mathcal{B}] * \begin{bmatrix}\mathcal{C}\\ \mathcal{D}\end{bmatrix} = \mathcal{A} * \mathcal{C} + \mathcal{B} * \mathcal{D} \in \mathbb{R}^{n\times n\times n_3}$
4. $\begin{bmatrix}\mathcal{A}_1 & \mathcal{B}_1\\ \mathcal{A}_2 & \mathcal{B}_2\end{bmatrix} * \begin{bmatrix}\mathcal{C}\\ \mathcal{D}\end{bmatrix} = \begin{bmatrix}\mathcal{A}_1 * \mathcal{C} + \mathcal{B}_1 * \mathcal{D}\\ \mathcal{A}_2 * \mathcal{C} + \mathcal{B}_2 * \mathcal{D}\end{bmatrix} \in \mathbb{R}^{(l+n)\times n\times n_3}$

3.2.2 New tensor products

In this part, we introduce now the *T-trace* transformation.

Definition 3.8. Let $\mathcal{A}$ be a tensor in $\mathbb{R}^{n_1\times n_1\times n_3}$. The tensor T-trace of $\mathcal{A}$ is a fiber-tensor of $\mathbb{R}^{1\times 1\times n_3}$ defined such that its i-th frontal slice is the trace of i-th frontal slice of $\mathcal{A}$, for $i = 1, \ldots, n_3$.

The T-trace($\mathcal{A}$) can be computed by the following Algorithm .

Algorithm 2 Tensor T-trace

1. **Input:** $\mathcal{A} \in \mathbb{R}^{n_1\times n_1\times n_3}$.
2. **Output:** $\mathbf{z} \in \mathbb{R}^{1\times 1\times n_3}$.
3. Set $\tilde{\mathcal{A}} = \texttt{fft}(\mathcal{A}, [], 3)$
   (a) for $i = 1, \ldots, n_3$
      i. $\mathbf{z}^{(i)} = trace(A^{(i)})$ (trace matrix)
   (b) End
4. $\mathbf{z} = \texttt{ifft}(\mathbf{z}, [], 3)$.
5. End

Now, we introduce new tensor products, that will be used for simplifying the algebraic computations of the main results.

Definition 3.9. Let $\mathbf{a} \in \mathbb{R}^{1\times 1\times n_3}$ and $\mathcal{B} = \mathcal{B}(i, j, :) = \mathbf{b}_{ij} \in \mathbb{R}^{m_1\times m_2\times n_3}$ for $i = 1, \ldots, m_1$, $j = 1, \ldots, m_2$. Then, the product $(\mathbf{a} \div \mathcal{B})$ is an $(m_1 \times m_2 \times n_3)$

tensor defined by

$$\mathbf{a} \div \mathsf{B} = \begin{bmatrix} \mathbf{a} * \mathbf{b}_{11} & \dots & \mathbf{a} * \mathbf{b}_{1m_2} \\ \vdots & \ddots & \vdots \\ \mathbf{a} * \mathbf{b}_{m_1 1} & \dots & \mathbf{a} * \mathbf{b}_{m_1 m_2} \end{bmatrix}$$

Remark 3.10. The $\div$ product is a generalisation of the product of a scalar with a matrix where the tubal-fiber plays the role of a scalar

3.2.2.1 The T-Kronecker and the Tubal-inner products

In the following we introduce the T-Kronecker product between two tensors as a generalisation of the classical Kronecker product for matrices.

Definition 3.11. (T-Kronecker product) Let $\mathsf{ffi} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathsf{B} \in \mathbb{R}^{m_1 \times m_2 \times n_3}$,the T-Kronecker product $\mathsf{ffi} \textcircled{2} \mathsf{B}$ between ffi and B is the $n_1 m_1 \times n_2 m_2 \times n_3$ tensor given by :

$$(\mathsf{ffi} \textcircled{2} \mathsf{B}) = (\mathsf{f} \widetilde{\textcircled{2}} \mathsf{B}) \times_3 F_{n_3}^{-1}$$

where the i-th frontal slice ($i = 1, \dots, n_3$) of $(\mathsf{f} \widetilde{\textcircled{2}} \mathsf{B})$ is given by,

$$(\mathsf{ffi} \textcircled{2} \mathsf{B})^{(i)} = (\mathsf{ffi} \times_3 F_{n_3})^{(i)} \otimes (\mathsf{B} \times_3 F_{n_3})^{(i)}$$

where $\otimes$ is the Kronecker product between two matrices. The matrices $(\mathsf{ffi} \times_3 F_{n_3})^{(i)}$ and $(\mathsf{B} \times_3 F_{n_3})^{(i)}$ are the i-th frontal slices of $\tilde{\mathsf{f}}$ and $\tilde{\mathsf{B}}$, respectively.

The T-Kronecker product of two tensors can be computed by the following algorithm.

Proposition 3.12. *Let* $\mathsf{ffi} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathsf{B} \in \mathbb{R}^{m_1 \times m_2 \times n_3}$, $\mathsf{C} \in \mathbb{R}^{n_2 \times r_1 \times n_3}$ *and* $\mathsf{D} \in \mathbb{R}^{m_2 \times r_2 \times n_3}$. *Then we have the following properties*

1. $(\mathsf{ffi} \textcircled{2} \mathsf{B})^T = \mathsf{ffi}^T \textcircled{2} \mathsf{B}^T$
2. $(\mathsf{ffi} \textcircled{2} \mathsf{B}) * (\mathsf{C} \textcircled{2} \mathsf{D}) = (\mathsf{ffi} * \mathsf{C}) \textcircled{2} (\mathsf{B} * \mathsf{D})$

Algorithm 3 Tensor T-Kronecker

1. **Input.** $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{m_1 \times m_2 \times n_3}$.
2. **Output.** $(\mathcal{A} \circledast \mathcal{B})$ is the tensor of size $n_1 m_1 \times n_2 m_2$.
3. Set $\tilde{\mathcal{A}} = \texttt{fft}(\mathcal{A}, [], 3)$ and $\tilde{\mathcal{B}} = \texttt{fft}(\mathcal{B}, [], 3)$
   (a) for $i = 1, \ldots, n_3$
   $(\mathcal{A} \circledast \mathcal{B})^{(i)} = (A^{(i)} \otimes B^{(i)})$.
   (b) End
4. $(\mathcal{A} \circledast \mathcal{B}) = \texttt{ifft}((\widetilde{\mathcal{A} \circledast \mathcal{B}}), [], 3)$
5. End

3. If $\mathcal{A} \in \mathbb{R}^{n \times n \times z}$ *and* $\mathcal{B} \in \mathbb{R}^{p \times p \times z}$ *are invertible then* $(\mathcal{A} \circledast \mathcal{B})^{-1}$ *is invertible and we have :*

$$(\mathcal{A} \circledast \mathcal{B})^{-1} = \mathcal{A}^{-1} \circledast \mathcal{B}^{-1}$$

Proof. Obviously, the results stems directly from the properties of the matrix-Kronecker. In fact, for $i = 1, \ldots, n_3$, we have :

$$\begin{aligned}((\mathcal{A}^T \circledast \mathcal{B}^T) \times_3 F_{n_3})^{(i)} &= ((\mathcal{A} \times_3 F_{n_3})^{(i)T} \otimes (\mathcal{B} \times_3 F_{n_3})^{(i)T}) \\ &= ((\mathcal{A} \times_3 F_{n_3})^{(i)} \otimes (\mathcal{B} \times_3 F_{n_3})^{(i)})^T \\ &= ((\mathcal{A} \circledast \mathcal{B})^T \times_3 F_{n_3})^{(i)}\end{aligned}$$

which shows that $(\mathcal{A} \circledast \mathcal{B})^T = (\mathcal{A}^T \circledast \mathcal{B}^T)$. The two other properties are shown in the same way. □

Next, we define a new Tubal-inner product that will be used later.

Definition 3.13. (Tubal-inner product) For $\mathcal{X}, \mathcal{Y}$ two tensors in $\mathbb{R}^{n_1 \times s \times n_3}$, we define the Tubal-inner products $\langle ., . \rangle_T$ is defined by:

$$\begin{aligned}\mathbb{R}^{n_1 \times s \times n_3} \times \mathbb{R}^{n_1 \times s \times n_3} &\longrightarrow \mathbb{R}^{1 \times 1 \times n_3} \\ (\mathcal{X}, \mathcal{Y}) &\longmapsto \langle \mathcal{X}, \mathcal{Y} \rangle_T = \text{T-trace}(\mathcal{X}^T * \mathcal{Y}).\end{aligned} \quad . \tag{3.5}$$

Let $\mathcal{X}_1, \ldots, \mathcal{X}_l$ be a collection of l third tensors in $\mathbb{R}^{n_1 \times s \times n_3}$, if

$$\langle \mathcal{X}_i, \mathcal{X}_j \rangle_T = \begin{cases} \alpha_i \mathbf{e} & i = j \\ 0 & i \neq j, \end{cases}.$$

where α_i is a non-zero scalar and MatVec($\mathbf{e}$) $= (1, 0, 0 \ldots, 0)^T$, then the set $\{\mathcal{X}_1, \ldots, \mathcal{X}_l\}$ is said to be a T-orthogonal collection of tensors. The collection is called T-orthonormal if $\alpha_i = 1, i = 1, \ldots, l$.

Notice that the T-trace of $(\mathcal{X}^T * \mathcal{Y})$ can be expressed via the 3-mode product as follows:

$$(\text{T-trace}(\mathcal{X}^T * \mathcal{Y}) \times_3 F_{n_3})^{(i)} = \text{trace}((\mathcal{X} \times_3 F_{n_3})^{(i)T} (\mathcal{Y} \times_3 F_{n_3})^{(i)}) \quad , i = 1, \ldots, n_3$$

Proposition 3.14. *Let* $\mathcal{ffi}, \mathcal{B}$ *and* $\mathcal{C}$ *be tensors of* $\mathbb{R}^{n_1 \times s \times n_3}$ *and* $\mathbf{a} \in \mathbb{R}^{1 \times 1 \times n_3}$. *Thenthe Tubal-inner product satisfies the following properties*

1. $\langle \mathcal{ffi}, \mathcal{B} + \mathcal{C} \rangle_T = \langle \mathcal{ffi}, \mathcal{B} \rangle_T + \langle \mathcal{ffi}, \mathcal{C} \rangle_T$.
2. $\langle \mathcal{ffi}, \mathbf{a} \div \mathcal{B} \rangle_T = \mathbf{a} * \langle \mathcal{ffi}, \mathcal{B} \rangle_T$.
3. $\langle \mathcal{ffi}, \mathcal{X} * \mathcal{B} \rangle_T = \langle \mathcal{X}^T * \mathcal{ffi}, \mathcal{B} \rangle_T$, *for* $\mathcal{X} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$.

Proof. For $i = 1, \ldots, n_3$, we have

$$\begin{aligned}(\text{T-trace}(\mathcal{ffi}^T * (\mathcal{B} + \mathcal{C})) \times_3 F_{n_3})^{(i)} &= \text{trace} \ (\mathcal{ffi} \times_3 F_{n_3})^{(i)T} ((\mathcal{B} \times_3 F_{n_3})^{(i)} + (\mathcal{C} \times_3 F_{n_3})^{(i)}) \\ &= \text{trace} \ (\mathcal{ffi} \times_3 F_{n_3})^{(i)T} (\mathcal{B} \times_3 F_{n_3})^{(i)} + (\mathcal{ffi} \times_3 F_{n_3})^{(i)T} (\mathcal{C} \times_3 \\ &= (\text{T-trace}(\mathcal{ffi}^T * \mathcal{B} + \mathcal{ffi}^T * \mathcal{C}) \times_3 F_{n_3})^{(i)}.\end{aligned}$$

which shows the first property. The other properties could be easily shown in the same way. □

3.2.2.2 T-Diamond product of third order tensor format

Now, we introduce the T-Diamond product between two tensors and give some algebraic properties.

Definition 3.15. Let $\mathcal{A} = [\mathcal{A}_1, \ldots, \mathcal{A}_p]$ where $\mathcal{A}_i$, $i = 1, ..., p$, is an $n_1 \times s \times n_3$ tensor and let $\mathcal{B} = [\mathcal{B}_1, \ldots, \mathcal{B}_l]$ where $\mathcal{B}_j$, $j = 1, ..., l$ is an $n_1 \times s \times n_3$ tensor. Then the T-diamond product $\mathcal{A}^T\diamond \mathcal{B}$ is the tensor of size $p \times l \times n_3$ given by :

$$(\mathcal{A}\diamond \mathcal{B}) = (\widetilde{\mathcal{A}\diamond \mathcal{B}}) \times_3 F_{n_3}^{-1},$$

where the i-th frontal slice of $(\widetilde{\mathcal{A}\diamond \mathcal{B}})$ is given by

$$(\mathcal{A}\diamond \mathcal{B})^{(i)} = (\mathcal{A} \times_3 F_{n_3})^{(i)T} \diamond (\mathcal{B} \times_3 F_{n_3})^{(i)},$$

where $\diamond$ is the diamond product between two matrices; for more details about the diamond product between two matrices, see [24].

The T-diamond product can be computed by the following algorithm.

Algorithm 4 Tensor T-Diamond product

1. **Input.** $\mathcal{A} \in \mathbb{R}^{n_1 \times ps \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_1 \times ls \times n_3}$.
2. **Output.** $\mathcal{A}^T\diamond \mathcal{B}$ is the tensor of size $p \times l \times n_3$
3. Set $\tilde{\mathcal{A}} = \texttt{fft}(\mathcal{A}, [], 3)$ and $\tilde{\mathcal{B}} = \texttt{fft}(\mathcal{B}, [], 3)$
   (a) for $i = 1, \ldots, n_3$
      i. $(\mathcal{A}^T\diamond \mathcal{B})^{(i)} = (A^{(i)T} \diamond B^{(i)})$ (Diamond matrix product)
   (b) End
4. $(\mathcal{A}^T\diamond \mathcal{B}) = \texttt{ifft}((\widetilde{\mathcal{A}^T\diamond \mathcal{B}}), [], 3)$
5. End

The next proposition gives some algebraic properties of the T-diamond product.

Proposition 3.16. *Let* $\mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathbb{R}^{n_1 \times ps \times n_3}$, $\mathcal{D} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ *and* $\mathcal{L} \in \mathbb{R}^{p^\times p^\times n_3}$,*We have the following proposals:*

1. $(\mathcal{A} + \mathcal{B})^T\diamond \mathcal{C} = \mathcal{A}^T\diamond \mathcal{C} + \mathcal{B}^T\diamond \mathcal{C}$
2. $\mathcal{A}^T\diamond (\mathcal{B} + \mathcal{C}) = \mathcal{A}^T\diamond \mathcal{B} + \mathcal{A}^T\diamond \mathcal{C}$
3. $(\mathcal{A}^T\diamond \mathcal{B})^T = \mathcal{B}^T\diamond \mathcal{A}$
4. $(\mathcal{D} * \mathcal{A})^T\diamond \mathcal{B} = \mathcal{A}^T\diamond (\mathcal{D}^T * \mathcal{B})$

5. $\mathcal{F}^{T}\blacklozenge(\mathcal{B} * (\mathcal{L} ② \mathrm{I}_{ssn_3})) = (\mathcal{F}^{T}\blacklozenge \mathcal{B}) * \mathcal{L}$

Proof. Obviously, the results are derived directly from the properties of the matrix-$\diamond$ product. For $i = 1, \ldots, n_3$ we have

$$\begin{aligned}\mathcal{F}^{T}\blacklozenge(\mathcal{B} * (\mathcal{L} ② \mathrm{I}_{ssn_3}))) \times_3 F_{n_3}^{(i)} &= (\mathcal{F} \times_3 F_{n_3})^{(i)T} \diamond (\mathcal{B} \times_3 F_{n_3})^{(i)}((\mathcal{L} \times_3 F_{n_3})^{(i)} \otimes (\mathrm{I}_{ssn_3} \times \\ &= \left[(\mathcal{F} \times_3 F_{n_3})^{(i)T} \diamond (\mathcal{B} \times_3 F_{n_3})^{(i)} \quad (\mathcal{L} \times_3 F_{n_3})^{(i)}\right] \\ &= ((\mathcal{F}^{T}\blacklozenge \mathcal{B}) * \mathcal{L}) \times_3 F_{n_3}^{(i)}.\end{aligned}$$

Finally we get : $\mathcal{F}^{T}\blacklozenge(\mathcal{B} * (\mathcal{L} ② \mathrm{I}_{ssn_3})) = (\mathcal{F}^{T}\blacklozenge \mathcal{B}) * \mathcal{L}$. The other results are obtained by following in the same manner. □

3.3 The tensor tubal global GMRES method

3.3.1 Tubal global QR factorization

Next, we present the Tubal-Global Gram-Schmidt (TGGS) process.

Definition 3.17. [89] Let $\mathbf{z} \in \mathbb{R}^{1\times 1\times n_3}$, the tubal rank of $\mathbf{z}$ is the number of its non-zero Fourier coefficients. If the tubal-rank of $\mathbf{z}$ is equal to n_3, we say that it is invertible and we denote by $(\mathbf{z})^{-1}$ the inverse of $\mathbf{z}$ iff: $\mathbf{z} * (\mathbf{z})^{-1} = (\mathbf{z})^{-1} * \mathbf{z} = \mathbf{e}$.

First, we need to introduce a normalization algorithm. This means that given a non-zero $\mathcal{F} \in \mathbb{R}^{n_1\times s\times n_3}$ we need to be able to write :

$$\mathcal{F} = \mathbf{a} \div \mathcal{Q} = \mathcal{Q} * (\mathbf{a} ② \mathrm{I}_{ssn_3})$$

where $\mathbf{a}$ is invertible and $\langle \mathcal{Q}, \mathcal{Q} \rangle_T = \mathbf{e}$, where $\mathbf{e}$ is the tubal-fiber such that $\text{unfold}(\mathbf{e}) = (1, 0, 0 \ldots, 0)^T$.

We consider the following normalization algorithm.

Algorithm 5 A normalization algorithm (Normalization(ffi))

1. **Input.** ffi $\in \mathbb{R}^{n_1 \times s \times n_3}$ and $tol > 0$.
2. **Output.** $\mathcal{Q} \in \mathbb{R}^{n_1 \times s \times n_3}$ and $\mathbf{a} \in \mathbb{R}^{1 \times 1 \times n_3}$ such that $\langle \mathcal{Q}, \mathcal{Q} \rangle_T = \mathbf{e}$.
3. Set $\tilde{\text{ffi}} = \texttt{fft}(\text{ffi}, [], 3)$
   (a) for $j = 1, \ldots, n_3$
      i. $\mathbf{a}^{(j)} = \text{trace}(A^{(j)T} A^{(j)}) = ||A^{(j)}||_F$
      ii. if $\mathbf{a}^{(j)} <$tol, stop
      iii. else $\mathcal{Q}^{(j)} = \frac{A^{(j)}}{\mathbf{a}^{(j)}}$
   (b) End for
4. $\mathcal{Q} = \texttt{ifft}(\tilde{\mathcal{Q}}, [], 3)$, $\mathbf{a} = \texttt{ifft}(\mathbf{a}, [], 3)$
5. End

The Algorithm 6 summarizes the different steps defining the TGQR of a tensor.

Algorithm 6 Tensor Tubal-Global QR decomposition

1. **Input.** $\mathcal{Z} = [\mathcal{Z}_1, \ldots, \mathcal{Z}_k]$ be an $n \times ks \times n_3$ tensor where $\mathcal{Z}_j$ is an $n \times s \times n_3$ tensor, for $j = 1, \ldots, k$. $s < n$.
2. Set $[\mathcal{Q}_1, \mathbf{r}_{1,1}] =$ Normalization($\mathcal{Z}_1$): using Algorithm 5.
3. For $j = 2, \ldots, k$
   (a) $\mathcal{W} = \mathcal{Q}_j$
   (b) for $i = 1, \ldots, j-1$
      i. $\mathbf{r}_{i,j} = \langle \mathcal{Q}_i, \mathcal{W} \rangle_T$
      ii. $\mathcal{W} = \mathcal{W} - \mathbf{r}_{i,j} \div \mathcal{Q}_i$
   (c) End for
   (d) $[\mathcal{Q}_j, \mathbf{r}_{j,j}] =$ Normalization($\mathcal{W}$): using Algorithm 5.
4. End

Proposition 3.18. *Let* $\mathcal{Z} = [\mathcal{Z}_1, \ldots, \mathcal{Z}_k]$ *be an* $n \times ks \times n_3$ *tensor where* $\mathcal{Z}_j$ *is ann* $\times s \times n_3$ *tensor, for* $j = 1, \ldots, k$. *Then from Algorithm 6,* $\mathcal{Z}$ *can be factored as*

$$\mathcal{Z} = \mathcal{Q} * (\mathcal{R} \textcircled{2} \mathcal{I}_{ssn_3}),$$

where $\mathcal{Q} = [\mathcal{Q}_1, \ldots, \mathcal{Q}_k]$ *is an* $(n \times ks \times n_3)$ *T-orthonormal tensor satisfying* $\mathcal{Q}^T \blacklozenge \mathcal{Q} = \mathcal{I}_{k,k,n_3}$ *and* $\mathcal{R}$ *is an upper triangular* $(k \times k \times n_3)$ *tensor (each frontal slice of* $\mathcal{R}$ *is an*

upper triangular matrix of size $k \times k$) given by

$$R = \begin{bmatrix} \mathbf{r}_{1,1} & \mathbf{r}_{1,2} & \cdots & \mathbf{r}_{1,k} \\ & \mathbf{r}_{2,2} & \cdots & \mathbf{r}_{2,k} \\ & & \ddots & \vdots \\ & & & \mathbf{r}_{k,k} \end{bmatrix} \in \mathbb{R}^{k \times k \times n_3}$$

Proof. This will be shown by induction on k. For $k = 1$, we have from Line 2 of Algorithm 6: $\langle \mathcal{Q}_1, \mathcal{Q}_1 \rangle_T = \mathbf{e}$. Assume now that the result is true for some k. Using the results of Proposition 3.14, we obtain

$$\begin{aligned} (\mathbf{r}_{k+1,k+1}) * \langle \mathcal{Q}_j, \mathcal{Q}_{k+1} \rangle_T &= \langle \mathcal{Q}_j, (\mathcal{W} - \sum_{i=1}^{k-1} \mathbf{r}_{i,k} \div \mathcal{Q}_i) \rangle_T \\ &= \langle \mathcal{Q}_j, \mathcal{W} \rangle_T - \sum_{i=1}^{k-1} \mathbf{r}_{i,k} * \langle \mathcal{Q}_j, \mathcal{Q}_i \rangle_T \\ &= (\mathbf{r}_{j,k} - \mathbf{r}_{j,k}) = \mathbf{o}, \; j = 1, \ldots, k. \end{aligned}$$

where $\mathbf{o}$ denote de zeros tube fiber of size $(1 \times 1 \times n_3)$ which all his entries are equal to zeros. Then we get $\mathcal{Q}^T \blacklozenge \mathcal{Q} = \mathcal{I}_{kkn_3}$. Let $\mathcal{Z} = [\mathcal{Z}_1, \ldots, \mathcal{Z}_k]$ be an $n \times ks \times n_3$ tensor where $\mathcal{Z}_j$ is an $n \times s \times n_3$ tensor. Then from Algorithm 6, we have $\mathcal{Z}_j = \sum_{i=1}^{j} \mathrm{r}_{i,j} \div \mathcal{Q}_i$ and the j-th lateral slice of $\mathcal{Z}$ is given by :

$$\begin{aligned} (\mathcal{Z})_j = \mathcal{Z}_j &= \sum_{i=1}^{j} \mathbf{r}_{i,j} \div \mathcal{Q}_i \\ &= \sum_{i=1}^{j} \mathcal{Q}_i * ((\mathbf{r}_{i,j}) \circledcirc \mathcal{I}_{s,s,n_3}) \\ &= [\mathcal{Q}_1, \ldots, \mathcal{Q}_j] * \begin{bmatrix} \mathbf{r}_{1,j} \\ \vdots \\ \mathbf{r}_{j,j} \end{bmatrix} \circledcirc \mathcal{I}_{s,s,n_3}. \end{aligned}$$

Let $\mathsf{R}_j = \begin{bmatrix} \mathbf{r}_{1,j} \\ \vdots \\ \mathbf{r}_{j,j} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{k \times 1 \times n_3}$ be the j-th lateral slice of the $(k \times k \times n_3)$ tensor $\mathsf{R} = [\mathsf{R}_1, \ldots, \mathsf{R}_k]$. Then we have the decomposition

$$\mathsf{Z}_j = [\mathsf{Q}_1, \ldots, \mathsf{Q}_j] * \mathsf{R}_j \circledcirc \mathsf{I}_{s,s,n_3} \qquad j = 1, \ldots, k.$$

Therefore, Z can be factored as $\mathsf{Z} = \mathsf{Q} * (\mathsf{R} \circledcirc \mathsf{I}_{s,s,n_3})$ where $[\mathsf{Q}_1, \ldots, \mathsf{Q}_k]$ is an $(n \times ks \times n_3)$ T-orthonormal tensor $\mathsf{Q}^T \blacklozenge \mathsf{Q} = \mathsf{I}_{kkn_3}$ and R is an upper trian- gular $(k \times k \times n_3)$ tensor. Notice that $\mathsf{Q}^T \blacklozenge \mathsf{Z} = \mathsf{Q}^T \blacklozenge (\mathsf{Q} * (\mathsf{R} \circledcirc \mathsf{I}_{ssn_3}))$, and by using the result 5 in Proposition 3.16, we get $\mathsf{Q}^T \blacklozenge \mathsf{Z} = (\mathsf{Q}^T \blacklozenge \mathsf{Q}) * \mathsf{R} = \mathsf{R}$. □

3.3.2 The tensor tubal-global Arnoldi process

In this section, we define the Tensor Tubal-Global Arnoldi (TTGA) process that could be considered as a generalisation of the global Arnoldi process defined in [81] for matrices. In [48], the authors introduced the TGA process. The main difference between the TTGA and the TGA is that for TGA process the tensor Krylov global subspace $\mathsf{TK}_m(\mathsf{ffi}, \mathsf{V})$ associated to the T-product is as follows

$$\mathsf{TK}_m(\mathsf{ffi}, \mathsf{V}) = \mathrm{Tspan}\{\mathsf{V}, \mathsf{ffi} * \mathsf{V}, \ldots, \mathsf{ffi}^{m-1} * \mathsf{V}\} = \left(\mathsf{Z} \in \mathbb{R}^{n \times s \times n_3}, \mathsf{Z} = \sum_{i=1}^{m} \alpha_i \, \mathsf{ffi}^{i-1} * \mathsf{V} \right) \tag{3.6}$$

where $\alpha_i \in \mathbb{R}$, $i = 1, \ldots, m$; $\mathsf{ffi} \in \mathbb{R}^{n \times n \times n_3}$ and $\mathsf{V} \in \mathbb{R}^{n \times s \times n_3}$ $(s << n)$. In the case of the TTGA process, the tensor Tubal global Krylov subspace of order m generated by ffi and V and denoted by $\mathsf{TK}_m^g(\mathsf{ffi}, \mathsf{V}) \subset \mathbb{R}^{n \times s \times n_3}$ is defined by

:

$$\mathcal{TK}_m^g(\mathcal{A}, \mathcal{V}) = \text{T-Span}\left\{\mathcal{V}, \mathcal{A} * \mathcal{V}, \mathcal{A}^2 * \mathcal{V}, \ldots, \mathcal{A}^{m-1} * \mathcal{V}\right\} \tag{3.7}$$

$$= \left\{\mathcal{Z} \in \mathbb{R}^{n \times s \times n^3}, \mathcal{Z} = \sum_{i=1}^{m} \mathbf{a}_i \div (\mathcal{A}^{i-1} * \mathcal{V})\right. \tag{3.8}$$

where $\mathbf{a}_i \in \mathbb{R}^{1 \times 1 \times n^3}$, $\mathcal{A}^{i-1} * \mathcal{V} = \mathcal{A}^{i-2} * \mathcal{A} * \mathcal{V}, i = 2, \ldots, m$ and $\mathcal{A}^0$ is the identity tensor .

The following TTGA process produces a T-orthogonormal basis of $\mathcal{TK}_m^g(\mathcal{A}, \mathcal{V})$. The algorithm is described as follows

Algorithm 7 The Tensor Tubal-Global Arnoldi

1. **Input.** $\mathcal{A} \in \mathbb{R}^{n \times n \times n^3}$, $\mathcal{V} \in \mathbb{R}^{n \times s \times n^3}$ and and the positive integer m.
2. Set $[\mathcal{V}_1, r_{1,1}] = \text{Normalization}(\mathcal{V})$
3. For $j = 1, \ldots, m$
   (a) $\mathcal{W} = \mathcal{A} * \mathcal{V}_j$
   (b) for $i = 1, \ldots, j$
      i. $\mathbf{h}_{i,j} = \langle \mathcal{V}_i, \mathcal{W} \rangle_T$
      ii. $\mathcal{W} = \mathcal{W} - \mathbf{h}_{ij} \div \mathcal{V}_i$
   (c) End for
   (d) $[\mathcal{V}_{j+1}, \mathbf{h}_{j+1,j}] = \text{Normalization}(\mathcal{W})$
4. End

Proposition 3.19. *Suppose that m steps of Algorithm 7 have been run. Then, the tensors* $\mathcal{V}_1, \ldots, \mathcal{V}_m$*, form a T-orthonormal basis of the Tubal-global Krylov subspace* $\mathcal{TK}_m^g(\mathcal{A}, \mathcal{V})$*.*

Proof. This will be shown by induction on m. For $m = 1$, we have from Line 2 of Algorithm 6 the relation $\langle \mathcal{V}_1, \mathcal{V}_1 \rangle_T = \mathbf{e}$. Assume now that the result is true for some m, then from Algorithm 6 and by using the results of Proposition

3.14, we get

$$
\begin{aligned}
(\mathbf{h}_{m+1,m}) * \langle \mathcal{V}_j, \mathcal{V}_{m+1} \rangle_T &= \langle \mathcal{V}_j, (\mathbf{h}_{m+1,m}) \div \mathcal{V}_{m+1} \rangle_T \\
&= \langle \mathcal{V}_j, (\mathcal{W} - \sum_{i=1}^{m} \mathbf{h}_{i,m} \div \mathcal{V}_i) \rangle_T \\
&= \langle \mathcal{V}_j, \mathcal{W} \rangle_T - (\sum_{i=1}^{m} \mathbf{h}_{i,m} * \langle \mathcal{V}_j, \mathcal{V}_i \rangle_T) \\
&= (\mathbf{h}_{j,m} - \mathbf{h}_{j,m}) = \mathbf{o}, \ = 1, \ldots, m.
\end{aligned}
$$

where $\mathbf{o}$ denote de zeros tube fiber of size $(1 \times 1 \times n_3)$ which all his entries are equal to zeros. Furthermore, from Line 3(d) of Algorithm 7, we immediately have $\langle \mathcal{V}_{m+1}, \mathcal{V}_{m+1} \rangle_T = \mathbf{e}$. Therefore, the result is true for $m + 1$ which completes the proof. □

Let $\mathcal{V}_m$ be the $(n \times sm \times n_3)$ tensor whose frontal slices are $\mathcal{V}_1, \ldots, \mathcal{V}_m$ and let $\mathcal{H}_m$ the $(m + 1) \times m \times n_3$ hessenberg tensor defined by Algorithm 7 (Hessemberg tensor mean that every frontal slice of $\mathcal{H}_m$ is a Hesemberg matrix) and by $\mathcal{H}_m$ the tensor obtained from $\mathcal{H}_m$ by deleting its last horizontal slice. $\mathcal{V}_m$ is the $(n \times (sm) \times n_3)$ tensor whose frontal slices are $\mathcal{V}_1, \ldots,$

$\mathcal{V}_m$, respectively. Using Definition 3.6, we can set

$$\mathcal{V}_m := [\mathcal{V}_1, \ldots, \mathcal{V}_m] \in \mathbb{R}^{n \times sm \times n_3},$$

$$\text{ffi} * \mathcal{V}_m := [\text{ffi} * \mathcal{V}_1, \ldots, \text{ffi} * \mathcal{V}_m] \in \mathbb{R}^{n \times sm \times n_3}$$

$$\mathcal{V}_{m+1} := [\mathcal{V}_m, \mathcal{V}_{m+1}] \in \mathbb{R}^{n_1 \times (m+1)s \times n_3}$$

$$\tilde{\mathcal{H}}_m = \begin{bmatrix} \mathbf{h}_{1,1} & \mathbf{h}_{1,2} & \cdots & \mathbf{h}_{1,m} \\ \mathbf{h}_{2,1} & \mathbf{h}_{2,2} & \cdots & \mathbf{h}_{2,m} \\ & \ddots & \ddots & \vdots \\ & & \mathbf{h}_{m,m-1} & \mathbf{h}_{m,m} \\ & & & \mathbf{h}_{m+1,m} \end{bmatrix} \in \mathbb{R}^{(m+1) \times m \times n_3}$$

$$\mathcal{H}_m = \begin{bmatrix} \mathbf{h}_{1,1} & \mathbf{h}_{1,2} & \cdots & \mathbf{h}_{1,m} \\ \mathbf{h}_{2,1} & \mathbf{h}_{2,2} & \cdots & \mathbf{h}_{2,m} \\ & \ddots & \ddots & \vdots \\ & & \mathbf{h}_{m,m-1} & \mathbf{h}_{m,m} \end{bmatrix} \in \mathbb{R}^{m \times m \times n_3}$$

Its easy to see that $\tilde{H}_m$ can be obtained from $\tilde{\mathcal{H}}_m$ by deleting the horizontal slice

$$[\mathbf{o}, \ldots, \mathbf{o}, \mathbf{h}_{m+1,m}] = \mathbf{h}_{m+1,m} * \mathcal{E}_m \in \mathbb{R}^{1 \times m \times n_3}$$

where $\mathbf{o}$ denote de zeros tube fiber of size $(1 \times 1 \times n_3)$ which all his entriesare equal to zeros, and $\mathcal{E}_m = [\mathbf{o}, \ldots, \mathbf{o}, \mathbf{e}] \in \mathbb{R}^{1 \times m \times n_3}$ where $\mathbf{e}$ the tube fiber such that $\text{unfold}(\mathbf{e}) = (1, 0, 0 \ldots, 0)^T$. The tensor $\tilde{\mathcal{H}}_m$ can be written as follows

$$\tilde{\mathcal{H}}_m = \begin{bmatrix} \mathcal{H}_m \\ \mathbf{h}_{m+1,m} * \mathcal{E}_m \end{bmatrix}$$

We can now state the following algebraic properties

Proposition 3.20. *Suppose that m steps of Algorithm 7 have been run. Then, the following statements hold:*

$$\begin{aligned}
\text{ffi} * \mathcal{V}_m &= \mathcal{V}_m * (\mathcal{H}_m \circledcirc \mathcal{I}_{ssn_3}) + \mathcal{V}_{m+1} * ((h_{m+1,m} * \mathcal{E}_m) \circledcirc \mathcal{I}_{ssn_3}),\\
\mathcal{V}_m^T \blacklozenge \mathcal{A} * \mathcal{V}_m &= \mathcal{H}_m,\\
\mathcal{A} * \mathcal{V}_m &= \mathcal{V}_{m+1} * (\tilde{\mathcal{H}}_m \circledcirc \mathcal{I}_{ssn_3}),\\
\mathcal{V}_{m+1}^T \blacklozenge \mathcal{A} * \mathcal{V}_m &= \tilde{\mathcal{H}}_m,\\
\mathcal{V}_m^T \blacklozenge \mathcal{V}_m &= \mathcal{I}_{mmn_3},
\end{aligned}$$

Proof. We give a proof only for the third relation, the other relations could be obtained in the same way. From Algorithm 7, we have $\text{ffi} * \mathcal{V}_j = \sum_{i=1}^{j+1} \mathbf{h}_{i,j} \div \mathcal{V}_i$. Using the fact that $\text{ffi} * \mathcal{V}_m = [\text{ffi} * \mathcal{V}_1, \ldots, \text{ffi} * \mathcal{V}_m]$, the j-th frontal slice of $\text{ffi} * \mathcal{V}_m$ is given by :

$$\begin{aligned}
(\text{ffi} * \mathcal{V}_m)_j = \text{ffi} * \mathcal{V}_j &= \sum_{i=1}^{j+1} \mathbf{h}_{i,j} \div \mathcal{V}_i\\
&= \sum_{i=1}^{j+1} \mathcal{V}_i * ((\mathbf{h}_{i,j}) \circledcirc \mathcal{I}_{ssn_3})\\
&= [\mathcal{V}_1, \ldots, \mathcal{V}_{j+1}] * \begin{bmatrix} \mathbf{h}_{1,j} \\ \vdots \\ \mathbf{h}_{j+1,j} \end{bmatrix} \circledcirc \mathcal{I}_{s,s,n_3}.
\end{aligned}$$

Let $\mathcal{H}_j = \begin{bmatrix} \mathbf{h}_{1,j} \\ \vdots \\ \mathbf{h}_{j+1,j} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{m+1 \times 1 \times n3}$ be the j-th lateral slice of of the Hessemberg tensor $\tilde{\mathcal{H}} = [\mathcal{H}_1, \ldots, \mathcal{H}_m]$. The we have

$$(\text{ffi} * \mathcal{V}_m)_j = [\mathcal{V}_1, \ldots, \mathcal{V}_{j+1}] * \mathcal{H}_j \circledcirc \mathcal{I}_{ssn_3} \qquad j = 1, \ldots, m.$$

and the result follows. □

3.3.3 The tensor Tubal-Global GMRES method

The tubal-global GMRES method is based on tubal-global Arnoldi process to build an orthonormal basis of the tensor tubal global Krylov subspace (3.7). First, we need to introduce a new T-l_2 norm defined in $\mathbb{R}^{m\times 1\times n_3}$.

Definition 3.21. (T-l_2-inner product) Let $\mathcal{X} \in \mathbb{R}^{m\times 1\times n_3}$, $\mathcal{Y} \in \mathbb{R}^{m\times 1\times n_3}$. Then the T-$l_2$ inner scalar product of $\mathcal{X}$ and $\mathcal{Y}$ is defined by

$$\langle \mathcal{X}, \mathcal{Y} \rangle_{T_{l_2}} = \frac{1}{n_3} \sum_{i=1}^{n_3} (\mathcal{X} \times_3 F_{n_3})^{(i)T} (\mathcal{Y} \times_3 F_{n_3})^{(i)}. \tag{3.9}$$

The associated T-l_2 norm is defined by

$$||\mathcal{Y}||_{T_{l_2}} = \frac{1}{\sqrt{n_3}} \left(\sum_{i=1}^{n_3} ||(\mathcal{Y} \times_3 F_{n_3})^{(i)}||_2^2 \right)^{\frac{1}{2}}, \tag{3.10}$$

where $||.||_2$ denote the usual vector l_2 norm.

Proposition 3.22. *Let* $\mathcal{U}, \mathcal{B}, \mathcal{C} \in \mathbb{R}^{m\times 1\times n_3}$ *and* $\alpha \in \mathbb{R}$, *then the T-l_2-inner product satisfies the following properties*

1. $\langle \mathcal{U}, \mathcal{B} + \mathcal{C} \rangle_{T_{l_2}} = \langle \mathcal{U}, \mathcal{B} \rangle_{T_{l_2}} + \langle \mathcal{U}, \mathcal{C} \rangle_{T_{l_2}}$.
2. $\langle \mathcal{U}, \alpha\mathcal{B} \rangle_{T_{l_2}} = \alpha \langle \mathcal{U}, \mathcal{B} \rangle_{T_{l_2}}$.
3. $\langle \mathcal{U}, \mathcal{X} * \mathcal{B} \rangle_{T_{l_2}} = \langle \mathcal{X}^T * \mathcal{U}, \mathcal{B} \rangle_{T_{l_2}}$, *for* $\mathcal{X} \in \mathbb{R}^{n_1\times n_1\times n_3}$.

Proof. The proofs come directly from the definitions of the T-l_2 inner scalar product.

□

Proposition 3.23. *Let* $\mathcal{Y} \in \mathbb{R}^{m\times 1\times n_3}$ *and* $\mathcal{V} \in \mathbb{R}^{n\times ms\times n_3}$ *such that* $\mathcal{V}^T \diamond \mathcal{V} = \mathcal{I}_{mmn_3}$. *Then*

$$||\mathcal{V} * (\mathcal{Y} \textcircled{2} \mathcal{I}_{ssn_3})||_F = ||\mathcal{Y}||_{T_{l_2}}.$$

Proof. We have :

$$\begin{aligned}
||V * (Y ② I_{ssn_3})||_F^2 &= \frac{1}{n_3}\Big(\sum_{i=1}^{n_3} ||(V \times_3 F_{n_3})^{(i)} (Y \times_3 F_{n_3})^{(i)} \otimes (I_{ssn_3} \times_3 F_{n_3})^{(i)} ||_F^2\Big) \\
&= \frac{1}{n_3}\Big(\sum_{i=1}^{n_3} (||(Y \times_3 F_{n_3})^{(i)}||_2^2)\Big) \\
&= ||Y||_{T,2}^2.
\end{aligned}$$

□

Next, we will see how to define the Tubal-Global GMRES. Consider the linear system of tensor equations

$$ffi * X = B \tag{3.11}$$

where $ffi \in R^{n \times n \times n3}$ assumed to be nonsingular, $B, X \in R^{n \times s \times n3}$ with $s \quad n$. If $n_3 = 1$ then the problem (3.11) reduces to a multiple linear systems of s equations . Let $X_0 \in R^{n \times s \times n3}$ be an arbitrary initial guess with the associated residual tensor $R_0 = B - ffi * X_0$. The aim of the tensor Tubal-Global GMRES method is to find, at some step m, an approximation X_m of the solution X^* of the problem ((3.11)) as follows

$$X_m - X_0 \in TK_m^g(ffi, R_0), \tag{3.12}$$

with

$$||R_m||_F = \min_{X - X_0 \in TK_m^g(ffi, R_0)} \{||B - ffi * X||_F\}. \tag{3.13}$$

Let $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m * (\mathcal{Y} \textcircled{2} \mathcal{I}_{ssn_3})$ with $\mathcal{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_m \end{bmatrix} \in \mathbb{R}^{m \times 1 \times n_3}$. Then the problem (3.13) is equivalent to

$$||\mathcal{R}_m||_F = \min_{\mathcal{Y} \in \mathbb{R}^{m \times 1 \times n_3}} ||\mathcal{R}_0 - \mathcal{V}_m * (\mathcal{Y} \textcircled{2} \mathcal{I}_{ssn_3})||_F. \tag{3.14}$$

Using the Proposition 3.23 and Proposition 3.12 and Step 2 of Algorithm 7, we get

$$\begin{aligned} ||\mathcal{R}_0 - \mathcal{V}_m * (\mathcal{Y} \textcircled{2} \mathcal{I}_{ssn_3})||_F &= ||\mathbf{r}_{1,1} \div \mathcal{V}_1 - (\mathcal{V}_{m+1} * (\tilde{\mathcal{H}}_m \textcircled{2} \mathcal{I}_{ssn_3})) * (\mathcal{Y} \textcircled{2} \mathcal{I}_{ssn_3}))||_F \\ &= ||\mathcal{V}_1 * (\mathbf{r}_{1,1} \textcircled{2} \mathcal{I}_{ssn_3}) - (\mathcal{V}_{m+1} * (\tilde{\mathcal{H}}_m \textcircled{2} \mathcal{I}_{ssn_3})) * (\mathcal{Y} \textcircled{2} \mathcal{I}_{ssn_3}))||_F \\ &= ||\mathcal{V}_{m+1} * (\mathcal{E}_1^{(m+1)} \textcircled{2} \mathcal{I}_{ssn_3}) * (\mathbf{r}_{1,1} \textcircled{2} \mathcal{I}_{ssn_3}) - (\mathcal{V}_{m+1} * ((\tilde{\mathcal{H}}_m * \mathcal{Y}) \\ &= ||\mathcal{V}_{m+1} * ((\mathcal{E}_1^{(m+1)} * \mathbf{r}_{1,1}) \textcircled{2} \mathcal{I}_{s,s,n_3}) - ((\tilde{\mathcal{H}}_m * \mathcal{Y}) \textcircled{2} \mathcal{I}_{ssn_3}))||_F \\ &= ||\mathcal{V}_{m+1} * (\mathcal{E}_1^{(m+1)} * \mathbf{r}_{1,1} - (\tilde{\mathcal{H}}_m * \mathcal{Y})) \textcircled{2} \mathcal{I}_{ssn_3})||_F \\ &= ||(\mathcal{E}_1^{(m+1)} * \mathbf{r}_{1,1} - (\tilde{\mathcal{H}}_m * \mathcal{Y}))||_{T_{I_2}}, \end{aligned}$$

where $\mathcal{E}_1^{(m+1)} = \begin{bmatrix} \mathbf{e} \\ \mathbf{o} \\ \vdots \\ \mathbf{o} \end{bmatrix} \in \mathbb{R}^{m+1 \times 1 \times n_3}$. Therefore, the tensor $\mathcal{Y}_m$ solving the minimisation problem (3.14) is given by

$$\mathcal{Y}_m = \arg \min_{\mathcal{Y} \in \mathbb{R}^{m \times 1 \times n_3}} ||\mathcal{E}_1^{(m+1)} * \mathbf{r}_{1,1} - (\tilde{\mathcal{H}}_m * \mathcal{Y})||_{T_{I_2}}. \tag{3.15}$$

The approximate solution is given as

$$\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m * (\mathcal{Y}_m \textcircled{2} \mathcal{I}_{ssn_3}) \tag{3.16}$$

3.3.4 Implementation of tensor Tubal-Global GMRES method

For the implementation of Algorithm 9, we use the tensor QR decomposition to solve the minimization problem (3.15). The Tensor QR decomposition is based on the application of the QR matrix decomposition to each sub-block of the obtained block diagonal matrix in the Fourier domain. In other word, for $\mathcal{F} \in \mathbb{R}^{n\times m\times n_3}$ we have $\mathcal{F} = \mathcal{Q} * \mathcal{R}$ where $\mathcal{Q}$ is an $n \times n \times n_3$ orthogonal tensor $\mathcal{Q}^T * \mathcal{Q} = \mathcal{Q} * \mathcal{Q}^T = \mathcal{I}_{nnn_3}$ and $\mathcal{R} \in \mathbb{R}^{n\times m\times n_3}$ triangular tensor.

The different steps are summarized as follows The following property will

Algorithm 8 Tensor T-QR decomposition

1. **Input.** $\mathcal{F} \in \mathbb{R}^{n\times m\times n_3}$.
2. **Output.** Orthogonal tensor $\mathcal{Q} \in \mathbb{R}^{n\times n\times n_3}$ and triangular tensor $\mathcal{R} \in \mathbb{R}^{m\times m\times n_3}$.
3. Set $\tilde{\mathcal{F}} = \mathbf{fft}(\mathcal{F}, [], 3)$
   (a) for $i = 1, \ldots, n_3$
   $[Q^{(i)}, R^{(i)}] = \mathrm{QR}(F^{(i)})$ (matrix QR decomposition)
   (b) End
4. $\mathcal{Q} = \mathbf{ifft}(\tilde{\mathcal{Q}}, [], 3)$, $\mathcal{R} = \mathbf{ifft}(\tilde{\mathcal{R}}, [], 3)$
5. End

be used to solve the problem (3.15)

Proposition 3.24. *Let* $\mathcal{Y} \in \mathbb{R}^{m\times 1\times n_3}$ *and* $\mathcal{Q} \in \mathbb{R}^{m\times m\times n_3}$ *such that* $\mathcal{Q}^T * \mathcal{Q} = \mathcal{Q} * \mathcal{Q}^T = \mathcal{I}_{mmn_3}$. *Then*

$$||\mathcal{Q} * \mathcal{Y}||_{T_{l_2}} = ||\mathcal{Y}||_{T_{l_2}}$$

Proof. The proof is a direct application of the T-l_2 norm. □

Now, we apply the T-QR decomposition to $\tilde{\mathcal{H}}_m$, and by using Proposition 3.24, we get

$$\begin{aligned} ||\mathcal{E}_1^{(m+1)} * \mathbf{r}_{1,1} - (\tilde{\mathcal{H}}_m * \mathcal{Y})||_{T_{l_2}} &= ||\mathcal{Q}_m^T * (\mathcal{E}_1^{(m+1)} * \mathbf{r}_{1,1} - (\tilde{\mathcal{H}}_m * \mathcal{Y}))||_{T_{l_2}} \\ &= ||\mathcal{Q}_m^T * (\mathcal{E}_1^{(m+1)} * \mathbf{r}_{1,1}) - \mathcal{Q}_m^T * (\tilde{\mathcal{H}}_m * \mathcal{Y})||_{T_{l_2}} \\ &= ||\tilde{\mathcal{G}}_m - \tilde{\mathcal{R}}_m * \mathcal{Y})||_{T_{l_2}} \end{aligned}$$

where $\tilde{G}_m = Q_m^T * (E_1^{(m+1)} * \mathbf{r}_{1,1})$ and $\tilde{R}_m = Q_m^T * \tilde{H}_m$.

Proposition 3.25. *Let $\tilde{G}_m$ and $\tilde{R}_m$ given as*

$$\tilde{R}_m = Q_m^T * \tilde{H}_m, \text{ and } \tilde{G}_m = Q_m^T * (E_{(m+1)1} * \mathbf{r}_{1,1}) = \begin{bmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{m+1} \end{bmatrix} \in \mathbb{R}^{m+1\times 1\times n_3}. \tag{3.17}$$

The solution $Y_m = arg \min_{Y\in \mathbb{R}^{m\times 1\times n_3}} ||E_1^{(m+1)} * \mathbf{r}_{1,1} - (\tilde{H}_m * Y))||_{T_{I_2}}$ is given by solving the following triangular tensor problem

$$R_m * Y_m = G_m, \tag{3.18}$$

where $R_m = \tilde{R}_m(1 : m, :, :)$ the tensor obtained by deleting the last horizontal slice of $\tilde{R}$ and $G_m = \begin{bmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_m \end{bmatrix} \in \mathbb{R}^{m\times 1\times n_3}$.

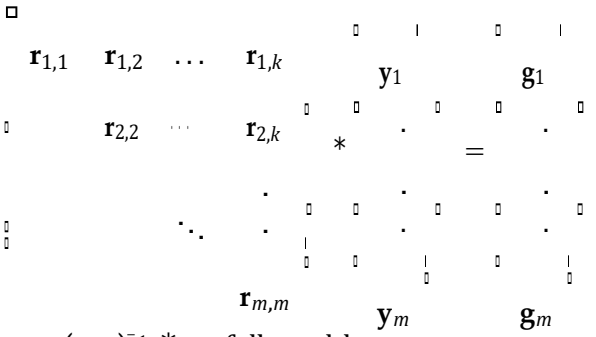
Proof.

$$\begin{aligned} ||E_1^{(m+1)} * \mathbf{r}_{1,1} - (\tilde{H}_m * Y)||_{T_{I_2}} &= ||Q_m^T * (E_1^{(m+1)} * \mathbf{r}_{1,1} - (\tilde{H}_m * Y))||_{T_{I_2}} \\ &= ||Q_m^T * (E_1^{(m+1)} * \mathbf{r}_{1,1}) - Q_m^T * (H_m * Y)||_{T_{I_2}} \\ &= ||\tilde{G}_m - \tilde{R}_m * Y)||_{T_{I_2}} \\ &= ||\mathbf{g}_{m+1}||^2_{T_{I_2}} + ||G_m - R_m * Y)||^2_{T_{I_2}} \end{aligned}$$

Assuming that R_m is invertible then Y_m solves triangular tensor problem (3.18). □

In the following, we introduce the tubal-back substitution method for solving the equation (3.18). This method follow the same steps of matrix back substitution , where the tube fibers (3-mode fibers), lateral slices and T-product play the role of scalars, vectors and matrix product respectively. In

other words, the solution to the triangular tensor system (3.18) -which can be wrriten as follows- :



begins with: $\mathbf{y}_m = (\mathbf{r}_{m,m})^{-1} * \mathbf{g}_m$ followed by :

$$\mathbf{y}_i = (\mathbf{r}_{i,i})^{-1} * (\mathbf{g}_i - \sum_{j=i+1}^{m} \mathbf{r}_{i,j} * \mathbf{y}_j) \quad \text{for } i = m-1, m-2, \ldots, 1.$$

where $(\mathbf{r}_{i,i})^{-1}$ stand for the inverse of the tube fiber $\mathbf{r}_{i,i}$ (Definition 3.17) for $i = m, m-1, \ldots, 1$

Algorithm 9 The Tensor Tubal-Global GMRES (m)

1. **Input.** $\mathcal{A} \in \mathbb{R}^{n \times n \times n3}$, $\mathcal{V}$, $\mathcal{B}$, $\mathcal{X}_0 \in \mathbb{R}^{n \times s \times n3}$, the maximum number of iteration $\text{Iter}_{\max}$, the restart parameter m and a tolerance $tol > 0$.
2. Compute $\mathcal{R}_0 = \mathcal{B} - \mathcal{A} * \mathcal{X}_0$.
3. For $k = 1, \ldots, \text{Iter}_{\max}$
   (a) Apply Algorithm 7 to compute $\mathcal{V}_m$ and $\tilde{\mathcal{H}}_m$.
   (b) Compute the T-QR decomposition of $\tilde{\mathcal{H}}_m$ using Algorithm 8 .
   (c) Compute $\mathcal{R}_m$ and $\mathcal{G}_m$ using the relations (3.17) .
   (d) Solve the triangular tensor system (3.18) to obtain $\mathcal{Y}_m$.
   (e) Compute $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m * (\mathcal{Y}_m \textcircled{2} \mathcal{I}_{ssn_3})$
4. If $||\mathcal{R}_m||_F < tol$, stop
5. else $\mathcal{X}_0 = \mathcal{X}_m$ and go to Step 2.
6. **Output.** $\mathcal{X}_m \in \mathbb{R}^{n \times s \times n3}$ the approximate solution of (3.11).

3.4 Tensor tubal-global Golub Kahan algorithm

Instead of using the tensor global Arnoldi process to generate a basis for the projected subspace, we can use the tensor tubal global Lanczos process. Here, we will use the tensor Golub Kahan algorithm related to the T-product. We notice that we already defined in [48] another version of the tensor Golub

Kahan algorithm for ill-posed problems with applications to color image processing.

Consider the linear system of tensor equations

$$\text{ffi} * \mathcal{X} = \mathcal{B} - \text{ffi} * \mathcal{X}_0 = \mathcal{R}_0 \tag{3.19}$$

where $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{B} \in \mathbb{R}^{n_1 \times s \times n_3}$ and $\mathcal{X}$, $\mathcal{X}_0 \in \mathbb{R}^{n_2 \times s \times n_3}$. The tensor tubal-global Golub Kahan bidiagonalization algorithm Algorithm 10 pro- duces a T-orthogonormal basis $\mathcal{U}_k = [\mathcal{U}_1, \ldots, \mathcal{U}_k, \mathcal{U}_{k+1}] \in \mathbb{R}^{n_1 \times (k+1)s \times n_3}$ and $\mathcal{V}_k = [\mathcal{V}_1, \ldots, \mathcal{V}_k] \in \mathbb{R}^{n_2 \times ks \times n_3}$ of the tensor Krylov subspace $\mathcal{TK}_g(\text{ffi} * \text{ffi}^T, \text{ffi} * \mathcal{R}_0)$ and $\mathcal{TK}_g(\text{ffi}^T * \text{ffi}, \text{ffi}^T * \mathcal{R}_0)$,respectively.

Algorithm 10 The Tensor Tubal-global Golub Kahan algorithm

1. **Inputs** The tensors ffi, $\mathcal{B}$, and an integer m.
2. Set $\mathcal{V}_0 = 0 \in \mathbb{R}^{n_1 \times s \times n_3}$ and $[\mathcal{U}_1, \mathbf{a}_1] = \text{Normalization}(\mathcal{R}_0)$.3.

For $j = 1, \ldots, k$

(a) $\tilde{\mathcal{V}} = \text{ffi} * \mathcal{U}_j - \mathbf{a}_j \div \mathcal{V}_{j-1}$

(b) Set $[\mathcal{V}_j, \mathbf{b}_j] = Normalization(\tilde{\mathcal{V}})$

(c) $\tilde{\mathcal{U}} = \text{ffi} * \mathcal{V}_j - \mathbf{b}_j \div \mathcal{U}_j$

(d) $[\mathcal{U}_{j+1}, \mathbf{a}_{j+1}] = \text{Normalization}(\tilde{\mathcal{U}})$

4. End for

Let $\tilde{\mathcal{C}}_k$ be the upper bidiagonal $(k+1) \times k \times n_3$ tensor

$$\tilde{\mathcal{C}}_k = \begin{bmatrix} \mathbf{b}_1 & & & \\ \mathbf{a}_2 & \mathbf{b}_2 & \ddots & \\ & \ddots & \ddots & \\ & & \mathbf{a}_k & \mathbf{b}_k \\ & & & \mathbf{a}_{k+1} \end{bmatrix}$$

and let $\mathcal{C}_k$ be the $(k \times k \times n_3)$ tensor obtain by deleting the last horizontal slice of $\tilde{\mathcal{C}}_k$ where the $\mathbf{a}_i$'s and $\mathbf{b}_i$'s are fibers. Then we have the following results

Proposition 3.26. *The tensors* $\mathcal{V}_k = [\mathcal{V}_1, \ldots, \mathcal{V}_k] \in \mathbb{R}^{n_2 \times ms \times n_3}$ *and* $\mathcal{U}_k = [\mathcal{U}_1, \ldots, \mathcal{U}_k, \mathcal{U}_{k+1}] \in \mathbb{R}^{n_1 \times (k+1)s \times n_3}$ *given by Algorithm10, have orthogonal lateral slices tensors* $\mathcal{V}_i \in \mathbb{R}^{n_2 \times s \times n_3}$ *and* $\mathcal{U}_i \in \mathbb{R}^{n_1 \times s \times n_3}$*, respectively, i.e.*

$$\langle \mathcal{V}_i, \mathcal{V}_j \rangle_T = \langle \mathcal{U}_i, \mathcal{U}_j \rangle_T = \begin{cases} \mathbf{e} & i = j \\ 0 & i \neq j. \end{cases}$$

Proof. This will be shown by induction on j. For $j = 1$, Algorithm10 shows that $\langle \mathcal{V}_1, \mathcal{V}_1 \rangle_T = \mathbf{e}$ and $\langle \mathcal{U}_1, \mathcal{U}_1 \rangle_T = \mathbf{e}$. Assume now that the result is true for some j. Then, from Algorithm 10 and using the results of Proposition 3.14, we conclude that

$$\langle \mathcal{V}_i, \mathcal{V}_j \rangle_T = \langle \mathcal{U}_i, \mathcal{U}_j \rangle_T = \begin{cases} \mathbf{e} & i = j \\ 0 & i \neq j \end{cases}, j = 1, \ldots, k.$$

□

Proposition 3.27. *The tensors produced by the tensor tubal-global Golub-Kahan algorithm satisfy the following relations*

$$\mathcal{AV}_k = \mathcal{U}_{k+1} * (\tilde{C}_k \textcircled{2} I_{ssn_3}) \tag{3.20}$$

$$= \mathcal{U}_k * (C_k \textcircled{2} I_{ssn_3}) + \mathcal{U}_{k+1} * ((a_{k+1} * E_k) \textcircled{2} I_{ssn_3}), \text{ and} \tag{3.21}$$

$$\mathcal{A}^T \mathcal{U}_k = \mathcal{V}_k * (C_k^T \textcircled{2} I_{ssn_3}) \tag{3.22}$$

$$\mathcal{R}_0 = \mathcal{U}_{k+1} * ((E_1^{(k+1)} * \mathbf{a}_1) \textcircled{2} I_{ssn_3}). \tag{3.23}$$

where $E_m = [\mathbf{o}, \ldots, \mathbf{o}, \mathbf{e}] \in \mathbb{R}^{1 \times k \times n_3}$, $E_1^{(k+1)} = \begin{bmatrix} \mathbf{e} \\ \mathbf{o} \\ \vdots \\ \mathbf{o} \end{bmatrix} \in \mathbb{R}^{k+1 \times 1 \times n_3}$, where $\mathbf{e}$ the tube fiber such that unfold$(\mathbf{e}) = (1, 0, 0 \ldots, 0)^T$ $\mathbf{o}$ denote de zeros tube fiber of size $(1 \times 1 \times n_3)$ which all his entries are equal to zeros.

Proof. The proofs come directly from the different steps of Algorithm 10. □

Next, we show how to apply theTTGK process to get approximate solutions to the tensor linear system (3.19).

Proposition 3.28. *Let* $X_k = V_k * (Y \textcircled{2} I_{ssn_3})$ *be an approximation of the solution* X^* *of the tensor linear system* (3.19) *where* $Y \in R^{k \times 1 \times n_3}$. *Then*

$$||R_0 - ffi * X_k||_F = ||E_1^{(k+1)} * a_1 - \tilde{C}_k * Y||_{T\ l_2} \tag{3.24}$$

Proof. Using (3.20, Proposition 3.23) and the fact that $R_0 = U_{k+1} * ((E_1^{(k+1)} * a_1) \textcircled{2} I_{ssn_3})$, we get

$$||R_0 - ffi * X_k||_F = ||U_{k+1} * ((E_1^{(k+1)} * a_1) \textcircled{2} I_{ssn\ _2}) - U_{k+1} * (\tilde{C}_m \textcircled{2} I_{ssn\ })_3 * (Y \textcircled{2} I_{ssn\ }))||_F$$

$$= ||U_{k+1} * ((E_1^{(k+1)} * a_1) \textcircled{2} I_{ssn_3}) - (\tilde{C}_k * Y)) \textcircled{2} I_{ssn\ _3}||_F = ||E_1^{(k+1)} * a_1 - (\tilde{C}_k$$

which ends the proof. □

The approximate solution X_k produced by this process is given by

$$X_k = V_k * (Y \textcircled{2} I_{ssn_3}),$$

where Y_k solves the low-order minimization problem

$$\min_{Y \in R^{k \times 1 \times n_3}} ||(E_1^{(k+1)} * a_1 - \tilde{C}_k * Y)||_{T\ l_2}.$$

The following algorithm summarizes the main steps to solve the linear tensor problem (3.11) using TTGK.

3.5 Numerical experiments

This section performs some numerical tests for the Tensor Tubal-Global GMRES and Tensor Tubal-Global Golub Kahan methods when solving solve the

Algorithm 11 Tensor Tubal-Global Golub Kahan (TTGK)

1. **Input:** $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{V}$, $\mathcal{B}$, $\mathcal{X}_0 \in \mathbb{R}^{n_2 \times s \times n_3}$, kmax the maximum number of iterations, and a tolerance *tol*.
2. **Output:** $\mathcal{X}_k \in \mathbb{R}^{n_2 \times s \times n_3}$ the approximate solution of the problem (3.19).
3. Compute $\mathcal{R}_0 = \mathcal{B} - \mathcal{A} * \mathcal{X}_0$.
4. For $k = 1, \ldots,$ kmax
   (a) Apply Algorithm 7 to compute $\mathcal{V}_k$ and $\tilde{\mathcal{H}}_k$.
   (b) Compute $\mathcal{Y}_k = \arg \min_{\mathcal{Y} \in \mathbb{R}^{k \times 1 \times n_3}} \|(\mathcal{E}_1^{(k+1)} * \mathbf{a}_1 - \tilde{\mathcal{C}}_k * \mathcal{Y})\|_{T_{I_2}}$ and the approximate solution $\mathcal{X}_k = \mathcal{X}_0 + \mathcal{V}_k * (\mathcal{Y}_k \textcircled{2} \mathcal{I}_{ssn_3})$.
5. If $\|\mathcal{R}_k\|_F < tol$, stop.
6. EndFor

linear tensor problem (3.11). All computations were carried out using the MATLAB R2018b environment with an Intel(R) Core i7-8550U CPU @1.80 GHz and processor 8 GB. The stopping criterion was

$$\frac{\|\mathcal{R}_k\|_{T_{I_2}}}{\|\mathcal{R}_0\|_{T_{I_2}}} < e,$$

where $e = 10^{-6}$ is a chosen tolerance and $\mathcal{R}_k$ the m-th residual associated to the approximate solution $\mathcal{X}_k$. In all the presented tables, we reported the obtained residual norms to achieve the desired convergence, the iteration number and the corresponding cpu-time.

We compared the required CPU-times (in seconds) to achieve the convergence for the methods:

1. **Algorithm 9**: TTGMRES(m) .
2. **T-GGMRES(m)[48]** : The restarted tensor global GMRES (T-GGMRES) [48].
3. **Algorithm 11**: TTGK .

3.5.1 Example1

The tensor $\mathcal{A}$ is constructed using n_3 frontal slices. In this example, the frontal slices are of size $n \times n$ and given as follows

$$A_i = \text{eye}(n) + \frac{i}{2\sqrt{n}} \text{rand}(n) \quad i = 1, \ldots, n_3$$

In this example, the right-hand side tensor $\mathcal{B}$ is constructed such that the exact solution $\mathcal{X}^*$ of the equation (3.11) is given by $\mathcal{X}^* = \text{ones}(n, s, n_3)$.. The integer m stand for the restarted parameter associated to the Tensor Tubal-Global GMRES(m) and Tensor Global GMRES T-GGMRES introduced in [48] . In Table 2.1, we reported the obtained relative residual norms, the

Table 3.1: *Results for Example 1. $e = 10^{-6}$, $s = 5$ and $n_3 = 4$*

Method	n	m	# its.	$\frac{\|R_k\|_{F_2}}{\|R_0\|_{F_2}}$	cpu-time in seconds
Algorithm 9	500	10	3	1.06×10^{-6}	**0.55**
Algorithm TTGkh	500	-	8	1.31×10^{-6}	1.45
Algorithm 9	1000	10	3	1.26×10^{-6}	**1.53**
T-GGMRES(m)[48]	1000	10	6	2.02×10^{-6}	1.92
Algorithm 9	1500	10	3	3.91×10^{-6}	**3.60**
Algorithm TTGkh	1500	-	6	1.57×10^{-6}	10.19

total number of required iterations to achieve the convergence and the corresponding cpu-times for Tubal-Global GMRES(m) and Tensor Tubal-Global Golub Kahan.

3.5.2 Example 2

In this example, the tensor $\mathcal{A}$ of size $m_0^2 \times m_0^2 \times n_3$, is the Laplacian tensor constructed by using the 7-point discretization of the three-dimensional Poisson equation (3.25). The tensor $\mathcal{B}$ was constructed such that $\mathcal{B} = \mathcal{A} * \mathcal{X}^*$ where $\mathcal{X}^* = \text{ones}(m_0^2, s, n_3)$. The integer m stand for the restarted parameter associated to the Tensor Tubal-Global GMRES(m) and Tensor Global GMRES T-GGMRES(m) introduced in [**?**].

The three-dimensional Poisson equation is given by

$$\begin{cases} -\nabla^2 v = f, & \Omega = \{(x,y,z) = 0 < x,y,z < 1\}, \\ v = 0 & \text{on } \partial\Omega, \end{cases} \tag{3.25}$$

with

$$\nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}.$$

The mesh step size is given by : $\Delta x = \Delta y = \Delta z = h = \frac{1}{(m_0+1)}$ where Δx, Δy, Δz the step sizes in the x-direction, y-direction and z-direction, respectively. The difference formula obtain by the standard central difference approximation is given by

$$6v_{ijk} - v_{i-1jk} - v_{i+1jk} - v_{ij-1k} - v_{ij+1k} - v_{ijk-1} - v_{ijk+1} = h^3 f_{ijk}. \tag{3.26}$$

The Laplacian tensor ffi can be obtained from the central difference approximations (3.26) in several forms. Here, ffi can be expressed as a third order tensor as follow

$$\text{unfold}(\text{ffi}) = [A_1\ A_2\ \ldots\ A_{n_3}]^T,$$

where

$$-A_{i-1} = A_{i+1} = \frac{1}{h^3}\begin{bmatrix} 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & & 1 & 0 \\ 0 & \ldots & 0 & 0 & 0 \end{bmatrix}; \text{ and } A_i = -\frac{1}{h^3}\begin{bmatrix} 0 & -1 & \ldots & 0 & 0 \\ -1 & 6 & -1 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & -1 & 6 & -1 \\ 0 & \ldots & 0 & -1 & 0 \end{bmatrix}$$

Table 3.2 reports on the obtained relative residual norms and the corresponding cpu-times to obtain the desired convergence. As can be seen from this table, the Tensor Tubal-Global GMRES method gives good results within small cpu-times as compared to Tensor Tubal-Global Golub Kahan.

Table 3.2: *Results for Example 2.* $n_3 = m_0^2$, $s = 3$ *and* $e = 10^{-6}$

method	m	Size ($m_0^2 \times m_0^2 \times m_0^2$)	# Iter.	$\frac{\|R_k\|_{T_{I_2}}}{\|R_0\|_{T_{I_2}}}$	cpu-time (in seconds)
Algorithm 9	10	$100 \times 100 \times 100$	2	9.73×10^{-6}	**0.52**
T-GGMRES(m)[48]	10	$100 \times 100 \times 100$	8	3.85×10^{-6}	0.74
Algorithm 11	–	$100 \times 100 \times 100$	13	2.46×10^{-6}	3.33
Algorithm 9	10	$144 \times 144 \times 144$	3	9.73×10^{-6}	**1.09**
T-GGMRES(m)[48]	10	$144 \times 144 \times 144$	7	1.86×10^{-6}	1.20
Algorithm 11	–	$144 \times 144 \times 144$	12	1.86×10^{-6}	7.54
Algorithm 9	15	$225 \times 225 \times 225$	2	1.51×10^{-6}	**2.55**
T-GGMRES(m)[48]	15	$225 \times 225 \times 225$	6	3.85×10^{-6}	5.50
Algorithm 11	–	$225 \times 225 \times 225$	13	4.99×10^{-6}	24.02

3.6 Conlusion

In this chapter, we presented some new Krylov subspace methods using the T-product and some new other tensor products. We gave new algebraic properties of these products that allowed us to build new tensor Krylov-subspace based algorithms for solving general linear tensor equations. We focussed on the tubal-global GMRES and the tubal-global Golub-Kahan algorithms. Some numerical tests on simple examples are also reported.

Chapter 4

Tensor GMRES and Golub-Kahan Bidiagonalization methods via the Einstein product with applications to image and video processing

In the present chapter, we are interested in developing iterative Krylov subspace methods in tensor structure to solve a class of multilinear systems via the Einstein product. In particular, we develop global variants of the GMRES and Golub–Kahan bidiagonalization processes in tensor framework. We further consider the case that mentioned equation may be possibly correspond to a discrete ill-posed problem with applications in color image and video processing. We will also introduce new block version of the proposed tensor processes. Numerical simulation shows that the proposed methods are promising and give suitable results.

4.1 Introduction

We are interested in approximating the solution of following linear tensor equation

$$\min \|\Phi(\mathcal{X}) - \mathcal{C}\|_F. \tag{4.1}$$

where Φ is a linear tensor mapping with $\mathcal{C} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$ and $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$ is an unknown tensor to be determined. The linear map- ping Φ could be

$$\varphi(\mathcal{X}) = \mathcal{A} *_N \mathcal{X}, \text{ or } \varphi(\mathcal{X}) = \mathcal{A} *_N \mathcal{X} *_M \mathcal{B},$$

where $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times I_1 \times \ldots \times I_N}$, $\mathcal{B} \in \mathbb{R}^{J_1 \times \ldots \times J_M \times J_1 \times \ldots \times J_M}$, $*_N$ and $*_M$ is the Einstein product.

The purpuse is to take advantage of the multidimensional structure to build rapid and robust methods for solving the related problems. For an extensive literature on tensors, one can see for example the good papers in [90, 91]. Over the last years, many specialized methods for solving tensor problems of type (4.1) have been developed, for tensor forms of the Arnoldi and Lanczos processes for well-posed problems. Huang et al. [76] pointed out that tensor equations of the form (4.1) appear in continuum physics, engineering, isotropic and anisotropic elastic models. Multilinear systems of the form (4.1) may also arise from discretization of the high-dimensional Poisson problem using finite difference approximations [76].

In the current chapter, we are interested in developing robust and fast iterative Krylov subspace methods via the Einstein product to solve ill-posed problems originating from color image and video processing applications. Standard and global Krylov subspace methods are suitable when dealing with grayscale images, e.g, [17, 18, 35], while Krylov subspace methods can handle similar applications when the blurring linear operator can be decomposed into a Kronecker product of two matrices; see [17, 18]. However, much

work has to be done to numerically solve problems related to multi channel images (e.g. color images, hyper-spectral images and videos). We show that modelling these problems in the form of tensor equations (4.1) make it possible to develop tensor iterative Krylov subspace methods more appealing and allows to significantly reduce the overall computational cost. We find that solving this kind of problems with the Einstein product can keep the computational cost low and the iterative methods based on this product are better than the matrix-based iterative methods. We show that tensor iterative methods based on the Einstein product have several advantages: modelling color image and video processing problems with higher order tensor representation keeps computational cost and memory requirement low and a one to- one correspondence between the solution and the PSF array facilitate the integration of complicated boundary conditions.

The remainder of paper is organized as follows. In Section 4.2 we present some symbols and notations used throughout the paper. Section 4.3 includes reviewing the adaptation of Tikhonov regularization for tensor equation (4.1). Then we propose tensor GMRES and Global Golub–Kahan methods via the Einstein product in conjunction with Tikhonov regularization. On the basis of Point Spread Function (PSF), we propose, in Section 4.4, a tensor formulation in the form of (4.1) that describes the blurring of color image and video processing. In Section 4.5 we present the block tensor Arnoldi process with some properties, we developed the block tensor GMRES and the block tensor Golub Kahan algorithms, respectively. Numerical examples are reported on restoring blurred and noisy color images and videos. Concluding remarks can be found in the last section.

4.2 Definitions and Notations

In [13], the $\boxtimes^N$ product between the N-mode tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1} \times \tilde{I}_N}$ has been defined as an $I_N \times \tilde{I}_N$ matrix whose (i, j)- th entry is

$$[\mathcal{X} \boxtimes^N \mathcal{Y}]_{ij} = \text{tr}(\mathcal{X}_{:\ldots:i} \boxtimes^{N-1} \mathcal{Y}_{:\ldots:j}), \qquad N = 3, 4, \ldots,$$

where

$$\mathcal{X} \boxtimes^2 \mathcal{Y} = \mathcal{X}^T \mathcal{Y}, \qquad \mathcal{X} \in \mathbb{R}^{I_1 \times I_2}, \mathcal{Y} \in \mathbb{R}^{I_1 \times \tilde{I}_2}.$$

Basically, the product $\mathcal{X} \boxtimes^N \mathcal{Y}$ is the contracted product of the N-mode tensors $\mathcal{X}$ and $\mathcal{Y}$ along the first $N-1$ modes.

It is immediate to see that for $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, we have

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \text{tr}(\mathcal{X} \boxtimes^N \mathcal{Y}), \qquad N = 2, 3, \ldots,$$

and

$$\|\mathcal{X}\|^2 = \text{tr}(\mathcal{X} \boxtimes^N \mathcal{X}) = \mathcal{X} \boxtimes^{(N+1)} \mathcal{X},$$

.

We end the current subsection by recalling the following useful proposition from [13].

Proposition 4.1. *Suppose that* $\mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times m}$ *is an* $(N+1)$*-mode tensor with the column tensors* $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ *and* $z = (z_1, z_2, \ldots, z_m)^T \in \mathbb{R}^m$. *For an arbitrary* $(N+1)$*-mode tensor* $\mathcal{A}$ *with* N*-mode column tensors* $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m$, *the following statement holds*

$$\mathcal{A} \boxtimes^{(N+1)} (\mathcal{B} \bar{\times}_{N+1} z) = (\mathcal{A} \boxtimes^{(N+1)} \mathcal{B}) z. \tag{4.2}$$

4.3 Krylov subspace methods via Einstein product and Tikhonov regularization

In this section, we define the tensor global Arnoldi generalizing the classical one [79] and propose iterative methods based on Global Arnoldi and Global Global Golub–Kahan bidiagonalization (GGKB) combined with Tikhonov regularization that will be applied to the restoration of color images and videos from an available blur- and noise-contaminated versions.

4.3.1 Tikhonov regularization

Many applications require the solution of several ill-conditioning systems of equations of the form (4.1) with a right hand side contaminated by an additive error,

$$\Phi(\mathcal{X}) = \mathcal{C} + \mathcal{E}, \tag{4.3}$$

where $\mathcal{E}$ is the tensor of error terms that may stem from measurement and discretization errors. An ill-posed tensor equation may appear in color image restoration, video restoration, and when solving some partial differential equations in several space dimensions. In order to diminish the effect of the noise in the data, we replace the original problem by a more regularized one. The most popular regularization method is due to Tikhonov [130] for which the original problem is replaced by the new one

$$\mathcal{X}_\mu = \operatorname*{argmin}_{\mathcal{X}} \|\Phi(\mathcal{X}) - \mathcal{C}\|_F^2 + \mu\|\mathcal{X}\|_F^2 \ . \tag{4.4}$$

The choice of μ affects how sensitive $\mathcal{X}_\mu$ is to the error $\mathcal{E}$ in the contaminated right-hand side. For the vector and the matrix cases, many techniques for choosing a suitable value of μ have been analyzed and illustrated in the literature; see, e.g., [138].

4.3.2 The Global GMRES method via Einstein product

Let Φ be a linear mapping from the tensor space $\mathbb{R}^{I_1\times\ldots\times I_N\times J_1\times\ldots\times J_M}$ onto $\mathbb{R}^{I_1\times\ldots\times I_N\times J_1\times\ldots\times J_M}$ The m-th tensor Krylov subspace is defined by

$$\mathbb{K}_m(\Phi, \mathcal{V}) = span\{\mathcal{V}, \Phi(\mathcal{V}), \ldots, \Phi^{m-1}(\mathcal{V})\} \subset \mathbb{R}^{I_1\times\ldots\times I_N\times J_1\times\ldots\times J_M}. \tag{4.5}$$

where $\Phi^i(\mathcal{V}) = \Phi(\Phi^{i-1}(\mathcal{V}))$. The algorithm for constructing an orthonormal basis of (4.5) is given as follows.

Algorithm 12 Tensor Global Arnoldi process via Einstein product

1. Inputs: A linear mapping Φ, and a tensor $\mathcal{V} \in \mathbb{R}^{I_1\times\ldots\times I_N\times J_1\times\ldots\times J_M}$ and an integer m.
2. Set $\beta = \|\mathcal{V}\|_F$ and $\mathcal{V}_1 = \mathcal{V}/\beta$.
3. For $j = 1, \ldots, m$
   (a) $\mathcal{W} = \Phi(\mathcal{V}_j)$.
   (b) for $i = 1, \ldots, j$.
   * $h_{ij} = \langle \mathcal{V}_i, \mathcal{W} \rangle$,
   * $\mathcal{W} = \mathcal{W} - h_{ij}\mathcal{V}_i$
   (c) endfor
   (d) $h_{j+1,j} = \|\mathcal{W}\|_F$. If $h_{j+1,j} = 0$, stop; else
   (e) $\mathcal{V}_{j+1} = \mathcal{W}/h_{j+1j}$.
4. EndFor

Let $\tilde{H}_m$ be the upper $(m+1 \times m)$ Hessenberg matrix whose entries are the h_{ij} from Algorithm 12 and let H_m be the matrix obtained from $\tilde{H}_m$ by deleting the last row. Then, it is not difficult to verify that the $\mathcal{V}_i$'s obtained from Algorithm 12 form an orthonormal basis of the tensor Krylov subspace $\mathbb{K}_m(\Phi, \mathcal{V})$. Analogous to [13, 79], we can prove the following proposition.

Proposition 4.2. *Let* $\mathcal{V}$ *be the* $(M+N+1)$*-mode tensor with frontal slices* $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_m$ *and* $\mathcal{W}_m$ *be the* $(M+N+1)$*-mode tensor with frontal slices* $\Phi(\mathcal{V}_1), \ldots, \Phi(\mathcal{V}_m)$*. Then*

$$\begin{aligned}\mathcal{W}_m &= \mathcal{V}_{m+1} \times_{(M+N+1)} \tilde{H}_m^T \\ &= \mathcal{V}_m \times_{(M+N+1)} H_m^T + h_{m+1,m}\, \mathcal{L}_m \times_{(M+N+1)} E_m,\end{aligned} \tag{4.6}$$

where $E_m = [0, 0, \ldots, 0, e_m]$*;* e_m *is the m-th column of the identity matrix* I_m *and* $\mathcal{L}_m$ *is the* $(M + N + 1)$*-mode whose frontal slices are all zero except that last one being equal to* $\mathcal{V}_{m+1}$*.*

Let $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$, Φ a linear tensor mapping, $\mathcal{C} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$ and consider the linear system of tensor equations

$$\Phi(\mathcal{X}) = \mathcal{C}. \tag{4.7}$$

Using Algorithm 12, we can propose the tensor global GMRES method to solve the problem (4.7). As for the global GMRES, we seek for an approximate solution $\mathcal{X}_m$, starting from $\mathcal{X}_0$ such that $\mathcal{X}_m \in \mathcal{X}_0 + \mathcal{K}_m(\Phi, \mathcal{V})$ and by solving the minimization problem

$$\|\mathcal{R}_m\|_F = \min_{\mathcal{X} \in \mathcal{X}_0 + \mathcal{K}_m(\Phi, \mathcal{V})} \|\mathcal{C} - \Phi(\mathcal{X})\|_F. \tag{4.8}$$

where $\mathcal{R}_m = \mathcal{C} - \Phi(\mathcal{X}_m)$. Assume that m steps of Algorithm 12 has been performed. Given an initial guess $\mathcal{X}_0$, we set

$$\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m \bar{\times}_{(M+N+1)} y_m, \tag{4.9}$$

which gives $\mathcal{R}_m = \mathcal{R}_0 - \mathcal{W}_m \bar{\times}_{(M+N+1)} y_m$. Using the relations (4.6) from Proposition 4.2, it follows that

$$\begin{aligned}
\|\mathcal{C} - \Phi(\mathcal{X}_m)\|_F &= \|\mathcal{V}_m \boxtimes^{(M+N+1)} (\mathcal{C} - \Phi(\mathcal{X}_m))\|_2 \\
&= \|\mathcal{V}_m \boxtimes^{(M+N+1)} (\mathcal{R}_0 - \mathcal{W}_m \bar{\times}_{(M+N+1)} y_m)\|_2 \\
&= \|\beta e_1^{m+1} - \mathcal{V}_m \boxtimes^{(M+N+1)} (\mathcal{W}_m \bar{\times}_{(M+N+1)} y_m)\|_2 \\
&= \|\beta e_1^{m+1} - (\mathcal{V}_m \boxtimes^{(M+N+1)} \mathcal{W}_m) y_m)\|_2.
\end{aligned}$$

Therefore, y_m is determined as follows:

$$y^m = \arg\min_y \|\beta e_1^{m+1} - \tilde{H}^m y\|_2. \tag{4.10}$$

The relations (4.9) and (4.10) define the T-GGMRES. Setting $\mathcal{X}_0 = 0$ and using the relations (4.8), (4.9) and (4.10) it follows that instead of solving the problem (4.4) we can consider the following low dimensional Tikhonov regularization problem

$$\|\beta e_1^{m+1} - \tilde{H}_m y\|_2^2 + \mu \|y\|_2^2. \tag{4.11}$$

The solution of the problem (4.11) is given by

$$y_{m,\mu} = \arg\min_y \left\| \begin{pmatrix} \tilde{H}_m \\ \sqrt{\mu} I \end{pmatrix} y - \begin{pmatrix} \beta e^{m+1} \\ 0 \end{pmatrix} \right\|_2. \tag{4.12}$$

The minimizer $y_{m,\mu}$ of the problem (4.12) is computed as the solution of the linear system of equations

$$\tilde{H}_{m,\mu} y = \tilde{H}_m^T \beta e_1^{m+1} \tag{4.13}$$

where $\tilde{H}_{m,\mu} = (\tilde{H}_m^T \tilde{H}_m + \mu I)$.

Notice that the Tikhonov problem (4.11) is a matrix one with small dimension as m is generally small. Hence it can be solved by some techniques such as the GCV method [60] or the L-curve criterion [35, 67, 69]. An appropriate selection of the regularization parameter μ is important in Tikhonov regularization. Here we can use the generalized cross-validation (GCV) method [60, 138]. For this method, the regularization parameter is chosen to minimize the GCV function

$$GCV(\mu) = \frac{\|\tilde{H}_m y_{m,\mu} - \beta e_1^{m+1}\|_2^2}{[tr(I - \tilde{H}_m \tilde{H}_{m,\mu}^{-1} \tilde{H}_m^T)]^2} = \frac{\|(I - \tilde{H}_m \tilde{H}_{m,\mu}^{-1} \tilde{H}_m^T) \beta e_1^{m+1}\|_2^2}{[tr(I - H_m H_{m,\mu}^{-1} \tilde{H}_m^T)]^2}$$

where $\tilde{H}_{m,\mu} = (\tilde{H}_m^T \tilde{H}_m + \mu I)$ and $y_{m,\mu}$ is the solution of (4.13). As the projected problem we are dealing with is of small size, we cane use the SVD decomposition of $\tilde{H}_m$ to obtain a more simple and computable expression of

$GCV(\mu)$. Consider the SVD decomposition of $\tilde{H}_m = U\Sigma V^T$. Then the GCV function could be expressed as (see [138])

$$GCV(\mu) = \frac{\sum_{i=1}^{m} \left(\frac{\tilde{g}_i}{\sigma_i^2 + \mu} \right)^2}{\sum_{i=1}^{m} \frac{1}{\sigma_i^2 + \mu}^2}, \tag{4.14}$$

where σ_i is the ith singular value of the matrix $\tilde{H}_m$ and $\tilde{g} = \beta_1 U^{T} e_1^{m+1}$.

In the practical implementation, it's more convenient to use a restarted version of the global GMRES. As the number of outer iterations increases, it is possible to compute the m-th residual without forming the solution. This is described in the following theorem.

Proposition 4.3. *At step m, the residual* $\mathcal{R}_m = \mathcal{C} - \Phi(\mathcal{X}_m)$ *produced by the tensor global GMRES method for solving (4.1) has the following expression*

$$\mathcal{R}_m = \mathcal{V}_{m+1} \bar{\times}_{(M+N+1)} (\gamma_{m+1} Q_m e_{m+1}), \tag{4.15}$$

where Q_m *is the unitary matrix obtained by QR decomposition of the upper Hessenberg matrix* $\tilde{H}_m$ *and* γ_{m+1} *is the last component of the vector* $\beta \tilde{Q}_m e_{m+1}$ *in which* $\beta = \|\mathcal{R}_0\|_F$ *and* $e_1 \in \mathbb{R}^{l}$ *is the last column of identity matrix. Furthermore,*

$$\|\mathcal{R}_m\|_F = |\gamma_{m+1}| \tag{4.16}$$

Proof. At step m, the residual $\mathcal{R}_m = \mathcal{R}_0 - \mathcal{W}_m \bar{\times}_{(M+N+1)} y_m$ can be expressed as

$$\begin{aligned} \mathcal{R}_m &= \mathcal{R}_0 - (\mathcal{V}_{m+1} \times_{(M+N+1)} \tilde{H}_m^T) \bar{\times}_{(M+N+1)} y_m \\ &= \mathcal{R}_0 - \mathcal{V}_{m+1} \bar{\times}_{(M+N+1)} (\tilde{H_m} y_m) \end{aligned}$$

by considering the QR decomposition $\tilde{H}_m = Q_m \tilde{U}_m$ of the $(m+1) \times m$ matrix $\tilde{H}_m$, we get

$$\mathcal{R}_m = \mathcal{R}_0 - \mathcal{V}_{m+1} \bar{\times}_{(M+N+1)} (Q_m \tilde{U_m} y_m).$$

Straightforward computations show that

$$\begin{aligned} \|\mathcal{R}_m\|_F^2 &= \|\mathcal{R}_0 - \mathcal{V}_{m+1} \bar{\times}_{(M+N+1)} (Q_m \tilde{U}_m y_m)\|_F^2 \\ &= \|\mathcal{V}_m \boxtimes^{(M+N+1)} (\mathcal{R}_0 - \mathcal{V}_{m+1} \bar{\times}_{(M+N+1)} (Q_m \tilde{U}_m y_m))\|_2^2 \\ &= \|Q_m (Q_m^T \beta e_1^{m+1} - \tilde{U}_m y_m)\|_2^2 \\ &= \|Q_m^T \beta e_1^{m+1} - \tilde{U}_m y_m\|_2^2 \\ &= \|z_m - \tilde{U}_m y_m\|_2^2 + |\gamma_{m+1}|^2 \end{aligned}$$

where z_m denotes vector obtained by deleting the last component of $Q_m^T \beta e_1^{m+1}$.

Since y_m solves problem (4.10), it follows that y_m is the solution of $\tilde{U}_m y_m = z_m$, i.e.,

$$\|z_m - \tilde{U}_m y_m\|_2 = 0.$$

Note that $\mathcal{R}_m$ can be written in the following form

$$\begin{aligned} \mathcal{R}_m &= \beta \mathcal{V}_{m+1} \bar{\times}_{(M+N+1)} e_1^{m+1} - \mathcal{V}_{m+1} \bar{\times}_{(M+N+1)} (\tilde{H}_m y_m) \\ &= \mathcal{V}_{m+1} \bar{\times}_{(M+N+1)} (\beta e_1^{m+1} - \tilde{H}_m y_m) \\ &= \mathcal{V}_{m+1} \bar{\times}_{(M+N+1)} (Q_m (Q_m^T \beta e_1^{m+1} - \tilde{U} y_m)) \\ &= \mathcal{V}_{m+1} \bar{\times}_{(M+N+1)} (Q_m \gamma_{m+1} e_{m+1}). \end{aligned}$$

Now the result follows immediately from the above computations. □

The tensor global GMRES algorithm with the Tikhonov regularization for the projected problem, is summarized as follows:

Algorithm 13 Global GMRES method via Einstein product

1. **Inputs** A linear mapping Φ, initial guess $\mathcal{X}_0$, a tolerance ε, number of iterations between restarts m and $Maxit$ a maximum number of outer iterations.
2. Compute $\mathcal{R}_0 = \mathcal{C} - \Phi(\mathcal{X}_0)$, set $\mathcal{V} = \mathcal{R}_0$ and $k = 0$
3. Determine the orthonormal frontal slices $\mathcal{V}_1, \ldots, \mathcal{V}_m$ of $\mathcal{V}_m$, and the upper Hessenberg matrix $\tilde{H}_m$ by applying Algorithm 12 to the pair $(\Phi, \mathcal{V})$.
4. Determine μ_k as the parameter minimizing the GCV function given by (4.14)
5. Determine y_m as the solution of low-dimensional Tikhonov regularization problem (6.18) and set $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m \bar{\times}_{(M+N+1)} y_m$
6. If $\|\mathcal{R}_m\|_F < \varepsilon$ or $k > Maxit$; Stop
   else: set $\mathcal{X}_0 = \mathcal{X}_m$, $k = k + 1$, Go to 2

4.3.3 The tensor Golub–Kahan method via the Einstein product

Instead of projecting orthogonally onto a Krylov subspace and using GMRES method for solving the obtained minimisation problem, one can apply oblique projection schemes based on biorthogonal bases for $\mathcal{K}_m(\Phi, \mathcal{V})$ and $\mathcal{K}_m(\Phi^T, \mathcal{W})$; see [84] for instance.

Here, we exploit the tensor Golub–Kahan algorithm via the Einstein product. It should be commented here that the Golub–Kahan algorithm has been already examined for solving ill-posed Sylvester and Lyapunov tensor equations with applications to color image restoration [13] using the classical n-mode product.

Let Φ a linear tensor mapping and $\mathcal{C} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$ be given tensor.

Then, the GGKB algorithm is summarized as follows.

Algorithm 14 Global Golub–Kahan algorithm via Einstein product

1. **Inputs** A linear mapping Φ and an integer l.
2. Set $\sigma_1 = \|\mathcal{C}\|_F$, $\mathcal{U}_1 = \mathcal{C}/\sigma_1$ and $\mathcal{V}_0 = 0$
3. For $j = 1, 2, \ldots, l$ Do
4. $\tilde{\mathcal{V}} = \Phi^T(\mathcal{U}_j) - \sigma_j \mathcal{V}_{j-1}$
5. $\rho_j = \|\tilde{\mathcal{V}}\|_F$ if $\rho_j = 0$ stop, else
6. $\mathcal{V}_j = \tilde{\mathcal{V}}/\rho_j$
7. $\tilde{\mathcal{U}} = \Phi(\mathcal{V}_j) - \rho_j \mathcal{U}_j$
8. $\sigma_{j+1} = \|\tilde{\mathcal{U}}\|_F$
9. if $\rho_j = 0$ stop, else
10. $\mathcal{U}_{j+1} = \tilde{\mathcal{U}}/\sigma_{j+1}$
11. EndDo

Assume that l steps of the GGKB process have been performed, we form the lower bidiagonal matrix $C_l \in \mathrm{R}^{l\times l}$

$$C_l = \begin{bmatrix} \rho_1 & & & \\ \sigma_2 & \rho_2 & & \\ & \ddots & \ddots & \\ & & \sigma_{l-1} & \rho_{l-1} \\ & & & \sigma_l \quad \rho_l \end{bmatrix}$$

and

$$\tilde{C}_l = \begin{bmatrix} C_l \\ \sigma_{l+1} e_l^T \end{bmatrix} \in \mathrm{R}^{(l+1)\times l}.$$

Proposition 4.4. *Assume that l have performed and all non-trivial entries of the matrix $\tilde{C}_l$ are positive. Let V_τ and U_τ be $(M + N + 1)$-mode tensors whose frontal slices are given by V_j and U_j for $j = 1, 2, \ldots, \tau$, respectively. Furthermore, suppose that W_τ and W_τ^* are $(M + N + 1)$-mode tensors having frontal slices $\Phi(\mathrm{V}_j)$ and $\Phi^T(\mathrm{U}_j)$ for $j = 1, 2, \ldots, \tau$, respectively. The following relations hold:*

$$\tilde{\mathrm{W}}_l = \mathrm{V}_{l+1} \times_{(M+N+1)} \mathrm{U}_l^T \times C_l, \tag{4.17}$$

$$\mathrm{W}_l^* = \mathrm{V}_l \times_{(M+N+1)} C_l^T. \tag{4.18}$$

Proof. From Lines 7 and 10 of Algorithm 14, we have

$$\Phi(\mathrm{V}_j) = \rho_j \mathrm{U}_j + \sigma_{j+1}\mathrm{U}_{j+1} \qquad j = 1, 2 \ldots, l$$

which conclude (4.17) from definition of n-mode product. Similarly, Eq. (4.18) follows from Lines 4 and 6 of Algorithm 14. □

Here, we apply the following Tikhonov regularization approach and solve the new problem

$$\min_{\mathrm{X}} \|\Phi(\mathrm{X}) - \mathrm{C}\|_F^2 + \mu^{-1}\|\mathrm{C}\|_F^2, \tag{4.19}$$

We comment on the use of μ^{-1} in (4.19) instead of μ below. As for the iterative tensor Global GMRES method discussed in the previous subsection, the computation of an accurate approximation $\mathcal{X}_\mu$ requires that a suitable value of the regularization parameter be used. In this subsection, we use the discrepancy principle to determine a suitable regularization parameter assuming that an approximation of the norm of additive error is available, i.e., we have a bound ε for $\|\mathcal{E}\|_F$. This priori information suggests that μ has to be determined such that,

$$\|\mathcal{C} - \mathbf{\Phi}(\mathcal{X}_\mu)\|_F = \eta e, \tag{4.20}$$

where $\eta > 1$ is the safety factor for the discrepancy principle. A zero-finding method can be used to solve (4.20) in order to find a suitable regularization parameter which also implies that $\|\mathcal{C} - \mathbf{\Phi}(\mathcal{X}_\mu)\|_F$ has to be evaluated for several μ-values. When the tensor $\mathcal{A}$ is of moderate size, the quantity $\|\mathcal{C} - \mathbf{\Phi}(\mathcal{X}_\mu)\|_F$ can be easily evaluated. This computation becomes expensive when $\mathcal{A}$ is a large tensor, which means that its evaluation by a zero-finding method can be very difficult and computationally expensive. In what follows, it is shown that this difficulty can be remedied by using a connection between the GGKB and Gauss-type quadrature rules. This connection provides approximations of moderate sizes to the quantity $\|\mathcal{C} - \mathbf{\Phi}(\mathcal{X}_\mu)\|_F$ and therefore gives a solution method to inexpensively solve (4.20) by evaluating these small quantities; see [17, 18] for discussion on this method.

Let us consider the following functions of μ,

$$\varphi(\mu) = \|\ddot{\mathcal{C}} - \mathbf{\Phi}(\mathcal{X}_\mu)\ddot{}\|_F^2 \tag{4.21}$$

$$G_k f_\mu = \|\mathcal{C}\|^2 e_1^T (\mu C_k C_k^T + I_k)^{-2} e_1, \tag{4.22}$$

$$R_{k+1} f_\mu = \|\mathcal{C}\|^2 e_1^T (\mu \hat{C}_k \hat{C}_k^T + I_{k+1})^{-2} e_1; \tag{4.23}$$

$\mathrm{G}_l f$ and $\mathrm{R}_{l+1} f_\mu$ are pairs of Gauss and Gauss-Radau quadrature rules, respectively, and they approximate $\varphi(\mu)$ as follows

$$\mathrm{G}_l f_\mu \leq \varphi(\mu) \leq \mathrm{R}_{l+1} f_\mu \tag{4.24}$$

As shown in [17, 18], for a given value of $l \geq 2$, we solve for μ the nonlinear equation

$$\mathrm{G}_l f_\mu = e^2 \tag{4.25}$$

by using Newton's method.

The use the parameter μ in (4.19) instead of $1/\mu$, implies that the left-hand side of (4.20) is a decreasing convex function of μ. Therefore, there is a unique solution, denoted by μ_ε, of

$$\varphi(\mu) = \varepsilon^2$$

for almost all values of $\varepsilon > 0$ of practical interest and therefore also of (4.25) for l sufficiently large; see [17, 18] for analyses. We accept μ_l that solve (4.20) as an approximation of μ, whenever we have

$$\mathrm{R}_{l+1} f_\mu \leq \eta \; \hat{e}^2 \tag{4.26}$$

If (4.26) does not hold for μ_l, we carry out one more GGKB steps, replacing l by $l + 1$ and solve the nonlinear equation

$$\mathrm{G}_{l+1} f_\mu = e^2 \tag{4.27}$$

see [17, 18] for more details. Assume now that (4.26) holds for some μ_l. The corresponding regularized solution is then computed by

$$\mathrm{X}_l = \mathrm{V}_l \bar{x}_{(M+N+1)} y_l, \tag{4.28}$$

where y_{μ_l} solves

$$(\bar{C}_l^T \bar{C}_l + \mu_l^{-1} I_l) y = \sigma_1 \bar{C}_l^T e_1, \qquad \sigma_1 = \|\mathbb{C}\|_F. \tag{4.29}$$

It is also computed by solving the least-squares problem

$$\min_{y \in \mathbb{R}^l} \left\| \begin{bmatrix} \mu^{1/2} \bar{C}_l \\ I_l \end{bmatrix} y - \sigma_1 \begin{bmatrix} \mu^{1/2} e_1 \\ 0 \end{bmatrix} \right\|_2. \tag{4.30}$$

The following result shows an important property of the approximate solution (4.28). We include a proof for completeness.

Proposition 4.5. *Under assumptions of Proposition 4.4, let μ_l solve (4.25) and let y_{μ_l} solve (4.30). Then the associated approximate solution (4.28) of (4.19) satisfies*

$$\|\Phi(\mathbb{X}_{\mu_l}) - \mathbb{C}\|_F^2 = R_{l+1} f_{\mu_l}$$

Proof. By Eq. 4.17, we have

$$\begin{aligned} \Phi(\mathbb{X}_{\mu_l}) = \sum_{i=1}^{l} (\Phi(\mathbb{V}_i)) y_{\mu_l}^i &= \mathbb{W}_l \bar{\times}_{(M+N+1)} y_{\mu_l} \\ &= \mathbb{U}_{l+1} \bar{\times}_{(M+N+1)} (\tilde{C}_l y_{\mu_l}) \end{aligned}$$

Using the above expression gives

$$\begin{aligned} \|\Phi(\mathbb{X}_{\mu_l}) - \mathbb{C}\|_F^2 &= \|\mathbb{U}_{l+1} \bar{\times}_{(M+N+1)} (\tilde{C}_l y_{\mu_l}) - \sigma_1 \mathbb{U}_1\|_F^2 \\ &= \|\mathbb{U}_{l+1} \bar{\times}_{(M+N+1)} (\tilde{C}_l y_{\mu_l}) - \mathbb{U}_{l+1} \bar{\times}_{(M+N+1)} (\sigma_1 e_1)\|_F^2 \\ &= \|\mathbb{U}_{l+1} \bar{\times}_{(M+N+1)} (\tilde{C}_l y_{\mu_l} - \sigma_1 e_1)\|_F^2 \\ &= \|\mathbb{U}_{l+1} \boxtimes^{(M+N+1)} (\mathbb{U}_{l+1} \bar{\times}_{(M+N+1)} (\tilde{C}_l y_{\mu_l} - \sigma_1 e_1))\|_F^2 \\ &= \|\mathbb{U}_{l+1} \boxtimes^{(M+N+1)} \mathbb{U}_{l+1}\|_2^2 \, \|\tilde{C}_l y_{\mu_l} - \sigma_1 e_1\|_2^2 \\ &= \|\tilde{C}_l y_{\mu_l} - \sigma_1 e_1\|_2^2 \end{aligned}$$

where we recall that $\sigma_1 = \|\mathcal{C}\|_F$. We now express y_{μ_l} with the aid of (4.29) and apply the following identity

$$I - A\left(A^TA + \mu^{-1}I\right)^{-1}A^T = \left(\mu AA^T + I\right)^{-1}$$

with A replaced by $\hat{C}_l$, to obtain

$$\begin{aligned}\|\Phi(\mathcal{X}_{\mu_l}) - \mathcal{C}\|_F^2 &= \sigma_1^2\|e_1 - \tilde{C}_l\left(\tilde{C}_l^T\tilde{C}_l + \mu_l^{-1}I_l\right)^{-1}\tilde{C}_l^T e_1\|_F^2 \\ &= \sigma_1^2 e_1^T\left(\mu_l\tilde{C}_l\tilde{C}_l^T + I_{l+1}\right)^{-2}e_1 \\ &= R_{l+1}f_{\mu_l}\end{aligned}$$

which conclude the assertion. □

The following algorithm summarizes the main steps to compute a regularization parameter and a corresponding regularized solution of (4.1) using GGKB and quadrature rules method for Tikhonov regularization.

Algorithm 15 GGKB and quadrature rules method for Tikhonov regularization via Einstein product

1. **Inputs** A linear mapping Φ, $\eta \leq 1$ and ε.
2. Determine the orthonormal bases $\mathcal{U}_{l+1}$ and $\mathcal{V}_l$ of tensors, and the bidiagonal matrices C_l and $\tilde{C}_l$ by implementing Algorithm 14.
3. Determine μ_l that satisfies (4.25) with Newton's method.
4. Determine y_{μ_l} by solving (4.30) and then compute X_{μ_l} by (4.28).

4.4 The tensor block GMRES process

4.4.1 The tensor block Arnoldi process via the Einstein product

In this section, we introduce the The tensor block Arnoldi process via Einstein product (TBAE) as a generalization of well know Block Arnoldi process

.

Let $\mathcal{A} \in \mathbb{R}^{I_1\times\ldots\times I_N\times I_1\times\ldots\times I_N}$ be a square tensor and $\mathcal{V} \in \mathbb{R}^{I_1\times\ldots\times I_N\times J_1\times\ldots\times J_M}$.

The m-th tensor block Krylov subspace is defined by

$$\mathrm{K}_m^{Block}(\mathrm{A},\mathrm{V}) = Range\left[\mathrm{V}, \mathrm{A} *_N \mathrm{V}, \ldots, \mathrm{A}^{m-1} *_N \mathrm{V}\right] \subset \mathrm{R}^{I_1 \times \ldots \times I_N} \tag{4.31}$$

where $\mathrm{A}^i = \mathrm{A} *_N \mathrm{A}^{i-1}, i = 1, \ldots, m-1, \mathrm{A}^0 = \mathrm{I}_N$ and

$$Range(Z) = \left\{ \mathrm{Y} \in \mathrm{R}^{I_1 \times \ldots \times I_N} / \mathrm{Y} = Z *_M \mathrm{X},\ \mathrm{X} \in \mathrm{R}^{J_1 \times \ldots \times J_M} \right\}$$

for $\mathrm{Z} \in \mathrm{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$.

Theorem 4.6. *(QR Factorization via the Einstein product) Let* $\mathrm{A} \in \mathrm{R}^{I_1 \times \ldots I_N \times J_1 \times \ldots \times J_M}$, *then there exists an orthogonal tensor* $\mathrm{Q} \in \mathrm{R}^{I_1 \times \times \ldots I_N \times J_1 \times \ldots \times J_M}$ - $\mathrm{Q}^T *_N \mathrm{Q} = \mathrm{I}_M$- *and* $\mathrm{R} \in \mathrm{R}^{J_1 \times \times \ldots J_M \times J_1 \times \times \ldots J_M}$ *triangular tensor such that*

$$\mathrm{A} = \mathrm{Q} *_M \mathrm{R} \tag{4.32}$$

We call (4.32) a QR factorization of the tensor A.

Proof. Let $\mathrm{A} \in \mathrm{R}^{I_1 \times \ldots I_N \times J_1 \times \ldots \times J_M}$, assume that the QR factorization of the unfolding matrix $A = \Psi_{IJ}(\mathrm{A})$ is : $A = QR$, we have :

$$A = QR$$

Since Ψ is a bijection, we get

$$\begin{aligned} \mathrm{A} &= \Psi_{IJ}^{-1}(A) = \Psi_{IJ}^{-1}(QR) \\ &= \mathrm{Q} *_M \mathrm{R} \end{aligned}$$

□

The algorithm for constructing an orthonormal basis of the tensor block Krylov subspace(4.31) is given as follows. Let

Algorithm 16 The tensor block Arnoldi process via the Einstein product

1. Inputs: A tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times I_1 \times \ldots \times I_N}$, a tensor $\mathcal{V} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$ and the integer m.
2. Set $\mathcal{V} = \mathcal{Q} *_M \mathcal{R}$, $\mathcal{V}^b_1 = \mathcal{Q}$ and $\mathcal{H}_{1,0} = \mathcal{R}$.
3. For $j = 1, \ldots, m$
4. $\mathcal{W} = \mathcal{A} *_N \mathcal{V}^b_j$
5. for $i = 1, \ldots, j$.
   * $\mathcal{H}_{ij} = \mathcal{V}^{b^T}_i *_N \mathcal{W}$,
   * $\mathcal{W} = \mathcal{W} - \mathcal{V}^b_i *_M \mathcal{H}_{ij}$
6. endfor
7. $\mathcal{W} = \mathcal{Q} *_M \mathcal{R}$, $\mathcal{V}^b_{j+1} = \mathcal{Q}$ and $\mathcal{H}_{j+1,j} = \mathcal{R}$.
8. EndFor

$$\mathcal{V}^b_m = \left[\mathcal{V}^b_1, \mathcal{V}^b_2, \ldots, \mathcal{V}^b_m\right] \in \mathbb{R}^{\mathbb{I}_N \times \gamma_1 \times \ldots \times \gamma_M}$$

$$\mathcal{V}^b_{m+1} = \left[\mathcal{V}^b_m, \mathcal{V}^b_{m+1}\right] \in \mathbb{R}^{\mathbb{I}_N \times \lambda_1 \times \ldots \times \lambda_M}$$

$$\mathcal{E}^T_m = [\mathcal{O}_M, \mathcal{O}_M, \ldots, \mathcal{I}_M] \in \mathbb{R}^{J^M \times \gamma_1 \times \ldots \times \gamma_M}$$

$$\mathcal{H}_m = (\mathcal{H}_{i,j})_{1 \le i,j \le m} \in \mathbb{R}^{\gamma_1 \times \ldots \times \gamma_M \times \gamma_1 \times \ldots \times \gamma_M}$$

$$\tilde{\mathcal{H}}_m = \begin{bmatrix} \mathcal{H}_m \\ \mathcal{H}_{m+1,m} *_M \mathcal{E}_m^T \end{bmatrix} \in \mathbb{R}^{\lambda_1 \times \ldots \lambda_M \times \gamma_1 \times \ldots \times \gamma_M}$$

where $\mathbb{I}_N = I_1 \times \ldots \times I_N, \gamma_i = mJ_i, i = 1, \ldots, M, \lambda_i = (m+1)J_i, i = 1, \ldots, M$ and $J^M = J_1 \times \ldots \times J_M$.

Proposition 4.7. *We can state now the following useful relations that could be easily proved.*

$$\mathcal{A} *_N \mathcal{V}^b_m = \mathcal{V}^b_{m+1} *_M \tilde{\mathcal{H}}_m \tag{4.33}$$

$$\mathcal{A} *_N \mathcal{V}^b_m = \mathcal{V}^b_m *_M \mathcal{H}_m + \mathcal{V}^b_{m+1} *_M \mathcal{H}_{m+1,m} *_M \mathcal{E}^T_m \tag{4.34}$$

Notice that in the case $N = M = 1$, Algorithm 16 reduces to the well known block Arnoldi process. In the newt subsection, we introduce a tensor version of the block GMRES algorithm.

4.4.2 The tensor Block GMRES method

Let $\mathcal{A} \in \mathbb{R}^{I_1\times\ldots\times I_N\times I_1\times\ldots\times I_N}$ and $\mathcal{V} \in \mathbb{R}^{I_1\times\ldots\times I_N\times J_1\times\ldots\times J_M}$. If the sub-columns of a tensor $\mathcal{U}$ of size $(I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M)$ are in $\mathcal{K}^{Block}(\mathcal{A}, \mathcal{V}, m)$, then $\mathcal{U}$ can be expressed as

$$\mathcal{U} = \sum_{i=1}^{m} \mathcal{A}^{i-1} *_N \mathcal{V} *_M \mathcal{S}$$

where $\mathcal{S} \in \mathbb{R}^{J_1\times\ldots\times J_M\times J_1\times\ldots\times J_M}$. We denote this linear operator as :

$$\mathcal{U} = \mathcal{Q}_m(\mathcal{A}) \circ \mathcal{V}$$

where $\mathcal{Q}_m$ is the tensor-valued polynomial given by

$$\mathcal{Q}_m(t) = \sum_{i=1}^{m} t^{i-1}\mathcal{S},\ t \in \mathbb{R}.$$

Let $\mathcal{X}_0 \in \mathbb{R}^{I_1\times\ldots\times I_N\times J_1\times\ldots\times J_M}$ be an initial guess tensor and $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_0$ be the corresponding residual. The aim of The tensor block Golub Kahan process via Einstein product (TBGMRESE) is to find, at each step m, an approximation $\mathcal{X}_m$ of the solution $\mathcal{X}^*$ of the problem (6.1), as follows

$$\mathcal{X}_m - \mathcal{X}_0 \in \mathcal{K}_m^{Block}(\mathcal{A}, \mathcal{R}_0) \tag{4.35}$$

and

$$\mathcal{R}_{i,m} \perp \mathcal{K}_m^{Block}(\mathcal{A}, \mathcal{A} *_N \mathcal{R}_0) \tag{4.36}$$

where $\mathcal{R}_{i,m}$ denote the i-th sub-column of $\mathcal{R}_m = \mathcal{C} - \mathcal{A} *_N \mathcal{X}_m$ and consider the tensor $\mathcal{L}_m = \mathcal{R}_0, \mathcal{A} *_N \mathcal{R}_0, \ldots, \mathcal{A}^{m-1} *_N \mathcal{R}_0 \in \mathbb{R}^{I_1\times\ldots\times I_N\times\gamma_1\times\ldots\times\gamma_M}$ and $\mathcal{W}_m = \mathcal{A} *_N \mathcal{L}_m$, where $\gamma_i = mJ_i, i = 1, \ldots, M$. The relation (4.35) can be

expressed as follows

$$\mathcal{X}_m = \mathcal{X}_0 + \mathcal{L}_m *_M \mathcal{Y}_m \tag{4.37}$$

where $\mathcal{Y}_m = \begin{bmatrix} \mathcal{Y}_1 \\ \vdots \\ \mathcal{Y}_m \end{bmatrix} \in \mathbb{R}^{\gamma_1 \times \ldots \times \gamma_M \times J_1 \times \ldots \times J_M}$, $\mathcal{Y}_i \in \mathbb{R}^{J_1 \times \ldots \times J_M \times J_1 \times \ldots \times J_M}$. The orthogonality relation (4.36) implies that

$$(\mathcal{W}_m^T *_N \mathcal{W}_m) *_M \mathcal{Y}_m = \mathcal{W}_m^T *_N \mathcal{R}_0 \tag{4.38}$$

Theorem 4.8. *Assuming that the tensor* $(\mathcal{W}_m^T *_N \mathcal{W}_m)$ *is nonsingular, then the approximation* $\mathcal{X}_m$ *and the corresponding residual* $\mathcal{R}_m$ *are expressed as follows*

$$\mathcal{X}_m = \mathcal{X}_0 + \mathcal{L}_m *_M (\mathcal{W}_m^T *_N \mathcal{W}_m)^{-1} *_M (\mathcal{W}_m^T *_N \mathcal{R}_0) \tag{4.39}$$

and

$$\mathcal{R}_m = \mathcal{R}_0 - \mathcal{W}_m *_M (\mathcal{W}_m^T *_N \mathcal{W}_m)^{-1} *_M (\mathcal{W}_m^T *_N \mathcal{R}_0) \tag{4.40}$$

Proof. Using the relations (4.37) and (4.38), the results follow. □

As we are dealing with an orthogonal projection, the residual norm of the iterates produced by the tensor block GMRES satisfy the following minimization property

$$\|\mathcal{R}_m\|_F = \min_{\mathcal{Z} \in \mathcal{K}_m(\mathcal{A}, \mathcal{R}_0)} \|\mathcal{C} - \mathcal{A} *_N \mathcal{Z}\|_F \tag{4.41}$$

Furthermore, using the fact that $\mathcal{Z} = \mathcal{X}_0 + \mathcal{L}_m *_M \mathcal{Y}$ with $\mathcal{Y} \in \mathbb{R}^{\gamma_1 \times \ldots \times \gamma_M \times J^M}$, relation(4.33) and the fact that $\mathcal{R}_0 = \mathcal{V}_1^b *_M \mathcal{H}_{1,0} = \mathcal{V}_{m+1}^b *_M \mathcal{E}_1 *_M \mathcal{H}_{1,0}$, we

get

$$\begin{aligned}\mathcal{R}_m &= \mathcal{R}_0 - \mathcal{A} *_N \mathcal{L}_m *_M \mathcal{Y} \\ &= \overset{b}{\mathcal{V}}_{m+1} *_M \mathcal{E}_1 *_M \mathcal{H}_{1,0} - \overset{b}{\mathcal{V}}_{m+1} *_M \tilde{\mathcal{H}}_m *_M \mathcal{Y} \\ &= \mathcal{V}^b_{m+1} *_M (\mathcal{E}_1 *_M \mathcal{H}_{1,0} - \tilde{\mathcal{H}}_m *_M \mathcal{Y})\end{aligned}$$

We finally obtain

$$\|\mathcal{R}_m\|_F = \min_{\mathcal{Y}} \|\mathcal{E}_1 *_M \mathcal{H}_{1,0} - \tilde{\mathcal{H}}_m *_M \mathcal{Y}\|_F, \tag{4.42}$$

and then the approximation $\mathcal{X}_m$ is given by

$$\mathcal{X}_m = \mathcal{X}_0 + \mathcal{L}_m *_M \mathcal{Y}_m, \tag{4.43}$$

where $\mathcal{Y}_m$ solves the problem (4.42).

4.5 Tensor Block Golub Kahan method

In this section, we introduce the tensor block version of Golub Kahan algorithm using the Einstein product. Consider the linear system of tensor equations

$$\mathcal{A} *_P \mathcal{X} = \mathcal{B} \tag{4.44}$$

where $\mathcal{A} \in \mathbb{R}^{I_1 \times \ldots I_N \times K_1 \times \ldots \times K_P}$ and $\mathcal{B} \in \mathbb{R}^{I_1 \times \ldots I_N \times J_1 \times \ldots \times J_M}$. Then, the tensor block Golub–Kahan bidiagonalization (TBGK) algorithm via Einstein prod- uct produces orthonormal bases $\mathcal{U}_{m+1} = [\mathcal{U}_1, \ldots, \mathcal{U}_m, \mathcal{U}_{m+1}] \in \mathbb{R}^{I_1 \times \ldots I_N \times \lambda_1 \times \ldots \times \lambda_M}$ and $\mathcal{V}_m = [\mathcal{V}_1, \ldots, \mathcal{V}_m] \in \mathbb{R}^{K_1 \times \ldots \times K_P \times \gamma_1 \times \ldots \times \gamma_M}$, where $\gamma_i = mJ_i, i = 1, \ldots, M$,$\lambda_i = (m+1)J_i, i = 1, \ldots, M$.

The tensor block Golub Kahan process via Einstein product (TBGKE) is summarized in the following algorithm. Let $\tilde{\mathcal{C}}_m \in \mathbb{R}^{\lambda_1 \times \ldots \lambda_M \times \gamma_1 \times \ldots \gamma_M}$ and $\mathcal{C}_m, \mathcal{D}_m \in$

Algorithm 17 The tensor block Golub Kahan algorithm

1. **Inputs** The tensors $\mathcal{A}$, $\mathcal{B}$, and an integer m.
2. Set $\mathcal{V}_0 = 0$ and $\mathcal{R}_0 = \mathcal{B} - \mathcal{A} *_P \mathcal{X}_0 = \mathcal{Q} *_M \mathcal{R}$, $\mathcal{U}_1 = \mathcal{Q}$, $\mathcal{A}_1 = \mathcal{R}$. 3.

For $j = 1, \ldots, m$

(a) $\tilde{\mathcal{V}} = \mathcal{A}^T *_N \mathcal{U}_j - \mathcal{V}_{j-1} *_M \mathcal{A}_j$

(b) $\tilde{\mathcal{V}} = \mathcal{Q} *_M \mathcal{R}$, (QR factorisation); and set $\mathcal{V}_j = \mathcal{Q}$, $\mathcal{B}_j = \mathcal{R}$.

(c) $\tilde{\mathcal{U}} = \mathcal{A} * \mathcal{V} - \mathcal{U} * \mathcal{B}$

(d) $\tilde{\mathcal{U}} = \mathcal{Q} *_M \mathcal{R}$, (QR factorisation); and set $\mathcal{U}_{j+1} = \mathcal{Q}$, $\mathcal{A}_{j+1} = \mathcal{R}$

4. End for

$\mathbb{R}^{\gamma_1 \times \ldots \gamma_M \times \gamma_1 \times \ldots \gamma_M}$ as follows

$$\tilde{\mathcal{C}}_m = \begin{bmatrix} \mathcal{B}_1 & & & \\ \mathcal{A}_2 & \mathcal{B}_2 & \ddots & \\ & \ddots & \ddots & \\ & & \mathcal{A}_m & \mathcal{B}_m \\ & & & \mathcal{A}_{m+1} \end{bmatrix}, \quad \mathcal{C}_m = \begin{bmatrix} \mathcal{B}_1 & & \\ \mathcal{A}_2 & \mathcal{B}_2 & \ddots \\ & \ddots & \ddots \\ & & \mathcal{A}_m \quad \mathcal{B}_m \end{bmatrix}, \quad \mathcal{D}_m = \begin{bmatrix} \mathcal{B}_1 & \mathcal{A}_2 & & \\ & \mathcal{B}_2 & \ddots & \\ & & \ddots & \mathcal{A}_m \\ & & & \mathcal{B}_m \end{bmatrix}$$

Then we have the following results

Proposition 4.9. *The tensors produced by the tensor block Golub-Kahan algorithm satisfy the following relations*

$$\mathcal{A} *_P \mathcal{V}_m = \mathcal{U}_{m+1} *_M \tilde{\mathcal{C}}_m, \tag{4.45}$$

$$= \mathcal{U}_m *_M \mathcal{C}_m + \mathcal{U}_{m+1} *_M \mathcal{A}_{m+1} *_M \mathcal{E}_m^T, \tag{4.46}$$

$$\mathcal{A}^T *_N \mathcal{U}_m = \mathcal{V}_m *_M \mathcal{D}_m, \tag{4.47}$$

$$\mathcal{R}_0 = \mathcal{U}_{m+1} *_M \mathcal{E}_1 *_M \mathcal{A}_1. \tag{4.48}$$

where $\mathcal{E}_m^T = [\mathcal{O}_M, \mathcal{O}_M, \ldots, \mathcal{I}_M] \in \mathbb{R}^{J_1 \times \ldots \times J_M \times \gamma_1 \times \ldots \times \gamma_M}$, $\mathcal{E}_1 = \begin{bmatrix} \mathcal{I}_M \\ \vdots \\ \mathcal{O}_M \end{bmatrix} \in \mathbb{R}^{\lambda_1 \times \ldots \times \lambda_M \times J_1 \times \ldots \times J_M}$

and $\mathcal{U}_m = [\mathcal{U}_1, \ldots, \mathcal{U}_m] \in \mathbb{R}^{I_N \times \gamma_1 \times \ldots \times \gamma_M}$.

Proof. The proofs come directly from different steps of Algorithm 17. □

.

Proposition 4.10. *Let* $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m *_M \mathcal{Y}_m \in \mathbb{R}^{K_1 \times \ldots \times K_P \times J_1 \times \ldots \times J_M}$ *with* $\mathcal{Y}_m \in \mathbb{R}^{\gamma_1 \times \ldots \times \gamma_M \times J_1 \times \ldots \times J_M}$ *where* $\mathcal{V}_m$ *is obtained from Algorithm 17, be an approximation of the solution* $\mathcal{X}^*$ *of the tensor linear system* (4.44) *. Then*

$$\arg \min_{\mathcal{X} = \mathcal{X}_0 + \mathcal{V}_m *_M \mathcal{Y}} \|\mathcal{A} *_P \mathcal{X} - \mathcal{B}\|_F \tag{4.49}$$

is given by $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m *_M \mathcal{Y}_m$ *where* $\mathcal{Y}_m$ *solves the minimisation problem*

$$\min_{\mathcal{Y}} ||\mathcal{E}_1 *_M \mathcal{A}_1 - \tilde{\mathcal{C}}_m *_M \mathcal{Y}||_F \tag{4.50}$$

Proof. Using the results of Proposition 4.9, we get

$$\begin{aligned} ||\mathcal{R}_0 - \mathcal{A} *_P \mathcal{X}_m||_F &= ||\mathcal{U}_{m+1} *_M \mathcal{E}_1 *_M \mathcal{A}_1 - \mathcal{U}_{m+1} *_M \tilde{\mathcal{C}}_m *_M \mathcal{Y}_m||_F \\ &= ||\mathcal{U}_{m+1} *_M (\mathcal{E}_1 *_M \mathcal{A}_1 - \tilde{\mathcal{C}}_m *_M \mathcal{Y}_m)||_F = ||\mathcal{E}_1 *_M \mathcal{A}_1 - \tilde{\mathcal{C}}_m *_M \mathcal{Y}_m||_F, \end{aligned}$$

which ends the proof.

The low dimensional minimisation tensor problem (4.50) is then solved by a QR factorisation of the tensor $\tilde{\mathcal{C}}_m$. □

4.6 Numerical results

This section provides some numerical results to show the performance of Algorithms 13 and Algorithm 15 when applied to the restoration of blurred and noisy color images and videos. For clarity and definiteness, we first focus on the formulation of a tensor model, describing the blurring that is taking place in the process of going from the exact to the blurred RGB image (or video). Notwithstanding what has just been said, recovering RGB (or video) from their blurry and noisy observations can be seen as a tensor problem of the form (6.1). Therefore, it's very important to understand how the model (6.1) can be constructed for RGB images and color video deblurring problems.

In what follows, we will concentrate only on the formulation of the tensor model for RGB image deblurring problems and will comment at the end of this section how a similar one can be formulated for color video deblurring problems. We recall that an RGB image is just multidimensional array of dimension $M \times N \times 3$ whose entries are the light intensity. Throughout this paper, we assume that the original RGB image has the same dimensions as the blurred one, and we refer to it as $N \times N \times 3$ tensor. Let $\mathcal{C}$ represent the available blurred RGB image, let $\mathcal{X}$ denote the desired unknown blurred RGB, and let $\mathcal{A}$ be the tensor describing the blurring that is taking place in the process of going from $\mathcal{X}$ to $\mathcal{C}$. It is well known in the literature of image processing that all the blurring operators can be characterized by a Point Spread Function (PSF) describing the blurring process and the boundary conditions outside the image, see [68]. Once the two-dimensional PSF array, P, is specified, we can as well build the blurring tensor $\mathcal{A}$. By using the fact that the blurring process of an RGB image is simply a multi-dimensional convolution operation of the PSF array P and the original three-dimensional image $\mathcal{X}$, the blurring tensor $\mathcal{A}$ can be easily constructed by placing the elements of P in the appropriate positions. Note that the PSF is a two-dimensional array P describing the image of a single white pixel, which makes its dimensions much smaller than N. Therefore, P contains all the required information about the blurring throughout the RGB image $\mathcal{C}$. To illustrate this, the discrete operation for multi-dimensional convolution using a 3×3 local and spatially invariant PSF array P with p_{22} is its center, and assuming zero boundary conditions, is given by:

$$\mathcal{C}_{ijk} = p_{33}\mathcal{X}_{i-1j-1k} + p_{32}\mathcal{X}_{i-1jk} + p_{31}\mathcal{X}_{i-1j+1k} + p_{23}\mathcal{X}_{ij-1k} + p_{22}\mathcal{X}_{ijk} \quad (4.51)$$

$$+ p_{21}\mathcal{X}_{ij+1k} + p_{13}\mathcal{X}_{i+1j-1k} + p_{12}\mathcal{X}_{i+1jk} + p_{11}\mathcal{X}_{i+1j+1k}, \quad (4.52)$$

for $i, j = 1, \ldots, N$ and $k = 1, 2, 3$. Here the zero boundary conditions are imposed so the values of $\mathcal{X}$ are zero outside the RGB image, i.e., $\mathcal{X}_{i0k} =$

$\mathcal{X}_{iN+1k} = \mathcal{X}_{0jk} = \mathcal{X}_{N+1jk} = 0$ for $0 < i, j < N+1$ and $k = 1, 2, 3$. The fourth order tensor $\mathcal{A} \in \mathbb{R}^{N \times N \times N \times N}$ associated with (4.51), with partition $(1, N, 1, N)$, can be partitioned into matrix blocks of size $N \times N$. Each block is denoted by $\mathcal{A}^{(2,4)}_{i_2,i_4} = \mathcal{A}(:, i_2, :, i_4) \in \mathbb{R}^{N \times N}$ with $i_2 = 1, \ldots, N$ and $i_4 = 1, \ldots, N$. The nonzero entries of the matrix block $\mathcal{A}^{(2,4)}_{a,b} \in \mathbb{R}^{N \times N}$ are given by

$$
\begin{aligned}
&(\mathcal{A}^{(2,4)}_{a,b})_{a-1b-1} = p_{33}; \quad && (\mathcal{A}^{(2,4)}_{a,b})_{ab+1} = p_{21} \\
&(\mathcal{A}^{(2,4)}_{a,b})_{a-1b} = p_{32}; \quad && (\mathcal{A}^{(2,4)}_{a,b})_{a+1b-1} = p_{13} \\
&(\mathcal{A}^{(2,4)}_{a,b})_{a-1b+1} = p_{31}; \quad && (\mathcal{A}^{(2,4)}_{a,b})_{a+1b} = p_{12} \\
&(\mathcal{A}^{(2,4)}_{a,b})_{ab-1} = p_{23}; \quad && (\mathcal{A}^{(2,4)}_{a,b})_{a+1b+1} = p_{11} \\
&(\mathcal{A}^{(2,4)}_{a,b})_{ab} = p_{22}
\end{aligned}
$$

for $a, b = 2, \ldots, N-1$.

The first following examples applies Algorithms 13 and 15 to the restoration of blurred color image and video that have been contaminated by Gaussian blur and by additive zero-mean white Gaussian noise. We consider the blurring to be local and spatially invariant. In this the case the entries of the Gaussian PSF array P are given by

$$
p_{ij} = \exp\left(-\frac{1}{2}\left(\frac{(i-k)}{\sigma}\right)^2 - \frac{1}{2}\left(\frac{(j-l)}{\sigma}\right)^2\right),
$$

where σ controls the width of the Gaussian PSF and (k, l) is its center, see [68]. Note that σ controls the amount of smoothing, i.e. the larger the σ, the more ill posed the problem. The original tensor image is denoted by $\hat{\mathcal{X}}$ in each example and $\mathcal{A}$ represents the blurring tensor. The tensor $\hat{\mathcal{C}} = \mathcal{A} *_N \hat{\mathcal{X}}$ represents the associated blurred and noise-free multichannel image. We generated a blurred and noisy tensor image $\mathcal{C} = \hat{\mathcal{C}} + \mathcal{N}$, where $\mathcal{N}$ is a noise tensor with normally distributed random entries with zero mean and with variance chosen to correspond to a specific noise level $\nu := \|\mathcal{N}\|_F / \|\hat{\mathcal{C}}\|_F$. To

determine the effectiveness of our solution methods, we evaluate

$$\text{RE} = \frac{\|\hat{\mathcal{X}} - \mathcal{X}_{\text{restored}}\|_F}{\|\hat{\mathcal{X}}\|_F}$$

and the Signal-to-Noise Ratio (SNR) defined by

$$\text{SNR}(\mathcal{X}_{\text{restored}}) = 10\log_{10} \frac{\|\hat{\mathcal{X}} - E(\hat{\mathcal{X}})\|_F^2}{\|\mathcal{X}_{\text{restored}} - \hat{\mathcal{X}}\|_F^2}$$

where $E(\hat{\mathcal{X}})$ denotes the mean gray-level of the uncontaminated image $\hat{\mathcal{X}}$. All computations were carried out using the MATLAB environment on an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz (8 CPUs) computer with 12 GB of RAM. The computations were done with approximately 15 decimal digits of relative accuracy.

4.6.1 Example 1

This example illustrates the performance of Algorithms 13 and 15 4 when applied to the restoration of 3-channel RGB color image that have been contaminated by Gaussian blur and additive noise. The original (unknown) RGB image $\hat{\mathcal{X}} \in \mathbb{R}^{256\times256\times3}$ is the **papav256** image from **MATLAB**. It is shown on the left-hand side of Figure 4.1. For the blurring tensor $\mathcal{A}$, we consider a PSF array P with $\sigma = 2$ under zero boundary conditions. The associated blurred and noisy RGB image $\mathcal{C} = \mathcal{A} *_N \hat{\mathcal{X}}$ is shown on the right-hand side of Figure 4.1. The noise level is $v = 10^{-3}$. Given the contaminated RGB image $\mathcal{C}$, we would like to recover an approximation of the original RGB image $\hat{\mathcal{X}}$. Table 4.1 compares, the computing time (in seconds), the relative errors and the PSNR of the computed restorations. Note that in this table, the allowed maximum number of outer iterations for Algorithm 13 with noise level $v = 10^{-2}$ was 4. The restoration for noise level $v = 10^{-3}$ is shown on the left-hand side of Figure 5.2 and it is obtained by applying Einstein tensor global GMRES method (Algorithm 13) with input $\mathcal{A}$, $\mathcal{C}$, $\mathcal{X}_0 = \mathcal{O}$, $\varepsilon = 10^{-6}$, $m = 10$

and **Maxit** = 10. Using GCV, the computed optimal value for the projected problem in Algorithm 2 was $\mu_5 = 9.44 \times 10^{-4}$. The restoration obtained with Algorithm 15 is shown on the right-hand side of Figure 4.2. The discrepancy principle with $\eta = 1.1$ is satisfied when $l = 61$ steps of the Einstein tensor GGKB method have been carried out, producing a regularization parameter given by $\mu_l = 2.95 \times 10^{-4}$. For comparison with existing approaches in the literature, we report in Table 4.1 the results obtained with the method proposed in [**?**] for solving the corresponding matrix form of (4.3). This method utilizes the connection between (standard) Golub–Kahan bidiagonalization and Gauss quadrature rules for solving large ill-conditioned linear systems of equations obtained by matricization of problem (4.3). We refer to this method as GKB. It is a solution method based on first reducing the blurring matrix to a small bidiagonal matrix with the aid of Golub–Kahan bidiagonalization (GKB) and then applying the connection between GKB and Gauss-type quadrature rules (the same as the ones in (4.22)-(4.23) to determine an approximation of x_μ that satisfies the discrepancy principle associated to the matrix problem. It determines the regularization parameter analogously to Algorithm 15, and uses a similar stopping criterion.

Table 4.1: *Results for Example 1.*

Noise level	Method	PSNR	RE	CPU-time (seconds)
10^{-3}	Algorithm 13	21.76	6.09×10^{-2}	8.28
	Algorithm 15	24.37	4.51×10^{-2}	7.29
	GKB	24.22	4.51×10^{-2}	18.45
10^{-2}	Algorithm 13	20.60	6.96×10^{-2}	3.31
	Algorithm 15	20.97	6.67×10^{-2}	1.58
	GKB	20.08	6.66×10^{-2}	5.52

4.6.2 Example 2

In this example, we evaluate the effectiveness of Algorithms 13 and 15 when applied to the restoration of a color video defined by a sequence of RGB images. Video restoration is the problem of restoring a sequence of k color

Figure 4.1: *Example 1: Original image (top), blurred and noisy image (bottom).*

images (frames). Each frame is represented by a tensor of $N \times N \times 3$ pix- els. In the present example, we are interested in restoring 10 consecutive frames of a contaminated video. We consider the xylophone video from MATLAB. The video clip is in MP4 format with each frame having 240×240 pixels. The (unknown) blur- and noise-free frames are stored in the tensor $\hat{\mathcal{C}} \in \mathbb{R}^{N \times N \times 3 \times 10}$. These frames are blurred by a blurring tensor $\mathcal{A}$ of the same kind and with the same parameters as in the previous example. Figure 5.5 shows the 5th exact (original) frame and the contaminated version, which is to be restored. Blurred and noisy frames are generated by $\hat{\mathcal{C}} = \mathcal{A} *_N \hat{\mathcal{X}}$ where the tensor $\mathcal{E}$ represents white Gaussian noise of levels $v = 10^{-3}$ or $v = 10^{-2}$. Table 4.2 displays the performance of algorithms. For Algorithm 13, we have used as an input $\mathcal{A}$, $\mathcal{C}$, $\mathcal{X}_0 = \mathcal{O}$, $\varepsilon = 10^{-6}$, $m = 10$ and **Maxit** $= 10$. For the ten outer iterations, minimizing the GCV function produces $\mu_{10} = 9.44 \times 10^{-4}$.

Figure 4.2: *Example 1: Restored image by Algorithm 4 (top), and restored image by Algorithm 2 (bottom).*

Using Algorithm 15, the discrepancy principle with $\eta = 1.1$ have been satisfied after $l = 59$ steps of the Einstein tensor GGKB method, producing a regularization parameter given by $\mu_l = 1.06 \times 10^{-4}$. The restorations obtained with Algorithms 13 and 15 are shown on the left-hand and right-hand sides of Figure 4.4, respectively.

Table 4.2: *Results for Example 2.*

Noise level	Method	PSNR	Relative error	CPU-time (second)
10^{-3}	Algorithm 13	15.48	6.84×10^{-2}	38.93
	Algorithm 15	19.24	4.43×10^{-2}	27.37
10^{-2}	Algorithm 13	14.50	7.65×10^{-2}	15.55
	Algorithm 15	15.13	7.11×10^{-2}	4.40

Figure 4.3: *Frame no. 5: Original frame (left), blurred and noisy frame (right).*

Figure 4.4: *Frame no. 5: Restored frame by Algorithm 4 (left), and restored frame by Algorithm 2 (right).*

4.7 Conclusion

In this chapter, we extended the GMRES and Golub–Kahan bidiagonalization in conjunction of Tikhonov regularization for solving (possibly) ill-conditioned multilinear systems via Einstein product with perturbed right-hand side. We gave also tensor block version of these processes and derive some theoretical results. Numerical experiments were disclosed for image and video processing to demonstrate the feasibility of the proposed iterative algorithms.

Chapter 5

Tensor GMRES and Golub Kahan methods via the T-product for color image processing

The present chapter is concerned with developing tensor iterative Krylov subspace methods to solve large multi-linear tensor equations. We use the T-product for two tensors to define tensor tubal global Arnoldi and ten- sor tubal global Golub-Kahan bidiagonalization algorithms. Furthermore, we illustrate how tensor–based global approaches can be exploited to solve ill-posed problems arising from recovering blurry multichannel (color) im- ages and videos, using the so-called Tikhonov regularization technique, to provide computable approximate regularized solutions. We also review a generalized cross-validation and discrepancy principle type of criterion for the selection of the regularization parameter in the Tikhonov regularization. Applications to image sequence processing are given to demonstrate the ef- ficiency of the algorithms.

5.1 Introduction

The aim of this paper is to solve the following tensor problem

$$\mathcal{M}(\mathcal{X}) = \mathcal{C}, \tag{5.1}$$

where $\mathcal{M}$ is a linear operator that could be described as

$$\mathcal{M}(\mathcal{X}) = \mathcal{A} * \mathcal{X}, \tag{5.2}$$

or as

$$\mathcal{M}(\mathcal{X}) = \mathcal{A} * \mathcal{X} * \mathcal{B}, \tag{5.3}$$

where $\mathcal{A}$, $\mathcal{X}$, $\mathcal{B}$ and $\mathcal{C}$ are three-way tensors, leaving the specific dimensions to be defined later, and $*$ is the T-product to be also defined later. The aim is to take advantage of this multidimensional structure to build rapid and robust iterative methods for solving large-scale problems. We will then, be interested in developing robust and fast iterative tensor Krylov subspace methods under tensor-tensor product framework between third-order tensors, to solve regularized problems originating from color image and video processing applications. Standard and global Krylov subspace methods are suitable when dealing with grayscale images, e.g, [17, 18, 35]. However, these methods might be time consuming to numerically solve problems related to multi channel images (e.g. color images, hyper-spectral images and videos).

For the Einstein product, both the Einstein tensor global Arnoldi and Einstein tensor global Golub-Kahan bidiagonalization algorithms have been established [48], which makes so natural to generalize these methods using the T-product. In this paper, we will show that the tensor-tensor product between third-order tensors allows the application of the global iterative methods, such as the global Arnoldi and global Golub-Kahan algorithms.

The tensor form of the proposed Krylov methods, together with using the fast Fourier transform (FFT) to compute the T-product between third-order tensors can be efficiently implemented on many modern computers and allows to significantly reduce the overall computational complexity. It is also worth mentioning that our approaches can be naturally generalized to higher-order tensors in a recursive manner.

The chapter is organized as follows. We shall first present in Section 5.2 some symbols and notations used throughout chapter. We also recall the concept of the T-product between two tensors. In Section 5.3, we define tensor global Arnoldi and Tensor Global Golub Kahan (TGK) algorithms that allow the use of the T-product. Section 5.4 reviews on the adaptation of Tikhonov regularization for the tensor equation (5.1) and then proposing a restarting strategy of the tensor global GMRES and tensor global Golub-Kahan approaches in connection with Gauss-type quadrature rules to inexpensively compute solutions of the regularization. Also we give a tensor formulation in the form of (5.1) that describes the cross-blurring of color image and then we present a few numerical examples on restoring blurred and noisy color images and videos. Concluding remarks can be found in Section 5.5.

5.2 Definitions and Notations

All definitions and properties related to the T-product are defined ealier; see chapter 3 section 3.2. Next, we introduce the T-diamond tensor product.

Definition 5.1. Let $\mathcal{F} = [\mathcal{F}_1, \ldots, \mathcal{F}_p] \in \mathbb{R}^{n_1 \times ps \times n_3}$, where $\mathcal{F}_i \in \mathbb{R}^{n_1 \times s \times n_3}$, $i = 1, \ldots, p$ and let $\mathcal{B} = [\mathcal{B}_1, \ldots, \mathcal{B}_l] \in \mathbb{R}^{n_1 \times ls \times n_3}$ with $\mathcal{B}_j \in \mathbb{R}^{n_1 \times s \times n_3}$, $j = 1, \ldots, l$. Then, the product $\mathcal{F}^T \diamond \mathcal{B}$ is the $p \times l$ matrix given by :

$$(\mathcal{F}^T \diamond \mathcal{B})_{i,j} = \langle \mathcal{F}_i, \mathcal{B}_j \rangle .$$

5.3 Tensor T-global GMRES and tensor T-global Golub-Kahan algorithms

5.3.1 The tensor T-global GMRES

Consider the following tensor linear system of equations

$$\mathcal{A} * \mathcal{X} = \mathcal{C}, \tag{5.4}$$

where $\mathcal{A} \in \mathbb{R}^{n \times n \times p}$, $\mathcal{C}$ and $\mathcal{X} \in \mathbb{R}^{n \times s \times p}$. We introduce the tensor Krylov subspace $\mathcal{TK}_m(\mathcal{A}, \mathcal{V})$ associated to the T-product, defined for the pair $(\mathcal{A}, \mathcal{V})$ as follows

$$\mathcal{TK}_m(\mathcal{A}, \mathcal{V}) = \text{Tspan}\{\mathcal{V}, \mathcal{A} * \mathcal{V}, \ldots, \mathcal{A}^{m-1} * \mathcal{V}\} = \left(\mathcal{Z} \in \mathbb{R}^{n \times s \times p}, \mathcal{Z} = \sum_{i=1}^{m} \alpha_i \mathcal{A}^{i-1} * \mathcal{V} \right) \tag{5.5}$$

where $\alpha_i \in \mathbb{R}$, $\mathcal{A}^{i} * \mathcal{V} = \mathcal{A}^{i-2} * \mathcal{A} * \mathcal{V}$, for $i = 2, \ldots, m$ and $\mathcal{A}^0$ is the identity tensor. In the following algorithm, we define the Tensor T-global Arnoldi algorithm.

Algorithm 18 Tensor T-global Arnoldi

1. **Input.** $\mathcal{A} \in \mathbb{R}^{n \times n \times p}$, $\mathcal{V} \in \mathbb{R}^{n \times s \times p}$ and the positive integer m.
2. Set $\beta = \|\mathcal{V}\|_F$, $\mathcal{V}_1 = \dfrac{\mathcal{V}}{\beta}$
3. For $j = 1, \ldots, m$
   - *(a)* $\mathcal{W} = \mathcal{A} * \mathcal{V}_j$
   - *(b)* for $i = 1, \ldots, j$
     - i. $h_{i,j} = \langle \mathcal{V}_i, \mathcal{W} \rangle$
     - *ii.* $\mathcal{W} = \mathcal{W} - h_{i,j} \mathcal{V}_i$
   - (c) End for
   - (d) $h_{j+1,j} = \|\mathcal{W}\|_F$. If $h_{j+1,j} = 0$, stop; else
   - (e) $\mathcal{V}_{j+1} = \mathcal{W}/h_{j+1,j}$.
4. End

It is not difficult to show that after m steps of Algorithm 18, the tensors $\mathcal{V}_1, \ldots, \mathcal{V}_m$, form an orthonormal basis of the tensor global Krylov subspace $\mathcal{TK}_m(\mathcal{A}, \mathcal{V})$. Let $\mathcal{V}_m$ be the $(n \times (sm) \times p)$ tensor with frontal slices $\mathcal{V}_1, \ldots, \mathcal{V}_m$ and let $\tilde{H}_m$ be the $(m+1) \times m$ upper Hessenberg matrix whose elements are the $h_{i,j}$'s defined by Algorithm 18. Let H_m be the matrix obtained from $\tilde{H}_m$ by deleting its last row; $H_{.,j}$ will denote the j-th column of the matrix H_m and $\mathcal{A} * \mathcal{V}_m$ is the $(n \times (sm) \times p)$ tensor with frontal slices $\mathcal{A} * \mathcal{V}_1, \ldots, \mathcal{A} * \mathcal{V}_m$:

$$\mathcal{V}_m := [\mathcal{V}_1, \ldots, \mathcal{V}_m] \quad \text{and} \quad \mathcal{A} * \mathcal{V}_m := [\mathcal{A} * \mathcal{V}_1, \ldots, \mathcal{A} * \mathcal{V}_m]. \quad (5.6)$$

We introduce the product ② defined by

$$\mathcal{V}_m \textcircled{2} y = \sum_{j=1}^{m} y_j \mathcal{V}_j, \quad y = (y_1, \ldots, y_m)^T \in \mathbb{R}^m,$$

and we set

$$\mathcal{V}_m \textcircled{2} H_m = [\mathcal{V}_m \textcircled{2} H_{.,1}, \ldots, \mathcal{V}_m \textcircled{2} H_{.,m}].$$

Then, it is easy to see that for all vectors u and v in $\mathbb{R}^m$, we have

$$\mathcal{V}_m \textcircled{2} (u+v) = \mathcal{V}_m \textcircled{2} u + \mathcal{V}_m \textcircled{2} v \quad \text{and} \quad (\mathcal{V}_m \textcircled{2} H_m) \textcircled{2} u = \mathcal{V}_m \textcircled{2} (H_m u). \quad (5.7)$$

With these notations, we can show the following result that will be useful later on.

Proposition 5.2. *Let $\mathcal{V}_m$ be the tensor defined by $[\mathcal{V}_1, \ldots, \mathcal{V}_m]$ where $\mathcal{V}_i \in \mathbb{R}^{n \times s \times p}$ are defined by the Tensor T-global Arnoldi algorithm. Then, we have*

$$\|\mathcal{V}_m \textcircled{2} y\|_F = \|y\|_2, \ \forall\, y = (y_1, \ldots, y_m)^T \in \mathbb{R}^m. \quad (5.8)$$

Proof. From the definition of the product ②, we have $\sum_{j=1}^{m} y_j \mathcal{V}_j = \mathcal{V}_m \textcircled{2} y$. Therefore,

$$\|\mathcal{V}_m \textcircled{2} y\|_F^2 = \left\langle \sum_{j=1}^{m} y_j \mathcal{V}_j, \sum_{j=1}^{m} y_j \mathcal{V}_j \right\rangle_F.$$

But, since the tensors $\mathcal{V}_i$'s are orthonormal, it follows that

$$\|\mathcal{V}_m \circledast y\|_F^2 = \sum_{j=1}^{m} y_j^2 = \|y\|_2^2,$$

which shows the result.

□

With the above notations, we can easily prove the results of the following proposition.

Proposition 5.3. *Suppose that m steps of Algorithm 18 have been run. Then, the following statements hold:*

$$\text{ffi} * \mathcal{V}_m = \mathcal{V}_m \circledast H_m + h_{m+1,m}\, \mathcal{O}_{n\times s\times p}, \ldots, \mathcal{O}_{n\times s\times p}, \mathcal{V}_{m+1}, \quad (5.9)$$

$$\text{ffi} * \mathcal{V}_m = \mathcal{V}_{m+1} \circledast \tilde{H}_m, \quad (5.10)$$

$$\mathcal{V}_m^T \text{ffi} * \mathcal{V}_m = H_m, \quad (5.11)$$

$$\mathcal{V}_{m+1}^T \text{ffi} * \mathcal{V}_m = \tilde{H}_m, \quad (5.12)$$

$$\mathcal{V}_m^T \mathcal{V}_m = I_m, \quad (5.13)$$

where I_m the identity matrix and $\mathcal{O}_{n\times s\times p}$ is the tensor having all its entries equal to zero.

Proof. From Algorithm 18, we have $\text{ffi} * \mathcal{V}_j = \sum_{i=1}^{j+1} h_{i,j}\mathcal{V}_i$. Using the fact that $\text{ffi} * \mathcal{V}_m = [\text{ffi} * \mathcal{V}_1, \ldots, \text{ffi} * \mathcal{V}_m]$, the j-th frontal slice of $\text{ffi} * \mathcal{V}_m$ is given by

$$(\text{ffi} * \mathcal{V}_m)_j = \text{ffi} * \mathcal{V}_j = \sum_{i=1}^{j+1} h_{i,j}\mathcal{V}_i.$$

Furthermore, from the definition of the ② product, we have

$$\begin{aligned}(\mathcal{V}_{m+1} \circledast \tilde{H}_m)_j &= \mathcal{V}_{m+1} \circledast H_{\cdot,j}, \\ &= \sum_{i=1}^{j+1} h_{i,j}\mathcal{V}_i,\end{aligned}$$

which proves the first two relations. The other relations follow from the definition of T-diamond product □

In the sequel, we develop the tensor T-global GMRES algorithm for solving the problem (5.4). It could be considered as generalization of the global GMERS algorithm [79]. Let $\mathcal{X}_0 \in \mathbb{R}^{n \times s \times p}$ be an arbitrary initial guess with the corresponding residual $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} * \mathcal{X}_0$. The aim of tensor T-global GMRES method is to find and approximate solution $\mathcal{X}_m$ approximating the exact solution $\mathcal{X}^*$ of (5.4) such that

$$\mathcal{X}_m - \mathcal{X}_0 \in \mathcal{TK}_m(\mathcal{A}, \mathcal{R}_0), \tag{5.14}$$

with the classical minimization property

$$\|\mathcal{R}_m\|_F = \min_{\mathcal{X} \in \mathcal{X}_0 + \mathcal{TK}_m(\mathcal{A}, \mathcal{R}_0)} \|\mathcal{C} - \mathcal{A} * \mathcal{X}\|_F. \tag{5.15}$$

Let $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m ② y$ with $y \in \mathbb{R}^m$, be the approximate solution satisfying (5.14). Then,

$$\|\mathcal{R}_m\|_F = \min_{y \in \mathbb{R}^m} \|\mathcal{R}_0 - (\mathcal{A} * \mathcal{V}_m) ② y\|_F,$$

where $\mathcal{A} * \mathcal{V}_m := [\mathcal{A} * \mathcal{V}_1, \ldots, \mathcal{A} * \mathcal{V}_m]$ is the $(n \times sm \times p)$ tensor defined earlier. Using Proposition 5.2 and the fact that $\mathcal{R}_0 = \|\mathcal{R}_0\|_F \mathcal{V}_1$ with $\mathcal{V}_1 = \mathcal{V}_{m+1} ② e_1$, where e_1 the first canonical basis vector in $\mathbb{R}^{m+1}$, we get

$$y = \arg \min_{y \in \mathbb{R}^m} \| \|\mathcal{R}_0\|_F e_1 - \tilde{H}_m y \|_2. \tag{5.16}$$

5.3.2 The Tensor T-global Golub Kahan algorithm

Instead of using the tensor T-global Arnoldi to generate a basis for the projection subspace, we can define T-version of the tensor global Lanczos process. Here, we will use the tensor Golub Kahan algorithm related to the T-product.

Let $\mathcal{A} \in \mathbb{R}^{n\times l\times p}$ be a tensor and let $\mathcal{U} \in \mathbb{R}^{l\times s\times p}$ and $\mathcal{V} \in \mathbb{R}^{n\times s\times p}$ two other given tensors. The Tensor T-global Golub Kahan bidiagonalization algorithm (associated to the T-product) is defined as follows

Algorithm 19 The Tensor T-global Golub Kahan algorithm

1. **Input.** The tensors $\mathcal{A}$, $\mathcal{V}$, and $\mathcal{U}$ and an integer m.
2. Set $\beta_1 = \|\mathcal{V}\|_F$, $\alpha_1 = \|\mathcal{U}\|_F$, $\mathcal{V}_1 = \mathcal{V}/\beta_1$ and $\mathcal{U}_1 = \mathcal{U}/\alpha_1$. 3.

for $j = 2, \ldots, m$

(a) $\tilde{\mathcal{V}} = \mathcal{A} * \mathcal{U}_{j-1} - \alpha_{j-1}\mathcal{V}_{j-1}$

(b) $\beta_j = \|\tilde{\mathcal{V}}\|_F$ if $\beta_j = 0$ stop, else

(c) $\mathcal{V}_j = \tilde{\mathcal{V}}_j/\beta_j$

(d) $\tilde{\mathcal{U}} = \mathcal{A}^T * \mathcal{V}_j - \beta_j\mathcal{U}_{j-1}$

(e) $\alpha_j = \|\tilde{\mathcal{U}}\|_F$

(f) if $\alpha_j = 0$ stop, else

(g) $\mathcal{U}_j = \tilde{\mathcal{U}}/\alpha_j$.

Let $\tilde{C}_m$ be the upper bidiagonal $((m+1)\times m)$ matrix

$$\tilde{C}_m = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \\ & & \beta_m & \alpha_m \\ & & & \beta_{m+1} \end{bmatrix}$$

and let C_m be the $(m \times m)$ matrix obtain by deleting the last row of $\tilde{C}_m$. We denote by $C_{.,j}$ the j-th column of the matrix C_m. Let $\mathcal{U}_m$ and $\mathcal{A} * \mathcal{U}_m$ be the $(l \times (sm) \times p)$ and $(n \times (sm) \times p)$ tensors with frontal slices $\mathcal{U}_1, \ldots, \mathcal{U}_m$ and $\mathcal{A} * \mathcal{U}_1, \ldots, \mathcal{A} * \mathcal{U}_m$, respectively, and let $\mathcal{V}_m$ and $\mathcal{A}^T * \mathcal{V}_m$ be the $(n \times (sm) \times p)$ and $(l \times (sm) \times p)$ tensors with frontal slices $\mathcal{V}_1, \ldots, \mathcal{V}_m$ and $\mathcal{A}^T * \mathcal{V}_1, \ldots, \mathcal{A}^T * \mathcal{V}_m$, respectively. We set

$$\mathcal{U}_m := [\mathcal{U}_1, \ldots, \mathcal{U}_m], \quad \text{and} \quad \mathcal{A} * \mathcal{U}_m := [\mathcal{A} * \mathcal{U}_1, \ldots, \mathcal{A} * \mathcal{U}_m], \tag{5.17}$$

$$\mathcal{V}_m := [\mathcal{V}_1, \ldots, \mathcal{V}_m], \quad \text{and} \quad \mathcal{A}^T * \mathcal{V}_m := [\mathcal{A}^T * \mathcal{V}_1, \ldots, \mathcal{A}^T * \mathcal{V}_m]. \tag{5.18}$$

Then, the following proposition can be established

Proposition 5.4. *The tensors produced by the tensor T-global Golub-Kahan algorithm satisfy the following relations*

$$\begin{aligned} \mathcal{A} * \mathcal{U}_m &= \mathcal{V}_{m+1} ② \tilde{C}_m, && (5.19)\\ &= \mathcal{V}_m ② C_m + \beta_{m+1} \begin{bmatrix} \mathcal{O}_{n\times s\times p}, \ldots, \mathcal{O}_{n\times s\times p}, \mathcal{V}_{m+1} \end{bmatrix}, && (5.20)\\ \mathcal{A}^T * \mathcal{V}_m &= \mathcal{U}_m ② \tilde{C}_m^T. && (5.21) \end{aligned}$$

Proof. Using $\mathcal{A} * \mathcal{U}_m = [\mathcal{A} * \mathcal{U}_1, \ldots, \mathcal{A} * \mathcal{U}_m] \in \mathbb{R}^{n\times(sm)\times n3}$, the $(j-1)$-th frontal slice of $(\mathcal{A} * \mathcal{U}_m)$ is given by

$$(\mathcal{A} * \mathcal{U}_m)_{j-1} = \mathcal{A} * \mathcal{U}_{j-1} = \alpha_{j-1}\mathcal{V}_{j-1} + \beta_j \mathcal{V}_j.$$

Furthermore, from the definition of the ② product, we have

$$\begin{aligned} (\mathcal{V}_{m+1} ② \tilde{C}_m)_{j-1} &= \mathcal{V}_{m+1} ② C_{:,j-1},\\ &= \alpha_{j-1}\mathcal{V}_{j-1} + \beta_j \mathcal{V}_j \end{aligned}$$

and for $j = m$, $\mathcal{U}_m ② C_{:,m} = \mathcal{A} * \mathcal{U}_m + \beta_{m+1}\mathcal{V}_{m+1}$ and the result follows. The other relations are proven is a similar way. □

5.4 Application to discrete-ill posed tensor problems

In what follows, we will apply the tensor global-GMRES and tensor Golub Kahan algorithms to some discrete ill posed problems. We consider the following discrete ill-posed tensor equation

$$\mathcal{A} * \mathcal{X} = \mathcal{C}, \quad \mathcal{C} = \hat{\mathcal{C}} + \mathcal{N}, \tag{5.22}$$

where $\mathcal{A} \in \mathbb{R}^{n \times n \times p}$, $\mathcal{X}$, $\mathcal{N}$ (additive noise) and $\mathcal{C}$ are tensors in $\mathbb{R}^{n \times s \times p}$.

In color image processing, $p = 3$, $\mathcal{A}$ represents the blurring tensor, $\mathcal{C}$ the blurry and noisy observed image, $\mathcal{X}$ is the image that we would like to restore and $\mathcal{N}$ is an unknown additive noise. Therefore, to stabilize the recovered image, regularization techniques are needed. There are several techniques to regularize the linear inverse problem given by equation (5.22); for the matrix case, see for example, [17, 35, 60, 67]. All of these techniques stabilize the restoration process by adding a regularization term, depending on some priori knowledge of the unknown image. One of the most regularization method is due to Tikhonov and is given as follows

$$\min_{\mathcal{X}} \{\|\mathcal{A} * \mathcal{X} - \mathcal{C}\|_F^2 + \mu \|\mathcal{X}\|_F^2\}. \tag{5.23}$$

As problem (5.22) is large, Tikhonov regularization (5.23) may be very expensive to solve. One possibility is instead of regularizing the original problem, we apply the Tikhonov technique to the projected problem (**??**) which leads to the following problem

$$y_{m,\mu} = \arg\min_{y \in \mathbb{R}^m} \|R_0 e_1 - \tilde{H}_m y\|_2^2 + \mu^2 \|y\|_2^2, \tag{5.24}$$

$$= \arg\min_{y \in \mathbb{R}^m} \left\| \begin{bmatrix} \tilde{H}_m \\ \mu I_m \end{bmatrix} y - \begin{bmatrix} \beta e_1 \\ 0 \end{bmatrix} \right\|_2. \tag{5.25}$$

The minimizer $y_{m,\mu}$ can also be computed as the solution of the following normal equations associated with (5.25)

$$\tilde{H}_{m,\mu} y = \tilde{H}_m^T, \quad \tilde{H}_{m,\mu} = (\tilde{H}_m^T \tilde{H}_m + \mu^2 I_m). \tag{5.26}$$

Note that since the Tikhonov problem (5.26) is now a matrix one with small dimension as m is generally small, the vector $y_{m,\mu}$, can thereby be inexpensively computed by some techniques such as the GCV method [60] or the L-curve criterion [35, 67].

In terms of practical implementations, it's more convenient to introduce a restarted version of the tensor Global GMRES. This strategy is essentially based on restarting the tensor T-global Arnoldi algorithm. Therefore, at each restart, the initial guess $\mathcal{X}_0$ and the regularization parameter μ are updated employing the last values computed when the the number of inner iterations required is fulfilled. We note that as the number outer iterations increases it is possible to compute the mth residual without having to compute extra T-products. At step m, the residual $\mathcal{R}_m = \mathcal{C} - \mathcal{A} * \mathcal{X}_m$ produced by the tensor Global GMRES method for tensor equation (5.1) has the following expression

$$\mathcal{R}_m = \mathcal{V}_{m+1} \textcircled{2} (\gamma_{m+1} Q_m e_{m+1}), \tag{5.27}$$

where Q_m is the unitary matrix obtained from the QR decomposition of the upper Hessenberg matrix $\tilde{H}_m$ and γ_{m+1} is the last component of the vector $\|\mathcal{R}_0\|_F Q_m^{\mathrm{T}} e_1$ and $e_{m+1} = (0, 0, \ldots, 1)^{\mathrm{T}} \in \mathbb{R}^{m+1}$.
Furthermore, it is easy to show that

$$\|\mathcal{R}_m\|_F = |\gamma_{m+1}|. \tag{5.28}$$

The tensor T-global GMRES method is summarized in the following algorithm

Algorithm 20 Implementation of Tensor T-global GMRES(m)

1. **Input.** $\mathcal{A} \in \mathbb{R}^{n \times n \times p}$, $\mathcal{V}, \mathcal{B}, \mathcal{X}_0 \in \mathbb{R}^{n \times s \times p}$, the maximum number of iteration $\mathrm{Iter}_{\max}$ and a tolerance $tol > 0$.
2. **Output.** $\mathcal{X}_m \in \mathbb{R}^{n \times s \times n3}$ approximate solution of the system (5.4).
3. $k = 1, \ldots, \mathrm{Iter}_{\max}$
   (a) Compute $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} * \mathcal{X}_0$.
   (b) Apply Algorithm 18 to compute $\mathcal{V}_m$ and $\tilde{H}_m$.
   (c) Determine μ_k as the parameter minimizing the GCV function [60].
   (d) Compute the regularized solution $y_{m_k,\mu}$ of the problem (5.25).
   (e) Compute the approximate solution $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m \textcircled{2} y_{m,\mu_k}$
4. If $\|\mathcal{R}_m\|_F < tol$, stop, else
5. Set $\mathcal{X}_0 = \mathcal{X}_m$ and go to 3-a.
6. End

We turn now to the tensor T-global Golub Kahan approach for the solving the Tikhonov regularization of the problem (5.1). Here, we apply the following Tikhonov regularization approach and solve the new problem

$$\min_{\mathcal{X}} \{\|\mathcal{A} * \mathcal{X} - \mathcal{C}\|_F^2 + \mu^{-1}\|\mathcal{X}\|_F^2\}. \tag{5.29}$$

The use of μ^{-1} in (5.29) instead of μ will be justified below. In the what follows, we briefly review the discrepancy principle approach to determine a suitable regularization parameter, given an approximation of the norm of the additive error. We then assume that a bound ε for $\|\mathcal{N}\|_F$ is available. This priori information suggests that μ has to be determined as soon as

$$\varphi(\mu) \leq \eta\varepsilon, \tag{5.30}$$

where $\varphi(\mu) = \|\mathcal{A} * \mathcal{X} - \mathcal{C}\|_F^2$ and $\eta \geq 1$ is refereed to as the safety factor for the discrepancy principle. A zero-finding method can be used to solve (5.30) in order to find a suitable regularization parameter which also implies that $\varphi(\mu)$ has to be evaluated for several μ-values. When the tensor $\mathcal{A}$ is of moderate size, the quantity $\varphi(\mu)$ can be easily evaluated. This evaluation becomes expensive when the matrix $\mathcal{A}$ is large, which means that its evaluation by a zero-finding method can be very difficult and computationally expensive. We will approximate φ to be able to determine an estimate of $\|\mathcal{A} * \mathcal{X} - \mathcal{C}\|_F^2$. Our approximation is obtained by using T-global Golub- Kahan bidiagonalization (T-GGKB) and Gauss-type quadrature rules. This connection provides approximations of moderate sizes to the quantity φ, and therefore gives a solution method to inexpensively solve (5.30) by evaluating these small quantities that can successfully and inexpensively be employed to compute μ as well as defining a stopping criterion for the T-GGKB itera- tions; see [17, 18] for discussion on this method.

Introduce the functions (of μ)

$$f_\mu(t) := (\mu t + 1)^{-2} \tag{5.31}$$

$$\mathrm{G}_{m,\mu} := \|\mathcal{C}\|_F^2 e_1^T f_\mu(C_m C_m^T) e_1, \tag{5.32}$$

$$\mathrm{R}_{m+1,\mu} := \|\mathcal{C}\|_F^2 e_1^T f_\mu(\tilde{C}_m \tilde{C}_m^T) e_1, \tag{5.33}$$

The quantities $\mathrm{G}_{m,\mu}$ and $\mathrm{R}_{m+1,\mu}$ are refereed to as Gauss and Gauss-Radau quadrature rules, respectively, and can be obtained after m steps of T-GGKB (Algorithm 25) applied to tensor $\mathcal{A}$ with initial tensor $\mathcal{C}$. These quantities approximate $\varphi(\mu)$ as follows

$$\mathrm{G}_{m,\mu} \leq \varphi(\mu) \leq \mathrm{R}_{m+1,\mu}. \tag{5.34}$$

Similarly to the approaches proposed in [17, 18], we therefore instead solve for μ the low dimensional nonlinear equation

$$\mathrm{G}_{m,\mu} = \varepsilon^2. \tag{5.35}$$

We apply the Newton's method to solve (5.35) that requires repeated evaluation of the function $\mathrm{G}_m f_\mu$ and its derivative, which are inexpensive computations for small m.

We now comment on the use of μ in (5.29) instead of $1/\mu$, implies that the left-hand side of (5.30) is a decreasing convex function of μ. Therefore, there is a unique solution, denoted by μ_ε, of

$$\varphi(\mu) = \varepsilon^2$$

for almost all values of $\varepsilon > 0$ of practical interest and therefore also of (5.35) for m sufficiently large; see [17, 18] for analyses. We accept μ_m that solve

(5.30) as an approximation of μ, whenever we have

$$\mathrm{R}_{m+1,\mu} \leq \eta \ \hat{e}^{2}_{2}. \tag{5.36}$$

If (5.36) does not hold for μ_m, we carry out one more GGKB steps, replacing m by $m + 1$ and solve the nonlinear equation

$$\mathrm{G}_{m+1,\mu} = e^{2}; \tag{5.37}$$

see [17, 18] for more details. Assume now that (5.36) holds for some μ_m. The corresponding regularized solution is then computed by

$$\mathsf{X}_{m,\mu_m} = \mathrm{U}_m \textcircled{2} y_{m,\mu_m} \tag{5.38}$$

where y_{m,μ_m} solves

$$(\tilde{C}_m^T \tilde{C}_m + \mu_m^{-1} I_m) y = \alpha_1 \tilde{C}_m^T e_1, \qquad \alpha_1 = \|\mathsf{C}\|_F. \tag{5.39}$$

It is also computed by solving the least-squares problem

$$\min_{y \in \mathrm{R}^m} \left\| \begin{bmatrix} \mu_m^{1/2} \tilde{C}_m \\ I_m \end{bmatrix}^{1/2} y - \alpha_1 \mu_m \ e_1 \right\|_2^2. \tag{5.40}$$

The following result shows an important property of the approximate solution (5.38). We include a proof for completeness.

Proposition 5.5. *Let μ_m solve (5.35) and let y_{m,μ_m} solve (5.40). Then the associated approximate solution (5.38) of (5.29) satisfies*

$$\|\mathsf{ffi} * \mathsf{X}_{m,\mu_m} - \mathsf{C}\|_F^2 = \mathrm{R}_{m+1,\mu_m}.$$

Proof. From the items of Proposition 5.4, we get the following matrix low dimensional least squares problem

$$\| \text{ffi} * X_{m,\mu_m} - C \|_F^2 = \| \tilde{C} y_{m,\mu_m} - \alpha e_1 \|_2^2$$

where $\alpha_1 = \|C\|_F$. We now express y_{m,μ_m} with the aid of (**??**) and by using the following matrix identity

$$I - A \left(A^T A + \mu^{-1} I \right)^{-1} A^T = \left(\mu A A^T + I \right)^{-1}$$

with A replaced by $\tilde{C}_m$, to obtain

$$\begin{aligned} \| \text{ffi} * X_{m,\mu_m} - C \|_F^2 &= \alpha_1^2 \| e_1 - C_m \left(C_m^T C_m + \mu_m I_{m-1} \right)^{-1} C_m^T e_1 \|_2^2, \\ &= \alpha^2 e_1^T \left(\mu_m \tilde{C}_m \tilde{C}_m^T + I_{m+1} \right)^{-2} e_1, \\ &= R_{m+1,\mu_m}. \end{aligned}$$

□

The following algorithm summarizes the main steps to compute a regularization parameter and a corresponding regularized solution of (5.1), using Tensor T-GGKB and quadrature rules method for Tikhonov regularization.

Algorithm 21 T-GGK and quadrature rules method for Tikhonov regularization

1. **Input.** ffi $\in \mathbb{R}^{n \times n \times n_3}$, C, η ' 1 and ε.
2. **Output.** T-GGKB steps m, μ_m and X_{m,μ_m}.
3. Determine the orthonormal bases U_{m+1} and V_m of tensors, and the bidiagonal C_m and $\tilde{C}_m$ matrices with Algorithm 19.
4. Determine μ_m that satisfies (5.35) with Newton's method.
5. Determine y_{m,μ_m} by solving (6.16) and then compute X_{m,μ_m} by (5.38).

We comment on the complexity of Algorithm 20 and Algorithm 26. First note that the overall computational cost for Algorithm 20 is dominated by the work required to determine V_m in Algorithm 18. The computational

effort required to determine V_m is dominated by the evaluation of m T-products, which demands approximately, $O(n_1 n_2 n_3 \log n_3 m)$ flops by using Algorithm 1. Concerning Algorithm 21, the computational complexity is dominated by the work needed to determine U_m and V_m in Algorithm 19, which demands approximately $O(n_1 n_2 n_3 \log n_3 m)$ flops. We show in the following section that Algorithm 20 and Algorithm 21 solve problem (5.1) in a time and cost less that the ones required when using the method proposed in [35] that uses the connection between (standard) Golub–Kahan bidiagonalization and Gauss quadrature rules for solving large ill-conditioned linear systems of equations of form (5.1).

5.5 Numerical results

This section performs some numerical tests on the methods of Tensor T-Global GMRES(m) and Tensor T-Global Golub Kahan algorithm given by Algorithm 20 and Algorithm 21, rspectively, when applied to the restoration of blurred and noisy color images and videos. For clarity, we only focus on the formulation of a tensor model (6.5), describing the blurring that is taking place in the process of going from the exact to the blurred RGB image. We recall that an RGB image is just multidimensional array of dimension $m \times n \times 3$ whose entries are the light intensity. Throughout this section, we assume that the the three channels of the RGB image has the same dimensions, and we refer to it as $n \times n \times 3$ tensor. Let $\hat{X}^{(1)}$, $\hat{X}^{(2)}$, and $\hat{X}^{(3)}$ be the $n \times n$ matrices that constitute the three channels of the original error-free color image $\hat{\mathcal{X}}$, and $\hat{C}^{(1)}$, $\hat{C}^{(2)}$, and $\hat{C}^{(3)}$ the $n \times n$ matrices associated with error-free blurred color image $\hat{\mathcal{C}}$. Because of some unique features in images, we seek an image restoration model that utilizes blur information, exploiting the spatially invariant properties. Let us also consider that both cross-channel and within-channel blurring take place in the blurring process of the original image. Let vec be the operator that transforms a matrix to a

vector by stacking the columns of the matrix from left to right. Then, the full blurring model is described by the following form

$$\mathbf{A}_{\text{color}} \otimes \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \hat{\mathbf{x}} = \hat{\mathbf{c}}, \tag{5.41}$$

where,

$$\hat{\mathbf{c}} = \begin{bmatrix} \text{vec}\left(\hat{\mathbf{C}}^{(1)}\right) \\ \text{vec}\left(\hat{\mathbf{C}}^{(2)}\right) \\ \text{vec}\left(\hat{\mathbf{C}}^{(3)}\right) \end{bmatrix}, \quad \hat{\mathbf{x}} = \begin{bmatrix} \text{vec}\left(\hat{\mathbf{X}}^{(1)}\right) \\ \text{vec}\left(\hat{\mathbf{X}}^{(2)}\right) \\ \text{vec}\left(\hat{\mathbf{X}}^{(3)}\right) \end{bmatrix},$$

and

$$\mathbf{A}_{\text{color}} = \begin{bmatrix} a_{\text{rr}} & a_{\text{rg}} & a_{\text{rb}} \\ a_{\text{gr}} & a_{\text{gg}} & a_{\text{gb}} \\ a_{\text{br}} & a_{\text{bg}} & a_{\text{bb}} \end{bmatrix}$$

$\mathbf{A}_{\text{color}}$ is the 3×3 matrix that models the cross-channel blurring, where each row sums to one. This matrix is obtained from [68]. We consider the special case where $a_{\text{rr}} = a_{\text{gg}} = a_{\text{bb}}$, $a_{\text{gr}} = a_{\text{rg}}$, $a_{\text{br}} = a_{\text{rb}}$, and $a_{\text{bg}} = a_{\text{gb}}$, which gives rise to a cross-channel circular mixing. $\mathbf{A}^{(1)} \in \mathbb{R}^{n \times n}$ and $\mathbf{A}^{(2)} \in \mathbb{R}^{n \times n}$ define within-channel blurring and they model the horizontal within blurring and the vertical within blurring matrices, respectively; for more details see [68]. The notation $\otimes$ denotes the Kronecker product of matrices; i.e. the Kronecker product of a $n \times p$ matrix $A = (a_{ij})$ and a $(s \times q)$ matrix $B = (b_{ij})$, is defined as the $(ns) \times (pq)$ matrix $A \otimes B = (a_{ij}B)$. By exploiting the circulant structure of the cross-channel blurring matrix $\mathbf{A}_{\text{color}}$ and the operators unfold and fold, it can be easily shown that (5.41) can be written in the following tensor form

$$\mathcal{ffi} * \hat{\mathcal{X}} * \mathcal{B} = \hat{\mathcal{C}}, \tag{5.42}$$

where $\mathcal{ffi}$ is a 3-way tensor such that $\mathcal{ffi}(:,:,1) = \alpha\mathbf{A}^{(2)}$, $\mathcal{ffi}(:,:,2) = \beta\mathbf{A}^{(2)}$ and $\mathcal{ffi}(:,:,3) = \gamma\mathbf{A}^{(2)}$ and $\mathcal{B}$ is a 3-way tensor with $\mathcal{B}(:,:,1) = (\mathbf{A}^{(1)})^T$, $\mathcal{B}(:,:$

$, 2) = 0$ and $\mathsf{B}(:, :, 3) = 0$. To test the performance of algorithms, the within blurring matrices $A^{(i)}$ have the following entries

$$a_{kl} = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(k-l)^2}{2\sigma^2}\right), & |k-l| \le r \\ 0, & \text{otherwise} \end{cases}$$

Note that σ controls the amount of smoothing, i.e. the larger the σ, the more ill posed the problem. We generated a blurred and noisy tensor image $\mathsf{C} = \hat{\mathsf{C}} + \mathsf{N}$, where N is a noise tensor with normally distributed random entries with zero mean and with variance chosen to correspond to a specific noise level $\nu := \|\mathsf{N}\|_F / \|\mathsf{C}\|_F$. To determine the effectiveness of our solution methods, we evaluate

$$\text{Relative error} = \frac{\|\hat{\mathsf{X}} - \mathsf{X}_{\text{restored}}\|_F}{\|\hat{\mathsf{X}}\|_F}$$

and the Signal-to-Noise Ratio (SNR) defined by

$$\text{SNR}(\mathsf{X}_{\text{restored}}) = 10\log_{10} \frac{\|\hat{\mathsf{X}} - E(\hat{\mathsf{X}})\|_F^2}{\|\mathsf{X}_{\text{restored}} - \hat{\mathsf{X}}\|_F^2},$$

where $E(\hat{\mathsf{X}})$ denotes the mean gray-level of the uncontaminated image $\hat{\mathsf{X}}$. All computations were carried out using the MATLAB environment on an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz (8 CPUs) computer with 12 GB of RAM. The computations were done with approximately 15 decimal digits of relative accuracy.

5.5.1 Example 1

In this example we present the experimental results recovered by Algorithm 20 and Algorithm 21 for the reconstruction of a cross-channel blurred color

images that have been contaminated by both within and cross blur, and additive noise. The cross-channel blurring is determined by the matrix

$$\mathbf{A}_{\text{color}} = \begin{bmatrix} 0.8 & 0.10 & 0.10 \\ 0.10 & 0.80 & 0.10 \\ 0.10 & 0.10 & 0.80 \end{bmatrix}.$$

We consider two RGB images from **MATLAB**, papav256 ($\hat{\mathcal{X}} \in \mathbb{R}^{256\times 256\times 3}$) and peppers ($\hat{\mathcal{X}} \in \mathbb{R}^{512\times 512\times 3}$). They are shown on Figure **??**. For the within-channel blurring, we let $\sigma = 4$ and $r = 6$. The considered noise levels are $v = 10^{-3}$ and $v = 10^{-2}$. The associated blurred and noisy RGB images $\mathcal{C} = \mathcal{A} * \hat{\mathcal{X}} * \mathcal{B} + \mathcal{N}$ for noise level $v = 10^{-3}$ are shown on Figure 5.2. Given the contaminated RGB image $\mathcal{C}$, we would like to recover an approximation of the original RGB image $\mathcal{X}$. The restorations for noise level $v = 10^{-3}$ are shown on Figure 6.3 and they are obtained by applying Algorithm 20 im- plementing the Tensor T-Global GMRES method, with $\mathcal{X}_0 = \mathcal{O}$, $tol = 10^{-6}, m = 10$ and $\text{Iter}_{\max} = 10$. Using GCV, the computed optimal value for the projected problem was $\mu_{10} = 3.82 \times 10^{-5}$. Table 5.1 compares, the com- puting time (in seconds), the relative errors and the SNR of the computed restorations. Note that in this table, the allowed maximum number of outer iterations for Algorithm 20 with noise level $v = 10^{-2}$ was $\text{Iter}_{\max} = 4$ and the maximum number of inner iterations was $m = 4$. The restorations ob- tained with Algorithm 21 are shown on Figure 5.4. For the papav256 color image, the discrepancy principle with $\eta = 1.1$ is satisfied when $m = 64$ steps of the Tensor T-GGKB method (Algorithm 25) have been carried out, producing a regularization parameter given by $\mu_m = 5.57 \times 10^{-5}$. For com- parison with existing approaches in the literature, we report in Table 5.1 the results obtained with the method proposed in [**?**]. This method utilizes the connection between (standard) Golub–Kahan bidiagonalization and Gauss quadrature rules for solving large ill-conditioned linear systems of equations

(5.41) which equivalent to the tensor problem (5.42). We refer to this method as GKB. It is a solution method based on first reducing $\mathbf{A}_{\text{color}} \otimes \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)}$ to a small bidiagonal matrix with the aid of Golub–Kahan bidiagonalization (GKB) and then applying the connection between GKB and Gauss-type quadrature rules (the same as the ones in (5.31)) to determine an approximation of x_μ that satisfies the discrepancy principle associated to the linear problem (5.41). It determines the regularization parameter analogously to Algorithm 21, and uses a similar stopping criterion. The FFT-based computation of the T-product is the only difference in Algorithm 20 and Algorithm 21. We can see that the methods yield restorations of the same quality, but the new proposed methods perform significantly better in terms of cpu-time. This is due to the fact that the T-product operations need fewer flops (approximately $36mn$ flops) than structure-ignoring evaluation of matrix-vector products with the large matrix $\mathbf{A}_{\text{color}} \otimes \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)}$ in (5.41) and its transpose, which requires approximately $2(3m)^2(3n)^2$ flops.

RGB images	Noise level	Method	SNR	Relative error	CPU-time (sec)
papav256	10^{-3}	Algorithm 20	21.01	6.64×10^{-2}	6.62
		Algorithm 21	20.41	7.12×10^{-2}	5.87
		GKB	20.99	7.12×10^{-2}	18.61
	10^{-2}	Algorithm 20	18.00	9.40×10^{-2}	1.18
		Algorithm 21	17.78	9.64×10^{-2}	1.11
		GKB	17.78	9.64×10^{-2}	5.79
peppers	10^{-3}	Algorithm 20	19.39	5.50×10^{-2}	24.32
		Algorithm 21	19.11	5.68×10^{-2}	25.63
		GKB	19.11	5.68×10^{-2}	78.13
	10^{-2}	Algorithm 20	16.23	7.92×10^{-2}	4.59
		Algorithm 21	15.61	8.50×10^{-2}	3.39
		GKB	15.61	8.50×10^{-2}	15.16

Table 5.1: *Results for Example 1.*

Figure 5.1: *Example 1: Original RGB images:* peppers *(left),* papav256 *(right).*

Figure 5.2: *Example 1: Blurred and noisy images,* peppers *(left),* papav256 *(right).*

Figure 5.3: *Example 1: Restored images by Algorithm 20,* peppers *(left),* papav256 *(right).*

Figure 5.4: *Example 1: Restored images by Algorithm 21,* peppers *(left),* papav256 *(right).*

5.5.2 Example 2

In this example, we evaluate the effectiveness of Algorithm 20 and Algorithm 21 when applied to the restoration of a color video defined by a sequence of RGB images. Video restoration is the problem of restoring a sequence of k color images (frames). Each frame is represented by a tensor of $n \times n \times 3$ pixels. In the present example, we are interested in restoring 10 consecutive frames of a contaminated video. Note that the processing of such given frames, one at a time, is extremely time consuming. We con- sider the xylophone video from MATLAB. The video clip is in MP4 format with each frame having 240×240 pixels. The (unknown) blur- and noise-free frames are stored in the tensor $\hat{\mathcal{X}} \in \mathbb{R}^{240 \times 240 \times 30}$, obtained by stacking the grayscale images that constitute the three channels of each blurred color frame. These frames are blurred by $\mathcal{ffi} * \hat{\mathcal{X}} * \mathcal{B} = \hat{\mathcal{C}}$, where $\mathcal{ffi}$ and $\mathcal{B}$ are a 3-way tensors such that $\mathcal{ffi}(:,:,1) = \mathbf{A}^{(2)}$, $\mathcal{B}(:,:,1) = (\mathbf{A}^{(1)})^T$ and $\mathcal{ffi}(:,:,i) = \mathcal{B}(:,:,i) = 0$, for $i = 2, ..., 30$, using $\sigma = 2$ and $r = 4$ to build the blurring matrices with periodic boundary conditions. This gives rises to block circulant with circulant blocks matrices. We consider white Gaussian noise of levels $v = 10^{-3}$ or $v = 10^{-2}$. Figure 5.5 shows the 5th exact (original) frame and the contaminated version with noise level $v = 10^{-3}$, which is to be restored. Table 5.2 displays the performance of Algorithm 20 and Algorithm 21. In Algorithm 20, we have used as an input for noise level $v = 10^{-3}$, $\mathcal{C}$, $\mathcal{X}_0 = \mathcal{O}$, $tol = 10^{-6}$, $m = 10$ and $\text{Iter}_{\max} = 10$.The chosen inner and outer iterations for noise level $v = 10^{-2}$ were $m = 4$ and $\text{Iter}_{\max} = 4$, respectively. For the ten outer iterations, minimizing the GCV function produces $\mu_{10} = 1.15 \times 10^{-5}$. Using Algorithm 21, the discrepancy principle with $\eta = 1.1$ have been satisfied after $m = 59$ steps of T-GGKB method (Algo- rithm 19), producing a regularization parameter given by $\mu_m = 1.06 \times 10^{-4}$. For comparison with existing approaches in the literature, we report in Ta- ble 5.2 the results obtained with the method proposed in [18]. This method

Figure 5.5: *Example 2: Original frame no. 5 (left), blurred and noisy frame no. 5 (right).*

utilizes the connection between the Global Golub–Kahan bidiagonalization and Gauss quadrature rules for solving large ill-conditioned linear systems of equations (5.41). We refer to this method as GGKB. It determines the regularization parameter analogously to Algorithm 21, and uses a similar stopping criterion. We can see that Algorithm 21 yields restorations of the same quality as the GGKB method, but the new proposed methods perform significantly better in terms of cpu-time. For completeness, the restorations obtained with Algorithm 20 and Algorithm 21 are shown on the left-hand and the right-hand side of Figure 5.6, respectively.

Table 5.2: *Results for Example 2.*

Noise level	Method	SNR	Relative error	CPU-time (second)
10^{-3}	Algorithm 20	11.22	1.09×10^{-1}	50.22
	Algorithm 21	16.17	6.31×10^{-2}	45.22
	GGKB	16.01	7.56×10^{-2}	60.10

5.6 Conclusion

In thischapter we proposed tensor versions of GMRES and Golub—Kahan bidiagonalization algorithms using the T-product, with applications to solving large-scale linear tensor equations arising in the reconstruction of blurred and noisy multichannel images and videos. We also introduced new tensor

Figure 5.6: *Example 2: restored frame no. 5 by Algorithm 20 (left), and restored frame no. 5 by Algorithm 21 (right).*

products as generalisations of other well known matrix products. The numerical experiments that we have performed show the effectiveness of the proposed schemes to inexpensively compute regularized solutions of high quality.

Chapter 6

Discrete cosine transform LSQR methods for multidimensional ill-posed problems

In the present work, we propose new tensor Krylov subspace method for ill posed linear tensor problems such as in color or video image restoration. Those methods are based on the tensor-tensor discrete cosine transform that gives fast tensor-tensor product computations. In particular, we will focus on the tensor discrete cosine versions of GMRES, Golub-Kahan bidiagonalisation and LSQR methods. The presented numerical tests show that the methods are very fast and give good accuracies when solving some linear tensor ill-posed problems.

6.1 Introduction

The aim of this paper is to solve the following tensor problem

$$\min_{\mathcal{X}} \|\mathcal{M}(\mathcal{X}) - \mathcal{C}\|_F \tag{6.1}$$

where $\mathbb{M}$ is a linear operator that could be described as

$$\mathbf{M}(\mathcal{X}) = \mathcal{A} *_c \mathcal{X}, \text{ or } \mathbf{M}(\mathcal{X}) = \mathcal{A} *_c \mathcal{X} *_c \mathcal{B}, \tag{6.2}$$

where $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is a three mode tensor, $\mathcal{X} \in \mathbb{R}^{n_2 \times s \times n_3}$, $\mathcal{B} \in \mathbb{R}^{s \times s \times n_3}$ and $\mathcal{C} \in \mathbb{R}^{n_1 \times s \times n_3}$ are three mode tensors, and $*_c$ is the cosine product to be also defined later. In this chapter, we will show that the tensor-tensor product between third-order tensors allows the application of the global iterative methods, such as the global Arnoldi and global Golub-Kahan algorithms. The tensor form of the proposed Krylov methods, together with using the fast cosine transform (DCT) to compute the c-product between third-order tensors can be efficiently implemented on many modern computers and allows to significantly reduce the overall computational complexity. It is also worth mentioning that our approaches can be naturally generalized to higher-order tensors in a recursive manner.

This chapter is organized as follows. We shall first present in Section 6.2 some symbols and notations used throughout the paper. We also recall some definitions related to the cosine product between two tensors. In Section 6.4 , we present some inexpensive approaches based on cosine global Krylov subspace methods combined with regularization techniques to solve the obtained ill-posed tensor problem (6.1) . Section 6.5 is dedicated to some numerical experiments.

6.2 Properties of the cosine product

In this subsection, we briefly review some concepts and notations, that play a central role for the elaboration of the tensor iterative methods based on the c-product; see [88] for more details on the c-product.

Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ be a third-order tensor, then the operations $\mathtt{mat}$ and its

reverse ten are defined by

$$\text{mat}(\mathcal{A}) = \underbrace{\begin{bmatrix} A_1 & A_2 & \dots & A_n \\ A_2 & A_1 & \dots & A_3 \\ \vdots & \vdots & \dots & \vdots \\ A_n & A_{n-1} & \dots & A_1 \end{bmatrix}}_{\text{Block Toeplitz}} + \underbrace{\begin{bmatrix} A_2 & \dots & A_n & 0 \\ \vdots & & & A_n \\ A_n & 0 & \dots & \vdots \\ 0 & A_n & \dots & A_2 \end{bmatrix}}_{\text{Block Hankel}} \in \mathbb{R}^{n_1 n_3 \times n_2 n_3}$$

where the matrices $A_i \in \mathbb{R}^{n_1 \times n_2}$, $i = 1, \dots, n_3$ are the frontal slices of the tensor $\mathcal{A}$. The reverse operation denoted by ten is such that

$$\text{ten}(\text{mat}(\mathcal{A})) = \mathcal{A}.$$

Let $\tilde{\mathcal{A}}$ be the tensor obtained by applying the DCT on all the tubes of the tensor $\mathcal{A}$. With the Matlab command dct as

$$\tilde{\mathcal{A}} = \text{dct}(\mathcal{A}, [\,], 3), \text{ and } \text{idct}(\tilde{\mathcal{A}}, [\,], 3) = \mathcal{A},$$

where idct denotes the Inverse Discrete Cosine Transform.

Remark 6.1. Notice that the tensor $\tilde{\mathcal{A}}$ can be computed by using the 3-mode product defined in [91] as follows:

$$\tilde{\mathcal{A}} = \mathcal{A} \times_3 M,$$

where M is the $n_3 \times n_3$ invertible matrix given by

$$M = W^{-1} C_{n_3}(I + Z),$$

and C_{n_3} denotes the $n_3 \times n_3$ Discrete Cosine Transform DCT matrix, $W = \text{diag}(C_{n_3}(:, 1))$ is the diagonal matrix made of the first column of the DCT matrix, Z is an $n_3 \times n_3$ circulant matrix which can be computed in MATLAB using the command $\mathsf{W} = \text{diag}(\text{ones}(\mathsf{n}_3 - 1, 1), 1)$ and I the $n_3 \times n_3$ identity

matrix; see [88] for more details.

6.3 Properties of the cosine product

In this subsection, we briefly review some concepts and notations, that play a central role for the elaboration of the tensor iterative methods based on the c-product; see [88] for more details on the c-product.

Let $\mathbf{A}$ be the matrix

$$\mathbf{A} = \begin{bmatrix} A^{(1)} & & & \\ & A^{(2)} & & \\ & & \ddots & \\ & & & A^{(n_3)} \end{bmatrix} \in \mathrm{R}^{n_3 n_1 \times n_3 n_2}, \tag{6.3}$$

where the matrices $A^{(i)}$'s are the frontal slices of the tensor $\tilde{\mathsf{ffi}}$. The block matrix mat(ffi) can also be block diagonalized using the DCT matrix and this gives

$$(C_{n3} \otimes I_{m1})\, \mathsf{mat}(\mathsf{ffi})\, (C_{h_3}^T \otimes I_{n2}) = \mathbf{A}. \tag{6.4}$$

Definition 6.2. The **c-product** between two tensors $\mathsf{ffi} \in \mathrm{R}^{n1 \times n_2 \times n_3}$ and $\mathsf{B} \in \mathrm{R}^{n2 \times m \times n_3}$ is an $n_1 \times m \times n_3$ tensor given by:

$$\mathsf{ffi} *_c \mathsf{B} = \mathsf{ten}(\mathsf{mat}(\mathsf{ffi})\mathsf{mat}(\mathsf{B})).$$

Notice that from the relation (6.3), we can show that the product $\mathsf{C} = \mathsf{ffi} *_c \mathsf{B}$ is equivalent to $\mathbf{C} = \mathbf{A}\,\mathbf{B}$. The following algorithm allows us to compute, in an efficient way, the c-product of the tensors ffi and B, see [88].

For the c-product, we have the following definitions and remarks

Definition 6.3. 1. The identity tensor $\mathsf{I}_{n_1 n_1 n_3}$ is the tensor such that all frontal slice of $\tilde{\mathsf{I}}_{n_1 n_1 n_3}$ is the identity matrix $I_{n_1 n_1}$.

Algorithm 22 Computing the c-product

Inputs: $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times m \times n_3}$

Output: $\mathcal{C} = \mathcal{A} *_c \mathcal{B} \in \mathbb{R}^{n_1 \times m \times n_3}$

1. Compute $\tilde{\mathcal{A}} = \mathrm{dct}(\mathcal{A}, [\,], 3)$ and $\tilde{\mathcal{B}} = \mathrm{dct}(\mathcal{B}, [\,], 3)$.
2. Compute each frontal slices of $\tilde{\mathcal{C}}$ by

$$C^{(i)} = A^{(i)} B^{(i)}$$

3. Compute $\mathcal{C} = \mathbf{idct}(\tilde{\mathcal{C}}, [\,], 3)$.

2. An $n_1 \times n_1 \times n_3$ tensor $\mathcal{A}$ is invertible, if there exists a tensor $\mathcal{B}$ of order $n_1 \times n_1 \times n_3$ such that

$$\mathcal{A} *_c \mathcal{B} = \mathcal{I}_{n_1 n_1 n_3} \qquad \text{and} \qquad \mathcal{B} *_c \mathcal{A} = \mathcal{I}_{n_1 n_1 n_3}.$$

In that case, we set $\mathcal{B} = \mathcal{A}^{-1}$. It is clear that $\mathcal{A}$ is invertible if and only if $\mathrm{mat}(\mathcal{A})$ is invertible.

3. The inner scalar product is defined by

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \sum_{i_3=1}^{n_3} a_{i_1 i_2 i_3} b_{i_1 i_2 i_3}$$

and he corresponding norm is given by $\|\mathcal{A}\|_F = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$.

4. An $n_1 \times n_1 \times n_3$ tensor $\mathcal{Q}$ is orthogonal if $\mathcal{Q}^T *_c \mathcal{Q} = \mathcal{Q} *_c \mathcal{Q}^T = \mathcal{I}_{n_1 n_1 n_3}$.

We now introduce the new c-diamond tensor-tensor product.

Definition 6.4. Let $\mathcal{A} = [\mathcal{A}_1, \ldots, \mathcal{A}_p] \in \mathbb{R}^{n_1 \times ps \times n_3}$, where $\mathcal{A}_i \in \mathbb{R}^{n_1 \times s \times n_3}$, $i = 1, \ldots, p$ and let $\mathcal{B} = [\mathcal{B}_1, \ldots, \mathcal{B}_l] \in \mathbb{R}^{n_1 \times ls \times n_3}$ with $\mathcal{B}_j \in \mathbb{R}^{n_1 \times s \times n_3}$, $j = 1, \ldots l$. Then, the product $\mathcal{A}^T \diamond_c \mathcal{B}$ is the $p \times l$ matrix given by :

$$(\mathcal{A}^T \diamond_c \mathcal{B})_{i,j} = \langle \mathcal{A}_i, \mathcal{B}_j \rangle \ .$$

6.4 Tensor discrete cosine global Krylov subspace methods

In this section, we propose iterative methods based on tensor cosine global Arnoldi and cosine global Golub– Kahan bidiagonlization (cosine-GGKB), combined with Tikhonov regularization, to solve some discrete ill posed problems. We consider the following discrete ill-posed tensor equation

$$\mathcal{A} *_c \mathcal{X} = \mathcal{C}, \quad \mathcal{C} = \hat{\mathcal{C}} + \mathcal{N}, \tag{6.5}$$

where $\mathcal{A} \in \mathbb{R}^{n \times m \times p}$, $\mathcal{X}$, $\mathcal{N}$ (additive noise) and $\mathcal{C}$ are tensors in $\mathbb{R}^{n \times s \times p}$. In color image processing, $p = 3$, $\mathcal{A}$ represents the blurring tensor, $\mathcal{C}$ the blurry and noisy observed image, $\mathcal{X}$ is the image that we would like to restore and $\mathcal{N}$ is an unknown additive noise. Therefore, to stabilize the recovered im- age, regularization techniques are needed. There are several techniques to regularize the linear inverse problem given by equation (6.5); for the matrix case, see for example, [17, 35, 60, 67]. All of these techniques stabilize the restoration process by adding a regularization term, depending on some priori knowledge of the unknown image. One of the most regularization method is due to Tikhonov and is given as follows

$$\min_{\mathcal{X}} \{ \|\mathcal{A} *_c \mathcal{X} - \mathcal{C}\|_F^2 + \lambda \|\mathcal{X}\|_F^2 \}. \tag{6.6}$$

Many techniques for choosing a suitable value of λ have been analysed and illustrated in the literature; see, e.g., [60, 67, 138] and references therein. In this paper we will use the discrepancy principle and the Generalized Cross Validation (GCV) techniques.

6.4.1 The tensor discrete cosine GMRES

Let $\mathcal{A} \in \mathbb{R}^{n \times n \times p}$ and $\mathcal{V} \in \mathbb{R}^{n \times s \times p}$. We introduce the tensor Krylov subspace $\mathcal{TK}_m^c(\mathcal{A}, \mathcal{V})$ associated to the cosine-product, defined for the pair $(\mathcal{A}, \mathcal{V})$ as follows

$$\mathcal{TK}_m^c(\mathcal{A}, \mathcal{V}) = \text{Tspan}\{\mathcal{V}, \mathcal{A} *_c \mathcal{V}, \ldots, \mathcal{A}^{m-1} *_c \mathcal{V}\} = \left(\mathcal{Z} \in \mathbb{R}^{n \times s \times p}, \mathcal{Z} = \sum_{i=1}^{m} \alpha_i \mathcal{A}^{i-1} *_c \mathcal{V} \right), \tag{6.7}$$

where $\alpha_i \in \mathbb{R}$, $\mathcal{A}^{i-1} *_c \mathcal{V} = \mathcal{A}^{i-2} *_c \mathcal{A} *_c \mathcal{V}$, for $i = 2, \ldots, m$ and $\mathcal{A}^0$ is the identity tensor. In the following algorithm, we define the Tensor Discrete Cosine Arnoldi (DCA) algorithm.

Algorithm 23 Tensor discrete cosine Arnoldi

1. **Input.** $\mathcal{A} \in \mathbb{R}^{n \times n \times p}$, $\mathcal{V} \in \mathbb{R}^{n \times s \times p}$ and the positive integer m.
2. Set $\beta = \|\mathcal{V}\|_F$, $\mathcal{V}_1 = \dfrac{\mathcal{V}}{\beta}$
3. For $j = 1, \ldots, m$
   - *(a)* $\mathcal{W} = \mathcal{A} *_c \mathcal{V}_j$
   - *(b)* for $i = 1, \ldots, j$
     - i. $h_{i,j} = \langle \mathcal{V}_i, \mathcal{W} \rangle$
     - *ii.* $\mathcal{W} = \mathcal{W} - h_{i,j} \mathcal{V}_i$
   - (c) End for
   - (d) $h_{j+1,j} = \|\mathcal{W}\|_F$. If $h_{j+1,j} = 0$, stop; else
   - (e) $\mathcal{V}_{j+1} = \mathcal{W}/h_{j+1,j}$.
4. End

It is not difficult to show that after m steps of Algorithm 23, the tensors $\mathcal{V}_1, \ldots, \mathcal{V}_m$ form an orthonormal basis of the tensor Krylov subspace $\mathcal{TK}_m(\mathcal{A}, \mathcal{V})$. Let $\mathcal{V}_m$ be the $(n \times (sm) \times p)$ tensor with frontal slices $\mathcal{V}_1, \ldots, \mathcal{V}_m$ and let $\tilde{H}_m$ be the $(m+1) \times m$ upper Hesenberg matrix whose elements are the $h_{i,j}$'s defined by Algorithm 23. Let H_m be the matrix obtained from $\tilde{H}_m$ by deleting its last row; $H_{.,j}$ will denote the j-th column of the matrix H_m and $\mathcal{A} *_c \mathcal{V}_m$ is

the $(n \times (sm) \times p)$ tensor with frontal slices $\text{ffi} *_c \mathcal{V}_1, \ldots, \text{ffi} *_c \mathcal{V}_m$:

$$\mathcal{V}_m := [\mathcal{V}_1, \ldots, \mathcal{V}_m] \quad \text{and} \quad \text{ffi} *_c \mathcal{V}_m := [\text{ffi} *_c \mathcal{V}_1, \ldots, \text{ffi} *_c \mathcal{V}_m]. \tag{6.8}$$

We introduce the product ② defined by

$$\mathcal{V}_m ② y = \sum_{j=1}^{m} y_j \mathcal{V}_j, \quad y = (y_1, \ldots, y_m)^T \in \mathbb{R}^m, \text{ and } \mathcal{V}_m ② H_m = [\mathcal{V}_m ② H_{:,1}, \ldots, \mathcal{V}_m ② H_{:,m}].$$

With the above notations, we can easily prove the results of the following proposition.

Proposition 6.5. *Suppose that m steps of Algorithm 23 have been run. Then, the following statements hold:*

$$\text{ffi} *_c \mathcal{V}_m = \mathcal{V}_m ② H_m + h_{m+1,m} \mathcal{O}_{n \times s \times p}, \ldots, \mathcal{O}_{n \times s \times p}, \mathcal{V}_{m+1} \tag{6.9}$$

$$\text{ffi} *_c \mathcal{V}_m = \mathcal{V}_{m+1} ② \tilde{H}_m, \tag{6.10}$$

$$\mathcal{V}_m^T ◆_c \text{ffi} *_c \mathcal{V}_m = H_m, \tag{6.11}$$

$$\mathcal{V}_{m+1}^T ◆_c \text{ffi} *_c \mathcal{V}_m = \tilde{H}_m, \tag{6.12}$$

$$\mathcal{V}_m^T ◆_c \mathcal{V}_m = I_m, \tag{6.13}$$

$$\|\mathcal{V}_m ② y\|_F = \|y\|_2, \ y \in \mathbb{R}^m, \tag{6.14}$$

where I_m the identity matrix and $\mathcal{O}_{n \times s \times p}$ is the tensor of size $(n \times s \times p)$ having all its entries equal to zero.

In the sequel, we briefly present Restarted tensor Discrete Cosine GMRES (DC-GMRES) algorithm to solve the problem (6.6). Let $\mathcal{X}_0 \in \mathbb{R}^{n \times s \times p}$ be an arbitrary initial guess with the corresponding residual $\mathcal{R}_0 = \mathcal{C} - \text{ffi} *_c \mathcal{X}_0$. The aim of DC-GMRES method is to find and approximate solution $\mathcal{X}_m$ approximating the exact solution $\mathcal{X}^*$ such that

$$\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m ② y, \tag{6.15}$$

where $y = y_{m,\lambda_m} \in \mathrm{R}^m$ solves the projected regularized minimization problem

$$y_{m,\lambda_m} = \arg\min_{y \in \mathrm{R}^m} \|\beta e_1 - \tilde{H}_m y\|_2^2 + \lambda^2 \|y\|_2^2, \tag{6.16}$$

$$= \arg\min_{y \in \mathrm{R}^m} \left\| \begin{bmatrix} \tilde{H}_m \\ \lambda_m I_m \end{bmatrix} y - \begin{bmatrix} \beta e_1 \\ 0 \end{bmatrix} \right\|_2^2, \tag{6.17}$$

where $\beta = \|\mathsf{R}_0\|$ and e_1 the first canonical basis vector in R^{m+1}. The min- imizer y_{m,λ_m} can also be computed as the solution of the following normal equations associated with (6.17)

$$\tilde{H}_{m,\lambda_m} y = \tilde{H}_m^T \beta e_1, \quad \tilde{H}_{m,\lambda_m} = (\tilde{H}_m^T \tilde{C}_m + \lambda_m^2 I_m). \tag{6.18}$$

Note that since the Tikhonov problem (6.18) is now a matrix one with small dimension as m is generally small, λ_m, can thereby be inexpensively computed by some techniques such as the GCV method [60] or the L-curve criterion [35, 67]. In this paper we consider the generalized cross-validation (GCV) method to choosing the regularization parameter [60, 138]. We take advantage of the SVD decomposition of the low dimensional matrix $\tilde{H}_m$ to obtain a more simple and computable expression of $GCV(\lambda_m)$. Consider the SVD decomposition $\tilde{C}_k = U \Sigma V^T$. Then, the GCV function can be expressed as (see [138])

$$GCV(\lambda_m) = \frac{\sum_{i=1}^{m} \left(\frac{\tilde{g}_i}{\sigma_i^2 + \lambda_m^2} \right)^2}{\left(\sum_{i=1}^{m} \frac{1}{\sigma_i^2 + \lambda_m^2} \right)^2}, \tag{6.19}$$

where σ_i is the ith singular value of the matrix $\tilde{H}_m$ and $\tilde{g} = \beta_1 U^T e_1$. The restarted tensor discrete cosine GMRES algorithm is summarized as follows:

Algorithm 24 Restarted tensor discrete cosine GMRES (DC-GMRES(m)) method with Tikhonov regularization

1. **Input.** $\mathcal{A} \in \mathbb{R}^{n\times n\times p}$, $\mathcal{C}, \mathcal{X}_0 \in \mathbb{R}^{n\times s\times p}$, an integer m for restarting, a maximum number of iterations $\text{Iter}_{\max}$ and a tolerance $tol > 0$.
2. **Output.** $\mathcal{X}_m \in \mathbb{R}^{n\times s\times p}$ approximate solution of the system (6.1).
3. Set $k = 0$ and compute $\mathcal{R}_0 = \mathcal{C} - \mathcal{A} *_c \mathcal{X}_0$..
4. Apply Algorithm 23 to the pair $(\mathcal{A}, \mathcal{R}_0)$ to compute $\mathcal{V}_m$ and $\tilde{H}_m$.
5. Determine λ_m as the parameter minimizing the GCV function given by (6.19)
6. Compute the regularized solution y_{m,λ_m} of the problem (6.17).
7. Compute the approximate solution $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m \textcircled{2} y_{m,\lambda_m}$
8. If $\|\mathcal{R}_m\|_F < tol$ or $k > itermax$, stop,
   else Set $\mathcal{X}_0 = \mathcal{X}_m, k = k + 1$ and go to Step 4.

6.4.2 The tensor discrete cosine Golub-Kahan method

We consider the tensor least squares problem

$$\min_{\mathcal{X}} \{\|\mathcal{A} *_c \mathcal{X} - \mathcal{C}\|^2, \ _F \tag{6.20}$$

where $\mathcal{A} \in \mathbb{R}^{n\times l\times p}$ and $\mathcal{C} \in \mathbb{R}^{n\times s\times p}$. Instead of using the DCA, we can use a dicrete cosine version of the tensor Lanczos process to generate a new basis that can be used for the projection. We will use the tensor Golub Kahan algorithm related to the c-product. defined as follows.

Algorithm 25 The Tensor discrete cosine Bidiagonalisation Golub Kahan algorithm

1. **Input.** The tensors $\mathcal{A}$, $\mathcal{C}$ and an integer m.
2. Set $\beta_1 = \|\mathcal{C}\|_F$, $\alpha_1 = \|\mathcal{A}^T *_c \mathcal{U}_1\|_F$, $\mathcal{U}_1 = \mathcal{C}/\beta_1$ and $\mathcal{V}_1 = (\mathcal{A}^T *_c \mathcal{U}_1)/\alpha_1$. 3.

for $j = 1, \ldots, m$

(a) $\tilde{\mathcal{U}} = \mathcal{A} *_c \mathcal{V}_j - \alpha_j \mathcal{U}_j$

(b) $\beta_{j+1} = \|\tilde{\mathcal{U}}\|_F$

(c) $\mathcal{U}_{j+1} = \tilde{\mathcal{U}}/\beta_{j+1}$

(d) $\tilde{\mathcal{V}} = \mathcal{A}^T *_c \mathcal{U}_{j+1} - \beta_{j+1}\mathcal{V}_j$

(e) $\alpha_{j+1} = \|\tilde{\mathcal{V}}\|_F$

(f) $\mathcal{V}_{j+1} = \tilde{\mathcal{V}}/\alpha_{j+1}$.

Let $\tilde{C}_m$ be the upper bidiagonal $((m+1) \times m)$ matrix

$$\tilde{C}_m = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \\ & & \beta_m & \alpha_m \\ & & & \beta_{m+1} \end{bmatrix}$$

and let C_m be the $(m \times m)$ matrix obtain by deleting the last row of $\tilde{C}_m$. We denote by $C_{.,j}$ the j-th column of the matrix C_m. Let $\mathcal{V}_m$ and $\mathcal{U}_m$ be the $(I \times (sm) \times p)$ and $(n \times (sm) \times p)$ tensors with frontal slices $V_1, \ldots, V_m$ and $\mathcal{U}_1, \ldots, \mathcal{U}_m$, respectively, and let $\mathcal{U}_m$ and $\mathcal{U}_m^T$ be the $(n \times (sm) \times p)$ and $(I \times (sm) \times p)$ tensors with frontal slices $U_1, \ldots, U_m$ and $\mathcal{U}_1, \ldots, \mathcal{U}_m$, respectively. We set

$$\mathcal{U}_m := [U_1, \ldots, U_m], \quad \text{and} \quad \mathcal{U} *_c \mathcal{V}_m := [\mathcal{U} *_c V_1, \ldots, \mathcal{U} *_c V_m], \tag{6.21}$$

$$\mathcal{V}_m := [V_1, \ldots, V_m], \quad \text{and} \quad \mathcal{U}^T *_c \mathcal{U}_m := [\mathcal{U}^T *_c U_1, \ldots, \mathcal{U}^T *_c U_m]. \tag{6.22}$$

Proposition 6.6. *The tensors produced by the tensor cosine Golub-Kahan algorithm satisfy the following relations*

$$\mathcal{U} *_c \mathcal{V}_m = \mathcal{U}_{m+1} ② \tilde{C}_m, \tag{6.23}$$

$$\mathcal{U}^T *_c \mathcal{U}_m = \mathcal{V}_m ② \tilde{C}_m^T, \tag{6.24}$$

$$\mathcal{U}_{m+1} ② (\beta_1 e_1) = \mathcal{C}, \tag{6.25}$$

$$\|\mathcal{U}_{m+1} ② z\|_F = \|z\|_2, \tag{6.26}$$

where $e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^{m+1}$ *and* z *is a vector of* $\mathbb{R}^{m+1}$.

To solve the least squares problem (6.20), we consider approximations defined as

$$\mathcal{X}_m = \mathcal{V}_m ② y_m, \tag{6.27}$$

satisfying the minimization property of the corresponding residual. As we explained earlier, the problems that we are concerned with are ill-posed problems and then regularization techniques are highly recommended in those cases. But as the problem is very large, we apply the regularization process to the projected problem derived from the minimization of the residual. This leads to a low dimensional Tikhonov formulation and then we seek for $y = y_m \in \mathrm{R}^m$ that solves the low dimensional linear system of equations

$$(\tilde{C}_m^T \tilde{C}_m + \lambda_m^2 I_m) y = \alpha_1 \tilde{C}_m^T e_1, \qquad \alpha_1 = \|\mathsf{C}\|_F, \tag{6.28}$$

which is also equivalent to solving the least-squares problem

$$\min_{y \in \mathrm{R}^m} \left\| \begin{bmatrix} \tilde{C}_m \\ \lambda_m I_m \end{bmatrix} y - \alpha_1 \begin{bmatrix} e_1 \\ 0 \end{bmatrix} \right\|_2 \tag{6.29}$$

The regularized parameter λ_m is computed by using the GCV function given by (6.19). The following algorithm summarizes the main steps of the described method.

Algorithm 26 The Tensor Discrete Cosine Golub-Kahan (DC-GK) method

1. **Input.** The tensors A, C.
2. Determine the orthonormal bases U_{m+1} and V_m of tensors, and the bidiagonal C_m and $\tilde{C}_m$ matrices with Algorithm 25.
3. Determine λ_m using GCV function.
4. Determine y_{m,λ_m} by solving (6.29) and then compute X_{m,λ_m} by (6.27).

In the next section, we derive a direct computation of the approximate Golub-Kahan solution by using a discrete cosine LSQR algorithm.

6.4.3 The discrete cosine-LQSR method

In this section, we develop the tensor version of the well know LSQR algorithm introduced in [61] based on c-product formalism. Let $\mathsf{A} \in \mathrm{R}^{n \times l \times p}$ be a tensor and let $\mathsf{C} \in \mathrm{R}^{n \times s \times p}$ a starting tensor.

The purpose of the Tensor Discrete LSQR (DC-LSQR) method is to find, at some step k, an approximation $\mathcal{X}_k$ of the solution $\mathcal{X}^*$ of the problem (6.20) such that

$$\mathcal{X}_k = \mathcal{V}_k \textcircled{2} y_k, \tag{6.30}$$

where $y_k \in \mathbb{R}^k$. The associated residual is given by

$$\mathcal{R}_k = \mathcal{C} - \mathcal{A} *_c \mathcal{X}_k = \beta_1 \mathcal{U}_1 - \mathcal{U}_{k+1} \textcircled{2} \tilde{C}_k \textcircled{2} y_k = \mathcal{U}_{k+1} \textcircled{2} (\beta_1 e_1 - \tilde{C}_k y_k) \tag{6.31}$$

and using Proposition 6.6, we get

$$\|\mathcal{U}_{k+1} \textcircled{2} (\beta_1 e_1 - \tilde{C}_k y_k)\|_F = \|\beta_1 e_1 - \tilde{C}_k y_k\|_2.$$

This minimization problem is accomplished by using the QR decomposition, where a unitary matrix Q_k is determined so that

$$Q_k[\tilde{C}_k \quad \beta_1 e_1] = \begin{bmatrix} R_k & f_k \\ 0 & \bar{\varphi}_{k+1} \end{bmatrix} = \begin{bmatrix} \rho_1 & \theta_2 & & & \varphi_1 \\ & \rho_2 & \theta_3 & & \vdots \\ & & \ddots & \ddots & \\ & & & \rho_{k-1} & \theta_k \\ & & & & \rho_k \\ & & & & \bar{\varphi}_{k+1} \end{bmatrix} \begin{matrix} \varphi_{k-1} \\ \varphi_k \end{matrix},$$

where ρ_i, θi are scalars. The matrix Q_k is a product of plane rotations designed to eliminate the sub-diagonals of $\tilde{C}_k$. This gives the following simple recurrence relation :

$$\begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix} \begin{bmatrix} \bar{\rho}_k & 0 & \bar{\varphi}_k \\ \beta_{k+1} & \alpha_{k+1} & 0 \end{bmatrix} = \begin{bmatrix} \rho_k & \theta_{k+1} & \varphi_k \\ 0 & \bar{\rho}_{k+1} & \bar{\varphi}_{k+1} \end{bmatrix},$$

where $\bar{\rho}_1 = \alpha_1$ and $\bar{\varphi}_1 = \beta_1$ and the scalars s_k, c_k are the nontrivial element of $Q_{k+1,k}$ the k-th plane rotation. We get

$$R_k y_k = f_k$$

and the approximate solution is given by :

$$\mathsf{X}_k = (\mathrm{V}_k \textcircled{2} R_k^{-1}) \textcircled{2} f_k.$$

Let $\mathrm{V}_k \textcircled{2} R_k^{-1} = \mathrm{P}_k = [\mathsf{P}_1 \ldots \mathsf{P}_k]$, then we have

$$\mathsf{X}_k = \mathrm{P}_k \textcircled{2} f_k$$

Notice that the tensor P_k can be computed from P_{k-1} and V_k as follows :

$$\mathsf{P}_k = (\mathsf{V}_k - \theta_k \mathsf{P}_{k-1})\rho_k^{-1}.$$

We also have $f_k = \begin{bmatrix} f_{k-1} \\ \varphi_k \end{bmatrix}$ in which $\varphi_k = c_k \bar{\varphi}_k$. Finally, X_k can be computed as follows

$$\mathsf{X}_k = \mathsf{X}_{k-1} + \varphi_k \mathsf{P}_k.$$

Furthermore, we have

$$\|\mathsf{R}_k\|_F = |\bar{\varphi}_{k+1}|.$$

The next algorithm which is named Discrete Cosine LSQR (DC-LSQR) algorithm, describes the whole process.

For ill posed problems, as it is the case for image or video restorations, we could have situations where the residual norm is small enough but the error norm is still large. As it is observed for those problems, the residual and the error norms could decrease in DC-LSQR till some iteration k and then the norm of the error becomes to increase. One possibility to overcome these

Algorithm 27 The Discrete Cosine LSQR (DC-LSQR) algorithm

1. **Input.** The tensors $\mathcal{A}$, $\mathcal{C}$, $\mathcal{X}_0 = \mathcal{O}$, *itermax*, the maximum number of allowed iterations and a tolerance $tol > 0$ for the stopping criterion.
2. Set $\beta_1 = \|\mathcal{C}\|_F$, $\alpha_1 = \|\mathcal{A}^T *_c \mathcal{U}_1\|_F$, $\mathcal{U}_1 = \mathcal{C}/\beta_1$ and $\mathcal{V}_1 = (\mathcal{A}^T *_c \mathcal{U}_1)/\alpha_1$, $\mathcal{W}_1 = \mathcal{V}_1$, $\bar{\rho}_1 = \alpha_1$ and $\bar{\varphi}_1 = \beta_1$
3. for $j = 1, \ldots, itermax$
   (a) $\mathcal{W}_j = \mathcal{A} *_c \mathcal{V}_j - \alpha_j \mathcal{U}_j$, $\beta_{j+1} = \|\mathcal{W}_j\|_F$ and $\mathcal{U}_{j+1} = \mathcal{W}_j/\beta_{j+1}$.
   (b) $\mathcal{V}_j = (\mathcal{A}^T *_c \mathcal{U}_{j+1}) - \beta_{j+1}\mathcal{V}_j$, $c_j = \frac{\rho_j}{\bar{\rho}_j}$, $\alpha_{j+1} = s_j \alpha_{j+1}$ and $\mathcal{V}_{j+1} = \mathcal{V}_j/\alpha_{j+1}$.
   (c) $\rho_j = (\bar{\rho}_j^2 + \beta_{j+1}^2)^{\frac{1}{2}}$, $c_j = \frac{\bar{\rho}_j}{\rho_j}$, $s_j = \frac{\beta_{j+1}}{\rho_j}$ and $\bar{\rho}_{j+1} = -c_j \alpha_{j+1}$.
   (d) $\theta_{j+1} = s_j \alpha_{j+1}$ and $\bar{\rho}_{j+1} = c_j \alpha_{j+1}$.
   (e) $\varphi_j = c_j \bar{\varphi}_j$ and $\bar{\varphi}_{j+1} = -s_j \bar{\varphi}_j$.
   (f) $\mathcal{X}_i = \mathcal{X}_{i-1} + \frac{\varphi_j}{\rho_j}\mathcal{W}_j$; $\mathcal{W}_{j+1} = \mathcal{V}_{j+1} - \frac{\theta_{j+1}}{\rho_j}\mathcal{W}_j$.
   (g) If $|\bar{\varphi}_{j+1}| < tol$ stop.

situations is to stop the iterations at some optimal k_{opt}. The L-curve criterion [35, 67] could be usefull to determine such optimal index k_{opt}. The method suggests to plot the curve $(\|\mathcal{R}_k\|, \|\mathcal{X}_k\|)$. Intuitively, the best regularization parameter should lie on the corner of the L-curve corresponding to the point on the curve with maximum curvature.

6.5 Numerical results

In this section, we give some numerical tests on the methods described in this paper. We compared the performances of the tensor discrete cosine GMRES describes in Algorithm 24, the tensor discrete cosine Golub-Kahan (DC-GK) algorithm given by Algorithm 26 and the tensor dicrete cosine LSQR (DC-LSQR) described in Algorithm 27 , when applied to the restoration of blurred and noisy color images. All computations were carried out using the Matlab environment on an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz (8 CPUs) computer with 12 GB of RAM. The computations were done with approximately 15 decimal digits of relative accuracy. Let $\hat{X}^{(1)}$, $\hat{X}^{(2)}$, and $\hat{X}^{(3)}$ be the $n \times n$ matrices that constitute the three channels of the original error-free color image $\hat{\mathcal{X}}$, and $\hat{C}^{(1)}$, $\hat{C}^{(2)}$, and $\hat{C}^{(3)}$ the $n \times n$ matrices associated with error-free blurred color image $\hat{\mathcal{C}}$. We consider that both cross-channel

and within-channel blurring take place in the blurring process of the original image. The full blurring model

$$\mathcal{ffi} *_c \hat{\mathcal{X}} *_c \mathcal{B} = \hat{\mathcal{C}}, \tag{6.32}$$

where $\mathcal{ffi}$ is a 3-way tensor such that $\mathcal{ffi}(:,:,1) = \alpha\mathbf{A}^{(2)}$, $\mathcal{ffi}(:,:,2) = \beta\mathbf{A}^{(2)}$ and $\mathcal{ffi}(:,:,3) = \gamma\mathbf{A}^{(2)}$ and $\mathcal{B}$ is a 3-way tensor with $\mathcal{B}(:,:,1) = (\mathbf{A}^{(1)})^T$, $\mathcal{B}(:,:,2) = 0$ and $\mathcal{B}(:,:,3) = 0$. α, β and γ are the entries of the following matrix

$$\mathbf{A}_{\text{color}} = \begin{bmatrix} \alpha & \gamma & \beta \\ \beta & \alpha & \gamma \\ \gamma & \beta & \alpha \end{bmatrix},$$

obtained from [68], that models the cross-channel blurring, in which each row sums is one. $\mathbf{A}^{(1)} \in \mathrm{R}^{n\times n}$ and $\mathbf{A}^{(2)} \in \mathrm{R}^{n\times n}$ define within-channel blurring and they model the horizontal within blurring and the vertical within blurring matrices, respectively; To test the performance of algorithms, the within blurring matrices $A^{(i)}$ have the following entries

$$a_{kl} = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(k-l)^2}{2\sigma^2}\right), & |k-l| \le r \\ 0, & \text{otherwise.} \end{cases}$$

Note that σ controls the amount of smoothing, i.e. the larger the σ, the more ill posed the problem. We generated a blurred and noisy tensor image $\mathcal{C} = \hat{\mathcal{C}} + \mathcal{N}$, where $\mathcal{N}$ is a noise tensor with normally distributed random entries with zero mean and with variance chosen to correspond to a specific noise level $\nu := \|\mathcal{N}\|_F / \|\mathcal{C}\|_F$. To compare the effectiveness of our solution methods, we evaluate

$$\text{Relative error} = \frac{\|\hat{\mathcal{X}} - \mathcal{X}_{\text{restored}}\|_F}{\|\hat{\mathcal{X}}\|_F}$$

Figure 6.1: *Original RGB images:* papav256 *(left) and* cat1024 *(right).*

and the Signal-to-Noise Ratio (SNR) defined by

$$\mathrm{SNR}(\mathsf{X}_{\mathrm{restored}}) = 10\log_{10} \frac{\|\hat{\mathsf{X}} - E(\hat{\mathsf{X}})\|_F^2}{\|\mathsf{X}_{\mathrm{restored}} - \hat{\mathsf{X}}\|_F^2},$$

where $E(\hat{\mathsf{X}})$ denotes the mean gray-level of the uncontaminated image $\hat{\mathsf{X}}$. In our experiments, we applied the three algorithms DC-GMRES(10), DC-GK and DC-LSQR for the reconstruction of a cross-channel blurred color images that have been contaminated by both within and cross blur, and additive noise. The cross-channel blurring is determined by the matrix

$$\mathbf{A}_{\mathrm{color}} = \begin{bmatrix} 0.8 & 0.10 & 0.10 \\ 0.10 & 0.80 & 0.10 \\ 0.10 & 0.10 & 0.80 \end{bmatrix}.$$

We consider two RGB images, papav256 ($\hat{\mathsf{X}} \in \mathrm{R}^{256\times 256\times 3}$) and cat1024 ($\hat{\mathsf{X}} \in \mathrm{R}^{1024\times 1024\times 3}$). They are shown on Figure 6.1. For the within-channel blur- ring, we let $\sigma = 4$ and $r = 6$. The associated blurred and noisy RGB images are obtained as $\mathsf{C} = \mathsf{ffi} * \hat{\mathsf{X}} * \mathsf{B} + \mathsf{N}$. Given the contaminated RGB image C, we would like to recover an approximation of the original RGB image $\hat{\mathsf{X}}$.

6.5.1 Example 1

In the first experiment, we used the papav256 color image of size $256 \times 256 \times 3$ with two different noise levels $v = 10^{-2}$ or $v = 10^{-3}$. In Table 6.1 we reported the obtained SNR, the corresponding relative error norm and the

required cpu-time for DC-GMRES(10), DC-GK and DC-LSQR with a noise level of 10^{-3}.

Table 6.1: *Results for Experiment 1 with* papav256*. Noise level* 10^{-3}*.*

RGB images	Method	SNR	Relative error	cpu-time (seconds)
papav256	DC-GMRES(10)	19.25	8.5×10^{-2}	5.21
	DC-GK	23.9	4.7×10^{-2}	1.62
	DC-LSQR	21.8	6.8×10^{-2}	1.23

As can be seen from those results, the DC-LSQR requires lower cpu-time as compared to the other two methods. However, DC-GK returns the best SNR. For this experiment, the optimal iteration number was $k_{opt} = 14$. A maximum number of iterations was $itermax = 10$ for DC-GMRES(10) and $mmax = 15$ for DC-GK. As we mentioned earlier, DC-GMRES(10) and DC-GK were run with the Tikhonov regularization technique (applied to the projected least squares problem) and we used GCV method for estimating the regularization parameters in each iteration of the processes. The obtained optimal parameters, at the final step were $\lambda_1 = 2.32 \times 10^{-5}$ for DC-GMRES(10) and $\lambda_1 = 1.24 \times 10^{-6}$ for DC-GK. Figure 6.2 shows the ob- tained blurred image and the restored one when using DC-LSQR method with noise level of 10^{-3}.

In Table 6.2, we reported the results obtained by DC-GMRES(10), DC-GK and DC-LSQR for the color image papav256 with a noise level of 10^{-2}. Here also, we used $k_{opt} = 15$ for DC-LSQR, $itermax = 15$ for DC-GMRES(10) and $mmax = 20$ for DC-GK. As can be seen, the DC-LSQR returns the best results when comparing the three methods.

Table 6.2: *Results for Example 1 with* papav256*. Noise level* 10^{-2}*.*

RGB image	Method	SNR	Relative error	cpu-time (seconds)
papav256	DC-GMRES(10)	17.24	1.5×10^{-2}	7.14
	DC-GK	20.4	7.2×10^{-2}	1.92
	DC-LSQR	20.2	8.5×10^{-2}	1.23

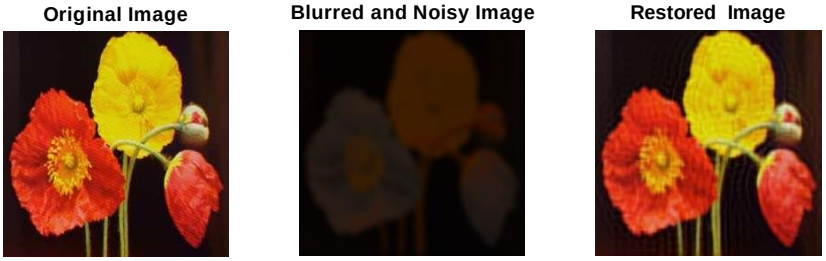


Figure 6.2: *Test for Example 1, with DC-LSQR for* papav256*, and noise level* 10^{-3}*. Original (left), noisy-blurred (center) and restored (right) .*

6.5.2 Example 2

In the second example, we used the color image cat1024 of dimension $1024 \times 1024 \times 3$. Here also, we compared the three methods using two noise levels $v = 10^{-3}$ and $v = 10^{-2}$. Table 6.3 reports on the obtained results for the noise level $v = 10^{-3}$. For this experiment, the optimal iteration number for DC-LSQR was 15, the maximum iteration number allowed to DC-GMRES(10) was 10 and a maximum of $mmax = 20$ iterations was for DC-BK. As can be seen from the obtained results, DC-LSQR returns the best results: for SNR and the total cpu-time.

Table 6.3: *Results for Example 2 with noise level* 10^{-3}*.*

RGB image	Method	SNR	Relative error	cpu-time (seconds)
cat1024	DC-GMRES(10)	14.96	4.54×10^{-2}	100.35
	DC-GK	19.25	6.97×10^{-2}	25.33
	DC-LSQR	18.87	5.43×10^{-2}	19.45

Figure 6.3: *Test for Example 2, with DC-LSQR for* cat1024*, and noise level* 10^{-3}*. Original (left), noisy-blurred (center) and restored (right).*

Figure 6.3 shows the obtained blurred image and the restored one when using DC-LSQR method with noise level of 10^{-3} for the image cat1024. Table 6.4 reports on the obtained results for the noise level $v = 10^{-2}$. For this experiment, the optimal iteration number for DC-LSQR was 20, the maximum iteration number allowed to DC-GMRES(10) was 15 and a maximum of $mmax = 25$ iterations was for DC-BK. As can be seen from this table, DC-LSQR returns the best results: for SNR and the total cpu-time. For the returned SNR, generally the two Golub Kahan based methods return similar results but the second formulation of the method which corresponds the DC-LSQR (Algorithm 27) requires less cpu-time.

Table 6.4: *Results for Example 2 with noise level* 10^{-2}*.*

RGB image	Method	SNR	Relative error	cpu-time (seconds)
cat1024	DC-GMRES(10)	14.53	9.62×10^{-2}	137.43
	DC-GK	15.87	8.06×10^{-2}	30.22
	DC-LSQR	15.75	8.17×10^{-2}	26.43

We also compared the performances of DC-Golub-Kahan with the T-Golub Kahan method that uses the tensor T-product. Usually, the approximate solution with the cosine product are better. In Table 6.5, we compared the obtained results for DC-GK and the T-GK algorithms for the image cat1024. We also run experiments comparing DC-LSQR and T-LSQR (which is obtained by replacing inDC-LSQR the c-product by the T-product) for the same example and the obtained results are also reported in Table 6.5.

Table 6.5: *Results for DC-Golub-Kahan, T-Golub Kahan, T-LSQR and DC-LSQR, with noise level* 10^{-2} *with the image* cat1024 *.*

Method	SNR	Relative error	cpu-time (seconds)
DC-GK	15.65	6.18×10^{-2}	30.46
T-GK	15.06	8.17×10^{-2}	36.26
T-LSQR	15.53	4.07×10^{-2}	28.35
DC-LSQR	15.85	1.32×10^{-2}	25.05

6.6 Conclusion

In this paper, we presented three discrete cosine Krylov-based methods, namely tensor DC-GMRES, DC-GK and DC-LSQR. The second two methods use the discrete cosine Golub-Kahan bidiagonalisation algorithm that we defined in this work. DC-GMRES and DC-GK are combined with the well known Tikhonov regularization method that is applied, at each iteration for the two algorithms, to the obtained projected low dimensional ill-posed least squares minimisation problem. The reported numerical tests show that the methods are very fact and can be used as restoration techniques for color image restoration.

Chapter 7

Tensor Krylov subspace methods via the T-product for large Sylvester Tensor Equations

In the present chapter, we introduce new tensor Krylov subspace methods for solving large Sylvester tensor equations. The proposed method uses the well- known T-product for tensors and tensor subspaces. We introduce some new tensor products and the related algebraic properties. These new products will enable us to develop third-order the tensor FOM (tFOM), GMRES (tGMRES), tubal bolck Arnoldi and the tensor tubal Block Arnoldi method to sole large Sylvester tensor equations. We give some properties related to these methods and propose some numerical experiments.

7.1 Introduction

The aim is to present numerical Tensor Krylov subspace methods for solving the Sylvester tensor equation

$$\mathcal{M}(\mathcal{X}) = \mathcal{C}, \tag{7.1}$$

where $\mathcal{M}$ is a linear operator that could be described as

$$\mathcal{M}(\mathcal{X}) = \mathcal{A} * \mathcal{X} - \mathcal{X} * \mathcal{B} \tag{7.2}$$

where $\mathcal{A}$, $\mathcal{X}$, $\mathcal{B}$ and $\mathcal{C}$ are three-way tensors, leaving the specific dimensions to be defined later. In the current paper, we are interested in developing robust and fast iterative Krylov subspace methods via T- product to solve the Sylvester tensor equation *STE* (7.1). In fact, when the tensors in equations (7.1) are of small sizes, the purpose is to extend matrix version of direct methods described in [8, 63] to third order tensors using the T-product formalism. This gives us the *t-Bartels-Stewart* algorithm. For large size tensors, we describe first new methods that will be defined as an orthogonal and oblique projection onto a tensor Krylov subspace. In particular, the tensor Full orthogonalization method (tFOM) and tensor generalized minimal residual method (tGMRES) will be examined. We will also introduce the well-Know tensor Tubal block Krylov methods via T-product for transforming the original large Sylvester equation to a low-dimensional Sylvester tensor equation. In particular, we will describe the Tubal Block Arnoldi (TBA) as a generalization of block Arnoldi matrix.

The paper is organized as follows: In Section 7.2, we give some notations and definitions. In Section 7.3, we develop the (tFOM) and (tGMRES) methods. the *t-Bartels-Stewart* method will be introduced in section 7.4. After defining a tubal QR factorization algorithm we defined in Section 7.5, we will establish the tensor TBA that allows us to introduce the t n TBAS method. Finally,

some numerical tests are reported in Section 7.6.

7.2 Definitions and notations

All definitions and properties related to the T-product are defined earlier; see chapter 3 section 3.2.

We consider the following Sylvester tensor equation

$$\mathcal{A} * \mathcal{X} + \mathcal{X} * \mathcal{B} = \mathcal{C} \tag{7.3}$$

Where $\mathcal{A} \in \mathbb{R}^{n \times n \times n3}$, $\mathcal{B} \in \mathbb{R}^{q \times q \times n3}$ and $\mathcal{X}, \mathcal{C} \in \mathbb{R}^{n \times q \times n3}$ respectively. $\mathcal{A}, \mathcal{B}$ and $\mathcal{C}$ are given, $\mathcal{X}$ is the unknown tensor to be determined.

Theorem 7.1. *Sylvester tensor equation (7.3) has a unique solution iif* $\Gamma(\mathbf{A}) \cap \Gamma(\mathbf{B}) = \emptyset$, *where the block diagonal matrix* $\mathbf{A}$, $\mathbf{B}$ *are defined by* (3.2), *and* $\Gamma(\mathbf{A})$, $\Gamma(\mathbf{B})$ *denotes the set of eigenvalues of the matrix* $\mathbf{A}$ *and* $\mathbf{B}$ *respectively.*

Proof. It is easy to show that the product $\mathcal{A} * \mathcal{X} + \mathcal{X} * \mathcal{B} = \mathcal{C}$ is equivalent to $\mathbf{C} = \mathbf{AX} + \mathbf{XB}$ (Sylvester matrix equation), which has a unique solution if and only if $\Gamma(\mathbf{A}) \cap \Gamma(\mathbf{B}) = \emptyset$. □

7.3 Tensor tFOM and tGMRES algorithms

7.3.1 The tFOM method

In the following, we examined the tensor full orthogonalization (tFOM) method, which is based on the tensor **T-global Arnoldi** process introduced in chapter 5 section 5.3 Algorithm 18. It could be considered as a generalization of the global FOM algorithm [79]. Let $\mathcal{X}_0 \in \mathbb{R}^{n \times s \times p}$ be an arbitrary initial guess with the corresponding residual $\mathcal{R}_0 = \mathcal{C} - \mathcal{M}(\mathcal{X}_0)$. The aim of the tensor T-global GMRES method is to find an approximate solution $\mathcal{X}_m$ approximating

the exact solution of $\mathcal{X}^*$ such that

$$\mathcal{X}_m - \mathcal{X}_0 = \mathcal{P}_m \in \mathcal{TK}_m(\mathbf{M}, \mathcal{R}_0), \tag{7.4}$$

and

$$\mathcal{R}_m \perp \mathcal{TK}_m(\mathbf{M}, \mathcal{R}_0) \tag{7.5}$$

$\mathcal{P}_m \in \mathcal{TK}_m(\mathbf{M}, \mathcal{R}_0)$ can be expressed as $\mathcal{P}_m = \mathcal{V}_m ② y$ with $y = (y_1, \ldots, y_m)^T \in \mathbb{R}^m$. From where the residual $\mathcal{R}_m$ is given by

$$\mathcal{R}_m = \mathcal{R}_0 - \mathcal{W}_m ② y \tag{7.6}$$

Using (7.6), the relation (7.5) can be expressed as

$$\langle \mathcal{V}_i, \mathcal{R}_m \rangle = \langle \mathcal{V}_i, \mathcal{R}_0 - \mathcal{W}_m ② y \rangle, \quad i = 1, \ldots, m \tag{7.7}$$

where he tensors $\mathcal{V}_1, \ldots, \mathcal{V}_m$, form an orthonormal basis of the tensor global Krylov subspace $\mathcal{TK}_m(\mathbf{M}, \mathcal{R}_0)$. The relation (7.7) can be expressed as :

$$H_m y = \|\mathcal{R}_0\|_F e_1^{(m)} \tag{7.8}$$

where $e_1^{(m)}$ is the first canonical basis vector in $\mathbb{R}^m$ and H_m is the Hessenberg matrix of size $(m \times m)$ obtained from Algorithm 18.

Proposition 7.2. *At step m, the norm of the residual* $\mathcal{R}_m = \mathcal{C} - \mathbf{M}(\mathcal{X}_m) = \mathcal{R}_0 - \mathcal{W}_m ② y_m$ *produced by the tFOM method for tensor equation (6.1) has the following expression*

$$\|\mathcal{R}_m\|_F = h_{m+1,m} \cdot y_m^{(m)} \cdot \ldots \tag{7.9}$$

where $y_m^{(m)}$ *is the last component of the vector* y_m

Proof. At step m, using the relations (7.6) and (7.5) the norm of the residual $\mathcal{R}_m$ can be expressed as

$$\|\mathcal{R}_m\|_F = \left\| \mathcal{R}_0 - \mathcal{V}_m \circledast (H_m y_m) + h_{m+1,m} \; \mathcal{O}_{n\times s\times p}, \ldots, \mathcal{O}_{n\times s\times p}, \mathcal{V}_{m+1} \; \circledast y_m \right\|_F$$

since $H_m y_m = \|\mathcal{R}_0\|_F \, e_1^{(m)}$, we get

$$\|\mathcal{R}_m\|_F = \left\| h_{m+1,m} \; \mathcal{O}_{n\times s\times p}, \ldots, \mathcal{O}_{n\times s\times p}, \mathcal{V}_{m+1} \; \circledast y_m \right\|_F = h_{m+1,m} \cdot y_m^{(m)} \cdot$$

which shows the results. □

7.3.2 The tGMRES method

In the sequel, we develop the tGMRES algorithm for solving the problem (7.3). It could be considered as a generalization of the global GMERS algorithm [79]. Let $\mathcal{X}_0 \in \mathbb{R}^{n\times s\times p}$ be an arbitrary initial guess with the corresponding residual $\mathcal{R}_0 = \mathcal{C} - \mathcal{M}(\mathcal{X}_0)$. The purpose of the tensor tGMRES method is to find an approximate solution $\mathcal{X}_m$ approximating the exact solution $\mathcal{X}^*$ of 7.1 such that

$$\mathcal{X}_m - \mathcal{X}_0 \in \mathcal{TK}_m(\mathcal{M}, \mathcal{R}_0), \tag{7.10}$$

with the classical minimization property

$$\|\mathcal{R}_m\|_F = \min_{\mathcal{X} \in \mathcal{X}_0 + \mathcal{TK}_m(\mathcal{M}, \mathcal{R}_0)} \|\mathcal{C} - \mathcal{M}(\mathcal{X})\|_F. \tag{7.11}$$

Let $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m \textcircled{2} y$ with $y \in \mathbb{R}^m$, be the approximate solution satisfying (7.10). Then,

$$\begin{aligned}\mathcal{R}_m &= \mathcal{C} - \mathcal{M}(\mathcal{X}_m),\\ &= \mathcal{C} - \mathcal{M}(\mathcal{X}_0 + \mathcal{V}_m \textcircled{2} y),\\ &= \mathcal{R}_0 - \mathcal{W}_m \textcircled{2} y.\end{aligned}$$

It follows then that

$$\|\mathcal{R}_m\|_F = \min_{y \in \mathbb{R}^m} \|\mathcal{R}_0 - \mathcal{W}_m \textcircled{2} y\|_F,$$

where $\mathcal{W}_m := [\mathcal{M}(\mathcal{V}_1), \ldots, \mathcal{M}(\mathcal{V}_m)]$ is the $(n \times sm \times p)$ tensor defined ear- lier. Using Proposition 5.2 and the fact that $\mathcal{R}_0 = \|\mathcal{R}_0\|_F \mathcal{V}_1$ with $\mathcal{V}_1 = \mathcal{V}_{m+1} \textcircled{2} e_1$, where e_1 the first canonical basis vector in $\mathbb{R}^{m+1}$, we get

$$\begin{aligned}\|\mathcal{R}_0 - (\text{ffi} * \mathcal{V}_m) \textcircled{2} y\|_F &= \|\mathcal{R}_0 - (\mathcal{V}_{m+1} \textcircled{2} \tilde{H}_m) \textcircled{2} y\|_F,\\ &= \|\|\mathcal{R}_0\|_F (\mathcal{V}_{m+1} \textcircled{2} e_1) - (\mathcal{V}_{m+1} \textcircled{2} \tilde{H}_m) \textcircled{2} y\|_F,\\ &= \|\mathcal{V}_{m+1} \textcircled{2} (\|\mathcal{R}_0\|_F e_1 - \tilde{H}_m y)\|_F,\\ &= \|\|\mathcal{R}_0\|_F\, e_1 - \tilde{H}_m y\|_2.\end{aligned}$$

Finally, we obtain

$$\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m \textcircled{2} y, \tag{7.12}$$

where,

$$y = \arg \min_{y \in \mathbb{R}^m} \|\|\mathcal{R}_0\|_F\, e_1 - \tilde{H}_m y)\|_2. \tag{7.13}$$

Proposition 7.3. *At step m, the residual* $\mathcal{R}_m = \mathcal{C} - \mathcal{M}(\mathcal{X}_m)$ *produced by the tensor global GMRES method for tensor equation (7.1) has the following expression*

$$\mathcal{R}_m = \mathcal{V}_{m+1} \textcircled{2} (\gamma_{m+1} Q_m e_{m+1}), \tag{7.14}$$

where Q_m *is the unitary matrix obtained from the QR decomposition of the upper*

Hessenberg matrix $\tilde{H}_m$ *and* γ_{m+1} *is the last component of the vector* $\|\mathsf{R}_0\|_F\, Q_m e_1^{\mathrm{T}}$ *and* $e_{m+1} = (0, 0, \ldots, 1)^{\mathrm{T}} \in \mathrm{R}^{m+1}$.

Furthermore,

$$\|\mathsf{R}_m\|_F = |\gamma_{m+1}|\,. \tag{7.15}$$

Proof. At step m, the residual R_m can be expressed as

$$\mathsf{R}_m = \mathrm{V}_{m+1} ② \quad \beta e_1 - \tilde{H}_m y_m \quad ,$$

by considering the QR decomposition $\tilde{H}_m = Q_m \tilde{U}_m$ of the $(m + 1) \times m$ matrix $\tilde{H}_m$, we get

$$\mathsf{R}_m = (\mathrm{V}_{m+1} ② Q_m) ② \quad \beta Q_m^T e_1 - \tilde{U}_m y_m \quad .$$

Since y solves problem (7.13), it follows that

$$\mathsf{R}_m = \mathrm{V}_{m+1} ② (\gamma_{m+1} Q_m e_{m+1})\,,$$

where γ_{m+1} is the last component of the vector $\beta Q_m^T e_1$. Therefore,

$$\begin{aligned} \|\mathsf{R}_m\|_F &= \|\mathrm{V}_{m+1} ② (\gamma_{m+1} Q_m e_{m+1})\|_F\,, \\ &= \|\gamma_{m+1} Q_m e_{m+1}\|_2\,, \\ &= |\gamma_{m+1}|\,, \end{aligned}$$

which shows the results. □

7.4 t-Bartels-Stewart method for Sylvester tensor equations of small size

In this section, we introduce the *t-Bartels-Stewart* method for Sylvester tensor equations of small size based on T-product formalism, as a generalisation of

Algorithm 28 tFOM and tGMRES Algorithms

1. **Input.** $\mathcal{A} \in \mathbb{R}^{n\times n\times n_3}$, $\mathcal{B} \in \mathbb{R}^{q\times q\times n_3}$, $\mathcal{X}_0, \mathcal{C} \in \mathbb{R}^{n\times q\times n_3}$, the maximum number of iteration $\text{Iter}_{\max}$ an integer m and a tolerance tol.
2. Compute $\mathcal{R}_0 = \mathcal{C} - \mathcal{M}(\mathcal{X}_0)$.
3. For $k = 1, \ldots, \text{Iter}_{\max}$
   (a) Apply Algorithm 23 to compute $\mathcal{V}_m$ and $\tilde{H}_m$.
   (b) Solve the problem :
   $$\begin{cases} H_m y_m = \|\mathcal{R}_0\|_F e_1^{(m)} & \textbf{(tFOM method)} \\ y_m = \arg\min_{y\in\mathbb{R}^m} \| \|\mathcal{R}_0\|_F e_1 - \tilde{H}_m y_m \|_2. & \textbf{(tGMRES method)} \end{cases}$$
   (c) Compute $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}_m \circledast y_m$
4. If $\|\mathcal{R}_m\|_F < tol$ **then**
   return $\mathcal{X}_m$;
5. else $\mathcal{X}_0 = \mathcal{X}_m$ and go to Step 2.
6. **Output.** $\mathcal{X}_m \in \mathbb{R}^{n\times q\times n_3}$ the approximate solution of (7.3).

the well known Bartels and Stewart algorithm proposed in [8]. Motivated by the matrix case, the *t-Bartels-Stewart* method is based on transforming the tensors $\mathcal{A}$ and $\mathcal{B}$ into *t-real schur* form -that will be defined later-. This gives a new triangular tensor equation that will be solved by a *t-back-substitution* method. First, we introduce the *t-real Schur* decomposition.

Theorem 7.4. *Let* $\mathcal{A} \in \mathbb{R}^{n\times n\times n_3}$. *Then* $\mathcal{A}$ *can be factored as*

$$\mathcal{A} = \mathcal{U} * \mathcal{R} * \mathcal{U}^T \tag{7.16}$$

where $\mathcal{R} \in \mathbb{R}^{n\times n\times n_3}$ *quasi upper triangular tensor (each frontal slice of* $\mathcal{R}$ *is quasi upper triangular) and* $\mathcal{U} \in \mathbb{R}^{n\times n\times n_3}$ *orthogonal tensor.*

Proof. We have

$$(F_{n_3} \otimes I_{n_1})\, \text{bcirc}(\mathcal{A})\, (F_{n_3}^* \otimes I_{n_1}) = \mathbf{A} = \begin{bmatrix} \tilde{\mathcal{A}}^{(1)} & & & \\ & \tilde{\mathcal{A}}^{(2)} & & \\ & & \ddots & \\ & & & \tilde{\mathcal{A}}^{(n_3)} \end{bmatrix},$$

Next, we compute the schur matrix decomposition of each frontal slice $\tilde{ffi}^{(i)}, i = 1, \ldots, n_3$. Then

$$\begin{bmatrix} \tilde{f}^{(1)} & & \\ & \ddots & \\ & & \tilde{f}^{(n_3)} \end{bmatrix} = \begin{bmatrix} \tilde{U}^{(1)} & & \\ & \ddots & \\ & & \tilde{U}^{(n_3)} \end{bmatrix} \begin{bmatrix} \tilde{R}^{(1)} & & \\ & \ddots & \\ & & \tilde{R}^{(n_3)} \end{bmatrix} \begin{bmatrix} \tilde{U}^{(1)\,T} & & \\ & \ddots & \\ & & \tilde{U}^{(n_3)T} \end{bmatrix} \tag{7.17}$$

Since $(F_{n_3}^* \otimes I_{n_1}), (F_{n_3} \otimes I_{n_1}), (F_{n_3}^* \otimes I_{n_1}), (F_{n_3} \otimes I_{n_1})$ are circulant matrices, by applying the appropriate matrices $(F_{n_3}^* \otimes I_{n_1}), (F_{n_3} \otimes I_{n_1})$ to the left and right of each matrix in (7.17) respectively, and folding up the result. This gives the result. □

The *t-real Schur* decomposition can be computed by the Algorithm 29.

Next, we transform the tensors ffi, B into *t-real Schur* forms $ffi = U_{ffi} * R_{ffi} * U_{ffi}^T$ and $B = U_B * R_B * U_B^T$. Then the Sylvester tensor equation (7.3) becomes

$$(U_{ffi} * R_{ffi} * U_{ffi}^T) * X + X * (U_B * R_B * U_B^T) = C$$

Algorithm 29 t-real Schur decomposition

1. **Input:** $ffi \in R^{n \times n \times n_3}$.
2. **Output:** $R \in R^{n \times n \times n_3}$ upper triangular tensor. $U \in R^{n \times n \times n_3}$ orthogonal tensor.
3. Set $\tilde{ffi} = \texttt{fft}(ffi, [], 3)$
   (a) for $i = 1, \ldots, n_3$
      i. $[\tilde{Q}^{(i)} \tilde{R}^{(i)}] = \text{Schur}(\tilde{ffi}^{(i)})$ (real Schur matrix decomposition)
   (b) End
4. $Q = \texttt{ifft}(\tilde{Q}, [], 3), R = \texttt{ifft}(\tilde{R}, [], 3)$
5. End

which equivalent to

$$U_{ffi}^T * \Big[(U_{ffi} * R_{ffi} * U_{ffi}^T) * X + X * (U_B * R_B * U_B^T) \Big] * U_B = U_{ffi}^T * C * U_B$$

Since U_{ffi} and U_B are orthogonal tensors, we get

$$R_{ffi} * Y + Y * R_B = C_1 \tag{7.18}$$

where $Y = U_{ffi}^T * X * U_B$ and $C_1 = U_{ffi}^T * C * U_B$.

Remark 7.5. The equation (7.18) can be solved by the *t-back-substitution* method. This method follows the same steps of matrix back substitution , where the tube fibers (3-mode fibers), lateral slices and T-product play the role of scalars, vectors and matrix product respectively.

Finally, the *t-Bartels-Stewart* method is implemented by Algorithm 30.

Algorithm 30 t-Bartels-Stewart method

1. **Input:** $ffi \in R^{n \times n \times n3}$, $B \in R^{q \times q \times n3}$ and $C \in R^{n \times q \times n3}$.
2. **Output:** $X_S \in R^{n \times q \times n3}$ solution of Sylvester tensor equation (7.3) of small size.
3. Compute *t-real Schur* forms

$$ffi = U_{ffi} * R_{ffi} * U_{ffi}^T, \quad B = U_B * R_B * U_B^T$$

   using Algorithm 29.
4. Compute Y_S solution of tensor equation (7.18) using *t-back-substitution* method.
5. Compute $X_S = U_{ffi} * Y_S * U_B^T$.

7.5 Tensor Krylov methods via T-product for solving large Sylvester tensor equations

In this section, we consider the case when the Sylvester tensor equation (7.1) and (8.2) is of large size. As in the matrix case, iterative projection methods have been developed; see [47, 75, 120]. These methods use Galerkin projection methods, such as the classical and the block Arnoldi techniques,

to produce low-dimensional Sylvester matrix equations that are solved by using direct methods.

In tensor case, the main idea is to transform the large Sylvester tensor equations using the well know the Tubal block Arnoldi -that will be introduced later- into law dimensional equations.

7.5.1 Tubal block Arnoldi method

In this subsection, we introduce the Tubal-Block-Arnoldi method via T-product as a generalization of the well know block Arnoldi method, see [47] for more details.

Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ be a square tensor and $\mathcal{V} \in \mathbb{R}^{n_1 \times s \times l_3}$, $s << n_1$. The tensor block Krylov subspace is defined by

$$\mathbb{K}_m^{Block}(\mathcal{A}, \mathcal{V}) = Range\left[\mathcal{V}, \mathcal{A} * \mathcal{V}, \ldots, \mathcal{A}^{m-1} * \mathcal{V}\right] \subset \mathbb{R}^{n_1 \times 1 \times n_3} \tag{7.19}$$

where $\mathcal{A}^i = \mathcal{A} * \mathcal{A}^{i-1}, i = 1, \ldots, m-1, \mathcal{A}^0 = \mathcal{I}_{n_1 n_1 n_3}$ and

$$Range(\mathcal{Z}) = \left\{ \vec{\mathcal{Y}} \in \mathbb{R}^{n_1 \times 1 \times n_3} / \vec{\mathcal{Y}} = \mathcal{Z} * \vec{\mathcal{X}}, \ \vec{\mathcal{X}} \in \mathbb{R}^{s \times 1 \times n_3} \right\}$$

for $\mathcal{Z} \in \mathbb{R}^{n_1 \times s \times n_3}$.

Before describing the tubal Block Arnoldi process, let us first introduce the Tubal-QR Factorization.

Theorem 7.6. *(Tubal-QR Factorization) Let* $\mathcal{A} = \left[\vec{\mathcal{A}}_1, \ldots, \vec{\mathcal{A}}_m\right] \in \mathbb{R}^{n_1 \times m \times n_3}$, $\vec{\mathcal{A}}_i \in \mathbb{R}^{n_1 \times 1 \times n_3}$ *for* $i = 1, \cdots, m$, *there exist orthogonal tensors* $\mathcal{U} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ *and* $\mathcal{R} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ *triangular tensor such that*

$$\mathcal{A} = \mathcal{Q} * \mathcal{R} \tag{7.20}$$

We call (7.20) the tubal QR factorization of the tensor $\mathcal{A}$.

The Tubal QR factorization is summarized in Algorithm (31).

The function $Normalization1(\vec{\mathcal{ffi}})$ described in [89] allows us to write a

Algorithm 31 Tubal-QR Factorization (Tubal-QR)

1. **Input.** $\mathcal{ffi} \in \mathbb{R}^{n \times m \times n_3}$
2. Set $[\mathcal{V}_1, R_{1,1,:}] = Normalization1(\vec{\mathcal{ffi}}_1)$
3. For $j = 1, \ldots, m$
   (a) $\mathcal{W} = \vec{\mathcal{ffi}}_j$,
   (b) for $i = 2, \ldots, j-1$
      i. $\mathcal{R}_{i,j,:} = \mathcal{V}_i^T * \mathcal{W}$
      ii. $\mathcal{W} = \mathcal{W} - \mathcal{V}_i * \mathcal{R}_{i,j,:}$
   (c) End for
   (d) $[\mathcal{Q}_j, \mathcal{R}_{j,j,:}] = Normalization1(\vec{\mathcal{W}})$
4. End
5. **Output.** $\mathcal{Q} \in \mathbb{R}^{n \times m \times n_3}$ orthogonal and $\mathcal{R} \in \mathbb{R}^{m \times m \times n_3}$ such that : $\mathcal{A} = \mathcal{Q} * \mathcal{R}$.

given tensor $\vec{\mathcal{ffi}} \in \mathbb{R}^{n \times 1 \times n_3}$ as follow : $\vec{\mathcal{ffi}} = \vec{\mathcal{U}} * \mathbf{a}$ where $\mathbf{a} \in \mathbb{R}^{1 \times 1 \times n_3}$ is invertible and $\vec{\mathcal{U}}^T * \vec{\mathcal{U}} = \mathbf{e}$.

Tubal Block Arnoldi method have to build an orthonormal basis of $\mathcal{V}_m^b$ of the Krylov subspace $\mathcal{K}_m^{Block}(\mathcal{ffi}, \mathcal{V})$.

Algorithm 32 The Tubal Block Arnoldi Algorithm (TBA)

1. **Input.** $\mathcal{A} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ be a square tensor, $\mathcal{V} \in \mathbb{R}^{n_1 \times s \times I_3}$, $s << n_1$ and an integer m.
2. Set $[\mathcal{V}_1^b, \mathcal{H}_0] =$ Tubal-QR($\mathcal{V}$).
3. For j = 1,...,m
   (a) $\mathcal{W} = \mathcal{A} * \mathcal{V}_j$
   (b) For i = 1,...,j
      i. $\mathcal{H}_{i,j} = \mathcal{V}_i^{bT} * \mathcal{W}$
      ii. $\mathcal{W} = \mathcal{W} - \mathcal{V}_i^b * \mathcal{H}_{i,j}$,
   (c) End.
   (d) $\mathcal{V}_{j+1}, \mathcal{H}_{j+1,j} ==$ Tubal-QR($\mathbb{W}$).
4. End

Using Definition 3.6, we can put away the tensors $\mathcal{H}_{i,j} \in \mathbb{R}^{s \times s \times n_3}$ into block tensors $\mathcal{H}_m$ and $\mathcal{H}_{m+1}$ defined as follow

$$\mathcal{H}_{m+1} = \begin{bmatrix} \mathcal{H}_{1,1} & \mathcal{H}_{1,2} & \cdots & \mathcal{H}_{1,m} \\ \mathcal{H}_{2,1} & \mathcal{H}_{2,2} & \cdots & \mathcal{H}_{2,m} \\ \ddots & \ddots & \ddots & \vdots \\ & & \mathcal{H}_{m,m-1} & \mathcal{H}_{m,m} \\ & & & \mathcal{H}_{m+1,m} \end{bmatrix} \in \mathbb{R}^{(m+1)s \times ms \times n_3}$$

$$\mathcal{H}_m = \begin{bmatrix} \mathcal{H}_{1,1} & \mathcal{H}_{1,2} & \cdots & \mathcal{H}_{1,m} \\ \mathcal{H}_{2,1} & \mathcal{H}_{2,2} & \cdots & \mathcal{H}_{2,m} \\ \ddots & \ddots & \ddots & \vdots \\ & & \mathcal{H}_{m,m-1} & \mathcal{H}_{m,m} \end{bmatrix} \in \mathbb{R}^{ms \times ms \times n_3}$$

It is not difficult to show that after m steps of Algorithm 32, the tensor $\mathcal{V}_m^b := [\mathcal{V}_1^b, \ldots, \mathcal{V}_m^b] \in \mathbb{R}^{n_1 \times ms \times n_3}$, where $\mathcal{V}_l^b \in \mathbb{R}^{n_1 \times s \times n_3}$ forms an orthonormal basis of the tensor block Krylov subspace $\mathbb{K}_m^{Block}(\mathcal{F}, \mathcal{V})$.

It is easy to see that $\mathcal{H}_m$ can be obtained from $\mathcal{H}_{m+1}$ by deleting the last block row

$$[\mathcal{O}_{ssn_3}, \ldots, \mathcal{O}_{ssn_3}, \mathcal{H}_{m+1,m}] = \mathcal{H}_{m+1,m} * \mathcal{E}_m \in \mathbb{R}^{s \times ms \times n_3}$$

where $\mathcal{O}_{ssn_3}$ denotes de zeros tensors of size $(s \times s \times n_3)$ which all his entriesare equal to zeros, and $\mathcal{E}_m = [\mathcal{O}_{ssn_3}, \ldots, \mathcal{O}_{ssn_3}, \mathcal{I}_{ssn_3}] \in \mathbb{R}^{s \times ms \times n_3}$

Let

$$\mathcal{F} * \mathcal{V}_m^b := [\mathcal{F} * \mathcal{V}_1, \ldots, \mathcal{F} * \mathcal{V}_m]. \in \mathbb{R}^{n_1 \times ms \times n_3}$$

$$\mathcal{V}_{m+1}^b := \left[\mathcal{V}_m^b, \mathcal{V}_{m+1}^b\right] \in \mathbb{R}^{n_1 \times (m+1)s \times n_3}$$

$$\mathcal{H}_m = (\mathcal{H}_{i,j})_{1 \le i,j \le m} \in \mathbb{R}^{ms \times ms \times n_3}$$

$$\mathcal{H}_{m+1} = \begin{bmatrix} \mathcal{H}_m \\ \mathcal{H}_{m+1,m} * \mathcal{E}_m \end{bmatrix} \in \mathbb{R}^{(m+1)s \times ms \times n_3}$$

With the above notations, we can easily prove the results of the following proposition

Proposition 7.7. *From Algorithm 32, we have*

$$\mathcal{A} * \mathcal{V}_m^b = \mathcal{V}_m^b * \mathcal{H}_m + \mathcal{V}_{m+1}^b * (\mathcal{H}_{m+1,m} * \mathcal{E}_m) \tag{7.21}$$

$$\mathcal{V}_m^{bT} * \mathcal{A} * \mathcal{V}_m^b = \mathcal{H}_m \tag{7.22}$$

$$\mathcal{V}_{m+1}^{bT} * \mathcal{A} * \mathcal{V}_m^b = \mathcal{H}_{m+1} \tag{7.23}$$

$$\mathcal{V}_m^{bT} * \mathcal{V}_m^b = \mathcal{I}_{ms} \tag{7.24}$$

where $\mathcal{I}_{ms} \in \mathbb{R}^{ms \times ms \times n_3}$ *denotes the identity tensor.*

Notice that in the case when $n_3 = 1$, Algorithm 32 reduces to the well known block Arnoldi process.

Proof. The proof comes directly from steps of Algorithm 32 and Proposition 3.7. □

7.5.2 Tubal block Arnoldi method for solving large Sylvester tensor equations (TBAS)

This subsection discusses the computation of an approximate solution of the tensor equations :

$$\mathcal{M}(\mathcal{X}) = \mathcal{C}, \tag{7.25}$$

where $\mathbb{M}$ is a linear operator that could be described as

$$\mathcal{M}(\mathcal{X}) = \mathcal{A} * \mathcal{X} - \mathcal{X} * \mathcal{B} \tag{7.26}$$

Where $\mathcal{A} \in \mathbb{R}^{n \times n \times n_3}$, $\mathcal{B} \in \mathbb{R}^{q \times q \times n_3}$ and $\mathcal{X}, \mathcal{C} \in \mathbb{R}^{n \times q \times n_3}$ respectively. $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ are given, $\mathcal{X}$ the unknown tensor to be determined.

Let $\mathcal{X}_0 \in \mathbb{R}^{n \times s \times p}$ be an arbitrary initial guess with the corresponding resid- ual $\mathcal{R}_0 = \mathcal{C} - \mathbf{M}(\mathcal{X}_0)$. The purpose of tensor tubal Block Arnoldi method for solving large Sylvester tensor equations (TBAS) method is to find an approximate solution $\mathcal{X}_m$ of the exact solution $\mathcal{X}^*$ of (7.25) such that

$$\mathcal{X}_m - \mathcal{X}_0 \in \mathbb{K}_m^{Block}(\mathbf{M}, \mathcal{R}_0), \tag{7.27}$$

with the classical orthogonality property

$$\mathcal{R}_m = \mathcal{C} - \mathbf{M}(\mathcal{X}_m) \perp \mathbb{K}_m^{Block}(\mathbf{M}, \mathcal{R}_0). \tag{7.28}$$

Using the fact that $\mathbb{K}_m^{Block}(\mathbf{M}, \mathcal{R}_0) = \mathbb{K}_m^{Block}(\mathcal{ffi}, \mathcal{R}_0)$ (which is not difficult to prove it), the relations (7.27) and (7.28) can be expressed as

$$\mathcal{X}_m - \mathcal{X}_0 \in \mathbb{K}_m^{Block}(\mathcal{ffi}, \mathcal{R}_0), \tag{7.29}$$

with the classical orthogonality property

$$\mathcal{R}_m = \mathcal{C} - \mathbf{M}(\mathcal{X}_m) \perp \mathbb{K}_m^{Block}(\mathcal{ffi}, \mathcal{R}_0). \tag{7.30}$$

Since $\mathbb{V}_m^b$ (defined earlier) forms an orthonormal basis of of the Krylov sub-space $\mathbb{K}_m^{Block}(\mathcal{ffi}, \mathcal{R}_0)$. The relations (7.29) and (7.30) can be written as

$$\mathcal{X}_m = \mathcal{X}_0 + \mathbb{V}_m^b * \mathcal{Y}_m, \quad \mathcal{Y}_m \in \mathbb{R}^{ms \times s \times n3}, \tag{7.31}$$

and

$$\mathbb{V}_m^{bT} * \mathcal{R}_m = 0 \tag{7.32}$$

Using the fact that $\mathcal{R}_m = \mathcal{C} - \mathbf{M}(\mathcal{X}_m) = \mathcal{C} - \mathcal{ffi} * \mathcal{X}_m + \mathcal{X}_m * \mathcal{B}$, the relations (7.31) and (7.32), we get the law dimensional equation

$$\mathbb{H}_m * \mathcal{Y}_m - \mathcal{Y}_m * \mathcal{B} = \mathcal{C}_1 \tag{7.33}$$

where $C_1 = V_{m}^{bT} * R_0$.

The tensor equation (7.33) will be solved by using the *t-Bartels-Stewart* method.

Proposition 7.8. *At step m, the residual norm of* $R_m = C - \mathcal{M}(X_m) = C - ffi * X_m + X_m * B$ *with* $X_m = X_0 + V_m^b * Y_m$ *can be expressed as follows :*

$$||R_m||_F = ||H_{m+1,m} * E_m * Y_m||_F \tag{7.34}$$

where $E_m = [O_{ssn_3}, \ldots, O_{ssn_3}, I_{ssn_3}] \in R^{s^x ms^x n3}$ *and* Y_m *solution of equation (7.33).*

Proof. At step m, the residual $R_m = C - \mathcal{M}(X_m) = C - ffi * X_m + X_m * B$ with $X_m = X_0 + V_m^b * Y_m$ can be written as :

$$R_m = R_0 - ffi * V_m * Y_m + V_m * Y_m * B$$

using the fact that $A * V_m^b = V_m^b * H_m + V_{m+1}^b * (H_{m+1,m} * E_m)$, the relation (7.33) and the fact that the tensor V_{m+1}^b is orthogonal, we get

$$||R_m||_F = ||H_{m+1,m} * E_m * Y_m||_F$$

□

To save memory and CPU-time requirements, the tubal Block Arnoldi method for solving large Sylvester tensor equation (TBAS) will be used in a restarted mode. This means that we have to restart the algorithm every m inner iterations, where m is a fixed integer. The restarted Tubal Block Arnoldi algorithm for solving (7.25), denoted by TBAS(m), is summarized as follows:

During the computation of the approximate solution X_m of (7.25), we assume that $\Gamma(\mathbf{H})_m \cap \Gamma(-\mathbf{B}) = \emptyset$, where the block diagonal matrix $\mathbf{H_m}$, $(-\mathbf{B})$ are defined by (3.2), and $\Gamma(\mathbf{H}_m)$, $\Gamma(-\mathbf{B})$ denotes the set of eigenvalues of the

Algorithm 33 The Tubal Block Arnoldi for solving large Sylvester tensor equation TBAS(m)

1. **Input.** ffi$\in \mathbb{R}^{n\times n\times n3}$, V, B $\in \mathbb{R}^{q\times q\times n3}$, $\mathcal{X}_0, \mathcal{C} \in \mathbb{R}^{n\times q\times n3}$, the maximum number of iteration $\text{Iter}_{\max}$ an integer m and a tolerance *tol*.
2. Compute $\mathcal{R}_0 = \mathcal{C} - \mathcal{M}(\mathcal{X}_0)$.
3. For $k = 1, \ldots, \text{Iter}_{\max}$
   (a) Apply Algorithm 32 to compute $\mathcal{V}_m^b$ and $\mathcal{H}_m$.
   (b) Apply Algorithm 30 to solve the problem : $\mathcal{H}_m * \mathcal{Y}_m - \mathcal{Y}_m * \mathcal{B} = \mathcal{V}^{bT}_m * \mathcal{R}_0$
   (c) Compute $\mathcal{X}_m = \mathcal{X}_0 + \mathcal{V}^b_m * \mathcal{Y}_m$
4. If $||\mathcal{R}_m||_F < tol$, stop
5. else $\mathcal{X}_0 = \mathcal{X}_m$ and go to Step 2.
6. **Output.** $\mathcal{X}_m \in \mathbb{R}^{n\times q\times n3}$ the approximate solution of (7.25).

matrix $\mathbf{H}_m$ and (**- B**) respectively.

7.6 Numerical experiments

This section performs some numerical tests for the Tensor Tubal-Global GM-RES and Tensor Tubal-Global Golub Kahan methods when solving solve the linear tensor problem (7.1). All computations were carried out using the MATLAB R2018b environment with an Intel(R) Core i7-8550U CPU @1.80 GHz and processor 8 GB. The stopping criterion was

$$||\mathcal{R}_k||_F < e,$$

where $e = 10^{-6}$ is a chosen tolerance and $\mathcal{R}_k$ is the k-th residual associated to the approximate solution $\mathcal{X}_k$. In all the presented tables, we reported the obtained residual norms to achieve the desired convergence, the iteration number and the corresponding CPU-time.

We will compare the results in our method to the results obtained by solving the equivalent problem $\mathbf{C} = \mathbf{AX} - \mathbf{XB}$ where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are the matrices

$$\mathbf{A} = \text{Diag}(\tilde{ffi}) = \begin{bmatrix} \tilde{ffi}^{(1)} & & & \\ & \tilde{ffi}^{(2)} & & \\ & & \ddots & \\ & & & \tilde{f}^{(n_3)} \end{bmatrix} = \text{BlockDiag}(\tilde{ffi}^{(1)}, \ldots, \tilde{ffi}^{(n_3)}),$$

$$\mathbf{B} = \text{Diag}(\tilde{B}) = \text{BlockDiag}(\tilde{B}^{(1)}, \ldots, \tilde{B}^{(n_3)})$$

$$\mathbf{C} = \text{Diag}(\tilde{C}) = \text{BlockDiag}(\tilde{C}^{(1)}, \ldots, \tilde{C}^{(n_3)})$$

and the matrices $\tilde{ffi}^{(i)}, \tilde{B}^{(i)}$ and $\tilde{C}^{(i)}$'s are the frontal slices of the tensor $\tilde{f}$, $\tilde{B}$ and $\tilde{C}$ respectively.

Notice that $\mathbf{A}$ is a matrix of size $(n_0 \times n_0)$, $n_0 = n \times n_3$, $\mathbf{B}$ is a matrix of size $(s_0 \times s_0)$, $s_0 = s \times n_3$ and $\mathbf{C}$ is a matrix of size $(n_0 \times s_0)$.

We compared the required CPU-times (in seconds) to achieve the convergence for the two methods:

1. **TBAS**: The tubal block Arnoldi–Sylvester method.
2. **BAS** : Resolution of the Sylvester matrix equation : $\mathbf{C} = \mathbf{AX} - \mathbf{XB}$ by using the block Arnoldi-Sylvester method introduced by El Guennouni and al; see [47].
3. **tFOM** : The tensor full orthogonalization method.
4. **tGMRES** : The tensor generalized minimal residual method.

In Table 7.1, we reported the obtained relative residual norms, the total number of required iterations to achieve the convergence and the corresponding CPU-times for restarted tubal Block Arnoldi(k) .

7.6.1 Example 1

Consider the convection-diffusion equation :

$$\begin{aligned} -\mu\,\Delta u + c^T\nabla\, u &= f \qquad \text{in } [0,1]^N \\ u &= 0 \qquad \text{in } \partial\Omega \end{aligned} \tag{7.35}$$

The tensor $\mathbb{A}$, $\mathbb{B}$ are obtained using n_3 frontal slices (which are obtained from a standard finite difference discretization of (7.35)). In this example, the frontal slices are of size $n \times n$ and $s \times s$, given as follows

$$\mathbb{A}^{(i)} = \frac{\mu}{h_1^2}\text{tridiag}(-1,2,-1) + \frac{a_i}{4h_1}\begin{bmatrix} 3 & -5 & 1 & & \\ 1 & 3 & -5 & \ddots & \\ & \ddots & \ddots & \ddots & 1 \\ & & 1 & 3 & -5 \\ & & & 1 & 3 \end{bmatrix};$$

and

$$\mathbb{B}^{(i)} = \frac{\mu}{h_2^2}\text{tridiag}(-1,2,-1) + \frac{b_i}{4h_2}\begin{bmatrix} 3 & -5 & 1 & & \\ 1 & 3 & -5 & \ddots & \\ & \ddots & \ddots & \ddots & 1 \\ & & 1 & 3 & -5 \\ & & & 1 & 3 \end{bmatrix};$$

For $i = 1, \ldots, n_3$. We set $a_i = i, b_i = n_3 + i, h_1 = \frac{1}{n+1}$ and $h_2 = \frac{1}{s+1}$.

In this example, the right-hand side tensor $\mathbb{C}$ is constructed using the Matlab command $\mathbb{C} = \text{rand}(n, s, n_3)$.

Table 7.1 reports on the obtained relative residual norms and the corresponding CPU-times to obtain the desired convergence. As can be seen from this table, the restarted tensor tubal block Arnoldi method (TBAS)(m) gives good results within small CPU-times .

Table 7.1: *Results for Example 2.* $e = 10^{-6}$, *and* $n_3 = 2$

$n\backslash s\backslash k$	Method	# its.	$\|R_k\|_F$	cpu-time in seconds
1000\3\10	**TABS**	11	5.55×10^{-7}	**3.03**
	BAS[47]	65	7.18×10^{-7}	7.22
	tFOM	66	7.57×10^{-7}	**6.69**
	tGMRES	39	9.50×10^{-7}	**4.34**
2000\3\6	**TABS**	12	6.86×10^{-8}	**12.02**
	BAS[47]	69	3.25×10^{-7}	17.61
	tFOM	69	6.91×10^{-7}	**15.16**
	tGMRES	41	7.38×10^{-7}	**14.05**

7.6.2 Example 2

In this example, the tensor $\mathcal{A}$ and $\mathcal{B}$ are constructed using $n_3 = 2$ frontal slices. The first frontal slices of $\mathcal{A}$ and $\mathcal{B}$ are obtained from the centred finite difference discretization of the operators

$$L_{\mathcal{A}(1)}(u) = \Delta u - f_{1,1}(x,y)\frac{\partial u}{\partial x} + f_{1,2}(x,y)\frac{\partial u}{\partial y} + g_{1,3}(x,y)u$$
$$L_{\mathcal{B}(1)}(u) = \Delta u - f_{2,1}(x,y)\frac{\partial u}{\partial x} + f_{2,2}(x,y)\frac{\partial u}{\partial y} + g_{2,3}(x,y)u$$

on the unit square $[0,1] \times [0,1]$ with homogeneous Dirichlet boundary conditions. The dimensions of the matrices $\mathcal{A}^{(1)}$ and $\mathcal{B}^{(1)}$ are $n = n_1^2, s = s_1^2$. where n_1 and s_1 are the number of inner grid points in each direction.
The second frontal slices of $\mathcal{A}$ and $\mathcal{B}$ are obtained from finite difference discretization on the unit square $[0,1] \times [0,1]$ of the operators

$$L(u) = -\Delta u + 2\mu\frac{\partial u}{\partial x} + 2\mu\frac{\partial u}{\partial y}$$

with Dirichlet boundary conditions. with Mesh sizes are $h_n = \frac{1}{n+1}$ in the x direction and $h_s = \frac{1}{s+1}$ in the y direction. This yields two tridiagonal frontal slices $\mathcal{A}^{(2)}$, $\mathcal{B}^{(2)}$ given by

$$\mathcal{A}^{(2)} = \text{tridiag}(-1 - \mu h_n, 2, -1 + \mu h_n); \quad \mathcal{B}^{(2)} = \text{tridiag}(-1 - \mu h_s, 2, -1 + \mu h_s)$$

In Table (7.2), the right-hand side tensor $\mathcal{C}$ is constructed randomly, $f_{1,1}(x,y) = \exp(x+y)$, $f_{1,2}(x,y) = 1000y$, $g_{1,3}(x,y) = x$, $f_{2,1}(x,y) = \sin(x+2y)$, $f_{2,2}(x,y) = 20\exp(x+y)$ and $g_{2,3}(x,y) = xy$.

In Table (7.3), the right-hand side tensor $\mathcal{C}$ is constructed randomly, $f_{1,1}(x,y) = \exp(xy)$, $f_{1,2}(x,y) = \sin(xy)$, $g_{1,3}(x,y) = (y^2 - x^2)$, $f_{2,1}(x,y) = 100\exp(x)$, $f_{2,2}(x,y) = 10xy$ and $g_{2,3}(x,y) = \sqrt{y^2 + x^2}$.

Table 7.2: *Results for Example 2.* $e = 10^{-6}$, $\mu = 1$ *and* $n_3 = 2$

$n\backslash s\backslash k$	Method	# its.	$\|\mathcal{R}_k\|_F$	cpu-time in seconds
1600\4\10	**TABS**	32	$\mathbf{9.69 \times 10^{-7}}$	**0.65**
	BAS[47]	31	1.37×10^{-7}	1.77
	tFOM	69	3.05×10^{-7}	**1.09**
	tGMRES	32	3.89×10^{-7}	**0.90**

Table 7.3: *Results for Example 2.* $e = 10^{-6}$, $\mu = 10$ *and* $n_3 = 2$

$n\backslash s\backslash k$	Method	# its.	$\|\mathcal{R}_k\|_F$	cpu-time in seconds
1600\4\10	**TABS**	34	4.17×10^{-8}	**1.03**
	BAS[47]	32	6.11×10^{-8}	2.08
	tFOM	35	1.83×10^{-7}	**1.71**
	tGMRES	29	2.95×10^{-7}	**1.59**

7.7 Conclusion

In this chapter, we presented some new Krylov subspace methods using the T-product and some new other tensor products. We gave new algebraic properties of these products that allowed us to build new tensor Krylov-subspace based algorithms for solving general linear tensor equations. We focussed on the tubal-global GMRES and the tubal-global Golub-Kahan algorithms. Some numerical tests on simple examples are also reported.

Chapter 8

Tensor extrapolation methods using the T-product

In this chapter, we mainly develop the well-known vector and matrix polynomial extrapolation methods in tensor framework. To this end, some new products between tensors are defined and the concept of positive definiteness is extended for tensors corresponding to T-product. Furthermore, we discuss on the solution of least-squares problem associated with a tensor equation using Tensor Singular Value Decomposition (TSVD). Motivated by the effectiveness of some proposed vector extrapolation methods in earlier papers, we describe how an extrapolation technique can be also implemented on the sequence of tensors produced by truncated TSVD (TTSVD) for solving possibly ill-posed tensor equations.

8.1 Introduction

In the last few years, efficient implementations of some of these extrapolation methods have been proposed by Sidi [124] for the RRE and MPE methods using QR decomposition while Jbilou and Sadok [83] gives an efficient implementation of the MMPE based on a LU decomposition with pivoting strategy. It was also shown that when applied to linearly generated vector

sequences, RRE and TEA methods are mathematically equivalent to GMRES and Lanczos methods, respectively. Those results were also extended to the block and global cases dealing with matrix sequences, see [79, 84]. Our aim in thischapter is to define the analogue of these vector and matrix extrapolation methods to the tensor framework.

Basically, in the present paper, we develop some tensor extrapolation methods namely, the Tensor RRE (TRRE), the Tensor MPE (TMPE), the Tensor MMPE (TMMPE) and the Tensor Topological *e*-Algorithm (TTEA). We define new products and establish some of their properties. As an application, it is shown that these new tensor extrapolation methods can be applied to sequences obtained by truncation of the Tensor Singular Value Decomposition (TSVD) to solve tensor discrete ill-posed problems.

The remainder of this chapter is organized as follows. In Section 8.2, we give notations, some basic definitions and properties related to tensors. Moreover, we introduce the concept of positive definiteness for tensors with respect to T-product, some new products are also defined between tensors and their properties are analyzed. In Section 8.3, we introduce the tensor versions of the vector polynomial extrapolation methods namely the Tensor Reduced Rank Extrapolation (TRRE), the Tensor Minimal Polynomial Extrapolation (TMPE), the Tensor Modified Minimal Polynomial Extrapolation (TMMPE), and the Tensor Topological *e*-Algorithm (TTEA). Section 8.4 describes the TSVD and its the truncated or low rank version. In addition, as an application, the TRRE method is applied to the sequence produced by truncated TSVD for solving linear discrete tensor ill-posed problems. Some numerical experiments are reported in Section 8.5 for the mentioned application. Concluding remarks are given in Section 8.6.

8.2 Definitions and new tensor products

All definitions and properties related to the T-product are defined ealier; see chapter 3 section 3.2.

The current section is concerned with two main parts. In the first part, we recall definitions and properties related to T-product. Furthermore, we develop the definition of positive definiteness for tensors and establish some basic results. The second part deals with presenting some new products between tensors which can be used for simplifying the algebraic computations of the main results.

8.2.1 Definitions and properties

Here we define the notion of positive definiteness for tensors in term of T-product which can be seen as a natural extension of the same concept for matrices.

Definition 8.1. The tensor $\mathcal{A} \in \mathbb{R}^{m\times m\times n}$ is said to be positive (semi) definiteif

$$(\mathcal{X}^T * \mathcal{A} * \mathcal{X})_{::1} > (\geq) 0,$$

for all nonzero tensors $\mathcal{X} \in \mathbb{R}^{m\times 1\times n}$.

Remark 8.2. As pointed out earlier the tensor $\mathcal{A}$ of order $m \times m \times n$ is invertible iff bcirc($\mathcal{A}$) is invertible. It is equivalent to say that $\mathcal{A}$ is invertible iff $\mathcal{A} * \mathcal{X} = \mathcal{O}$ implies $\mathcal{X} = \mathcal{O}$ where $\mathcal{X} \in \mathbb{R}^{m\times 1\times n}$ and $\mathcal{O}$ is zero tensor.

Here we comment that it is not difficult to verify that the summation of positive definite tensors is positive definite.

Remark 8.3. In view of Remark 8.2, it is immediate to see that every positive definite tensor is invertible. It can be also verified that the inverse of a positive definite tensor is also positive definite. For any tensor $B \in \mathbb{R}^{n_1\times n_2\times n_3}$

and arbitrary nonzero tensor $\mathcal{X} \in \mathbb{R}^{n_2 \times 1 \times n_3}$, it can be seen that

$$(\mathcal{X}^T * \mathcal{B}^T * \mathcal{B} * \mathcal{X})_{::1} = ((\mathcal{B} * \mathcal{X})^T * \mathcal{B} * \mathcal{X})_{::1} = \|\mathcal{B} * \mathcal{X}\|^2 \geq 0,$$

in which the last equality follows from [89]. As a result, the tensor $\mathcal{B}^T * \mathcal{B}$ is positive semi definite. Evidently, for any scalar $e > 0$,

$$(\mathcal{X}^T * (\mathcal{B}^T * \mathcal{B} + e\, \mathcal{I}_{n_2 n_2 n_3}) * \mathcal{X})_{::1} = \|\mathcal{B} * \mathcal{X}\|^2 + e\|\mathcal{X}\|^2 > 0,$$

This shows that the tensor $\mathcal{B}^T * \mathcal{B} + e\, \mathcal{I}_{n_2 n_2 n_3}$ is positive definite.

Definition 8.4. Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. Then the tensor $\mathcal{X} \in \mathbb{R}^{n_2 \times n_1 \times n_3}$ satisfyingthe following four conditions

(a) $\mathcal{A} * \mathcal{X} * \mathcal{A} = \mathcal{A}$ (b) $\mathcal{X} * \mathcal{A} * \mathcal{X} = \mathcal{X}$,

(c) $(\mathcal{A} * \mathcal{X})^T = \mathcal{A} * \mathcal{X}$, (d) $(\mathcal{X} * \mathcal{A})^T = \mathcal{X} * \mathcal{A}$.

is called the Moore-Penrose inverse of $\mathcal{A}$ and denoted by $\mathcal{A}^\dagger$.

In view of [90, Lemmas 3.3 and 3.16] and by using straightforward computations, one can easily observe that conditions (a)-(d) determine $\mathcal{A}^\dagger$ uniquely. Here we further comment that if $\mathcal{A}$ is invertible, then $\mathcal{A}^\dagger = \mathcal{A}^{-1}$.

For $\mathcal{X}, \mathcal{Y}$ two tensors in $\mathbb{R}^{n_1 \times 1 \times n_3}$, theT-scalar product $\langle ., . \rangle$ is a bilinear form defined by:

$$\begin{array}{ccc} \mathbb{R}^{n_1 \times 1 \times n_3} \times \mathbb{R}^{n_1 \times 1 \times n_3} & \longrightarrow & \mathbb{R}^{1 \times 1 \times n_3} \\ (\mathcal{X}, \mathcal{Y}) & \longrightarrow & \langle \mathcal{X}, \mathcal{Y} \rangle = \mathcal{X}^T * \mathcal{Y} \end{array} . \tag{8.1}$$

Let $\mathcal{X}_1, \ldots, \mathcal{X}_l$ a collection of l third tensors in $\mathbb{R}^{n_1 \times 1 \times n_3}$, if

$$\langle \mathcal{X}_i, \mathcal{X}_j \rangle = \begin{cases} \alpha_i \mathbf{e} & i = j \\ 0 & i \neq j \end{cases}.$$

where α_i is a non-zero scalar, then the set $\mathcal{X}_1, \ldots, \mathcal{X}_l$ is said to be an orthogonal collection of tensors. The collection is called orthonormal if $\alpha_i = 1$,

$i = 1, \ldots, l$.

Definition 8.5. An $n \times n \times l$ real-values tensor $\mathcal{Q}$ is said to be orthogonal if $\mathcal{Q}^T * \mathcal{Q} = \mathcal{Q} * \mathcal{Q}^T = \mathcal{I}_{nnl}$.

We end this part with the following proposition.

Proposition 8.6. *Suppose that* $\mathcal{F}$ *and* $\mathcal{B}$ *are two tensors of order* $n_1 \times n_2 \times n_3$. *Then*

$$(\mathcal{F}^T * \mathcal{B})_{ij:} = (\mathcal{F}_{:i:})^T * \mathcal{B}_{:j:}.$$

Proof. Let $\mathcal{I}_\tau$ be an $n_2 \times 1 \times n_3$ tensor whose all frontal slices are zero except its first frontal being the τ-th column of the $n_2 \times n_2$ identity matrix. Then we have

$$\mathcal{B} * \tilde{\mathcal{I}}_j = \mathcal{B}_{:j:} \qquad \text{and} \qquad \tilde{\mathcal{I}}_i^T * \mathcal{F}^T = \mathcal{F}_k^T = (\mathcal{F}_{:i:})^T.$$

Notice also that

$$(\mathcal{F}^T * \mathcal{B})_{ij:} = \tilde{\mathcal{I}}_i^T * (\mathcal{F}^T * \mathcal{B}) * \tilde{\mathcal{I}}_j.$$

From [90, Lemma 3.3], it is known that

$$\tilde{\mathcal{I}}_i^T * (\mathcal{F}^T * \mathcal{B}) * \tilde{\mathcal{I}}_j = (\tilde{\mathcal{I}}_i^T * \mathcal{F}^T) * (\mathcal{B} * \tilde{\mathcal{I}}_j),$$

which completes the proof. □

We end this part by the following simple remark which is helpful in the sequel for more clarification.

Remark 8.7. In what follows, we may need to solve a system of tensor equations in the following form:

$$\mathcal{F}_{i1} * \mathcal{X}_1 + \cdots + \mathcal{F}_{ik} * \mathcal{X}_k = \mathcal{B}_i, \qquad i = 1, 2, \ldots, k, \tag{8.2}$$

where $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ are the unknown tensors to be determined. The above system is uniquely solvable iff the following system of equations has only

the trivial solution,

$$\mathcal{A}_{i1} * \mathcal{X}_1 + \cdots + \mathcal{A}_{ik} * \mathcal{X}_k = \mathcal{O}, \qquad i = 1, 2, \ldots, k,$$

where $\mathcal{O}$ stands for the zero tensor. When the diagonal tensors $\mathcal{A}_{11}, \mathcal{A}_{22}, \ldots, \mathcal{A}_{kk}$ are invertible, we can easily conclude that (8.2) has a unique solution in the following two special cases:

* $\mathcal{A}_{ij} = 0$ for $j > i$,
* $\mathcal{A}_{ij} = 0$ for $j < i$,

for $i, j = 1, 2, \ldots, k$.

8.2.2 New tensor products

In order to simplify derivation of generalized extrapolation methods in tensor format, we need to define new tensor products.

Definition 8.8. Let $\mathcal{A}$ and $\mathcal{B}$ be 4-mode tensors with frontal slices $\mathcal{A}_i \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B}_j \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ for $i = 1, 2, \ldots, l$ and $j = 1, 2, \ldots, k$, respectively. The product $\mathcal{A} \blacklozenge \mathcal{B}$ is defined as a 5-mode tensor of order $n_2 \times n_2 \times n_3 \times k \times l$ defined as follows:

$$(\mathcal{A} \blacklozenge \mathcal{B})_{:::ji} = \mathcal{A}_i^T * \mathcal{B}_j,$$

for $i = 1, 2, \ldots, l$ and $j = 1, 2, \ldots, k$. In the case where $k = 1$, i.e. $\mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{A} \blacklozenge \mathcal{B}$ is a 4-mode tenor whose i-th frontal slice is given by $\mathcal{A}_i^T * \mathcal{B}$ for $i = 1, 2, \ldots, l$.

Here we comment that the $\blacklozenge$ product can be seen as a generalization of $\lozenge$-product between two matrices given in [24]. In the sequel, we further present an alternative product called $\circledast$-product which can be seen as an extension of the $*$-product between a set of matrices and vectors; see [81] for more details.

Definition 8.9. Let $\mathcal{A}$ be a 5-mode tensor of order $n_2 \times n_1 \times n_3 \times k \times l$ and $\mathcal{B}$ be a 4-mode tensor with frontal slices $B_1, \ldots, B_k \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. The product $\mathcal{A} \natural \mathcal{B}$ is defined as a 4-mode tensor of order $n_2 \times n_2 \times n_3 \times l$ such that

$$(\mathcal{A} \natural \mathcal{B})_{:::i} = \sum_{j=1}^{k} \mathcal{A}_{:::ji} * B_j, \quad i = 1, 2, \ldots, l.$$

Evidently, a 4-mode tensor of order $n_2 \times n_1 \times n_3 \times k$ can be considered as an 5-mode tensor of order $n_2 \times n_1 \times n_3 \times k \times 1$. As a result, if $\mathcal{A}$ is a 4-mode tensor of order $n_2 \times n_1 \times n_3 \times k$, i.e., $l = 1$, then $\mathcal{A} \natural \mathcal{B} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ defined by $\mathcal{A} \natural \mathcal{B} = \sum_{\eta=1}^{k} \mathcal{A}_\eta * B_\eta$ where $\mathcal{A}_i$ stands for the i-th frontal slice of $\mathcal{A}$ for $i = 1, 2, \ldots, k$.

In the main results, it is worth to define a notion of the left inverse of a 5-mode tensors dealing with $\natural$, $*$ and $\blacklozenge$ products. To this end, we define the $*$-product between two 5-mode tensors which can be seen as an extension of $*$ product.

Definition 8.10. Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times k \times l}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times n_1 \times n_3 \times k \times l}$, the product $\mathcal{A} * \mathcal{B}$ is a 5-mode tensor of order $n_1 \times n_1 \times n_3 \times k \times k$ such that for $\tau, \eta = 1, 2, \ldots, k$,

$$(\mathcal{A} * \mathcal{B})_{:::\tau\eta} = \sum_{j=1}^{l} \mathcal{A}_{:::\eta j} * \mathcal{B}_{:::\tau j}.$$

Definition 8.11. The tensor $\mathcal{B}^+ \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times k \times l}$ is called a left inverse of $\mathcal{B} \in \mathbb{R}^{n_2 \times n_1 \times n_3 \times k \times l}$, if

$$(\mathcal{B}^+ * \mathcal{B})_{:::\tau\eta} = \begin{cases} \mathcal{I}_{n_1 n_1 n_3} & \tau = \eta \\ \mathcal{O} & \tau \neq \eta \end{cases}$$

for $\tau, \eta = 1, 2, \ldots, k$. Here $\mathcal{O}$ is the 3-mode zero tensor of order $n_1 \times n_1 \times n_3$.

Now we establish a proposition which reveals the relation between the two proposed products $*$ and $\natural$.

Proposition 8.12. *Let* $\mathcal{F} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times k \times k}$, $\mathcal{B} \in \mathbb{R}^{n_2 \times n_1 \times n_3 \times k \times k}$ *and* $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times k}$. *Then, the following relation holds*

$$(\mathcal{F} * \mathcal{B}) * \mathcal{Y} = \mathcal{F}^* * (\mathcal{B} * \mathcal{Y}).$$

Here $\mathcal{F}^*$ *stands for an 5-mode tensor of order* $n_1 \times n_2 \times n_3 \times k \times k$ *associated with* $\mathcal{F}$ *where* $\mathcal{F}^*_{:::ij} = \mathcal{F}_{:::ji}$ *for* $1 \le i, j \le k$.

Proof. It is clear that both side of the above relation are 4-mode tensors of order $n_1 \times n_2 \times n_3 \times k$. To prove the assertion, we show that the frontal slices of both sides are equal. Let $1 \le z \le k$, we can observe that

$$\begin{aligned}
((\mathcal{F} * \mathcal{B}) * \mathcal{Y})_{:::z} &= \sum_{l=1}^{k} (\mathcal{F} * \mathcal{B})_{:::lz} * \mathcal{Y}_l \\
&= \sum_{l=1}^{k} \sum_{\mu=1}^{k} \mathcal{F}_{:::z\mu} * \mathcal{B}_{:::l\mu} * \mathcal{Y}_l \\
&= \sum_{\mu=1}^{k} \mathcal{F}_{:::z\mu} * \sum_{l=1}^{k} \mathcal{B}_{:::l\mu} * \mathcal{Y}_l \\
&= \sum_{\mu=1}^{k} \mathcal{F}_{:::z\mu} * (\mathcal{B} * \mathcal{Y})_{:::\mu} \\
&= \sum_{\mu=1}^{k} \mathcal{F}^*_{:::\mu z} * (\mathcal{B} * \mathcal{Y})_{:::\mu} = (\mathcal{F}^* * (\mathcal{B} * \mathcal{Y}))_{:::z}.
\end{aligned}$$

The result follows immediately from the above computations. □

Remark 8.13. Notice that the system (8.2) in Remark 8.7 can be rewritten in the following form

$$\mathcal{F} * \mathcal{X} = \mathcal{B},$$

where $\mathcal{F}$ is 5-mode tensor such that $\mathcal{F}_{:::ji} = \mathcal{F}_{ij}$ and $\mathcal{X}$, $\mathcal{B}$ are 4-mode tensors with frontal slices $\mathcal{X}_i$, $\mathcal{B}_i$ for $i, j = 1, 2, \ldots, k$. In theoretical point of view, it turns out that the solution of above equation can be uniquely derived if the left inverse of $\mathcal{F}$ (uniquely) exists. In fact if $\mathcal{C} = \mathcal{F}^+$ then $\mathcal{C}^* * (\mathcal{F} * \alpha) = \mathcal{C}^* * \mathcal{B}$. Now, from the previous proposition, we get $\mathcal{X} = \mathcal{C}^* * \mathcal{B}$.

We end this section by a proposition which can be established by using straightforward algebraic computations.

Proposition 8.14. *Let* $\mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathbb{R}^{n_1 \times s \times n_3 \times l}$ *and* $\mathcal{D} \in \mathbb{R}^{s \times n_2 \times n_3}$. *The following statements hold:*

1. $(\mathcal{A} + \mathcal{B}) \diamond \mathcal{C} = \mathcal{A} \diamond \mathcal{C} + \mathcal{B} \diamond \mathcal{C}$
2. $\mathcal{A} \diamond (\mathcal{B} + \mathcal{C}) = \mathcal{A} \diamond \mathcal{B} + \mathcal{A} \diamond \mathcal{C}$
3. $(\mathcal{A} \diamond \mathcal{B}) * \mathcal{D} = \mathcal{A} \diamond (\mathcal{B} * \mathcal{D})$.

8.3 Extrapolation methods based on tensor formats

In this section, we define new tensor extrapolation methods. In the first part, we present in the tensor polynomial-type extrapolation methods by using the new tensor products introduced in the preceding section. In the second part, a tensor topological *e*-algorithm is developed. We notice that when we are dealing with vectors, all these new methods reduce to the classical vector extrapolation methods.

8.3.1 Tensor polynomial-type extrapolation methods

Corresponding to a given sequence of tensors $(\mathcal{S}_n)$ in $\mathbb{R}^{n_1 \times n_2 \times n_3}$, we consider the transformation $(\mathcal{T}_k^{(n)})$ defined by

$$\mathcal{T}_k(\mathcal{S}_n) = \mathcal{T}_k^{(n)} := \begin{array}{ll} \mathbb{R}^{n_1 \times n_2 \times n_3} & \to \mathbb{R}^{n_1 \times n_2 \times n_3} \\ \mathcal{S}_n & \mapsto \mathcal{T}_k^{(n)} = \mathcal{S}_n + \mathcal{G}_{k,n} * \alpha_k \end{array} \tag{8.3}$$

where the 4-mode $\mathcal{G}_{k,n}$ with frontal slices $\mathcal{G}_i(n) \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is given for $i = 1, \ldots, k$. The 4-mode tensor α_k is unknown and its frontal slices are denoted by $\alpha_i^{(n)} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ for $i = 1, \ldots, k$. As the vector and matrix cases, in extrapolation methods, we aim to determine the unknown tensors. To this

end, we use the transformation $\tilde{T}_k^{(n)}$ obtained from $T_k^{(n)}$ as follows:

$$\tilde{T}_k^{(n)} = \tilde{T}_k(S_n) = S_{n+1} + \mathcal{G}_{k,n+1} * \alpha_k,$$

here, the 4-mode $\mathcal{G}_{k,n+1}$ has frontal slices $G_i(n+1) \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ for $i = 1, \ldots, k$. Let Δ denote the forward difference operator on the index n defined as $\Delta S_n = S_{n+1} - S_n$ and $\Delta\mathcal{G}_{k,n}$ stand for the 4-mode tensor whose frontal slices are given by $\Delta G_i(n) = G_i(n+1) - G_i(n)$ for $i = 1, \ldots, k$. The general- ized residual of $T^{(n)}$ is represented by $R(T^{(n)})$ defined as follows:

$$\begin{aligned}\tilde{R}(T_k^{(n)}) &= \tilde{T}_k(S_n) - T_k(S_n) \\ &= \Delta S_n + \Delta\mathcal{G}_{k,n} * \alpha_k. \end{aligned} \tag{8.4}$$

For an arbitrary given set of tensors $Y_1^{(n)}, \ldots, Y_k^{(n)} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, let $\tilde{\mathbf{H}}_{k,n}$ and $\tilde{\mathbf{L}}_{k,n}$ denote the subspaces generated by $\Delta G_1(n), \ldots, \Delta G_k(n)$ and $Y_1^{(n)}, \ldots, Y_k^{(n)}$ respectively. We comment that subspace generated by $\Delta G_1(n), \ldots, \Delta G_k(n)$ is the subspace generated with respect to the multiplication $*$. Evidently, we have

$$R(T_k^{(n)}) - \Delta S_n \in \tilde{\mathbf{H}}_{k,n} \tag{8.5}$$

and the unknown tensors $\alpha_i^{(n)}$ are determined by imposing the following condition,

$$R(T_k^{(n)}) \in \tilde{\mathbf{L}}_{k,n}^{\perp}.$$

Let $\mathcal{L}_{k,n}$ be a 4-mode tensor with frontal slices $Y_1^{(n)}, \ldots, Y_k^{(n)}$, the above orthogonality condition can be equivalently expressed by

$$\mathcal{L}_{k,n}^{\blacklozenge} * R(T_k^{(n)}) = \mathcal{O}, \tag{8.6}$$

where O is a zero tensor of order $n_2 \times n_2 \times n_3 \times k$. Therefore, from Proposi-tion 8.14, we have

$$\begin{aligned}\mathsf{O} &= \mathsf{L}_{k,n}\blacklozenge\, R(T_k^{(n)}) \\ &= \mathsf{L}_{k,n}\blacklozenge\, (\Delta S_n + \Delta \mathsf{G}_{k,n} * \alpha_k) \\ &= \mathsf{L}_{k,n}\blacklozenge\, \Delta S_n + \mathsf{L}_{k,n}\blacklozenge\, (\Delta \mathsf{G}_{k,n} * \alpha_k) = \mathsf{L}_{k,n}\blacklozenge\, \Delta S_n + (\mathsf{L}_{k,n}\blacklozenge\, \Delta \mathsf{G}_{k,n}) * \alpha_k.\end{aligned}$$

In fact the unknown tensor α_k can be seen as the solution of thee following tensor equation,

$$(\mathsf{L}_{k,n}\blacklozenge\, \Delta \mathsf{G}_{k,n}) * \alpha_k = -\mathsf{L}_{k,n}\blacklozenge\, \Delta S_n.$$

The choices of sequence of tensors $G_1(n), \ldots, G_k(n)$ and $Y_1^{(n)}, \ldots, Y_k^{(n)}$ determine the type of the tensor polynomial extrapolation method. In fact, for all these polynomial-type methods, the auxiliary sequence of tensors is given by $G_i(n) = \Delta S_{n+i-1}$ for $i = 1, \ldots, k$ $(n \geq 0)$. The following choices for $Y_1^{(n)}, \ldots, Y_k^{(n)}$ can be used,

$$\begin{aligned} Y_i^{(n)} &= \Delta S_{n+i-1} \qquad \text{for TMPE,} \\ Y_i^{(n)} &= \Delta^2 S_{n+i-1} \qquad \text{for TRRE,} \\ Y_i^{(n)} &= Y_i \qquad \text{for TMMPE,}\end{aligned}$$

where the operator Δ^2 refers to the second forward difference with respect to the index n defined as

$$\Delta^2 S_n = \Delta S_{n+1} - \Delta S_n \quad \text{and} \quad \Delta^2 G_i(n) = \Delta G_i(n+1) - \Delta G_i(n),$$

for $i = 1, \ldots, k$. The approximation $T_k^{(n)}$ produced by TMPE, TRRE and TMMPE can be also expressed as follows:

$$T_k^{(n)} = \sum_{j=0}^{k} S_{n+j} * \gamma_j^{(k)}$$

and the unknown tensors $\gamma_0^{(k)}, \gamma_1^{(k)}, \ldots, \gamma_k^{(k)}$ are determined by imposing the following condition

$$\sum_{j=0}^{k} \gamma_j^{(k)} = \mathrm{I}_{n_2 n_2 n_3} \quad \text{and} \quad \sum_{j=0}^{k} \eta_{i,j}^{(n)} * \gamma_j^{(k)} = \mathrm{O}_{n_2 n_2 n_3} \quad 0 \leq i < k \tag{8.7}$$

such that $\gamma_k^{(k)}$ is assumed to be invertible. Here $\mathrm{O}_{n_2 n_2 n_3} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ is the tensor with all entries equal to zero, $\eta_{i,j}^{(n)} = (Y_{i+1}^{(n)})^T * \Delta S_{n+j}$.

From now on and for simplification, we set $n = 0$. The system of equations (8.7) is given in the following form

$$\begin{cases} \gamma_0^{(k)} + \gamma_1^{(k)} + \cdots + \gamma_k^{(k)} = \mathrm{I}_{n_2 n_2 n_3} \\ (Y_1^{(0)})^T * \Delta S_0 * \gamma_0^{(k)} + (Y_1^{(0)})^T * \Delta S_1 * \gamma_1^{(k)} + \cdots + (Y_1^{(0)})^T * \Delta S_k * \gamma_k^{(k)} = \mathrm{O}_{n_2 n_2 n_3} \\ (Y_2^{(0)})^T * \Delta S_0 * \gamma_0^{(k)} + (Y_2^{(0)})^T * \Delta S_1 * \gamma_1^{(k)} + \cdots + (Y_2^{(0)})^T * \Delta S_k * \gamma_k^{(k)} = \mathrm{O}_{n_2 n_2 n_3} \\ \cdots\cdots\cdots\cdots\cdots \\ (Y_k^{(0)})^T * \Delta S_0 * \gamma_0^{(k)} + (Y_k^{(0)})^T * \Delta S_1 * \gamma_1^{(k)} + \cdots + (Y_k^{(0)})^T * \Delta S_k * \gamma_k^{(k)} = \mathrm{O}_{n_2 n_2 n_3} \end{cases} \tag{8.8}$$

Let $\beta_i^{(k)} = \gamma_i^{(k)} * (\gamma_k^{(k)})^{-1}$ where $(\gamma_k^{(k)})^{-1}$ is the inverse of $\gamma_k^{(k)}$, i.e., $\gamma_k^{(k)} * (\gamma_k^{(k)})^{-1} = \mathrm{I}_{n_2 n_2 n_3}$ for $0 \leq l \leq k$. The system of equations (8.8) can be rewritten as follows

$$\begin{cases} (Y_1^{(0)})^T * \Delta S_0 * \beta_0^{(k)} + \cdots + (Y_1^{(0)})^T * \Delta S_{k-1} * \beta_{k-1}^{(k)} = -(Y_1^{(0)})^T * \Delta S_k \\ \cdots\cdots\cdots\cdots\cdots \\ (Y_k^{(0)})^T * \Delta S_0 * \beta_0^{(k)} + \cdots + (Y_k^{(0)})^T * \Delta S_{k-1} * \beta_{k-1}^{(k)} = -(Y_k^{(0)})^T * \Delta S_k \end{cases} \tag{8.9}$$

The above system can be transformed in the following form

$$(\mathsf{L}_{k,n} \blacklozenge \mathsf{V}_k) * \mathbf{fi}_k = -(\mathsf{L}_{k,n} \blacklozenge \Delta S_k) \tag{8.10}$$

where $\mathbf{fi}_k$ is the 4-mode tensor with the k frontal slices $\beta_0^{(k)}, \ldots, \beta_{k-1}^{(k)}$ and V_k is a 4-mode tensor whose i-th frontal slice is given by ΔS_{i-1} for $i = 1, 2, \ldots, k$. We consider the case that (8.10) has unique solution (see Remark 8.7). Notice that from the first relation in (8.8), we get

$$\beta_0^{(k)} + \beta_1^{(k)} + \cdots + \beta_{k-1}^{(k)} + \mathbf{I}_{n_2 n_2 n_3} = (\gamma_k^{(k)})^{-1}.$$

Therefore, setting $\beta_k^{(k)} = \mathbf{I}_{n_2 n_2 n_3}$, we conclude that the inverse of $\sum_{l=0}^{k} \beta_l^{(k)}$ exists when $\gamma_k^{(k)}$ is assumed to be invertible. It is not difficult to verify that

$$\gamma_i^{(k)} = \beta_i^{(k)} * \Big(\sum_{l=0}^{k} \beta_l^{(k)}\Big)^{-1} \quad \text{for} \quad 0 \le i < k. \tag{8.11}$$

Having $\gamma_0, \gamma_1, \ldots, \gamma_k$ computed, we set

$$\alpha_0^{(k)} = \mathbf{I}_{n_2 n_2 n_3} - \gamma_0^{(k)}, \quad \alpha_j^{(k)} = \alpha_{j-1}^{(k)} - \gamma_j^{(k)}, \quad 1 \le j < k \quad \text{and} \quad \alpha_{k-1}^{(k)} = \gamma_k^{(k)}. \tag{8.12}$$

Setting $T_k = T_k^{(0)}$, we get

$$T_k = S_0 + \sum_{j=0}^{k-1} V_j * \alpha_j^{(k)} = S_0 + \mathsf{V}_k * \alpha_k, \tag{8.13}$$

where $V_j = \Delta S_j$ the $(j+1)$-th frontal slice of V_k for $j = 0, \ldots, k-1$ and α_k is a 4-mode tensor with frontal slices $\alpha_0^{(k)}, \ldots, \alpha_{k-1}^{(k)}$. To determine $\gamma_i^{(k)}$ for $i = 0, 1, \ldots, k$, we first we need to compute $\beta^{(k)}$ by solving system of equations (8.10). Using (8.4), (8.12) and (8.13), the generalized residual $R(T_k)$

can be also seen as follows:

$$R(T_k) = \sum_{i=0}^{k} V_i * \gamma_i^{(k)} = \mathsf{V}_{k+1} * \gamma_k \tag{8.14}$$

in which γ_k and V_{k+1} are 4-mode tensors with whose i-th frontal slices are respectively given by $\gamma_{i-1}^{(k)}$ and V_{i-1} for $i = 1, 2, \ldots, k+1$.

8.3.2 The tensor toplogical *e*- transformation

For vector sequences, Brezinski [25] proposed the well known topological *e*-algorithm (TEA) which is a generalization of the scalar *e*-algorithm [**?**] known as a technique for transforming slowly convergent or divergent se- quences. In this section, we briefly see how to extend this idea in tensor framework and define the Tensor Toplogical *e*-Transformation (TTET). The following results can be regarded as the generalization of discussions in [80, Subsection 3.2] to the tensor framework.

Let (S_n) be a given sequence of tensors in $\mathrm{R}^{n_1 \times n_2 \times n_3}$. We consider approximations $E_k(S_n) = E_k^{(n)}$ of the limit or the anti-limit of the of sequence $(S_n)_{n \in \mathrm{N}}$ such that

$$E_k^{(n)} = S_n + \sum_{i=1}^{k} \Delta S_{n+i-1} * \beta_i^{(n)}, \qquad n \geq 0. \tag{8.15}$$

where $\beta_i^{(n)} \in \mathrm{R}^{n_2 \times n_2 \times n_3}$ are the unknown tensors to be determined for $i = 1 \ldots, k$. We set

$$\tilde{E}_{k,j}^{(n)} = S_{n+j} + \sum_{i=1}^{k} \Delta S_{n+i+j-1} * \beta_i^{(n)} \qquad j = 1, \ldots, k,$$

where $\tilde{E}_{k,0}^{(n)} = E_k^{(n)}$. Let $\tilde{R}_j(E_k^{(n)})$ denote the j-th generalized residual tensor, i.e.,

$$R_j(E_k^{(n)}) = \tilde{E}_{k,j}^{(n)} - \tilde{E}_{k,j-1}^{(n)}.$$

Let $Y \in \mathrm{R}^{n_1 \times n_2 \times n_3}$ be a given third order tensor. The coefficients $\beta_i^{(n)}$ in (8.15) are computed such that

$$\sum_{i=1}^{k} (Y^T * \Delta^2 S_{n+i+j-1}) * \beta^{(n)}{}_i = \mathsf{O}, \qquad j = 0, 1 \ldots, k-1.$$

where $\mathsf{O} \in \mathrm{R}^{n_2 \times n_2 \times n_3}$. The above conditions result in the following system of equations:

$$(Y^T * \Delta^2 S_n) * \beta^{(n)}{}_1 + \cdots + (Y^T * \Delta^2 S_{n+k-1}) * \beta^{(n)}{}_k = -Y^T * \Delta S_n$$

$$\vdots \qquad\qquad \vdots$$

$$(Y^T * \Delta^2 S_{n+k-1}) * \beta_1^{(n)} + \cdots + (Y^T * \Delta^2 S_{n+2k-2}) * \beta^{(n)}{}_k = -Y^T * \Delta S_{n+k-1}$$

In the case that the above system is uniquely solvable we can obtain $F_k^{(n)}$.

8.4 Application of tensor extrapolation methods to solve ill-posed tensor problems

The main purpose of this section is to adopt the idea used by Jbilou et al. [87] for solving a class of ill-posed tensor equations. To this end, first, we need to give a brief overview on some basic concepts related to tensor SVD (TSVD) and its truncated version [90] based on the T-product of tensors.

8.4.1 TTSVD: Truncated Tensor Singular Value Decomposition

Truncating the SVD of a matrix can lead to an efficient approximation for its Moore–Penrose inverse which is helpful for solving least-squares problem. More precisely, the truncated version consumes less space of storage when the rank of matrix is not very large. This inspired Miao et al. [107] to extend the theory of TSVD [90] to truncated TSVD (TTSVD).

In this part, first, we review some existing results for TSVD and TTSVD. Then an explicit form is obtained for the minimum norm solution of least-squares problem with respect to the T-product. In the sequel, an F-diagonal tensor refers to a third order tensor whose all frontal slices are diagonal.
The following theorem is proved in [90] where the existence of TSVD is established by construction.

Theorem 8.15. *[89] Let* $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ *be a real valued-tensor, then there existsorthogonal tensors* $\mathcal{U} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ *and* $\mathcal{V} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ *such that*

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T \tag{8.16}$$

in which $\mathcal{S} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ *is an F-diagonal tensor.*

The notion of TTSVD of $\mathcal{A}$ was used in [90, 107]. In particular, the following result was proved in [107, Corollary 6].

Proposition 8.16. *[107] Let* $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ *be a real valued-tensor and*

$$\mathcal{A}_k = \mathcal{U}_{(k)} * \mathcal{S}_{(k)} * \mathcal{V}_{(k)}^T. \tag{8.17}$$

where $\mathcal{U}_{(k)} \in \mathbb{R}^{n_1 \times k \times n_3}$ *and* $\mathcal{V}_{(k)} \in \mathbb{R}^{n_2 \times k \times n_3}$ *are unitary tensors extracted fromTSVD of* $\mathcal{A}$*. The Moore–Penrose inverse of tensor* $\mathcal{A}_k$ *is given by*

$$\mathcal{A}_k^\dagger = \mathcal{V}_{(k)} * \mathcal{S}_{(k)}^\dagger * \mathcal{U}_{(k)}^T,$$

where $\mathcal{S}_k \in \mathbb{R}^{k \times k \times n_3}$ *is an F-diagonal tensor and* $k < min(n_1, n_2)$ *is called the tubal-rank of* $\mathcal{A}$ *based on the T-product.*

The expression (8.17) can be regarded the rank-k approximation of the tensor $\mathcal{A}$, denoted by $\mathcal{A} \approx \mathcal{A}_k$. We comment that the decomposition (8.17) is also called the tensor compact SVD (T-CSVD), see [107] for instance.

The TTSVD of ffi can be expressed as follows:

$$\mathrm{ffi}_k = \sum_{j=1}^{k} \bar{U}_j * d_j * \bar{V}_j^T \tag{8.18}$$

where $\bar{U}_j = \mathsf{U}_{(k)}(:, j, :) \in \mathbb{R}^{n_1 \times 1 \times n_3}$, $\bar{V}_j = \mathsf{V}_{(k)}(:, j, :) \in \mathbb{R}^{n_2 \times 1 \times n_3}$ and $d_j = \mathsf{S}(j, j, :) \in \mathbb{R}^{1 \times 1 \times n_3}$ for $j = 1, 2, \ldots, k$.

Remark 8.17. Let $\mathrm{ffi} = \mathsf{U} * \mathsf{S} * \mathsf{V}^T$ be the TSVD of ffi. In view of Proposition 8.6, we can see that

$$\bar{U}_i^T * \bar{U}_j = (\mathsf{U}^T * \mathsf{U})_{ij:} \qquad \text{and} \qquad \bar{V}_i^T * \bar{V}_j = (\mathsf{V}^T * \mathsf{V})_{ij:}.$$

Therefore, we have $\bar{U}_i^T * \bar{U}_j$ and $\bar{V}_i^T * \bar{V}_j$ are zero tubal-scalar of length n_3 for $i \neq j$; $\bar{U}_i^T * \bar{U}_i = \mathbf{e}$ and $\bar{V}_i^T * \bar{V}_i = \mathbf{e}$.

The following theorem reveals that ffi_k is an optimal approximation of a tensor, see [90] for the proof.

Theorem 8.18. *Let the TSVD of* $\mathrm{ffi} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ *be given by* $\mathrm{ffi} = \mathsf{U} * \mathsf{S} * \mathsf{V}^T$ *and for* $k < \min(n_1, n_2)$*, the tensor* ffi_k *given by* (8.18) *satisfies*

$$\mathrm{ffi}_k = \operatorname{argmin}_{\tilde{\mathrm{f}} \in M} \|\mathrm{ffi} - \tilde{\mathrm{f}}\|$$

where $M = \{\mathsf{C} = \mathsf{X} * \mathsf{Y} \mid \mathsf{X} \in \mathbb{R}^{n_1 \times k \times n_3}, \mathsf{Y} \in \mathbb{R}^{k \times n_2 \times n_3}\}$.

In view of Theorem 8.18, the Moore-Penrose inverse of tensor ffi is efficiently estimated on $\tilde{M} = \{\mathsf{C} = \mathsf{X} * \mathsf{Y} \mid \mathsf{X} \in \mathbb{R}^{n_2 \times k \times n_3}, \mathsf{Y} \in \mathbb{R}^{k \times n_1 \times n_3}\}$ by

$$\mathrm{ffi}^{\dagger} \approx \sum_{j=1}^{k} \bar{V}_j * d_j^{\dagger} * \bar{U}_j^T, \tag{8.19}$$

in which tubal-scalar $d_j^{\dagger} \in \mathbb{R}^{1 \times 1 \times n_3}$ stands for the $(j, j, :)$ entry of $\mathsf{S}_{(k)}^{\dagger}$.

The following theorem has a key role in deriving the results of the next section.

Theorem 8.19. *Assume that* $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ *and* $\mathcal{B} \in \mathbb{R}^{n_1 \times s \times n_3}$. *If* $\hat{\mathcal{X}} = \mathcal{A}^{\dagger} * \mathcal{B}$, *then*

$$\|\mathcal{A} * \hat{\mathcal{X}} - \mathcal{B}\| = \min_{\mathcal{X} \in \mathbb{R}^{n_2 \times s \times n_3}} \|\mathcal{A} * \mathcal{X} - \mathcal{B}\|. \tag{8.20}$$

Moreover, for any $\tilde{\mathcal{X}} \in \mathbb{R}^{n_1 \times s \times n_3}$ *such that* $\tilde{\mathcal{X}} \neq \hat{\mathcal{X}}$ *and* $\|\mathcal{A} * \hat{\mathcal{X}} - \mathcal{B}\| = \|\mathcal{A} * \tilde{\mathcal{X}} - \mathcal{B}\|$, *then* $\|\hat{\mathcal{X}}\| < \|\tilde{\mathcal{X}}\|$.

Proof. Let $A = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$ be the TSVD of $\mathcal{A}$. From [90, Lemma 3.19], it canbe seen that

$$\|\mathcal{A} * \mathcal{X} - \mathcal{B}\| = \|\mathcal{U}^T * (\mathcal{A} * \mathcal{X} - \mathcal{B})\|.$$

By some straightforward computations, we have

$$\|\mathcal{U}^T * (\mathcal{A} * \mathcal{X} - \mathcal{B})\| = \|\mathcal{S} * \mathcal{V}^T * \mathcal{X} - \mathcal{U}^T * \mathcal{B}\|.$$

Setting $\mathcal{Z} = \mathcal{V}^T * \mathcal{X}$ and $\mathcal{W} = \mathcal{U}^T * \mathcal{B}$, we get

$$\begin{aligned} \|\mathcal{U}^T * (\mathcal{A} * \mathcal{X} - \mathcal{B})\| &= \|\mathcal{S} * \mathcal{Z} - \mathcal{W}\| \\ &= \|(F_{n_3}^* \otimes I_{n_1})\mathrm{MatVec}(\mathcal{S} * \mathcal{Z}) - (F_{n_3}^* \otimes I_{n_1})\mathrm{MatVec}(\mathcal{W})\|_F \\ &= \|(F_{n_3}^* \otimes I_{n_1})\mathrm{bcirc}(\mathcal{S})\mathrm{MatVec}(\mathcal{Z}) - (F_{n_3}^* \otimes I_{n_1})\mathrm{MatVec}(\mathcal{W})\|_F \end{aligned} \tag{8.21}$$

where $\|\cdot\|_F$ is the well-known Frobenius matrix norm, the notation $\otimes$ stands for the Kronecker product and the matrix F_{n_3} is the discrete Fourier matrix

of size $n_3 \times n_3$ defined by (see [**?**])

$$F_{n_3} = \frac{1}{\sqrt{n_3}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{n_3-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(n_3-1)} \\ & 3 & 6 & 9 & & 3\,n\,1 \\ & \dot{1} & \dot{\omega} & \dot{\omega} & \ddots & \omega\,(\dot{3}-) \\ 1 & \omega^{n_3-1} & \omega^{2(n_3-1)} & \omega^{3(n_3-1)} & \cdots & \omega^{(n_3-1)(n_3-1)} \end{bmatrix},$$

in which $\omega = e^{-2\pi \mathbf{i}/n_3}$ is the primitive n_3–th root of unity in which $\mathbf{i} = \sqrt{-1}$and F^* denotes the conjugate transpose of F.

For simplicity we set $Z = (F_{n_3}^* \otimes I_{n_2})\text{MatVec}(\mathsf{Z})$ and $W = (F_{n_3}^* \otimes I_{n_1})\text{MatVec}(\mathsf{W})$, hence Eq.8.21 can be rewritten as follows:

$$\|\mathsf{U}^T * (\text{ffi} * \mathsf{X} - \mathsf{B})\| = \|(F_{n_3}^* \otimes I_{n_1})\text{bcirc}(\mathsf{S})(F_{n_3} \otimes I_{n_2})Z - W\|_F.$$

Notice that

$$\|Z\|_F = \|\text{MatVec}(\mathsf{Z})\|_F = \|\mathsf{Z}\| = \|\mathsf{V}^T * \mathsf{X}\| = \|\mathsf{X}\|. \tag{8.22}$$

From [**?**, **?**], it is known that there exists diagonal (rectangular) matrices $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2, \ldots, \mathbf{\Sigma}_{n_3}$ such that

$$(F_{n_3}^* \otimes I_{n_1})\text{bcirc}(\mathsf{S})(F_{n_3} \otimes I_{n_2}) = \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_1 & & & \\ & \mathbf{\Sigma}_2 & & \\ & & \ddots & \\ & & & \mathbf{\Sigma}_{n_3} \end{bmatrix},$$

and

$$(F_{n_3}^* \otimes I_{n_2})\text{bcirc}(\mathcal{S}^\dagger)(F_{n_3} \otimes I_{n_1}) = \Sigma^\dagger = \begin{bmatrix} \Sigma_1^\dagger & & & \\ & \Sigma_2^\dagger & & \\ & & \ddots & \\ & & & \Sigma_{n_3}^\dagger \end{bmatrix}.$$

From the above computations, we have

$$\|\mathcal{A} * \mathcal{X} - \mathcal{B}\| = \|\Sigma Z - W\|_F.$$

It is well-known from the literature that the minimum Frobenius norm solution of $\|\Sigma Z - W\|_F$ over $\mathbb{R}^{n_2 n_3 \times s}$ is given by $\hat{Z} = \Sigma^\dagger W$, i.e.,

$$\hat{Z} = \text{argmin}_{Z \in \mathbb{R}^{n_2 n_3 \times s}} \|\Sigma Z - W\|_F.$$

Let $\hat{\mathcal{X}}$ be the tensor such that

$$\begin{aligned} \text{MatVec}(\mathcal{V}^T * \hat{\mathcal{X}}) &= (F_{n_3} \otimes I_{n_2})\Sigma^\dagger W \\ &= (F_{n_3} \otimes I_{n_2})\Sigma^\dagger (F_{n_3}^* \otimes I_{n_1})\text{MatVec}(\mathcal{B}) \\ &= \text{bcirc}(\mathcal{S}^\dagger)\text{MatVec}(\mathcal{U}^T * \mathcal{B}). \end{aligned}$$

It is immediate to deduce the following equality

$$\text{MatVec}(\mathcal{V}^T * \hat{\mathcal{X}}) = \text{MatVec}(\mathcal{S}^\dagger * \mathcal{U}^T * \mathcal{B}),$$

which is equivalent to say that $\mathcal{V}^T * \hat{\mathcal{X}} = \mathcal{S}^\dagger * \mathcal{U}^T * \mathcal{B}$. Finally, the result follows from (8.22). □

Similar to the TSVD [90], the TTSVD can be obtained using the fast Fourier transform. Notice that circulant matrices are diagonalizable via the normalized DFT. This fact was used for proving Theorem 8.15 and deriving the

Matlab pseudocode for TSVD; see Algorithm 34 for more details. Here, we further use this fact and present a Matlab pseudocode for computing TTSVD and the corresponding approximation of Moore-Penrose inverse in Algorithm 35. Note that the Matlab function `pinv(.)` reduces to `inv(.)` when the diagonal matrix S in Step 2 of the algorithm is nonsingular.

Algorithm 34 Matlab pseudocode for TSVD

Input. $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$;
Step 1. Set $\mathcal{D} = \text{fft}(\mathcal{A}, [\,], 3)$;
Step 2. for $i = 1, 2, \ldots, n_3$
$[U, S, V] = \text{svd}(\mathcal{D}(:, :, i))$;
$\mathcal{U}(:, :, i) = U$; $\mathcal{V}(:, :, i) = V$; $\mathcal{S}(:, :, i) = S$;
end
Step 3. $\mathcal{U} = \text{ifft}(\mathcal{U}, [\,], 3)$; $\mathcal{V} = \text{ifft}(\mathcal{V}, [\,], 3)$; $\mathcal{S} = \text{ifft}(\mathcal{S}, [\,], 3)$;
Output. $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$.

Algorithm 35 Matlab pseudocode for TTSVD and the corresponding approximation of Moore–Penrose inverse

Input. $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $k \leq \min(n_1, n_2)$;
Step 1. Set $\mathcal{D} = \text{fft}(\mathcal{A}, [\,], 3)$;
Step 2. for $i = 1, 2, \ldots, n_3$
$[U, S, V] = \text{svd}(\mathcal{D}(:, :, i))$;
$\mathcal{U}(:, :, i) = U(:, 1:k)$; $\mathcal{V}(:, :, i) = V(:, 1:k)$;
$\mathcal{S}(:, :, i) = S(1:k, 1:k)$; $\mathcal{S}^{\dagger}(:, :, i) = \text{pinv}(S(1:k, 1:k))$;
end
Step 3. $\mathcal{U}_{(k)} = \text{ifft}(\mathcal{U}, [\,], 3)$; $\mathcal{V}_{(k)} = \text{ifft}(\mathcal{V}, [\,], 3)$; $\mathcal{S}_{(k)} = \text{ifft}(\mathcal{S}, [\,], 3)$; $\mathcal{S}^{\dagger}_{(k)} = \text{ifft}(\mathcal{S}^{\dagger}, [\,], 3)$;
Output. $\mathcal{A}_k = \mathcal{U}_{(k)} * \mathcal{S}_{(k)} * \mathcal{V}^T_{(k)}$ and $\mathcal{A}^{\dagger}_k = \mathcal{V}_{(k)} * \mathcal{S}^{\dagger}_{(k)} * \mathcal{U}^T_{(k)}$.

8.4.2 Tensor extrapolation methods applied to TTSVD sequences in ill-posed problems

Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ be given and consider its TSVD, i.e., let us apply Algorithm 34 for $\mathcal{A}$. In case the matrix bcirc($\mathcal{A}$) has too many singular values being close to zero, the tensor $\mathcal{A}$ is called ill-determined rank.

In this subsection, we discuss solutions of tensor equations in the following form

$$\mathcal{A} * \mathcal{X} = \mathcal{B} \tag{8.23}$$

where the ill-determined rank tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and right-hand side $\mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ are given and $\mathcal{X} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ is an unknown tensor to be determined. We assume that the right-hand side $\mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is contaminated by an error $\mathcal{E}$, i.e., $\mathcal{B} = \tilde{\mathcal{B}} + \mathcal{E}$ where $\tilde{\mathcal{B}}$ denotes the unknown error-free right-hand side. We are interested in determining an accurate approximation of the minimum norm (least-squares) solution of (8.23). From Theorem 8.19, it is known that the exact solution is given by

$$\tilde{\mathcal{X}} = \mathcal{A}^{\dagger} * \mathcal{B}.$$

Systems of tensors equations (8.23) with a tensor of ill-determined rank often are referred to as linear discrete tensor ill-posed problems. They arise in several areas in science and engineering, such as the restoration of color and multispectral images [17, 18, 99], blind source separation [101], when one seeks to determine the cause of an observed effect.

In the matrix case, when the size of the rank deficient matrix A is moderate, using the Truncated SVD is a popular method for computing an approximation for the (least-squares) solution of (in)consistent linear system of equations $Ax = b$. The approach replaces the matrix $A^{\dagger}$ by a low-rank approximation; see, e.g., Golub and Van Loan [62] or Hansen [69]. Following the same idea, we approximate the exact solution by using the expression (8.19) and available right-hand side $\mathcal{B}$, i.e., $\tilde{\mathcal{X}} \approx \mathcal{A}^{\dagger} * \mathcal{B}$. Basically, we take $\tilde{\mathcal{X}}_k \approx \tilde{\mathcal{X}}$ in which,

$$\tilde{\mathcal{X}}_k = \sum_{j=1}^{k} \bar{V}_j * d_j^{\dagger} * \bar{U}_j^T * \tilde{\mathcal{B}},$$

where $\bar{U}_j = \mathcal{U}_{(k)}(:, j, :) \in \mathbb{R}^{n_1 \times 1 \times n_3}$, $\bar{V}_j = \mathcal{V}_{(k)}(:, j, :) \in \mathbb{R}^{n_2 \times 1 \times n_3}$ and $d_j = \mathcal{S}(j, j, :) \in \mathbb{R}^{1 \times 1 \times n_3}$ are determined by Algorithm 35 for $j = 1, 2, \ldots, k$. For the matrix case, Jbilou et al. [87] proposed the application of the RRE to the sequence of vectors generated by the truncated SVD. We follow the same

idea and consider the application of TRRE to the sequence of tensors generated by the truncated TSVD (TTSVD). To do so, we set

$$Y_i = \Delta^2 S_{i-1} \quad 1 \le i \le l-2,$$

in which $l = \min(n_1, n_2)$ and $(S_k)_{k\ge 0}$ is the tensor sequence generated by the TTSVD. Thus,

$$S_k = \mathrm{ffi}_k^{\dagger} * \tilde{\mathrm{B}} = \sum_{j=1}^{k} V_j * \delta_j \tag{8.24}$$

where $\delta_j = d_j^{\dagger} * U_j^{T} * \tilde{\mathrm{B}}$ and S_0 is set to be a zero tensor of order $n_2 \times n_2 \times n_3$. It can be observed that

$$\begin{aligned} \Delta S_{k-1} &= S_k - S_{k-1} \\ &= \sum_{j=1}^{k} \bar{V}_j * \delta_j - \sum_{j=1}^{k-1} \bar{V}_j * \delta_j \\ &= \bar{V}_k * \delta_k. \end{aligned} \tag{8.25}$$

We assume here that $\delta_k^T * \delta_k$ is invertible and notice that if $\delta_k^T * \delta_k$ is zero, then we can delete the corresponding member from the sequence (8.24) and compute the next one by keeping the same index notation. To overcome the cases $\delta_k^T * \delta_k$ is numerically non invertible, we can use a small shift $e\mathrm{I}$ (say $e = 1e-8$) on the positive semi definite tensor $\delta^T{}_k * \delta_k$; see Remark 8.3.

Let $\Delta\mathcal{S}_k$ and $\Delta^2\mathcal{S}_k$ be 4-mode tensors whose i-th frontal slices are given by ΔS_{i-1} and $\Delta^2 S_{i-1}$ for $i = 1, 2, \ldots, k$, respectively, for $i = 1, 2, \ldots, k$

Notice that $\Delta^2 S_{j-1} = \bar{V}_{j+1} * \delta_{j+1} - \bar{V}_j * \delta_j$ for $j = 1, 2, \ldots, k$. Hence,

$$\begin{aligned} (\Delta^2\mathcal{S}_k \blacklozenge \Delta\mathcal{S}_k)_{:::ji} &= (\delta_{i+1}^T * \bar{V}_{i+1}^T - \delta_i^T * \bar{V}_i^T) * \bar{V}_j * \delta_j \\ &= \delta_{i+1}^T * \bar{V}_{i+1}^T * \bar{V}_j * \delta_j - \delta_i^T * \bar{V}_i^T * \bar{V}_j * \delta_j \quad \text{for} \quad i, j = 1, 2, \ldots, k. \end{aligned}$$

Using Remark 8.17, we can deduce that the nonzero frontal slices of $\Delta^2\mathcal{S}_k \blacklozenge \Delta\mathcal{S}_k$ are given by

$$(\Delta^2\mathcal{S}_k \blacklozenge \Delta\mathcal{S}_k)_{:::(i+1)i} = \delta^T_{i+1} * \delta_{i+1},$$

and

$$(\Delta^2\mathcal{S}_k \blacklozenge \Delta\mathcal{S}_k)_{:::ii} = -\delta^T_i * \delta_i.$$

Straightforward computations together with Remark 8.17 show that the frontal slices of the 4-mode tensor $(\Delta^2\mathcal{S}_k \blacklozenge \Delta\mathcal{S}_k)$ are equal to zero except the last frontal slice being equal to $\delta^T_{k+1} * \delta_{k+1}$. For notational simplicity, we define $\Theta_{i+1} = \delta^T_{i+1} * \delta_{i+1}$ for $i = 0, \ldots, k$. In summary, we need to solve the following tensor equation

$$(\Delta^2\mathcal{S}_k \blacklozenge \Delta\mathcal{S}_k) * \beta_k = -(\Delta^2\mathcal{S}_k \blacklozenge \Delta\mathcal{S}_k),$$

where β_k is a 4-mode tensor with frontal slices $\beta^{(k)}_0, \beta^{(k)}_1, \ldots, \beta^{(k)}_{k-1}$. Or equivalently, we need to find the solution of the following system of tensor equations:

$$\begin{aligned}
-\Theta_1 * \beta^{(k)}_0 + \Theta_2 * \beta^{(k)}_1 &= \mathcal{O} \\
-\Theta_2 * \beta^{(k)}_1 + \Theta_3 * \beta^{(k)}_2 &= \mathcal{O} \\
&\ddots \\
-\Theta_k * \beta^{(k)}_{k-1} &= -\Theta_{k+1}
\end{aligned}, \tag{8.26}$$

here $\mathcal{O}$ stands for zero tensor of order $n_2 \times n_2 \times n_3$. It is immediate to see that

$$\beta^{(k)}_i = \Theta^{-1}_{i+1} * \Theta_{k+1} \quad 0 \le i < k, \tag{8.27}$$

is a solution of (8.26). Evidently, we have

$$\sum_{i=0}^{k} \beta^{(k)}_i = \sum_{i=0}^{k} \Theta^{-1}_{i+1} * \Theta_{k+1}, \tag{8.28}$$

where $\beta_k^{(k)} = \mathbf{I}_{n_2 n_2 n_3}$. Note that $\sum_{i=0}^{k} \Theta_{i+1}^{-1}$ is the summation of positive definite tensors which implies its invertibility. In fact, we have

$$\left(\sum_{i=0}^{k} \beta_i^{(k)}\right)^{-1} = \Theta_{k+1}^{-1} * \left(\sum_{i=0}^{k} \Theta_{i+1}^{-1}\right)^{-1}.$$

We comment here that the assumption made on the invertibility of $\gamma_k^{(k)}$ in Subsection 8.3.1 is implicitly satisfied here. By the discussions in Subsection 8.3.1, from Eq. (8.11), we can derive $\gamma_j^{(k)}$ for $j = 0, 1, \ldots, k-1$ noticing that $\gamma_k^{(k)} = \mathbf{I}_{n_2 n_2 n_3} - \sum_{i=0}^{k-1} \gamma_i^{(k)}$. More precisely, we have

$$\begin{aligned}
\gamma_j^{(k)} &= \Theta_j^{-1} * \Theta_{k+1} * \left(\sum_{i=0}^{k} \beta_i^{(k)}\right)^{-1} \\
&= \Theta_{j+1}^{-1} * \Theta_{k+1} * \Theta_{k+1}^{-1} * \left(\sum_{i=0}^{k} \Theta_{i+1}^{-1}\right)^{-1} \\
&= \Theta_{j+1}^{-1} * \left(\sum_{i=0}^{k} \Theta_{i+1}^{-1}\right)^{-1} \qquad j = 0, 1, \ldots, k-1. \qquad (8.29)
\end{aligned}$$

Having $\gamma_0^{(k)}, \gamma_1^{(k)}, \ldots, \gamma_k^{(k)}$ obtained, we can further compute $\alpha_i^{(k)}$ for $i = 0, 1, \ldots, k-1$ by (8.12).

Finally, the extrapolated third tensor can be written as follows:

$$T_k = \Delta \mathcal{S}_k * \alpha^{(k)} \qquad (8.30)$$

where $\Delta \mathcal{S}_k$ and $\alpha^{(k)}$ are 4-mode tensors whose j-th frontal slices are receptively given by $\Delta S_{j-1} = S_j - S_{j-1} = \bar{V}_j * \delta_j$ and $\alpha^{(k)}{}_{j-1}$ for $j = 1, 2, \ldots, k$.
The generalized residual can be written in the following form

$$R(T_k) = \Delta \mathcal{S}_k * \gamma^{(k)} = \sum_{i=0}^{k} \bar{V}_{i+1} * \delta_{i+1} * \gamma_i^{(k)},$$

where $\gamma^{(k)}$ is a 4-mode tensor with frontal slices $\gamma_0^{(k)}, \gamma_2^{(k)}, \ldots, \gamma_k^{(k)}$. By Remark 8.17, one can derive

$$\begin{aligned} R(T_k)^T * R(T_k) &= \sum_{i=0}^{k} (\gamma_i^{(k)})^T * \delta_{i+1}^T * \delta_{i+1} * \gamma_i^{(k)} \\ &= \sum_{i=0}^{k} (\gamma_i^{(k)})^T * \Theta_{i+1} * \gamma_i^{(k)} \end{aligned}$$

Notice that Θ_{i+1} is symmetric, i.e., $\Theta_{i+1} = \Theta_{i+1}^T$. It can be verified that the inverse of a symmetric tensor is also a symmetric tensor. From Eqs. (8.27) and (8.29), we can observe

$$\begin{aligned} R(T_k)^T * R(T_k) &= \sum_{i=0}^{k} (\gamma_i^{(k)})^T * \Theta_{i+1} * \gamma_i^{(k)} \\ &= \sum_{i=0}^{k} (\gamma_i^{(k)})^T * \left(\sum_{l=0}^{i} \Theta_{l+1}^{-1} \right) * \Theta_{i+1} * \left(\sum_{l=0}^{i} \Theta_{l+1}^{-1} \right) \\ &= \sum_{j=0}^{k} \Theta_{j+1}^{-1} = \Theta_k * \Theta_k^{-1} * \sum_{j=0}^{k} \Theta_{j+1}^{-1}. \end{aligned}$$

Now we can deduce that (see the proof of [90, Lemma 3.19] for more details)

$$\|R(T_k)\|^2 = \operatorname{trace}\left(\Theta_k * \gamma_{k-1}^{(k)} \right)_{::1}. \tag{8.31}$$

For ill-posed problems, the value of $\|R(T_k)\|$ decrease when k increases and is sufficiently small. However, the norm of $R(T_k)$ may increase with k. Hence, similar to [87], we may need to simultaneously exploit an alternative stopping criterion when the problem is ill-posed. To this end, we can stop iterations once either $\|R(T_k)\|$ or

$$\eta_k := \frac{\|T_{k+1} - T_k\|}{\|T_k\|} = \frac{\sqrt{\operatorname{trace}\left(((T_{k+1} - T_k)^T * (T_{k+1} - T_k)) \right)_{::1}}}{\sqrt{\operatorname{trace}\left(T_k^T * T_k \right)_{::1}}} \tag{8.32}$$

is smaller than a prescribed tolerance. Using Remark 8.17 and some computations, one may simplify the above relation exploiting the following relations,

$$(T_{k+1} - T_k)^T * (T_{k+1} - T_k) = \sum_{j=1}^{k} (\alpha_{j-1}^{(k+1)} - \alpha_{j-1}^{(k)})^T * \Theta_j * (\alpha_{j-1}^{(k+1)} - \alpha_{j-1}^{(k)}) + \alpha_{k}^{(k+1)} * \Theta_{k+1} * \alpha_{k}^{(k+1)},$$

and

$$T_k^T * T_k = \sum_{j=1}^{k} (\alpha_{j-1}^{(k)})^T * \Theta_j * \alpha_{j-1}^{(k)}.$$

We end this part by summarizing the above discussions in Algorithm **??**.

Algorithm 36 The TRRE-TTSVD Algorithm

Input. $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, tolerance e and $k \leq \min(n_1, n_3)$ for TTSVD of $\mathcal{A}$;

Step 1. Run Algorithm 34 to compute an approximation for its Moore-Penrose inverse, i.e., $\mathcal{A}^\dagger = \mathcal{V} * \mathcal{S}^\dagger * \mathcal{U}^T$.

Step 2. Set $S_0 = \mathcal{O}$, $S_1 = \bar{V}_1 * d_1^\dagger * \bar{U}_1^T * \bar{\mathcal{B}}$, $T_1 = S_1$ and $k = 2$.

Step 3. tol $= 1$.

Step 4. while tol $< e$

Compute S_k from (8.24)

Compute $\gamma_j^{(k)} = \Theta_{l+1}^{-1} * \left(\sum_{l=0}^{k} \Theta_{l+1}^{-1} \right)^{-1}$ for $j = 0, 1, \ldots, k-1$

Set $\gamma_k^{(k)} = \mathcal{I}_{n_2 n_2 n_3} - \sum_{i=0}^{k-1} \gamma_i^{(k)}$.

Compute

$$\alpha_0^{(k)} = \mathcal{I}_{n_2 n_2 n_3} - \gamma_0^{(k)}, \quad \alpha_j^{(k)} = \alpha_{j-1}^{(k)} - \gamma_j^{(k)}, \quad 1 \leq j < k \quad \text{and} \quad \alpha_{k-1}^{(k)} = \gamma_k^{(k)}.$$

Compute the approximation T_k using (8.30).

Compute $\|R(T_k)\|$ (cf. (8.31)) and η_k (cf. (8.32)). tol $= \min(\|R(T_k)\|, \eta_k)$.

$k = k + 1$.

end

8.5 Numerical experiments

In this part we consider a test problem to numerically compare the performance of Algorithm 36 with TTSVD for solving ill-conditioned problems. All computations are carried out using MATLAB with machine epsilon about 10^{-16}.

The desired solution $\tilde{\mathcal{X}}$ is chosen as a tensor with all entries being equal to one. The error-free right-hand side is given by $\tilde{\mathcal{B}} = \mathcal{A} * \tilde{\mathcal{X}}$ and the associated error-contaminated right-hand side $\mathcal{B} = \tilde{\mathcal{B}} + \mathcal{E}$ where the error-tensor $\mathcal{E}$ has normally distributed entries with zero mean and is normalized to correspond to a specific noise-level

$$\nu = \frac{\|\mathcal{E}\|}{\|\tilde{\mathcal{B}}\|} \tag{8.33}$$

For all the experiments, we reported the relative error norms

$$\|\mathcal{S}_k - \tilde{\mathcal{X}}\|/\|\tilde{\mathcal{X}}\| \text{ and } \|\mathcal{T}_k - \tilde{\mathcal{X}}\|/\|\tilde{\mathcal{X}}\|$$

corresponding to TTSVD and TRRE-TSVD, respectively.

We used the Regularization Tools package by Hansen [**?**] to define the frontal slices $A_1 = shaw$, $A_2 = baart$ and $A_3 = foxgood$ of the tensor $\mathcal{A}$. The computed condition numbers of the frontal slices of $\mathcal{A}$ are $\kappa(A_1) = 1.25 \cdot 10^{20}$ and $\kappa(A_2) = 1.5 \cdot 10^{18}$ and $\kappa(A_3) = 7 \cdot 10^{19}$ where $\kappa(A_i) = \|A_i\|_2 \, \|A_i^{-1}\|_2$. ONotice that only a small number of those singular values are not close de zero which means that the problem is very ill-conditioned. The computed tubal rank (the number of nonzero elements in the first frontal slices of $\mathcal{S}$) is $rank_t(\mathcal{A}) = 32$. We set $n_1 = n_2 = 400$, $n_3 = 3$ and chose $\nu = 10^{-2}, 10^{-3}$. The obtained relative error norms corresponding to approximate solutions computed by TRRE-TTSVD and TTSVD are reported in Table 8.1 and Table 8.2.

Method	TRRE-TTSVD	TTSVD
k	$\|\mathcal{T}_k - \tilde{\mathcal{X}}\|/\|\tilde{\mathcal{X}}\|$	$\|\mathcal{S}_k - \tilde{\mathcal{X}}\|/\|\tilde{\mathcal{X}}\|$
6	$2.2 \cdot 10^{-1}$	$1.2 \cdot 10^{-1}$
10	$4.3 \cdot 10^{-2}$	$1.7 \cdot 10^{-1}$

Table 8.1: *The computed relative errors for TTSVD and TRRE-TTSVD ($\nu = 10^{-2}$).*

We observed numerically that after iteration $k = 10$ the error-norm for TTSVD begins to increase while the one for TRRE-TTSVD stagnates and this is the same phenomena observed in the vector case.

Method	TRRE-TTSVD	TTSVD
k	$\|\mathcal{T}_k - \tilde{\mathcal{X}}\|/\|\tilde{\mathcal{X}}\|$	$\|\mathcal{S}_k - \tilde{\mathcal{X}}\|/\|\tilde{\mathcal{X}}\|$
4	$4.1 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$
12	$4.2 \cdot 10^{-3}$	$2.4 \cdot 10^{-2}$

Table 8.2: *The computed relative errors for TTSVD and TRRE-TTSVD ($\nu = 10^{-3}$).*

8.6 Conclusion

We proposed in this chapter some tensor extrapolation methods. The first class contains the tensor extrapolation polynomial-type methods while for the second class, we introduced the tensor topological *e*-transformation. These techniques can be regarded as generalizations of the well-known vector and matrix extrapolation methods. Besides, we introduced some new products between two tensors which can simplify the derivation of extrapolation methods based on tensor format. Some theoretical results were also established including the properties of introduced tensor products and an expression for the minimum norm least-square solution of a tensor equation. Finally, the proposed techniques were applied on the sequence of tensors corresponding to the truncated tensor singular value decomposition which can be used for solving tensor ill-posed problems.

Chapter 9

Tensor global extrapolation methods using the n-mode and the Einstein products

In this chapter, we present new Tensor extrapolation methods as generalizations of well known vector, matrix and block extrapolation methods such as polynomial extrapolation methods or *e*-type algorithms. We will define new tensor products that will be used to introduce global tensor extrapolation methods. We discuss the application of these methods to the solution of linear and non linear tensor systems of equations and propose an efficient implementation of these methods via the Global- QR decomposition.

9.1 Introduction

In the present chapter , we consider tensor sequence transformations and propose new tensor extrapolation methods that generalize the classical vector ones. Using the Einstein product, we define some new tensor products that will allow us to develop the new methods based on orthogonal or oblique projections onto subspaces of small dimensions. It will be shown

that when the tensor sequence is generated linearly then the proposed methods are theoretically equivalent to some tensor Krylov subspace methods such as the tensor version of GMRES and Lanczos methods developed recently in [24].

The remainder of this chapter is organized as follows. In Section 9.2, we give notations, some basic definitions and properties related to tensors. In Section9.3, we introduce the tensor versions of the vector polynomial extrapolation methods namely the Tensor Global Reduced Rank Extrapolation (TG-RRE), Tensor Global Minimal Polynomial Extrapolation (TG-MPE) Tensor Global Modified Minimal Polynomial Extrapolation (TG-MMPE). We also give a Global tensor version of the topological *e*-transformation. Section 9.4 describes the application of the proposed methods for solving linear and nonlinear tensor system of equations. In Section 9.5, we introduce efficient implementations via the tensor global-QR decomposition and in the last section, we present some numerical experiments.

9.2 Preliminaries and notations

In the following, we introduce the new $\copyright^{(N+M+1)}$ product between two tensors which is a generalization of diamond product introduced in .

Definition 9.1. The $\copyright^{(N+M+1)}$ tensor product between two $(M + N + 1)$-mode tensors $\mathcal{X} = [\mathcal{X}_1, \ldots, \mathcal{X}_l] \in \mathbb{R}^{I_1 \times I_2 \times I_3 \ldots \times I_N \times J_1 \times J_2 \times J_3 \ldots \times J_M \times l}$ and $\mathcal{Y} = [\mathcal{Y}_1, \ldots, \mathcal{Y}_p] \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M \times p}$ where $\mathcal{X}_i \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$ and $\mathcal{Y}_j \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$, is the $l \times p$ matrix whose (i, j) entries, $i = 1, \ldots, l$ and $j = 1, \ldots, p$, is given by

$$(\mathcal{X} \copyright^{(N+M+1)} \mathcal{Y})_{i,j} = \langle \mathcal{X}_i, \mathcal{Y}_j \rangle = tr(\mathcal{Y}^T \overset{*}{}_N \mathcal{X}_i).$$

Next, we give a proposition that will be used later.

Proposition 9.2. *Assume that* $\mathcal{U} = [\mathcal{U}_1, \ldots, \mathcal{U}_m] \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times J_1 \times J_2 \times \ldots \times J_M \times m}$ *is an* $(N + M + 1)$*-mode tensor such that* $\mathcal{U}_1, \ldots, \mathcal{U}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times J_1 \times J_2 \times \ldots \times J_M}$ *and let* $y = (y_1, \ldots, y_m)^T \in \mathbb{R}^m$*. Then for an arbitrary* $(N + M + 1)$*-mode tensor* $\mathcal{Y} = [\mathcal{Y}_1, \ldots, \mathcal{Y}_m]$ *such that* $\mathcal{Y}_1, \ldots, \mathcal{Y}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times J_1 \times J_2 \times \ldots \times J_M}$*, we have:*

$$\mathcal{Y} \circledcirc^{(N+M+1)} (\mathcal{U} \tilde{\times}_{(N+M+1)} y) = (\mathcal{Y} \circledcirc^{(N+M+1)} \mathcal{U}) y,$$

Proof. The proof comes easily from the definition of the two involved tensor products. □

Definition 9.3. The set of $(N + M)$-mode tensors $\mathcal{U}_1, \ldots, \mathcal{U}_m \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_M}$ is called orthonormal if

$$\langle \mathcal{U}_i, \mathcal{U}_j \rangle = \delta_{i,j} (= 1 \; if \; i = j \; and \; 0 \; elswhere).$$

Suppose that $\mathcal{U} = [\mathcal{U}_1, \ldots, \mathcal{U}_m] \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times J_1 \times J_2 \times \ldots \times J_M \times m}$ is an $(N + M + 1)$-mode tensor such that $\mathcal{U}_1, \ldots, \mathcal{U}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times J_1 \times J_2 \times \ldots \times J_M}$. If the $(M + N)$-mode tensors $\mathcal{U}_1, \ldots, \mathcal{U}_m$ are orthonormal, then we have :

$$\mathcal{U} \circledcirc^{(N+M+1)} \mathcal{U} = I_m.$$

9.3 Tensor extrapolation methods

In this section, we present two classes of new tensor global extrapolation methods. The first class contains the tensor polynomial based methods while the second class is devoted to the global tensor topological *e*-algorithm.

9.3.1 Tensor global-polynomial extrapolation methods

Let $(\mathcal{S}_n)$ be a sequence of tensors of $\in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N \times K_1 \times K_2 \ldots \times K_M}$ and consider the transformation T_k, $k = 1, 2, \ldots$ from $\mathbb{R}^{I_1 \times I_2 \ldots \times I_N \times K_1 \times K_2 \ldots \times K_M}$ onto $\mathbb{R}^{I_1 \times I_2 \ldots \times I_N \times K_1 \times K_2 \ldots \times K_M}$

and defined by

$$\mathcal{T}_k^{(n)} = T_k(\mathcal{S}_n) = \mathcal{S}_n + \sum_{j=0}^{k-1} \alpha_j^{(k)} \mathcal{G}_j(n) \tag{9.1}$$

$$= \mathcal{S}_n + \mathcal{H}_k(n) \tilde{\times}_{(N+M+1)} \alpha^{(k)} \tag{9.2}$$

where $\alpha^{(k)} = (\alpha_0^{(k)}, \ldots, \alpha_{k-1}^{(k)})^T \in \mathbb{R}^k$ and $\mathcal{H}_k(n) = [\mathcal{G}_0(n), \ldots, \mathcal{G}_{k-1}(n)]$ is the (N+M+1)-mode tensor defined from the given auxiliary tensor sequences $(\mathcal{G}_i(n))_n \in \mathbb{R}^{I_1 \times \ldots \times I_N \times K_1 \times \ldots \times K_M}$; $i = 0, \ldots, k-1$. We will see later how tochoose the vector $\alpha^{(k)} \in \mathbb{R}^k$.

Let $\tilde{T}_k$ denote the new transformation obtained from (T_k) as follows:

$$\tilde{T}_k(\mathcal{S}_n) = \mathcal{S}_{n+1} + \sum_{j=0}^{k-1} \alpha_j^{(k)} \mathcal{G}_j(n+1) \tag{9.3}$$

$$= \mathcal{S}_{n+1} + \mathcal{H}_k(n+1) \tilde{\times}_{(N+M+1)} \alpha^{(k)}. \tag{9.4}$$

Notice that the scalars $\alpha_j^{(k)}$ are the same in the expressions of $T_k(\mathcal{S}_n)$ and $\tilde{T}_k(\mathcal{S}_n)$. We define now the generalized residual of $\mathcal{T}_k^{(n)}$ by

$$\tilde{\mathcal{R}}_k^n = \tilde{\mathcal{R}}(\mathcal{T}_k^{(n)}) = \tilde{T}_k(\mathcal{S}_n) - T_k(\mathcal{S}_n)$$

$$= (\mathcal{S}_{n+1} - \mathcal{S}_n) + \sum_{j=0}^{k-1} \alpha_j^{(k)} (\mathcal{G}_j(n+1) - \mathcal{G}_j(n))$$

$$= \Delta\mathcal{S}_n + \sum_{j=0}^{k-1} \alpha_j^{(k)} (\Delta\mathcal{G}_j(n)).$$

Then we get

$$\tilde{\mathcal{R}}(\mathcal{T}_k^{(n)}) = \Delta\mathcal{S}_n + \Delta\mathcal{H}_k(n) \tilde{\times}_{(N+M+1)} \alpha^{(k)}, \tag{9.5}$$

where the first forward differences $\Delta\mathcal{S}_n = \mathcal{S}_{n+1} - \mathcal{S}_n$ and $\Delta\mathcal{H}_k(n) = [\Delta\mathcal{G}_0(n), \ldots, \Delta\mathcal{G}_{k-1}(n)] \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N \times K_1 \times K_2 \ldots \times K_M \times k}$. The vector $\alpha^{(k)}$ is obtained from the orthogonality

relation:

$$\tilde{\mathrm{R}}(T_k^{(n)}) \in (span\{\mathrm{Y}_0^{(n)}, \ldots, \mathrm{Y}_{k-1}^{(n)}\})^{\perp} \tag{9.6}$$

where $\mathrm{Y}_0^{(n)}, \ldots, \mathrm{Y}_{k-1}^{(n)} \in \mathrm{R}^{I_1 \times I_2 \times \ldots \times I_N \times K_1 \times K_2 \times \ldots \times K_M}$ are given tensors.
Here, the $span\{\mathrm{Y}_0^{(n)}, \ldots, \mathrm{Y}_{k-1}^{(n)}\}$ is the tensor subspace generated by the tensors $\mathrm{Y}_0^{(n)}, \ldots, \mathrm{Y}_{k-1}^{(n)}$.

If $\tilde{\mathbf{H}}_{k,n}$ and $\tilde{\mathbf{L}}_{k,n}$ denote the tensor subspaces $\tilde{\mathbf{H}}_{k,n} = span\{\Delta \mathrm{G}_0(n), \ldots, \Delta \mathrm{G}_{k-1}(n)\}$ and $\tilde{\mathbf{L}}_{k,n} = span\{\mathrm{Y}_0^{(n)}, \ldots, \mathrm{Y}_{k-1}^{(n)}\}$, then from (9.5) and (9.6), the generalized residual satisfies the following relations:

$$\tilde{\mathrm{R}}(T_k^{(n)}) - \Delta \mathrm{S}_n \in \tilde{\mathbf{H}}_{k,n} \tag{9.7}$$

and

$$\tilde{\mathrm{R}}(T_k^{(n)}) \in (\tilde{\mathbf{L}}_{k,n})^{\perp}. \tag{9.8}$$

Let $\mathrm{L}_{k,n} = [\mathrm{Y}_0^{(n)}, \ldots, \mathrm{Y}_{k-1}^{(n)}] \in \mathrm{R}^{I_1 \times I_2 \ldots \times I_N \times K_1 \times K_2 \ldots \times K_M \times k}$, then the relations (9.7) and (9.8) could be expressed as follows :

$$\tilde{\mathrm{R}}(T_k^{(n)}) - \Delta \mathrm{S}_n = \Delta \mathrm{H}_k(n) \tilde{\times}_{(N+M+1)} \alpha^{(k)} \tag{9.9}$$

and

$$\mathrm{L}_{k,n} \circledcirc^{(N+M+1)} \tilde{R}(T_k^{(n)}) = 0. \tag{9.10}$$

Assuming that $(\mathrm{L}_{k,n} \circledcirc^{N+M+1} \Delta \mathrm{H}_k(n))$ is nonsingular, the vector $\alpha^{(k)}$ appearing in the expression (9.5) of the generalized residual $\tilde{\mathrm{R}}(T_k^{(n)})$ is given as:

$$\alpha^{(k)} = -(\mathrm{L}_{k,n} \circledcirc^{(N+M+1)} \Delta \mathrm{H}_k(n))^{-1} (\mathrm{L}_{k,n} \circledcirc^{(N+M+1)} \Delta \mathrm{S}_n). \tag{9.11}$$

Therefore, the approximation $T_k^{(n)}$ is computed as follows

$$T_k^{(n)} = \mathrm{S}_n + \mathrm{H}_k(n) \tilde{\times}_{(N+M+1)} \alpha^{(k)} \tag{9.12}$$

For the tensor global polynomial extrapolation methods, namely Tensor Global MPE (TG-MPE),Tensor Global RRE (TG-RRE) and Tensor Global MMPE (TG-MMPE), the auxiliary sequences are given as

$$G_l^{(n)} = \Delta S_{(n+l)},\ l = 0, \ldots, k-1;\ n \geq 0$$

Let $\Delta^i V_k(n)$ be the tensor defined by

$$\Delta^i V_k(n) = [\Delta^i S_n, \ldots, \Delta^i S_{n+k-1}] \in R^{I_1 \times I_2 \ldots \times I_N \times K_1 \times K_2 \ldots \times K_M \times k}, \qquad i = 1, 2,$$

where Δ^2 is the second forward difference $\Delta^2 S_n = \Delta S_{n+1} - \Delta S_n$.

In this case, the approximations $T_k^{(n)}$ given in relation (9.12) can be expressed as :

$$T_k^{(n)} = S_n - \Delta V_k(n) \underset{(N+M)}{\times} 1 ((L_{k,n} \odot^{(N+M+1)} \Delta^2 V_k(n))^{-1} (L_{k,n} \odot^{(N+M+1)} \Delta S_n)). \tag{9.13}$$

It is clear that $T_k^{(n)}$ exists and is unique if and only if the square matrix $L_{k,n} \odot^{(N+M+1)} \Delta^2 V_k(n)$ is nonsingular. The generalized residual can be expressed as follows

$$\tilde{R}(T_k^{(n)}) = \Delta S_n + \Delta^2 V_k(n) \tilde{\times}_{(N+M+1)} ((L_{k,n} \odot^{(N+M+1)} \Delta^2 V_k(n))^{-1} (L_{k,n} \odot^{(N+M+1)} \Delta S_n)). \tag{9.14}$$

The choice of the tensors $Y_0^{(n)}, \ldots, Y_{k-1}^{(n)} \in R^{I_1 \times I_2 \times \ldots \times I_N \times K_1 \times K_2 \times \ldots \times K_M}$ required in the orthogonality relation (**??**) determines the global-polynomial tensor

extrapolation method. For the three well known extrapolation polynomial-based methods, we have the following choices

$$\begin{aligned} Y_l^{(n)} &= \Delta S_{n+l} \quad \text{for TG-MPE,} \\ Y_l^{(n)} &= \Delta^2 S_{n+l} \quad \text{for TG-RRE,} \\ Y_{l+1}^{(n)} &= Y_{n+l} \quad \text{for TG-MMPE.} \end{aligned}$$

Next, we will propose an efficient implementation of theses methods. For this purpose, we first express the approximation $T_k^{(n)}$ given in relation (9.1) in a different way.

Using the relation (9.1), the fact that $G_l^{(n)} = \Delta S_{(n+l)}$, $l = 0, \ldots, k-1$; $n \geq 0$, and $\tilde{R}(T_k^{(n)}) \in (span\{Y_0^{(n)}, \ldots, Y_{k-1}^{(n)}\})^{\perp}$, the (TG-RRE), (TG-MPE) and (TG-MMPE) extrapolation methods produce approximations $T_k^{(n)}$ of the form

$$T_k^{(n)} = \sum_{j=0}^{k} \gamma_j^{(k)} S_{n+j}, \tag{9.15}$$

where

$$\sum_{j=0}^{k} \gamma_j^{(k)} = 1 \quad \text{and} \quad \sum_{j=0}^{k} \eta_{l,j} \gamma_j^{(k)} = 0 \quad 0 \leq l < k, \tag{9.16}$$

with $\eta_{l,j} = \langle Y_l^{(n)}, \Delta S_{n+j} \rangle$.

The system of linear equations (9.16) can be written as

$$
\begin{cases}
\gamma_0^{(k)} + \gamma_1^{(k)} + \ldots + \gamma_k^{(k)} = 1 \\
\gamma_0^{(k)}(\mathrm{Y}_0^{(n)}, \Delta \mathrm{S}_n) + \gamma_1^{(k)}(\mathrm{Y}_1^{(n)}, \Delta \mathrm{S}_{n+1}) + \ldots + \gamma_k^{(k)}(\mathrm{Y}_k^{(n)}, \Delta \mathrm{S}_{n+k}) = 0 \\
\gamma_0^{(k)}(\mathrm{Y}_1^{(n)}, \Delta \mathrm{S}_n) + \gamma_1^{(k)}(\mathrm{Y}_2^{(n)}, \Delta \mathrm{S}_{n+1}) + \ldots + \gamma_k^{(k)}(\mathrm{Y}_{k+1}^{(n)}, \Delta \mathrm{S}_{n+k}) = 0 \\
\ldots\ldots\ldots\ldots\ldots\ldots \\
\gamma_0^{(k)}(\mathrm{Y}_{k-1}^{(n)}, \Delta \mathrm{S}_n) + \gamma_1^{(k)}(\mathrm{Y}_k^{(n)}, \Delta \mathrm{S}_{n+1}) + \ldots + \gamma_k^{(k)}(\mathrm{Y}_{2k-1}^{(n)}, \Delta \mathrm{S}_{n+k}) = 0
\end{cases}
\tag{9.17}
$$

Let $\beta_l^{(k)} = \gamma_0^{(k)} / \gamma_k^{(k)}$ for $0 \le l \le k$, then

$$
\gamma_l^{(k)} = \frac{\beta_l^{(k)}}{\sum_{l=0}^{k} \beta_l^{(k)}} \quad \text{for} \quad 0 \le l < k \quad \text{and} \quad \beta_k^{(k)} = 1. \tag{9.18}
$$

With these notations, the linear system of equations (8.8) becomes

$$
\begin{cases}
\beta_0^{(k)}(\mathrm{Y}_0^{(n)}, \Delta \mathrm{S}_n) + \ldots + \beta_{k-1}^{(k)}(\mathrm{Y}_{k-1}^{(n)}, \Delta \mathrm{S}_{n+k-1}) = -(\mathrm{Y}_0^{(n)}, \Delta \mathrm{S}_{n+k}) \\
\ldots\ldots\ldots\ldots\ldots\ldots \\
\beta_0^{(k)}(\mathrm{Y}_{k-1}^{(n)}, \Delta \mathrm{S}_n) + \ldots + \beta_{k-1}^{(k)}(\mathrm{Y}_{2k-1}^{(n)}, \Delta \mathrm{S}_{n+k-1}) = -(\mathrm{Y}_{k-1}^{(n)}, \Delta \mathrm{S}_{n+k}).
\end{cases}
\tag{9.19}
$$

The above system of equations can also be expressed in the following compact form

$$
(\mathrm{L}_{k,n} \, ©^{(N+M+1)} \, \Delta \mathrm{V}_k(n)) \beta^{(k)} = -(\mathrm{L}_{k,n} \, ©^{(N+M+1)} \, \Delta \mathrm{S}_{n+k}) \tag{9.20}
$$

where $\beta^{(k)} = [\beta_0^{(k)}, \ldots, \beta_{k-1}^{(k)}]^T$. Assume now that $\gamma_0^{(k)}, \gamma_1^{(k)}, \ldots, \gamma_k^{(k)}$ have been calculated and introduce the new variables

$$
\delta_0^{(k)} = 1 - \gamma_0^{(k)}, \qquad \delta_j^{(k)} = \delta_{j-1}^{(k)} - \gamma_j^{(k)}, \quad 1 \le j < k \quad \text{and} \quad \delta_{k-1}^{(k)} = \gamma_k^{(k)}, \tag{9.21}
$$

then the tensor approximation $T_k^{(n)}$ can be expressed as

$$T_k^{(n)} = S_n + \sum_{j=0}^{k-1} \delta_j \Delta S_{n+j} = S_n + \Delta_n V_k(\tilde{x}_{(N+M+1)}) \delta^{(k)} \tag{9.22}$$

where $\delta^{(k)} = [\delta_0^{(k)}, \ldots, \delta_{k-1}^{(k)}]^T$.

9.3.2 The tensor global topological *e*-transformation

In [25], Brezinski proposed a generalization of the scalar *e*-algorithm for vector sequences called the topological *e*-algorithm (TEA). The matrix case has been introduced by Jbilou and Sadok in [84]. In this section we define the tensor global topological *e*-transformation (TG-TET).

Let (S_n) be a sequence of tensors of $\in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N \times K_1 \times K_2 \ldots \times K_M}$ and consider approximations $E_k(S_n) = E_k^{(n)}$ of the limit of the tensor sequence $(S_n)_{n \in \mathbb{N}}$ such that

$$E_k^{(n)} = S_n + \sum_{i=1}^{k} a_i \Delta S_{n+i-1}, \qquad n \geq 0. \tag{9.23}$$

where $a_i^{(n)} \in \mathbb{R}$, for $i = 1, \ldots k$. We introduce the new tensor transformation $\tilde{E}_{k,j}^{(n)}$, $j = 1, \ldots k$ defined by

$$\tilde{E}_{k,j}^{(n)} = S_{n+j} + \sum_{i=1}^{k} a_i \Delta S_{n+i+j-1} \quad j = 1, \ldots, k. \tag{9.24}$$

We set $\tilde{E}_{k,0}^{(n)} = E_k^{(n)}$ and define the j-th tensor generalized residual as follows

$$\tilde{R}_j(E_k^{(n)}) = \tilde{E}_{k,j}^{(n)} - \tilde{E}_{k,j-1}^{(n)}. \tag{9.25}$$

The coefficients $a_i^{(n)}$ involved in the expression (8.15) of $E_k^{(n)}$ are computed such that each *j*-th generalized residual is orthogonal to some chosen tensor

$\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N \times K_1 \times K_2 \ldots \times K_M}$, that is

$$\langle \mathcal{Y}, \tilde{\mathcal{R}}_j(\mathcal{E}_k^{(n)}) \rangle = 0; \qquad j = 1, \ldots, k. \tag{9.26}$$

Let $D_{k,n}$ denote the following matrix

$$D_{k,n} = \begin{bmatrix} \langle \mathcal{Y}, \Delta^2 \mathcal{S}_n \rangle & \ldots\ldots\ldots & \langle \mathcal{Y}, \Delta^2 \mathcal{S}_{n+k-1} \rangle \\ \langle \mathcal{Y}, \Delta^2 \mathcal{S}_{n+1} \rangle & \ldots\ldots\ldots & \langle \mathcal{Y}, \Delta^2 \mathcal{S}_{n+k} \rangle \\ \vdots & \vdots & \vdots \\ \langle \mathcal{Y}, \Delta^2 \mathcal{S}_{n+k-1} \rangle & \ldots\ldots\ldots & \langle \mathcal{Y}, \Delta^2 \mathcal{S}_{n+2k-2} \rangle \end{bmatrix}. \tag{9.27}$$

Then , from the orthogonality relation (**??**), $a^{(n)} = (a_1^{(n)}, \ldots, a_k^{(n)})^T \in \mathbb{R}^k$ is given by

$$a^{(n)} = -D_k^{-1} z_n^{(n)} \tag{9.28}$$

where $z^{(n)} = (\langle \mathcal{Y}, \Delta \mathcal{S}_n \rangle, \ldots, \langle \mathcal{Y}, \Delta \mathcal{S}_{n+k-1} \rangle)^T$. We assume that the matrix $D_{k,n}$ is nonsingular. Hence, the approximation $\mathcal{E}_k^{(n)}$ exists, is unique and is expressed as

$$\mathcal{E}_k^{(n)} = \mathcal{S}_n + \Delta \mathcal{W}_k(n) \tilde{\times}_{(N+M+1)} \beta^{(n)} \tag{9.29}$$

where $\mathcal{W}_k(n) = [\Delta \mathcal{S}_n, \ldots, \Delta \mathcal{S}_{n+k-1}]$.

In the next sections, we will see how these tensor global extrapolation methods would be applied for solving linear and nonlinear tensor equations.

9.4 Application to tensor linear/ non linear systems of equations

9.4.1 Application to tensor linear systems

Consider the following tensor linear system of equations

$$\mathcal{A} *_N \mathcal{X} = \mathcal{B}, \tag{9.30}$$

where $\mathcal{A}$ is a tensor in $\mathbb{R}^{I_1\times I_2\times \ldots\times I_N\times I_1\times I_2\times \ldots\times I_N}$, $\mathcal{B} \in \mathbb{R}^{I_1\times I_2\times \ldots\times I_N\times J_1\times J_2\times \ldots\times J_M}$ and $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times \ldots\times I_N\times J_1\times J_2\times \ldots\times J_M}$ is the unknown tensor to be determined. We assume here that the square tensor $\mathcal{A}$ is nonsingular so that (9.30) has a unique solution $\mathcal{X}^* = \mathcal{A}^{-1} * \mathcal{B}$. The purpose here is to apply tensor extrapolation methods to the problem (9.30) to compute approximate solutions to the solution $\mathcal{X}^*$ of (9.30). Starting from an initial tensor guess $\mathcal{S}_0$, we construct the linear sequence $(\mathcal{S}_n)_n$ as follows

$$\mathcal{S}_{n+1} = (\mathcal{I} - \mathcal{A}) *_N \mathcal{S}_n + \mathcal{B} \tag{9.31}$$

Notice that if the sequence $(\mathcal{S}_n)_n$ is convergent, then its limit $\mathcal{S} = \mathcal{X}^*$ is the solution of the system (9.30).

In [48, 77], tensor Krylov subspaces via the Einstein product have been introduced including the tensor GMRES and some tensor based Lanczos methods. The next theorem shows that, when applied to the generated linear sequence (9.31), the proposed tensor extrapolation methods are Tensor Krylov subspace methods and are mathematically equivalent to some well known Krylov based methods such as the tensor GMRES.

Theorem 9.4. *When applied to the sequence generated by (9.31), TG-RRE, and TG-MPE are tensor Krylov subspace methods and are mathematically equivalent to the tensor global GMRES and the tensor global Arnoldi methods, respectively.*

Proof. Notice first that for tensor linear sequences (9.31) the generalized residual becomes the true residual. In fact, from (9.31), we have $\Delta \mathcal{S}_n = \mathcal{B} - \mathcal{A} *_N \mathcal{S}_n = \mathcal{R}(\mathcal{S}_n)$ the residual of the tensor $\mathcal{S}_n$. Since $\Delta^2 \mathcal{S}_n = -\mathcal{A} *_N \Delta \mathcal{S}_n$ we have $\Delta^2 \mathcal{V}_k(n) = [\Delta^2 \mathcal{S}_n, \ldots, \Delta^2 \mathcal{S}_{n+k-1}] = -\mathcal{A} *_N \Delta \mathcal{V}_k(n)$ where $\Delta \mathcal{V}_k(n) = [\Delta \mathcal{S}_n, \ldots, \Delta \mathcal{S}_{n+k-1}]$. Consequently using (9.5) and (9.31), the generalized residual of the approximation $(\mathcal{T}_k^{(n)})$ is the true residual

$$\tilde{\mathcal{R}}(\mathcal{T}_k^{(n)}) = \mathcal{R}(\mathcal{T}_k^{(n)}) = \mathcal{B} - \mathcal{A} *_N \mathcal{T}_k^{(n)}.$$

For simplicity and unless specified otherwise, we set $n = 0, \mathcal{T}_k^{(0)} = \mathcal{T}_k, \mathcal{S}_0 = \mathcal{X}_0$ the initial guess and drop the index n in all our notations. When applied to the sequence generated by the linear relation (9.31), the TG-RRE, TG-MMPE and the TG-MPE above produce approximations $\mathcal{X}_k = \mathcal{T}_k$ such that the corresponding residual $\mathcal{R}_k = \mathcal{B} - \mathcal{A} *_N \mathcal{T}_k$ satisfies the relations

$$\mathcal{R}_k - \Delta \mathcal{S}_0 \in \tilde{\mathbf{H}}_k = -\mathcal{A} *_N \tilde{\mathbf{V}}_k$$
$$\mathcal{R}_k \in (\tilde{\mathbf{L}}_k)^{\perp}$$

where $\tilde{\mathbf{V}}_k = span\{\Delta \mathcal{S}_0, \ldots, \Delta \mathcal{S}_{k-1}\}$ and $\mathbf{L}_k \tilde{\equiv} \mathbf{H}_k$ for TG-RRE, $\mathbf{L}_k \equiv \tilde{\mathbf{V}}_k$ for TG-MPE and $\mathbf{L}_k \tilde{\equiv} \mathbf{Y}_k \tilde{=} span\{\mathcal{Y}_0, \ldots, \mathcal{Y}_{k-1}\}$ for TG-MMPE where $\mathcal{Y}_0, \ldots, \mathcal{Y}_{k-1}$ are some chosen tensors.

Notice that, since $\tilde{\mathbf{V}}_k = \mathcal{K}_m(\mathcal{A}, R_0)$ (the tensor Krylov subspace defined in [48, 77]), the extrapolation methods above are tensor global Krylov subspace methods. TG-RRE is an orthogonal projection and is theoretically equivalent to the tensor global GMRES while TG-MPE is oblique projection method and is equivalent to the tensor global Arnoldi method. □

As the TG-RRE is an orthogonal projection method, we also have the classical minimization property for the residual

$$||\mathcal{R}_{TG\text{-}RRE}|| = \min_{\mathcal{T} \in \mathcal{X}_0 + \mathcal{K}_m(\mathcal{A}, R_0)} ||\mathcal{B} - \mathcal{A} *_N \mathcal{T}_k||.$$

When the linear process (9.31) is convergent, it is more useful in practice to apply the tensor extrapolation methods after some fixed number of iterations. To save memory, we can also use the algorithm in a cycling mode which means that the iterations are restarted after a fixed number m of iterations. The algorithm is summarized as follows

Algorithm 37 TG-RRE, TG-MPE and TG-MMPE Algorithms

Step 1. $k = 0$ choose $\mathcal{X}_0$ and the numbers p and m.

Step 2. Basic iteration

Set $\mathcal{T}_0 = \mathcal{X}_0$

$\mathcal{Z}_0 = \mathcal{T}_0$

$\mathcal{Z}_{j+1} = (\mathcal{I} - \mathcal{A}) *_N \mathcal{Z}_j + \mathcal{B}, \qquad j = 0, \ldots, p-1$

Step 3. Extrapolation schema

$\mathcal{S}_0 = \mathcal{Z}_p$

$\mathcal{S}_{n+1} = (\mathcal{I} - \mathcal{A}) *_N \mathcal{S}_n + \mathcal{B}, \qquad n = 0, \ldots, m$

Compute the approximation $\mathcal{T}_{k+1}^{(0)}$ by TG-RRE, TG-MPE or TG-MMPE.

Set $\mathcal{X}_0 = \mathcal{T}_{k+1}^{(0)}$, $k = k + 1$ and go to step 2.

9.4.2 Application to non linear tensor systems

Consider the nonlinear system of tensor equations

$$\mathcal{G}(\mathcal{X}) = \mathcal{B} \tag{9.32}$$

with $\mathcal{G}(\mathcal{X})$ an operator from $\mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times J_1 \times J_2 \times \ldots \times J_M}$ onto $\mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times J_1 \times J_2 \times \ldots \times J_M}$, and $\mathcal{X}^* \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times J_1 \times J_2 \times \ldots \times J_M}$ the solution of the system of equation(9.32).

For any arbitrary tensor $\mathcal{X}$, the residual is given by

$$\mathcal{R}(\mathcal{X}) = \mathcal{G}(\mathcal{X}) - \mathcal{X} \tag{9.33}$$

Let $(\mathcal{S}_n)_n$ be the sequence of tensors generated from an initial guess $\mathcal{S}_0$ as follows

$$\mathcal{S}_{n+1} = \mathcal{G}(\mathcal{S}_n), \quad j = 0, 1 \ldots.$$

To get approximate solutions to a solution of (9.32), we apply the tensor extrapolation methods above to the sequence $(\mathcal{S}_n)$. As for the linear case, the different steps are summarized in the following algorithm.

Algorithm 38 TG-RRE, TG-MPE and TG-MMPE for nonlinear systems

Step 1. $k = 0$ choose $\mathcal{X}_0$ and the numbers p and m.

Step 2. Basic iteration

Set $\mathcal{T}_0 = \mathcal{X}_0$

$\mathcal{W}_0 = \mathcal{T}_0$

$\mathcal{W}_{j+1} = \mathcal{G}(\mathcal{W}_j) \quad j = 0, \ldots.., p-1$

Step 3. Extrapolation scheme

$\mathcal{S}_0 = \mathcal{W}_p$

If $||\mathcal{S}_1 - \mathcal{S}_0||_F < e$ stop

otherwise generate $\mathcal{S}_{n+1} = \mathcal{G}(\mathcal{S}_n), \quad n = 0, \ldots., m$

Compute the approximation $\mathcal{T}_k^{(0)}$ by TG-RRE, TG-MPE or TG-MMPE.

Set $\mathcal{X}_0 = \mathcal{T}_k^{(0)}$, $k = k + 1$ and go to step 2.

As we stated earlier, when applied to linearly generated tensor sequences, the proposed tensor extrapolation methods produce the same iterates as some well known tensor Krylov subspace methods but differ in the way that the approximations are computed. In the case where the process of generating the sequence $(\mathcal{S}_n)_n$ is not known and what is known is only the terms of this sequences, Krylov-based methods or Newton-type methods could not be used and in that case, extrapolation methods are well come. Such a problem can be found for example in some statistical-problems.

9.5 The global-QR implementation of TG-MPE/TG-RRE

The purpose of this section is to give an efficient implementation of TG-MPE and TG-RRE using a generalisation of the technique based on the QR decomposition and given in [125] for vector MPE and vector RRE methods. We first introduce the Tensor Global-QR decomposition. Let $\mathcal{U} \in \mathbb{R}^{I_1 \times I_2 \times \times \ldots I_M \times J_1 \times J_2 \times \ldots \times J_N \times k}$, be an $(M + N + 1)$-mode tensor with the column tensors $\mathcal{U}_0, \ldots, \mathcal{U}_{k-1} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M \times J_1 \times J_2 \times \ldots \times J_N}$. Then, there is an $(M + N + 1)$-mode orthogonal tensor $\mathcal{Q} = [\mathcal{Q}_0, \ldots, \mathcal{Q}_{k-1}] \in \mathbb{R}^{I_1 \times I_2 \times \times \ldots I_M \times J_1 \times J_2 \times \ldots \times J_N \times k}$ satisfying $\mathcal{Q} \, \copyright^{(N+M+1)} \, \mathcal{Q} = I_{k \times k}$ and an upper triangular matrix $R \in \mathbb{R}^{k \times k}$ such that

$$\mathcal{U} = \mathcal{Q} \times_{(M+N+1)} R \tag{9.34}$$

The tensor decomposition (9.34) will be called the Tensor Global-QR (TG-QR) decomposition of $\mathcal{A}$ and is summarized in the following algorithm.

Algorithm 39 The global-QR decomposition

1. Given $\mathcal{U}_0$, and compute the scalar r_{00} and the tensor $\mathcal{Q}_0$ by $r_{0,0} = \langle \mathcal{U}_0, \mathcal{U}_0 \rangle_2^{\frac{1}{2}}$ and $\mathcal{Q}_0 = \frac{\mathcal{U}_0}{r_{00}}$.
2. For $j = 1, \ldots, k$
   (a) $\mathcal{W} = \mathcal{U}_j$
   (b) For $i = 0, \ldots, j-1$ do
      i. $r_{i,j} = \langle \mathcal{Q}_i, \mathcal{W} \rangle$
      ii. $\mathcal{W} = \mathcal{W} - r_{i,j}\mathcal{Q}_i$
      iii. End for
   (c) $r_{j,j} = \langle \mathcal{W}, \mathcal{W} \rangle_2^{\frac{1}{2}}$
   (d) $\mathcal{Q}_j = \frac{\mathcal{W}}{r_{j,j}}$
   (e) End for
3. End

To show that the tensor $\mathcal{Q}$ is orthogonal, we just proceed by induction on k. The coefficients of the matrix R are the $r_{i,j}$'s given by Algorithm 39. Next, we give a proposition to be used.

Proposition 9.5. *[91] Let* $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$, $A \in \mathbb{R}^{J_n \times I_n}$ *and* $y \in \mathbb{R}^{J_n}$, *then we have*

$$\mathcal{X} \times_{(n)} A\tilde{\times}_{(n)} y = \mathcal{X} \tilde{\times}_{(n)} (A^T y) .$$

For simplicity, we set here $n = 0$ and then the approximation $T_k^{(0)}$ (defined earlier in 8.13) is given by

$$T_k^{(0)} = \mathcal{S}_0 + \sum_{j=0}^{k-1} \delta_j^{(k)} \Delta\mathcal{S}_j = \mathcal{S}_0 + \Delta\mathcal{V}_k(\,)^0 \tilde{\times}_{(N+M+1)} \delta^{(k)}$$

Substituting now $\mathcal{V}_k = \mathcal{V}_k(0) = \mathcal{Q} \times_{(N+M+1)} R$ and using Proposition (9.5), we get

$$\begin{aligned} T_k^{(0)} &= \mathcal{S}_0 + \Delta\mathcal{V}_k \tilde{\times}_{(M+N+1)} \delta^{(k)} \\ &= \mathcal{S}_0 + \mathcal{Q} \tilde{\times}_{(M+N+1)} (R^T \delta^{(k)}), \end{aligned}$$

which gives

$$T_k^{(0)} = \mathcal{S}_0 + \sum_{j=0}^{k-1} \theta_j \mathcal{Q}_j, \tag{9.35}$$

where θ_j is the $(j+1)$ component of the column vector $R^T\delta^{(k)}$; the matrix R and the tensors $\mathcal{Q}_j \in \mathrm{R}^{I_1 \times \ldots I_M \times J_1 \times \ldots \times J_N}$, $j = 0, \ldots, k-1$ are given by Algorithm 39. To compute $\delta^{(k)}$, we have first to compute $\beta^{(k)} = (\beta_0^{(k)}, \ldots, \beta_{k-1}^{(k)})^T$ and use the relations given in (9.21).

For TG-MPE, the coefficients $\gamma_i^{(k)}$'s of the vector $\gamma^{(k)}$ are determined by computing the vector $\beta^{(k)}$, the solution of the linear system of equations (8.9)

$$(\Delta\mathcal{V}_k \, \mathbb{O}^{(N+M+1)} \, \Delta\mathcal{V}_k) \, \beta^{(k)} = -(\Delta\mathcal{V}_k \, \mathbb{O}^{(N+M+1)} \, \Delta\mathcal{S}_k) \tag{9.36}$$

where

$$(\Delta\mathcal{V}_k \, \mathbb{O}^{(N+M+1)} \, \Delta\mathcal{V}_k) = \begin{bmatrix} \langle \Delta\mathcal{S}_0, \Delta\mathcal{S}_0 \rangle & \ldots & \langle \Delta\mathcal{S}_0, \Delta\mathcal{S}_{k-1} \rangle \\ \langle \Delta\mathcal{S}_1, \Delta\mathcal{S}_0 \rangle & \ldots & \langle \Delta\mathcal{S}_1, \Delta\mathcal{S}_{k-1} \rangle \\ \vdots & \ldots & \vdots \\ \langle \Delta\mathcal{S}_{k-1}, \Delta\mathcal{S}_0 \rangle & \ldots & \langle \Delta\mathcal{S}_{k-1}, \Delta\mathcal{S}_{k-1} \rangle \end{bmatrix} \in \mathrm{R}^{k \times k} \tag{9.37}$$

and $(\Delta \mathcal{V}_k \circledcirc^{(N+M+1)} \Delta \mathcal{S}_k) = (\langle \Delta S_0, \Delta S_k \rangle, \ldots, \langle \Delta S_{k-1}, \Delta S_k \rangle)^T \in \mathbb{R}^k$. For TG-RRE, $\beta^{(k)}$ are determined by solving the linear system of equation

$$(\Delta^2 \mathcal{V}_k \circledcirc^{(N+M+1)} \Delta \mathcal{V}_k)\beta^{(k)} = -(\Delta^2 \mathcal{V}_k \circledcirc^{(N+M+1)} \Delta \mathcal{S}_k)$$

where

$$(\Delta^2 \mathcal{V}_k \circledcirc^{(N+M+1)} \Delta \mathcal{V})_k = \begin{bmatrix} \langle \Delta^2 S_0, \Delta S_0 \rangle & \cdots & \langle \Delta^2 S_0, \Delta S_{k-1} \rangle \\ \langle \Delta^2 S_1, \Delta S_0 \rangle & \cdots & \langle \Delta^2 S_1, \Delta S_{k-1} \rangle \\ \vdots & \cdots & \vdots \\ \langle \Delta^2 S_{k-1}, \Delta S_0 \rangle & \cdots & \langle \Delta^2 S_{k-1}, \Delta S_{k-1} \rangle \end{bmatrix} \in \mathbb{R}^{k \times k}, \tag{9.38}$$

and $(\Delta^2 \mathcal{V}_k \circledcirc^{(N+M+1)} \Delta \mathcal{S}_k) = (\langle \Delta^2 S_0, \Delta S_k \rangle, \ldots, \langle \Delta^2 S_{k-1}, \Delta S_k \rangle)^T \in \mathbb{R}^k$. Finally, once $\beta^{(k)}$ is computed, the coefficients $\gamma^{(k)}$'s are given by

$$\gamma_l^{(k)} = \frac{\beta_l^{(k)}}{\sum_{l=0}^{k} \beta_l^{(k)}} \quad \text{for} \quad 0 \le l < k \quad \text{and} \quad \beta_k^{(k)} = 1. \tag{9.39}$$

The following algorithm summarizes the main steps.

Algorithm 40 Implementation of TG-MPE/TG-RRE via the global-QR decomposition

1. $\mathcal{S}_0$ is a given initial guess and k is a fixed index.
2. Apply Algorithm 39 to compute the global-QR decomposition of the tensor $\Delta \mathcal{V}_k$.
3. Compute $\beta^{(k)}$ by solving the linear system (9.37) or (9.38).
4. Compute the coefficients $\gamma_j^{(k)}$ from (9.39).
5. Compute $\delta^{(k)}$ from the relation (9.21) and θ_j is $(j+1)$ component of the column vector $R^T \delta^{(k)}$.
6. Compute $T_k^{(0)}$ by $T_k^{(0)} = \mathcal{S}_0 + \sum_{j=0}^{k-1} \theta_j \mathcal{Q}_j$

As a numerical test, we consider the linear system of tensor equations given by (9.30) where $\mathcal{A}$ is a random tensor $N = 3$, $I_1 = I_2 = 20$, $I_3 = 10$, $J_1 = 20$, $J_2 = 10$ and $J_3 = 5$ as in [77]. The exact solution is the tensor $\mathcal{X}^*$ whose elements are all one and the right hand side $\mathcal{B}$ is given by $\mathcal{B} = \mathcal{A} *_N \mathcal{X}^*$. The computations are carried out using MATLAB 7.4 with machine epsilon about $2 \cdot 10^{-16}$. We took $m = 10$, $p = 1$ and stppoed the iteration when the relative error norm of the residual was less than 10^{-6}. Then after $k = 6$ iterations, Algorithm 37 for the tensor RRE method, gives the relative residual norm $\frac{\mathcal{R}_k}{\mathcal{R}_0} = 8.5 \times 10^{-7}$ where the initial guess is $\mathcal{X}_0 = \mathcal{O}$.

9.6 Conclusion

In this chapter we introduced new tensor global extrapolation methods to accelerate the convergence of some tensor sequences. The proposed methods are generalisations to the tensor case of some well known vector extrapolation methods such as the reduced rank extrapolation or the topological epsilon algorithm. The new methods were defined as orthogonal or oblique projection processes using the Einstein product and also some new interesting tensor products. We showed how to apply the derived algorithms to tensor linear and nonlinear systems of tensor equations. Application to some interesting problems such as the multilinear page rank is still under investigation.

Chapter 10

Conclusions and perspectives

The work done during my thesis, allowed me to discover a new and interesting field related to numerical analysis, it is about "**tensors**". After a general introduction evoking the problems related to this research axis, we introduce the tubal Krylov subspace methods via the well know T-product [89, 90], which can be considered as the generalization of the matrix Krylov subspace methods. Then, in chapters 4, 5 and 6, we have described the methods of tonsorial Krylov subspaces for the processing of color images and videos. Indeed, we have proposed three different approaches that aim essentially the exploitation of these methods for the restoration of images and videos. The first approach is based on the Einstein product[46], the second uses the T-product [89, 90] and the third applies the c-product [88]. In addition, we have proposed in chapter 7 tensor Krylov subspace methods for solving Sylvester equations. In particular, we have introduced the global and block tensor version of this method. Finally, in chapters 8 and 9, the tensor version of the extrapolation methods. In particular, we have developed two versions of these methods. The first one is based on the Einstein product and the second one uses the T-product.

This area of research has more potential to grow. We briefly quote some research axes that can be developed using the tensor formalism :

* Tensor decomposition and Images and vidéos completion.

- Machine learning and artificial intelligence.
- Preconditioned iterative methods.
- Resolution of tensor Riccati Equations and numerical methods for solving large scale dynamical systems.
- Tensor network analysis and numerical methods for TV regularization.

Finally, we can say that the use of matrices is no longer useful. **Tensors are the future**.

Bibliography

[1] O. Abidi, M. Hached, and K. Jbilou, A global rational Arnoldi method for model reduction, J. Comput. Appl. Math., 325 (2017), pp. 175–187.

[2] O. Abidi, M. Heyouni, and K. Jbilou, On some properties of the extended block and global Arnoldi methods with applications to model reduction, Numer. Algo., 75 (2017), pp. 285–304.

[3] H. Alqahtani, and L. Reichel, Simplified anti-Gauss quadrature rules with applications in linear algebra, Numer. Algo., 77 (2018), pp. 577–602.

[4] H. Alqahtani and L. Reichel, Generalized block anti-Gauss quadrature rules, Numer. Math., 143 (2019), pp. 605–648.

[5] R. Ballester-Ripoll, S. K. Suter, and R. Pajarola, "Analysis of tensor approximation for compression-domain volume visualization," Computers & Graphics, 47 (2015), pp. 34–47.

[6] Z. Bai, M. Fahey, and G. Golub, Some large-scale matrix computation problems, J. Comput. Appl. Math., 74 (1996), pp. 71–89.

[7] H. Bakouki, A.H. Bentbib, M. Heyouni, and K. Jbilou, An extended nonsymmetric block Lanczos method for model reduction in large scale dynamical systems, Calcolo, 55 (2018), pp. 13–36 .

[8] R.H. Bartels, G.W. Stewart, Algorithm 432: solution of the matrix equation AX + XB = C, Circ. Syst. Signal Proc., 13 (1994), pp. 820–826.

[9] A. Bavelas, A mathematical model for groupst ructure, Human Organization, 7 (1948), pp. 16–30.

[10] J. Baglama, D. Calvetti, and L. Reichel, Algorithm 827: irbleigs: A MATLAB program for computing a few eigenpairs of a large sparse Hermitian matrix, ACM Trans. Math. Software, 29 (2003), pp. 337–348.

[11] J. Baglama, D. Calvetti, and L. Reichel, IRBL: An implicitly restarted block-Lanczos method for large-scale Hermitian eigenproblems, SIAM J. Sci. Comput., 24 (2003), pp. 1650–1677.

[12] M. Bellalij, L. Reichel, G. Rodriguez, and H. Sadok, Bounding matrix functionals via partial global block Lanczos decomposition, Appl. Numer. Math., 94 (2015), pp. 127–139.

[13] F. P. A. Beik, K. Jbilou, M. Najafi-Kalyani and L. Reichel, Golub–Kahan bidiagonalization for ill-conditioned tensor equations with applications. Numerical Algorithms, 84 (2020), pp. 1535–1563.

[14] F. P. A Beik, M. Najafi-Kalyani and L. Reichel , Iterative Tikhonov regularization of tensor equations based on the Arnoldi process and some of its generalizations, Appl. Numer. Math., 151(2020), pp. 1425–447.

[15] F. P. A. Beik, A. El Ichi , K. Jbilou and R. Sadaka, Tensor extrapolation methods with applications, Numerical Algorithms, Numerical Algorithms, 87 (2021), pp. 1421–1444.

[16] F. P. A. Beik, M. El Guide, A. El Ichi, and K. Jbilou,Tensor GMRES and Golub-Kahan Bidiagonalization methods via the Einstein product with applications to image and video processing. (submitted).

[17] A.H. Bentbib, M. El Guide, K. Jbilou and L. Reichel, Global Golub–Kahan bidiagonalization applied to large discrete ill-posed problems, Journal of Computational and Applied Mathematics, 322 (2017), pp. 46–56.

[18] A.H. Bentbib, M. El Guide, K. Jbilou, E. Onunwor and L. Reichel, Solution methods for linear discrete ill-posed problems for color image restoration, BIT Numerical Mathematics, 58 (3)(2018), pp. 555—576.

[19] A.H. Bentbib, K. Jbilou and El M. Sadek, On some extended block

Krylov based methods for large scale nonsymmetric Stein matrix equations, Mathematics, 5 (2017), 21.

[20] C. Brezinski, P. Fika, M. Mitrouli, Moments of a linear operator on a Hilbert space, with applications to the trace of the inverse of matrices and the solution of equations, Numer. Linear Algebra Appl., 19 (2012), pp. 937–953.

[21] P. Bonacich, Factoring and weighting approaches to status scores and clique identification, Journal of Mathematical Sociology, 2 (1972), pp. 113–120.

[22] P. Bonacich, Power and centrality: a family of measures. American Journal of Socio., 92 (1987), pp. 1170–1182.

[23] A. Bouhamidi, K. Jbilou, L. Reichel, and H. Sadok, A generalized global Arnoldi method for ill posed matrix equations, J. Comput. Appl. Math., 236 (2012), pp. 2078–2089.

[24] R. Bouyouli, K. Jbilou, R. Sadaka, and H. Sadok, Convergence properties of some block Krylov subspace methods for multiple linear systems, J. Comput. Appl. Math., 196 (2006), pp. 498–511.

[25] C. Brezinski, Généralisation de la transformation de Shanks, de la table de Padéé et l'epsilon algorithm. Calcolo, (1975), pp. 317–360 .

[26] K. Braman, Third-order tensors as linear operators on a space of matrices, Lin. Alg. Appl., 433 (2010), pp. 1241–1253.

[27] M. Brazell, N. Li. C. Navasca and C. Tamon, Solving Multilinear Systems Via Tensor Inversion SIAM J. Matrix Anal. Appl., 34 (2013), pp. 542–570.

[28] C. Brezinski, M. Redivo Zaglia, Extrapolation Methods. Theory and Practice. North-Holland, Amsterdam (1991).

[29] AR. Bro, "Multi-way analysis in the food industry: models, algorithms, and applications," Ph.D. dissertation, Royal Veterinary and Agricultural University, Denmark, (1998).

[30] AR. Bro, “PARAFAC. tutorial and applications,” Chemometrics and intelligent laboratory systems, vol. 38, no. 2 (1997), pp. 149–171.

[31] P. M. S. Burt and J. H. de Morais Goulart, “Evaluating the potential of VolterraPARAFAC IIR models,” in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), (2013), pp. 5745–5749.

[32] O. De la Cruz Cabrera, M. Matar, and L. Reichel, Analysis of directed networks via the matrix exponential, J. Comput. Appl. Math., 355 (2019), pp. 182–192.

[33] S. Cabay, , L. W. Jackson, A polynomial extrapolation method for finding limits and antilimits for vector sequences. SIAM J. Numer. Anal., 13 (1976), pp. 734–752,

[34] G. Caldarelli, Scale-Free Networks, Oxford University Press, Oxford, (2007).

[35] D. Calvetti, L. Reichel, and F. Sgallari, Application of anti-Gauss quadrature rules in linear algebra, in Applications and Computation of Orthogonal Polynomials, W. Gautschi, G. H. Golub, and G. Opfer, eds., Birkhauser, Basel, (1999), pp. 41–56.

[36] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, “Tensor decompositions for signal processing applications: From two-way to multiway component analysis,” IEEE Signal Processing Magazine, vol. 32, no. 2 (2015), pp. 145–163.

[37] M. Crouzeix, and C. Palencia, The numerical range is a $1 + \sqrt{2}$ -spectral set., SIAM J. Matrix Anal. Appl., 38 (2017), pp. 649–655.

[38] B. Datta, M. Heyouni, and K. Jbilou, The global Arnoldi process for solving the Sylvester- Observer equation., J. Comput. Appl. Math., 29 (2010), pp. 527–544.

[39] T. Davis and Y. Hu, The SuiteSparse Matrix Collection, https://sparse.tamu.edu.

[40] J. Dauwels, K. Srinivasan, M. Ramasubba Reddy, and A. Cichocki, "Multi-channel eeg compression based on matrix and tensor decompositions," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), (2011), pp. 629–632.

[41] R. Diestel. Graph Theory. Springer, Berlin, (2000).

[42] V. Druskin and L. Knizhnerman, Extended Krylov subspaces: Approximation of the matrix square root and related functions, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 755–771.

[43] V. Druskin, and L. Knizhnerman, Gaussian spectral rules for the three-point second differences: I. A two-point positive definite problem in a semi-infinite domain, SIAM J. on Num. Anal., 37 (1998), pp. 403–422.

[44] F. Dufrenois, A. El Ichi, M. K. Jbilou and M. Hached, Cosine multidimensional linear discrete analysis methods for face recognition in large data, (In preparation).

[45] V. P. Eddy., Extrapolation to the limit of a vecteur sequence. In: P. C. C. Wang. (ed.), Information Linkage Between Applied Mathematics and Industry, pp. 387–396. Academic Press, New-York (1979).

[46] A. Einstein, The foundation of the general theory of relativity. In: Kox AJ, Klein MJ, Schulmann R, editors. The collected papers of Albert Einstein. Vol. 6, Princeton (NJ): Princeton University Press; 2007, pp. 146–200.

[47] A. El Guennouni, K. Jbilou, A.J. Riquet, Block Krylov subspace methods for solving large Sylvester equations, Numer. Algebra, 29 (2002), pp. 75–96.

[48] M. Elguide, A. El Ichi, K. Jbilou, F.P.A Beik, Tensor GMRES and Golub-Kahan Bidiagonalization methods via the Einstein product with applications to image and video processing, **2020**, arXiv preprint arXiv:2005.07458.

[49] M. El Guide, A. El Ichi, and K. Jbilou, Discrete cosine transform LSQR

methods for multidimensional ill-posed problems J. Math. Model., (2021).

[50] M. El Guide, A. El Ichi, K. Jbilou and R. Sadaka, On tensor GMRES and Golub-Kahan methods via the T-product for color image processing, The Electronic Journal of Linear Algebra, 37(2021), pp. 524–543.

[51] A. El Ichi , K. Jbilou and R. Sadaka, On tensor tubal-Krylov subspace methods, Linear and Multilinear Algebra, (Accepted).

[52] A. El Ichi , K. Jbilou and R. Sadaka, Tensor Global Extrapolation Methods Using the n-Mode and the Einstein Products, Mathematics, 8(8) (2020), 1298.

[53] A. El Ichi, Tensor Krylov subspace methods via the T-product for large Sylvester tensor equations, (submitted).

[54] G. Favier, A. Y. Kibangou, and T. Bouilloc, "Nonlinear system modeling and identification using Volterra-PARAFAC models," International Journal of Adaptive Control and Signal Processing, vol. 26, no. 1 (2012), pp. 30–53.

[55] C. Fenu, D. Martin, L. Reichel, and G. Rodriguez, Block Gauss and anti-Gauss quadrature with application to networks, SIAM J. Mat. Anal. Appl., 34 (2013), pp. 1655–1684.

[56] L. Freeman, A set of measures of centrality based on betweenness, Socio., 40 (1977), pp. 35–41.

[57] L. Freeman, Centrality in networks. I: Conceptual clarification, Soc. Net., 1 (1979), pp. 215–239.

[58] W. Gautschi, Orthogonal Polynomials: Computation and Approximation, Oxford University Press, Oxford, 2004.

[59] W. Gautschi,The interplay between classical analysis and (numerical) linear algebra—a tribute to Gene H. Golub, Electron. Trans. Numer. Anal., 13 (2002), pp. 119–147.

[60] G. H. Golub, M. Heath, G. Wahba, Generalized cross-validation as a method for choosing a good ridge parameter, , Technometrics 21(1979), pp. 215–223.

[61] G.H. Golub, W. Kahan, Algorithm LSQR is based on the Lanczos process and bidiagonalization procedure, SIAM J. Numer. Anal. 2 (1965), pp. 205–224.

[62] G. H. Golub and C. F. Van Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore, 1996.

[63] G.H. Golub, S. Nash, C. Van Loan, A Hessenberg–Schur method for the problem AX + XB = C, IEEC Trans. Automat. Contr. AC24 (1979), pp. 909–913.

[64] G. H. Golub, and G. Meurant, Matrices, Moments and Quadrature with Applications, Princeton University Press, Princeton, 2010.

[65] G. H. Golub, and G. Meurant, Matrices, moments and quadrature, in: D.F. Griffiths, G.A. Watson (Eds.), Numerical Analysis 1993, Longman, Essex, England, 1994, pp.105–156.

[66] G. H. Golub, and C. F. Van Loan, Matrix Computations fourth edition, Johns Hopkins University Press, Baltimore, 2013.

[67] P. C. Hansen Analysis of discrete ill-posed problems by means of the L-curve, SIAM Rev., 34(1992), pp. 561–580.

[68] P. C. Hansen, J. Nagy, and D. P. O'Leary, Deblurring Images: Matrices, Spectra, and Filtering, SIAM, Philadelphia, 2006.

[69] P. C. Hansen, Regularization tools version 4.0 for MATLAB 7.3, Numer. Algorithms, 46 (2007), pp. 189–194.

[70] R. A. Harshman, "Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multi-modal factor analysis," UCLA Working Papers in Phonetics, vol. 16 (1970), pp. 1–84, .

[71] F. L. Hitchkock, "The expression of a tensor or a polyadic as a sum of products," Journal of Mathematics and Physics, vol. 6, no. 1–4 (1927),

pp. 164–189.

[72] M. Heyouni, Extended Arnoldi methods for large low-rank Sylvester matrix equations, Appl. Numer. Math., 60 (2010), pp.1171–1182.

[73] M. Heyouni and K. Jbilou, An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equation, Electron. Trans. Numer. Math., 33 (2009), pp. 53–62.

[74] N. J. Higham, Functions of matrices: theory and computation. SIAM, Philadelphia, (2008).

[75] D.Y. Hu, L. Reichel, Krylov subspace methods for the Sylvester equation, Linear Algebra Appl. 174 (1992), pp. 283–314.

[76] B. Huang, Y. Xie and C. Ma, Krylov subspace methods to solve a class of tensor equations via the Einstein product. Num. Lin. Alg. Appl., 26 (2019), e2254.

[77] B.Huang , Y.Xie , C.Ma . Krylov subspace methods to solve a class of tensor equations via the Einstein product, Numer. Lin. Alg. Appl., (2019).

[78] C. Jagels, K. Jbilou, and L. Reichel, The extended global Lanczos method, Gauss-Radau quadrature, and matrix function approximation, J. Comput. Appl. Math., 381 (2021) Art. 113027.

[79] K. Jbilou, H. Sadok, Analysis of some vector extrapolation methods for linear systems. Numer. Math., 70 (1995), pp. 73–89 .

[80] K. Jbilou and A. Messaoudi, Block extrapolation methods with applications, Applied Numerical Mathematics, 106 (2016), pp. 154–164.

[81] K. Jbilou, A. Messaoudi, and H. Sadok, Global FOM and GMRES algorithms for matrix equations, Appl. Numer. Math, 31 (1999), pp. 49–63.

[82] K. Jbilou, Low rank approximate solutions to large Sylvester matrix equations, Appl. Math. Comp., 177 (2006), 365–376.

[83] K. Jbilou and H. Sadok, LU-implementation of the modified minimal polynomial extrapolation method. IMA J. Numer. Anal., 19 (1999), pp.

549–561.

[84] K. Jbilou, H. Sadok, Matrix polynomial and epsilon-type extrapolation methods with applications. Numer Algo., 68 (2015), pp. 107–119.

[85] K. Jbilou, H. Sadok, and A. Tinzeft, Oblique projection methods for multiple linear systems. Elect. Trans. Num. Anal., 20 (2005), pp. 119–138.

[86] K. Jbilou, A. Messaoudi, K. Tabaa, : Some Schur complement identities to matrix extrapolation methods. Lin. Alg. Appl., 392 (2004), pp. 195–210 .

[87] K. Jbilou, L. Reichel and H. Sadok, Vector extrapolation enhanced TSVD for linear discrete ill-posed problems, Numer. Algorithms, 51 (2009), pp. 195–208.

[88] E. Kernfeld, M. Kilmer, and S. Aeron, Tensor-tensor products with invertible linear trans- forms, Linear Algebra Appl., 485 (2015), pp.545–570.

[89] Kilmer M, Braman K, Hao N, et al. Third-Order Tensors as Operators on Matrices: A Theoretical and Computational Framework with Applications in Imaging. SIAM Journal on Matrix Analysis Appl., (34) 2013, pp.148–172.

[90] Kilmer M, Martin C. Factorization strategies for third-order tensors. Linear Algebra Appl., 435 (2011), pp. 641–658.

[91] T. G. Kolda and J. Sun, "Scalable tensor decompositions for multi-aspect data mining," in Proceedins of the 8th IEEE international conference on data mining, (2008), pp. 363–372.

[92] L. A. Krukier. Implicit difference schemes and an iterative method for solving them for a certain class of systems of quasi-linear equations., Sov. Math., 23 (1979), pp. 43–55.

[93] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Nat. Bur. Stand.,

45 (1950), pp. 255–282.

[94] D. P. Laurie, Anti-Gaussian quadrature formulas, Math. Comp., 65 (1996), pp. 735–747.

[95] HL. De Lathauwer, "Signal processing based on multilinear algebra," Ph.D. dissertation, (1997).

[96] S. Leurgans and R. T. Ross, "Multilinear models: applications in spectroscopy," Statistical Science, (1992) pp. 289–310,

[97] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 1 (2013), pp. 208–220 .

[98] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," in Proceedings of the IEEE International Conference on Computer Vision, Sept (2009), pp. 2114–2121.

[99] F. Li, M. K. Ng and R. J. Plemmons, Coupled segmentation and denoising/deblurring for hyperspectral material identification, Numer. Lin. Alg. Appl., 19 (2012), pp. 153–173.

[100] M. Najafi-Kalyani,F.P. A Beik, K. Jbilou, On global iterative schemes based on Hessenberg process for (ill-posed) Sylvester tensor equations, Journal of Computational and Applied Mathematics,373 (2020), pp. 112–216.

[101] N. Li, C. Navasca and C. Glenn, Iterative methods for symmetric outer product tensor decomposition, Electron. Trans. Numer. Anal., 44 (2015), pp. 124–139.

[102] K. Lund, The tensor t-function: a definition for functions of third-order tensors, Num. Lin. Alg. Appl., 27 (2020) e2288.

[103] A. G. Mahyari, D. Zoltowski, E. Bernat, and S. Aviyente, "A tensor decomposition based approach for detecting dynamic network states from EEG," IEEE Transactions on Biomedical Engineering, (2016).

[104] M. Mesina, Convergence acceleration for the iterative solution of $= Ax + f$. Comput. Meth. Appl. Mech. Eng., 10(2) (1977), pp.165–173.

[105]J. L. Morrison, R. Breitling, D. J. Higham, and D. R. Gilbert, A lock-and-key model for protein-protein interactions, Bioinformatics, 2 (2006), pp. 2012–2019.

[106]M. Morup, "Applications of tensor (multiway array) factorizations and decompositions in data mining," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 1, no. 1 (2011), pp. 24–40 .

[107]Y. Miao, L. Qi and Y. Wei, Generalized Tensor Function via the Tensor Singular Value Decomposition based on the T-Product, Lin. Alg. Appl., 590 (2020), pp. 258–303.

[108]M.E.J. Newman, Networks: An Introduction. Oxford University Press, Oxford, (2010).

[109]M. Najafi-Kalyani, F.P.A Beik,K. Jbilou, On global iterative schemes based on Hessenberg process for (ill-posed) Sylvester tensor equations, Journal of Computational and Applied Mathematics, 373 (2020), pp. 112216.

[110]T. T. Ngo, M. Bellalij, and Y. Saad, The trace ratio optimization problem, SIAM Rev., 54 (2012), pp. 545–569.

[111]M. K. Ng, R. H. Chan, W. Tang, A fast algorithm for deblurring models with Neumann boundary conditions, SIAM Journal on Scientific Computing, 21 (1999), pp. 851–866.

[112]A. Ozerov, C. Févotte, R. Blouet, and J.-L. Durrieu, "Multichannel nonnegative tensor factorization with structured constraints for user-guided audio source separation," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), (2011), pp. 257–260

[113] E. E. Papalexakis, T. M. Mitchell, N. D. Sidiropoulos, C. Faloutsos, P. P.

Talukdar, and B. Murphy, "Turbo-SMT: Parallel coupled sparse matrix-tensor factorizations and applications," Statistical Analysis and Data Mining: The ASA Data Science Journal, vol. 9, no. 4 (2016), pp. 269–290.

[114]S. Pozza, M. S. Pranić, and Z. Strakoš, The Lanczos algorithm and complex Gauss quadrature, Electron. Trans. Numer. Anal., 50 (2018), pp. 1–19.

[115]B. P. Pugatchev, Acceleration of the convergence of iterative processes and a method for solving systems of nonlinear equations, U.S.S.R. Comput. Math. and Math. Phys., 17 (1978), pp. 199–207 .

[116]L. N. Ribeiro, A. L. De Almeida, and V. Zarzoso, "Enhanced block term decomposition for atrial activity extraction in atrial fibrillation ECG," in Proceedings of the 9th IEEE Sensor Array and Multichannel Signal Processing Workshop, (2016).

[117]S. F. Roohi, D. Zonoobi, A. A. Kassim, and J. L. Jaremko, "Dynamic MRI reconstruction using low rank plus sparse tensor decomposition," in Proceedings of the IEEE International Conference on Image Process- ing (ICIP), (2016), pp. 1769–1773.

[118]L. Reichel, G. Rodriguez, and S. Seatzu, Error estimates for large-scale ill-posed problems, Numer. Algorithms, 51 (2009), pp. 341–361.

[119]L. Reichel, G. Rodriguez, and T. Tang, New block quadrature rules for the approximation of matrix functions, Lin. Alg. Appl., 502 (2016), pp. 299–326.

[120]Y. Saad, Numerical solution of large Lyapunov equations, in: M.A. Kaashoek, J.H. van Schuppen, A.C. Ran (Eds.), Proceedings of the International Symposium MTNS-89, Boston, Signal Processing, Scattering, Operator Theory and Numerical Methods, vol. 3, Birkhauser, (1990), pp. 503–511.

[121]Y. Saad, J. Chelikowsky, and S. Shontz, Numerical methods for electronic structure calculations of materials, SIAM Rev., 52 (2010), pp.3–54.

[122]S. Sahnoun and P. Comon, "Joint source estimation and localization," IEEE Transactions on Signal Processing, vol. 63, no. 10, (2015) pp. 2485–2495 .

[123]A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in Proceedings of the 22nd international conference on Machine learning. ACM, (2005), pp. 792–799.

[124]A. Sidi, Convergence and stability of minimal polynomial and reduced rank extrapolation algorithms, SIAM J. Numer. Anal., 23 (1986), pp. 197–209.

[125]A. Sidi, Efficient implementation of minimal polynomial and reduced rank extrapolation methods, Journal of Computational and Applied Mathematics, (1991), pp.305–337 .

[126]A. Sidi, W.F. Ford and D.A. Smith:Acceleration of convergence of vector sequences , SIAM J. Numer. Anal., 23(1986), pp. 178–196 .

[127]V. Simoncini, A new iterative method for solving large-scale Lyapunov matrix equations, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288.

[128]V. Simoncini, Extended Krylov subspace for parameter dependent systems, Appl. Numer. Math., 60 (2010), pp. 550–560.

[129]N. D. Sidiropoulos, R. Bro, and G. B. Giannakis, "Parallel factor analysis in sensor array processing," IEEE transactions on Signal Processing, vol. 48, no. 8 (2000), pp. 2377–2388.

[130]A.N. Tikhonov, Regularization of incorrectly posed problems, Soviet Math., 4 (1963), pp. 1624–1627.

[131]JL. R. Tucker, "Implications of factor analysis of three-way matrices for measurement of change," in Problems in measuring change., C. W. Harris, Ed. Madison WI: University of Wisconsin Press, (1963), pp. 122—137.

[132] JL. R. Tucker, "The extension of factor analysis to three-dimensional

matrices," in Contributions to mathematical psychology., H. Gulliksen and N. Frederiksen, Eds. New York: Holt, Rinehart and Winston, (1964), pp. 110–127.

[133]JL. R. Tucker, "Some mathematical notes on three-mode factor analysis," Psychometrika, vol. 31, no. 3 (1966), pp. 279–311.

[134]EE. Tyrtyshnikov, Mosaic-skeleton approximations, Calcolo, 33 (1996), pp. 47–57.

[135] R. S. Varga, Geršgorin and His Circles, Springer, Berlin, (2004).

[136]M. A. O. Vasilescu and D. Terzopoulos, Multilinear analysis of image ensembles: TensorFaces, in ECCV 2002: Proceedings of the 7th European Conference on Computer Vision, Lecture Notes in Comput. Sci. 2350, Springer, (2002), pp. 447-460.

[137]M. A. O. Vasilescu and D. Terzopoulos, Multilinear image analysis for facial recognition, in ICPR 2002: Proceedings of the 16th International Conference on Pattern Recognition, (2002), pp. 511-514.

[138]G. Wahba, Pratical approximation solutions to linear operator equations when the data are noisy, SIAM J. Numer. Anal., 14 (1977), pp. 651–667.

Résumé

Dans ces dernières années, plusieurs axes de recherches liées aux tenseurs (matrices multidimensionnelles) ont été exploités, optimisation non linéaire, algèbre multilinéaire, analyse non linéaire, Statistiques, langage de programmation.....Vu que dans la modélisation de plusieurs phénomènes, la représentation tensorielle à prouver efficacité face à celle matricielle. En effet, les données réelles traitées à partir des différents problèmes entraînants souvent des demandes écrasantes sur les ressources informatiques (stockage, coût de calcul.....). Convaincus par l'efficacité de l'approche tensorielle, nous proposons dans cette thèse, une étude qui cible la généralisation des méthodes numériques matricielles via le formalisme tensoriel. En particulier, nous nous sommes concentrés sur l'extension des méthodes des sous espaces de Krylov et leurs applications. Plus précisément, nous avons introduit les méthodes de sous espace de Krylov tensorielles, type global et par blocs liés aux différents produits tensoriels existant déjà. Nous avons même réussi à développer une nouvelle catégorie de ces méthodes nommée méthodes des sous espaces de Krylov tubales. De plus, nous avons proposé des applications de ces méthodes dans la résolution des systèmes tensoriels linéaires et non linéaires, la restauration des images et vidéos, ainsi que la résolution des équations tensorielles de Sylvester. Nous traitons aussi les méthodes liées aux machines Learning et la reconnaissance faciale.

Mots-clefs (5) : Tenseurs, sous espace de Krylov, produit d'Einstein, produit à n mode, T-Product, produit cosine, traitement des images et des vidéos, équations de Sylvester, reconnaissance faciale, extrapolation.

Abstract

In recent years, several research areas related to tensors (multidimensional matrices) have been exploited, non linear optimization, multilinear algebra, non linear analysis, statistics, programming language. In the modeling of several phenomena, the tensor representation proves efficiency in front of the matrix one. Indeed, the real data processing from different problems often leading to overwhelming demands on the computer resources (storage, calculation cost. . .). Convinced of the efficiency of the tonsorial approach, we propose in this thesis, a study that targets the generalization of numerical matrix methods via tonsorial formalism. In particular, we have focused on the extension of the Krylov subspace methods and their applications. More precisely, we have introduced tonsorial global and block Krylov subspace methods related to the various tensor products already existing. We have even succeeded in developing a new class of these methods named tubal Krylov subspace methods. Moreover, we propose applications of these methods in the resolution of linear and nonlinear tensor systems, image and video restoration, as well as the resolution of Sylvester tensor equations

---

Key Words (5): Tensors, Krylov subspace, Einstein product, n mode product, T-product, Cosine product, image and video processing, Sylvester equations, facial recognition, Extrapolation.