Numerical linear algebra and data analysis in large dimensions using tensor format

Martina Iannacito

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR
DE L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

MATHÉMATIQUES APPLIQUÉES ET CALCUL SCIENTIFIQUE

Par **Martina Iannacito**

Numerical linear algebra and data analysis in large dimensions using tensor format

Sous la direction de : **Olivier Coulaud** et **Luc Giraud**

Soutenue le 9 décembre 2022
Membres du jury :

K. Meerbergen	Professeur	Katholieke Universiteit Leuven	Président du jury Rapporteur
D. Kressner	Professeur	École polytechnique fédérale de Lausanne	Rapporteur
A. Franc	Directeur de Recherche	Inrae, Inria centre at the University of Bordeaux	Examinateur
A. Nouy	Professeur des universités	Université de Nantes	Examinateur
V. Simoncini	Professeure	Alma Mater Studiorum Università di Bologna	Examinatrice
N. Vannieuwenhoven	Professeur assistant	Katholieke Universiteit Leuven	Examinateur
O. Coulaud	Directeur de Recherche	Inria centre at the University of Bordeaux	Directeur de thèse
L. Giraud	Directeur de Recherche	Inria centre at the University of Bordeaux	Co-directeur

Algèbre linéaire numérique et analyse de données en grande dimensions utilisant le format tenseur

Résumé : L'objectif de ce travail est d'établir quelles propriétés théoriques des techniques d'algèbre linéaire classique développées dans deux contextes différents, à savoir l'algèbre linéaire numérique et l'analyse de données, sont préservées et lesquelles sont perdues, lorsqu'elles sont étendues aux tenseurs de rang faible. En outre, ce manuscrit vise à mettre en évidence les avantages et les inconvénients d'une approche tensorielle par rapport à son homologue matricielle classique dans les deux domaines considérés, avec une attention particulière aux aspects computationnels.

Dans la partie d'algèbre linéaire numérique, nous étudions expérimentalement les effets des erreurs d'arrondi sur un solveur itératif et plusieurs méthodes d'orthogonalisation, lorsqu'ils sont étendus aux tenseurs par le formalisme du Train Tensoriel (TT). Dans tous les algorithmes considérés, nous introduisons des étapes d'arrondi supplémentaires, avec l'algorithme de compression TT-rounding, pour faire face aux contraintes de mémoire, toujours cruciales lorsqu'on traite des tenseurs. Nos tests suggèrent que pour ces algorithmes, les limites classiques dues à la propagation des erreurs d'arrondi restent valables, en remplaçant la précision de l'arithmétique par celle de l'algorithme TT-rounding.
Le solveur itératif considéré est le Generalised Minimal RESidual (GMRES). Nous comparons notre version de TT-GMRES avec une réalisation précédente, en montrant numériquement sa grande robustesse. De plus, nous abordons le problème de la résolution simultanée par TT-GMRES de nombreux systèmes linéaires au format TT et établissons des bornes qui garantissent la qualité numérique de la solution individuelle extraite.
Les schémas classiques d'orthogonalisation généralisés aux tenseurs sont CGS, CGS2, MGS, MGS2, Householder et Gram. Pour compléter leur étude, nous étudions comment ils affectent les performances du solveur de problèmes aux valeurs propres basé sur des itérations de sous-espaces étendu aux tenseurs avec le format TT.

Dans la partie analyse de données, nous étudions deux techniques d'analyse de données, l'une destinée aux données de variables catégorielles et l'autre aux données climatiques, généralisées aux tenseurs par le biais du format Tucker, en soulignant les avantages et les inconvénients de ce choix par rapport à l'approche matricielle correspondante.
L'Analyse des Correspondances (AC) est un outil bien connu pour visualiser et interpréter des tableaux catégoriels à deux variables. Nous étudions géométriquement la généralisation de l'AC aux tableaux multivoies par la technique de décomposition tensorielle de Tucker, contribuant ainsi à la compréhension de l'Analyse des Correspondances MultiVoies (ACMV). Les résultats théoriques sont complétés par des exemples de ACMV appliqués à des ensembles de données. En particulier, nous réalisons l'ACMV sur le jeu de données écologique original mis à notre disposition dans le cadre du projet Malabar.
Pour les données climatiques, nous considérons l'analyse de la Fonction Orthogonale Empirique (EOF). En particulier, nous montrons comment récupérer le résultat final de l'EOF en s'appuyant sur le format compressé de Tucker. Cette approche peut être avantageuse sur le plan du calcul si les données sont disponibles directement au format Tucker. Pour être complet, nous étudions numériquement l'effet de l'approximation des données par le modèle de Tucker sur le résultat EOF final.

Mots-clés : calcul tensoriel, compression de rang faible, algèbre linéaire numérique, analyse de données.

Numerical linear algebra and data analysis in large dimensions using tensor format

Abstract: This work aims to establish which theoretical properties of classical linear algebra techniques developed in two different contexts, that are numerical linear algebra and data analysis, are saved and which are lost, once they are extended to tensors through tensor compression algorithms. Moreover, this manuscript aims to highlight the benefits and the flaws of a tensor approach compared to its classical matrix counterpart in the two considered frameworks paying particular attention to the computational aspects.

In the numerical linear algebra part, we study experimentally the rounding error effects for an iterative solver and several orthogonalization kernels, when they are extended to the tensor framework through the Tensor Train (TT) formalism. In all the considered algorithms, we introduce additional rounding steps, through the TT-rounding algorithm to face memory constraints, always crucial when dealing with tensors. Our experiments suggest that for these algorithms the classical bounds based on rounding error analysis hold, replacing the unit round-off of the finite precision arithmetic with the precision of the TT-rounding algorithm.
The considered iterative solver is Generalised Minimal RESidual (GMRES). We compare our version of TT-GMRES with the previous realization, showing numerically its major robustness. Moreover, we address the problem of solving simultaneously through TT-GMRES many linear systems in TT-format and establishing bounds that guarantee the numerical quality of the individual extracted solutions.
The classical orthogonalization schemes generalized to tensors are CGS, CGS2, MGS, MGS2, Householder, and Gram. To complete their study, we study how they affect the performance of the subspace iteration eigensolver extended to tensors through the TT-format.

In the data analysis part, we investigate two data analysis techniques, one meant for categorical variables data and one for climate data, generalized to tensors through the Tucker format, highlighting the benefits and the flaws of this choice compared to the corresponding matrix approach.
A well-known tool for visualizing and interpreting categorical two-variable tables is Correspondence Analysis (CA). We study geometrically the generalization of CA to multiway tables through the Tucker tensor decomposition technique, contributing to the understanding of the MultiWay Correspondance Analysis (MWCA). The theoretical results are complemented by examples of MWCA applied to real-life datasets. In particular, we perform the MWCA on the original ecology dataset made available in the Malabar project.
For climate data, we consider the Empirical Orthogonal Function (EOF) analysis. In particular, we show how to retrieve the final EOF outcome relying on the Tucker compressed format. This approach may be computationally beneficial if the data are made available directly in Tucker format. For completeness, we study numerically the effect of the data approximation through the Tucker model on the final EOF outcome.

Keywords: tensor calculus, low rank compression, numerical linear algebra, data analysis.

Équipe-projet CONCACE
Centre Inria de l'université de Bordeaux, 334005 Talence, France.

Acknowledgements

As is often the case, this section happens to be the last and maybe the most engaging to be written. So I hope you, reader, will be patient and merciful.

Preparing this Ph.D. thesis has been an incredible adventure with its expected and unpredictable challenges. Thanks to the extraordinary people I had the chance to meet during this journey, I grew up from a professional and personal viewpoint, discovering skills and resources I did not think I had. This is why I will take as much space as I need to thank them all.

Firstly, I am deeply grateful to my advisors Olivier Coulaud and Luc Giraud. Since the very first moment, Olivier, you helped me navigate research and French bureaucracy, providing all the support I needed. Your office door was open every time I had questions or doubts, and you always addressed them with patience and kindness, which I deeply admire. Scientific exchanges have often fuelled my curiosity, prompting me to look at the topics from other angles, while our personal talks have always brought me back to a good mood. Not only, I learned a lot about tensors and good coding practices, but also how to be a better co-worker. For this reason, I have to praise the support of Luc too. I did not have the chance to spend much time in real life with you, Luc, but during our weekly video meetings, you taught me a lot, introducing me to the finite precision arithmetic and GMRES method. Your question about writing the Householder transformation algorithm in tensor format gave me food for thought, my favorite research aspect. As if that was not enough, your diplomacy in handling working relations and your consideration for the well-being of everyone have been of inspiration to me. Thank you both for supervising my work, and helping my development throughout the good and the bad times. Your encouragement made me overcome difficult moments, and your feedback significantly raised the quality of my thesis.

I would also like to express my gratitude to the member of my Ph.D. jury. I want to thank especially Daniel Kressner and Karl Meerbergen, who kindly accepted to review this manuscript, offering valuable insights and pointing out other possible unsolved, but related questions. My next thought goes to Nick Vannieuwenhoven, who not only took his time to carefully read the thesis highlighting some weaknesses but also invited me to participate in the minisymposium he co-organized for the SIAM Conference on Mathematics of Data Science. Attending this conference has been an amazing experience, during which I met many brilliant researchers and learned a lot from them. I want to express my gratitude to Anthony Nouy and Valeria Simoncini, for the interesting talks they gave and for the constructive criticisms they provided during the thesis defence. Finally, I would like to

thank Alain Franc for letting me work on the Malabar data and for sharing his expertise with me.

I can't help expressing my gratitude to the previously HiePACS, currently Concace and Topal team members. My first thought goes to Alena Shilova, Esragul Korkmaz, and Yanfei Xiang. Thank you, Alena, for your inspiring strength and passion, especially when you initiated me to babyfoot. Thank you, Esra, for your imaginative crazy talks and your endless availability whenever I needed help or support. Thank you Yanfei, for your deep kindness and your spontaneous cheerfulness, which frequently remarked me of the beauty of life. I will never forget our chats, our jokes, and our girl babyfoot games. Then, I want to express my appreciation for Marek Felsoci's good mood and pragmatism, Mathieu Verité's understanding and availability, and Romain Peressoni's humour and diplomacy. Thank you, Marek, for your incredible translations and for helping me move boxes. Thank you, Mathieu, for your understating and for reminding me that going through Ph.D. difficulties is normal. Last but not least, thank you, Romain, for all the fun you created, the help in writing diplomatic emails, the babyfoot suggestions you gave me, and for having been an extremely helpful secretary, in spite of the salary I offered you. I am grateful to those who left the team for new adventures, such as Gilles Marait and Nick Schenkels, and to those who just joined it, such as Jean-François David and Xunyi Zhao. Thank you, Gilles, for your patience in answering all my informatics doubts and for the open-hearted help you provided me, especially in preparing the pot[1]. Thank you, Nick, even if we spent little time together because of covid, your work played a key role in my thesis. Thank you, Jean-François for your tenderness and shyness, which reminded me of my first months in France. Thank you, Xunyi, for your contagious energy and positiveness, your adventures made me even more curious about life. I must not forget to express my gratitude to Emmanuel Agullo. You helped me dive deeply into the finite precision arithmetic and progressing in my research path, but most importantly, you made me a better babyfoot player (even if some scandalmongers would say a better cheater, assuming it is even possible). Moreover, I want to express my gratitude to Mathieu Faverge, who let me teach his numerical algorithm class at Enseirb, and instructed me together with Amina Guermouche on the possibilities to proceed in my career. Lastly, I am deeply grateful to all the remaining members of the teams, as Abdou Guermouche, Alycia Lisito, Aurélien Esnard, Florent Pruvost, Guillaume Sylvand, Lionel Eyraud-Dubois, Olivier Beaumont, Pierre Estérie, Pierre Ramet, Yulia Gusak, who were always available for a talk or advice.

All this experience would not have been possible without all the casual meetings that led me here. So, I will take some more lines to express the gratitude I felt and did not mention in the past. Firstly I want to thank Alessandra Bernardi, my master advisor, who introduced me to tensors and igniting in me the spark for this surprising and vast topic, as only the best teachers know how to do. Then, I have to thank Claudia De Lazzari, who has been like a big sister to me during my master's and Ph.D. time, providing me feedback, support, and understanding. From my past years, I can't help from thanking Dario Reggiani, Fabio Ricchieri, Giovanni Calza, Giovanni Zazzali (who never shows up,

[1] After defence reception

unfortunately), Giulia Vighi, Marco Langella, Maria Laura Rossi, Rocco Mora, Sara Martani and Stefano Ardizzoni. You all together made the three years at the University of Parma unforgettable. Gathering every year during Christmas or summer times feels like getting back to the old, but gold times. My deep gratitude goes to all the friends I made during my high-school time in Cremona, namely Alexei Farina, Anna Ginestri, Carla Faroni, Carolina Carrera, Lara Fusar Poli, Irene Forzoni, Marco Luvié, Maria Politi, Mattia Pollenghi, Nicole Chiozzi, Valentina Gualazzi, Valentina Negro, Valeria Lodigiani. I am well aware I am not spending enough time with you, but I am proud to say that even if the quantiy of time has reduced, the quality has not changed over the years. A thought of recognition for the people who helped me settle in Bordeaux goes to Masataka Sawayama, Cristina Vaghi, Michele Giuliano Carlino, Luca Cirrottola, and Silvia Pagliarini. Talking from time to time with them, in my mother tongue for some, has been of great help to my mood. I am grateful to my future advisor Lieven De Lathauwer, who helped me enlarge my knowledge during the postdoc interview and who I am sure will help me deepen my research. I am looking forward to joining the Stadius team in Belgium.

Even if I can already hear Olivier, Esra, and Yanfei murmuring about the excessive length of this section[2], there are still people I need to thank. To begin with, I am deeply thankful to my parents, Amelia and Alfonso, who since the very first moment have lovely fed me and my curiosity, inspiring me to do my best. I am aware sometimes it is difficult to be with me, but I love you so much for all you ever gave and still give me. Then, I have to thank Daniela, my younger sister, who went through hot and cold with me, but who never forgot to show me her love. I am deeply grateful to my family members, my cousins Benedetto and Valentino, aunts Antonella, Lucia, and Tiziana, uncle Gianfranco, grandmother Gina, and to Aldo, Danilo, Francesco, Giovanni, and Rosa, who unfortunately can not share this moment with us, but who certainly would be proudly joyful. My real last thought goes to you, Rocco, who spent so many happy years close to me, always knowing how to wring a smile from me, even in my worst moments. Thank you. And thank you passing reader, who has come this far and still has much to read.

[2]It really happened!

Contents

Résumé étendu

Tôt ou tard, presque toutes les disciplines, des mathématiques pures à l'ingénierie, de la physique aux sciences naturelles, de l'économie aux sciences sociales, se tournent vers l'algèbre linéaire et ses résultats numériques pour énoncer mathématiquement leurs problèmes et pour espérer les résoudre. Les améliorations technologiques récentes, rapides et en croissance continue, ouvrent de nouvelles questions mathématiques et informatiques, ce qui rend nécessaire le développement de l'algèbre linéaire et de ses branches appliquées. En particulier, la popularité croissante des données volumineuses et des problèmes de haute dimension stimule les progrès de l'algèbre multilinéaire et de ses aspects numériques.

Les principaux objets d'étude de l'algèbre multilinéaire sont les tenseurs et les fonctions multilinéaires. Les tenseurs en tant qu'outils mathématiques sont apparus dès le XIXème et le XXème siècle dans les études de géométrie de Riemann, Christoffel, Ricci-Curbastro et Levi-Civita et dans les travaux de physique de Voigt et Hamilton [42]. Cependant, la première étude mathématique sur leur nature est apparue plus tard en 1927 par Hitchcock [42, 72]. A partir de ce moment, de nombreux travaux issus de domaines très différents ont contribué à la théorie des tenseurs, en proposant des algorithmes de factorisation des tenseurs, par exemple, le Tucker, le FACteur PARAllel (PARAFAC) la DECOMposition CANonique (CANDECOM), les techniques Matrix-Product State (MPS) étudiées respectivement par Cattel [27], Tucker [137], Harshman [65], Carroll et Chang [25], White [2]. Ces résultats ont été redécouverts et développés pour aboutir aux techniques de factorisation et d'approximation tensorielles les plus populaires actuellement, à savoir la décomposition polyadique canonique (CP) [72], la décomposition de Tucker [137] calculée avec l'algorithme HOSVD (High Order Singular Value Decomposition), (récemment renommé Multilinear Singular Value Decomposition [36]), la décomposition de Tucker hiérarchique (H-Tucker) [53] et la décomposition TT (Tensor Train) [108]. Nous renvoyons le lecteur à [84] pour plus de détails sur ces techniques. Ces méthodes et les algorithmes qui leur sont associés, avec leurs avantages et leurs inconvénients, ont des domaines d'application spécifiques, ce qui pousse à approfondir leurs propriétés mathématiques et à trouver de nouvelles possibilités d'application.

En s'appuyant sur les différentes méthodes de décomposition tensorielle, l'objectif de ce travail est d'établir quelles propriétés théoriques des techniques classiques d'algèbre linéaire développées dans deux contextes différents, à savoir l'algèbre linéaire numérique et l'analyse de données, sont conservées et lesquelles sont perdues, une fois qu'elles sont étendues aux tenseurs. Une attention particulière est consacrée aux aspects computationnels. En outre, ce manuscrit vise à mettre en évidence les avantages et les inconvénients

d'une approche tensorielle par rapport à son homologue matricielle classique dans les deux cadres considérés.

Algèbre linéaire numérique

La façon dont un algorithme propage les erreurs d'arrondi est un sujet clé en algèbre linéaire numérique. Classiquement, l'étude est effectuée avec le modèle IEEE de l'arithmétique à virgule flottante, c'est-à-dire en affirmant que la représentation informatique de $x \in \mathbb{R}$ est $\hat{x} = x + \delta$ avec $|\delta| \leq u$, où u est le *unité d'arrondi* et il limite la précision de la représentation de x. De même, le calcul arithmétique en précision finie induit des perturbations relatives similaires dans tous les calculs élémentaires.

L'objectif de la Partie I est d'étudier numériquement les effets de l'erreur d'arrondi pour un solveur itératif et six schémas d'orthogonalisation classiques lorsqu'ils sont étendus au cadre tensoriel par le formalisme TT. Dans tous les algorithmes considérés, nous introduisons des étapes d'arrondi supplémentaires, par le biais de l'algorithme d'arrondi TT [108] pour faire face aux contraintes de mémoire, toujours cruciales lorsqu'on traite des tenseurs. Nos expériences suggèrent que pour ces algorithmes, les limites classiques dues à la propagation des erreurs d'arrondi sont valables, en remplaçant l'arrondi unitaire u de l'arithmétique de précision finie par la précision δ de l'algorithme d'arrondi TT.

Generalized Minimal RESidual

La Generalized Minimal RESidual (GMRES) est une méthode itérative robuste permettant de résoudre des systèmes linéaires. Le GMRES repose sur la technique du sous-espace de Krylov pour l'approximation de la solution. La solution approchée est calculée en minimisant la norme du résidu du système linéaire sur l'espace de Krylov associé, dont la dimension augmente de manière itérative. Le problème de minimisation est simplifié par la construction d'une base orthogonale de l'espace de Krylov à travers le MGS et le schéma de Householder, ce qui conduit à l'implémentation du MGS-GMRES et du Householder-GMRES présentés respectivement dans [124] et [142]. Un critère d'arrêt significatif et fréquemment utilisé est basé sur l'erreur à rebours associée à la matrice et au côté droit du système linéaire. Ce choix est motivé par la stabilité à rebours des deux variantes de GMRES dans l'arithmétique classique en virgule flottante, prouvée respectivement dans [112] et [40]. Cette propriété de stabilité à rebours garantit que la solution calculée par GMRES peut être interprétée comme la solution du même système linéaire avec des perturbations de norme relative sur la matrice et le côté droit qui sont limitées par l'arrondi unitaire de la précision de travail.

Contribution

La première contribution à l'étude du solveur GMRES, présentée au début du Chapitre 2, provient d'un travail conjoint avec quelques scientifiques de l'équipe Inria de Concace (anciennement HiePACS), à savoir E. Agullo, O. Coulaud, L. Giraud, G. Marait

et N. Schenkels. Le but de notre travail est d'étudier numériquement la stabilité arrière de GMRES lorsque la précision de la représentation des données et celle des calculs sont différentes. En effet, dans certaines situations, une partie des données doit être compressée avec une précision différente (généralement inférieure) de celle du calcul en raison de contraintes de mémoire de stockage. La procédure de compression introduit des perturbations, qui sont numériquement décrites par une erreur de représentation limitée par la précision de la compression. Ainsi, nous avons deux erreurs de représentation, l'une due à la compression et l'autre liée à l'arithmétique à précision finie. Si la compression perturbe les données par composant, l'analyse théorique de stabilité arrière de GMRES [40, 113] s'applique toujours. Cependant, il existe des cas où les erreurs de compression affectent les données de manière normalisée, de sorte que les hypothèses mathématiques de l'analyse [40, 113] ne s'appliquent pas facilement. Nos exemples numériques suggèrent que même avec des perturbations de compression normales, les deux variantes stables en arrière de GMRES restent stables en arrière, c'est-à-dire que l'erreur arrière normale atteignable est du même ordre que le maximum entre la compression et la précision de calcul. Tous les résultats de ce travail sont également présentés dans [3].

La deuxième partie du Chapitre 2 présente la deuxième contribution liée à la GMRES : l'étude numérique d'un algorithme GMRES robuste adapté au cas tensoriel avec le formalisme TT, brièvement appelé TT-GMRES. Ce solveur itératif est destiné à résoudre des systèmes linéaires, dont l'opérateur agit parmi des espaces tensoriels d'ordre d. Pour simplifier, nous appellerons ces systèmes linéaires des systèmes linéaires tensoriels d'ordre d. Nous présentons l'algorithme TT-GMRES avec l'erreur à rebours comme critère d'arrêt et quelques exemples numériques, qui soutiennent sa stabilité à rebours, obtenus à partir de la discrétisation par des grilles cartésiennes de certaines EDP classiques. En effet, GMRES dans le format TT représente une étude de cas des perturbations normales, décrites dans la première partie du Chapitre 2 et dans [3]. Pour être complet, notre algorithme TT-GMRES est comparé à une précédente adaptation de GMRES au cadre tensoriel présentée dans [39]. Nous montrons que la réalisation proposée dans [39] n'est pas stable en arrière et ne garantit pas la précision de la solution calculée. De plus, nous nous concentrons sur la résolution avec TT-GMRES des systèmes linéaires tensoriels dont les opérateurs ou les côtés droits dépendent d'un paramètre. Sous une certaine hypothèse mathématique, lorsque l'opérateur ou le côté droit dépend d'un paramètre, la solution de p systèmes linéaires tensoriels d'ordre d peut être calculée en résolvant un unique système linéaire tensoriel d'ordre $(d+1)$ avec une taille p le long du $(d+1)$-ième mode. Nous prouvons théoriquement que dans ce cadre, il existe des bornes d'erreur arrière reliant la solution du système tensoriel unique d'ordre d à sa solution correspondante extraite de la solution du système linéaire tensoriel d'ordre $(d+1)$. La qualité de ces bornes est également étudiée numériquement sur divers exemples. Pour être aussi complet que possible, nous étudions numériquement la qualité du préconditionneur. Ces résultats sont rassemblés dans [31].

Schémas d'orthogonalisation

Nous considérons six algorithmes pour calculer une base orthonormale de l'espace couvert par un ensemble de vecteurs linéairement indépendants : Gram-Schmidt Classique (CGS) et sa version avec réorthogonalisation (CGS2), Gram-Schmidt Modifié (MGS) et sa mise en oeuvre avec réorthogonalisation (MGS2), la transformation de Householder et l'algorithme de Gram. L'estimation de la *perte d'orthogonalité* de la base calculée est importante pour évaluer la qualité d'un schéma d'orthogonalisation. La perte d'orthogonalité pour Householder est prouvée dans [143] comme étant $\mathcal{O}(u)$; la même limite est valable pour MGS2 et CGS2 comme montré dans[51, 129] sous l'hypothèse que $\kappa(A)u < 1$ et $\kappa^2(A)u < 1$ respectivement, où $\kappa(A)$ est le nombre de condition de l'ensemble des vecteurs d'entrée à orthogonaliser. Dans [15], l'auteur indique que si $\kappa(A)u < 1$, alors la perte d'orthogonalité du MGS est $\mathcal{O}(\kappa(A)u)$, tandis que celle du CGS est $\mathcal{O}(\kappa^2(A)u)$. Enfin, dans [131], il est prouvé que $\mathcal{O}(\kappa^2(A)u)$ limite également la perte d'orthogonalité de la technique d'orthogonalisation de Gram.

Contribution

Dans le Chapitre 3, nous étudions d'un point de vue numérique la perte d'orthogonalité de ces six algorithmes d'orthogonalisation classiques et populaires, correctement adaptés pour fonctionner dans le format tensoriel. Plus précisément, nous reformulons les algorithmes de Gram, CGS et MGS, avec et sans réorthogonalisation, en remplaçant l'arithmétique vectorielle classique par l'arithmétique TT. Alors que ces méthodes lisent directement les tenseurs, l'algorithme de transformation de Householder nécessite un plan minutieux et bien pensé, que nous décrivons, puisque les entrées des tenseurs en format compressé ne sont pas directement accessibles. De plus, dans les six schémas d'orthogonalisation, nous introduisons une ou plusieurs étapes d'arrondi à une précision donnée, basée sur l'arrondi TT [108]. Les étapes d'arrondi jouent un rôle crucial : d'une part, elles maintiennent l'exigence de mémoire à un niveau abordable, d'autre part, leur précision prend le rôle de l'arrondi unitaire de l'analyse classique de stabilité à rebours. En effet, nos expériences numériques suggèrent que les limites théoriques prouvées pour les schémas d'orthogonalisation classiques dans le contexte IEEE s'appliquent toujours dans le cadre TT, où la précision de l'arrondi remplace l'arrondi unitaire. Ces résultats, également disponibles dans [32], n'ont jamais été rapportés auparavant, à notre connaissance.

L'analyse des besoins en mémoire et l'application de ces noyaux dans un eigensolveur classique, adapté correctement au cadre tensoriel, complètent l'étude numérique des schémas d'orthogonalisation au format TT. Nous considérons la méthode d'itération subspatiale [9] pour travailler avec des tenseurs dans le format TT, et nous étudions les performances des différents algorithmes d'orthogonalisation. Enfin, nous comparons la qualité et les propriétés des paires propres en fonction des différents noyaux d'orthogonalisation pour un cas test académique, à savoir l'opérateur Laplacien exprimé au format TT [81].

Analyse des données

L'analyse des données est la discipline de la science des données qui explore les méthodes d'extraction, d'interprétation et de visualisation des caractéristiques importantes des données. Plusieurs techniques développées en analyse de données reposent sur des outils d'algèbre linéaire, par exemple, l'Analyse en Composantes Principales (ACP) et toutes ses variantes adaptées aux différents contextes, qui sont basées sur la SVD. Depuis l'arrivée d'ensembles de données de grande taille provenant d'entreprises technologiques publiques et privées, les modèles tensoriels ont gagné en popularité, offrant de nouvelles directions de recherche dans différents domaines.

La Partie II a pour but d'étudier certaines techniques d'analyse de données pour les tenseurs exprimés au format Tucker, en soulignant les avantages et les défauts de ce choix par rapport à l'approche matricielle correspondante.

Analyse des correspondances

L'Analyse des Correspondances (AC) est l'outil couramment utilisé pour analyser et interpréter les tableaux de contingence, c'est-à-dire les ensembles de données dont les entrées sont des comptes ou des fréquences de deux catégories de deux variables différentes [57]. L'AC est basée sur l'ACP et, par conséquent, elle est à la fois une technique d'analyse et de visualisation, fournissant simultanément un point de vue algébrique, géométrique et statistique sur les données [12, 14, 57, 58].

Lorsque plus de deux variables catégorielles sont présentes dans les ensembles de données, généralement appelés *tableaux à voies multiples*, la technique la plus utilisée pour l'analyse est l'Analyse des Correspondances Multiples (ACM). L'ACM procède en associant un tableau d'indicateurs ou de Burt, qui est mathématiquement une matrice de zéros et de uns, au tableau multivoie [58]. En raison de ce choix, toutes les interactions entre plus de deux variables sont perdues et ne peuvent être récupérées [10]. Cependant, il existe un outil alternatif appelé Analyse des Correspondances MultiVoies (ACMV) [11], basé sur la CP [72] et sur le modèle de Tucker [137]. Les formulations algébriques et statistiques de l'ACMV ont été présentées, au moins pour trois variables, dès les années 90, voir par exemple [44, 87], mais ses potentialités ne sont pas pleinement explorées, comme l'indiquent [11, 88].

Contribution

Dans le Chapitre 4, nous étudions l'ACMV principalement du point de vue de la géométrie et de la visualisation, afin de combler le vide existant pour cette technique généralisée aux tenseurs [11, 88]. Nos contributions théoriques et pratiques à ACMV proviennent d'un travail conjoint avec O. Coulaud et A. Franc, chercheur Inrae dans l'équipe BioGeCo Inrae, également membre de l'équipe du projet Pléiade Inria. Cette collaboration vise à étudier le rôle et les avantages de l'utilisation de la théorie tensorielle pour l'étude des tableaux de contingence multivoie écologiques. Tout d'abord, nous contribuons à la

description des propriétés géométriques du ACMV. Après avoir développé sa structure algébrique basée sur le modèle de Tucker pour un nombre générique de variables, nous nous concentrons sur la visualisation de l'ACMV et sur les interprétations géométriques qui en sont déduites. Pour mettre en évidence les avantages et les inconvénients d'une approche tensorielle dans la pratique, deux cas d'étude montrent la différence entre la visualisation et l'interprétation de l'ACM et de l'AC.

La table de contingence multivoie fournie par le projet Malabar (IFREMER, CNRS, INRAE, Labex COTE) [6] nous permet de tester nos résultats théoriques sur un cas d'étude réel en écologie. Notre contribution pratique est l'analyse de ce jeu de données Malabar. Nous appliquons les résultats de l'ACMV développés précédemment pour interpréter géométriquement les relations entre les variables de données. Pour être complet, nous comparons les résultats de l'ACMV avec ceux de l'AC, en transformant convenablement les données.

Analyse des fonctions orthogonales empiriques

En climatologie, une méthode populaire pour l'étude et la visualisation de données dépendant de variables spatiales et temporelles est l'analyse des Fonctions Orthogonales Empiriques (EOF), basée sur l'ACP. La technique EOF est considérée à la fois comme une réduction dimensionnelle et comme une technique d'extraction de motifs [64], conduisant à la visualisation indépendante de l'information spatiale et temporelle portée par les données analysées. A notre connaissance, aucune description de la méthode EOF dans le cadre tensoriel n'a été présentée jusqu'à présent, même si des études de données climatiques basées sur des modèles tensoriels sont disponibles, par exemple [149].

Contribution

Au cours de la collaboration avec O. Coulaud et L. Terray, chercheur au sein de l'équipe Climat, Environnement, Couplage et Incertitudes (CECI), une équipe de recherche commune entre le Cerfacs et le CNRS, nous avons étudié les avantages et les inconvénients de l'introduction de la théorie tensorielle dans la méthode EOF. Dans le Chapitre 5, nous concluons que du point de vue du coût de calcul, une approche tensorielle n'est pas bénéfique en général. Cependant, nous montrons que le coût de calcul de la technique EOF peut être réduit, sous la forte hypothèse que les données climatiques sont disponibles ou approximées en format Tucker. En effet, nous récupérons théoriquement les résultats EOF à partir des données compressées exprimées au format Tucker. Pour compléter l'étude, nous approximons un jeu de données par la technique HOSVD et étudions les erreurs entre les résultats EOF générés par les techniques classiques et tensorielles.

Extended summary

Sooner or later, almost every discipline, from pure mathematics to engineering, from physics to natural science, from economics to social science, turns to linear algebra and its numerical results for mathematically stating their problems and for hopefully solving them. The recent fast and continuously growing technological improvements open new mathematical and computational questions, making necessary further development in linear algebra and its applied branch. In particular, the increased popularity of big data and high dimensional problems is driving the progress of multilinear algebra and its numerical aspects.

Multilinear algebra main objects of investigation are tensors and multilinear functions. Tensors as mathematical tools appeared already between the XIX and the XX century in the geometry studies of Riemann, Christoffel, Ricci-Curbastro, and Levi-Civita and in the physics works of Voigt and Hamilton [42]. However, the first mathematical study on their nature appeared later in 1927 by Hitchcock [42, 72]. From that moment, many works from very different domains contributed to the tensor theory, proposing tensor factorization algorithms, for example, the Tucker, the PARAllel FACtor (PARAFAC) the CANonical DECOMposition (CANDECOM), Matrix-Product State (MPS) techniques investigated by Cattel [27], Tucker [137], Harshman [65], Carroll and Chang [25], White [2] respectively. Those results were rediscovered and developed further leading to the currently most popular tensor factorization and approximation techniques, which are the Canonical Polyadic decomposition (CP) [72], the Tucker decomposition [137] computed with High Order Singular Value Decomposition (HOSVD) algorithm, (recently renamed as Multilinear Singular Value Decomposition [36]), the Hierarchical Tucker decomposition (H-Tucker) [53] and the Tensor Train (TT) decomposition [108]. We refer the reader to [84] for further details about these techniques. These methods and their related algorithms, with advantages and drawbacks, have specific areas of applications, which in turn push for further investigation of their mathematical properties and new application possibilities.

Relying on the different tensor decomposition methods, the purpose of this work is to establish which theoretical properties of classical linear algebra techniques developed in two different contexts, that are numerical linear algebra and data analysis, are saved and which are lost, once they are extended to tensors. Particular attention is dedicated to the computational aspects. Moreover, this manuscript aims to highlight the benefits and the flaws of a tensor approach compared to its classical matrix counterpart in the two considered frameworks.

Numerical linear algebra

How an algorithm propagates rounding errors is a key topic in numerical linear algebra. Classically, the investigation is performed with floating point arithmetic IEEE model, i.e., stating that the computer representation of $x \in \mathbb{R}$ is $\hat{x} = x + \delta$ with $|\delta| \leq u$, where u is the *unit round-off* and it bounds the accuracy of the representation of x. Likewise, the finite precision arithmetic calculation induces similar relative perturbations in all elementary computations.

The purpose of Part I is to study numerically the rounding error effects for an iterative solver and six classical orthogonalization schemes when they are extended to the tensor framework through the TT-formalism. In all the considered algorithms we introduce additional rounding steps, through the TT-rounding algorithm [108] to face memory constraints, always crucial when dealing with tensors. Our experiments suggest that for those algorithms classical bounds due to rounding error propagation hold, replacing the unit round-off u of the finite precision arithmetic by the precision δ of the TT-rounding algorithm.

Generalized Minimal RESidual

The Generalized Minimal RESidual is a robust iterative method to solve linear systems. GMRES relies on the Krylov subspace technique for approximating the solution. The approximated solution is computed by minimizing the norm of the linear system residual on the associated Krylov space, whose dimension increases iteratively. The minimization problem is simplified by constructing an orthogonal basis of the Krylov space through the MGS and the Householder scheme, leading to the MGS-GMRES and Householder-GMRES implementation presented in [124] and [142] respectively. A meaningful and frequently used stopping criterion is based on the backward error associated with the matrix and the right-hand side of the linear system. This choice is motivated by the backward stability of both GMRES variants in the classical floating point arithmetic proved in [112] and [40] respectively. This backward stability property ensures that the solution computed by GMRES can be interpreted as the solution of the same linear system with relative norm perturbations on the matrix and the right-hand side that are bounded by the unit round-off of the working precision.

Contribution

The first contribution to the study of the GMRES solver, presented at the beginning of Chapter 2, comes from a joint work with few scientists of the Concace (previously HiePACS) Inria team, namely E. Agullo, O. Coulaud, L. Giraud, G. Marait and N. Schenkels. The purpose of our work is to study numerically the backward stability of GMRES when the accuracy of the data representation and the computations are different. Indeed, in some situations, part of the data needs to be compressed with an accuracy different (usually lower) than the computational one due to storing memory constraints.

The compression procedure introduces perturbations, which are numerically described by a representation error bounded by the compression accuracy. Thus, we have two representation errors, one due to the compression and one linked to the finite precision arithmetic. If the compression perturbs the data componentwise, the theoretical backward stability analysis of GMRES [40, 113] still applies. However, there are cases when the compression errors affect the data normwise so that the mathematical assumptions of the analysis [40, 113] do not readily apply. Our numerical examples suggest that even with normwise compression perturbations, the two backward stable variants of GMRES remain backward stable, that is the attainable normwise backward error is of the same order as the maximum between the compression and the computational accuracy. All the results of this work are also presented in [3].

The second part of Chapter 2 presents the second contribution related to GMRES: the numerical study of a robust GMRES algorithm adapted to the tensor case with the TT-formalism, shortly called TT-GMRES. This iterative solver is meant for solving linear systems, whose operator acts among tensor spaces of order d. We will refer for simplicity to those linear systems as tensor linear systems of order d. We present the TT-GMRES algorithm with the backward error as a stopping criterion and some numerical examples, that support its backward stability, obtained from the discretization through Cartesian grids of some classical PDEs. Indeed, GMRES in TT-format represents a case study of normwise perturbations, described in the first part of Chapter 2 and in [3]. For completeness, our TT-GMRES algorithm is compared with a previous adaptation of GMRES to the tensor framework presented in [39]. We show that the realization proposed in [39] is not backward stable and does not guarantee the accuracy of the computed solution. Moreover, we focus on solving with TT-GMRES tensor linear systems whose operators or right-hand sides depend on a parameter. Under some mathematical hypothesis, when the operator or the right-hand side is parameter dependent, the solution of p tensor linear systems of order d can be computed by solving a unique tensor linear system of order $(d+1)$ with size p along the $(d+1)$-th mode. We prove theoretically that in this framework there exist backward error bounds linking the solution of the single tensor system of order d with its corresponding one extracted from the solution of the tensor linear system of order $(d+1)$. The quality of those bounds is also studied numerically on various examples. To be as comprehensive as possible, we numerically investigate the quality of the preconditioner. These results are collected in [31].

Orthogonalization schemes

We consider six algorithms to compute an orthonormal basis of the space spanned by a set of linearly independent vectors: Classical Gram-Schmidt (CGS) and its version with re-orthogonalization (CGS2), Modified Gram-Schmidt (MGS) and its implementation with re-orthogonalization (MGS2), Householder transformation and the Gram algorithm. Estimating the *loss of orthogonality* of the computed basis is important to evaluate the quality of an orthogonalization scheme. The loss of orthogonality for Householder is proved in [143] to be $\mathcal{O}(u)$; the same bound holds for MGS2 and CGS2 as showed in [51,

129] under the hypothesis that $\kappa(A)u < 1$ and $\kappa^2(A)u < 1$ respectively, where $\kappa(A)$ is the condition number of the input set of vectors to orthogonalize. In [15], the author states that if $\kappa(A)u < 1$, than the loss of orthogonality of MGS is $\mathcal{O}(\kappa(A)u)$, while the CGS one is $\mathcal{O}(\kappa^2(A)u)$. Finally, in [131], it is proved that $\mathcal{O}(\kappa^2(A)u)$ bounds also the loss of orthogonality of the Gram orthogonalization technique.

Contribution

In Chapter 3, we study from a numerical point of view the loss of orthogonality of these six classical and popular orthogonalization algorithms, properly adapted to work in the tensor format. More precisely, we reformulate the Gram, CGS and MGS algorithm, with and without re-orthogonalization, replacing the classical vector arithmetic by the TT-arithmetic. While those methods directly read for tensors, the Householder transformation algorithm requires a careful and well-thought-out outline, that we describe, since entries of compressed-format tensors are not directly accessible. Moreover, in all the six orthogonalization schemes, we introduce one or more rounding steps at a given accuracy, based on the TT-rounding [108]. The rounding steps play a crucial role: on one side, they keep the memory requirement affordable, on the other, their accuracy takes over the role of the unit round-off of the classical backward stability analysis. Indeed, our numerical experiments suggest that the theoretical bounds proved for the classical orthogonalization schemes in the IEEE context still apply in the TT-framework, where the rounding accuracy replaces the unit round-off. Those results, available also in [32], were never reported before, to the best of our knowledge.

The analysis of the memory requirement and the application of these kernels into a classical eigensolver, adapted properly to the tensor framework, complete the numerical study of the orthogonalization schemes in TT-format. We consider the subspace iteration method [9] to work with tensors in TT-format, and investigate the performance of the different orthogonalization algorithms. Finally, we compare the quality and the properties of the eigenpairs depending on the different orthogonalization kernels for an academic test case, i.e., the Laplacian operator expressed in TT-format [81].

Data analysis

Data analysis is the data science discipline that explores methods to extract, interpret and visualize important features of the data. Several techniques developed in data analysis rely on linear algebra tools, for example, Principal Component Analysis (PCA) and all its variations tuned to the different contexts, which are based on SVD. Since the advent of large-size datasets coming from public and private tech companies, tensor models have gained popularity providing new research directions in different fields.

The Part II aim is to investigate some data analysis techniques for tensors expressed in Tucker format, highlighting the benefits and the flaws of this choice compared to the corresponding matrix approach.

Correspondence analysis

Correspondence Analysis (CA) is the commonly used tool for analysing and interpreting contingency tables, i.e., datasets whose entries are counts or frequencies of two categories of two different variables [57]. CA is based on PCA and consequently, it is both an analysis and a visualization technique, providing simultaneously an algebraic, a geometric and a statistical point of view on the data [12, 14, 57, 58].

When more than two categorical variables are present in the datasets, usually referred to as *multiway tables*, the most used technique for the analysis is Multiple Correspondence Analysis (MCA). MCA proceeds by associating an indicator or Burt table, which mathematically is a matrix of zeros and ones, to the multiway table [58]. Because of this choice, all the interactions among more than two variables are lost and can not be retrieved [10]. However, there exists an alternative tool called MultiWay Correspondence Analysis (MWCA) [11], based on the CP [72] ad on the Tucker [137] model. The MWCA algebraic and statistical formulations were presented, at least for three variables, already in the 90s, see for example [44, 87], but its potentialities are not fully explored, as stated in [11, 88].

Contribution

In Chapter 4, we study MWCA mainly on the geometric and visualization side, to fulfil the existing gap for this technique generalized to tensors [11, 88]. Our theoretical and practical contributions to MWCA derive from joint work with O. Coulaud and A. Franc, Inrae research scientist in the BioGeCo Inrae team, also a member of the Pleiade Inria project team. This collaboration aims to investigate the role and the benefits of using the tensor theory for studying ecological multiway contingency tables. Firstly, we contribute to the description of the geometrical properties of MWCA. After developing its algebraic structure based on the Tucker model for a generic number of variables, we focus on MWCA visualization and on the geometrical interpretations that are inferred. To highlight the benefit and the flaws of a tensor approach in practice, two study cases show the difference between the MWCA and the CA visualization and interpretation.

The multiway contingency table provided by the Malabar project (IFREMER, CNRS, INRAE, Labex COTE) [6] allows us to test our theoretical results on a real ecology study case. Our practical contribution is the analysis of this Malabar dataset. We apply the previously developed MWCA results to geometrically interpret the relations among the data variables. For completeness, we compare the MWCA results with the CA ones, transforming conveniently the data.

Empirical Orthogonal Function analysis

In climatology, a popular method for the study and the visualization of data depending on space and time variables is Empirical Orthogonal Functions (EOF), based on PCA. The EOF technique is regarded both as a dimensional reduction and as a pattern extraction technique [64], leading to the independent visualization of the space and of the time

information carried by the analyzed data. To the best of our knowledge, no description of the EOF method in the tensor framework has been presented so far, even if studies of climate data based on tensor models are available, for example [149].

Contribution

During the collaboration with O. Coulaud and L. Terray, a research scientist in Climate, Environment, Coupling, and Uncertainties (CECI), a joint research team between Cerfacs and CNRS, we investigate the advantages and the drawbacks of introducing the tensor theory in the EOF method. In Chapter 5, we conclude that from the computational cost point of view a tensor approach is not beneficial in general. However, we show that the computational cost of the EOF technique can be reduced, under the strong assumption that the climate data are made available or approximated in Tucker format. Indeed, we theoretically retrieve the EOF results from the compressed data expressed in Tucker format. To complete the study, we approximate a dataset through the HOSVD technique and study the errors between the EOF results generated through classical and tensor techniques.

List of Symbols

Objects

$\underline{\mathbf{a}}_k$	k-th Tensor Train core of $\mathbf{a}$
$\underline{A}_k(i_k)$	i_k-th slice along mode 2 of k-th Tensor Train core $\underline{\mathbf{a}}_k$
$\mathfrak{V}^*$	dual of a linear subspace
$\mathbb{F}$	generic algebraic field
$\mathbb{I}_n$	identity matrix of size n
$\mathfrak{V}$	linear subspace
$A^{(i)}$	matricization of tensor $\mathbf{a}$ along mode i
A	matrix
$\mathcal{A}$	linear space endowed with a metric
$\mathbb{N}$	integer number set
$\boldsymbol{\mathcal{A}}$	linear or multilinear operator
$\mathbb{O}(m \times n)$	set of orthogonal matrices of m rows and n columns
$\mathbb{Q}$	rational number field
$\mathbb{R}$	real number field
$\mathbb{R}_+$	real positive number set
$\mathcal{A}$	set
$\mathbf{a}^{[i_k,k]}$	i_k-th slice along mode k
$\mathbf{a}$	multidimensional array as element of a tensor space
$\mathbf{A}$	multidimensional array representing a multilinear operator
a	vector

Operations

${}_{\mathcal{H}_\mathbf{L}}\bullet_{\mathcal{H}_\mathbf{R}}$	contraction of modes in $\mathcal{H}_\mathrm{L}$ and $\mathcal{H}_\mathrm{R}$ for left and right tensor respectively
$\langle \cdot, \cdot \rangle$	inner product
$\otimes_\mathbf{K}$	Kronecker product
$\|\cdot\|_2$	L2-norm
$\|\cdot\|$	Frobenious norm
$\otimes$	tensor product
$\times_k$	matrix-tensor product along mode k

Chapter 1

Notation and preliminary results

1.1 Introduction and notation

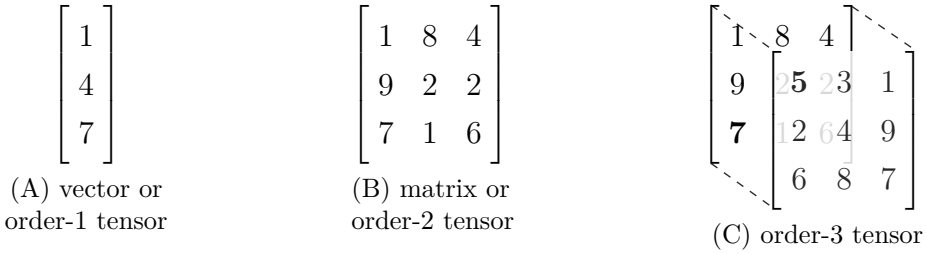
A tensor is a mathematical object carrying multiple meanings; we present briefly the main three: tensors as multidimensional arrays, as elements of a tensor space, and as representations of multilinear operators. From a very practical viewpoint, often used in computer science, a tensor is simply a multidimensional array. Exactly as a vector of $\mathbb{R}^n$ is identified with the one-dimensional array which stores its n real coordinates with respect to the canonical basis, a tensor of $\mathbb{R}^{n_1 \times \cdots \times n_d}$ can be seen as an array of d dimensions storing all its entries.

Definition 1.1.1. *The object* $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *is a* tensor*, the integer* d *defines its* order*, the integer* n_k *its* k*-th* mode size*.*

A visual representation of tensors as multidimensional arrays of order $d \in \{1, 2, 3\}$ is given in Figure 1.1.

Henceforth tensors are denoted by lowercase bold letters, matrices by uppercase letters, and vectors by lowercase ones. The expression $\mathbf{a}(i_1, \ldots, i_d)$ stands for the $(i_1, \ldots, i_d)$-th entry of $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. The analogous expression is used for matrix and vector components. To further clarify the chosen notation, we present an example.

Example 1.1.1. The multidimensional array $\mathbf{a}$ in Figure 1.1C is a tensor of order 3 with



(A) vector or order-1 tensor

(B) matrix or order-2 tensor

(C) order-3 tensor

Figure 1.1 – Tensors as multidimensional tensors of order $d \in \{1, 2, 3\}$.

mode sizes $(3, 3, 2)$. Its $(1, 1, 1)$-th element is $\mathbf{a}(1, 1, 1) = 5$, while its $(3, 1, 2)$-th element is $\mathbf{a}(3, 1, 2) = 7$.

From a more theoretical point of view typical of some disciplines, as multilinear algebra for example, tensors of order d are elements belonging to the tensor product, denoted by $\otimes$, of d linear subspaces $\mathfrak{V}_k$ defined over the field $\mathbb{F}$ for $k \in \{1, \dots, d\}$. Let $\{v_1^{(k)}, \dots, v_{r_k}^{(k)}\}$ be a basis of $\mathfrak{V}_k$ the linear subspace of $\mathbb{R}^{n_k}$ of dimension r_k, then $\mathfrak{V}_1 \otimes \cdots \otimes \mathfrak{V}_k$ is a linear space generated by the elements

$$v_{i_1}^{(1)} \otimes \cdots \otimes v_{i_d}^{(d)}$$

for every $i_k \in \{1, \dots, r_k\}$ and $k \in \{1, \dots, d\}$. Consequently every tensor $\mathbf{a} \in \mathfrak{V}_1 \otimes \cdots \otimes \mathfrak{V}_k$ can be written as a linear combination of the basis elements, by the definition of linear space, i.e.,

$$\mathbf{a} = \sum_{i_1, \dots, i_d = 1}^{r_1, \dots, r_d} \lambda_{i_1, \dots, i_d} v_{i_1}^{(1)} \otimes \cdots \otimes v_{i_d}^{(d)}.$$

The linear space $\mathfrak{V}_1 \otimes \cdots \otimes \mathfrak{V}_k$ is the *tensor product space*, in which we highlight a special set of elements.

Definition 1.1.2. *The tensor* $\mathbf{v} \in \mathfrak{V}_1 \otimes \cdots \otimes \mathfrak{V}_d$ *is an* elementary tensor *if it can be expressed as*

$$\mathbf{v} = v_1 \otimes \cdots \otimes v_d$$

with $v_k \in \mathfrak{V}_k$ *for every* $k \in \{1, \dots, d\}$.

The $(i_1, \dots, i_d)$-th element of the elementary tensor $\mathbf{v}$ is equal to the product of the i_k component of $v_k \in \mathfrak{V}_k$ for all $k \in \{1, \dots, d\}$, that is $\mathbf{v}(i_1, \dots, i_d) = v_1(i_1) \cdots v_d(i_d)$. It is convenient to present an example, which will clarify this last statement.

Example 1.1.2. Let $\mathfrak{V}_k$ be a subspace of dimension 1 generated by $v_k \in \mathbb{R}^{n_k}$ for $k \in \{1, 2, 3\}$ with

$$v_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \qquad v_2 = \begin{bmatrix} 9 \\ 1 \\ 1 \end{bmatrix} \qquad \text{and} \qquad v_3 = \begin{bmatrix} 2 \\ 8 \\ 0 \end{bmatrix}.$$

Then the tensor product space $\mathfrak{V}_1 \otimes \mathfrak{V}_2 \otimes \mathfrak{V}_3$ is generated by the order 3 tensor

$$\mathbf{v} = v_1 \otimes v_2 \otimes v_3,$$

that is all the elements of $\mathfrak{V}_1 \otimes \mathfrak{V}_2 \otimes \mathfrak{V}_3$ can be expressed as $(\lambda \mathbf{v})$ with $\lambda \in \mathbb{R}$.

Tensors as multidimensional arrays and as elements of the tensor product space are linked. Indeed, a vector of a linear space is uniquely identified by its coordinates once the linear space basis is fixed, similarly, a tensor as an element of the tensor product space is uniquely identified by a multidimensional array once a basis of the tensor product is set. We illustrate this link using the previous example.

Example 1.1.3. Considering the vectors v_h defined in Example 1.1.2, we express their tensor product in the canonical basis of the tensor space. Each vector v_h can be expressed as a linear combination of the canonical basis vectors $\{e^{(h)}_{i_h}\}$ of $\mathfrak{V}_h$ for $h \in \{1,2,3\}$ where the coefficients are its entries, i.e.,

$$\begin{aligned} v_1 &= \sum_{i=1}^{2} v_1(i) e^{(1)}_i = e^{(1)}_1 + 2e^{(1)}_2 \\ v_2 &= \sum_{j=1}^{3} v_2(j) e^{(2)}_j = 9e^{(2)}_1 + e^{(2)}_2 + e^{(2)}_3 \\ v_3 &= \sum_{k=1}^{3} v_3(k) e^{(3)}_k = 2e^{(3)}_1 + 8e^{(3)}_2 \end{aligned}$$

where $e^{(1)}_1 = [1,0]^\top$, $e^{(1)}_1 = [0,1]^\top$, $e^{(2)}_1 = e^{(3)}_1 = [1,0,0]^\top$, $e^{(2)}_2 = e^{(3)}_2 = [0,1,0]^\top$ and $e^{(2)}_3 = e^{(3)}_3 = [0,0,1]^\top$. As consequence, the tensor $\mathbf{v}$ gets

$$\mathbf{v} = v_1 \otimes v_2 \otimes v_3 = \Big(\sum_{i=1}^{2} v_1(i) e^{(1)}_i\Big) \otimes \Big(\sum_{j=1}^{3} v_2(j) e^{(2)}_j\Big) \otimes \Big(\sum_{k=1}^{3} v_3(k) e^{(3)}_k\Big)$$

and thanks to the multilinearity of the tensor product, this last equation becomes

$$\begin{aligned} \mathbf{v} &= \Big(\sum_{i=1}^{2} v_1(i) e^{(1)}_i\Big) \otimes \Big(\sum_{j=1}^{3} v_2(j) e^{(2)}_j\Big) \otimes \Big(\sum_{k=1}^{3} v_3(k) e^{(3)}_k\Big) \\ &= \sum_{i=1}^{2}\sum_{j=1}^{3}\sum_{k=1}^{3} \Big(v_1(i) v_2(j) v_3(k)\Big) e^{(1)}_i \otimes e^{(2)}_j \otimes e^{(3)}_k \\ &= \sum_{i=1}^{2}\sum_{j=1}^{3}\sum_{k=1}^{3} \mathbf{v}'(i,j,k) e^{(1)}_i \otimes e^{(2)}_j \otimes e^{(3)}_k \end{aligned}$$

where $\mathbf{v}'$ is a 3-dimensional array of modes $(2,3,3)$, whose (i_1,i_2,i_3)-th entry is the product of the i_h-th entry of v_h for $h \in \{1,2,3\}$, that is $\mathbf{v}'(i_1,i_2,i_3) = v_1(i_1)v_2(i_2)v_3(i_3)$. In this example, once the canonical basis is fixed for $\mathbb{R}^2$ and $\mathbb{R}^3$, the $(1,1,1)$ element of $\mathbf{v}$ is $1 \times 2 \times 9 = 18$.

Henceforth a d-dimensional array will represent the coordinates of a $\mathbb{R}^{n_1} \otimes \cdots \otimes \mathbb{R}^{n_d}$ tensor with respect to its canonical basis $\{e^{(1)}_{i_1} \otimes \cdots \otimes e^{(d)}_{i_d}\}$ where $\{e^{(k)}_{i_k}\}$ is the canonical basis of $\mathbb{R}^{n_k}$ for every $k \in \{1,\dots,d\}$. This perspective will not be addressed further in this work; we refer the reader to [92] for more details.

Another classical interpretation arises in the context of multilinear algebra where a tensor is regarded as a multilinear operator among the tensor product of linear spaces, in analogy with the matrix case. Indeed a matrix $A \in \mathbb{R}^{n_1 \times n_2}$ is both a real value array of order 2, it is an element of the linear space $\mathbb{R}^{n_1 \times n_2}$ and it describes the action of a linear operator from $\mathbb{R}^{n_2}$ to $\mathbb{R}^{n_1}$, once a basis of $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$ has been fixed. In this manuscript, we decide to denote by uppercase bold calligraphic letters the multilinear

operator $\boldsymbol{\mathcal{A}} : \mathbb{R}^{n_1} \otimes \cdots \otimes \mathbb{R}^{n_d} \to \mathbb{R}^{m_1} \otimes \cdots \otimes \mathbb{R}^{m_d}$, while the corresponding letter in uppercase bold, in this case $\mathbf{A} \in \Re^{(n_1 \times m_1) \times \cdots \times (n_d \times m_d)}$, stands for the multidimensional array representing the multilinear operator $\boldsymbol{\mathcal{A}}$ once the canonical basis of $\mathbb{R}^{n_1} \otimes \cdots \otimes \mathbb{R}^{n_d}$ and $\mathbb{R}^{m_1} \otimes \cdots \otimes \mathbb{R}^{m_d}$ are fixed. From now on, a tensor representing a multilinear operator among a tensor product of spaces will be referred to as *tensor operator*. Remark that the tensor operator $\mathbf{A}$ is a tensor of order $2d$ according to our definition, which acts on tensors of order d. This choice is aligned with the canonical linear algebra case, where a matrix is an array of dimension 2 representing the action of a linear operator over a vector expressed as an array of dimension 1.

1.2 Tensor basic operations

This section presents the basic tensor calculus operations used in the entire manuscript and the most common object manipulations. In the first part, we establish an order for performing tensor manipulations. The second part presents the operations among tensors, regarded as multidimensional arrays, by analogy with the corresponding ones defined for matrices or vectors. Remark that tensor operations and manipulations can be performed even when the tensor is expressed in a compact form. We will provide further details on how to perform certain operations or manipulations in compressed format, once the decompositions are presented.

1.2.1 Tensor reshaping

Many algorithms benefit or rely on manipulations of the multidimensional array representing tensors. In the classical matrix context, it is common to select from a given matrix $A \in \mathbb{R}^{n_1 \times n_2}$ a column $c_j \in \mathbb{R}^{n_1}$ or a row $r_i \in \mathbb{R}^{n_2}$, that is $r_i = A(i,:)$ and $c_j = A(:,j)$, where the symbol ':' stands for taking all the entries along the row or column mode, using the `python` or MATLAB notation. This operation of extracting from a 2-dimensional array a 1-dimensional array is generalized to the tensors by slices and fibres.

Definition 1.2.1. *Let $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a d order tensor, then $\mathbf{a}^{[i_k,k]}$ is a $(d-1)$ order tensor representing the i_k-*th slice along mode *k whose generic element writes*

$$\mathbf{a}^{[i_k,k]}(i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_d) = \mathbf{a}(i_1, \ldots, i_{k-1}, i_k, i_{k+1}, \ldots, i_d)$$

for every $i_j \in \{1, \ldots, n_j\}$ and $j \in \{1, \ldots, d\}$. This concept is easily generalized to multiple mode slices. In particular the $(i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_d)$ slice along modes $(1, \ldots, k-1, k+1, \ldots, d)$ is called mode k fibre*. If the order is $d = 3$, then the (j,k) slice along modes $(2,3)$ is said* column fibre*, while the (i,k) slice along modes $(1,3)$ is the* row fibre *and the (i,j) slice along modes $(1,2)$ is the* tube fibre*.*

Since multimode slices will not appear in this document, we do not introduce a specific notation, but we propose an example to present practically the idea of fibres and slices for order 3 tensors. The following example stresses that fibres can be seen as an analogue of matrix rows and columns

Example 1.2.1. Consider the tensor $\mathbf{a} \in \mathbb{R}^{3\times3\times2}$ depicted in Figure 1.1C. The first and second slices with respect to mode 3 are the (3×3) matrices

$$\mathbf{a}^{[1,3]} = \begin{bmatrix} 5 & 3 & 1 \\ 2 & 4 & 9 \\ 6 & 8 & 7 \end{bmatrix} \qquad \text{and} \qquad \mathbf{a}^{[2,3]} = \begin{bmatrix} 1 & 8 & 4 \\ 9 & 2 & 2 \\ 7 & 1 & 6 \end{bmatrix}.$$

The $(1,1)$ column fibre corresponds to the vector $[5,2,6]^\top$, which is also the first column of the slice $\mathbf{a}^{[1,3]}$, the $(1,1)$ row fibre corresponds to the vector $[5,3,1]^\top$, i.e., the first row of $\mathbf{a}^{[1,3]}$, and the $(1,1)$ tube fibre to the vector $[5,1]^\top$.

Another common practice in the tensor framework is reorganizing the tensors into vectors or matrices. To do so, the fibres along a specific mode are arranged into a vector, defining the *vectorization* of a tensor, or as columns of a matrix, constructing the *tensor matricization* or *unfolding*. For both the vectorization and the matricization, we follow the order proposed in [84].

Definition 1.2.2. *Let $\mathcal{N}_k = \{1,\dots,n_k\}$ be a subset of $\mathbb{N}$ for every $k \in \{1,\dots,d\}$, define the function $\phi : \mathcal{N}_1 \times \cdots \times \mathcal{N}_d \to \mathcal{N}$ such that*

$$\phi(i_1,\dots,i_d) = 1 + \sum_{\ell=1}^{d}(i_\ell - 1)m_\ell \qquad \textit{with} \qquad m_\ell = \prod_{h=1}^{\ell-1} n_h$$

The image of the function ϕ is equal to the set $\mathcal{N} = \{1,\dots,n\}$ where n is equal to the mode size product, that is $n = \prod_{h=1}^{d} n_h$. Henceforth the expression $\overline{i_1,\dots,i_d}$ denotes the image of $(i_1,\dots,i_d)$ through ϕ, i.e., $\overline{i_1,\dots,i_d} = \phi(i_1,\dots,i_d)$.
Let $\mathbf{a} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ be an order-d, its vectorization $vec(\mathbf{a}) \in \mathbb{R}^n$ with n is such that

$$vec(\mathbf{a})(j) = \mathbf{a}(i_1,\dots,i_d) \qquad \textit{with} \qquad j = \phi(i_1,\dots,i_d).$$

The vectorization is also used to reshape a tensor into a matrix, applying it to tensor slices.

Definition 1.2.3. *Let $\mathbf{a} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ be a d order tensor and let $n_{\neq k} = (n_1 \cdots n_d)/n_k$, then the $(n_k, n_{\neq k})$ matrix $A^{(k)}$ denotes the* mode k matricization or k unfolding *whose (i,j) element writes*

$$A^{(k)}(i,j) = \mathbf{a}(j_1,\dots,j_{k-1},i,j_{k+1},\dots,j_d)$$

for every $i \in \{1,\dots,n_k\}$ and $j = \overline{j_1,\dots,j_{k-1},j_{k+1},\dots,j_d}$. This definition is extended to multimode matricization, using the index linear combination for computing the row and column index consistently.

From now on, we denote the matricization of a tensor along a certain mode by the same letter in uppercase with the mode index between round brackets as superscript, as proposed in [36]. Notice that, generally speaking, the fibres arrangement order is not significant as long as it is consistent throughout the entire work. We use the previous examples to illustrate concretely the tensor vectorization and matricization.

Example 1.2.2. Consider the tensor $\mathbf{a} \in \mathbb{R}^{3\times3\times2}$ depicted in Figure 1.1C. The vectorization of $\mathbf{a}$ stacks vertically all the row fibres fixing (i_2, i_3) for increasing values of i_2 and i_3 in the given order. The vectorization of $\mathbf{a}$ writes with the row fibres as

$$\text{vec}(\mathbf{a}) = \begin{bmatrix} r_{(1,1)} \\ r_{(2,1)} \\ \vdots \\ r_{(3,2)} \end{bmatrix}$$

where $r_{(i_2,i_3)} \in \mathbb{R}^3$ represents the (i_2, i_3) row fibre, that is $r_{(i_2,i_3)} = \mathbf{a}(:, i_2, i_3)$ for $i_2 \in \{1, 2, 3\}$ and $i_3 \in \{1, 2\}$. The result of the vectorization is

$$\text{vec}(\mathbf{a})^\top = [5, 2, 6, 3, 4, 8, 1, 9, 7, 1, 9, 7, 8, 2, 1, 4, 2, 6].$$

The matricization of $\mathbf{a}$ with respect to mode 1 is $A^{(1)} \in \mathbb{R}^{3\times6}$, obtained stacking one next to the other all the row fibres, i.e.,

$$A^{(1)} = [r_{(1,1)},\, r_{(2,1)},\, \cdots,\, r_{(3,2)}]$$

with $r_{(i_2,i_3)} \in \mathbb{R}^3$ represents the (i_2, i_3) row fibre. The final result of the matricization with respect to mode 1 is

$$A^{(1)} = \begin{bmatrix} 5 & 3 & 1 & 1 & 8 & 4 \\ 2 & 4 & 9 & 9 & 2 & 2 \\ 6 & 8 & 7 & 7 & 1 & 6 \end{bmatrix}.$$

The tensor matricization is connected with the Kronecker product, another relevant operation in the tensor framework.

Definition 1.2.4. *The* Kronecker product *between matrices $A \in \mathbb{R}^{n_1\times n_2}$ and $B \in \mathbb{R}^{m_1\times m_2}$ is $A \otimes_K B$ a $(n_1m_1) \times (n_2m_2)$ matrix such that*

$$A \otimes_K B = \begin{bmatrix} A(1,1)B & \cdots & A(1,n_2)B \\ \vdots & \ddots & \vdots \\ A(n_1,1)B & \cdots & A(n_1,n_2)B \end{bmatrix}.$$

With an abuse of notation, the Kronecker product of two vectors $a \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ is the vector $a \otimes_{\text{K}} b \in \mathbb{R}^{nm}$ defined as

$$a \otimes_{\text{K}} b = \begin{bmatrix} a(1)b \\ \vdots \\ a(n)b \end{bmatrix}.$$

By construction the k-th element of $a \otimes_{\text{K}} b$ is equal to $a(i)b(j)$ with $k = j + m(i-1) = \overline{ij}$. From this last remark, it follows this tiny but useful lemma.

Lemma 1.2.3. *Given an elementary tensor* $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$*, expressed as*

$$\mathbf{a} = v_1 \otimes \cdots \otimes v_d$$

with $v_\ell \in \mathbb{R}^{n_\ell}$*, then its matricization with respect to mode* ℓ *writes*

$$A^{(\ell)} = v_\ell \otimes \Big(v_d \otimes_K \cdots \otimes_K v_{\ell+1} \otimes_K v_{\ell-1} \otimes_K \cdots \otimes_K v_1\Big)$$

for every $\ell \in \{1, \ldots, d\}$

Proof. Since this is a classical result in the tensor theory, we prove it just for the mode 1 matricization under the hypothesis $d = 3$. By construction the $(i_1, \overline{i_2 i_3})$-th element of $A^{(1)}$ is

$$A^{(1)}(i_1, \overline{i_2 i_3}) = v_1(i_1)v_2(i_2)v_3(i_3).$$

By construction $v_2(i_2)v_3(i_3)$ is the $(\overline{i_2 i_3})$-th element of $v_3 \otimes_{\mathrm{K}} v_2$, i.e.,

$$A^{(1)}(i_1, \overline{i_2 i_3}) = v_1(i_1)\Big(v_3 \otimes_{\mathrm{K}} v_2\Big)(\overline{i_3 i_2}).$$

and thanks to the definition of the tensor product, it follows

$$A^{(1)}(i_1, \overline{i_2 i_3}) = v_1(i_1)\Big(v_3 \otimes_{\mathrm{K}} v_2\Big)(\overline{i_3 i_2}) = \Big(v_1 \otimes (v_3 \otimes_{\mathrm{K}} v_2)\Big)(i_1, \overline{i_3 i_2}),$$

that is the thesis. □

Another property of the Kronecker product we use in the following sections is called *mixed-product property*. Let $A_i \in \mathbb{R}^{n_i \times m_i}$, $B_i \in \mathbb{R}^{m_i \times p_i}$ for $i \in \{1, 2\}$, then the Kronecker product of $(A_1 B_1)$ and $(A_2 B_2)$ is expressed as the matrix product of the Kronecker product of A_i with B_i, that is

$$(A_1 B_1) \otimes_{\mathrm{K}} (A_2 B_2) = (A_1 \otimes_{\mathrm{K}} A_2)(B_1 \otimes_{\mathrm{K}} B_2). \tag{1.1}$$

1.2.2 Tensor calculus

As previously mentioned, tensors in this document are thought of as multidimensional arrays, belonging to the space $\mathbb{R}^{n_1 \times \cdots \times n_d}$, which is a linear space. Consequently, the tensor sum and the product with real scalars are linearly defined component-wise.

The first operations presented involves tensors and matrices. Let $M_k \in \mathbb{R}^{m \times n_k}$ and $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_k \times \cdots \times n_d}$ be a matrix and a tensor respectively, then the *matrix-tensor product* between M_k and $\mathbf{a}$ along the mode k is denoted by $(M_k \times_k \mathbf{a}) \in \mathbb{R}^{n_1 \times \cdots \times m \times \cdots \times n_d}$, whose $(i_1, \ldots, j_k, \ldots, i_d)$-th element is

$$\Big(M_k \times_k \mathbf{a}\Big)(i_1, \ldots, j_k, \ldots, i_d) = \sum_{i_k=1}^{n_k} M_k(j_k, i_k)\mathbf{a}(i_1, \ldots, i_k, \ldots, i_d).$$

Given d matrices $M_k \in \mathbb{R}^{m_k \times n_k}$, then $(M_1, \dots, M_d)\mathbf{a}$ denotes the product of the tuple of matrices $(M_1, \dots, M_d)$ and the tensor $\mathbf{a}$, whose general element writes

$$\begin{aligned}\Big((M_1, \dots, M_d)\mathbf{a}\Big)(j_1, \dots, j_d) &= \Big(M_1 \times_1 \dots M_d \times_d \mathbf{a}\Big)(j_1, \dots, j_d)) \\ &= \sum_{i_1, \dots, i_d = 1}^{n_1, \dots, n_d} M_1(j_1, i_1) \dots M_d(j_d, i_d)\mathbf{a}(i_1, \dots, i_k, \dots, i_d)\end{aligned}$$

for $j_k \in \{1, \dots, n_k\}$ and $k \in \{1, \dots, d\}$. Remark that this operation is order-independent. The matricization of a tensor expressed as a matrix-tensor product satisfies the following lemma, which will be widely used in Chapter 4.

Lemma 1.2.4. *[83, Proposition 3.7] Let $\mathbf{a} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ be a tensor and let $M_k \in \mathbb{R}^{m_k \times n_k}$ be a matrix for every $k \in \{1, \dots, d\}$, then the matricization of $\mathbf{b} = (M_1, \dots, M_d)\mathbf{a}$ with respect to mode ℓ is*

$$B^{(\ell)} = M_\ell A^{(\ell)} (M_d \otimes_K \dots \otimes_K M_{\ell+1} \otimes_K M_{\ell-1} \otimes_K \dots \otimes_K M_1)^\top$$

for every $\ell \in \{1, \dots, d\}$.

In the tensor context, the *tensor contraction* generalizes the matrix-vector product. Let $\mathbf{A} \in \mathbb{R}^{(n_1 \times m_1) \times \dots \times (n_d \times m_d)}$ and $\mathbf{b} \in \mathbb{R}^{m_1 \times \dots \times m_d}$ be a tensor operator and a tensor, the *tensor contraction* of $\mathbf{A}$ and $\mathbf{b}$ is $(\mathbf{A} \bullet \mathbf{b}) \in \mathbb{R}^{n_1 \times \dots \times n_d}$ defined as

$$(\mathbf{A} \bullet \mathbf{b})(i_1, \dots i_d) = \sum_{j_1, \dots, j_d = 1}^{m_1, \dots, m_d} \mathbf{A}(i_1, j_1, \dots, i_d, j_d)\mathbf{b}(j_1, \dots, j_d).$$

The tensor contraction is defined for any tensor since any tensor can be regarded as a tensor operator. Therefore given two tensors $\mathbf{a} \in \mathbb{R}^{n_1 \times \dots \times p \times \dots \times n_{d_1}}$ and $\mathbf{b} \in \mathbb{R}^{m_1 \times \dots \times p \times \dots \times m_{d_2}}$ with just the h-th and k-th mode having the same size, then $(\mathbf{a}\ {}_h\bullet_k\ \mathbf{b})$ is their *tensor contraction* with respect to mode h and k respectively. If tensor $\mathbf{c} \in \mathbb{R}^{n_1 \times \dots \times n_{d_1} \times m_1 \times \dots \times m_{d_2}}$ is the order-$(d_1 + d_2 - 2)$ tensor from the contraction $(\mathbf{a}\ {}_h\bullet_k\ \mathbf{b})$, then its $(i_1, \dots, i_{h-1}, i_{h+1}, \dots, i_{d_1}, j_1 \dots, j_{h-1}, j_{h+1}, \dots, j_{d_2})$-th element is c such that

$$\begin{aligned}c &= (\mathbf{a}\ {}_h\bullet_k\ \mathbf{b})(i_1, \dots, i_{h-1}, i_{h+1}, \dots, i_{d_1}, j_1 \dots, j_{k-1}, j_{k+1}, \dots, j_{d_2}) \\ &= \sum_{\ell=1}^{p} \mathbf{a}(i_1, \dots, \ell, \dots, i_{d_1})\mathbf{b}(j_1, \dots, \ell, \dots, j_{d_2}).\end{aligned} \tag{1.2}$$

The contraction between tensors can be extended to more modes, replacing in Equation (1.2) the single index h and k by sets of indices. To shorten the notation, we omit the bullet symbol and the mode indices, when the modes to contract are clear from the context, for example given $\mathbf{a} \in \mathbb{R}^{n_1 \times p \times m_1}$ and $\mathbf{b} \in \mathbb{R}^{p \times n_2 \times m_2}$, then the contraction $\mathbf{a} \bullet \mathbf{b}$ is among the second mode of $\mathbf{a}$ with the first one of $\mathbf{b}$, since they are the only modes having the same dimension. The tensor contraction enables us to compute mulitilinear endomorphism powers, exactly as in the matrix framework the matrix-vector product is

used to compute the matrix powers representing the composition of the associated linear endomorphism. Let $\boldsymbol{\mathcal{A}} : \mathbb{R}^{n_1 \times \cdots \times n_d} \to \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a multilinear endomorphism and let the tensor $\mathbf{A} \in \mathbb{R}^{(n_1 \times n_1) \times \cdots \times (n_d \times n_d)}$ be its representation with respect to the canonical basis of $\mathbb{R}^{n_1 \times \cdots \times n_d}$. Then $\mathbf{B} \in \mathbb{R}^{(n_1 \times n_1) \times \cdots \times (n_d \times n_d)}$ is the tensor representing with respect to the canonical basis $\boldsymbol{\mathcal{A}}^2$, the composition of $\boldsymbol{\mathcal{A}}$ with itself, whose element $\mathbf{B}(i_1, j_1, \ldots, i_d, j_d)$ writes

$$\begin{aligned}\mathbf{B}(i_1, j_1, \ldots, i_d, j_d) &= \Big(\mathbf{A} \ {}_{\mathcal{H}_\mathrm{L}}\bullet_{\mathcal{H}_\mathrm{R}} \mathbf{A}\Big)(i_1, j_1, \ldots, i_d, j_d) \\ &= \sum_{k_1, \ldots, k_d = 1}^{n_1, \ldots, n_d} \mathbf{A}(i_1, k_1, \ldots, i_d, k_d)\mathbf{A}(k_1, j_1, \ldots, k_d, j_d)\end{aligned}$$

with $\mathcal{H}_\mathrm{L} = \{2, 4, \ldots, 2d\}$ and $\mathcal{H}_\mathrm{R} = \{1, 3, \ldots, 2d-1\}$. From this, we recursively obtain the tensor associated with $\boldsymbol{\mathcal{A}}^h$ for $h \in \mathbb{N}$. Remark that the contraction enables us to express also the matrix-tensor product since matrices are in fact order-2 tensors. Given $M_k \in \mathbb{R}^{m \times n_k}$ and $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_k \times \cdots \times n_d}$, then $(M_k \times_k \mathbf{a})$ the matrix-tensor product along mode k is expressed as a contraction of M_k and $\mathbf{a}$ as

$$(M_k \times_k \mathbf{a}) = M_k \ {}_2\bullet_k \mathbf{a}$$

by its definition.

As already stated in Section 1.1, the tensor space $\mathbb{R}^{n_1 \times \cdots \times n_d}$ is a linear space and it is usually endowed by an inner product, which extends the inner product defined for the simple linear space $\mathbb{R}^n$. Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be two order-d tensors, then $\langle \mathbf{a}, \mathbf{b} \rangle \in \mathbb{R}$ is their inner product defined as

$$\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i_1, \ldots, i_d = 1}^{n_1, \ldots, n_d} \mathbf{a}(i_1, \ldots, i_d)\mathbf{b}(i_1, \ldots, i_d).$$

Remark that by construction the inner product is also expressed as the tensor contraction over all modes, that is as $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a} \bullet \mathbf{b}$. As in the classical linear algebra framework, the tensor inner product on $\mathbb{R}^{n_1 \times \cdots \times n_d}$ induces a norm on the same space defined as

$$\|\mathbf{a}\| = \sqrt{\langle \mathbf{a}, \mathbf{a} \rangle}$$

for every $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. By construction, this norm extends the idea of the vector L2-norm and of the Frobenius matrix to tensors.

Since tensors represent also multilinear operators, it is worthwhile defining the 2-norm induced by the vector L2-norm. Let $\mathbf{A} \in \mathbb{R}^{(n_1 \times m_1) \times \cdots \times (n_d \times m_d)}$ be an order-$2d$ tensor representing a multilinear operator $\boldsymbol{\mathcal{A}}$ from $\mathbb{R}^{m_1 \times \cdots \times m_d}$ to $\mathbb{R}^{n_1 \times \cdots \times n_d}$ with respect to the canonical basis, then $\|\mathbf{A}\|_2$ is the 2-norm of $\boldsymbol{\mathcal{A}}$ defined as

$$\|\mathbf{A}\|_2 = \max\left\{ \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} : \mathbf{x} \in \mathbb{R}^{m_1 \times \cdots \times m_d} \setminus \{0\} \right\}.$$

1.3 Tensor decomposition

As pointed out in Section 1.1, after fixing a basis of $\mathbb{R}^{n_1} \otimes \cdots \otimes \mathbb{R}^{n_d}$, we can identify an element of the tensor linear space with the multidimensional array representing it. Let $\{e_{i_1}^{(1)} \otimes \cdots \otimes e_{i_d}^{(d)}\}$ be the canonical basis of $\mathbb{R}^{n_1} \otimes \cdots \otimes \mathbb{R}^{n_d}$, with an abuse of notation every tensor $\mathbf{a} \in \mathbb{R}^{n_1} \otimes \cdots \otimes \mathbb{R}^{n_d}$ is expressed as

$$\mathbf{a} = \sum_{i_1,\ldots,i_d=1}^{n_1,\ldots,n_d} \mathbf{a}(i_1, \ldots, i_d) e_{i_1}^{(1)} \otimes \cdots \otimes e_{i_d}^{(d)} \tag{1.3}$$

where $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is the multidimensional array storing its entries. The expression of tensors given in Equation (1.3) is usually referred to as *dense or canonical format*. This representation is affected by the so-called 'curse of dimensionality'. Indeed if storing one entry of an array costs one unit of memory, then storing $\mathbf{a}$ requests $\mathcal{O}(n^d)$ units of memory with $n = \max\{n_1, \ldots, n_d\}$, which scales exponentially with the order of the tensor. To face this significant practical limit, throughout the years, different compression techniques have been proposed. Each method can be regarded firstly as an exact way of decomposing a tensor and secondly as a procedure to obtain an approximation of it, which satisfies a certain criterion.

In the following sections, we describe two main decomposition and approximation techniques, the Tucker and the Tensor-Train one. For both methods, we present the theoretical and computational aspects. It is convenient to denote by $\mathbb{O}(n \times m)$ the set of matrices of size (n, m) with orthogonal columns, briefly referred by orthogonal matrix set.

1.3.1 Tucker decomposition

As clearly reconstructed in [84], the history of the Tucker decomposition dates back to 1963 when Tucker proposed it in [135], even if it was already in Hitchcock paper of 1927 [72]. It was further improved and more detailed described in [97, 136, 137]. It has been renamed many times over the years, as *Three-mode factor analysis*, *Three-mode factor PCA*, *n-mode PCA*, *n-mode SVD*, see [36, 80, 90, 137, 141].

Definition 1.3.1. *Let $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a d order tensor, its Tucker decomposition writes*

$$\mathbf{a} = \sum_{i_1=1}^{r_1} \cdots \sum_{i_d=1}^{r_d} \mathbf{c}(i_1, \ldots, i_d)\, u_{i_1}^{(1)} \otimes \cdots \otimes u_{i_d}^{(d)}$$

where the multidimensional array $\mathbf{c}$ is the core tensor*, $\{u_{i_1}^{(\ell)}, \ldots, u_{r_\ell}^{(\ell)}\}$ is an orthonormal basis of a $\mathbb{R}^{n_\ell}$ subspace of size r_ℓ, which is equal the rank of $A^{(\ell)}$, the matricization along mode ℓ of $\mathbf{a}$ for every $\ell \in \{1, \ldots, d\}$. We refer to the tuple $(r_1, \ldots, r_d)$ as the* multilinear rank *of $\mathbf{a}$, as stated in [36]. Compactly, the orthogonal Tucker decomposition is expressed as*

$$\mathbf{a} = (U_1, \ldots, U_d)\mathbf{c}$$

where the orthogonal matrix $U_\ell \in \mathbb{O}(n_\ell \times r_\ell)$ *has* $u_{i_\ell}^{(\ell)}$ *as* ℓ*-th column. The matrix* U_ℓ *is called* ℓ*-th* factor matrix *for every* $\ell \in \{1, \dots, d\}$.

The idea of the Tucker decomposition is expressing a tensor $\mathbf{a}$ of multilinear rank $(r_1, \dots, r_d)$ in an orthonormal basis $\{u_{i_1}^{(1)} \otimes \cdots \otimes u_{i_d}^{(d)}\}$ for $i_k \in \{1, \dots, r_k\}$ and $k \in \{1, \dots, d\}$. Consequently, if we denote by $\mathfrak{U}_k$ the subspace of dimension r_k of $\mathbb{R}^{n_k}$ spanned by $\{u_1^{(k)}, \dots, u_{r_k}^{(k)}\}$, then tensor $\mathbf{a}$ belongs to $\mathfrak{U}_1 \otimes \cdots \otimes \mathfrak{U}_d$.

Starting from this Tucker decomposition, we formulate an approximation problem as follows. Given a tensor $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, its best Tucker approximation with prescribed multilinear rank $(r_1, \dots, r_d)$ is the tensor $\mathbf{a}^*$ such that $\|\mathbf{a} - \mathbf{a}^*\|$ is minimal, i.e.,

$$\mathbf{a}^* = \underset{\substack{\operatorname{rank}(B^{(k)}) \leq r_k \\ k=1,\dots,d}}{\operatorname{argmin}} \|\mathbf{a} - \mathbf{b}\|$$

where $B^{(k)}$ is the k-th mode matricization of $\mathbf{b} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. If tensor $\mathbf{b}$ has multilinear rank $(r_1, \dots, r_d)$, then it is an element of the tensor space $\mathfrak{U}_1 \otimes \cdots \otimes \mathfrak{U}_d$ where $\mathfrak{U}_k$ is a subspace of $\mathbb{R}^{n_k}$ of size r_k for every $k \in \{1, \dots, d\}$. Thus, given a tensor of order d, the problem of finding the best approximation at a prescribed multilinear rank becomes a problem of searching d subspaces of prescribed dimensions, which minimize the projection error. Therefore the best Tucker approximation problem is a natural extension of the approximation problem characterising the Principal Component Analysis (PCA), because the unknowns are the spaces $\mathfrak{U}_k$ for every $k \in \{1, \dots, d\}$ under constraints of dimensions.

1.3.1.1 Computational aspects

While PCA approximation problem has an optimal solution, guaranteed by the Eckart–Young theorem [41] and provided by the Singular Value Decomposition(SVD), the Tucker approximation problem has not a known closed formula for the solution. In this section, we describe the most popular algorithms for computing the Tucker decomposition and approximation

Nowadays the most common algorithm for computing the Tucker decomposition is the High Order Singular Value Decomposition (HOSVD), proposed in [36] and displayed in Algorithm 1. Given an input tensor $\mathbf{a}$, of which the Tucker decomposition is requested, iteratively HOSVD matricizes it along mode i and computes its left singular vectors arranged in matrix $U_i \in \mathbb{O}(n_i \times r_i)$ where $r_i = \operatorname{rank}(A^{(i)})$, for $i \in \{1, \dots, d\}$, as in Algorithm 1 line 3. Thanks to the orthonormality of U_i, that is $U_i^\top U_i = \mathbb{I}_{r_i}$, from the definition of Tucker decomposition, it follows that the core tensor is actually given by $\mathbf{c} = (U_1^\top, \dots, U_d^\top)\mathbf{a}$, as computed in Algorithm 1 line 5. The factor matrices are the left singular matrices U_k for every $k \in \{1, \dots, d\}$.

To address the Tucker approximation problem, the authors of [36] proposed the Truncated HOSVD (T-HOSVD) method, given in Algorithm 2, a slightly different version of HOSVD. T-HOSVD takes the multilinear rank $(r_1, \dots, r_d)$ as input parameter together with the tensor $\mathbf{a}$ to approximate. In the truncated version of HOSVD, the full SVD is replaced by a Truncated SVD (T-SVD). Namely, at line 3 of Algorithm 2, we compute

Algorithm 1 $(U_1, \dots, U_d), \mathbf{c} = \text{HOSVD}(\mathbf{a})$

1: **input:** $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$
2: **for** $i = 1, \dots, d$ **do**
3: $\quad U_i, \Sigma_i, V_i = \texttt{SVD}\big(A^{(i)}\big)$ ▷ SVD of each matricization
4: **end for**
5: $\mathbf{c} = (U_1^\top, \dots, U_d^\top)\mathbf{a}$ ▷ compute the core tensor
6: **return:** $(U_1, \dots, U_d)$, $\mathbf{c}$

just the first r_i left singular vectors of $A^{(i)}$, instead of the full decomposition returned in line 3 of Algorithm 1. All the remaining steps of Algorithm 1 and 2 are exactly equal. The approximation provided by T-HOSVD is not the optimal one. In fact, if we denote by $\hat{\mathbf{a}}$ the approximation provided by T-HOSVD of $\mathbf{a}$ at multilinear rank $(r_1, \dots, r_d)$, then the approximation error is bounded by

$$\|\mathbf{a} - \hat{\mathbf{a}}\|^2 \leq \sum_{i=1}^{d} \left\|\tilde{\Sigma}_i\right\|^2 \qquad \text{where} \qquad \tilde{\Sigma}_i(j,j) = \begin{cases} 0 & j \leq r_i \\ \Sigma_i(j,j) & j > r_i \end{cases}$$

where Σ_i is the diagonal singular values matrix of $A^{(i)}$ for every $i \in \{1, \dots, d\}$.

Algorithm 2 $(\hat{U}_1, \dots, \hat{U}_d), \hat{\mathbf{c}} = \text{T-HOSVD}(\mathbf{a}, r)$

1: **input:** $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, $r \in \mathbb{N}^d$
2: **for** $i = 1, \dots, d$ **do**
3: $\quad \hat{U}_i, \hat{\Sigma}_i, \hat{V}_i = \texttt{T-SVD}\big(A^{(i)}, r(i)\big)$ ▷ truncated SVD of each matricization
4: **end for**
5: $\hat{\mathbf{c}} = (\hat{U}_1^\top, \dots, \hat{U}_d^\top)\mathbf{a}$ ▷ compute the core tensor
6: **return:** $(\hat{U}_1, \dots, \hat{U}_d)$, $\hat{\mathbf{c}}$

We want to mention, without entering in details, the Sequentially Truncated HOSVD (ST-HOSVD), proposed in [140] to compute the Tucker approximation given a multilinear rank. Differently from T-HOSVD, ST-HOSVD initializes the tensor core by the input one. Then it iteratively updates the tensor core $\hat{\mathbf{c}}$ multiplying it by the transposed truncated left singular vector matrix $\hat{U}_i^\top$ along mode i, that is, $\hat{\mathbf{c}}$ is updated by $\hat{U}_i^\top \times_i \hat{\mathbf{c}}$. This choice reduces iteratively the size of the matrix decomposed by the truncated SVD, decreasing the computational costs. However not always the approximation error of ST-HOSVD is lower than the T-HOSVD. Moreover, the ST-HOSVD approximation error depends on the matricization order, while the T-HOSVD one is independent. We refer the reader to [140] for a complete ST-HOSVD analysis and comparison with other approximation techniques.

The last method for computing the Tucker approximation is called High Order Orthogonal Iterations (HOOI), proposed in [37]. The HOOI is an Alternating Least Squares (ALS) bases algorithm, which outcompeted the other ALS algorithms proposed in [90]

Algorithm 3 $(\hat{U}_1, \dots, \hat{U}_d), \hat{\mathbf{c}} = \text{HOOI}(\mathbf{a}, r, \delta)$

input: $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, $r \in \mathbb{N}^d$, $\delta \in \mathbb{R}_+$
$(\hat{U}_1, \dots, \hat{U}_d), \mathbf{c} = \texttt{HOSVD}(\mathbf{a})$ ▷ initialize the core tensor and the factors by the HOSVD
while $\texttt{error} > \delta$ **do**
  for $i = 1, \dots, d$ **do**
  ▷ compute the core tensor without the i-th factor
    $\mathbf{c} = (\hat{U}_1^\top, \dots, \hat{U}_{i-1}^\top, \mathbb{I}_{n_i}, \hat{U}_{i+1}^\top, \dots, \hat{U}_d^\top)\mathbf{a}$
  ▷ compute the truncated SVD of the core tensor without i-th factor
    $\hat{U}_i, \hat{\Sigma}_i, \hat{V}_i = \texttt{T-SVD}(C^{(i)}, r(i))$
    $\texttt{error} = \dfrac{\|\mathbf{a} - \hat{\mathbf{a}}\|}{\|\mathbf{a}\|}$ where $\hat{\mathbf{a}} = (\hat{U}_1, \dots, \hat{U}_d)\mathbf{c}$ ▷ check the relative error
  end for
end while
$\hat{\mathbf{c}} = (\hat{U}_1^\top, \dots, \hat{U}_d^\top)\mathbf{a}$
return: $(\hat{U}_1, \dots, \hat{U}_d)$, $\hat{\mathbf{c}}$

and in [80]. The ALS part aims to minimize the projection error along each tensor mode. Indeed, as previously highlighted, finding the best Tucker approximation of $\mathbf{a}$ at multilinear rank $(r_1, \dots, r_d)$ can be reformulated as finding the subspaces $\mathfrak{U}_k$ of $\mathbb{R}^{n_k}$ of dimension r_k whose tensor product minimizes the projection error, that is $\left\|\mathbf{a} - (\hat{U}_1, \dots, \hat{U}_d)\hat{\mathbf{c}}\right\|$ where $\hat{U}_k \in \mathbb{O}(n_k \times r_k)$ columns form an orthogonal basis of $\mathfrak{U}_k$. Elaborating the square of the projection error, we get that

$$\left\|\mathbf{a} - (\hat{U}_1, \dots, \hat{U}_d)\hat{\mathbf{c}}\right\|^2 = \|\mathbf{a}\|^2 - \left\|(\hat{U}_1^\top, \dots, \hat{U}_d^\top)\mathbf{a}\right\|^2,$$

we refer the reader to [37, 84] for a complete sequence of computations. Since the norm of $\mathbf{a}$ is fixed, to reduce the projection error we have to maximize the term $\left\|(\hat{U}_1^\top, \dots, \hat{U}_d^\top)\mathbf{a}\right\|$. In particular, for each mode i the HOOI algorithm solves the maximization problem

$$\hat{U}_i = \underset{V_i \in \mathbb{O}(n_i \times r_i)}{\operatorname{argmax}} \left\|(\hat{U}_1^\top, \dots, \hat{U}_{i-1}^\top, V_i^\top, \hat{U}_{i+1}^\top, \dots, \hat{U}_d^\top)\mathbf{a}\right\|.$$

Thanks to Corollary 1.2.4, this last equation cab also be written as

$$\hat{U}_i = \underset{V_i \in \mathbb{O}(n_i \times r_i)}{\operatorname{argmax}} \left\|V_i^\top A^{(i)}(\hat{U}_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} \hat{U}_{i+1} \otimes_{\mathrm{K}} \hat{U}_{i-1} \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} \hat{U}_1)\right\|. \tag{1.4}$$

and it is solved by the first r_i left singular vectors from the SVD of $A^{(i)}$ for every $i \in \{1, \dots, d\}$. The local maximization problem stated in Equation (1.4) is repeatedly updated and solved for every mode i, until the projection error $\left\|\mathbf{a} - (\hat{U}_1, \dots, \hat{U}_d)\hat{\mathbf{c}}\right\|$ gets smaller than the given tolerance threshold $\delta \in \mathbb{R}_+$. For simplicity in Algorithm 3, we sketch the HOOI version proposed in [84].

1.3.1.2 Memory requirement

At the beginning of Section 1.3, we pointed out that storing an order-d tensor in dense format requests $\mathcal{O}(n^d)$ of memory units where n is the maximal mode dimension. If the same tensor is expressed in the Tucker format by a tuple of d matrices and by a core tensor of order d, then the storing cost is $\mathcal{O}(dnr + r^d)$ units of memory where r is the maximal multilinear rank component. Indeed storing the entire core tensor needs $\mathcal{O}(r^d)$ units of memory, while storing one of the d factor matrix costs $\mathcal{O}(nr)$ units of memory. Remark that the Tucker approximation does not really overcome the curse of dimensionality, since the storing costs of the core tensor still grow exponentially with the order of the tensor. Therefore other compression techniques have been presented as the Hierarchical Tucker [54] or the more promising Tensor-Train [111], presented in details in the following section.

1.3.2 Tensor Train decomposition

The Tensor-Train (TT) decomposition is a representation that overcomes the curse of dimensionality, since the memory requirement to store a tensor in TT-format grows linearly with the order d. It was introduced in [110], almost at the same time of the Hierarchical Tucker decomposition [54], as a mathematical formulation of the matrix product states [2, 48] concept used in quantum physics. The TT-decomposition was described in detail in [111]. The key idea of Tensor Train is expressing a tensor of order d as the contraction of d tensors of order 3, which leads to a significant saving in memory, as illustrated after.

Definition 1.3.2. *Let* $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ *be an order-d tensor, its Tensor-Train decomposition writes*

$$\mathbf{a} = \underline{\mathbf{a}}_1 \underline{\mathbf{a}}_2 \cdots \underline{\mathbf{a}}_d,$$

where $\underline{\mathbf{a}}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ *is called k-th* TT-core *for* $k \in \{1, \ldots, d\}$*, with* $r_0 = r_d = 1$*. The value* r_k *is the* k-th TT-rank*.*

Notice that $\underline{\mathbf{a}}_1 \in \mathbb{R}^{r_0 \times n_1 \times r_1}$ and $\underline{\mathbf{a}}_d \in \mathbb{R}^{r_{d-1} \times n_d \times r_d}$ reduce essentially to matrices, but for the notation consistency we represent them as tensor. Henceforth the k-th TT-core of a tensor is denoted by the same bold letter underlined with a subscript k. If $\mathbf{a}$ is expressed in TT-format, then its $(i_1, \ldots, i_d)$-th element writes

$$\mathbf{a}(i_1, \ldots, i_d) = \sum_{j_0, \ldots, j_d = 1}^{r_0, \ldots, r_d} \underline{\mathbf{a}}_1(j_0, i_1, j_1) \underline{\mathbf{a}}_2(j_1, i_2, j_2) \ldots \underline{\mathbf{a}}_{d-1}(j_{d-2}, i_{d-1}, j_{d-1}) \underline{\mathbf{a}}_d(j_{d-1}, i_d, j_d).$$

Remark that the index j_0 and j_d play no role since they are always equal to 1; they are added for completeness. Given an index $i_k \in \{1, \ldots, n_k\}$, we denote the i_k-th matrix slice of $\underline{\mathbf{a}}_k$ with respect to mode 2 by $\underline{A}_k(i_k)$, i.e., $\underline{A}_k(i_k) = \underline{\mathbf{a}}_k^{[i_k,2]}$. Then each element of tensor $\mathbf{a}$ expressed in TT-format is given by the product of d matrices, i.e.,

$$\mathbf{a}(i_1, \ldots, i_d) = \underline{A}_1(i_1) \cdots \underline{A}_d(i_d)$$

with $\underline{A}_k(i_k) \in \mathbb{R}^{r_{k-1} \times r_k}$ for every $i_k \in \{1, \dots, n_k\}$ and $k \in \{2, \dots, d-1\}$, while $\underline{A}_1(i_1) \in \mathbb{R}^{1 \times r_1}$ and $\underline{A}_d(i_d) \in \mathbb{R}^{r_{d-1} \times 1}$. Remark that $\underline{A}_1(i_1)$ and $\underline{A}_d(i_d)$ are vectors, but as before to have a homogeneous notation they can be written as matrices with a single row or column.

The arithmetic operations among tensors in TT-format are performed in a particular way, benefiting from the chosen formalism. Indeed, given two tensors in TT-format $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, let $\underline{\mathbf{a}}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ and $\underline{\mathbf{b}}_k \in \mathbb{R}^{s_{k-1} \times n_k \times s_k}$ be their k-th TT-core respectively, the k-th TT-core of $\mathbf{c} = (\mathbf{a} + \mathbf{b})$ is $\underline{\mathbf{c}}_k \in \mathbb{R}^{(r_{k-1}+s_{k-1}) \times n_k \times (r_k+s_k)}$ defined block-wise as

$$\underline{C}_k(i_k) = \begin{bmatrix} \underline{A}_k(i_k) & \mathbb{0} \\ \mathbb{0} & \underline{B}_k(i_k) \end{bmatrix} \tag{1.5}$$

where $\mathbb{0}$ denotes the matrix filled with zeros of the appropriate size, for every $i_k \in \{1, \dots, n_k\}$ and $k \in \{2, \dots, d-1\}$. The first and last TT-core of $\mathbf{c} = (\mathbf{a} + \mathbf{b})$ are $\underline{\mathbf{c}}_1 \in \mathbb{R}^{1 \times n_1 \times (r_1+s_1)}$ and $\underline{\mathbf{c}}_d \in \mathbb{R}^{(r_{d-1}+s_{d-1}) \times n_d \times 1}$, that can be written as

$$\underline{C}_1(i_1) = \begin{bmatrix} \underline{A}_1(i_1) & \underline{B}_1(i_1) \end{bmatrix} \qquad \text{and} \qquad \underline{C}_d(i_d) = \begin{bmatrix} \underline{A}_d(i_d) \\ \underline{B}_d(i_d) \end{bmatrix}$$

with $i_1 \in \{1, \dots, n_1\}$ and $i_d \in \{1, \dots, n_d\}$. Finally, we verify that by construction the $(i_1, \dots, i_d)$-th element of $\mathbf{c}$ is equal to the sum of the corresponding element of ($\mathbf{a}$ and $\mathbf{b}$), that is

$$\begin{aligned} \mathbf{c}(i_1, \dots, i_d) &= \begin{bmatrix} \underline{A}_1(i_1) & \underline{B}_1(i_1) \end{bmatrix} \begin{bmatrix} \underline{A}_2(i_2) & \mathbb{0} \\ \mathbb{0} & \underline{B}_2(i_2) \end{bmatrix} \cdots\cdots \begin{bmatrix} \underline{A}_d(i_d) \\ \underline{B}_d(i_d) \end{bmatrix} \\ &= \underline{A}_1(i_1) \cdots \underline{A}_d(i_d) + \underline{B}_1(i_1) \cdots \underline{B}_d(i_d) \\ &= \mathbf{a}(i_1, \dots, i_d) + \mathbf{b}(i_1, \dots, i_d) \\ &= \Big(\mathbf{a} + \mathbf{b}\Big)(i_1, \dots, i_d). \end{aligned}$$

Similarly let $\lambda \in \mathbb{R}$ be a scalar and $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a tensor in TT-format, then the tensor $(\lambda \mathbf{a})$ has the k-th TT-core equal to the corresponding one of $\mathbf{a}$ for $k \in \{2, \dots, d\}$, while the 1-st TT-core is equal to the first TT-core of $\mathbf{a}$ multiplied by λ. Thus, the $(i_1, \dots, i_d)$-th element of $(\lambda \mathbf{a})$ gets

$$\begin{aligned} \Big(\lambda \mathbf{a}\Big)(i_1, \dots, i_d) &= \Big(\lambda \underline{A}_1(i_1)\Big) \underline{A}_2(i_2) \cdots \underline{A}_d(i_d) \\ &= \lambda \Big(\underline{A}_1(i_1) \cdots \underline{A}_d(i_d)\Big) \\ &= \lambda \mathbf{a}(i_1, \dots, i_d). \end{aligned}$$

We report now the formulas provided in [108] for other operations, that not necessary correspond to the optimal algorithmic implementation. Also, the computation of the inner product benefits from the TT-formalism. Given $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ two tensors of order d in TT-format, their inner product writes

$$\begin{aligned} \langle \mathbf{a}, \mathbf{b} \rangle &= \sum_{i_1, \dots, i_d = 1}^{n_1, \dots, n_d} \mathbf{a}(i_1, \dots, i_d) \mathbf{b}(i_1, \dots, i_d) \\ &= \sum_{i_1, \dots, i_d = 1}^{n_1, \dots, n_d} \Big(\underline{A}_1(i_1) \cdots \underline{A}_d(i_d)\Big) \Big(\underline{B}_1(i_1) \cdots \underline{B}_d(i_d)\Big). \end{aligned}$$

If the product of the TT-cores of $\mathbf{a}$ and $\mathbf{b}$ of this last equation is regarded as a degenerated case of Kronecker product among matrices of size $(1,1)$, that are scalars, then we get

$$\begin{aligned}\langle \mathbf{a}, \mathbf{b}\rangle &= \sum_{i_1,\dots,i_d=1}^{n_1,\dots,n_d} \Big(\underline{A}_1(i_1)\cdots\underline{A}_d(i_d)\Big)\Big(\underline{B}_1(i_1)\cdots\underline{B}_d(i_d)\Big) \\ &= \sum_{i_1,\dots,i_d=1}^{n_1,\dots,n_d} \Big(\underline{A}_1(i_1)\cdots\underline{A}_d(i_d)\Big) \otimes_{\mathrm{K}} \Big(\underline{B}_1(i_1)\cdots\underline{B}_d(i_d)\Big).\end{aligned}$$

Applying the mixed-product property of the Kronecker product, stated in Equation (1.1) in the last equation, it becomes

$$\begin{aligned}\langle \mathbf{a}, \mathbf{b}\rangle &= \sum_{i_1,\dots,i_d=1}^{n_1,\dots,n_d} \Big(\underline{A}_1(i_1)\cdots\underline{A}_d(i_d)\Big) \otimes_{\mathrm{K}} \Big(\underline{B}_1(i_1)\cdots\underline{B}_d(i_d)\Big) \\ &= \sum_{i_1,\dots,i_d=1}^{n_1,\dots,n_d} \Big(\underline{A}_1(i_1) \otimes_{\mathrm{K}} \underline{B}_1(i_1)\Big)\cdots\Big(\underline{A}_d(i_d) \otimes_{\mathrm{K}} \underline{B}_d(i_d)\Big)\end{aligned}.$$

To compute the scalar product of two tensors in TT-format we compute the Kronecker product of their TT-cores, multiply them and sum over all the indexes of each mode.

The TT-formalism enables us to express compactly also tensor operators between tensor spaces.

Definition 1.3.3. *Let* $\mathbf{A} \in \mathbb{R}^{(n_1\times m_1)\times\cdots\times(n_d\times m_d)}$ *be a tensor operator, its Tensor-Train decomposition writes*

$$\mathbf{A} = \underline{\mathbf{A}}_1 \cdots \underline{\mathbf{A}}_d,$$

where $\underline{\mathbf{A}}_k \in \mathbb{R}^{r_{k-1}\times n_k\times m_k\times r_k}$*, is its* k*-th* TT-core*, with* $r_0 = r_d = 1$*. Henceforth we call* TT-matrix *a tensor in TT-format representing a tensor operator; while we refer by* TT-vector *to a tensor regarded as an element of a linear tensor space.*

The $(i_1, j_1, \dots, i_d, j_d)$-th element of $\mathbf{A}$ a TT-matrix writes

$$\mathbf{A}(i_1, j_1, \dots, i_d, j_d) = \sum_{h_0,\dots,h_d=1}^{r_0,\dots,r_d} \underline{\mathbf{A}}_1(h_0, i_1, j_1, h_1)\cdots\underline{\mathbf{A}}_d(h_{d-1}, i_d, j_d, h_d).$$

Let $\underline{A}_k(i_k, j_k) \in \mathbb{R}^{r_{k-1}\times r_k}$ be the (i_k, j_k)-th slice with respect to mode $(2,3)$ of $\underline{\mathbf{A}}_k$ for every $i_k, \in \{1,\dots,n_k\}$, $j_k \in \{1,\dots,m_k\}$ and $k \in \{1,\dots,d\}$. Then the last equation is equivalently expressed as

$$\mathbf{A}(i_1, j_1, \dots, i_d, j_d) = \underline{A}_1(i_1, j_1)\cdots\underline{A}_d(i_d, j_d).$$

As remarked in Section 1.2, the inner product is the contraction over all the indexes of every mode of two tensors. Thus, as the inner product can be computed smartly if the tensors are expressed in TT-format, also the contraction among a TT-matrix and a TT-vector, that is the action of a multilinear operator over a tensor in the domain, benefits

from the TT-format. Let $\mathbf{A} \in \mathbb{R}^{(n_1 \times m_1) \times \cdots \times (n_d \times m_d)}$ be a TT-matrix and $\mathbf{b} \in \mathbb{R}^{m_1 \times \cdots \times m_d}$ be a TT-vector, the $(i_1, \ldots, i_d)$-th element of their contraction writes

$$\begin{aligned}\Big(\mathbf{Ab}\Big)(i_1, \ldots, i_d) &= \sum_{j_1, \ldots, j_d = 1}^{m_1, \ldots, m_d} \mathbf{A}(i_1, j_1, \ldots, i_d, j_d)\mathbf{b}(j_1, \ldots, j_d) \\ &= \sum_{j_1, \ldots, j_d = 1}^{m_1, \cdots, m_d} \Big(\underline{A}_1(i_1, j_1) \cdots \underline{A}_d(i_d, j_d)\Big)\Big(\underline{B}_1(j_1) \cdots \underline{B}_d(j_d)\Big).\end{aligned}$$

As in the case of the inner product, if this last product is regarded as a degenerate Kronecker product, then the mixed-product property stated in (1.1) applies and leads to

$$\begin{aligned}\Big(\mathbf{Ab}\Big)(i_1, \ldots, i_d) &= \sum_{j_1, \ldots, j_d = 1}^{m_1, \ldots, m_d} \Big(\underline{A}_1(i_1, j_1) \cdots \underline{A}_d(i_d, j_d)\Big)\Big(\underline{B}_1(j_1) \cdots \underline{B}_d(j_d)\Big) \\ &= \sum_{j_1, \ldots, j_d = 1}^{m_1, \ldots, m_d} \Big(\underline{A}_1(i_1, j_1) \cdots \underline{A}_d(i_d, j_d)\Big) \otimes_{\mathrm{K}} \Big(\underline{B}_1(j_1) \cdots \underline{B}_d(j_d)\Big) \\ &= \sum_{j_1, \ldots, j_d = 1}^{m_1, \ldots, m_d} \Big(\underline{A}_1(i_1, j_1) \otimes_{\mathrm{K}} \underline{B}_1(j_1)\Big) \cdots \Big(\underline{A}_d(i_d, j_d) \otimes_{\mathrm{K}} \underline{B}_d(j_d)\Big) \\ &= \Big(\sum_{j_1 = 1}^{m_1} \underline{A}_1(i_1, j_1) \otimes_{\mathrm{K}} \underline{B}_1(j_1)\Big) \cdots \Big(\sum_{j_d = 1}^{m_d} \underline{A}_d(i_d, j_d) \otimes_{\mathrm{K}} \underline{B}_d(j_d)\Big).\end{aligned}$$

Said differently, the k-th TT-core of $\mathbf{c} = (\mathbf{Ab}) \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is $\underline{\mathbf{c}}_k \in \mathbb{R}^{(r_{k-1}s_{k-1}) \times n_k \times (r_k s_k)}$ such that for every $i_k \in \{1, \ldots, n_k\}$

$$\underline{C}_k(i_k) = \sum_{j_k = 1}^{m_k} \underline{A}_k(i_k, j_k) \otimes_{\mathrm{K}} \underline{B}_k(j_k) \tag{1.6}$$

for every $k \in \{1, \ldots, d\}$.

1.3.2.1 Memory requirement

The purpose of the TT-format declared by their authors in [110, 111] is getting rid of the curse of dimensionality, which affects other decomposition, for example, the Tucker one as underlined in Section 1.3.1.2. Given a tensor $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, the storage cost in TT-format is $\mathcal{O}(dnr^2)$, since we need to store d TT-cores requesting $\mathcal{O}(r^2 n)$ units of memory, under the assumptions that n and r as the maximum of the mode size and of the TT-ranks respectively. Similarly storing a tensor operator requests $\mathcal{O}(dmnr^2)$ memory units, where n and m are the maxima of the mode size of the domain and the image tensor space, while r is the maximum of the TT-ranks. In this case the memory footprint growths linearly with the tensor order, but to keep this formalism advantageous the value r has to stay bounded and small. Since the TT-rank determines the benefit in storing a tensor in TT-format, usually knowing the maximal TT-rank is sufficient to get an idea of the benefit. However, to be more accurate, we introduce the compression ratio measure.

Let $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a tensor in TT-format, then the compression ratio is the ratio among the storage cost of $\mathbf{a}$ in TT-format over the dense format storage cost, i.e.,

$$\frac{\sum_{i=1}^{d} r_{i-1} n_i r_i}{\prod_{j=1}^{d} n_j} \tag{1.7}$$

where r_i is the i-th TT-rank of $\mathbf{a}$.

A first drawback of the TT-format appears with the addition of two TT-tensors. Indeed given two TT-tensors $\mathbf{a}$ and $\mathbf{b}$ with k-th TT-rank r_k and s_k respectively, then the k-th TT-rank of $(\mathbf{a} + \mathbf{b})$ is equal to $r_k + s_k$, as shown in Equation (1.5). Consequently, when two tensors in TT-format are summed, the TT-ranks of their sum grow as the sum of the TT-ranks. In particular if $(2\mathbf{a})$ is computed as $(\mathbf{a} + \mathbf{a})$, then its TT-ranks are the double of the TT-ranks of the tensor obtained multiplying $\mathbf{a}$ by 2. Similarly when a TT-matrix $\mathbf{A}$ is contracted with a TT-vector $\mathbf{b}$, then their TT-ranks are multiplied. If r_k and s_k are the k-th TT-rank of $\mathbf{A}$ and $\mathbf{b}$, then the k-th TT-rank of $(\mathbf{Ab})$ is $(r_k s_k)$, as highlighted in Equation (1.6). In the following section dedicated to computation aspects, we discuss a solution to address this issue.

1.3.2.2 Tensor-Train compression

The TT-rank growth is a crucial point in the implementation of algorithms using TT-tensors. It may lead to memory deficiencies and prevent the calculation to complete. To address this issue, a rounding algorithm to reduce the TT-rank was introduced in [111], together with the algorithm that converts a dense format tensor into TT-format. We present firstly this latter one, sketched in Algorithm 4. To convert a dense tensor $\mathbf{a}$ in TT-format we prescribe a representation accuracy threshold $\delta \in \mathbb{R}_+$ which bounds the relative error, i.e.,

$$\frac{\|\mathbf{a} - \tilde{\mathbf{a}}\|}{\|\mathbf{a}\|} \leq \delta$$

where $\tilde{\mathbf{a}}$ is the representation of $\mathbf{a}$ in TT-format. To ensure the relative error accuracy, we define δ' scaling δ by the norm of the input tensor and the problem order, as

$$\delta' = \delta \frac{||\mathbf{a}||}{\sqrt{d-1}}$$

As first step, the input tensor $\mathbf{a}$ is matricized along the first mode, then $A^{(1)}$ of size $(n_1, n_2 \cdots n_d)$ is decomposed with the truncated SVD at accuracy δ'. The left orthogonal matrix $\hat{U}_1$ is reshaped into the first TT-core to have size $(1, n_1, r_1)$ where r_1 is the number of columns of $\hat{U}_1$, see line 7 of Algorithm 4. We define matrix $C_2 \in \mathbb{R}^{r_1 \times (n_2 \cdots n_d)}$ as the product of a diagonal matrix with the selected singular values times the associated right orthogonal matrix. Finally C_2 is reshaped to have size $(r_1 n_2, n_3 \cdots n_d)$. Then iteratively matrix C_i is decomposed with the truncated SVD, its left orthogonal matrix $\hat{U}_i$ is reshaped to become the i-th TT-core; the product of the singular value diagonal matrix $\hat{\Sigma}_i$ and the right orthogonal matrix $\hat{V}_i^\top$ defines the new matrix C_{i+1}, which is reshaped to have $(r_i n_{i+1})$ rows and $(n_{i+2} \cdots n_d)$ columns.

Algorithm 4 $\underline{\mathbf{a}}_1, \dots, \underline{\mathbf{a}}_d = \mathrm{TT}(\mathbf{a}, \delta)$

1: **input:** $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ in dense format, $\delta \in \mathbb{R}_+$
2: $\delta' = \frac{\delta}{\sqrt{d-1}} \|\mathbf{a}\|$ ▷ scale the accuracy δ
3: $r_0 = 1$ and $p = (n_2 \cdots n_d)$
4: $C_1 = \texttt{matricize}\big(\mathbf{a}, \texttt{mode} = 1\big)$ ▷ initialize the first TT-core
5: **for** $i = 1, \dots, d-1$ **do**
6: $\quad \hat{U}_i, \hat{\Sigma}_i, \hat{V}_i = \texttt{T-SVD}(C_i, \delta')$ ▷ compute the truncated SVD
▷ define the i-th TT-core
7: $\quad \underline{\mathbf{a}}_i = \texttt{reshape}(\hat{U}_i, [r_{i-1},\, n_i,\, r_i])$ where r_i is the number of columns of $\hat{U}_i$
8: $\quad C = \hat{\Sigma}_i \hat{V}_i^\top$
9: $\quad p = p / n_{i+1}$
▷ move the singular values and the right orthogonal basis on the right, to compute the next TT-core
10: $\quad C_{i+1} = \texttt{reshape}(C, [r_i n_{i+1},\, p])$
11: **end for**
12: $\underline{\mathbf{a}}_d = C$ ▷ define the last TT-core
13: **return:** $\underline{\mathbf{a}}_1, \dots, \underline{\mathbf{a}}_d$

As mentioned in Section 1.3.2.1, many basic operations among tensors in TT-format induce growth in the TT-ranks, reducing the storing benefit of this formalism. To address this flaw, the *TT-rounding* algorithm in [111] it was introduced a , which takes in input a tensor $\mathbf{a}$ in TT-format and an accuracy threshold $\delta \in \mathbb{R}_+$ and returns $\hat{\mathbf{a}}$ a tensor in TT-format such that

$$\frac{\|\mathbf{a} - \hat{\mathbf{a}}\|}{\|\mathbf{a}\|} \leq \delta.$$

As in Algorithm 4, the first step is scaling the rounding accuracy by the square root of the order minus one and multiplying it by the norm of the input tensor. Then, from the last to the second, each TT-core is matricized along mode 1, transposed and factorized through the QR algorithm, see Algorithm 5, line 4. The row orthogonal factor $Q_i^\top$ is reshaped to define the i-th TT-core $\underline{\mathbf{a}}_i$, while the R factor is used to update the $(i-1)$-th TT-core $\underline{\mathbf{a}}_{i-1}$, see line 7 of Algorithm 5. Indeed, from the QR-factorization theory, the columns of Q_i span the same space, that is, carry the same information, as the columns of $\underline{\mathbf{a}}_i$ matricized along mode 1 and transposed. Once completed this loop, usually referred to as the right-to-left orthogonalization loop, we proceed with the compression of the TT-cores. From the first to the second-last, each TT-core is matricized along mode 3, transposed to have $(r_{i-1} n_i)$ rows and decomposed with the truncated SVD given accuracy δ', as depicted in line 11 and 12 of Algorithm 5. Finally the left orthogonal matrix $\hat{U}_i$ from the truncated SVD is reshaped to define the new truncated i-th TT-core, while the singular value diagonal matrix times the right orthogonal one $(\hat{\Sigma}_i \hat{V}_i^\top)$ are used to update the successive $(i+1)$-th TT-core, see lines 13 and 14 of Algorithm 5. As stated in [111], given an input tensor $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, the computational cost, in terms of floating point operations, of the TT-rounding method, described in Algorithm 5 is $\mathcal{O}(dnr^3)$, where n

and r are the maximal mode size and TT-rank respectively. Indeed, the cost of each of the d QR-factorization and each truncated SVD is estimated as $\mathcal{O}(nr^3)$ float point operations, since both are performed over a matrix of size (nr, r) (or its transposed).

To estimate the benefit of applying the TT-rounding over an input TT-vector, we introduce the compression gain measure. Let $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a TT-vector with i-th TT-rank equal to r_i, if $\hat{\mathbf{a}}$ is its approximation obtained from the TT-rounding at accuracy $\delta \in \mathbb{R}_+$ with s_i the i-th TT-rank, then the compression gain of the TT-rounding is equal to the ratio of the compression ratio, i.e.,

$$q = \frac{\sum_{i=1}^{d} r_{i-1} n_i r_i}{\sum_{j=1}^{d} s_{j-1} n_j s_j}. \tag{1.8}$$

To better understand the meaning of the compression gain, we consider the following example. Let $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a TT-vector and $\hat{\mathbf{a}}$ its approximation at accuracy δ from the TT-rounding, then if the compression gain is equal to $q \in \mathbb{Q}$, it means that after the compression, storing $\hat{\mathbf{a}}$ needs $1/q$ of the memory we would need to store $\mathbf{a}$.

1.4 Concluding remarks

The purpose of this chapter was to give an overview presenting tensors, their main operations, and the decomposition techniques, on which the following chapters will rely. Tensors carry multiple meanings, which together with the sometimes tangled notation, may easily confuse the reader.

In particular Section 1.1 of this chapter aims to clarify three different natures of tensors, i.e., tensors as multidimensional arrays, as an abstract element of tensor product of linear spaces, as multilinear operator among linear spaces. We emphasized that also in the classical linear algebra framework matrices are regarded from these three different viewpoints, depending on the purpose.

Section 1.2 presents tensor slices and fibres, introducing a specific notation, which we will simplify in other chapters of this manuscript. The tensor matricization with its performing order, and its connection with the Kronecker product are described as well. In this same section, tensor operations, from the most common, such as the tensor sum or the product of tensor and scalars, to more peculiar ones, such as the tensor product or the tensor contraction, are presented in detail. We underline as much as possible that many tensor operations, such as the matrix-times-tensor one or the inner product, are actually contractions, to which we assign a name because of their importance in the tensor theory.

The second part illustrates two tensor decomposition techniques, used in the chapters that follow. We describe firstly the Tucker model and the Tucker approximation problem, relating it with the classical matrix best rank r approximation problem. In Section 1.3.1, we address the computational key aspects related to the Tucker approximation problem. The best-known algorithms, i.e., HOSVD, T-HOSVD, and HOOI are entirely illustrated and analysed. Moreover, we present the benefit of approximating a tensor by the Tucker

Algorithm 5 $\hat{\underline{\mathbf{a}}}_1, \ldots, \hat{\underline{\mathbf{a}}}_d = \text{TT-rounding}\Big((\underline{\mathbf{a}}_1, \ldots, \underline{\mathbf{a}}_d),\, \delta\Big)$

input: $\underline{\mathbf{a}}_i \in \mathbb{R}^{r_{i-1} \times n_i \times r_i}$ for every $i \in \{1, \ldots, d\}$, $\delta \in \mathbb{R}_+$
$\delta' = \frac{\delta}{\sqrt{d-1}} \|\mathbf{a}\|$ ▷ scale the rounding accuracy δ
for $i = d, \ldots, 2$ **do**
    $A_i = \texttt{matricize}(\underline{\mathbf{a}}_i, \texttt{mode} = 1)$ ▷ matricize each core
    ▷ compute the QR-factorization of the transposed matricization
    $Q_i, R = \texttt{QR}(A_i^\top)$
    ▷ set the i-th core equal to its orthogonal factor Q
    $\underline{\mathbf{a}}_i = \texttt{reshape}(Q_i^\top, [r_{i-1},\, n_i,\, r_i])$
    ▷ move the upper triangular factor R to the previous core
    $\underline{\mathbf{a}}_{i-1} = \underline{\mathbf{a}}_{i-1} \times_3 R^\top$
end for
$s_0 = 1$
for $i = 1, \ldots, d-1$ **do**
    $A_i = \texttt{matricize}(\underline{\mathbf{a}}_i, \texttt{mode} = 3)$ ▷ matricize the i-th core
    ▷ compute the truncated SVD of the transposed matricization
    $\hat{U}_i, \hat{\Sigma}_i, \hat{V}_i = \texttt{T-SVD}(A_i^\top, \delta')$
    ▷ define the approximated i-th TT-core
    $\hat{\underline{\mathbf{a}}}_i = \texttt{reshape}(\hat{U}_i, [s_{i-1},\, n_i,\, s_i])$ with s_i equal to the number of columns of $\hat{U}_i$
    ▷ move the singular values and the right orthogonal basis to the next TT-core
    $\underline{\mathbf{a}}_{i+1} = \mathbf{a}_{i+1} \times_1 (\hat{\Sigma}_i \hat{V}_i^\top)$
end for
$\hat{\underline{\mathbf{a}}}_d = \mathbf{a}_d$ ▷ define the last TT-core
return: $\hat{\underline{\mathbf{a}}}_1, \ldots, \hat{\underline{\mathbf{a}}}_d$

model from the memory storage requirement viewpoint. The following Section 1.3.2 focuses on the Tensor-Train model. After stating its definition, we describe how to benefit from the TT-formalism, when performing some basic tensor operations. It is highlighted that this formalism is suitable also for representing tensors as multilinear operators, becoming consequently one of the most popular tensor decomposition techniques in the numerical linear algebra community. As for the Tucker case, the computational aspects, i.e., the fundamental algorithms for representing in TT-format and approximating in TT-format a tensor, are analysed. Moreover, the memory benefits and flaws in storing a tensor in TT-format are presented to complete the Tensor Train section.

Part I

Numerical linear algebra

I.I Introduction

As stated in 1947 in [106], numerical linear algebra studies, among other topics, the errors that appear when linear algebra operations are computationally performed. Generally speaking, numbers are represented with round-off errors inside computers. To those representation errors, computational ones are added when arithmetic operations are performed. Potentially the computational result of a sequence of linear algebra operations, for example, the computed solution of a linear system, could be extremely far from the theoretically expected one. Therefore, mathematicians and computer scientists studied and still do how rounding errors propagate when a numerical linear algebra algorithm is used, developing the so-called numerical stability analysis. A very common approach to studying the error propagation in numerical linear algebra is the backward stability analysis, popularized by Wilkinson during the 1960s in [143, 144]. We say that the computational solution of a problem is backward stable if it is the exact solution of the same problem with a perturbation. The algorithm producing this computed result is consequently said backward stable. In floating point arithmetic, the perturbation is usually expressed in terms of unit round-off with which a representation and a computation accuracy are usually associated; we present a brief overview of the floating point system in the following sections.

I.I.I Solving linear system: the Generalized Minimal RESidual

The backward stability analysis has been carried out over algorithms often meant to solve linear systems. Indeed, among the many research topics addressed in numerical linear algebra, solving linear systems of equations has always been a key one. The techniques for computing a solution of the linear systems are usually classified into two groups: direct and iterative methods. The former computes the solution of the given linear system in a finite number of steps, while the latter ones iteratively converge toward the exact solution from an initial guess. Among the direct solvers, it is worthwhile mentioning back substitutions, Gauss elimination, LU factorization, QR factorization, and Cholesky factorization. Examples of classical iterative methods are the conjugate gradient, the biconjugate gradient, the MINimal RESidual (MINRES) [112] and the Generalised Minimal RESidual (GMRES)[124].

The GMRES solver was presented in 1986 by Saad and Schultz as a generalization of the 1975 solver MINRES. They are both based on the Krylov subspace technique, previously described by Krylov in 1931 [91]. The key idea of GMRES is approximating the solution of the input linear system by minimizing the residual norm in the considered Krylov space, whose dimension increases iteratively. To achieve this purpose, GMRES relies on the Arnoldi relation, firstly presented in 1951 [4], which simplifies the minimization problem through a projection. Moreover, an orthogonalization kernel is usually employed in this solver to construct a basis of the Krylov subspace; the most common choices are Modified Gram-Schmidt (MGS), from 1986 original work [124] and Householder from 1988 variant [142], because of the numerical quality of their basis. As a

stopping criterion for GMRES, the choice suggested in the literature [40, 59, 113] is the backward error based on the linear system matrix and right-hand side associated with the iterative solution. When the backward error is lower than a prescribed threshold, the iterations stop, implying that the current iterative solution can be considered as the exact solution of a perturbed problem, where the relative norm of the perturbation is lower than a prescribed threshold. The backward stability of both MGS and Householder GMRES implementations was theoretically proved in [113] and [40] respectively, showing that the attainable normwise backward error of GMRES is of the same order as the unit round-off.

I.I.II Orthogonalization schemes

It is quite common for numerical linear algebra algorithms to require the orthogonalization of a set of vectors, which have different computational properties. In a theoretical context, an orthogonal basis is generated from a set of linear independent vectors through the Gram-Schmidt process, proposed by Schmidt in 1907 in [127], who credited also Gram's previous work of 1883 [52]. The link between the two techniques was highlighted in 1935 by Wong [147], but it is worthwhile mentioning that similar procedures appeared in previous works of Gauss, Legendre, Laplace, and Cauchy [96]. As remarked in [96], from an algorithmic point of view the methods proposed by Gram and Schmidt differ, even though in a theoretical framework they lead to the same results. Their tiny difference went unnoticed for many years, as mentioned in [143]. Nowadays, we know that the Modified Gram-Schmidt (MGS), closer to the algorithm proposed in [52], is more stable than the Classical Gram-Schmidt (CGS), which is similar to the method described in [127], comparing the results of [51] and [15].

Essentially, we can affirm that the closer the orthogonal basis is to the expected theoretical one, the more reliable the orthogonalization kernel will be inside a more articulated algorithm. The loss of orthogonality is the preferred tool to estimate how much the computed orthogonal basis differs from the expected theoretical one. Therefore over the years, mathematicians and computer scientists proposed new orthogonalization schemes, trying to reduce the loss of orthogonality of MGS, improving its quality as an orthogonalization kernel. In 1958, Householder transformation made its first appearance in [76] and it became a new alternative and reliable tool for orthogonalizing vector sets. The loss of orthogonality of the Householder orthogonalization algorithm, proved in [145], outcompetes the CGS and MGS one, even if this kernel is computationally more expensive than the other two. In more recent years, other orthogonalization methods have been proposed as the Gram-Schmidt algorithm with re-orthogonalization loops, which firstly appeared in 1966 and was further studied [1, 33, 116], or the Gram algorithm proposed in 2002 [131].

I.II Finite precision arithmetic

In this section, we present the fundamental concepts around finite precision arithmetic, that will play a key role in Chapter 2 and 3.

Because of their finiteness, computers store a finite set of numbers from the infinite real set, approximating them with a finite number of digits. The floating-point system is a representation model for describing the storing possibilities of computers.

Definition I.II.i. *Given the natural numbers $\beta, t \in \mathbb{N}$, the floating point number system is a set $\mathcal{F} \subset \mathbb{R}$ whose generic element $x \in \mathcal{F}$ writes*

$$x = \pm m \times \beta^{e-t} \tag{I.ii}$$

where the significand *$m \in \mathbb{N}_0$ is such that $0 \leq m \leq \beta^t - 1$ and $e \in \mathbb{Z}$ is the* exponent *lower and upper bounded by $e_{min} \in \mathbb{Z}$ and $e_{max} \in \mathbb{Z}$, i.e., $e_{min} \leq e \leq e_{max}$. The value β is called the* base *and t the* precision *of the floating point system $\mathcal{F}$.*
The range *of the non-zero floating point numbers in $\mathcal{F}$ is*

$$\mathcal{R}(\mathcal{F}) = \{y \in \mathcal{F} \Big| \beta^{e_{min}-1} \leq |y| \leq \beta^{e_{max}}(1 - \beta^{-t})\}.$$

We briefly present as an example of a floating point system the: IEEE standard arithmetic, which appeared as IEEE standard 754 in 1985 [77]. It is one of the most popular floating point systems [68] and the one used for all the numerical experiments presented in the following chapters.

Example I.II.i. The IEEE standard arithmetic floating point system includes two formats. The single precision format has basis $\beta = 2$, precision $t = 24$ and lower exponential bound $e_{min} = -125$ while the maximum is $e_{max} = 128$. The double precision format has the same basis, but precision $t = 53$, $e_{min} = -1021$ and $e_{max} = 1024$.The

Once a floating point system $\mathcal{F}$ is considered, we can introduce the rounding function.

Definition I.II.ii. *Given the basis β and precision t of the floating point system $\mathcal{F}$, let $\mathcal{G} \subset \mathbb{R}$ be the set of all real numbers x expressed as $y = m \times \beta^{e-t}$ for every $e \in \mathbb{Z}$ and $m \in \mathbb{N}_0$. The rounding function $fl : \mathbb{R} \to \mathcal{G}$ associates each $x \in \mathbb{R}$ with the closest element in $\mathcal{G}$, i.e.,*

$$fl(x) = \arg\min_{y \in \mathcal{G}} |y - x|.$$

Notice that $\mathcal{F} \subset \mathcal{G}$ by definition. Thus, if $|fl(x)| \geq \max\{|y| \text{ with } y \in \mathcal{F}\}$, then the rounding *overflows*, while if $0 < |fl(x)| \leq \min\{|y| \text{ with } y \in \mathcal{F} - \{0\}\}$, then the rounding *underflows*. Another key concept in finite precision arithmetic is the unit round-off.

Definition I.II.iii. *Let $\beta \in \mathbb{N}$ and $t \in \mathbb{N}$ be the floating point system $\mathcal{F}$ basis and precision respectively, then the unit round-off $u \in \mathbb{R}_+$ is*

$$u = \frac{1}{2}\beta^{1-t}.$$

In the IEEE standard arithmetic floating point system, the single precision unit round-off is $u = 10^{-8}$, while for the double precision format it is $u = 10^{-16}$.
The unit round-off and the rounding are strictly connected, as shown in the following classical theorem.

Theorem I.II.ii. *[68, Theorem 2.2] Let $\mathcal{F}$ be a floating point system, if $x \in \mathbb{R}$ lies in $\mathcal{R}(\mathcal{F})$ the range of $\mathcal{F}$, then there exists $\delta \in \mathbb{R}$ such that*

$$fl(x) = x(1+\delta) \qquad \textit{with} \qquad |\delta| \leq u.$$

This theorem states that the rounding of x in the considered floating point system is equal to the mathematical value of x times a factor very close to 1.

I.III Rounding error analysis

The representation of real numbers through a floating point system introduces necessarily some errors, called rounding errors. These rounding errors may propagate or they may cancel out [68] through a sequence of operations, that is an algorithm. Thus, in numerical linear algebra, the rounding error analysis is a fundamental step to evaluate the quality of the considered algorithms.

Once the floating point system $\mathcal{F}$ is chosen, the rounding error analysis is performed under the *standard model* assumption that if $x, y \in \mathcal{F}$ then there exists $\delta \in \mathbb{R}$ such that

$$fl(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta) \qquad \text{with} \qquad |\delta| \leq u$$

with u the unit round-off of $\mathcal{F}$ and op being a basic arithmetic operation among sum, difference, multiplication, or division. Sometimes among the basic operations satisfying the previous equation we include also the square root one. This hypothesis states that the computed value of $(x \operatorname{op} y)$ is equal to the value of theoretical operation $(x \operatorname{op} y)$ times a factor that is extremely close to 1. This standard model assumption holds in particular for the IEEE standard arithmetic system.

The rounding error analysis aims to establish bounds that express the effects of rounding errors on the outcome of an operation sequence. There exist different approaches for studying rounding errors. For example, let $g : \mathbb{R} \to \mathbb{R}$ be function such that $y = g(x)$, when the function g is implemented on the machine, we get $\hat{y} = g(x)$ for every $x \in \mathbb{R}$. A first possibility is investigating how close y and its computed value $\hat{y}$ are, bounding the relative *forward error* written as

$$e_f(\hat{y}) = \frac{|y - \hat{y}|}{|y|}.$$

Another possibility is seeking how many and how large are the values Δx such that $\hat{y} = g(x + \Delta x)$, bounding the *backward error* expressed as

$$e_b(\hat{x}) = \frac{|x - \hat{x}|}{|x|} = \frac{|\Delta x|}{|x|} \qquad \text{with} \qquad \hat{y} = g(\hat{x}) = g(x + \Delta x).$$

Thus, investigating the quality of the backward error means studying how large the perturbation Δx of the value of x can be, that is how close x and $\hat{x} = x + \Delta x$ can be, to

lead to the same final value of $\hat{y}$. The two main benefits of this backward error analysis approach are the follows. First, the rounding error is interpreted as a perturbation of the input data, so the solution we get is the best up to the quality of the available data [68]. Second, the forward error can be addressed separately through the perturbation theory [68]. Consequently, the backward error analysis purpose is to establish when an algorithm is backward stable in the considered floating point system.

Definition I.III.i. *The method to compute $y = g(x)$ is* backward stable *in the considered floating point system $\mathcal{F}$ if denoted by $\hat{y}$ the computed value of y, for every $x \in \mathbb{R}$ we have*

$$\hat{y} = g(x + \Delta x) \qquad with \qquad \frac{|\Delta x|}{|x|} \leq cu$$

with $c \in \mathbb{R}_+$ small and u the unit round-off for $\mathcal{F}$.

Frequently, in the backward error analysis of algorithms working with matrices, the matrix condition number, defined next, has a central role.

Definition I.III.ii. *Let $A \in \mathbb{R}^{n \times n}$ be a matrix, the condition number $\kappa_2(A) \in \mathbb{R}_+$ with respect to the L2 norm is*

$$\kappa_2(A) = ||A||_2 ||A^{-1}||_2 = \frac{\sigma_1}{\sigma_r}$$

where σ_1 and σ_r are the maximum and minimum non-zero singular values of A respectively, with $r = rank(A)$.

We refer the reader to [68] for a complete presentation of the floating point system and the backward stability analysis of many numerical linear algebra algorithms.

I.IV Tensor formalism

In Chapters 2 and 3, tensors and multilinear operator are represented through the TT-format [108], described in Section 1.3.2, to describe numerical linear algebra algorithms.

Chapter 2

A robust GMRES in TT-format

2.1 Introduction

In many domains of sciences and engineering, the problems to be solved can naturally be modelled mathematically as linear systems of equations, i.e., as

$$Ax = b$$

where $A \in \mathbb{R}^{n \times n}$ represents a linear endomorphism of $\mathbb{R}^n$, $b \in \mathbb{R}^n$ is the right-hand side and $x \in \mathbb{R}^n$ is the searched solution. Two main approaches have emerged in the search for techniques to solve linear systems: direct and iterative methods. Direct solvers compute the solution in a finite number of steps, usually transforming equivalently the matrix A, as LU or QR-factorization methods, while iterative solvers approximate iteratively the solution starting from an initial guess, as conjugate gradient or minimal residual method. The Generalized Minimal RESidual (GMRES) method is an iterative solver based on the Krylov subspace technique [91], which approximates iteratively the solution minimizing the linear system residual in the associated Krylov space. Two main variants of GMRES exist: the original one [124] that constructs an orthogonal basis of the Krylov space through the Modified Gram-Schmidt kernel (MGS-GMRES) and an alternative one [144] relying on Householder transformations, namely Householder-GMRES (H-GMRES). In [40, 113], the authors prove the norm-wise backward stability of both these algorithms under the classical IEEE model assumption, i.e., with the unit rounding error of floating point operations and the data storage both bounded by the same unit round-off u, which depends on the selected working arithmetic. This chapter aims at investigating the backward stability of GMRES when the leading part of the rounding error is related to the selected data representation. The use of low-rank tensor approximation, employed to tackle the 'curse of dimensionality' when solving tensor linear systems, represents a practical study case.

When storing all the data in classical IEEE floating point format is not affordable, part of the data has to be compressed, introducing relative perturbations on them that will later on be involved in further computations. If the data compression introduces

relative component-wise perturbations, the existing backward stability analyses for MGS-GMRES [40] and H-GMRES [113] still apply. However, if only the norm-wise relative perturbations of the data storage can be controlled, technical details of the theoretical backward stability results do not readily apply. In the first half of this chapter, we investigate numerically the backward stability of both GMRES variants in a synthetic context, imposing perturbations that target a certain accuracy component-wise or norm-wise, and in a practical framework, using an agnostic lossy compressor. The key idea of our study is to compress vectors all at once, into which particular scalars are therefore not constrained by any memory alignment. In particular, they do not need to stick with an available (hardware) precision such as fp32 or fp16. Scalars do not live in "two or more precisions chosen from a small number of available precisions" [68] but on a continuum of possible accuracy. We call such as scheme *variable accuracy data storage*, or, for short, *variable accuracy* [3].

Different attempts [8, 39, 86] have been made over the years to extend iterative solvers to the tensor framework to compute the solution of high order tensor linear systems, that is i.e., as

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

where $\mathbf{A}$ represents multilinear endomorphism on $\mathbb{R}^{n_1 \times \cdots \times n_d}$, $\mathbf{b} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is the right-hand side and $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is the searched solution. For sake of simplicity, henceforth these linear systems will be referred to as order d tensor linear systems, where d is the order of the tensor space where they are defined. They all rely on compression techniques, as HOSVD [36], Hierarchical Tucker [54] or TT [108], meant to tackle the 'curse of dimensionality', i.e., the exponential growth of the variables number in function of the order of the linear system. Since these compression techniques introduce perturbations of a prescribed accuracy to avoid memory deficiencies, their application inside iterative solvers fits in the variable accuracy scheme. The second half of this chapter studies numerically the backward stability of GMRES extended to the tensor framework with the TT-formalism (TT-GMRES), comparing our results with another GMRES realization in TT-format presented in [39]. Moreover, we highlight how the tensor structure enables us to solve simultaneously many tensor linear systems of order d depending on a parameter by simply reformulating the problem in a space of order $(d+1)$.

The remainder of this Chapter is structured as follows. Section 2.2 collects the results related to the variable accuracy of GMRES in the matrix framework. After summarizing in Section 2.2.1 the backward stability theoretical results for GMRES, we present the experiments with component-wise and norm-wise perturbations, both from a synthetic compression and an agnostic lossy compressor, in Section 2.2.2 and 2.2.3 respectively. The variable accuracy study of GMRES extended to the tensor framework is found in Section 2.3. The TT-GMRES algorithm is described in Section 2.3.1, while Section 2.3.2 collects all the theoretical results related to the simultaneous solution of many parameter dependent tensor linear systems. We report about our numerical experiments with TT-GMRES in Section 2.3.3. In particular, in Section 2.3.3.1 we compare our TT-GMRES version with the one proposed in [39] from the backward stability viewpoint, proving that our method is more robust. The experiments with parameter dependent multilinear

operator or right-hand side are analysed in Section 2.3.3.2, illustrating the tightness of the bounds constructed in Section 2.3.2.

2.2 GMRES in matrix computation framework

In this section we present our variable accuracy approach applied to classical linear systems. After recalling briefly the theoretical backward stability results for GMRES, we describe the variable accuracy approach. Finally, we report about the numerical experiments we perform introducing component-wise and norm-wise perturbation, coming either from a synthetic compression or from an agnostic lossy compressor.

2.2.1 Background on GMRES

Starting from the zero initial guess, GMRES [124] constructs a series of approximations x_k in Krylov subspaces of increasing dimension k so that the residual norm of the sequence of iterates is decreasing over these nested spaces. More specifically:

$$x_k = \underset{x \in \mathcal{K}_k(A,b)}{\operatorname{argmin}} \|b - Ax\|,$$

with

$$\mathcal{K}_k(A,b) = \operatorname{span}\{b, Ab, \ldots, A^{k-1}b\}$$

the k-dimensional Krylov subspace spanned by A and b. In practice, an orthogonal matrix $V_k = [v_1, \ldots, v_k] \in \mathbb{R}^{n \times k}$ and an upper Hessenberg matrix $\bar{H}_k \in \mathbb{R}^{(k+1)\times k}$ are iteratively constructed using the Arnoldi procedure such that $\operatorname{span}\{V_k\} = \mathcal{K}_k(A,b)$ and

$$AV_k = V_{k+1}\bar{H}_k, \qquad \text{with} \qquad V_{k+1}^T V_{k+1} = I_{k+1}.$$

This is often referred to as the Arnoldi relation. Consequently, $x_k = V_k y_k$ with

$$y_k = \underset{y \in \mathbb{R}^k}{\operatorname{argmin}} \left\| \beta e_1 - \bar{H}_k y \right\|,$$

where $\beta = \|b\|$ and $e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^{k+1}$ so that in exact arithmetic the following equality holds between the least square residual and the true residual

$$\|\tilde{r}_k\| = \|\beta e_1 - \bar{H}_k y\| = \|b - Ax_k\|. \tag{2.1}$$

In finite precision calculation, this equality no longer holds. Another matrix equality can be formed based on the Arnoldi relation that reads

$$(v_1, AV_k) = V_{k+1}R_{k+1}, \tag{2.2}$$

where R_{k+1} is a $(k+1) \times (k+1)$ upper triangular matrix that writes

$$R_{k+1} = \left(e_1, \bar{H}_k\right).$$

Equation (2.2) shows that the Arnoldi algorithm actually computes a QR factorization of (v_1, AV_k). Such a factorization could also be computed using Householder transformations, this is the core idea of the H-GMRES variant proposed by Walker [142] and depicted in Algorithm 7. The original GMRES by Saad and Schultz [124] is based on a more classic Arnoldi algorithm that uses Modified Gram-Schmidt procedure to orthogonalize the Krylov basis, often referred to as MGS-GMRES and depicted in algorithm 6. We refer to [124, 125] for a more detailed presentation.

Algorithm 6 x, `hasConverged` = MGS-GMRES(A, b, m, ε)

input: A, x_0, b, ε.
$r_0 = b$, $\beta = \|r_0\|$ and $v_1 = r_0/\beta$
for $k = 1, \ldots$ **do**
    $w_k = av_k$
    for $i = 1, \ldots, k$ **do**
        $\bar{H}(i,k) = \langle v_i, w \rangle$
        $w_k = w_k - \bar{H}(i,k)v_i$
    end for
    $\bar{H}(k+1,k) = \|w_k\|$
    $v_{k+1} = \Big(1/\bar{H}(k+1,k)\Big)w_k$
    $y_k = \underset{y \in \mathbb{R}^k}{\operatorname{argmin}} \left\| \beta e_1 - \bar{H}_k y \right\|$ with $\bar{H}_k = H(:k+1, :k)$
    $x_k = x_0 + v_k y_k$
    if $\eta_{A,b}(x_k) < \varepsilon$ **then**
        `hasConverged = True`
        break
    end if
end for
return: $x = x_k$, `hasConverged`

Because the orthonormal basis V_k has to be stored, a restart parameter defining the maximal dimension of the search Krylov space is used to control the memory footprint of the solver in both the implementations. If the maximum dimension of the search space is reached without converging, the algorithm is restarted using the final iterate as the initial guess for a new cycle of GMRES. Furthermore, it is often needed to consider a preconditioner to speed-up the convergence. Using right-preconditioned GMRES consists of considering a non singular matrix M, the so-called preconditioner that approximates the inverse of A in some sense. In that case, GMRES is applied to the preconditioned system $AMt = b$. Once the solution t has been computed the solution of the original system is recovered as $x = Mt$. The right-preconditioned GMRES is sketched in Algorithm 8 for a restart parameter `m` and a convergence threshold ε .

The backward error analysis, popularized by J.H. Wilkinson who contributed significantly to its development [143, 144], gives powerful stopping criteria for iterative solvers, as GMRES, based indeed on the backward error [40, 59, 113]. In particular, Rigal and

Algorithm 7 x, `hasConverged` = H-GMRES(A, b, m, ε)

input: A, x_0, b, ε.
$r_0 = b$, $\beta = \|r_0\|$
compute u_1 to define the Householder reflector $P_1 = I - 2u_1u_1^T$ such that $P_1 r_0 = \beta e_1$
for $k = 1, \ldots, m$ **do**
    $v_k = \prod_{j=k}^{1} P_j e_k$
    $w = \prod_{j=1}^{k} P_j A v_k$
    compute u_{k+1} to define the Householder reflector $P_{k+1} = I - 2u_{k+1}u_{k+1}^T$ s.t. $P_{k+1}w(k+1:n) = \|w(k+1:n)\| e_{k+1}(k+1:n)$ and $p_{k+1}w(1:k) = w(1:k)$
    $\bar{H}(:,k) = (P_{k+1}w)(1:k+1)$
    $y_k = \underset{y \in \mathbb{R}^k}{\operatorname{argmin}} \left\| \beta e_1 - \bar{H}_k y \right\|$ with $\bar{H}_k = H(:k+1, :k)$
    ▷ compute the current iterate
    $z = 0$
    for $j = k, \ldots, 1$ **do**
        $z = P_j(y_k(j)e_j + z)$
    end for
    $x_k = x_0 + z$
    if $\eta_{A,b}(x_k) < \varepsilon$ **then**
        `hasConverged = True`
        break
    end if
end for
return: $x = x_k$, `hasConverged`

Algorithm 8 x, `hasConverged` = Right-GMRES(A, M, b, x_0, `m`, ε)

input: A, M, b, `m`, ε.
`hasConverged = False`
$x = x_0$
while `not(hasConverged)` **do**
    $r = b - Ax$ ▷ Iterative refinement step with at most m GMRES iterations on AM
    t_k, `hasConverged` = GMRES(AM, r, `m`, ε)
    $x = x + Mt_k$ ▷ Update the unpreconditionned with the computed correction
end while
return: x, `hasConverged`

Gache [121] showed that the approximate solution x_k at iteration k can be interpreted as the exact solution of a perturbed linear system where the relative norms of the perturbations can be easily computed. We denote $\eta_{A,b}(x_k)$ this norm-wise backward error for linear systems

$$\begin{aligned}\eta_{A,b}(x_k) &= \min_{\Delta A,\Delta b} \{\tau > 0 : \|\Delta A\| \leq \tau\|A\|\,,\ \|\Delta b\| \leq \tau\|b\| \\ &\qquad \text{and } (A+\Delta A)x_k = b + \Delta b\} \\ &= \frac{\|Ax_k - b\|}{\|A\|\|x_k\| + \|b\|}. \end{aligned} \tag{2.3}$$

In some circumstances, a simpler backward error criterion based on perturbations only in the right-hand side can also be considered, that leads to the second possible choice

$$\begin{aligned}\eta_b(x_k) &= \min_{\Delta b} \{\tau > 0 : \|\Delta b\| \leq \tau\|b\| \text{ and } Ax_k = b + \Delta b\} \\ &= \frac{\|Ax_k - b\|}{\|b\|}. \end{aligned} \tag{2.4}$$

Based on the basic IEEE model $fl(a \operatorname{op} b) = (a \operatorname{op} b)(1+\varepsilon)$, with $\operatorname{op} \in \{+, -, \times, \div\}$, $|\varepsilon| < u$ and u the unit round-off of the working precision, many theoretical results exist in the literature. In particular, it is known [145, p. 152-161] that the Householder QR factorization of a set of n-dimensional vectors $\mathcal{S} = \{s_1, \ldots, s_m\}$ generates an orthonormal basis $\tilde{Q}$ with orthonormality quality

$$\tilde{Q}^T\tilde{Q} = I + E \text{ with } \|E\|_2 \approx u, \tag{2.5}$$

while MGS produces [15] a $\tilde{Q}$ factor such that

$$\tilde{Q}^T\tilde{Q} = I + E \text{ with } \|E\|_2 \approx \kappa(S)\, u, \tag{2.6}$$

where $\kappa(S)$ is the 2-norm condition number of the matrix S whose i-th column is $s_i \in \mathcal{S}$.

Thanks to the backward stability of Householder QR given by (2.5) for computing the Householder vectors, the backward stability of H-GMRES was established in [40, Theorem 4.1 and Corollary 4.2] assuming that the matrix A is not close to singularity.

Remark 2.2.1. Under technical assumptions [40, Corollary 4.2], it is shown that the norm-wise backward error at the last iterate x_n of H-GMRES is such that $\eta_{A,b}(x_n) = \mathcal{O}(u)$, where u is the unit round-off error of the working precision.

In MGS-GMRES, it is known that the Arnoldi basis progressively departs from orthogonality as indicated by (2.6). However, the following result has been shown in [114]

$$\eta_{A,b}(x_k) \cdot \|I - \tilde{V}_k^T\tilde{V}_k\|_F = \mathcal{O}(u), \tag{2.7}$$

where the columns of $\tilde{V}_k \in \mathbb{R}^{n\times k}$, computed by MGS-Arnoldi, form a basis for the Krylov space $\mathcal{K}_k(A, b)$. This result implies that it is impossible to have a significant loss of orthogonality as long as the norm-wise relative backward error is very small. Later, the backward stability of MGS-GMRES was proved in [113], that reads:

Theorem 2.2.2 ([113]). *Assuming that A is not close to singularity, that is,*

$$\sigma_{min}(A) \gg n^2||A||u,$$

where u is the unit round-off error of the working precision; let $k \in \mathbb{N}$ be the first integer such that

$$\kappa(\tilde{V}_{k+1}) > \frac{4}{3}, \tag{2.8}$$

then x_k satisfies

$$\eta_{A,b}(x_k) = \mathcal{O}(u).$$

We refer to [113] for the details of the tedious and technical proof and to [104, p. 227] for a short exposure. We also note that for technical reason, the proof is established using the Frobenius norm of A to define $\eta_{A,b}$ so that the norm-wise perturbations on A are measured in Frobenius norm and not in Euclidean norm. In practice, the backward stability is observed using the 2-norm and we only consider it in the sequel.

2.2.2 Numerical experiments with component-wise perturbations

For the numerical experiments of this and the following section, the norm of the linear operator is estimated using a randomized SVD [103, 122]. In addition, for all numerical experiments right preconditioned GMRES is considered to enable a fast convergence, so that GMRES does effectively solve

$$AMz = b$$

where M is a preconditioner operator. For calculation performed in regular matrix case, we consider a preconditioner based on an $ILU(t)$ [123] factorization. In the preconditioned context, the backward error $\eta_{AM,b}(z_k)$ is the one that GMRES can drive down to $\mathcal{O}(u)$.

2.2.2.1 Variable accuracy approach

We are interested in the numerical behaviour of GMRES where the processed data may be numerically altered for two reasons. The first possible source is the employed finite precision arithmetic discussed above. We still denote u the associated unit round-off error. In this work, we further assume that the data may be compressed leading to another possible source of inaccuracy due to the corresponding compressed data representation. More precisely, we denote δ-storage$(\cdot)$ the function that enables to store a given data in a format, referred to as δ-component-wise data format, that induces an unit round-off δ; that is:

$$\delta\text{-storage}(a) = a(1+\xi) \text{ with } |\xi| \leq \delta.$$

When a basic calculation is performed and the result stored in a δ-representation we have

$$fl_\delta(a \operatorname{op} b) = \delta\text{-storage}(fl(a \operatorname{op} b)) = (a \operatorname{op} b)(1+\varepsilon)(1+\xi),$$

with $|\varepsilon| \leq u$ and $|\xi| \leq \delta$. Neglecting the second order term $\varepsilon\xi$, we obtain

$$fl_\delta(a \operatorname{op} b) = (a \operatorname{op} b)(1 + \varepsilon + \xi),$$

with $|\varepsilon + \xi| \leq u + \delta$, so that all the theoretical results, presented in the previous section, that were established with the same unit round-off u for the basic operations and the data representation still hold, but with unit round-off $u + \delta$. In can be noted that in the particular case $u \ll \delta$, it further reduces to $fl_\delta(a \operatorname{op} b) = (a \operatorname{op} b)(1 + \xi)$, with $|\xi| \leq \delta$ and those same results then hold with unit round-off δ. For a matter of readability, we will make this particular assumption (results hold with unit round-off δ) in rest of this section, but they can be more generally interpreted without this assumption, in which case a $u + \delta$ shall be considered instead.

2.2.2.2 δ-component-wise data storage

We numerically investigate the behaviour of the MGS-GMRES and H-GMRES where the storage function δ-storage$(\cdot)$ is used to have a δ-representation of all the vectors of size n. Regular IEEE calculation is used and data associated with small dimension matrices and vectors are stored in regular IEEE format. That are all the data involved in the least squares problem solution.

In Figure 2.1, we plot the convergence history of the norm-wise backward error for both MGS-GMRES and H-GMRES for various values of δ for a test matrix from the Florida test collection [35]. The dashed vertical blue line indicates the iteration where $\kappa(\tilde{V}_k) > \frac{4}{3}$ for MGS-GMRES. It can be seen that the backward stability property does still hold for δ round-off unit due to data representation. Although this is not predicted by any theoretical argument, the convergence of the two GMRES variants is identical up to the proximity of δ. On this example, the convergence of MGS-GMRES and H-GMRES perfectly overlap. Furthermore the relation given by (2.7) between the loss of orthogonality and the backward error for MGS-GMRES, represented by the dashed green curve, does also hold for δ unit round-off. To illustrate that both data representation and finite precision calculation play a role in the attainable accuracy, we present numerical experiments in Figure 2.2 where the calculation are performed either in fp32 (left plots) or fp64 (right plots). On the first row, where the component-wise perturbation is $\delta = 10^{-4}$, which is much larger than u_{32} and u_{64}, the various GMRES implementations do exhibit the same convergence and identical attainable accuracy, i.e., $\mathcal{O}(\delta)$. On the second row, the fp32 implementation has an attainable accuracy that is $\mathcal{O}(u_{32})$ because the δ component-wise perturbation is hidden by the fp32 format. The fp64 calculation reveals the attainable accuracy already observed in Figure 2.1, that is $\mathcal{O}(\delta)$, since $\delta \gg u_{64}$. It can be seen that the loss of orthogonality of the MGS-Arnoldi basis (which is characterized by a condition number that deviates from one and becomes larger than 4/3 after the iteration marked by the vertical dashed blue line) is also affected by δ-representation of the basis vectors. More results on other test examples are depicted in [3, Appendix A].

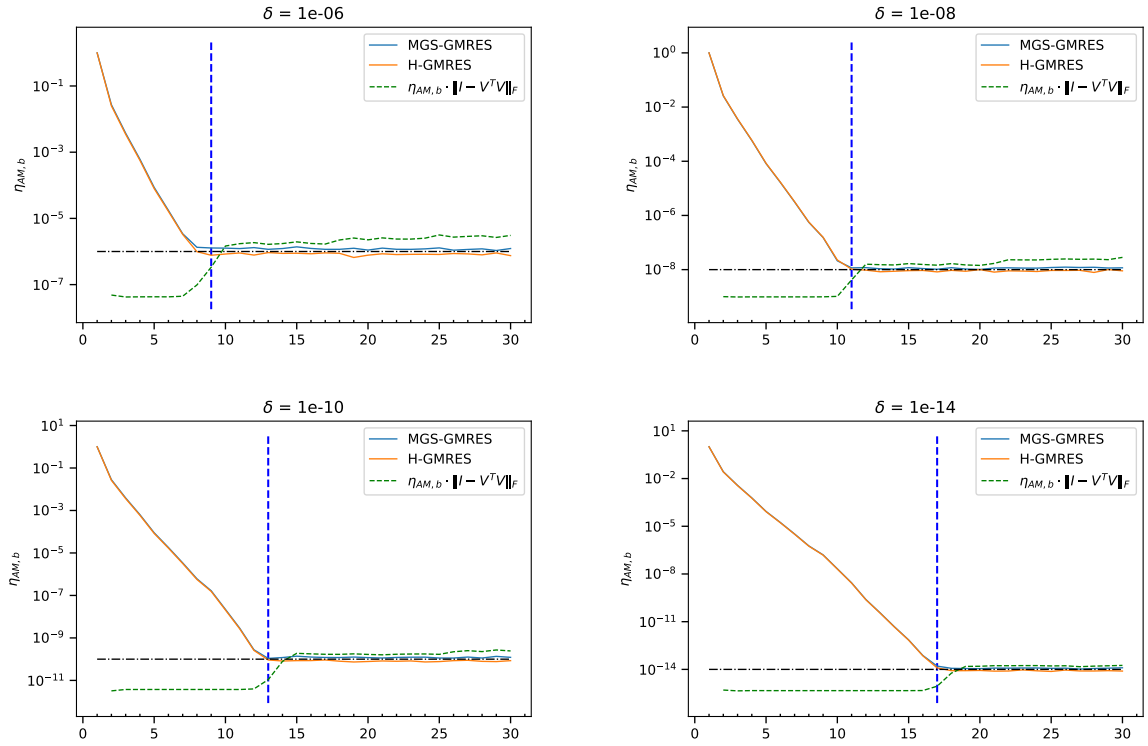


Figure 2.1 – Convergence history of $\eta_{AM,b}$ for gre-115 with ILU(10^{-1}) using δ-data component-wise representation. The horizontal dashed black line indicates δ. The vertical dash blue line represents the first iteration where $\kappa(V_k) > 4/3$ in MGS-GMRES.

2.2.2.3 Solution techniques using SZ compressed format

In this section we present a first practical application of the GMRES algorithm in variable accuracy. We consider an implementation where the Arnoldi basis in MGS-GMRES or the reflector vectors in H-GMRES are compressed to alleviate their memory footprint. More precisely, in MGS-GMRES we compress w_k in line 10 before normalizing it to define v_{k+1}. In H-GMRES, we compress the vectors u_k that defines the Householder reflectors P_k in lines 3 and 7 of Algorithm 7. For those experiments, we used the SZ [38] lossy compressor, an agnostic compressor that does not attempt to exploit underlying numerical properties of the vectors but operates on their binary representation. For the experiments in this section, we use the capability of SZ to ensure a prescribed component-wise relative error between the original data $z \in \mathbb{R}^n$ and the decompressed data $\tilde{z} \in \mathbb{R}^n$, that is,

$$\max_{i=1,\ldots,n} \frac{|z(i) - \tilde{z}(i)|}{|z(i)|} \leq \delta,$$

with $z(i)$ the ith component of z.

Similarly to the previous section, we display the convergence histories of MGS-GMRES and H-GMRES in Figure 2.3, for a few values of the compression control parameter

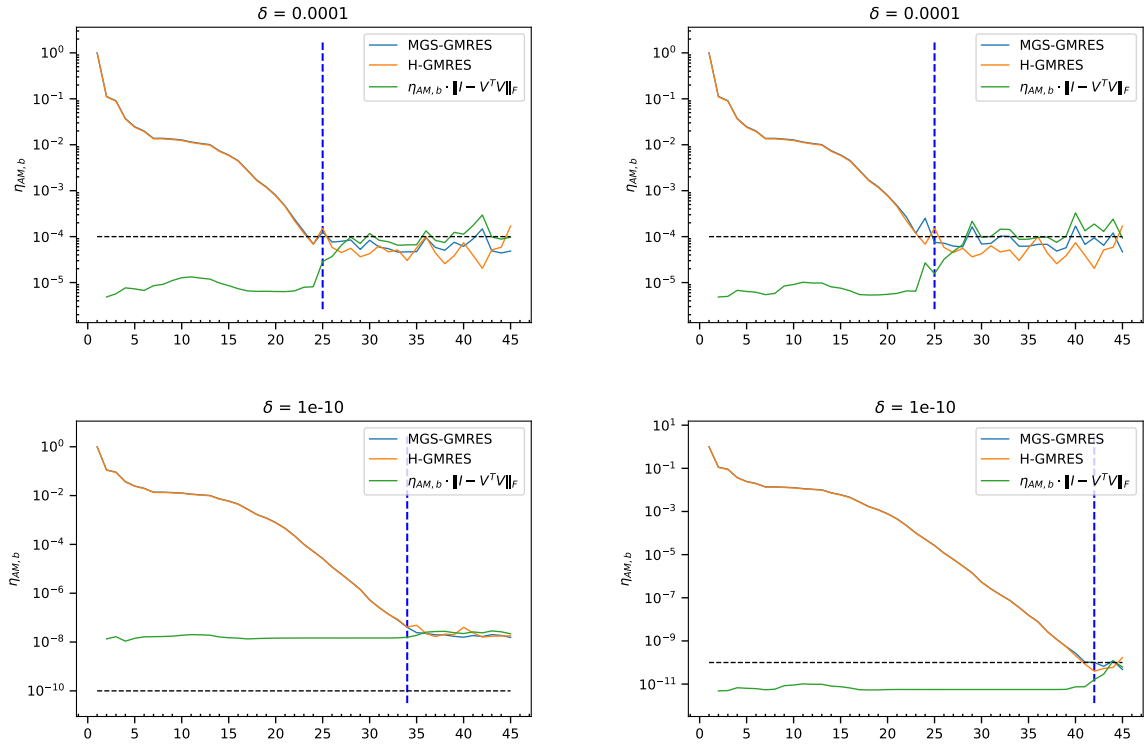


Figure 2.2 – Convergence history of $\eta_{AM,b}$ for gre-343 with ILU($2 \cdot 10^{-1}$) using δ-data component-wise representation with fp32 calculation (left) and fp64 calculation (right). The horizontal dashed black line indicates δ. The vertical dash blue line represents the first iteration where $\kappa(V_k) > 4/3$ in MGS-GMRES.

δ. Although the δ component-wise perturbations only occur when storing the Arnoldi basis or reflector vectors, the general trend of the convergence histories of $\eta_{AM,b}$ remains qualitatively the same. The attainable accuracy is slightly, but not significantly, better. The memory saving enabled by SZ is reported in Table 2.1. A general expected rule of thumb is: "the larger δ, the larger the memory gain". For some matrices, e.g., gre_343, more than 20% of memory saving is observed for a 10^{-14} accuracy. It illustrates the possible significant benefit of these GMRES implementations based on a compressor like SZ for the solution of large problems.

2.2.3 Numerical experiments with norm-wise perturbations

In many contexts, having a control on the relative component-wise error on the vectors is not feasible and only a norm-wise monitoring is possible. Although the existing backward stability analyses of both considered GMRES variants do not readily apply, because some of the technical arguments of the proof are no longer valid, we show through numerical experiments that the property still holds in practice. In Section 2.2.3.1, we report on experiments where the results of all calculations performed on length n vec-

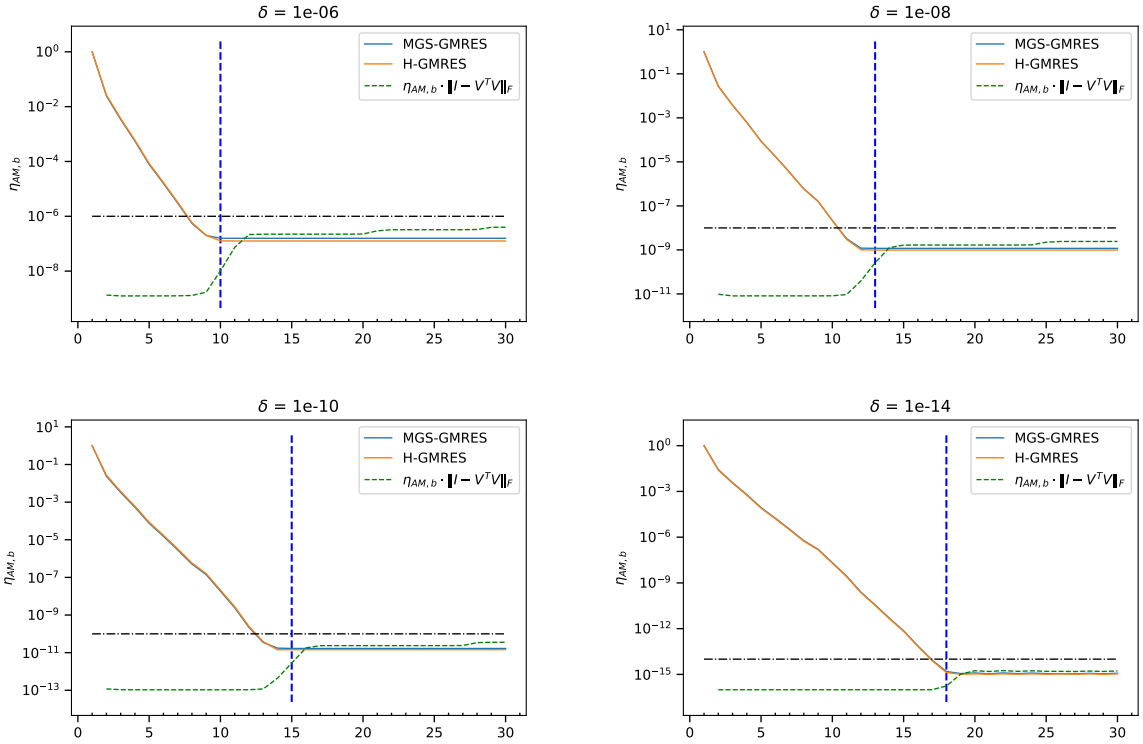


Figure 2.3 – Convergence history of $\eta_{AM,b}$ for gre-115 with ILU(10^{-1}) using SZ-component-wise storage for the Arnoldi basis and Householder reflectors. The horizontal dashed black line indicates δ. The vertical dash blue line represents the first iteration where $\kappa(V_k) > 4/3$ in MGS-GMRES.

	δ					
	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
gre_115 with ILU(10^{-1})						
MGS	9.0	25.1	25.7	16.3	6.6	3.9
Householder	39.6	30.2	30.4	22.2	15.4	12.4
gre_185 with ILU(10^{-1})						
MGS	3.9	32.8	25.6	16.2	5.4	0.0
Householder	22.3	38.3	31.4	23.4	14.2	6.9
gre_343 with ILU(10^{-1})						
MGS	22.3	50.8	46.7	38.5	30.1	21.9
Householder	24.7	52.2	47.7	39.9	31.7	24.1
arc130 with ILU($8 \cdot 10^{-4}$)						
MGS	15.3	15.2	12.8	17.5	14.3	6.9
Householder	38.4	5.8	25.5	21.5	18.2	9.4
e05r0000 with ILU(10^{-2})						
MGS	0.1	36.2	29.5	20.2	9.4	0.2
Householder	6.7	37.0	29.7	20.8	10.4	2.6
e05r0400 with ILU(10^{-2})						
MGS	1.1	36.0	27.8	18.7	7.9	0.0
Householder	5.4	37.6	29.4	20.8	10.6	3.4
cavity03 with ILU(10^{-2})						
MGS	2.4	39.7	31.2	22.0	10.9	0.6
Householder	14.1	40.5	32.3	23.0	12.4	4.2
pde225 with ILU($3 \cdot 10^{-1}$)						
MGS	1.6	34.3	28.3	19.1	8.2	0.1
Householder	9.6	37.9	31.8	23.3	13.4	5.4

Table 2.1 – Percentage of memory saving using SZ-component-wise representation.

tor is artificially perturbed by a relative norm-wise perturbation. Next we consider two practical computational contexts where such norm-wise perturbations are encountered in some steps of the algorithms.

2.2.3.1 δ-norm-wise data storage

In this section we consider the situation where all the length n vectors involved in the algorithms are stored in a norm-wise δ-representation. That is, any vector $z \in \mathbb{R}^n$ is replaced by $\bar{z}$ so that

$$\frac{\|z - \bar{z}\|}{\|z\|} \leq \delta. \tag{2.9}$$

We display the convergence history of the norm-wise backward error for both considered GMRES variants, as well as the product of the backward error times the loss of orthogonality for MGS-GMRES, in Figure 2.4. It can be seen that the trends are very similar to those that can be observed in Figures 2.1. The same observations can be made: both the attainable backward error accuracy and the product $\eta_{AM,b}\|I - V^T V\|_F$ reach values close to δ.

2.2.3.2 Solution techniques using SZ compressed format

We report the convergence histories of MGS-GMRES and H-GMRES for a few values of the norm-wise compression control parameter δ in Figure 2.5. These results are the norm-wise counterparts of those displayed in Figure 2.3 for component-wise compression. It can be observed that the general trends are similar and that the attainable value of $\eta_{AM,b}$ always becomes close to δ. Because ensuring norm-wise relative error is easier for the SZ-compressor than ensuring component-wise relative error, the memory saving gains reported in Table 2.2 are larger than those in Table 2.1.

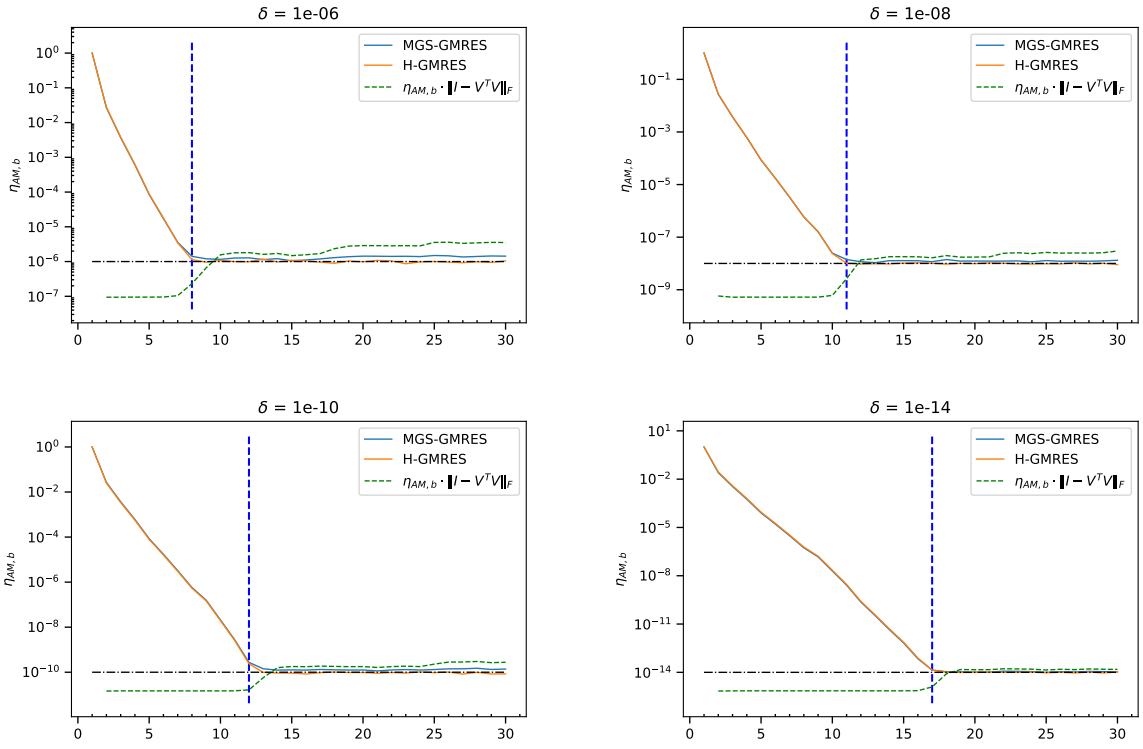


Figure 2.4 – Convergence history of $\eta_{AM,b}$ for gre-115 with ILU(10^{-1}) using δ-data normwise representation. The horizontal dashed black line indicates δ. The vertical dash blue line represents the first iteration where $\kappa(V_k) > 4/3$ in MGS-GMRES.

	δ					
	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
gre_115 with ILU(10^{-1})						
MGS	5.7	8.8	32.9	31.3	23.6	14.3
Householder	50.4	43.3	43.5	35.9	29.0	21.9
gre_185 with ILU(10^{-1})						
MGS	7.2	25.6	37.9	27.7	18.6	7.5
Householder	38.4	45.6	46.6	37.6	29.7	20.6
gre_343 with ILU(10^{-1})						
MGS	22.3	42.0	51.8	46.9	40.3	30.5
Householder	24.5	51.4	58.3	49.5	44.0	35.5
arc130 with ILU($8 \cdot 10^{-4}$)						
MGS	74.6	63.8	51.3	40.4	23.7	21.3
Householder	75.3	70.3	60.9	34.1	14.9	11.5
e05r0000 with ILU(10^{-2})						
MGS	5.6	8.4	40.7	32.3	21.2	11.3
Householder	4.4	40.1	44.7	34.9	24.5	15.7
e05r0400 with ILU(10^{-2})						
MGS	0.1	16.1	39.5	29.4	18.5	9.1
Householder	19.7	42.2	45.5	35.5	25.1	16.5
cavity03						
MGS	0.7	25.6	40.7	29.9	19.1	9.8
Householder	12.5	45.9	47.0	36.9	26.6	18.0
pde225 with ILU($3 \cdot 10^{-1}$)						
MGS	10.8	32.1	37.7	29.4	19.2	8.0
Householder	19.8	43.8	46.5	37.3	29.0	19.2

Table 2.2 – Percentage of memory saving using SZ-norm-wise representation.

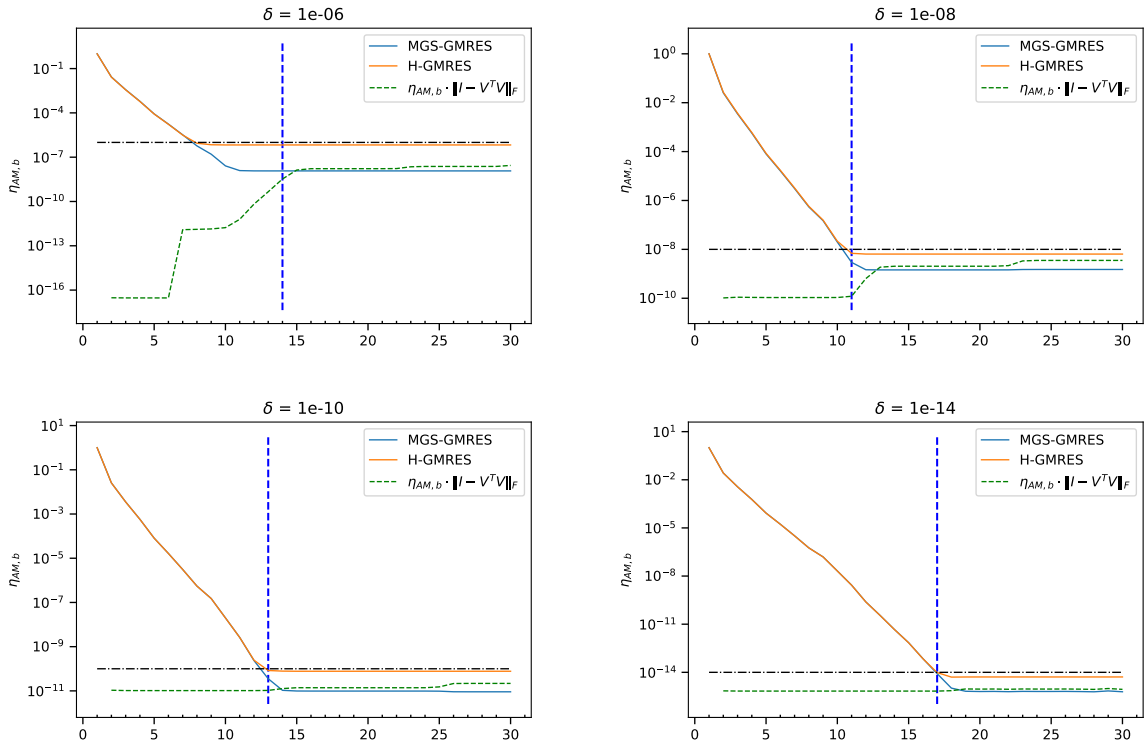


Figure 2.5 – Convergence history of $\eta_{AM,b}$ for e05r0000 with ILU(10^{-2}) using SZ-norm-wise representation of the Arnoldi basis and Householder reflectors. The horizontal dashed black line indicates δ. The vertical dash blue line represents the first iteration where $\kappa(V_k) > 4/3$ in MGS-GMRES.

2.3 Tensor Train GMRES

The second half of this chapter focuses on generalizing GMRES to tensor linear systems through the TT-formalism, see Section 1.3.2. To prevent memory issues, we introduce in the classical GMRES outline compression steps through the TT-rounding algorithm [108]. Under this choice, our TT-GMRES represents a perfect study case for our variable accuracy approach of GMRES with norm-wise perturbation.

After describing our TT-GMRES algorithm, we study how to solve simultaneously many tensor linear systems when the operator or the right-hand side depend on a parameter. In particular, we present some theoretical bounds that guarantee the numerical quality of the solution for a single parameter tensor linear system when it is extracted from the simultaneous solution. Finally, experiments highlighting the robustness of TT-GMRES and the tightness of these bounds are analysed.

2.3.1 Preconditioned GMRES in Tensor Train format

Assume $\mathbf{A} \in \mathbb{R}^{(n_1 \times n_1) \times \cdots \times (n_d \times n_d)}$ to be a tensor operator and $\mathbf{b} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ a tensor, then the general tensor linear system is

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.10}$$

with $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. Notice that by setting $d = 1$ we have the standard linear system from classical matrix computation. A possible way for solving (2.10) is using a tensor-extended version of GMRES. Since all the operations appearing in this iterative solver are feasible with the TT-formalism, we assume that all the objects are expressed in TT-format. The main drawback of this approach is the repetition of sums and contractions in the different loops, which leads to TT-rank growth and possible memory over-consumption. Therefore introducing compression steps in TT-GMRES is essential but particular attention should be paid to the choice of the rounding parameter to ensure that the prescribed GMRES tolerance ε can be reached. Our TT-GMRES algorithm is fully presented in Algorithm 9.

In Algorithm 9 and 10 there is an additional input parameter δ, i.e., the rounding accuracy. The TT-rounding algorithm at accuracy δ is applied to the result of the contraction between $\mathbf{A}$ and the last Krylov basis vector computed in Line 4, to the new Krylov basis vector after orthogonalization in Line 9 and to the updated iterative solution, Line 13. The purpose is to balance the rank growth due to the tensor contraction or sum, that occurred in the immediately previous step, with the rounding. As it will be observed in the numerical experiments of Section 2.3.3, the rounding accuracy δ has to be chosen lower or equal than the GMRES target accuracy ε.

2.3.2 Solution of parametric problems in Tensor Train format

In this section, we investigate the situation where either the tensor representation of the linear operator or the right-hand side has a mode related to a parameter that is discretized. In the case of the parametric linear operator, we are interested in the numerical

Algorithm 9 $\mathbf{x}$, `hasConverged` = TT-GMRES($\mathbf{A}$, $\mathbf{b}$, `m`, ε, δ)

input: $\mathbf{A}$, $\mathbf{b}$, `m`, ε, δ.
$\mathbf{r_0} = \mathbf{b}$, $\beta = \|\mathbf{r_0}\|$ and $\mathbf{v}_1 = (1/\beta)\mathbf{r}_0$
for $k = 1, \ldots,$ `maxit` **do**
    $\mathbf{w} =$ `TT-round`$(\mathbf{A}\mathbf{v}_k, \delta)$ ▷ MGS variant
    for $i = 1, \ldots, k$ **do**
        $\bar{H}(i,k) = \langle \mathbf{v}_i \, , \mathbf{w} \rangle$
        $\mathbf{w} = \mathbf{w} - \bar{H}_{i,k}\mathbf{v}_i$
    end for
    $\mathbf{w} =$ `TT-round`$(\mathbf{w}, \delta)$
    $\bar{H}(k+1,k) = \|\mathbf{w}\|$
    $\mathbf{v}_{k+1} = (1/\bar{H}(k+1,k))\mathbf{w}$
    $y_k = \text{argmin}_{y \in \mathbb{R}^k} \left\| \beta e_1 - \bar{H}_k y \right\|$ with $\bar{H}_k = H(:k+1, :k)$
    $\mathbf{x}_k =$ `TT-rounding`$\left(\sum_{j=1}^{k+1} y_k(j)\mathbf{v}_j, \delta\right)$
    if $\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x}_k) < \varepsilon$ **then**
        `hasConverged = True`
        break
    end if
end for
return: $\mathbf{x} = \mathbf{x}_k$, `hasConverged`

Algorithm 10 $\mathbf{x}$, `hasConverged` = Right-GMRES($\mathbf{A}$, $\mathbf{M}$, $\mathbf{b}$, $\mathbf{x_0}$, `m`, ε, δ)

input: $\mathbf{A}$, $\mathbf{M}$, $\mathbf{b}$, `m`, ε, δ.
`hasConverged = False`
$\mathbf{x} = \mathbf{x_0}$
while `not(hasConverged)` **do**
▷ Iterative refinement step with at most m GMRES iterations on $\mathbf{AM}$
    $\mathbf{r} =$ `TT-round`$(\mathbf{b} - \mathbf{A}\mathbf{x}, \delta)$
    $\mathbf{t}_k$, `hasConverged` = GMRES($\mathbf{AM}$, $\mathbf{r}$, `m`, ε, δ)
▷ Update the unpreconditionned with the computed correction
    $\mathbf{x} =$ `TT-round`$(\mathbf{x} + \mathbf{M}\mathbf{t_k}, \delta)$
end while
return: $\mathbf{x}$, `hasConverged`

quality of the computed solutions when we solve for all the parameters at once compared to the solution computed when the parametric systems are treated independently. In the case of the right-hand sides depending on a parameter, we investigate the links between the search space of TT-GMRES enabling the solution of all the right-hand sides at once and the spaces built by the GMRES solver on each right-hand side considered independently.

2.3.2.1 Parameter dependent linear operators

This subsection focuses on a specific type of parametric tensor operators expressed as $\mathbf{A}_\alpha = \mathbf{B}_0 + \alpha\mathbf{B}_1$ with $\alpha \in \mathbb{R}$ and $\mathbf{B}_0, \mathbf{B}_1$ two tensor operators of $\mathbb{R}^{(n_1\times n_1)\times\cdots\times(n_d\times n_d)}$. Assuming that α takes p different real values in the interval $[a, b]$, we define p linear systems of the form

$$\mathbf{A}_\ell \mathbf{y}_\ell = \mathbf{b}_\ell \tag{2.11}$$

where $\mathbf{A}_\ell = \mathbf{B}_0 + \alpha_\ell \mathbf{B}_1$, $\mathbf{b}_\ell \in \mathbb{R}^{n_1\times\cdots\times n_d}$, $\alpha_\ell \in [a, b]$ for every $\ell \in \{1, \ldots, p\}$. At this level, it is possible to choose between classically solving each system independently or solving them simultaneously in a higher order space defining the so-called "all-in-one" system. This latter system writes

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.12}$$

where $\mathbf{A} \in \mathbb{R}^{(p\times p)\times(n_1\times n_1)\times\cdots\times(n_d\times n_d)}$ such that

$$\mathbf{A}(h, \ell, i_1, j_1, \ldots, i_d, j_d) = \begin{cases} \mathbf{A}_\ell(i_1, j_1, \ldots, i_d, j_d) & \text{if} \quad h = \ell, \\ 0 & \text{if} \quad h \neq \ell, \end{cases} \tag{2.13}$$

and the right-hand side is $\mathbf{b} \in \mathbb{R}^{p\times n_1\times\cdots\times n_d}$ defined as

$$\mathbf{b}(\ell, i_1, \ldots, i_d) = \mathbf{b}_\ell(i_1, \ldots, i_d) \tag{2.14}$$

for $i_k, j_k \in \{1, \ldots, n_k\}$, $k \in \{1, \ldots, d\}$ and $\ell, h \in \{1, \ldots, p\}$. The tensor operator $\mathbf{A}$ writes in a compact format as

$$\mathbf{A} = \mathbb{I}_p \otimes \mathbf{B}_0 + \mathrm{diag}(\alpha_1, \ldots, \alpha_p) \otimes \mathbf{B}_1.$$

The (ℓ, ℓ)-th slice of $\mathbf{A}$ with respect to modes $(1, 2)$ is denoted

$$\mathbf{A}^{[\ell]} = \mathbf{B}_0 + \alpha_\ell \mathbf{B}_1 = \mathbf{A}_\ell \tag{2.15}$$

and similarly the ℓ-th slice of $\mathbf{b}$ with respect to the first mode is $\mathbf{b}^{[\ell]} = \mathbf{b}_\ell$ by construction. So that Equation (2.11) also writes

$$\mathbf{A}^{[\ell]}\mathbf{x}^{[\ell]} = \mathbf{b}^{[\ell]}$$

with $\mathbf{x}^{[\ell]} = \mathbf{y}_\ell$. It shows that once the "all-in-one" system, Equation (2.12), has been solved, the solution related to a specific parameter can be extracted as a slice of the "all-in-one" solution, obtaining an *extracted individual solution*. In other words, given the

k-th iterate $\mathbf{x}_k$ of the "all-in-one" system, the extracted individual solution for the ℓ-th problem is $\mathbf{x}_k^{[\ell]}$, i.e., the ℓ-th slice with respect to the first mode defined as

$$\mathbf{x}_k^{[\ell]} = \mathbf{x}_k(\ell, i_1, \dots, i_d).$$

In the following propositions, we investigate the relation between the backward error of the "all-in-one" system solution and the extracted individual one. The equalities given for the "all-in-one" system are clearly true if the tensor and the tensor operators are given in full format, but they hold also in TT-format.

In TT-format

As stated in Section 1.2, given a tensor $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ in TT-format with TT-cores $\underline{\mathbf{a}}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$, we denote by $\mathbf{a}^{[k,i_k]}$ the i_k-th slice with respect to mode k, which in TT-format is expressed as

$$\mathbf{a}^{[k,i_k]} = \underline{\mathbf{a}}_1 \cdots \underline{\mathbf{a}}_{k-1} \underline{A}_k(i_k) \underline{\mathbf{a}}_{k+1} \cdots \underline{\mathbf{a}}_d$$

with $\underline{A}_k(i_k) \in \mathbb{R}^{r_{k-1} \times r_k}$. Since henceforth we will take slices only with respect to the first mode, instead of writing $\mathbf{a}^{[1,i_1]}$ for the i_1-th slice on the first mode we will simply write $\mathbf{a}^{[i_1]}$. Similarly $\mathbf{A}^{[\ell]}$ denotes the (ℓ,ℓ)-th slice of $\mathbf{A} \in \mathbb{R}^{(p\times p)\times(n_1\times n_1)\times\cdots\times(n_d\times n_d)}$ with respect to the first two modes.

We start constructing the "all-in-one" system tensor linear operator. Let $\mathbf{C}, \mathbf{G} \in \mathbb{R}^{(n_1\times n_1)\times\cdots\times(n_d\times n_d)}$ be two TT-matrices with k-th TT-core $\underline{\mathbf{c}}_k \in \mathbb{R}^{r_k\times n_k\times n_k\times r_{k+1}}$ and $\underline{\mathbf{g}}_k \in \mathbb{R}^{q_k\times n_k\times n_k\times q_{k+1}}$ for $k \in \{1,\dots,d\}$ with $q_1 = r_1 = r_{d+1} = q_{d+1} = 1$, whose TT-expression is

$$\mathbf{C} = \underline{\mathbf{c}}_1 \cdots \underline{\mathbf{c}}_d \qquad \text{and} \qquad \mathbf{G} = \underline{\mathbf{g}}_1 \cdots \underline{\mathbf{g}}_d. \tag{2.16}$$

Given a diagonal matrix $D = \operatorname{diag}(\alpha_1, \dots, \alpha_p)$, we define $\mathbf{A} \in \mathbb{R}^{(p\times p)\times(n_1\times n_1)\times\cdots\times(n_d\times n_d)}$ as

$$\mathbf{A} = \mathbb{I}_p \otimes \mathbf{C} + D \otimes \mathbf{G}. \tag{2.17}$$

Then the expression of $\underline{\mathbf{a}}_k \in \mathbb{R}^{(r_k+q_k)\times n_k\times n_k\times(r_{k+1}+q_{k+1})}$ the k-th TT-core of $\mathbf{A}$ is

$$\underline{A}_k(i_k,j_k) = \begin{bmatrix} \underline{C}_k(i_k,j_k) & \mathbb{0} \\ \mathbb{0} & \underline{G}_k(i_k,j_k) \end{bmatrix} \qquad \text{and} \qquad \underline{A}_d(i_d,j_d) = \begin{bmatrix} \underline{C}_d(i_d,j_d) \\ \underline{G}_d(i_d,j_d) \end{bmatrix} \tag{2.18}$$

for every $i_k, j_k \in \{1,\dots,n_k\}$ and $k \in \{1,\dots,d\}$. The first TT-core $\underline{\mathbf{a}}_0 \in \mathbb{R}^{1\times p\times p\times 2}$ writes

$$\underline{A}_0(\ell,m) = \delta_{\ell,m} a_\ell \qquad \text{with} \qquad a_\ell^\top = \begin{bmatrix} 1 & \alpha_\ell \end{bmatrix} \tag{2.19}$$

with $\delta_{\ell,m}$ the Kronecker delta, for $\ell, m \in \{1,\dots,p\}$. The final TT-expression of $\mathbf{A}$ is

$$\mathbf{A} = \underline{\mathbf{a}}_0 \underline{\mathbf{a}}_1 \cdots \underline{\mathbf{a}}_d.$$

Remark now that $\mathbf{A}^{(\ell,m)}$ the (ℓ,m)-th slice with respect to mode 1 of $\mathbf{A}$ is

$$\mathbf{A}^{[\ell,m]} = \underline{A}_0(\ell,m)\underline{\mathbf{a}}_1 \cdots \underline{\mathbf{a}}_d = \delta_{\ell,m} a_\ell \underline{\mathbf{a}}_1 \cdots \underline{\mathbf{a}}_d.$$

If $\ell \neq m$, then $\mathbf{A}^{[\ell,m]} = \mathbb{0}$. On the other side, if ℓ and m are equal, then

$$\mathbf{A}^{[\ell,\ell]} = \mathbb{I}(\ell,\ell)a_\ell\, \underline{\mathbf{a}}_1 \cdots \underline{\mathbf{a}}_d = \underline{\mathbf{c}}_1 \cdots \underline{\mathbf{c}}_d + \alpha_\ell\, \underline{\mathbf{g}}_1 \cdots \underline{\mathbf{g}}_d = \mathbf{C} + \alpha_\ell\, \mathbf{G}.$$

We illustrate now the construction of the "all-in-one" system right-hand side from the p individual right-hand sides. Let $\mathbf{b}_\ell \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a TT-vector for every $\ell \in \{1, \ldots, p\}$ with TT-cores $\underline{\mathbf{b}}_{\ell,k} \in \mathbb{R}^{s_{\ell,k} \times n_k \times s_{\ell,k+1}}$ for every $k \in \{1, \ldots, d\}$ with $s_1 = s_{d+1} = 1$, i.e.,

$$\mathbf{b}_\ell = \underline{\mathbf{b}}_{\ell,1} \cdots \underline{\mathbf{b}}_{\ell,d} \tag{2.20}$$

and its $(i_1, \ldots, i_d)$ element writes

$$\mathbf{b}_\ell(i_1, \ldots, i_d) = \underline{B}_{\ell,1}(i_1) \cdots \underline{B}_{\ell,d}(i_d)$$

with $\underline{B}_{\ell,k}(i_k) \in \mathbb{R}^{s_{\ell,k} \times s_{\ell,k+1}}$, $\underline{B}_{\ell,1}(i_1) \in \mathbb{R}^{1 \times s_{\ell,2}}$ and $\underline{B}_{\ell,d}(i_d) \in \mathbb{R}^{s_{\ell,d} \times 1}$. For simplicity we impose $s_k = s_{\ell,k}$ for every $\ell \in \{1, \ldots, p\}$ and $k \in \{1, \ldots, d\}$.

Remark 2.3.1. This assumption on the TT-rank of $\mathbf{b}_\ell$ is not binding. Indeed setting $s_k = \max_{h \in \{1,\ldots,p\}} s_{h,k}$, then each core tensor $\underline{\mathbf{b}}_{h,k}$ of mode sizes $(s_{h,k}, n_h, s_{h,k+1})$ can be extended with zeros to (s_k, n_h, s_{k+1})

We want to construct a tensor $\mathbf{b} \in \mathbb{R}^{p \times n_1 \times \cdots \times n_d}$ such that its ℓ-th slice with respect to the first mode of $\mathbf{b}$ is $\mathbf{b}_\ell$, i.e., $\mathbf{b}^{[\ell]} = \mathbf{b}_\ell$. As consequence, the k-th TT-core of $\mathbf{b}$ is $\underline{\mathbf{b}}_k \in \mathbb{R}^{ps_k \times n_k \times ps_{k+1}}$ such that

$$\underline{B}_k(i_k) = \begin{bmatrix} \underline{B}_{1,k}(i_k) & & \\ & \ddots & \\ & & \underline{B}_{p,k}(i_k) \end{bmatrix} \in \mathbb{R}^{ps_k \times ps_{k+1}}$$

for $k \in \{1, \ldots, d-1\}$, while $\underline{\mathbf{b}}_d \in \mathbb{R}^{ps_{d-1} \times n_d \times 1}$ and $\underline{\mathbf{b}}_0 \in \mathbb{R}^{1 \times p \times ps_1}$ are

$$\underline{B}_d(i_d) = \begin{bmatrix} \underline{B}_{1,d}(i_d) \\ \vdots \\ \underline{B}_{p,d}(i_d) \end{bmatrix} \in \mathbb{R}^{ps_d \times 1} \qquad \text{and} \qquad \underline{B}_0(\ell) = \begin{bmatrix} 0 \cdots 1 \cdots 0 \end{bmatrix} \in \mathbb{R}^{1 \times ps_1}$$

with the ℓ-th component of $\underline{B}_0(\ell)$ being the only non-zero element. The TT-expression of $\mathbf{b}$ is

$$\mathbf{b} = \underline{\mathbf{b}}_0 \underline{\mathbf{b}}_1 \cdots \underline{\mathbf{b}}_d. \tag{2.21}$$

By construction, we have that the ℓ-th slice of $\mathbf{b}$ with respect to mode 1 is

$$\begin{aligned} \mathbf{b}^{[\ell]} &= \underline{B}_0(\ell) \underline{\mathbf{b}}_1 \cdots \underline{\mathbf{b}}_d \\ &= \underline{\mathbf{b}}_{\ell,1} \cdots \underline{\mathbf{b}}_{\ell,d} \\ &= \mathbf{b}_\ell. \end{aligned}$$

Let consider $\mathbf{A}$ and $\mathbf{b}$ as defined in Equation (2.17) and (2.21), given $\mathbf{x} \in \mathbb{R}^{p \times n_1 \times \cdots \times n_d}$ and define the new vector

$$\mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{b}.$$

We want to prove that $\mathbf{r}^{[\ell]}$ the ℓ-th slice with respect to the first mode of $\mathbf{r}$ is equal to the difference of the ℓ-th slices, i.e.,

$$\mathbf{r}^{[\ell]} = \mathbf{A}^{[\ell,\ell]}\mathbf{x}^{[\ell]} - \mathbf{b}_\ell = (\mathbf{C} + \alpha_\ell \mathbf{G})\mathbf{x}^{[\ell]} - \mathbf{b}_\ell \tag{2.22}$$

since the (ℓ,ℓ)-th slice of $\mathbf{A}$ is $\mathbf{C} + \alpha_\ell\mathbf{G}$ for every $\ell \in \{1,\dots,p\}$. Remark that the ℓ-th slice of $\mathbf{b}$ is $\mathbf{b}_\ell$ by construction. As consequence, the Equation (2.22) is true if we show that the ℓ-th slice of the contraction between $\mathbf{A}$ and $\mathbf{x}$ is equal to the contraction of their ℓ-th slices, i.e.,

$$(\mathbf{A}\mathbf{x})^{[\ell]} = \mathbf{A}^{[\ell,\ell]}\mathbf{x}^{[\ell]}.$$

Lemma 2.3.2. *Given* $\mathbf{A}$, $\mathbf{C}$, $\mathbf{G}$ *as in Equations* (2.16) *and* (2.17), *let* $\mathbf{x} \in \mathbb{R}^{p\times n_1\times\cdots\times n_d}$ *be a* $(d+1)$*-order tensor. Then the* ℓ*-th slice of* $\mathbf{y} = \mathbf{A}\mathbf{x}$ *is equal to the product of their* ℓ*-th slices, i.e.*

$$(\mathbf{A}\mathbf{x})^{[\ell]} = \mathbf{A}^{[\ell,\ell]}\mathbf{x}^{[\ell]} = (\mathbf{C} + \alpha_\ell\mathbf{G})\mathbf{x}^{[\ell]}.$$

Define $\mathbf{w} = (\mathbf{C} + \alpha_\ell\mathbf{G})\mathbf{x}^{[\ell]}$, *then* $\mathbf{y}^{[\ell]}$ *the* ℓ*-th slice of* $\mathbf{y}$ *with respect to mode* 1 *is equal to* $\mathbf{w}$, *i.e.,*

$$\mathbf{y}^{[\ell]} = \mathbf{w}.$$

Proof. Let $\mathbf{y} = \mathbf{A}\mathbf{x}$, then by definition the $(i_1,\dots,i_d)$ element of the ℓ-th slice with respect to the first mode is

$$\begin{aligned}\mathbf{y}^{[\ell]}(i_1,\dots,i_d) &= \left(\mathbf{A}\mathbf{x}^{[\ell]}\right)(i_1,\dots,i_d) = \\ &= \sum_{h,j_1,\dots,j_d=1}^{p,n_1,\dots,n_d} \mathbf{A}(\ell,h,i_1,j_1,\dots,i_d,j_d)\mathbf{x}(h,j_1,\dots,j_d).\end{aligned}$$

By construction $\mathbf{A}(\ell,h,i_1,j_1,\dots,i_d,j_d)$ is nonzero, only if ℓ and h are equal, leading to

$$\begin{aligned}\mathbf{y}^{[\ell]}(i_1,\dots,i_d) &= \sum_{j_1,\dots,j_d=1}^{n_1,\dots,n_d} \mathbf{A}(\ell,\ell,i_1,j_1,\dots,i_d,j_d)\mathbf{x}(\ell,j_1,\dots,j_d) \\ &= \sum_{j_1,\dots,j_d=1}^{n_1,\dots,n_d} \left(\mathbf{A}^{[\ell]}\right)(i_1,j_1,\dots,i_d,j_d)\left(\mathbf{x}^{[\ell]}\right)(j_1,\dots,j_d).\end{aligned}$$

by the definition of first mode slides. As already observed the (ℓ,ℓ)-th slice of $\mathbf{A}$ with respect to the first mode is equal to $\mathbf{C} + \alpha_\ell\mathbf{G}$. Thus, the previous equation becomes

$$\mathbf{y}^{[\ell]}(i_1,\dots,i_d) = \sum_{j_1,\dots,j_d=1}^{n_1,\dots,n_d} (\mathbf{C} + \alpha_\ell\mathbf{G})(i_1,j_1,\dots,i_d,j_d)\left(\mathbf{x}^{[\ell]}\right)(j_1,\dots,j_d),$$

i.e., the thesis. □

By the result of Lemma 2.3.2, we get that the ℓ-th slice of $\mathbf{A}\mathbf{x}$ writes

$$(\mathbf{A}\mathbf{x})^{[\ell]} = \mathbf{C}\mathbf{x}^{[\ell]}.$$

Therefore the ℓ-th slice of $\mathbf{r}$ is

$$\mathbf{r}^{[\ell]} = \mathbf{C}\mathbf{x}^{[\ell]} - \mathbf{b}^{[\ell]}.$$

As a conclusive result of this construction, we want to show that

$$||\mathbf{r}||^2 = \sum_{\ell=1}^{p} ||\mathbf{r}^{[\ell]}||^2.$$

Lemma 2.3.3. *Given* $\mathbf{s} \in \mathbb{R}^{n_0 \times n_1 \times \cdots \times n_d}$ *and its* i_0*-th slice with respect to the first mode* $\mathbf{s}^{(i_0)}$ *then*

$$||\mathbf{s}||^2 = \sum_{i_0=1}^{n_0} ||\mathbf{s}^{[i_0]}||^2.$$

Proof. By definition, the squared Frobenious norm of $\mathbf{s}$ is

$$\|\mathbf{s}\|^2 = \sum_{i_0,i_1,\ldots,i_d=1}^{n_0,n_1,\ldots,n_d} \mathbf{s}^2(i_0, i_1, \ldots, i_d) = \sum_{i_0=1}^{n_0} \Big(\sum_{i_1,\ldots,i_d=1}^{n_1,\ldots,n_d} \mathbf{s}^2(i_0, i_1, \ldots, i_d) \Big).$$

By the definition of slice, the last equation can be written as

$$\|\mathbf{s}\|^2 = \sum_{i_0=1}^{n_0} \Big(\sum_{i_1,\ldots,i_d=1}^{n_1,\ldots,n_d} \mathbf{s}^2(i_0, i_1, \ldots, i_d) \Big) = \sum_{i_0=1}^{n_0} \Big(\sum_{i_1,\ldots,i_d=1}^{n_1,\ldots,n_d} \Big(\mathbf{s}^{[i_0]}\Big)^2 (i_1, \ldots, i_d) \Big),$$

but by the definition of squared Frobenious norm of $\mathbf{s}^{[i_0]}$, we obtain

$$\|\mathbf{s}\|^2 = \sum_{i_0=1}^{n_0} \Big(\sum_{i_1,\ldots,i_d=1}^{n_1,\ldots,n_d} \Big(\mathbf{s}^{[i_0]}\Big)^2 (i_1, \ldots, i_d) \Big) = \sum_{i_0=1}^{n_0} ||\mathbf{s}^{[i_0]}||^2,$$

i.e., the thesis. □

By the result of Lemma 2.3.3, we have

$$||\mathbf{r}||^2 = \sum_{\ell=1}^{p} ||\mathbf{r}^{[\ell]}||^2.$$

Theoretical bounds

Once the "all-in-one" system has been completely described in its construction in TT-format, we present some bounds that enable us to tune the convergence threshold when solving for multiple parameters while guaranteeing a prescribed quality for the individual extracted solutions. In particular, the bound given by Equation (2.23) in Proposition 2.3.4 shows that if a certain accuracy ε is expected for the extracted individual solution in terms of the backward error in (2.4), a more stringent convergence threshold should be used for the "all-in-one" system solution that should be set to $\varepsilon/\sqrt{p}$.

Proposition 2.3.4. *Let the "all-in-one" operator* $\mathbf{A} \in \mathbb{R}^{(p\times p)\times(n_1\times n_1)\times\ldots(n_d\times n_d)}$ *and right-hand side* $\mathbf{b} \in \mathbb{R}^{p\times n_1\times\cdots\times n_d}$ *be as in Equations* (2.13) *and* (2.14) *respectively, we consider the "all-in-one" system*

$$\mathbf{A}\mathbf{x} = \mathbf{b}.$$

Let $\mathbf{A}_\ell \in \mathbb{R}^{(n_1\times n_1)\times\cdots\times(n_d\times n_d)}$ *be the tensor operator as in Equation* (2.15) *and let* $\mathbf{b}_\ell \in \mathbb{R}^{n_1\times\cdots\times n_d}$ *be a tensor such that* $||\mathbf{b}_\ell|| = 1$*, that defines the individual linear systems*

$$\mathbf{A}_\ell\mathbf{y}_\ell = \mathbf{b}_\ell$$

with $\mathbf{A}_\ell = \mathbf{A}^{[\ell]}$ *and* $\mathbf{b}_\ell = \mathbf{b}^{[\ell]}$ *for every* $\ell \in \{1,\ldots,p\}$*. Let* $\mathbf{x}_k$ *denote the* k*-th "all-in-one" iterate, we have*

$$\eta_{\mathbf{b}}(\mathbf{x}_k)\sqrt{p} \geq \eta_{\mathbf{b}_\ell}(\mathbf{x}_k^{[\ell]}) \tag{2.23}$$

for $\ell \in \{1,\ldots,p\}$*.*

Proof. For the sake of simplicity we use $\eta_{\mathbf{b}}$ and $\eta_{\mathbf{b}_\ell}$ squared throughout the proof and discard the subscript of the k-th "all-in-one" iterate. The quantity $\eta^2_{\mathbf{b}_\ell}(\mathbf{x}^{[\ell]})$ explicitly gets

$$\eta^2_{\mathbf{b}_\ell}(\mathbf{x}^{[\ell]}) = \frac{\left\|\mathbf{A}_\ell\mathbf{x}^{[\ell]} - \mathbf{b}_\ell\right\|^2}{\left\|\mathbf{b}_\ell\right\|^2}$$

while $\eta^2_{\mathbf{b}}(\mathbf{x})$ is

$$\eta^2_{\mathbf{b}}(\mathbf{x}) = \frac{\left\|\mathbf{A}\mathbf{x} - \mathbf{b}\right\|^2}{\left\|\mathbf{b}\right\|^2}. \tag{2.24}$$

Thanks to the diagonal structure of $\mathbf{A}$ and the Frobenius norm definition, Equation (2.24) writes

$$\eta^2_{\mathbf{b}}(\mathbf{x}) = \frac{\sum_{\ell=1}^n \left\|\left(\mathbf{A}\mathbf{x} - \mathbf{b}\right)^{[\ell]}\right\|^2}{\sum_{j=1}^p \left\|\mathbf{b}^{[j]}\right\|^2} = \frac{\sum_{\ell=1}^n \left\|\mathbf{A}_\ell\mathbf{x}^{[\ell]} - \mathbf{b}_\ell^2\right\|}{\sum_{j=1}^p \left\|\mathbf{b}_j\right\|^2} = \frac{\sum_{\ell=1}^p \eta^2_{\mathbf{b}_\ell}(\mathbf{x}^{[\ell]})}{p} \tag{2.25}$$

since $||\mathbf{b}||^2 = \sum_{j=1}^n ||\mathbf{b}_j||^2 = p$. From the square root of both sides of this last equation, the result follows. □

For the backward error based on perturbation of both the linear operator and the right-hand side defined by (2.3), a similar result can be derived. While informative this result has a lower practical interest as the term $\rho_\ell(\mathbf{x}_k)$ in (2.26) depends on the solution; so defining the convergence threshold for the 'all-in-one' solution to guarantee the individual backward error requires some a priori information on the solution norms.

Proposition 2.3.5. *With the same hypothesis and notation as for Proposition 2.3.4 for* $\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x}_k)$ *and* $\eta_{\mathbf{A}_\ell,\mathbf{b}_\ell}(\mathbf{x}_k^{[\ell]})$ *associated linear systems* $\mathbf{A}\mathbf{x} = \mathbf{b}$ *and* $\mathbf{A}_\ell\mathbf{y}_\ell = \mathbf{b}_\ell$ *respectively, for every* $\ell \in \{1,\ldots,p\}$*, we have*

$$\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x}_k)\,\rho_\ell(\mathbf{x}_k) \geq \eta_{\mathbf{A}_\ell,\mathbf{b}_\ell}(\mathbf{x}_k^{[\ell]}) \qquad \textit{where} \qquad \rho_\ell(\mathbf{x}_k) = \frac{\|\mathbf{A}\|_2\|\mathbf{x}_k\| + \sqrt{p}}{\left\|\mathbf{A}_\ell\mathbf{x}_k^{[\ell]}\right\| + 1} \tag{2.26}$$

with $\mathbf{x}_k$ *the* k*-th "all-in-one" iterate and* $\mathbf{x}_k^{[\ell]}$ *its* ℓ*-th slice with respect to mode* 1*.*

Proof. For the sake of simplicity, as previously, the subscript of the k-th "all-in-one" iterate is dropped. The quantity $\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x})$ explicitly writes

$$\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x}) = \frac{\|\mathbf{A}\mathbf{x}-\mathbf{b}\|}{\|\mathbf{A}\|_2\|\mathbf{x}\|+\|\mathbf{b}\|}.$$

If the previous equation is multiplied equivalently by $\eta_{\mathbf{b}}(\mathbf{x})$, it gets

$$\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x}) = \frac{\|\mathbf{A}\mathbf{x}-\mathbf{b}\|}{\|\mathbf{A}\|_2\|\mathbf{x}\|+\|\mathbf{b}\|}\frac{\eta_{\mathbf{b}}(\mathbf{x})}{\eta_{\mathbf{b}}(\mathbf{x})} = \frac{\|\mathbf{b}\|}{\|\mathbf{A}\|_2\|\mathbf{x}\|+\|\mathbf{b}\|}\eta_{\mathbf{b}}(\mathbf{x}) = \frac{\sqrt{p}}{\|\mathbf{A}\|_2\|\mathbf{x}\|+\sqrt{p}}\eta_{\mathbf{b}}(\mathbf{x}) \quad (2.27)$$

by the definition of $\eta_{\mathbf{b}}(\mathbf{x})$ and $\|\mathbf{b}\| = \sqrt{p}$. Similarly $\eta_{\mathbf{A}_\ell,\mathbf{b}_\ell}(\mathbf{x}^{[\ell]})$ is expressed in function of $\eta_{\mathbf{b}_\ell}(\mathbf{x}^{[\ell]})$ as

$$\eta_{\mathbf{A}_\ell,\mathbf{b}_\ell}(\mathbf{x}^{[\ell]}) = \frac{\|\mathbf{b}_\ell\|}{\|\mathbf{A}_\ell\|_2\|\mathbf{x}^{[\ell]}\|+\|\mathbf{b}_\ell\|}\eta_{\mathbf{b}_\ell}(\mathbf{x}^{[\ell]}) = \frac{1}{\|\mathbf{A}_\ell\|_2\|\mathbf{x}^{[\ell]}\|+1}\eta_{\mathbf{b}_\ell}(\mathbf{x}^{[\ell]}) \quad (2.28)$$

since $\|\mathbf{b}_\ell\| = 1$. Multiplying each side of Equation (2.27) by $(\|\mathbf{A}\|_2\|\mathbf{x}\| + \sqrt{p})$, it follows

$$\Big(\|\mathbf{A}\|_2\|\mathbf{x}\| + \sqrt{p}\Big)\eta_{\mathbf{A},\mathbf{b}} = \eta_{\mathbf{b}}\sqrt{p}.$$

Thanks to the result of Proposition 2.3.4, we have

$$\Big(\|\mathbf{A}\|_2\|\mathbf{x}\| + \sqrt{p}\Big)\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x}) = \eta_{\mathbf{b}}(\mathbf{x})\sqrt{p} \geq \eta_{\mathbf{b}_\ell}(\mathbf{x}^{[\ell]}) = \Big(\|\mathbf{A}_\ell\|_2\Big\|\mathbf{x}^{[\ell]}\Big\| + 1\Big)\eta_{\mathbf{A}_\ell,\mathbf{b}_\ell}(\mathbf{x}^{[\ell]}) \quad (2.29)$$

from Equation (2.28). Dividing both sides of Equation (2.29) by $\|\mathbf{A}_\ell\|_2\Big\|\mathbf{x}^{[\ell]}\Big\|+1$, it becomes

$$\frac{\|\mathbf{A}\|_2\|\mathbf{x}\| + \sqrt{p}}{\|\mathbf{A}_\ell\mathbf{x}^{[\ell]}\| + 1}\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x}) \geq \eta_{\mathbf{A}_\ell,\mathbf{b}_\ell}(\mathbf{x}^{[\ell]}) \quad (2.30)$$

since $\|\mathbf{A}_\ell\|_2\Big\|\mathbf{x}^{[\ell]}\Big\| \geq \Big\|\mathbf{A}_\ell\mathbf{x}^{[\ell]}\Big\|$ by the definition of the L2 norm. □

2.3.2.2 Parameter dependent right-hand sides

We consider a particular case of this "all-in-one" approach. We intend to solve p tensor linear systems with the same multilinear operator and different right-hand sides. Given a linear tensor operator $\mathbf{A}_0 \in \mathbb{R}^{(n_1\times n_1)\times\cdots\times(n_d\times n_d)}$, we define the ℓ-th tensor linear system as

$$\mathbf{A}_0\mathbf{y}_\ell = \mathbf{b}_\ell \quad (2.31)$$

with $\mathbf{b}_\ell \in \mathbb{R}^{n_1\times\cdots\times n_d}$ for every $\ell \in \{1,\ldots,p\}$. To solve simultaneously all the right-hand sides expressed in Equation (2.31), we repeat the construction introduced in Subsection 2.3.2, except that $\mathbf{A}_0$ is repeated on the 'diagonal' of tensor linear operator $\mathbf{A}$ defined in Equation (2.13). Thanks to the tensor properties, the tensor operator $\mathbf{A} \in \mathbb{R}^{(p\times p)\times(n_1\times n_1)\times\cdots\times(n_d\times n_d)}$ writes

$$\mathbf{A} = \mathbb{I}_p \otimes \mathbf{A_0}$$

so that $\mathbf{A}^{[\ell]} = \mathbf{A}_0$ for every $\ell \in \{1, \dots, p\}$. The right-hand side $\mathbf{b}$ is defined similarly to the previous section, that is $\mathbf{b}^{[\ell]} = \mathbf{b}_\ell$. If the initial guess is $\mathbf{x}_0 \in \mathbb{R}^{p \times n_1 \times \cdots \times n_d}$ equal to the null tensor, then at the k-th iteration TT-GMRES minimizes with respect to $\mathbf{x}$ the norm of the residual $\mathbf{A}\mathbf{x} - \mathbf{b}$ on the space

$$\mathcal{K}_k(\mathbf{A}, \mathbf{b}) = \operatorname{span}\Big\{\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \dots, \mathbf{A}^{k-1}\mathbf{b}\Big\},$$

i.e., we seek a tensor $\mathbf{x}_k \in \mathcal{K}_k(\mathbf{A}, \mathbf{b})$ such that

$$\mathbf{x}_k = \underset{\mathbf{x} \in \mathcal{K}_k(\mathbf{A}, \mathbf{b})}{\operatorname{argmin}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|.$$

Due to the diagonal structure of $\mathbf{A}$, the Frobenius norm of $\mathbf{r}_k = \mathbf{A}\mathbf{x}_k - \mathbf{b}$ is naturally written as follows

$$||\mathbf{r}_k||^2 = \sum_{\ell=1}^{p} ||\mathbf{b}_\ell - \mathbf{A}_0 \mathbf{x}_k^{[\ell]}||^2$$

with $\mathbf{x}_k^{[\ell]}$ being the ℓ-th slice with respect to the first mode of $\mathbf{x}_k$, as in the previous sections. Thanks to the diagonal structure of $\mathbf{A}$, we have that the ℓ-th slice of the Krylov basis vector $\mathbf{A}^h\mathbf{b}$ with respect to the first mode is $\mathbf{A}_0^h \mathbf{b}_\ell$. Consequently the ℓ-th slices of the basis vectors of $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$ span the Krylov space $\mathcal{K}_k(\mathbf{A}_0, \mathbf{b}_\ell)$. It means that the individual solutions defined by the slices $\mathbf{x}_k^{[\ell]}$ of the iterate from the "all-in-one" TT-GMRES scheme lie in the same space as the $\mathbf{y}_{\ell,k}$ generated by TT-GMRES applied to the individual systems $\mathbf{A}_0\mathbf{y}_\ell = \mathbf{b}_\ell$ with $\mathbf{y}_{\ell,0} = 0$. While the two iterates belong to the same space, they are different since the former, $\mathbf{x}_k^{[\ell]}$, is built by minimizing the residual norm of $\mathbf{A}\mathbf{x} - \mathbf{b}$ over $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$ and the latter, $\mathbf{y}_{\ell,k}$, by minimizing the residual norm of $\mathbf{A}_0\mathbf{y}_\ell = \mathbf{b}_\ell$ over $\mathcal{K}_k(\mathbf{A}_0, \mathbf{b}_\ell)$. If we neglect the effect of the rounding, one can expect that

$$\left\|\mathbf{b}_\ell - \mathbf{A}_0 \mathbf{x}_k^{[\ell]}\right\| \geq \|\mathbf{b}_\ell - \mathbf{A}_0 \mathbf{y}_{\ell,k}\|.$$

Remark 2.3.6. We notice that a block TT-GMRES method could also be defined for the solution of such multiple right-hand side problems. In that situation, each individual residual norm would be minimized over the same space spanned by the sum of the individual Krylov space. This would be somehow the dual approach to the one described above, where we minimize the sum of the residual norms on each individual Krylov space.

Regarding the numerical quality of the extracted solution compared to the individually computed solution, the bound stated in Proposition 2.3.4 is still true. As in the previous section an informative, but with lower practical interest, bound similar Proposition 2.3.5 can be derived.

Proposition 2.3.7. *Under the hypothesis of Proposition 2.3.5, if* $\mathbf{A} = \mathbb{I}_p \otimes \mathbf{A}_0$*, then for* $\eta_{\mathbf{A},\mathbf{v}}(\mathbf{x}_k)$ *and* $\eta_{\mathbf{A}_\ell,\mathbf{b}_\ell}(\mathbf{x}_k^{[\ell]})$ *associated with the linear systems* $\mathbf{A}\mathbf{x} = \mathbf{b}$ *and* $\mathbf{A}_0\mathbf{y}_\ell = \mathbf{b}_\ell$ *respectively, for every* $\ell \in \{1, \dots, p\}$ *the following inequality holds*

$$\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x}_k)\, \psi_\ell(\mathbf{x}_k) \geq \eta_{\mathbf{A}_\ell,\mathbf{b}_\ell}(\mathbf{x}_k^{[\ell]}) \qquad \textit{where} \qquad \psi_\ell(\mathbf{x}_k) = \frac{\|\mathbf{x}_k\| + \sqrt{p}/\|\mathbf{A}_0\|_2}{\left\|\mathbf{x}_k^{[\ell]}\right\| + 1/\|\mathbf{A}_0\|_2}. \tag{2.32}$$

Proof. The result follows from the thesis of Proposition 2.3.5, since $\|\mathbf{A}\|_2 = \|\mathbf{A}_0\|_2$ □

Corollary 2.3.8. *Given a sequence of iterative solutions* $\{\mathbf{x}_k\}_{k\in\mathbb{N}}$ *and a value* ν*, if there exists a* $k^*_\ell \in \mathbb{N}$ *such that* $|\,||\mathbf{A}_\ell \mathbf{x}_k^{[\ell]}|| - 1| \leq \nu$ *for every* $k \geq k^*_\ell$*, then*

$$\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x}_k)\,\rho^*(\mathbf{x}_k) \geq \eta_{\mathbf{A}_\ell,\mathbf{b}_\ell}(\mathbf{x}_k^{[\ell]}) \qquad \textit{where} \qquad \rho^*(\mathbf{x}_k) = \frac{\|\mathbf{A}\|_2\|\mathbf{x}_k\| + \sqrt{p}}{2-\nu} \tag{2.33}$$

for every $\ell \in \{1,\dots,p\}$ *and for every* $k \in \mathbb{N}$ *such that* $k \geq k^{**}$ *where* $k^{**} = \max k^*_\ell$*.*

Corollary 2.3.9. *Under the hypothesis of Corollary 2.3.8, if there exists a* $k^\dagger \in \mathbb{N}$ *such that* $\left\|\mathbf{x}_k^{[\ell]}\right\| \leq \|\mathbf{A}^{-1}\|_2\sqrt{p}$ *for every* $k \geq k^\dagger$*, then*

$$\eta_{\mathbf{A},\mathbf{v}}(\mathbf{x}_k)\,\rho^\dagger \geq \eta_{\mathbf{A}_\ell,\mathbf{v}_\ell}(\mathbf{x}_k^{[\ell]}) \qquad \textit{where} \qquad \rho^\dagger = \frac{\sqrt{p}}{2-\nu}(1+\kappa_2(\mathbf{A})) \tag{2.34}$$

with $\kappa_2(\mathbf{A}) = ||\mathbf{A}||_2||\mathbf{A}^{-1}||_2$ *for every* $\ell \in \{1,\dots,p\}$ *and for every* $k \in \mathbb{N}$ *such that* $k \geq k^\ddagger$ *where* $k^\ddagger = \max\{k^{**}, k^\dagger\}$ *with* k^{**} *given in Corollary 2.3.8.*

2.3.3 Numerical experiments

In this section, we investigate the numerical behaviour of the TT-GMRES solver for linear problems with increasing order as it naturally arises in some partial differential equation (PDE) studies. We start by illustrating how the TT-operators of our numerical examples are directly constructed in TT-format, thanks to their peculiarity. For all the examples, we illustrate numerical concerns related to the algorithm convergence and computational costs, with a focus on memory growth and memory saving.

The linear operators of the main problems, addressed in the following sections, are *Laplace-like* operators. The Laplace-like tensor operator $\mathbf{A} \in \mathbb{R}^{(n_1\times m_1)\cdots\times(n_d\times m_d)}$ is the sum of operators written as

$$\begin{aligned}\mathbf{A} =& M_1 \otimes R_2 \otimes R_3 \otimes \cdots \otimes R_{d-2} \otimes R_{d-1} \otimes R_d \\ &+ L_1 \otimes M_2 \otimes R_3 \otimes \cdots \otimes R_{d-2} \otimes R_{d-1} \otimes R_d \\ &+ \cdots + L_1 \otimes L_2 \otimes L_3 \otimes \cdots \otimes L_{d-2} \otimes M_{d-1} \otimes R_d \\ &+ L_1 \otimes L_2 \otimes L_3 \otimes \cdots \otimes L_{d-2} \otimes L_{d-1} \otimes M_d\end{aligned} \tag{2.35}$$

with $L_k, M_k, R_k \in \mathbb{R}^{n_k\times m_k}$ for every $k \in \{1,\dots,d\}$. As relevant property, these linear operators are expressed in TT-format with TT-rank 2, i.e.,

$$\mathbf{A} = \begin{bmatrix} L_1 & M_1 \end{bmatrix} \otimes \begin{bmatrix} L_2 & M_2 \\ 0 & R_2 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} L_{d-1} & M_{d-1} \\ 0 & R_{d-1} \end{bmatrix} \otimes \begin{bmatrix} M_d \\ R_d \end{bmatrix} \tag{2.36}$$

as proved in [81, Lemma 5.1]. Remarking that the general expression of the discrete d-order Laplacian on a uniform grid of n points in each direction is

$$\boldsymbol{\Delta}_d = \Delta_1 \otimes \mathbb{I}_n \otimes \cdots \otimes \mathbb{I}_n + \cdots + \mathbb{I}_n \otimes \mathbb{I}_n \otimes \cdots \otimes \Delta_1$$

where $\mathbb{I}_n$ is the identity matrix of size n and $\Delta_1 \in \mathbb{R}^{n\times n}$ is the discrete matrix Laplacian using the central-point finite difference scheme with discretization step $h = \frac{1}{n+1}$, i.e.,

$$\Delta_1 = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & \dots & 1 & -2 \end{bmatrix}.$$

Then the TT-expression of $\mathbf{\Delta}_d$ is

$$\mathbf{\Delta}_d = \begin{bmatrix} \mathbb{I}_n & \Delta_1 \end{bmatrix} \otimes \begin{bmatrix} \mathbb{I}_n & \Delta_1 \\ \mathbb{0} & \mathbb{I}_n \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \mathbb{I}_n & \Delta_1 \\ \mathbb{0} & \mathbb{I}_n \end{bmatrix} \otimes \begin{bmatrix} \Delta_1 \\ \mathbb{I}_n \end{bmatrix}. \tag{2.37}$$

To solve linear systems efficiently, we consider an approximation of the inverse of the discrete Laplacian operator, $\mathbf{M}$, as a preconditioner [61, 62]. This operator writes

$$\mathbf{M} = \sum_{k=-q}^{q} c_k \exp(-t_k\Delta_1) \otimes \cdots \otimes \exp(-t_k\Delta_1) \tag{2.38}$$

where $c_k = \xi t_k$, $t_k = \exp(k\xi)$ and $\xi = \frac{\pi}{q}$. Thanks to the previously stated property of the sum of TT-vectors, we conclude that the TT-ranks of $\mathbf{M}$ will be at most $2q+1$. In Section 2.3.3 we consider the linear system $\mathbf{Ax} = \mathbf{b}$ and to speed up its convergence we apply the preconditioner TT-matrix $\mathbf{M}$, effectively solving $\mathbf{AMt} = \mathbf{b}$. The preconditioner TT-matrix $\mathbf{M}$ is always computed by q addends equal to a quarter of the grid step size. To keep the TT-rank of the preconditioner small, we choose to round it to 10^{-2}. The choice of the number of addends and the rounding compression is further discussed in Section 2.3.3.1.

To evaluate the converge of the TT-GMRES at the k-th iteration, we display in Section 2.3.3 the stopping criterion $\eta_{\mathbf{AM},\mathbf{b}}$, that is

$$\eta_{\mathbf{AM},\mathbf{b}}(\mathbf{t}_k) = \frac{\|\mathbf{AMt}_k - \mathbf{b}\|}{\|\mathbf{AM}\|_2\|\mathbf{t}_k\| + \|\mathbf{b}\|} \tag{2.39}$$

with $\mathbf{t}_k$ the preconditioned approximated solution at the k-th iteration. We compute exactly the norm of residual, of the right-hand side, and the iterative preconditioned approximated solution. The L2-norm of the preconditioner operator $\mathbf{AM}$ is instead computed by the following sampling approximation. Let $\mathcal{W}$ be a set of normalized TT-vectors generated randomly from a normal distribution, then $\|\mathbf{AM}\|_2$ is approximated by the maximum of the norm of the image of the elements of $\mathcal{W}$ through $\mathbf{AM}$, i.e.,

$$\|\mathbf{AM}\|_2 \approx \max_{\mathbf{w}\in\mathcal{W}} \|\mathbf{AMw}\|.$$

Similarly, the L2-norm of $\mathbf{A}$ is also approximated by $\max\left\{\|\mathbf{Aw}\| \; s.t. \mathbf{w} \in \mathcal{W}\right\}$. Because we are interested in the magnitude of these norms, we keep this norm estimation process simple and only compute 10 random TT-vectors of $\mathcal{W}$.

In order to investigate the main numerical features of the GMRES implementation described in the previous section we consider two classical PDEs, i.e., the Poisson and convection-diffusion equations.

The Poisson problem writes

$$\begin{cases} -\Delta u = f \quad \text{in} \quad \Omega = [0,1]^3, \\ u = 0 \quad \text{in} \quad \partial\Omega, \end{cases} \tag{2.40}$$

where $f : \mathbb{R}^3 \to \mathbb{R}$ is such that the analytical solution of this Poisson problem is $u : [0,1]^3 \to \mathbb{R}$ defined as $u(x,y,z) = (1-x^2)(1-y^2)(1-z^2)$. Let set a grid of n points per mode over Ω, the discretization of the Laplacian over the Cartesian grid is the linear operator $-\boldsymbol{\Delta}_d$ defined in Equation (2.37) with $d = 3$. Let $\mathbf{b} \in \mathbb{R}^{n\times n\times n}$ be the discrete right-hand side in TT-format such that $\mathbf{b}(i_1, i_2, i_3) = f(x_{i_1}, y_{i_2}, z_{i_3})$.

The convection-diffusion problem, identical to the one considered in [39], writes

$$\begin{cases} -\Delta u + 2y(1-x^2)\dfrac{\partial u}{\partial x} - 2x(1-y^2)\dfrac{\partial u}{\partial y} = 0 \quad \text{in} \quad \Omega = [-1,1]^3\,, \\ u_{\{y=1\}} = 1 \qquad \text{and} \qquad u_{\partial\Omega\setminus\{y=1\}} = 0\,. \end{cases} \tag{2.41}$$

Setting a grid of n points per mode over $[-1,1]^3$, the Laplacian is discretized as in Equation (2.37) with $d = 3$. Let $\boldsymbol{\nabla}_x$ be discretization of the first derivative of u with respect to mode 1 defined as $\boldsymbol{\nabla}_x = \nabla_1 \otimes \mathbb{I}_n \otimes \mathbb{I}_n$, similarly $\boldsymbol{\nabla}_y$ is the discrete first derivative with respect to mode 2 written as $\boldsymbol{\nabla}_y = \mathbb{I}_n \otimes \nabla_1 \otimes \mathbb{I}_n$, where ∇_1 is the order-2 central finite difference matrix, i.e.,

$$\nabla_1 = \frac{1}{2h}\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & -1 & 0 & 1 \\ 0 & 0 & \dots & -1 & 0 \end{bmatrix}.$$

Let $v : [-1,1]^3 \to \mathbb{R}^2$ be a function such that $v(x,y,z) = \Big(2y(1-x^2), -2x(1-y^2)\Big)$, the two components of v are discretized over the Cartesian grid set on $[-1,1]^3$ defining two tensors $\mathbf{V}_1, \mathbf{V}_2 \in \mathbb{R}^{(n\times n)\times(n\times n)\times(n\times n)}$ such that $\mathbf{V}_1 = \mathrm{diag}(1-x^2) \otimes \mathrm{diag}(2y) \otimes \mathbb{I}_n$ and $\mathbf{V}_2 = \mathrm{diag}(-2x) \otimes \mathrm{diag}(1-y^2) \otimes \mathbb{I}_n$. Then the discrete diffusion term $\mathbf{D}$ writes

$$\begin{aligned} \mathbf{D} &= \mathbf{V}_1 \bullet \boldsymbol{\nabla}_x + \mathbf{V}_2 \bullet \boldsymbol{\nabla}_y \\ &= \mathrm{diag}(1-x^2)\nabla_1 \otimes \mathrm{diag}(2y) \otimes \mathbb{I}_n + \mathrm{diag}(-2x) \otimes \mathrm{diag}(1-y^2)\nabla_1 \otimes \mathbb{I}_n\,. \end{aligned} \tag{2.42}$$

The final operator passed to the TT-GMRES algorithm is $\mathbf{A} = -\boldsymbol{\Delta}_3 + \mathbf{D}$, the right-hand side is the TT-vector $\mathbf{b} \in \mathbb{R}^{n\times n\times n}$ and the initial guess is the zero TT-vector $\mathbf{x}_0$. To ensure a fast convergence, similarly to [39], we consider a right preconditioner $\mathbf{M}$ from Equation (2.38) for this test example.

2.3.3.1 Main features and robustness properties

In this section, we first illustrate in Section 2.3.3.1 the major differences between our GMRES implementation and the one proposed in [39] that mostly highlights the robustness of our variant. We motivate the need for effective preconditioners in Section 2.3.3.1 and illustrate the performance and the main features of preconditioned GMRES in Section 2.3.3.1. All the experiments were performed using `python 3.6.9` and with the tensor toolbox `ttpy 1.2.0` [109].

Comparison with previous tensor GMRES algorithm

In this section we describe the TT-GMRES introduced in [39], which we refer to as relaxed TT-GMRES, that attempts to use advanced features enabled by the inexact GMRES theory [16, 50, 128, 139]. In particular, these inexact GMRES theoretical results show that some perturbations can be introduced in the linear operator when enlarging the Krylov space so that the magnitude of these perturbations can grow essentially as the inverse of the true residual norm of the current iterate. In that context, the accuracy of computation of the linear operator can be relaxed, which motivated the use of this terminology in [16, 50]. The inexact GMRES theory assumes exact arithmetic so that Equation (2.1) holds. In practice, this equality becomes invalid as soon as some loss of orthogonality appears in the Arnoldi basis so that

$$\|\tilde{r}_k\| = \|\beta e_1 - \bar{H}_k y\| \neq \|r_k\| = \|b - Ax_k\|; \tag{2.43}$$

that is, the norms of the least squares residual and the true residual differ.

In a TT-computational context, these inexact Krylov results motivated the heuristic presented in [39], which consists in transferring the perturbation policy from the matrix to the output of the matrix-vector product. More precisely, the variable perturbation magnitude is implemented by varying the rounding threshold δ applied to the tensor resulting from the matrix-vector product along the iterations. Furthermore, the magnitude of the rounding δ is computed using the least squares residual norm rather than the true residual norm for practical computational reasons. A possible consequence of this choice is that δ is somehow artificially increased.

Although the rounding is performed exactly at the same step in the two algorithms, there are two differences between our TT-GMRES and the relaxed TT-GMRES [39]. The first difference is related to the rounding threshold policy that is variable (or relaxed to use the terminology of the pioneering paper on inexact GMRES [16]) and constant in our case. We simply define the value of δ essentially to the value of the target accuracy in terms of backward error (2.3) ((2.39) when a preconditioner is used). The second difference is related to the stopping criterion that is defined in terms of backward error (2.3) in our case ((2.39) when a preconditioner is used) while it is based on a scaled least squares residual defined by Equation (2.44) in [39]:

$$\tilde{\eta}_{\mathbf{b}}(\mathbf{x}_k) = \frac{\|\tilde{r}_k\|}{\|\mathbf{b}\|}. \tag{2.44}$$

Because in practice the true residual differs from the least squares residual, this latter is monotonically decreasing towards zero, such a stopping criterion can lead to an earlier stop.

We choose this stopping criterion based on backward error because it is the one for which, in the matrix framework, GMRES is backward stable in finite precision [113]. Through intensive numerical experiments [3], we observed that our TT-GMRES inherits the same backward stability property. Indeed if δ is the rounding accuracy and $\mathbf{x}_k$ the GMRES solution at iteration k, then $\eta_{\mathbf{A},\mathbf{b}}(\mathbf{x}_k)$ is $\mathcal{O}(\delta)$ as δ is the dominating part of the rounding error occurring during the numerical calculation. Consequently assuming $\delta \leq \varepsilon$, our GMRES variant is able to ensure a ε-backward stable solution. This property is well

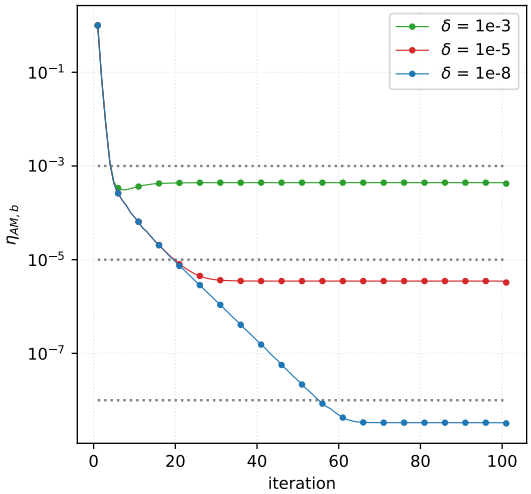
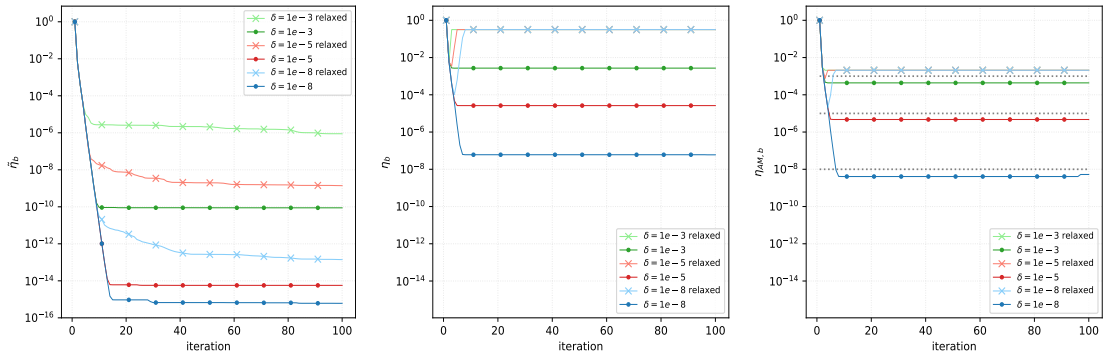


Figure 2.6 – Convergence history of TT-GMRES on a 3-d convection-diffusion problem, $n = 64$, for three different rounding accuracies δ.

illustrated in Figure 2.6 in the case of preconditioned GMRES. The 3d convection-diffusion problem with 63 discretization points is solved using 3 different rounding accuracies, i.e., $\delta \in \{10^{-3}, 10^{-5}, 10^{-8}\}$, and a maximum of 100 iterations. For each value of δ, the backward error $\eta_{\mathbf{AM},\mathbf{b}}(\mathbf{t}_k)$ decreases and stagnates around δ.

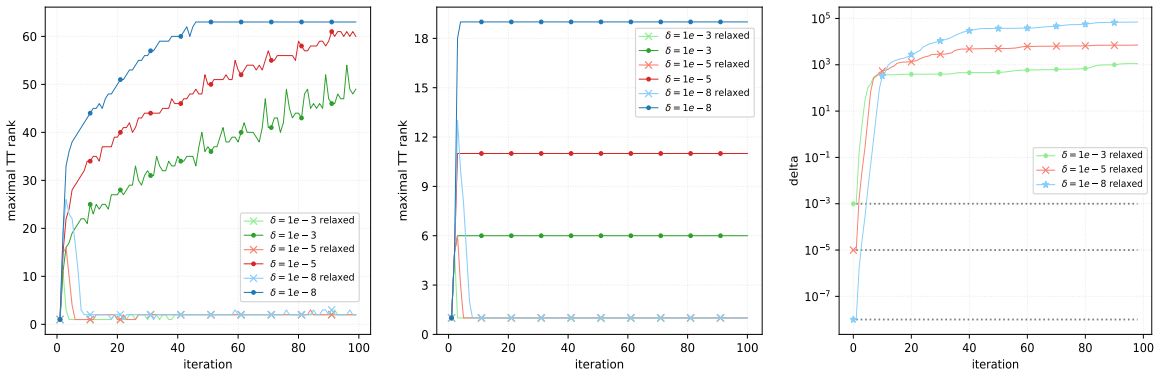
The second significant difference between the two GMRES variants is the choice of the rounding threshold along the iterations that is constant for us and varies as the inverse $\|\tilde{r}_k\|$ in [39]. This variation of the rounding is illustrated in Figure 2.7. We solve with the two different algorithms the same convection-diffusion problem with 63 discretization points in each spatial dimension. We select three different rounding accuracies $\delta \in \{10^{-3}, 10^{-5}, 10^{-8}\}$ and perform 100 iterations of full GMRES (i.e., no restart). In Figure 2.7F we see the extreme growth of the rounding threshold, when it is scaled by the norm of $\tilde{r}_k$, the least-squares residual norm that becomes smaller and smaller. When the rounding accuracy becomes significantly large, the TT-ranks in relaxed TT-GMRES are cut to 1, losing almost all the information carried in the tensor. Figure 2.7A shows the scaled residual used as stopping criterion in [39]. We observe that if δ is not relaxed along



(A) Convergence history with $\tilde{\eta}_{\mathbf{b}}$ with least squares residual

(B) Convergence history with $\eta_{\mathbf{b}}$ with true residual

(C) Convergence history $\eta_{\mathbf{AM},\mathbf{b}}$ with true residual



(D) Max TT-rank of the last Krylov vector

(E) Max TT-rank of the iterative solution

(F) History of the relaxed δ values

Figure 2.7 – TT-GMRES and relaxed TT-GMRES for the solution of 3-d convection-diffusion problem with $n = 63$.

the iterations, the value of $\tilde{\eta}_{\mathbf{b}}$ decreases extremely quickly, reaching 10^{-10} for $\delta = 10^{-3}$ and at least 10^{-14} for the other rounding accuracies. On the other hand, if the rounding accuracy is relaxed during the iterations, we see that in all the cases $\tilde{\eta}_{\mathbf{b}}$ reaches at least 10^{-6}. However, the comparison of Figure 2.7A and Figure 2.7B illustrates the numerical difference of the least squares residual norm and the true residual norms given by Equation (2.43). This comparison reveals that $\tilde{\eta}_{\mathbf{b}}(\mathbf{x}_k)$ with the relaxed δ converges, but $\eta_{\mathbf{b}}(\mathbf{x}_k)$, that is also a backward error as defined in (2.4), does not. It means that the solutions computed using the relaxed δ are meaningless in terms of backward error accuracy. Similar conclusions can be drawn from Figure 2.7C that presents the history of $\eta_{\mathbf{AM},\mathbf{b}}$ for the two algorithms. When the rounding accuracy is kept constant, we recover a backward

stable behaviour similar to the one proved for finite precision calculation in classical linear system solution in matrix format. Indeed $\eta_{\mathbf{AM,b}}$ always reaches and stagnates around the selected constant value of δ. On the contrary, when δ is relaxed at each iteration, the quantity $\eta_{\mathbf{AM,b}}$ stagnates quickly slightly above 10^{-3}, whatever the starting value of δ. From these two figures, we conclude that relaxing the rounding accuracy and using $\tilde{\eta}_{\mathbf{b}}$ as a stopping criterion, together or independently, do not provide any insight into the quality of the computed solution.

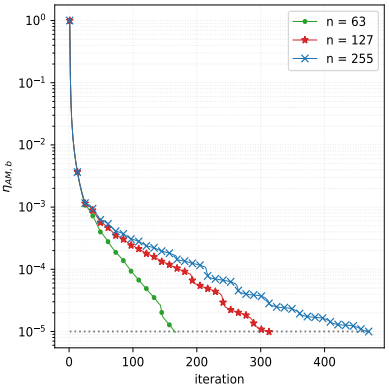
Obviously, the choice of relaxing the rounding accuracy has a powerful effect on the rank of the last Krylov basis vector and the solution, as illustrated by Figure 2.7D and 2.7E. Indeed in the case of the last Krylov basis vector, its TT-rank oscillates around 1 for all the iterations, after the 15-th one approximately. Similarly, the solution TT-rank stays equal to 1, after increasing at the very first steps. Unfortunately, the computed solutions are numerically meaningless. In the following, we consider calculation with convergence threshold and rounding accuracy equal to 10^{-5}, that is, $\delta = \varepsilon = 10^{-5}$, with a maximum of 500 iterations and restart $m = 25$.

Poisson problem

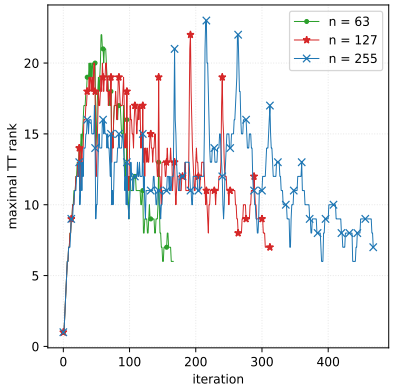
We consider the restarted TT-GMRES for the solution of the 3-d Poisson problem with $n \in \{63, 127, 255\}$. Figure 2.8A shows that the algorithm is able to converge to the prescribed tolerance $\varepsilon = 10^{-5}$ with the number of iterations that increases with the number of discretization points. This high number of steps to solve a quite simple PDE motivates the need for a preconditioner. Indeed in general the larger the number of TT-GMRES iterations, the larger the TT-rank growth for the Krylov basis vectors; consequently, the higher the computational cost per iteration. For that example, it can be seen in Figure 2.8B that the rank of the current iterate grows significantly during the first iterations (first 100 iterations for $n = 63$ and the first 200 steps for $n \in \{127, 255\}$) before decreasing in a non monotonic way. We infer that this particular behavior is related to the separable nature of the analytical solution, which is

$$u(x, y, z) = \left[\text{diag}(1 - x^2)\right] \otimes \left[\text{diag}(1 - y^2)\right] \otimes \left[\text{diag}(1 - z^2)\right]$$

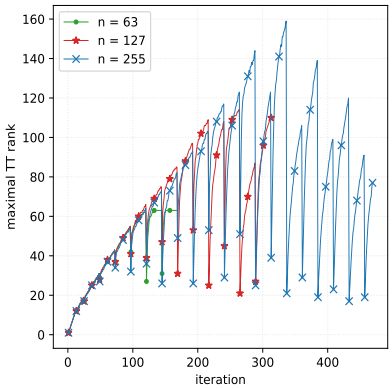
with rank 1 and as consequence its TT-rank is also bounded by 1. After some iterations TT-GMRES seems to capture the main structure of the solution, being able to almost halve the TT-ranks, as it is visible in Figure2.8B. Another quantity monitored during the iterations is the growth of the last Krylov vector TT-ranks. In Figure 2.8C the maximum TT-rank of the last Krylov vector presents a steep increase during the first phase, followed by a slightly decreasing phase. The behavior of the maximum TT-rank establishes the trend in the compression ratio of the last vector and the entire basis. Indeed the curves of Figures 2.8D and 2.8E are the same of Figure 2.8C scaled by a constant, equal to n^3 for the first and kn^3 for the second where k is equal to the current iteration in the restart. Lastly, in Figure 2.8C mainly during the second phase, there are many consecutive drops in the maximum TT-ranks which appear with a specific frequency. They are due to the restart after every other 25-th iteration. In fact, at restart the new Krylov vector is equal to the normalized rounded residual, whose basic starting TT-ranks is the one of $\mathbf{x}$, equal



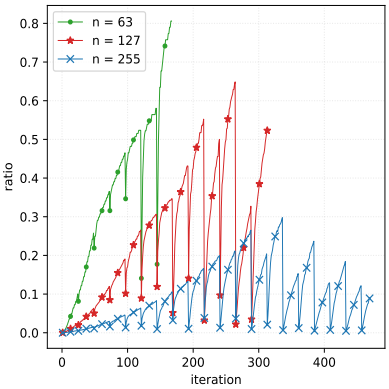
(A) Convergence history



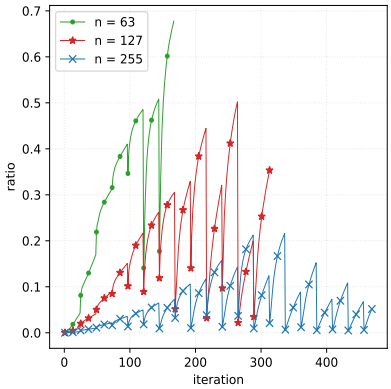
(B) Maximal TT-rank of the iterative solution



(C) Maximal TT-rank of the last Krylov vector



(D) Compression ratio for the last Krylov vector



(E) Compression ratio for the entire Krylov basis

Figure 2.8 – 3-d Poisson problem using $\delta = \varepsilon = 10^{-5}$.

to 21 at maximum. Lastly notice that in the worst case storing the last Krylov vector and the entire Krylov basis request approximately 80% for $n = 63$ of the memory that would be used for storing entirely them. Furthermore, this ratio decreases when the number of points per mode increases (i.e., $n \in \{63, 127, 255\}$), which is an appealing feature of the TT-format that allows the solution of larger problems for a given memory budget compared to the situation where the full tensors would have to be stored.

Preconditioner parameter study

In this section, the preconditioner firstly introduced in (2.38) is further investigated. In particular, we focus on the effect on the convergence of the number of addends and on the compression accuracy chosen to compute it. As in Equation (2.38), let $\mathbf{M} \in \mathbb{R}^{(n\times n)\times\cdots\times(n\times n)}$ be the d-order TT-matrix that approximates the inverse of the

discrete Laplacian $\mathbf{\Delta}_d$, cf. [61, 62], defined as

$$\mathbf{M} = \sum_{k=-q}^{q} c_k \exp(-t_k \Delta_1) \otimes \cdots \otimes \exp(-t_k \Delta_1)$$

where $c_k = \eta t_k$, $t_k = \exp(k\eta)$ and $\eta = \pi/q$. As already mentioned, since $\mathbf{M}$ is a sum of tensors, its TT-rank is greater or equal than $2q + 1$. The magnitude of TT-ranks of $\mathbf{M}$ conditions the TT-ranks of the Krylov basis vectors and the final solution. Therefore it is convenient to keep $\mathbf{M}$ TT-rank significantly small, either by reducing the number of addends, i.e., choosing a low value for q, or by compressing $\mathbf{M}$ with an accuracy τ. With the help of a Poisson problem, we illustrate the trade-off between the number of addends and compression accuracy which leads to the optimal convergence. We consider the Poisson problem, written as

$$\begin{cases} -\Delta u = 1 & \text{in} \quad \Omega = [0,1]^3 \\ u = 0 & \text{in} \quad \partial\Omega \end{cases}$$

so that the effect of the preconditioner is as evident as possible. Let $-\mathbf{\Delta}_3$ be the discretization of the Laplacian operator over a grid of $n = 63$ points per mode. Similarly let $\mathbf{b} \in \mathbb{R}^{n \times n \times n}$ be a tensor with all the entries equal to 1. Then, setting $\mathbf{A} = -\mathbf{\Delta}_3$, TT-GMRES solves the tensor linear system $\mathbf{Ax} = \mathbf{b}$, preconditioning it on the right as

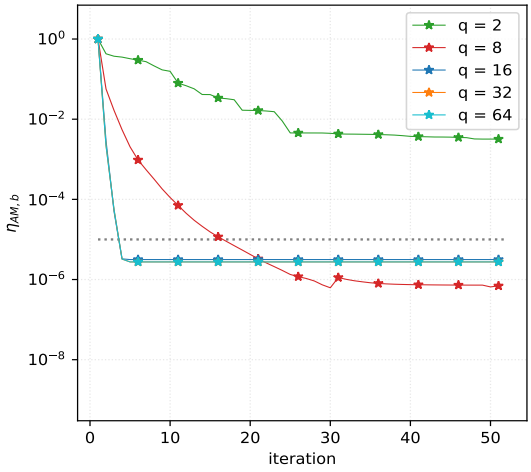
$$\mathbf{A}\mathbf{M}_{q,\tau}\mathbf{t} = \mathbf{b}$$

for $q \in \{2, 8, 16, 32, 64\}$ and $\tau \in \{10^{-2}, 10^{-8}\}$. The parameters of TT-GMRES are tolerance $\varepsilon = 10^{-16}$, rounding accuracy $\delta = 10^{-5}$, dimension of the Krylov space $m = 25$ and a maximum of 2 restart. We set TT-GMRES tolerance equal to the machine precision so that the algorithm performs all the 50 iterations. In Table 2.3, we report the maximal TT-rank of $\mathbf{M}_{q,\tau}$ and the $\|\mathbf{A}\mathbf{M}_{q,\tau}\|_2$ rounded at the third digits for all the combinations of q and τ. Remark that fixed a value for q the L2 norm of the preconditioned linear system is the same up to the third digits for both the values of τ. This seems to suggest that the number of addends plays a key role in determining the quality of the preconditioner, while the rounding accuracy τ affects more significantly the TT-rank, removing kind of unnecessary information. Indeed for $\tau = 10^{-2}$ and $q \geq 8$, the maximal value of the TT-rank is always 5, but depending on an increasing number of addends, the L2 norm gets closer to 1. Similarly for $\tau = 10^{-8}$ and $q \geq 32$, the maximal TT-rank is 15 and the rounded L2 norm is equal to 1.

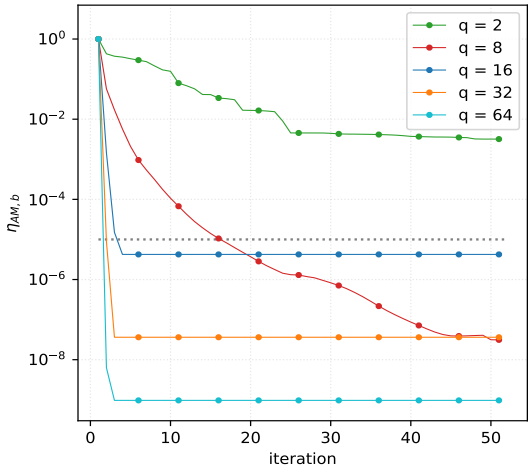
	$\tau = \mathbf{10^{-2}}$					$\tau = \mathbf{10^{-8}}$				
q	**2**	**8**	**16**	**32**	**64**	**2**	**8**	**16**	**32**	**64**
Max TT-rank of $\mathbf{M}_{q,\tau}$	2	5	5	5	5	2	7	13	15	15
L2 norm of $\mathbf{A}\mathbf{M}_{q,\tau}$	0.012	0.276	0.949	1.00	1.00	0.012	0.276	0.949	1.00	1.00

Table 2.3 – Preconditioner properties for grid step $n = 63$.

Looking at the convergence history in Figures 2.9A and 2.9B, a value of $q \geq 16$ is already sufficient to reach in a very low number of iterations the bound 10^{-5}, due to the TT-GMRES rounding value δ. Figure 2.9B shows clearly that by keeping more information in the preconditioner, TT-GMRES may reach very low levels. However in Figure 2.9D we observe the side effect of more information. The TT-rank of the last Krylov vector increases significantly for a very accurate preconditioner. More precisely, comparing Figures 2.9C and 2.9D, the TT-rank of the last Krylov vector doubles if the preconditioner is more accurately rounded. Notice also that in Figure 2.9C, the TT-rank for $q \in \{16, 32, 64\}$ is almost the same. For the solution viewpoint, the rounding accuracy chosen for the preconditioner has not a big impact on its TT-rank. Indeed, as plotted in Figure 2.9E and 2.9F, for both the values of τ and for all $q \geq 8$, the TT-rank of the solution is equal to 5, while only for $q = 2$ it increases, meaning that only 5 addends are not sufficient to speed up the discrete Laplacian convergence.



(A) Convergence history for $\tau = 10^{-2}$ and rounding $\delta = 10^{-5}$

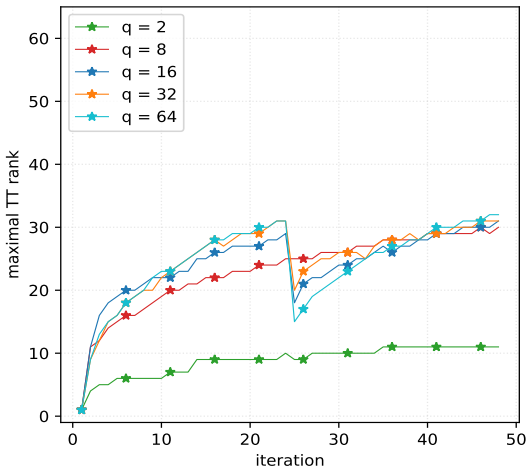


(B) Convergence history for $\tau = 10^{-8}$ and rounding $\delta = 10^{-5}$

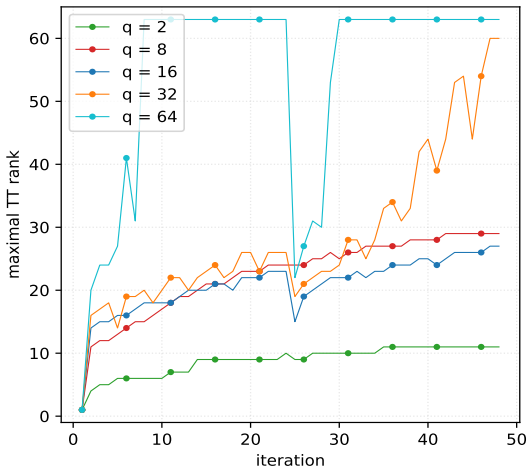
Figure 2.9 – 3-d Poisson problem, comparing preconditioners.

Convection-diffusion

We test three different grid sizes, i.e., $n \in \{63, 127, 255\}$, with preconditioner $\mathbf{M}$ from Equation (2.38) with $q \in \{16, 32\}$. Indeed without it, even with the smallest size, TT-GMRES does not converge to the prescribed tolerance $\varepsilon = 10^{-5}$ in a reasonable number of iterations. However using the preconditioner defined in Equation (2.38), an approximated solution is found in 5 or less iterations, as displayed in Figure 2.10A. The preconditioner in this case has an extremely strong effect, from which the TT-rank growth and the memory consumption benefit. In Figure 2.10B the maximum TT-rank exceeds in the worst case the value of 35, but to fully interpret this information the compression ratio must be taken into consideration. Figure 2.10C shows that in the worst case to store the last Krylov

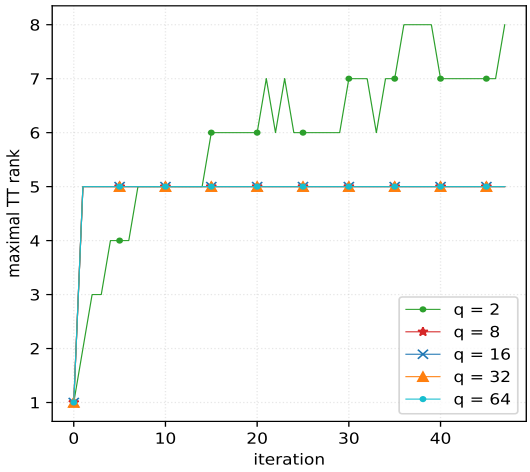


(C) Maximal TT-rank of the last Krylov vector for $\tau = 10^{-2}$

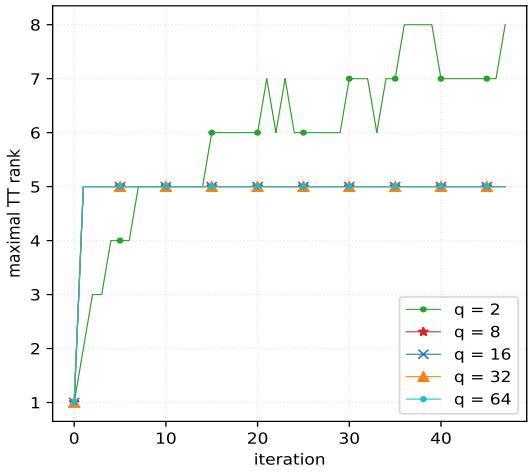


(D) Maximal TT-rank of the last Krylov vector for $\tau = 10^{-8}$

Figure 2.9 – 3-d Poisson problem, comparing preconditioners.



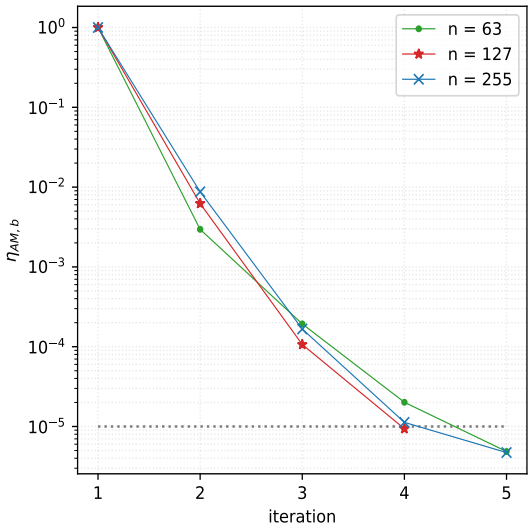
(E) Maximal TT-rank of the iterative solution for $\tau = 10^{-2}$



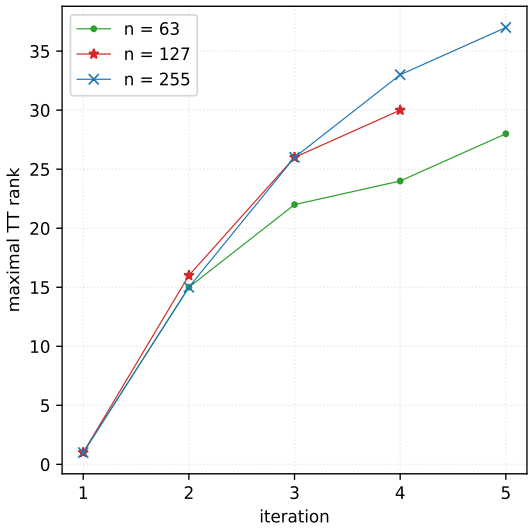
(F) Maximal TT-rank of the iterative solution for $\tau = 10^{-8}$

Figure 2.9 – 3-d Poisson problem, comparing preconditioners.

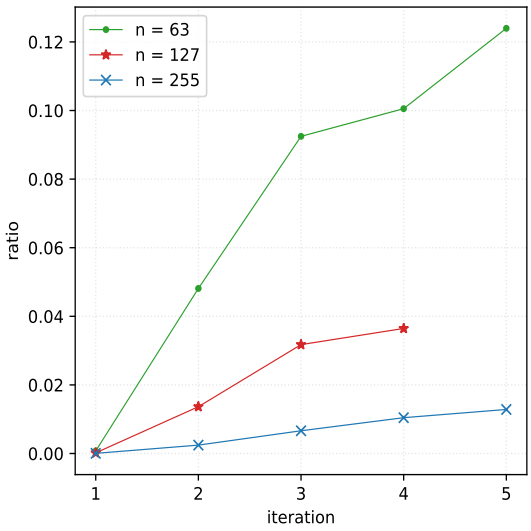
vector in TT-format we use approximately 12% of the memory we would need to store the full tensor. Similarly, in Figure 2.10D, we see that storing in TT-format the entire Krylov basis request in the worst case only 7% of the memory that would be used to store the full tensors basis. Although not reported in this document, a more stringent accuracy would require a smaller rounding threshold and consequently a larger memory to store the TT-vectors.



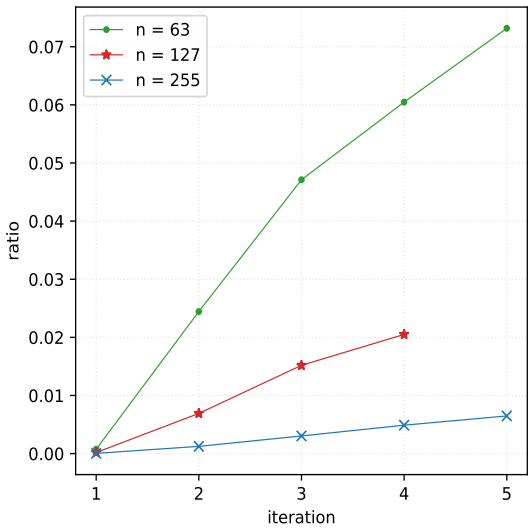
(A) Convergence history



(B) Maximal TT-rank of the last Krylov vector



(C) Compression ratio for the last Kyrolv vector



(D) Compression ratio for the entire Krylov basis

Figure 2.10 – 3-d Convection-diffusion using $\delta = \varepsilon = 10^{-5}$.

2.3.3.2 Solution of parameter-dependent linear operators

This section focuses on 4-d PDEs, namely parametric convection-diffusion and stationary heat equations. The domain of both problems is obtained as a Cartesian product of a 3-d space domain and a further parameter space. The common idea for these PDEs is solving for all discrete parameter values simultaneously, getting an "all-in-one" solu-

tion. The structure of the operators enables us to check numerically the quality of the theoretical bounds stated in Section2.3.2.

Parametric convection-diffusion

The parametric convection-diffusion problem is a variation of Problem (2.41), defined as

$$\begin{cases} -\alpha\Delta u + 2y(1-x^2)\dfrac{\partial u}{\partial x} - 2x(1-y^2)\dfrac{\partial u}{\partial y} = 0 \quad \text{in} \quad \Omega = [-1,1]^3\,, \\ \\ u_{\{y=1\}} = 1 \qquad \text{and} \qquad u_{\partial\Omega\setminus\{y=1\}} = 0\,. \end{cases} \tag{2.45}$$

As in Section 2.3.3.1 let define a grid of n points along each direction of Ω, then the final discrete operator of this PDE is $\mathbf{A}_\alpha = \alpha\boldsymbol{\Delta}_3 + \mathbf{D}$ with $\alpha \in [1, 10]$ and $\mathbf{D}$ defined in Equation (2.42). Similarly, the right-hand side $\mathbf{c}_\alpha \in \mathbb{R}^{n\times n\times n}$ depends on the parameter $\alpha \in [1, 10]$ because of the boundary conditions. To solve for multiple discrete values of α, getting an "all-in-one" problem and solution, we tensorize $\boldsymbol{\Delta}_3$ and $\mathbf{D}$ by a diagonal matrices, adding a fourth dimension. The tensor operator for the simultaneous solution is $\mathbf{A} \in \mathbb{R}^{(p\times p)\times(n\times n)\times(n\times n)\times(n\times n)}$ defined as

$$\mathbf{A} = A \otimes \boldsymbol{\Delta}_d + \mathbb{I}_p \otimes \mathbf{D}\,,$$

where $A = \mathrm{diag}(\alpha_1, \dots \alpha_p)$ with $\alpha_i \in [1, 10]$ logarithmically distributed for $i \in \{1, \dots, p\}$. The right-hand side of the "all-in-one" problem is $\mathbf{b} \in \mathbb{R}^{p\times n\times n\times n}$ such that

$$\mathbf{b}^{[\ell]} = \frac{1}{\|\mathbf{c}_{\alpha_\ell}\|}\mathbf{c}_{\alpha_\ell} \qquad \text{for} \qquad \ell \in \{1, \dots, p\}$$

using the slice notation introduced in Section 2.3.2. By construction $\|\mathbf{b}\| = \sqrt{p}$, i.e., the discrete "all-in-one" problem fits into the hypothesis of Proposition 2.3.5 and 2.3.7. Remark that the "all-in-one" linear operator is directly constructed as a TT-matrix from the TT-matrix of the single linear system, while the "all-in-one" right-hand side is constructed as a full tensor and then converted into a TT-vector.

TT-GMRES is used for solving the "all-in-one" linear system for $n \in \{63, 127, 255\}$ and $p = 20$, with the order-d preconditioner $\overline{\mathbf{M}}$ defined in Equation (2.38) with $q \in \{16, 32\}$ tensorized with the identity

$$\mathbf{M} = \mathbb{I}_p \otimes \overline{\mathbf{M}}. \tag{2.46}$$

Figure 2.11A shows that the algorithm converges in less than 20 iterations for the first two values of n and in less than 25 for $n = 255$; that is, no restart is needed. For the computational side, Figure 2.11B displays the maximal TT-rank of the last Krylov vector, which in the worst case is lower than 100. This result translates in terms of memory by a need of slightly more than 4% of the memory that would be required to store the full Krylov vector in the worst case, as highlighted by Figure 2.11C. Looking at the cost of storing the entire Krylov basis in Figure 2.11D, we see that TT-format requires around 2% of the memory necessary to store the entire Krylov basis in full tensor format. We now investigate the tightness of the bound given in Proposition 2.3.5 and

2.3.7. Figure 2.13 shows the quality of the bound for $\eta_{\mathbf{b}_\ell}$ for $\ell \in \{1, \dots, p\}$. For all the values of n, the $\eta_{\mathbf{b}_1}$ curve dominates the other during the first half of the iterations. In the optimal case, the difference between $\eta_{\mathbf{b}_\ell}$ and $\eta_{\mathbf{b}}$ is lower than one order of magnitude. To plot the $\eta_{\mathbf{AM},\mathbf{b}}$ bound from Proposition 2.3.5, we define a vector $\upsilon_\ell \in \mathbb{R}^w$ whose k-th component corresponds to the value of the coefficient ρ_ℓ from Equation (2.26) evaluated for the solution at the k-th iteration, i.e.,

$$\upsilon_\ell(k) = \rho_\ell(\mathbf{t}_k) \qquad \text{for every} \qquad k \in \{1, \dots, w\}$$

with w equal to the iteration number to reach the convergence. Let ℓ_m and ℓ_M the parameter index for which the norm of υ_ℓ is minimal and maximal respectively, i.e.,

$$\ell_m = \underset{\ell \in \{1,\dots,p\}}{\operatorname{argmin}} \|\upsilon_\ell\| \quad \text{and} \quad \ell_M = \underset{\ell \in \{1,\dots,p\}}{\operatorname{argmax}} \|\upsilon_\ell\| \tag{2.47}$$

which in our specific case are equal to 1 and 14 respectively. In Figure 2.13 we display in $\eta_{\mathbf{AM},\mathbf{b}}(\mathbf{t}_k)$ scaled by ρ_ℓ (see Equation (2.26) from Proposition 2.3.5) and by ρ^* (see Equation (2.33) from Corollary 2.3.8) versus $\eta_{\mathbf{A}_\ell\overline{\mathbf{M}},\mathbf{b}_\ell}(\mathbf{t}_k^{[\ell]})$ for $\ell \in \{5, 20\}$ and for all the values of n.

The three scaled curves overlap from the third iterations for all the grid sizes, meaning that the approximation of the scaling coefficient given by ρ^* is extremely valid in this example. We see that the orange curve corresponding to $\eta_{\mathbf{A}_5\overline{\mathbf{M}},\mathbf{b}_5}$ and the blue one for $\eta_{\mathbf{A}_{20}\overline{\mathbf{M}},\mathbf{b}_{20}}$ intersect frequently, with a difference of one order at most. Moreover the difference between $\eta_{\mathbf{A}_5\overline{\mathbf{M}},\mathbf{b}_5}$ and $\eta_{\mathbf{AM},\mathbf{b}}$ scaled by ρ_5 is lower than one order of magnitude in the optimal case, while in the worst case it is not larger than two orders. Therefore we conclude that for this PDE the bound of the "all-in-one" for the individual solution is quite tight. Notice that to estimate ρ^* no extra computation is required, while the norm of $\mathbf{A}_\ell\overline{\mathbf{M}}\mathbf{t}_k^{[\ell]}$ has to be computed to get the value of $\rho_\ell(\mathbf{t}_k)$.

Heat equation with parametrized diffusion coefficient

We consider the heat equation with parametrized diffusion coefficient studied in [86] and defined as

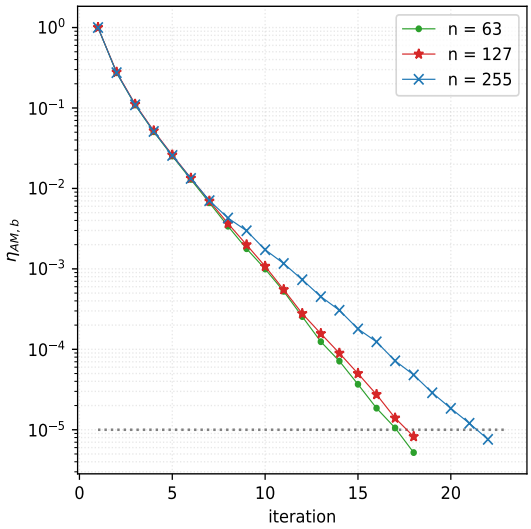
$$\begin{cases} -\nabla \cdot (\sigma_\theta(x,y,z)\nabla u(x,y,z)) &= 1 \quad \text{in} \quad \Omega = [-1,1]^3, \\ u &= 0 \quad \text{in} \quad \partial\Omega. \end{cases} \tag{2.48}$$

where the coefficient σ_θ is a piece-wise constant function such that

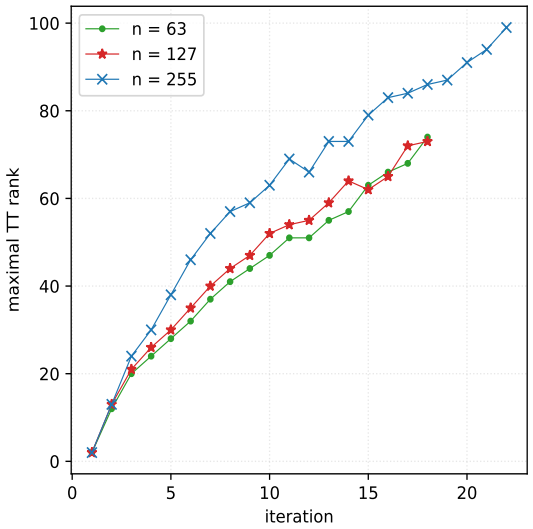
$$\sigma_\theta(x,y,z) = \begin{cases} 1+\theta & \text{in} \quad [-0.5, 0.5]^3, \\ 1 & \text{elsewhere,} \end{cases}$$

with $\theta \in [0, 10]$. The function σ_θ, rewritten as $\sigma_\theta(x,y,z) = 1 + \theta\mathbb{1}_\Xi(x,y,z)$ where $\mathbb{1}_\Xi$ is the indicator function of Ξ, provides a linear dependency on θ for the PDE. If Ξ_x is the projection of set Ξ over the x-axis and similarly for Ξ_y and Ξ_z, then

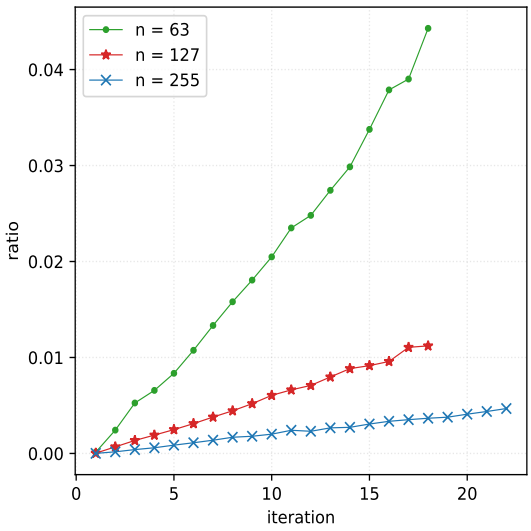
$$\sigma_\theta(x,y,z) = 1 + \theta\mathbb{1}_{\Xi_x}(x)\mathbb{1}_{\Xi_y}(y)\mathbb{1}_{\Xi_z}(z).$$



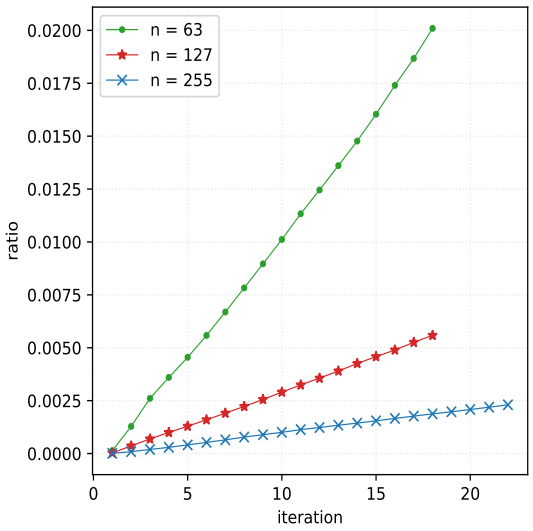
(A) Convergence history



(B) Maximal TT-rank of the last Krylov vector



(C) Compression ratio for the last Kyrolv vector

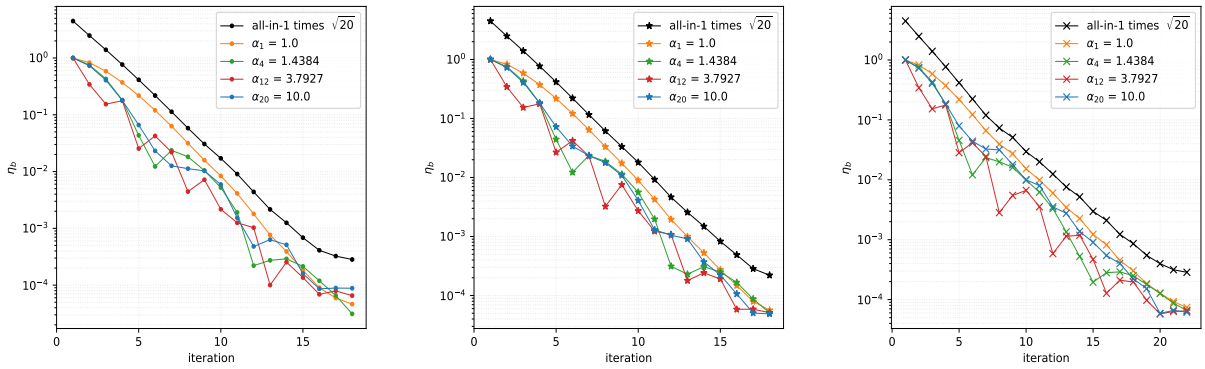


(D) Compression ratio for the entire Krylov basis

Figure 2.11 – 4-d Parametric convection-diffusion using $\delta = \varepsilon = 10^{-5}$.

The problem stated in Equation (2.48) writes equivalently

$$\begin{cases} -\Delta u(x,y,z) - \theta \nabla \cdot \left(\mathbb{1}_{\Xi_x}(x) \mathbb{1}_{\Xi_y}(y) \mathbb{1}_{\Xi_z}(z) \nabla u(x,y,z) \right) & = 1 \quad \text{in} \quad \Omega = [-1,1]^3, \\ u & = 0 \quad \text{in} \quad \partial\Omega. \end{cases} \tag{2.49}$$

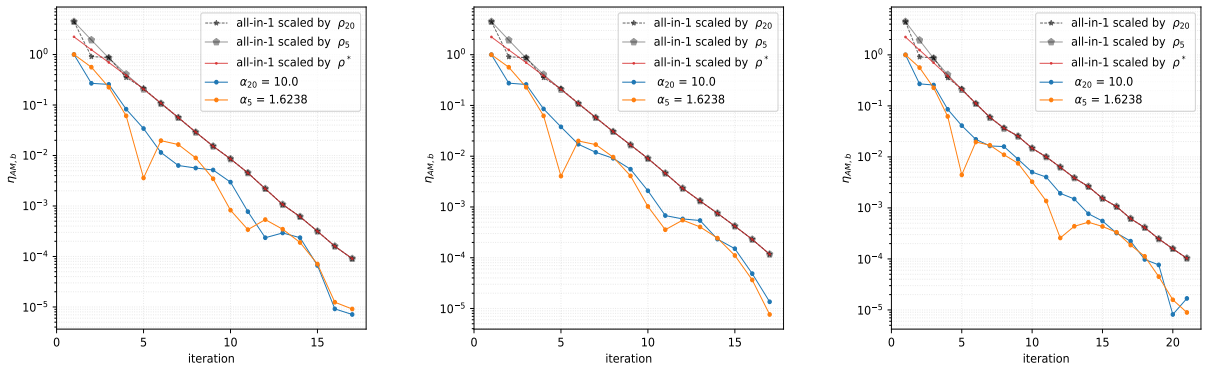


(A) Convergence history in $\eta_{\mathbf{b}}$ for $n = 63$

(B) Convergence history in $\eta_{\mathbf{b}}$ for $n = 127$

(C) Convergence history in $\eta_{\mathbf{b}}$ for $n = 255$

Figure 2.12 – 4-d Parametric convection-diffusion $\eta_{\mathbf{b}}$ bound using $\delta = \varepsilon = 10^{-5}$.



(A) Convergence history in $\eta_{\mathbf{AM},\mathbf{b}}$ for $n = 63$

(B) Convergence history in $\eta_{\mathbf{AM},\mathbf{b}}$ for $n = 127$

(C) Convergence history in $\eta_{\mathbf{AM},\mathbf{b}}$ for $n = 255$

Figure 2.13 – 4-d Parametric convection-diffusion $\eta_{\mathbf{AM},\mathbf{b}}$ bound using $\delta = \varepsilon = 10^{-5}$.

After setting a grid on n points along each direction on Ω, the first term $\mathbf{B}_0$ of the operator in (2.49) is discretized by the 3-d Laplacian $\mathbf{\Delta}_3$. For the second term $\mathbf{B}_1$, notice that the indicator function $\mathbb{1}_\Xi$ is trivially not differentiable on Ξ boundaries. So it is approximated on the grid points, paying attention to not set them on $\partial\Xi$. The final expression of $\mathbf{B}_1$ is

$$\mathbf{B}_1 = D_x\Delta_1 \otimes D_y \otimes D_z + D_x \otimes D_y\Delta_1 \otimes D_z + D_x \otimes D_y \otimes D_z\Delta_1$$

where Δ_1 is the 1-d discrete Laplacian, $D_x = \mathrm{diag}(\mathbb{1}_{\Xi_{x_i}}) \in \mathbb{R}^{n\times n}$ and similarly for D_y and D_x. Remark that $\mathbf{B}_1$ is a Laplacian-like operator, which is expressed in TT-format according to Equation (2.35) and (2.36). The final discrete TT-operator of Problem (2.48) is

$$\mathbf{A}_\theta = \mathbf{B}_0 + \theta\mathbf{B}_1.$$

The right-hand side is $\mathbf{c} \in \mathbb{R}^{n\times n\times n}$ such that $\mathbf{c}(i_1, i_2, i_3) = 1$ for $i_k \in \{1, \ldots, n\}$ for $k \in \{1, 2, 3\}$. To study the quality of the bounds expressed in Proposition 2.3.4 and 2.3.5,

the tensor $\mathbf{c}$ is normalized, i.e., it is scaled by $1/n^3$. Since we want to solve for p values of θ in $[0, 10]$ simultaneously, i.e., we want to solve p-times the discrete Problem (2.48) for different values of θ, we tensorize $\mathbf{B}_0$ and $\mathbf{B}_1$ by a diagonal matrices, adding a fourth dimension. The tensor discrete operator $\mathbf{A} \in \mathbb{R}^{(p\times p)\times(n\times n)\times(n\times n)\times(n\times n)}$ of the "all-in-one" problem writes

$$\mathbf{A} = \mathbb{I}_p \otimes \mathbf{B}_0 + \Theta \otimes \mathbf{B}_1$$

where $\Theta = \text{diag}(\theta_1, \ldots, \theta_p)$ for $\theta_i \in [0, 10]$ uniformly distributed for $i \in \{1, \ldots, p\}$. The right-hand side of the "all-in-one" problem is

$$\mathbf{b} = \mathbb{I}_p \otimes \mathbf{c}.$$

Remark that since $\|\mathbf{c}\| = 1$ by construction, then $\|\mathbf{b}\| = \sqrt{p}$. We perform experiments with full TT-GMRES (i.e., no restart) for $n \in \{63, 127\}$ and $p = 20$, with the preconditioner defined in Equation (2.46) with $q \in \{16, 32\}$. Figure 2.14A shows that TT-GMRES converges to the prescribed tolerance in approximately 20 iterations. From the point of view of the memory consumption, in Figure 2.14B we see that for $n = 63$ the maximum TT-rank is lower than 200, while for $n = 127$ it is lower than 250. In terms of memory saving, Figure 2.14C shows that in the worst case we are using only 10% and less than 5% of the memory necessary to store one full tensor of the Krylov basis and the entire full basis respectively.

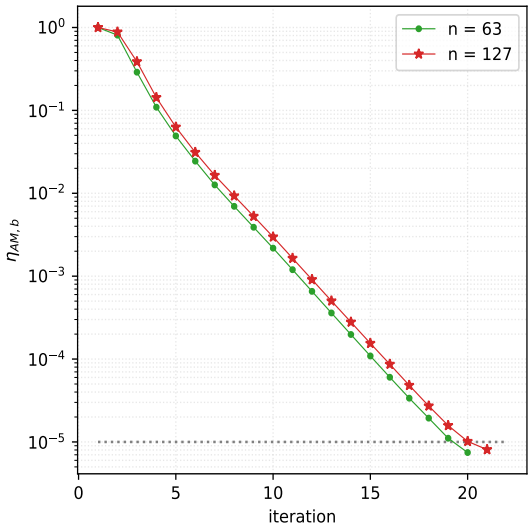
In Figure 2.15 we have the relation of $\eta_{\mathbf{b}}$ and $\eta_{\mathbf{b}_\ell}$ for $\ell \in \{1, \ldots, p\}$. All the curves present the same shape, with the one associated with $\theta_1 = 0$ being the most peculiar one. We see that in the optimal case the distance between the "all-in-one" curve and the individual ones is lower than one order of magnitude, while in the worst case, realized by $\theta_1 = 0$, the difference is approximately of two orders. A similar argument holds for $\eta_{\mathbf{AM},\mathbf{b}}$ bound. As in Section 2.3.3.2, we compute ℓ_m and ℓ_M, as defined in Equation (2.47), which are equal to $\ell_m = 20$ and $\ell_M = 1$ respectively. In Figure 2.16 we see that the two curves $\eta_{\mathbf{A}_\ell\overline{\mathbf{M}},\mathbf{b}_\ell}$ have a starting and ending overlapping part, while in the internal part they differ by less than one order of magnitude. The three scaled curves for $\eta_{\mathbf{AM},\mathbf{b}}$ overlap from the third iteration. As in the previously studied case, ρ^* from Corollary 2.3.8 provides a good approximation of the scaling coefficient. In the optimal case, the distance is of one order of magnitude approximately, while in the worst one a little more than one order.

2.3.3.3 Solution of parameter dependent right-hand sides

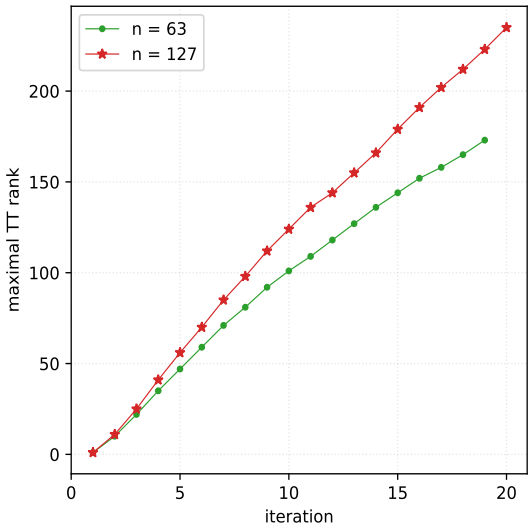
This section aims to investigate the numerical properties of some examples in the context of the multiple right-hand side solution, following the tensorized approach described in Section 2.3.2.

Poisson problem

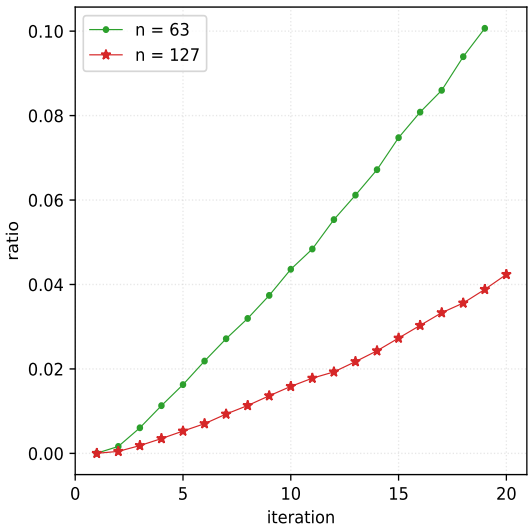
In this section, we solve simultaneously multiple Poisson problems stated in Equation (2.40) with modified right-hand sides. Let $-\mathbf{\Delta}_3$ be the discretization of the Laplacian over a Cartesian grid of n points per mode for the domain $\Omega = [0, 1]^3$. Let $\mathbf{b} \in \mathbb{R}^{n\times n\times n}$



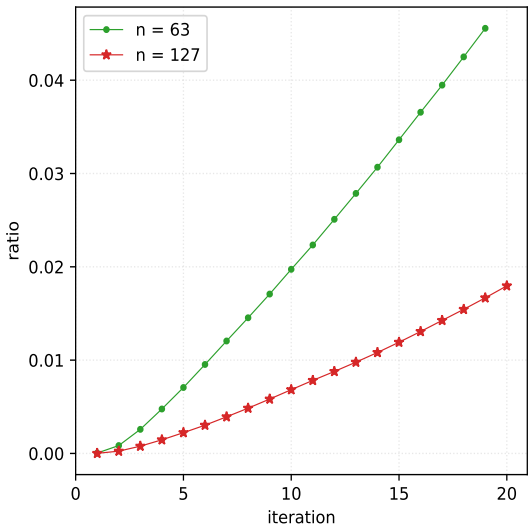
(A) Convergence history



(B) Maximal TT-rank of the last Krylov vector



(C) Compression ration for the last Kyrolv vector



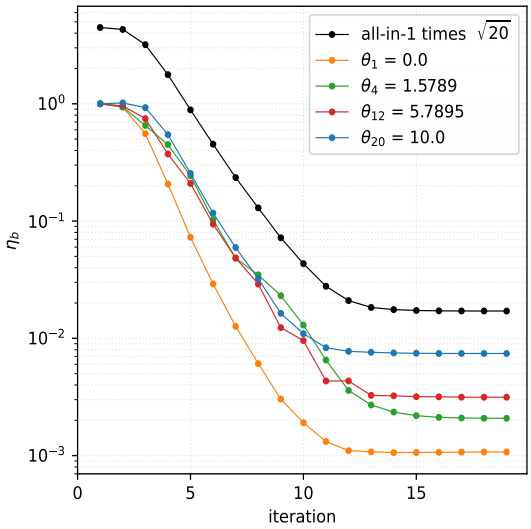
(D) Compression ratio for the entire Krylov basis

Figure 2.14 – 4-d Heterogeneous convection-diffusion using $\delta = \varepsilon = 10^{-5}$.

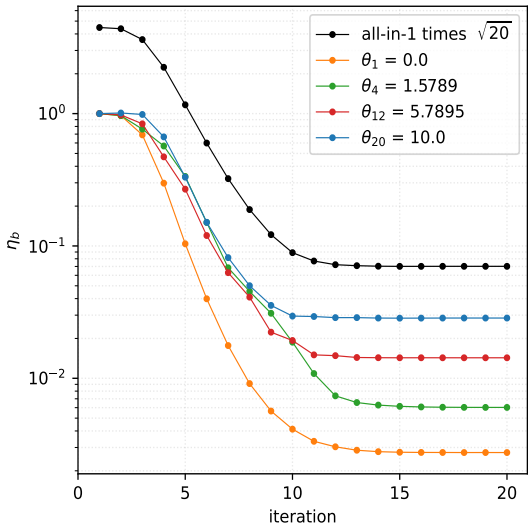
be the right-hand side discretization defined in Section 2.3.3.1. We define the individual linear system as

$$-\mathbf{\Delta}_3 \mathbf{y}_\ell = \mathbf{b} + \mathbf{e}^{[\ell]}$$

where $\mathbf{e}^{[\ell]} \in \mathbb{R}^{n \times n \times n}$ is the ℓ-th slice with respect to the first mode of $\mathbf{e} \in \mathbb{R}^{p \times n \times n \times n}$ a realization of the normal distribution $\mathcal{N}(0, 1)$. Since the aim is solving simultaneously the p problems, as in Section 2.3.2, we define the "all-in-one" tensor linear operator

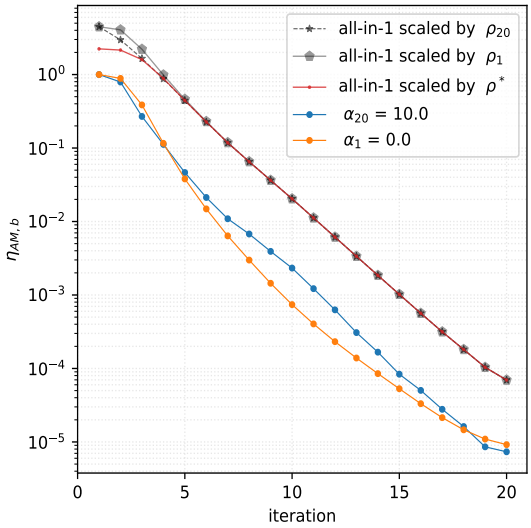


(A) Convergence history in $\eta_{\mathbf{b}}$ for $n = 63$

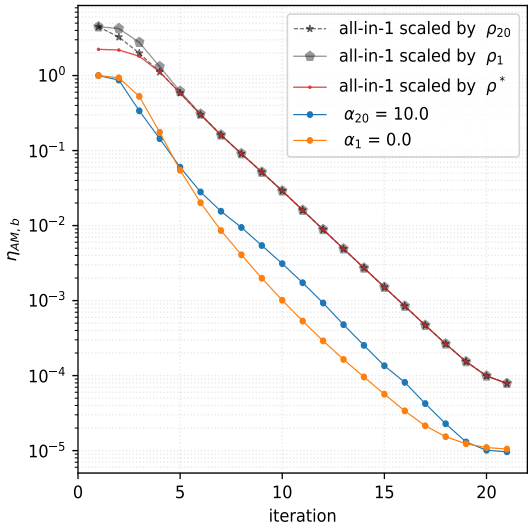


(B) Convergence history in $\eta_{\mathbf{b}}$ for $n = 127$

Figure 2.15 – 4-d Heterogeneous convection-diffusion $\eta_{\mathbf{b}}$ bound using $\delta = \varepsilon = 10^{-5}$.



(A) Convergence history in $\eta_{\mathbf{AM},\mathbf{b}}$ for $n = 63$



(B) Convergence history in $\eta_{\mathbf{AM},\mathbf{b}}$ for $n = 127$

Figure 2.16 – 4-d Heterogeneous convection-diffusion $\eta_{\mathbf{AM},\mathbf{b}}$ bound using $\delta = \varepsilon = 10^{-5}$.

$\mathbf{A} \in \mathbb{R}^{(p\times p)\times(n\times n)\times(n\times n)\times(n\times n)}$

$$\mathbf{A} = \mathbb{I}_p \otimes (-\boldsymbol{\Delta}_3)$$

while the "all-in-one" right-hand side is $\mathbf{c} \in \mathbb{R}^{p\times n\times n\times n}$ such that

$$\mathbf{c} = \mathbb{I}_p \otimes \mathbf{b} + \mathbf{e}.$$

We consider the solution of the problem with $n \in \{63, 127, 255\}$ and $p = 20$. To speed up the convergence we introduce the preconditioner defined in (2.46) with $q \in \{16, 32\}$. Notice that theoretically, the TT-rank of $\mathbf{c}$ may become extremely large, leading to memory over-consumption and higher computational costs. To face this drawback, we impose a small TT-rank to $\mathbf{e}^{[\ell]}$, so that the TT-rank of $\mathbf{c}$ ends up being 11 at maximum. To study the bounds stated in Section 2.3.2, we need to comply with the hypothesis so that we scale each individual right-hand side by its norm, so that $\|\mathbf{c}\| = \sqrt{p}$.

As we can see in Figure 2.17A, TT-GMRES converges in 5 iterations for $n = 63$, in 7 for $n = 127$ and in 9 for $n = 255$. Figure 2.17B shows that the TT-rank of the last Krylov vector becomes quickly large, with maximum values ranging from 200 to 300. However looking at Figures 2.17C and 2.17D, the compression ratio for a single basis vector and for the entire basis remains extremely small, from 0.05 to 0.2 for the first one and from 0.02 and 0.14 for the entire basis, meaning that the TT approach is still effective from the memory point of view. As in the parametric operator case, we study the bounds expressed in Propositions 2.3.4 and 2.3.7. In Figure 2.18, we see that the bound for $\eta_{\mathbf{b}}$ is always quite tight, around 1 order of magnitude approximately. To use the result of Proposition 2.3.7, we set w equal to the number of iterations to converge and for every $\ell \in \{1, \dots, p\}$, we define the vector $\gamma_\ell \in \mathbb{R}^w$ such that

$$\gamma_\ell(i) = \psi_\ell(\mathbf{t}_k) \quad \text{for every} \qquad k \in \{1, \dots, w\}.$$

We define ℓ_m and ℓ_M as the indexes which realize the minimum and the maximum of γ_ℓ norm, i.e.,

$$\ell_m = \underset{\ell \in \{1,\dots,p\}}{\operatorname{argmin}} \|\gamma_\ell\| \quad \text{and} \quad \ell_M = \underset{\ell \in \{1,\dots,p\}}{\operatorname{argmax}} \|\gamma_\ell\| \,. \tag{2.50}$$

In this specific case for each grid point step, the value of ℓ_m and ℓ_M is reported in Figure 2.19. The same Figure shows that the bound in this specific case is quite good, with approximately less than 1 order of magnitude of difference, in the optimal and the worst case. Moreover, the three scaled "all-in-one" curves overlap from the second iteration, suggesting again that ρ^* from Corollary 2.3.8 is a good approximation of the scaling factors.

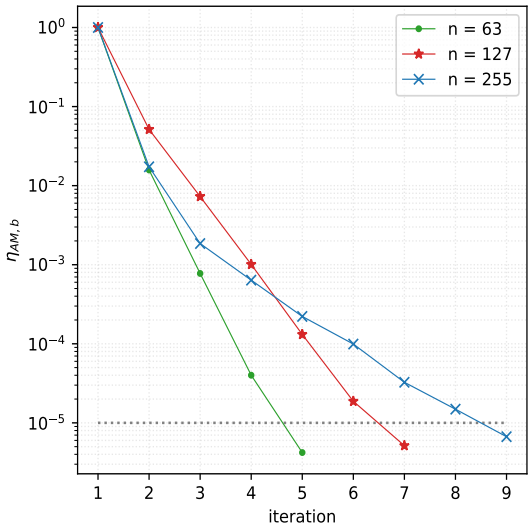
Convection-diffusion problem

As previously, this section aims to illustrate the solution of multiple convection-diffusion problem (2.41), with different right-hand sides. Let $\mathbf{A}_0$ be the discretization of (2.41) operator over a Cartesian grid of n points per mode for the domain $\Omega = [0, 1]^3$. Let $\mathbf{b} \in \mathbb{R}^{n \times n \times n}$ be the right-hand side discretization defined in Section 2.3.3.1. We define the individual linear system as

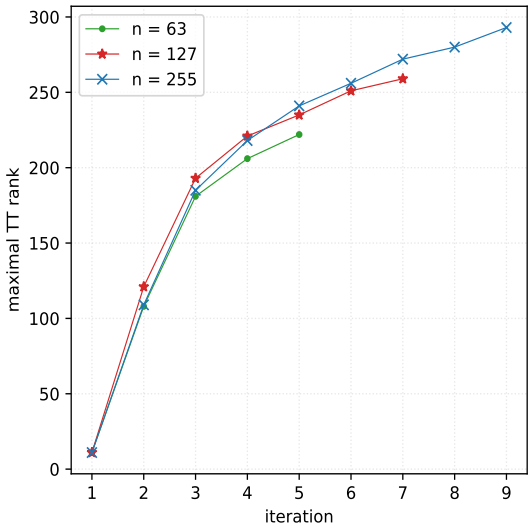
$$\mathbf{A}_0 \mathbf{y}_\ell = \mathbf{b} + \mathbf{e}_\ell$$

where $\mathbf{e}_\ell \in \mathbb{R}^{n \times n \times n}$ is a realization of the normal distribution $\mathcal{N}(0, 1)$ for every $\ell \in \{1, \dots, p\}$. Since the aim is solving simultaneously the p problems, as in Section 2.3.2, we define the "all-in-one" tensor linear operator $\mathbf{A} \in \mathbb{R}^{(p \times p) \times (n \times n) \times (n \times n) \times (n \times n)}$

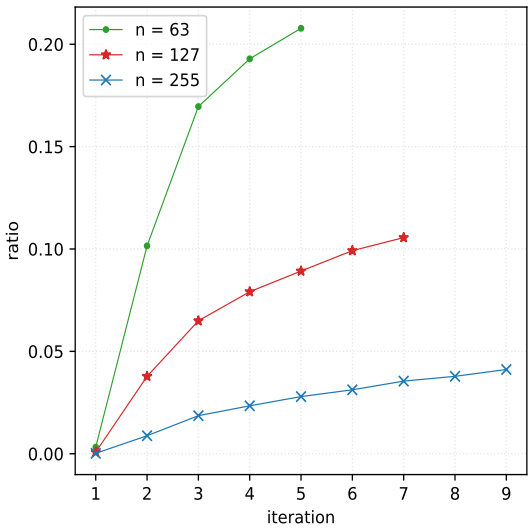
$$\mathbf{A} = \mathbb{I}_p \otimes (-\boldsymbol{\Delta}_3)$$



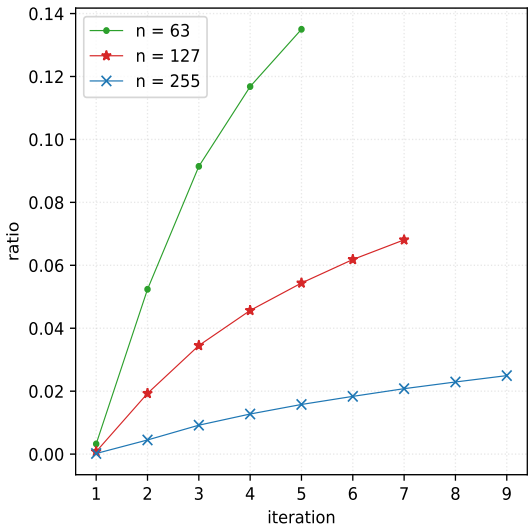
(A) Convergence history



(B) Maximal TT-rank of the last Krylov vector



(C) Compression ration for the last Kyrolv vector



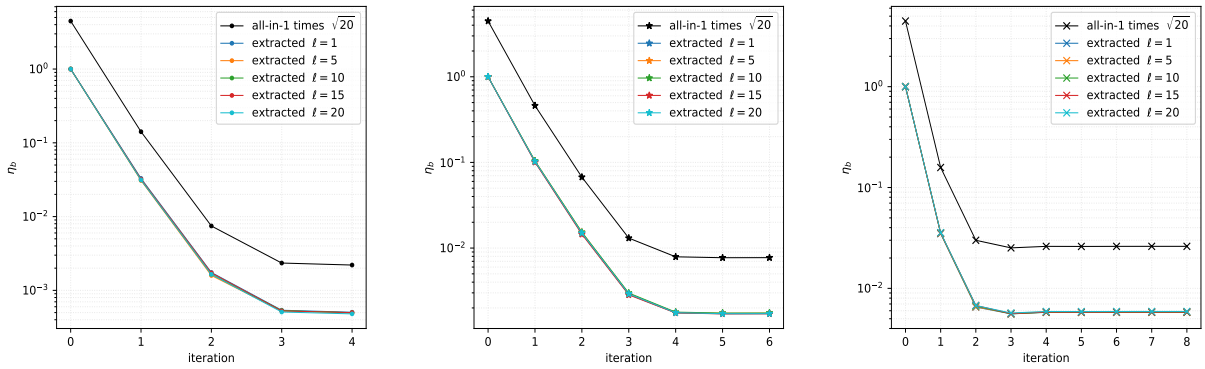
(D) Compression ratio for the entire Krylov basis

Figure 2.17 – 4-d multiple right-hand side Poisson problem using $\delta = \varepsilon = 10^{-5}$.

while the "all-in-one" right-hand side is $\mathbf{c} \in \mathbb{R}^{p \times n \times n \times n}$ such that

$$\mathbf{c}(\ell, i_1, i_2, i_3) = \mathbf{b}(i_1, i_2, i_3) + \mathbf{e}_\ell(i_1, i_2, i_3).$$

for every $i_k \in \{1, \ldots, n_k\}$, $\ell \in \{1, \ldots, p\}$ for $k \in \{1, \ldots, 3\}$. The problem is solved for $n \in \{63, 127\}$ and $p = 20$. As in all the previous cases of study, we use the preconditioner stated in (2.46) with $q \in \{16, 32\}$ and we impose a small TT-rank to $\mathbf{e}_\ell$ so that the

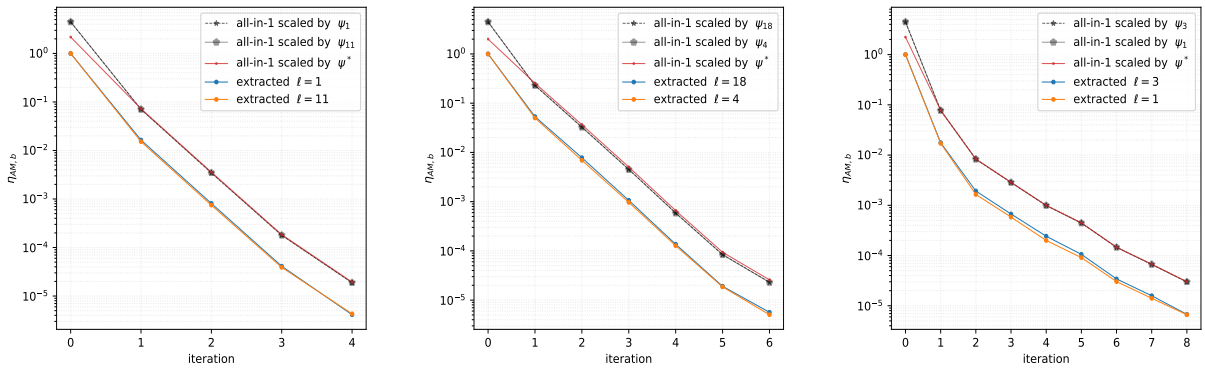


(A) Convergence history in $\eta_{\mathbf{b}}$ for $n = 63$

(B) Convergence history in $\eta_{\mathbf{b}}$ for $n = 127$

(C) Convergence history in $\eta_{\mathbf{b}}$ for $n = 127$

Figure 2.18 – 4-d Poisson problem $\eta_{\mathbf{b}}$ bound using $\delta = \varepsilon = 10^{-5}$.



(A) Convergence history in $\eta_{\mathbf{AM,b}}$ for $n = 63$

(B) Convergence history in $\eta_{\mathbf{AM,b}}$ for $n = 127$

(C) Convergence history in $\eta_{\mathbf{AM,b}}$ for $n = 127$

Figure 2.19 – 4-d multiple right-hand side Poisson problem $\eta_{\mathbf{AM,b}}$ bound using $\delta = \varepsilon = 10^{-5}$.

TT-rank of $\mathbf{c}$ ends up being 11 at maximum.

Figure 2.20A illustrates the convergence history in 5 iterations for both the grid sizes. If we compare Figure 2.20A with 2.10A, we observe that the curves are very similar. Generally speaking, the number of iterations for GMRES to converge, neglecting the effect of the rounding, is equal to the number of eigenvectors that span the subspace where the right-hand side lives. This implies that if all the right-hand sides belong to the same linear subspace, the number of iterations necessary to converge is the same, implying that under this hypothesis solving for 1 or p right-hand sides requires the same number of iterations. This point is further discussed in Section 2.3.3.3. From the point of view of the memory consumption, the comparison of Figure 2.10B and 2.20B shows that the solution for 20 right-hand sides leads to TT-rank significantly larger, from 25 to 30 in the single right-hand side solution versus more than 200 for the 20 right-hand

side "all-in-one" system. However, if we had solved 20 systems independently, summing all the TT-ranks, we could have reached a maximum of 500 up to 700. This becomes more interesting if we compare the compression ratios for the last Krylov vector, looking at Figure 2.10C and 2.20C. We have a ratio from 0.02 up to 0.12 for a single right-hand side solution versus 0.1 up to 0.17 for the simultaneous one, which shows that these ratios are extremely closed, considering that in the second case we are solving in a higher dimension. A similar argument holds for the ratio of compression of the entire Krylov basis. In Figure 2.20D, the ratio is between 0.06 and 0.12, while in Figure 2.10D it is between 0.01 and 0.07.

In Figure 2.21, we present the bound for $\eta_{\mathbf{b}}$ stated in Proposition 2.3.4. We see that it is quite tight during the first iterations and gets looser at the end, setting at more than 1 order of difference. As in the previous subsection, we compute ℓ_m and ℓ_M according to Equation (2.50), deciding which curves are plotted in Figure 2.22. The resulting bound, displayed in Figure 2.22, is quite tight, being of slightly less than 1 order of magnitude approximately, with the three scaled curves overlapping from the second iteration.

Multiple right-hand sides: a focus on eigenvectors

In this appendix, we study further the convergence of a multiple right-hand side problem. Indeed comparing the convergence history of the convection-diffusion problem, see Subsection 2.3.3.1 and of the multiple right-hand side convection-diffusion problem discussed in 2.3.3.3, notice that the number of iterations necessary to converge is equal, 5 in both cases. This appendix explains the causes of the phenomenon.

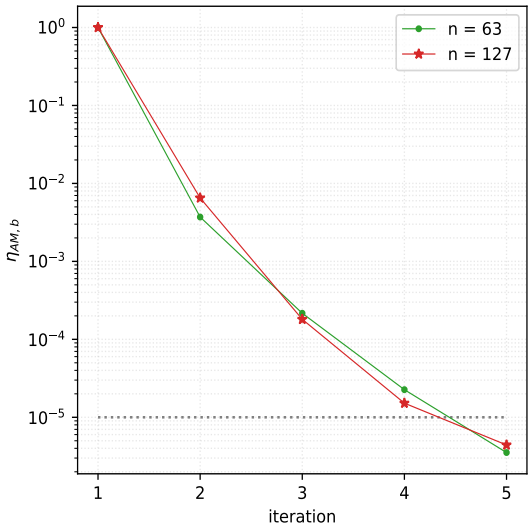
As we already explained, given a (tensor) linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ and the null tensor as initial guess, at the k-th iteration GMRES minimizes with respect to $\mathbf{x}$ the norm of the residual $\mathbf{A}\mathbf{x} - \mathbf{b}$ on the Krylov space of dimension k defined as

$$\mathcal{K}_k(\mathbf{A}, \mathbf{b}) = \text{span}\Big\{\mathbf{b}, \mathbf{A}\mathbf{b}, \dots, \mathbf{A}^{k-1}\mathbf{b}\Big\}$$

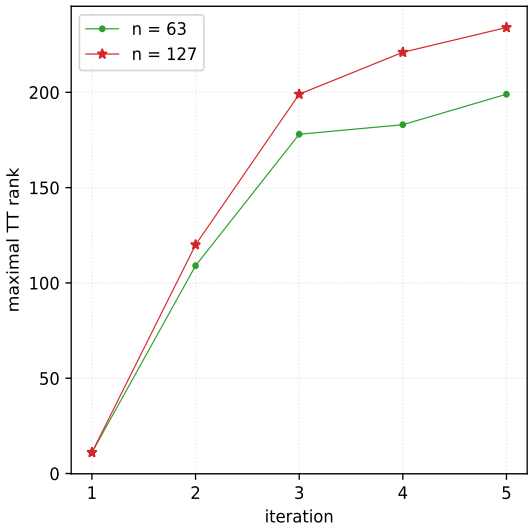
where $\mathbf{A}^h$ is obtained from h contractions over the indexes $(1, 3, \dots, 2d-1, 2d+1)$ of tensor operator $\mathbf{A}$. If $\mathbf{b}$ is equal to $\mathbf{e}_i$ an eigenvector of the tensor operator $\mathbf{A}$, then the Krylov space writes

$$\begin{aligned}\mathcal{K}_k(\mathbf{A}, \mathbf{b}) &= \text{span}\Big\{\mathbf{b}, \mathbf{A}\mathbf{b}, \dots, \mathbf{A}^{k-1}\mathbf{b}\Big\} \\ &= \text{span}\Big\{\mathbf{e}_i, \lambda_i\mathbf{e}_i, \dots, \lambda_i^{k-1}\mathbf{e}_i\Big\} \\ &= \text{span}\Big\{\mathbf{e}_i\Big\}\end{aligned}$$

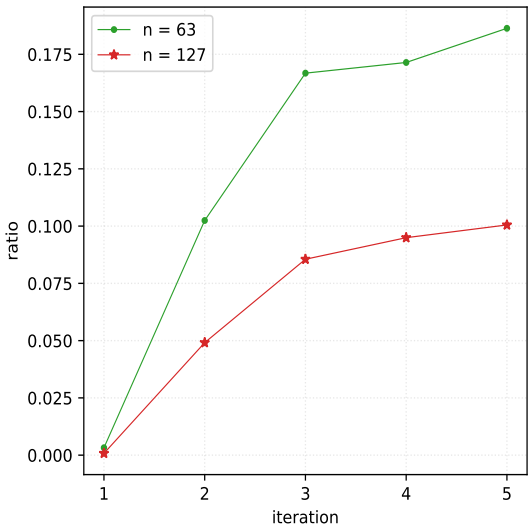
where λ_i is the i-th eigenvalue of $\mathbf{A}$. The Krylov space dimension is equal to 1, i.e., the number of eigenvectors $\mathbf{e}_i$ necessary to express the right-hand side. Theoretically, the number of iterations necessary to converge, i.e., the dimension of the Krylov space where the exact solution lives, is by linearity equal to the number of eigenvectors necessary to express the right-hand side as their linear combination. In the problems presented in 2.3.3.3, we add a random generated tensor to the chosen right-hand side. Comparing the results for a single right-hand side and multiple ones in the convection-diffusion problem,



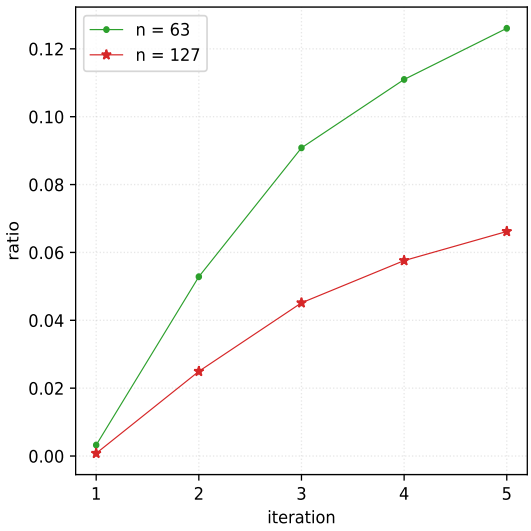
(A) Convergence history



(B) Maximal TT-rank of the last Krylov vector



(C) Compression ration for the last Kyrolv vector

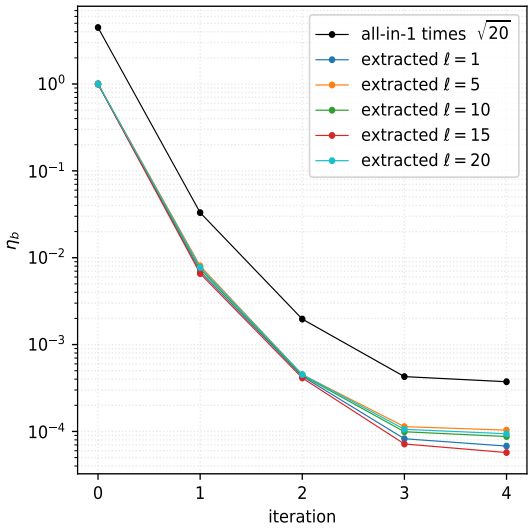


(D) Compression ratio for the entire Krylov basis

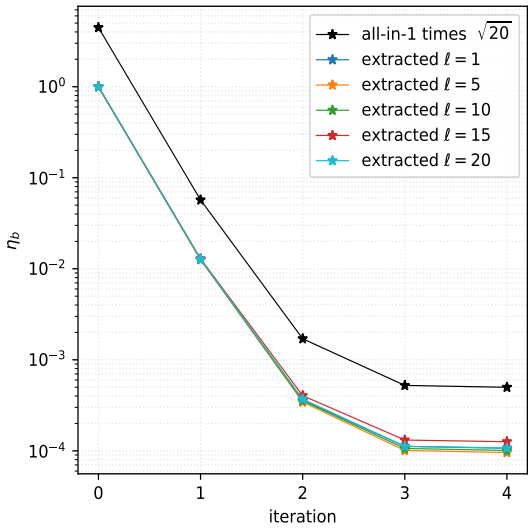
Figure 2.20 – 4-d multiple right-hand sides convection-diffusion problem using $\delta = \varepsilon = 10^{-5}$.

we may conclude that the introduced error has not increased the number of eigenvectors, for the tolerance chosen.

Let now consider a more peculiar problem with two right-hand sides, living in subspaces generated by different eigenvectors. More in detail, one right-hand side belongs to the subspace generated by a single eigenvector, while the other to the subspace generated

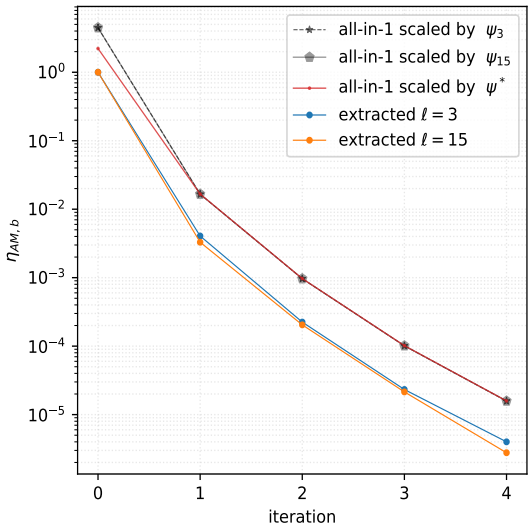


(A) Convergence history in $\eta_{\mathbf{b}}$ for $n = 63$

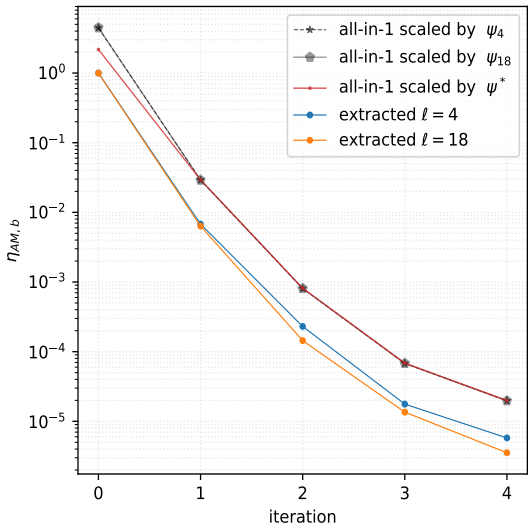


(B) Convergence history in $\eta_{\mathbf{b}}$ for $n = 127$

Figure 2.21 – 4-d convection-diffusion problem $\eta_{\mathbf{b}}$ bound using $\delta = \varepsilon = 10^{-5}$.



(A) Convergence history in $\eta_{\mathbf{AM,b}}$ for $n = 63$



(B) Convergence history in $\eta_{\mathbf{AM,b}}$ for $n = 127$

Figure 2.22 – 4-d multiple right-hand side convection-diffusion problem $\eta_{\mathbf{AM,b}}$ bound using $\delta = \varepsilon = 10^{-5}$.

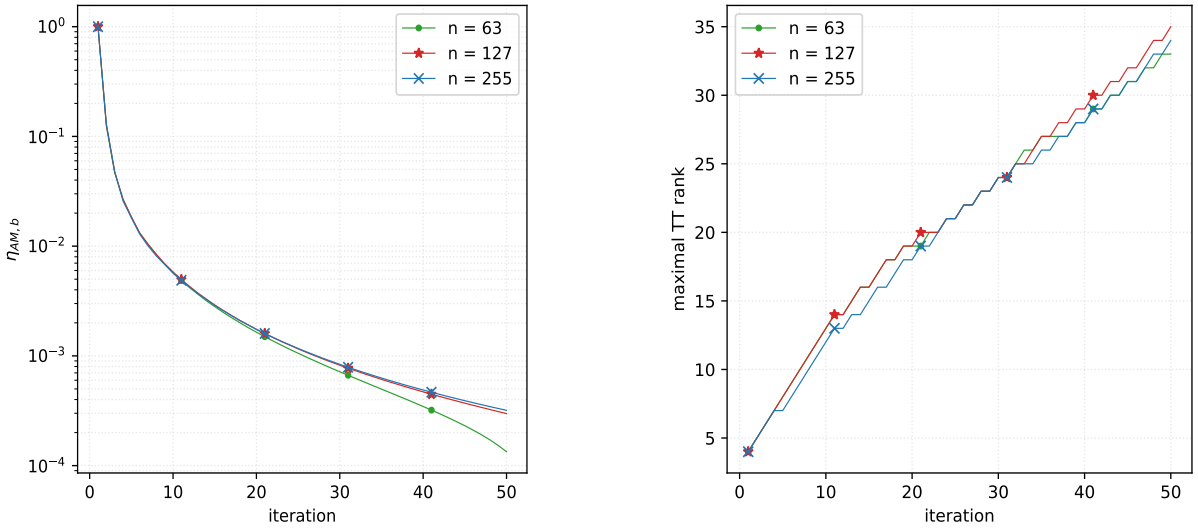
by other j different eigenvectors. Thanks to our previous argument, theoretically the two systems converge independently with a different number of iterations, one for the first and j for the second. When we solve the two systems together, we expect the "all-in-on" system to converge as the slowest one, i.e., as the slowest converging one. Let $\mathbf{e_1}, \ldots, \mathbf{e}_{j+1}$

be the first $(j+1)$ different eigenvectors of the 3-dimensional discrete Laplacian $-\mathbf{\Delta}_3$. We consider the two following linear systems

$$-\mathbf{\Delta}_3 \mathbf{y}_1 = \mathbf{e}_1 \tag{2.51}$$

$$-\mathbf{\Delta}_3 \mathbf{y}_2 = \sum_{\ell=2}^{j+1} \mathbf{e}_\ell. \tag{2.52}$$

As described in Section 2.3.2, we define the "all-in-one" linear system with the 'diagonal' tensor operator $\mathbf{A}$ and the "all-in-one" right-hand side $\mathbf{b}$. TT-GMRES is used to solve this "all-in-one" system for $j = 10$, for a grid step size equal to $n \in \{63, 127, 255\}$, without preconditioner, with tolerance ε and rounding accuracy δ equal to 10^{-5}, no restart and a maximum of 50 iterations. Figure 2.23A shows the convergence history of the problem with the three different grid sizes. Since there is no preconditioner the convergence is kind of slow if compared with Figure 2.17A. At the same time, the TT-ranks grow not too quickly, because both there is not a randomly generated error and there are just two right-hand sides. The residual curve associated with Equation 2.51 in blue is almost

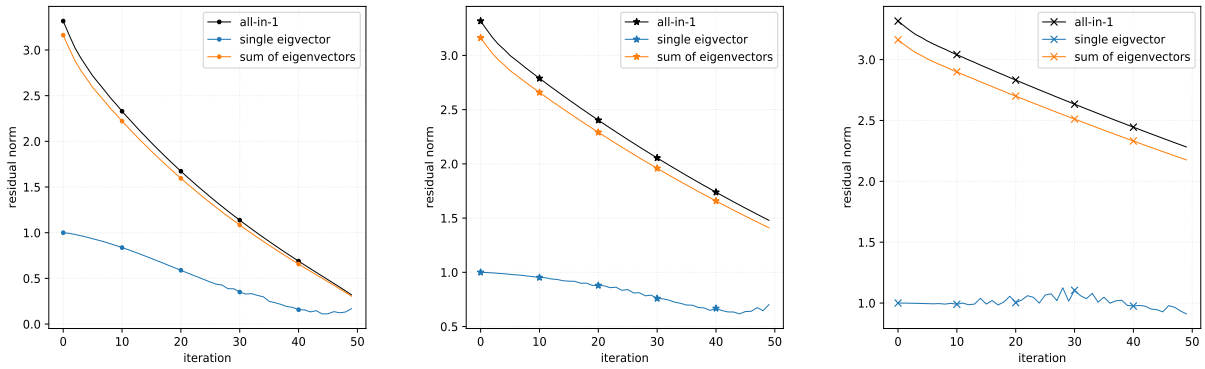


(A) Convergence history

(B) Maximal TT-rank of the last Krylov vector

Figure 2.23 – 4-d multiple right-hand side problem with eigenvector.

flattened for all the grid sizes, suggesting that GMRES almost immediately minimized it completely. On the other side the orange curve of problem (2.52) residual decreases slowly, meaning that minimizing its norm requires more steps. Lastly in all the plots, the "all-in-one" residual curve follows exactly the eigenvector sum problem residual, confirming that the general convergence of the "all-in-one" system is decided by the slowest converging system, that is our intuition, confirmed in for all the grid sizes in Figure 2.24. As last remark, the "all-in-one" doesn't converge in 10 iterations because of the rounding effect, which slows down the convergence.



(A) Residual norm for $n = 63$ (B) Residual norm for $n = 127$ (C) Residual norm for $n = 255$

Figure 2.24 – 4-d multiple right-hand side problem, residual comparison.

2.4 Conclusive remarks

The focus of this chapter is set on the backward stability of the iterative solver GMRES when the data are perturbed. We study numerically the GMRES backward stability in the framework, referred to as variable accuracy, where the storage and the computational accuracy are decoupled. This choice reflects and models the need of compressing data due to memory constraints, leading to perturbations in their representation. In particular, the storage perturbations are either synthetic in the matrix case or due to practical implementation choices, for example when an agnostic lossy compressor or the TT-formalism are applied in the matrix and tensor context respectively. Our experiments show that the backward stability of GMRES is maintained in the variable accuracy approach, whether they are component-wise, consistently with the theoretical results of [40, 113] either they are norm-wise. Our realization of GMRES in TT-format is further compared numerically with the previous relaxed version of TT-GMRES, presented in [39], concluding that our is significantly more robust than the older one. Moreover, we develop some theoretical backward stable bounds for the simultaneous solution of many order d tensor linear systems through a unique order $(d+1)$ tensor linear systems. The tightness of these bounds is numerically analysed too.

This chapter is formed by two main sections: the first, i.e., Section 2.2 dedicated to the matrix results, and the second, that is Section 2.3, to the tensor ones. After recalling the main linear algebra theorems related to the backward stability of GMRES in Section 2.2.1, we describe our variable accuracy approach in Section 2.2.2.1. We report about the numerical stability of GMRES when the perturbations due to compression simulations or to an agnostic lossy compressor are component-wise in Section 2.2.2 and in Section 2.2.3 when they are norm-wise.

The second half of this chapter, namely Section 2.3, aims to investigate the GMRES solver extended to the tensor framework through the TT-formalism, which is a particular case of the norm-wise perturbation in the variable accuracy approach. The TT-GMRES

algorithm structure is presented in Section 2.3.1. In Section 2.3.2, we focus on the solution of many tensor linear systems of order d whose operator or right-hand side depends on a parameter. After stating the assumptions, we illustrate how to mode many tensor linear systems of order d into a unique tensor linear system of order $(d+1)$. Moreover, we theoretically prove some bounds linking the solution of a single tensor linear system of order d with the corresponding slice extracted from the solution of the order $(d+1)$ tensor linear system, presenting the construction with dense format tensors and TT-format ones, in Section 2.3.2.1. These bounds are meant to guarantee a certain numerical quality when solving many tensor linear systems at one. All the numerical experiments are collected in Section 2.3.3. In particular, in Section 2.3.3.1, the numerical robustness of our TT-GMRES algorithm is shown in comparison with the relaxed implementation of GMRES in TT-format, proposed in [39]. Further experiments and outcomes related to the backward stability of our TT-GMRES solver are reported in Section 2.3.3.1 and 2.3.3.1. Except for the results in Section 2.3.3.1, all the others are obtained with the help of a tensor right preconditioner for the multilinear operator. The chosen preconditioner depends on several parameters, whose effects are numerically analysed in Section 2.3.3.1. The outcomes related to parameter-dependent multilinear operators are collected in Section 2.3.3.2, while the numerical outputs of parameter-dependent right-hand sides are found in Section 2.3.3.3.

Chapter 3

Orthogonalization schemes in TT-format

3.1 Introduction

Many numerical methods require to generate from any set of m independent vectors of $\mathbb{R}^n$ an orthogonal basis of the subspace of $\mathbb{R}^n$ spanned by these vectors. Different orthogonalization algorithms have been proposed during the years to accomplish this task. From another viewpoint, they allow also the computation of the matrix QR factorization. Indeed, if the input m vectors of $\mathbb{R}^n$ are organized into a matrix $A \in \mathbb{R}^{n\times m}$, then some orthogonalization schemes are able to factorize A into the product of an orthogonal matrix $Q \in \mathbb{R}^{n\times m}$ and an upper triangular one $R \in \mathbb{R}^{m\times m}$. Among the most important numerical algorithms, we consider the Classical Gram-Schmidt (CGS) [52, 127], the Modified Gram-Schmidt (MGS) [52, 127], their versions with re-orthogonalization, named CGS2 and MGS2 [1, 33, 116], the Gram approach [131] and the Householder transformation [76]. CGS and MGS are algorithms realizing the Gram-Schmidt method whose fundamental idea is sequentially removing from an input vector its projection along the previously computed orthonormal vectors and normalizing it after. The CGS2 and MGS2 procedures are meant to enhance the CGS and MGS qualities orthogonalizing once more in the same way the basis computed with CGS and the MGS respectively. The Gram method computes the orthogonal basis taking advantage of the Cholesky factorization of the Gram matrix, defined by the scalar product of input vector, while the Householder transformation relies on orthogonal reflections constructed from the input vector set and used to reflect the canonical basis.

A key point for these orthogonalization algorithms is their numerical stability in finite precision calculation, estimated by the *loss of orthogonality* that expresses how much the computational rounding errors affect the computed basis orthogonality. This aspect has been deeply studied throughout the years, leading to many interesting theoretical results, relating the loss of orthogonality to the linearly dependency of the input vectors. The authors of [15, 51] establish some theoretical bounds for CGS and MGS loss of orthogonality, showing that the basis produced by MGS is better in terms of orthogonality

than the CGS one. The bounds proved in [51] for CGS2 and MGS2, confirms that this reorthogonalization effectively improves the orthogonality of the computed basis. Bounds for the loss of orthogonality of the Householder transformation of the Gram method are proved in [143] and [131] respectively.

All the cited algorithms naturally translate in the tensor world, i.e., starting from a set of m tensors of $\mathbb{R}^{n_1 \times \cdots \times n_d}$, they produce an orthogonal basis for the relative subspace of dimension m of $\mathbb{R}^{n_1 \times \cdots \times n_d}$. Clearly, these orthogonalization schemes can work with dense tensors, see Equation (1.3), but as already stated, the storage and operation cost for full format tensor grow exponentially with the tensor order. Thus, it is convenient to work with compressed tensor format.

As inevitably happens with TT-algorithms, the sequence of operations defining the orthogonalization kernels in TT-format, described in Section 1.3.2, leads to a growth in the TT-ranks, making necessary some rounding steps that may affect the orthogonality quality of the basis. The objective of this chapter is to investigate both the orthogonality of the computed TT-vector basis and to relate the results with the theoretical results of classical numerical matrix computation. To complete the analysis, we study also a possible application of these kernels: the subspace iteration eigensolver [126], an eigensolver requiring orthogonalization schemes. In particular, we extend it to the tensor framework with the TT-formalism, defining the the SUBSPace ITeration in TT-format (TT-SUBSPIT), and we test it with the six orthogonalization kernels in TT-format, previously studied. Thus, the TT-SUBSPIT numerical performances, i.e., the quality of the eigenvalues, are investigated in relation with the orthogonalization kernels plugged into.

The remainder of this chapter is organized as follows. Section 3.2 begins with the description of the six orthogonalization schemes extended to the tensor context through the TT-formalism. We address also the complexity in terms of TT-rounding operations, representing the computationally most expensive operation. In Section 3.2.4 we recall briefly the classical numerical linear algebra theoretical bounds related the loss of orthogonality of these schemes. The theoretical results are linked with the numerical experiments, collected in Section 3.2.5, of the same orthogonalization schemes extended with the TT-format. The similarities between the classical orthgonalization kernels and their TT-versions are summarized in Section 3.2.6. The second Section 3.3 presents the TT-SUBSPIT algorithm, the TT-version of the subspace iteration eigensolver, as a case of study for these orthogonalization techniques. In Section 3.3.1 we describe the TT-SUBSPIT algorithm and in Section 3.3.2 we present the output of the numerical examples. In particular, we analyse the quality of the eigenvalue approximations, of the convergence and the memory requirements depending on the selected orthogonalization kernel.

3.2 Orthogonalization schemes

In the following sections, we describe the classical orthogonalization kernels, we propose their extensions to TT-vectors and we discuss their numerical stability comparing it with the theoretical results of classical matrix computation.

3.2.1 Classical and Modified Gram-Schmidt

In theoretical linear algebra the Gram-Schmidt process [52, 127] is a tool to produce an orthonormal basis from a given set of vectors. Let $\mathcal{A} = \{a_1, \ldots, a_m\}$ be a set of m linearly independent vectors of $\mathbb{R}^n$, then the key idea of the Gram-Schmidt process is to incrementally build an orthonormal basis of the space spanned by the elements of $\mathcal{A}$. At the i-th step, the i-th element a_i is made orthogonal to the previously computed $(i-1)$ orthonormal vectors $\{q_1, \ldots, q_{i-1}\}$, subtracting from a_i its projection along q_j, given by the scalar product of a_i and q_j for $j \in \{1, \ldots, i-1\}$. After normalization the new vector is q_i, the i-th vector of the final orthonormal basis. This mechanism is easily transported in the tensor framework. Consequently instead of stating the theory of Gram-Schmidt procedure with the tensor notation, we illustrate the two different realizations of this theoretical tool only in the TT-format. We carefully underline the differences with the classical matrix implementations.

3.2.1.1 Classical schemes without reorthogonalization

The direct implementation of the Gram-Schmidt process is known as *Classical Gram-Schmidt* (CGS) and its TT-version is sketched in Algorithm 11. Iteratively TT-CGS initializes $\mathbf{p}_i$ to $\mathbf{a}_i$ for every $i \in \{1, \ldots, m\}$, see line 3. Then in the core loop, the algorithm subtracts from $\mathbf{p}_i$ the projection of $\mathbf{a}_i$ along the $(i-1)$ previously computed tensors $\mathbf{q}_j$ of the new orthogonal basis, as described in lines 5 and 6. As last step, $\mathbf{p}_i$ is normalized and added in the new orthonormal basis $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_i\}$. The projections of $\mathbf{a}_i$ along $\mathbf{q}_j$ for $j \in \{1, \ldots, i-1\}$ define the i-th column of R, which is consequently upper triangular by construction. The i-th diagonal entry of R is the norm of $\mathbf{p}_i$ computed in line 9. These steps appear in both the tensor and in the matrix version of the CGS algorithm. However when dealing with compressed format tensors, we must ensure that different algorithm steps do not reduce significantly the compression quality. Since we are dealing with low-rank TT-vectors, we have to ensure that the TT-ranks stay small. Indeed assuming that $\mathbf{p}_k$ and $\mathbf{q}_j$ have maximum TT-rank r for every $j \in \{1, \ldots, k-1\}$, then after $(k-1)$ subtractions, i.e., after $(k-1)$ repetitions of line 6, the TT-rank of $\mathbf{p}$ will be bounded by kr. To limit the TT-rank growth, we introduce in line 8 the compression of $\mathbf{p}_i$ through the TT-rounding algorithm with accuracy δ, which is the most computationally expensive operation in orthogonalization algorithms. Consequently the TT-CGS complexity depends on the number of TT-rounding calls and their complexity, which is known to be $\mathcal{O}(dnr^3)$ where d is the order of the TT-vector rounded, n and r are the maximum of the mode size and of the TT-rank respectively. However in TT-CGS and in the other orthogonalization methods studied, the TT-rank is not always known, since we are prescribing a rounding accuracy δ. Therefore here and after, we estimate the complexity of the orthogonalization algorithm in terms of the number of TT-rounding operations. The complexity of TT-CGS is equal to m TT-rounding operations.

Algorithm 11 $\mathcal{Q}, R = \text{TT-CGS}(\mathcal{A}, \delta)$

1: **input:** $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$
2: **for** $i = 1, \ldots, m$ **do**
3: $\quad \mathbf{p} = \mathbf{a}_i$
4: $\quad$ **for** $j = 1, \ldots, i-1$ **do**
▷ compute the projection of $\mathbf{a}_i$ along $\mathbf{q}_j$
5: $\quad\quad R(i,j) = \langle \mathbf{a}_i, \mathbf{q}_j \rangle$
▷ remove the projection of $\mathbf{a}_i$ along $\mathbf{q}_j$
6: $\quad\quad \mathbf{p} = \mathbf{p} - R(i,j)\mathbf{q}_j$
7: $\quad$ **end for**
8: $\quad \mathbf{p} = \texttt{TT-round}(\mathbf{p}, \delta)$
9: $\quad R(i,i) = ||\mathbf{p}||$
10: $\quad \mathbf{q}_i = 1/R(i,i)\,\mathbf{p}$ ▷ normalize $\mathbf{p}$
11: **end for**
12: **return:** $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$, R

Algorithm 12 $\mathcal{Q}, R = \text{TT-MGS}(\mathcal{A}, \delta)$

1: **input:** $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$
2: **for** $i = 1, \ldots, m$ **do**
3: $\quad \mathbf{p} = \mathbf{a}_i$
4: $\quad$ **for** $j = 1, \ldots, i-1$ **do**
▷ compute the projection of $\mathbf{p}$ along $\mathbf{q}_j$
5: $\quad\quad R(i,j) = \langle \mathbf{p}, \mathbf{q}_j \rangle$
▷ remove the projection of $\mathbf{p}$ along $\mathbf{q}_j$
6: $\quad\quad \mathbf{p} = \mathbf{p} - R(i,j)\mathbf{q}_j$
7: $\quad$ **end for**
8: $\quad \mathbf{p} = \texttt{TT-round}(\mathbf{p}, \delta)$
9: $\quad R(i,i) = ||\mathbf{p}||$
10: $\quad \mathbf{q}_i = 1/R(i,i)\,\mathbf{p}$ ▷ normalize $\mathbf{p}$
11: **end for**
12: **return:** $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$, R

In the classical matrix framework Classical Gram-Schmidt is known for suffering from loss of orthogonality in the computed basis, as discussed later on, cf. [15]. The *Modified Gram-Schmidt* (MGS) algorithm introduces in the Classical Gram-Schmidt a tiny algorithmic change, which is sufficient to guarantee a generally better numerical orthogonality. Indeed, in TT-MGS we remove the projection along $\mathbf{q}_j$ of $\mathbf{p}_i$ and not of $\mathbf{a}_i$ for every $j \in \{1, \ldots, i-1\}$, as we see comparing line 5 of Algorithm 11 and 12 respectively. This modification reduces the error propagation, improving the general stability of the algorithm both in the classical matrix and in the tensor case, as we discuss in Section 3.2.4. For the remaining steps, the two algorithms are exactly identical. Under the same assumptions stated previously to estimate the complexity, we conclude that TT-MGS computational complexity in TT-format is equal to TT-CGS one, given by m TT-rounding calls.

3.2.1.2 Classical schemes with reorthogonalization

CGS and MGS are know to have stability issues, described in details in Section 3.2.4, i.e., the close the input vectors are to linear dependency, the more the algorithms propagate the rounding errors, spoiling the final orthogonality of the new basis. As reported in [51], over the years, different articles, as for example [1, 33, 73], addressed this flaw introducing re-orthogonalization steps, that is, orthogonalizing repeatedly through the same approach the basis produced by the algorithm itself. In [51], the authors showed theoretically that one re-orthogonalization step is sufficient to improve significantly the orthogonality of the new basis generated by CGS and MGS. We present briefly the idea of CGS and MGS with re-orthogonalization, called CGS2 and MGS2, in the tensor case, highlighting those steps performed only in the TT-version.

In CGS2, sketched in Algorithm 13, as in CGS, the input TT-vector $\mathbf{a}_i$ is orthogonalized with respect to the $(i-1)$ previously computed orthogonal TT-vector $\{\mathbf{q}_1, \ldots, \mathbf{q}_{i-1}\}$,

subtracting from $\mathbf{a}_i$ its projection along $\mathbf{q}_j$. The projection of $\mathbf{a}_i$ along $\mathbf{q}_j$ defines the (j,i) element of the first matrix R_1. With these first $(i-1)$ iterations, given in line 6 of Algorithm 13, we define the TT-vector $\mathbf{p}_1$, which is then rounded in the TT-GCS2, see Algorithm 13 line 10. Up to this point, TT-CGS2 acts exactly as TT-CGS. However in TT-CGS2, after the rounding, the TT-vector $\mathbf{p}_1$ is orthogonalized again against $\{\mathbf{q}_1, \ldots, \mathbf{q}_{i-1}\}$, defining $\mathbf{p}_2$. The projections along $\mathbf{q}_j$ of $\mathbf{p}_1$ determine the (j,i) component of the second matrix R_2. Once $\mathbf{p}_2$ is completely defined, it is rounded and normalized, defining the i-th orthogonal TT-vector $\mathbf{q}_i$, as stated in line 12 and 13 of Algorithm 13. The norm of $\mathbf{p}_2$ at the i-th iteration defines the (i,i)-th diagonal component of R_2. The R factor from the QR decomposition computed by CGS2 is obtained summing R_1 and R_2. The difference of MGS2 from CGS2 appears in line 7 of Algorithm 14 and 13 respectively. During the first orthogonalization loop, that is for $k=1$ in line 4 of both methods, in the classical Gram-Schmidt version $\mathbf{a}_i$ is projected along $\mathbf{q}_j$ defining $\mathbf{p}_1$, while in the modified Gram-Schmidt one $\mathbf{p}_1$ is projected along $\mathbf{q}_j$ updating itself. In the second orthogonalization loop, i.e., when $k=2$ in line 4 of both algorithms, in TT-CGS2 it is removed from $\mathbf{p}_1$ its projection along $\mathbf{q}_j$ defining $\mathbf{p}_2$, while in the TT-MGS2 version, it is $\mathbf{p}_2$ the TT-vector projected along $\mathbf{q}_j$ and updated. All the remaining steps, including the TT-rounding and the construction of R_1, R_2 and their sum R, are identical between TT-CGS2 and TT-MGS2. Remark that the rounding steps are performed only in the TT-version of MGS2 and CGS2.

Algorithm 13 $Q, R = \text{TT-CGS2}(\mathcal{A}, \delta)$

1: **input:** $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$
2: **for** $i = 1, \ldots, m$ **do**
3: $\quad \mathbf{p}_0 = \mathbf{a}_i$
▷ repeat twice the orthogonalization loop
4: $\quad$ **for** $k = 1, 2$ **do**
5: $\quad\quad \mathbf{p}_k = \mathbf{p}_{k-1}$
6: $\quad\quad$ **for** $j = 1, \ldots, i-1$ **do**
▷ compute the projection of $\mathbf{p}_{k-1}$ along $\mathbf{q}_j$
7: $\quad\quad\quad R_k(i,j) = \langle \mathbf{p}_{k-1}, \mathbf{q}_j \rangle$
▷ subtract the projection of $\mathbf{p}_{k-1}$ along $\mathbf{q}_j$
8: $\quad\quad\quad \mathbf{p}_k = \mathbf{p}_k - R_k(i,j)\mathbf{q}_j$
9: $\quad\quad$ **end for**
10: $\quad\quad \mathbf{p}_k = \texttt{TT-round}(\mathbf{p}_k, \delta)$
11: $\quad$ **end for**
12: $\quad R_2(i,i) = ||\mathbf{p}_2||$
13: $\quad \mathbf{q}_i = 1/R_2(i,i)\mathbf{p}_2$ ▷ normalize $\mathbf{p}_2$
14: **end for**
▷ compute the R factor from the repeated orthogonalization loop
15: $R = R_1 + R_2$
16: **return:** $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$, R

Algorithm 14 $Q, R = \text{TT-MGS2}(\mathcal{A}, \delta)$

1: **input:** $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$
2: **for** $i = 1, \ldots, m$ **do**
3: $\quad \mathbf{p}_0 = \mathbf{a}_i$
▷ repeat twice the orthogonalization loop
4: $\quad$ **for** $k = 1, 2$ **do**
5: $\quad\quad \mathbf{p}_k = \mathbf{p}_{k-1}$
6: $\quad\quad$ **for** $j = 1, \ldots, i-1$ **do**
▷ compute the projection of $\mathbf{p}_k$ along $\mathbf{q}_j$
7: $\quad\quad\quad R_k(i,j) = \langle \mathbf{p}_k, \mathbf{q}_j \rangle$
▷ subtract the projection of $\mathbf{p}_k$ along $\mathbf{q}_j$
8: $\quad\quad\quad \mathbf{p}_k = \mathbf{p}_k - R_k(i,j)\mathbf{q}_j$
9: $\quad\quad$ **end for**
10: $\quad\quad \mathbf{p}_k = \texttt{TT-round}(\mathbf{p}_k, \delta)$
11: $\quad$ **end for**
12: $\quad R_2(i,i) = ||\mathbf{p}_2||$
13: $\quad \mathbf{q}_i = 1/R_2(i,i)\mathbf{p}_2$ ▷ normalize $\mathbf{p}_2$
14: **end for**
▷ compute the R factor from the repeated orthogonalization loop
15: $R = R_1 + R_2$
16: **return:** $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$, R

On the basis of the previously used hypothesis, the computational complexity of TT-CGS2 and TT-MGS2 is estimated as $2m$ TT-rounding operations, since during the m

iterations we have two temporary TT-vectors $\mathbf{p}_1$ and $\mathbf{p}_2$ rounded.

3.2.2 Gram approach

In [131], the authors propose an algorithm, that we refer to as Gram's algorithm, to generate orthogonal basis starting from a set m linearly independent vectors of $\mathbb{R}^n$ with $m \ll n$. This scheme is based on the Gram matrix, which under the hypothesis $m \ll n$ is significantly small. The key idea is decomposing the small Gram matrix through its Cholesky factorization and benefiting from it to generate the orthogonal basis. We briefly describe the main ideas of this orthogonalization scheme in the classical matrix framework and we describe in details the implementation of the Gram algorithm in the TT-format. Indeed the tensor realization of this procedure is extremely close to the matrix one, so that describing the tensor case and its differences with the classical matrix one is sufficient to ensure a good comprehension.

Given a set of vectors $\mathcal{A} = \{a_1, \ldots, a_m\}$ with $a_i \in \mathbb{R}^n$, the Gram matrix $G \in \mathbb{R}^{m\times m}$ is defined by the their scalar product as $G(i,j) = \langle a_i, a_j \rangle$ for every $i, j \in \{1, \ldots, m\}$. Equivalently, let a_i be i-th column of the matrix $A \in \mathbb{R}^{n\times m}$, then in the matrix computation the Gram matrix is

$$G = A^\top A. \tag{3.1}$$

If the elements of $\mathcal{A}$ are linearly independent, then G is symmetric positive definite. As consequence, its Cholesky factorization exists and is expressed as $G = LL^\top$ with $L \in \mathbb{R}^{m\times m}$ a lower triangular matrix. Let now denote the transpose of L by R, then the Gram matrix is

$$G = R^\top R. \tag{3.2}$$

Comparing Equation (3.1) and (3.2), we conclude that R is the R factor from the QR decomposition of A, i.e., it expresses the same information of A in a different basis, up to the action of an invertible matrix. The matrix Q from the QR decomposition of A is written as $Q = AR^{-1}$ where $R = L^\top$. The columns of Q form an orthogonal basis $\mathcal{Q}= \{q_1, \ldots, q_m\}$, whose j-th element strictly speaking is a linear combination of the first j elements of $\mathcal{A}$, i.e.,

$$q_j = \sum_{k=1}^{j} R^{-1}(k,j) a_k.$$

Remark 3.2.1. Notice that by construction the condition number of G is the square of the one of A. Consequently if the condition number of A associated with the set of input vectors $\mathcal{A}$ is larger than the inverse of the squared root of the working precision of the considered arithmetic, e.g., $u_{64} \approx 10^{-16}$ for 64-bit calculation, the associated Gram matrix G is numerically singular and its Cholesky decomposition is no longer defined. This constitutes the main practical drawback of this method.

This procedure produces an orthonormal basis starting from a set of linear independent vectors, which is naturally extended to TT-vectors. As described in Algorithm 15, given a set of TT-vectors $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ with $\mathbf{a}_i \in \mathbb{R}^{n_1\times\cdots\times n_d}$, we construct the Gram matrix

$G \in \mathbb{R}^{m\times m}$ through the tensor scalar product and we compute its Cholesky factorization getting the lower triangular matrix $L \in \mathbb{R}^{m\times m}$. As in the matrix case, $R \in \mathbb{R}^{m\times m}$ the transpose of L expresses the same information as the TT-vectors of $\mathcal{A}$, but with respect to a different basis. Following the matrix approach, we retrieve this basis, i.e., the orthonormal set $\mathcal{Q}$ whose element $\mathbf{q}_i \in \mathbb{R}^{n_1\times\cdots\times n_d}$ is defined as

$$\mathbf{q}_i = \sum_{k=1}^{i} R^{-1}(k,i)\mathbf{a}_k.$$

Remark 3.2.2. In the matrix framework, the orthogonal vector q_j are obtained from the elements of $\mathcal{A}$ by back-substitutions involving the R factor. However this approach cannot be easily translated into the tensor framework, where the inverse of R has to be explicitly computed.

As for the other orthogonalization techniques, in the tensor case, we prevent memory issues, monitoring the TT-ranks. Indeed, assuming that all the TT-vectors of $\mathcal{A}$ have TT-ranks bounded by r, then the i-th TT-vector constructed in line 12 has maximum TT-rank bounded by ir. Since this value grows linearly with m, in line 14 we introduce a rounding step at prescribed accuracy δ. As in Sections 3.2.1 and 3.2.2, notice that the

Algorithm 15 $\mathcal{Q}, R = \text{TT-Gram}(\mathcal{A}, \delta)$

1: **input:** $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$
2: **for** $i = 1, \dots, m$ **do**
3: **for** $j = 1, \dots, i$ **do**
▷ construct the Gram matrix through the inner product of the input TT-vectors
4: $G(i,j) = G(j,i) = \langle \mathbf{a}_i, \mathbf{a}_j \rangle$
5: **end for**
6: **end for**
7: $L = \texttt{cholesky}(G)$ ▷ compute the Cholesky factorization
8: $R = L^\top$ and $R^{-1} = \texttt{invert}(R)$ ▷ define the R factor of the QR-factorization
9: **for** $i = 1, \dots, m$ **do**
10: $\mathbf{p} = R^{-1}(i,1)\mathbf{a}_1$
11: **for** $j = 2, \dots, i$ **do**
▷ construct the i-th new basis TT-vector as a linear combination of the $(i-1)$ input TT-vectors
12: $\mathbf{p} = \mathbf{p} + R^{-1}(i,j)\mathbf{a}_j$
13: **end for**
14: $\mathbf{q}_i = \texttt{TT-round}(\mathbf{p}, \delta)$ ▷ round the TT-vector before adding it to the basis
15: **end for**
16: **return:** $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$, R

complexity of the TT-Gram algorithm is given by m TT-rounding operations. However, in this particular case, we can even estimate the cost of each single rounding step and consequently of the entire algorithm. Indeed, the maximum TT-rank of the rounded TT-vector $\mathbf{q}_i$ is bounded by $i\,r$, under the assumption that the maximum TT-rank and the maximum mode size of $\mathbf{a}_i$ are bounded by $r \in \mathbb{N}$ and $n \in \mathbb{N}$ respectively. Consequently,

the computational cost, that is the number of floating point operations, of each rounding operation, the most expensive step in the entire algorithm, is known and is equal $\mathcal{O}(dni^3r^3)$; summing up over $i \in \{1, \dots, m\}$, we conclude that the TT-Gram algorithm cost is $\mathcal{O}(dnm^4r)$ floating point operations.

3.2.3 Householder reflections

In the classical matrix framework, the Householder transformations are widely applied to generate orthogonal base, thanks to their remarkable stability properties, illustrated in the following sections. We briefly present the theoretical construction of a Householder transformation, underlining how to generate an orthogonal basis. The second half of this section describes in detail how we extend the Householder transformation to the tensor context, with a specific attention to the implementation of the Householder orthogonalization scheme in TT-format.

Given a vector $x \in \mathbb{R}^n$, the Householder reflector moves x along a chosen direction, which usually is an element of the canonical basis or a linear combination of them. We illustrate the construction of the Householder reflector in the general case. Let $x \in \mathbb{R}^n$ be the vector we want to reflect along the normalized vector $y \in \mathbb{R}^n$, the *Householder reflection or transformation* is a linear operator $\boldsymbol{\mathcal{H}} : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\boldsymbol{\mathcal{H}}(x) = \|x\| \, y \qquad \text{with} \qquad \|y\| = 1.$$

The Householder reflector is represented with respect to the canonical basis of $\mathbb{R}^n$ by the matrix $H \in \mathbb{R}^{n \times n}$ such that $H = \mathbb{I}_n - 2u \otimes u$ where $u \in \mathbb{R}^n$ is the *Householder vector* defined as

$$u = \frac{x - z}{\|x - z\|} \qquad \text{with} \qquad z = \|x\| \, y. \tag{3.3}$$

The matrix H representing the Householder reflection is unitary and entirely defined by the Householder vector u. Moreover the action of an Householder reflector is computed by one scalar product with the Householder vector u and one algebraic vector summation. Indeed given a vector $z \in \mathbb{R}^n$ and an Householder reflector $H = \mathbb{I}_n - 2u \otimes u$, the image of w through H is

$$Hw = w - 2\langle w, \, u \rangle u. \tag{3.4}$$

In particular, if u is defined as in Equation (3.3), then we explicitly verify that $Hx = \|x\| \, y$. Firstly remark that

$$\begin{aligned} \|x - z\|^2 &= \langle x - \|x\| \, y, \, x - \|x\| \, y \rangle \\ &= \|x\|^2 - 2\|x\| \, \langle x, \, y \rangle + \|x\|^2 \|y\|^2 \\ &= 2\Big(\|x\|^2 - \|x\| \, \langle x, \, y \rangle\Big) \end{aligned}$$

since $\|y\| = 1$ by hypothesis. Thanks to this result and Equation (3.4), we get

$$\begin{aligned} Hx &= x - 2\langle x,\, u\rangle u \\ &= x - \frac{2}{2\Big(\|x\|^2 - \|x\|\,\langle x,\, y\rangle\Big)}\langle x,\, x - \|x\|\,y\rangle(x - \|x\|\,y) \\ &= x - \frac{1}{\Big(\|x\|^2 - \|x\|\,\langle x,\, y\rangle\Big)}\Big(\|x\|^2 - \|x\|\,\langle x,\, y\rangle\Big)(x - \|x\|\,y) \\ &= \|x\|\,y. \end{aligned}$$

The Householder transformations are generally used to compute the QR factorization of a matrix, but they find application also in situations where a set of vectors has to be converted into an orthogonal basis. We examine briefly the two possibilities. Given a matrix $A \in \mathbb{R}^{n\times m}$, we construct m Householder reflections such that the k-th one moves the k-th column of A along a linear combination of the first k canonical basis vectors. Said differently, the k-th Householder transformation sets to zero the last $(n-k)$ entries of the k-th column of A. Consequently, after m Householder reflections the matrix A ends up being upper triangular. We illustrate more in detail how the algorithm iteratively proceeds. Let $a_1 \in \mathbb{R}^n$ be the first column of A, the first step consists of reflecting it along the first canonical basis vector e_1, i.e., in constructing the Householder reflector H_1 such that $H_1 a_1 = \|a_1\|\,e_1$. Consequently the first Householder vector $u_1 \in \mathbb{R}^n$ is

$$u_1 = a_1 \pm \|a_1\|\,e_1$$

and normalized. For stability reasons, cf. [134], the sign of the norm of a_1 is determined by the sign of the first component of a_1, that is a plus if $a_1(1) > 0$, a minus otherwise. The first Householder reflector H_1 is applied to all the columns of A, that is $\tilde{a}_j = H_1 a_j$ for $j \in \{1, \dots, m\}$. Henceforth we denote by $\tilde{a}_j$ the j-th column of A updated by all the previously defined $(j-1)$ Householder transformations for $j \in \{2, \dots, m\}$. Notice that the first Householder transformation moves the first column of A along a multiple of the first canonical basis vector e_1, i.e., it sets the last $(n-1)$ entries of the first column of A to zero. Then we proceed reflecting the second column of A, updated by H_1 along a linear combination of the first two canonical basis vectors e_1 and e_2. Let $u_2 \in \mathbb{R}^n$ be the Householder vector defining the second Householder reflector H_2, which updates a second time $\tilde{a}_j$ the j-th column of A for $j \in \{2, \dots, m\}$. Thus, at this point $\tilde{a}_2$ only has its first two components nonzero. The k-th Householder reflection H_k moves $\tilde{a}_k \in \mathbb{R}^n$ the k-th column of A, updated by the first $(k-1)$ Householder reflections, along a linear combination of the first k elements of the canonical basis of $\mathbb{R}^n$, i.e., $H_k \tilde{a}_k = \sum_{\ell=1}^{k} \alpha_\ell e_\ell$ with $\sqrt{\sum_{\ell=1}^{k} \alpha_\ell^2} = ||\tilde{a}_k||$. Before normalization, the k-th Householder vector is $u_k \in \mathbb{R}^n$ such that

$$u_k = \tilde{a}_k - \sum_{\ell=1}^{k} \beta_\ell e_\ell \tag{3.5}$$

where for every $\ell \in \{1, \dots, k-1\}$ we have

$$\beta_\ell = \tilde{a}_k(\ell) \qquad \text{and} \qquad \beta_k = \pm\sqrt{||\tilde{a}_k||^2 - \sum_{\ell=1}^{k-1} \beta_\ell^2}.$$

Again for stability reasons, cf. [134], β_k is positive if $\tilde{a}_k(k)$ is positive, negative otherwise. Then u_k is normalized.

Remark 3.2.3. By construction the first $(k-1)$ entries of u_k are zeros, the last $(n-k)$ ones are equal to the corresponding ones of a_k. The k-th component of u_k is given by the difference of the k-th component of a_k and the quantity β_k, that is $u_k(k) = \tilde{a}_k(k) - \beta_k$. Said differently, only the last $(n-k+1)$ entries of $\tilde{a}_k$ have a determinant role. Thanks to this property, in the matrix context, the Householder QR factorization has a reduced and simplified construction. The k-th Householder vector is $\hat{u}_k \in \mathbb{R}^{n-k+1}$ defined from the norm of $\tilde{a}_k$ with the first $(k-1)$ entries being zeros, i.e.,

$$\hat{u}_k = \gamma_k e_1 \qquad \text{where} \qquad \gamma_k = \sqrt{\sum_{j=k}^{n} \big(\tilde{a}_k(j)\big)^2}$$

with $e_1 \in \mathbb{R}^{n-k+1}$ for every $k \in \{1, \dots, m\}$. The k-th Householder transformation $H_k \in \mathbb{R}^{(n-k+1)\times(n-k+1)}$ is defined from $\hat{u}_k$ and is applied only on the last $(n-k+1)$ components of $\tilde{a}_j$ for $j \in \{k, \dots, m\}$ and $k \in \{1, \dots, m\}$. This reduced approach is not reproducible in the tensor framework, where the object is expressed in compressed format. Therefore we present it even in the matrix case in the most general way, that is not the one adopted to make an implementation in matrix computation.

Once the k-th Householder transformation has been applied to $\tilde{a}_j$ for $j \in \{k, \dots, m\}$, the vector $\tilde{a}_k$ has its last $(n-k)$ component equal to zero. The application of m Householder reflections to A leads to the upper triangular matrix R, i.e., the R factor of the QR decomposition. To compute the Q factor, we multiply the m Householder reflection matrices. For the majority of the applications, forming the Householder vectors and knowing the Householder transformations implicitly, as given in Equation (3.4), is sufficient. However, if we want to produce an orthogonal basis out of a generic set of m vectors $\mathcal{A} = \{a_1, \dots, a_m\}$ with $a_k \in \mathbb{R}^n$, a further step has to be considered. As for computing the QR factorization, H_k the k-th Householder transformation is defined by the k-th Householder vector u_k, given in Equation (3.5), for every $k \in \{1, \dots, m\}$. To generate the set of orthonormal vectors $\mathcal{Q} = \{q_1, \dots, q_m\}$, we apply H_k the first k Householder transformations to the k-th canonical basis vector e_k in reverse order, i.e.,

$$q_k = H_1 \cdots H_k e_k.$$

As in the previous sections, we extend with some modifications this approach to the tensor case. Let $\mathcal{A}$ be a set of m TT-vectors $\mathbf{a}_i \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ for every $i \in \{1, \dots, m\}$. To construct the Householder transformations, we first define a *canonical* basis for a tensor subspace of dimension m of $\mathbb{R}^{n_1 \times \cdots \times n_d}$. In order to do so, fixed $\mathcal{N}_i = \{1, \dots, n_i\}$ for every

$i \in \{1, \dots, d\}$ and $n = \prod_{j=1}^{d} n_j$, let define the function $\psi : \mathcal{N}_1 \times \cdots \times \mathcal{N}_d \to \{1, \dots, n\}$ such that

$$\psi(i_1, \dots, i_d) = i_1 + \sum_{\alpha=2}^{d} (i_\alpha - 1) m_\alpha \qquad \text{with} \qquad m_\alpha = \prod_{\beta=1}^{\alpha-1} n_\beta.$$

Since ψ is invertible, we denote its inverse by $\phi : \{1, \dots, n\} \to \mathcal{N}_1 \times \cdots \times \mathcal{N}_d$ such that $\phi(i) = (i_1, \dots, i_d)$. As consequence, $\psi(\phi(i)) = i$ and $\phi(\psi(i_1, \dots, i_d)) = (i_1, \dots, i_d)$ for $i \in \{1, \dots, n\}$ and $i_k \in \{1, \dots, n_k\}$ with $k \in \{1, \dots, d\}$. The basis we fix for the subspace of dimension m of $\mathbb{R}^{n_1 \times \cdots \times n_d}$ is $\mathcal{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_m\}$ with

$$\mathbf{e}_i = e_{i_1} \otimes \cdots \otimes e_{i_d} \qquad \text{with} \qquad (i_1, \dots, i_d) = \phi(i), \tag{3.6}$$

where e_{i_k} is the i_k-th canonical basis vector of $\mathbb{R}^{n_k}$ for $k \in \{1, \dots, d\}$. Henceforth we use index i to denote the i-th element of the canonical basis, implying $i = \psi(i_1, \dots, i_d)$.

As stated in Remark 3.2.3, the first $(k-1)$ components of u_k are zeros, the last $(n-k)$ components are equal to corresponding ones of $\tilde{a}_k$, while the k-th one is given by the difference of the k-th component of $\tilde{a}_k$ and the quantity β_k, defined in Equation (3.5). We want to transport this structure in the tensor case. However if the element of $\mathcal{A}$ are in TT-format, it is not possible to directly access to the tensor components and we need to recover them, either by multiplying the TT-cores with the correct index or by computing the scalar product with the element of $\mathcal{E}$. Let $\tilde{\mathbf{a}}_k$ be the result of $(k-1)$ Householder reflections applied to $\mathbf{a}_k$, the k-th Householder TT-vector is $\mathbf{u}_k \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ defined as

$$\mathbf{u}_k = \tilde{\mathbf{a}}_k - \sum_{j=1}^{k} R(j,k) \mathbf{e}_j$$

where $R(j,k) = \langle \tilde{\mathbf{a}}_k, \mathbf{e}_j \rangle$ for every $j \in \{1, \dots, k\}$ and $R(k,k) = \pm \sqrt{||\tilde{\mathbf{a}}(k)_k||^2 - \sum_{\ell=1}^{k-1} R(\ell,k)^2}$ as described in lines 8 and 10 of Algorithm 16 respectively. The k-th component of r_k takes a positive sign if $\langle \tilde{\mathbf{a}}_k, \mathbf{e}_k \rangle > 0$, otherwise it takes a negative sign, extending in the tensor framework the stability preserving idea given in [134]. The j-th component of r_k corresponds to the (j,k) component of the R factor. As previously pointed out, we have to ensure that the TT-rank of $\mathbf{u}_k$ remains small for every $k \in \{1, \dots, m\}$. Assuming that the maximum TT-rank of $\tilde{\mathbf{a}}_k$ is bounded by r_a, then after removing the first $(k-1)$ components of $\tilde{\mathbf{a}}_k$, i.e., after line 8, the TT-rank of $\mathbf{u}_k$ is bounded by $(r_a + k - 1)$. We introduce at this step the first TT-rounding call on the Householder TT-vector $\mathbf{u}_k$, whose TT-rank is decreased depending on the accuracy value δ. Then the k-th component of r_k is subtracted, determining a further growth in the TT-rank of $\mathbf{u}_k$. Since $\mathbf{u}_k$ plays a key role in the Householder transformation and consequently its TT-rank has a strong impact on the entire process, we decide to perform a further TT-rounding step over $\mathbf{u}_k$ at accuracy δ.

After describing the construction of an Householder TT-vector, which is summarized in Algorithm 16, the focus is on the process that generates an approximately orthonormal TT-vector set from a generic TT-vector set $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$, as depicted in Algorithm 18. To reflect the k-th TT-vector of $\mathcal{A}$ along a linear combination of the first k elements of

Algorithm 16 $\mathbf{u}, r = \texttt{TTH-vec}(\mathbf{a}, \mathcal{E}, \delta)$

1: **input:** $\mathbf{a} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, $\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_i\}$, $\delta \in \mathbb{R}_+$
2: $s = 0$
3: $\mathbf{w} = \mathbf{a}$
4: **for** $j = 1, \ldots, i-1$ **do**
▷ compute the component of $\mathbf{a}$ along the j-th canonical basis TT-vector
5: $r(j) = \langle \mathbf{a}, \mathbf{e}_j \rangle$
6: $s = s + (r(j))^2$
▷ set to zero the component of $\mathbf{a}$ along the j-th canonical basis TT-vector $\mathbf{e}_j$
7: $\mathbf{w} = \mathbf{w} - r(j)\mathbf{e}_j$
8: **end for**
9: $\mathbf{w} = \texttt{TT-round}(\mathbf{w}, \delta)$
▷ subtract from the norm of $\mathbf{a}$ the contribution of the components set to zero
10: $r(i) = \text{sign}(\langle \mathbf{a}, \mathbf{e}_i \rangle)\sqrt{||\mathbf{a}||^2 - s}$
11: $\mathbf{w} = \mathbf{w} - r(i)\mathbf{e}_i$
12: $\mathbf{w} = \texttt{TT-round}(\mathbf{w}, \delta)$
13: $\mathbf{u} = (1/||\mathbf{z}||)\mathbf{w}$
14: **return:** $\mathbf{u}$, r

$\mathcal{E}$, we generate with Algorithm 16 the k-th Householder TT-vector $\mathbf{u}_k$, which defines implicitly the Householder transformation $\mathbf{H}_k$. We store the first k components of r_k in the k-th column of the upper triangular matrix $R \in \mathbb{R}^{m \times m}$. Then $\mathbf{H}_k$ is implicitly applied using $\mathbf{u}_k$ to form $\tilde{\mathbf{a}}_j$ for every $j \in \{k, \ldots, m\}$, as expressed in line 7 of Algorithm 18, following the same approach as in matrix case, i.e.,

$$\mathbf{H}_k(\tilde{\mathbf{a}}_j) = \tilde{\mathbf{a}}_j - 2\langle \tilde{\mathbf{a}}_j, \mathbf{u}_k \rangle \mathbf{u}_k.$$

Algorithm 17 applies a given Householder reflection to a specific input vector. Remark

Algorithm 17 $\mathbf{b} = \texttt{apply-H-vec}(\mathbf{a}, \mathbf{u})$

1: **input:** $\mathbf{a}, \mathbf{u} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$
2: $\mathbf{b} = \mathbf{a} - 2\langle \mathbf{a}, \mathbf{u} \rangle \mathbf{u}$ ▷ apply the Householder reflection defined by $\mathbf{u}$ to $\mathbf{a}$
3: **return:** $\mathbf{b}$

that $(k-1)$ transformations are performed on $\mathbf{a}_k$, computing $\tilde{\mathbf{a}}_k$, before generating the k-th reflector, leading the maximum TT-rank of $\mathbf{a}_k$ to be potentially much larger than its initial value. Therefore to keep the TT-rank of $\tilde{\mathbf{a}}_k$ reasonably small, we perform a TT-rounding before generating the associated Householder TT-vector, see line 10.

The final part of the TT-Householder transformation algorithm 18 generates the new set $\mathcal{Q}$ of orthonormal TT-vectors. Let $\mathbf{q}_i$ be the i-th element of $\mathcal{Q}$ obtained applying the first i Householder reflections in reverse order to $\mathbf{e}_i$ from the canonical basis $\mathcal{E}$, see line 16 of Algorithm 18. To keep the maximum TT-rank of $\mathbf{q}_i$ limited and to avoid running out of memory, each $\mathbf{q}_i$ is rounded at accuracy δ, as stated in line 18 of Algorithm 18.

Algorithm 18 $\mathcal{Q}, R = \text{TT-Householder}(\mathcal{A}, \delta)$

input: $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$, $\delta \in \mathbb{R}_+$
let $\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$ be the canonical basis of a subspace of dimension m of $\mathbb{R}^{n_1 \times \cdots \times n_d}$
$\mathbf{w} = \mathbf{a}_1$
for $i = 1, \ldots, m$ **do**
  ▷ construct the i-th Householder TT-vector
    $\mathbf{u}_i$, $R(:i, i) = \texttt{TTH-vec}(\mathbf{w}, \mathcal{F}_i, \delta)$ with $\mathcal{F}_i = \{\mathbf{e}_1, \ldots, \mathbf{e}_i\}$
    for $j = i, \ldots, m$ **do**
      $\mathbf{a}_j = \texttt{apply-H-vec}(\mathbf{a}_j, \mathbf{u}_i)$ ▷ update the j-th element of $\mathcal{A}$
    end for
    if i < m **then**
  ▷ round the TT-vector that will define the successive Householder reflection
      $\mathbf{w} = \texttt{TT-round}(\mathbf{a}_{i+1}, \delta)$
    end if
end for
for $i = 1, \ldots, m$ **do** ▷ compute the new orthogonal basis
    $\mathbf{q}_i = \mathbf{e}_i$
    for $j = i, \ldots, 1$ **do**
      $\mathbf{q}_i = \texttt{apply-H-vec}(\mathbf{q}_i, \mathbf{u}_j)$ for ▷ reflect the i-th element of $\mathcal{E}$
    end for
    $\mathbf{q}_i = \texttt{TT-round}(\mathbf{q}_i, \delta)$
end for
return: $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$, R

From the complexity viewpoint, the TT-Householder algorithm costs $4m$ TT-rounding operations: two for each Householder TT-vector, one for each TT-vector $\tilde{\mathbf{a}}_k$ after the $(k-1)$-th reflection and one for each $\mathbf{q}_k$ orthogonal TT-vector. Consequently, the TT-Householder algorithm is more expensive than all the other orthogonalization methods, that is 4 times more expensive than CGS and MGS, twice more than CGS2 and MGS2.

3.2.4 Stability comparison

A central issue for the orthogonalization algorithms is the loss of orthogonality, i.e., how much the rounding errors propagate and affect the orthogonality of the computed basis. The loss of orthogonality of an orthogonalization scheme applied to the set $\mathcal{A}_m = \{a_1, \ldots, a_m\}$ is defined by the L2-norm of the difference among the identity matrix of size m and the Gram matrix defined from the m vectors generated by the orthogonalisation algorithm. We state more formally the definition. Let $\mathcal{Q}_m = \{q_m, \ldots, q_m\}$ be a set of m vectors, obtained from an orthogonalisation scheme applied to the m vectors of the input set $\mathcal{A}_m$. Let $q_i \in \mathcal{Q}_m$ be the i-th column of the matrix $Q_m \in \mathbb{R}^{n \times m}$ for every $i \in \{1, \ldots, m\}$, then the Gram matrix associated with the set $\mathcal{Q}_m$ is $Q_m^\top Q_m$. Remark that the (i, j) element of $Q_m^\top Q_m$ is the scalar product of q_i and q_j, that is $Q_m^\top Q_m(i, j) = \langle q_i, q_j \rangle$ for every $i, j \in \{1, \ldots, m\}$. Then, the loss of orthogonality of the considered algorithm

for a basis of size m is equal to

$$||\mathbb{I}_m - Q_m^\top Q_m||_2. \tag{3.7}$$

In the classical matrix framework, an orthogonalization scheme is said *numerically stable* if the loss of orthogonality of the basis it computes is of the order of the unit round-off u of the working arithmetic. The following theoretical results holding true for the six orthogonalization schemes in classical linear algebra constitute a base line for the comparison with the numerical results obtained in the tensor framework, discussed in Section 3.2.5.

In [51, Theorem 1] the authors prove that given $\mathcal{A}_m = \{a_1, \dots, a_m\}$ a set of m vectors, the loss of orthogonality for a basis obtained by CGS is bounded by a positive constant times the unit round-off u, Definition I.II.iii, times the squared condition number of the matrix $A_m \in \mathbb{R}^{n\times m}$ whose j-th column is $a_j \in \mathbb{R}^n$ for $j \in \{1, \dots, m\}$, i.e.,

$$||\mathbb{I}_m - Q_m^\top Q_m||_2 \sim \mathcal{O}\Big(u\kappa^2(A_m)\Big) \tag{3.8}$$

as long as $\kappa^2(A_m)u \ll 1$. In [15], an upper bound for the loss of orthogonality of MGS is provided . Indeed the loss of orthogonality for a basis of m vectors produced by MGS from $\{a_1, \dots, a_m\}$ is upper bounded by a constant times the unit round-off u times the condition number of A_m as previously defined from $\mathcal{A}_m$ elements, i.e.,

$$||\mathbb{I}_m - Q_m^\top Q_m||_2 \sim \mathcal{O}\Big(u\kappa(A_m)\Big) \tag{3.9}$$

as long as $\kappa(A_m)u \ll 1$. The authors of [131, Theorem 4.1] who proposed the Gram orthogonalization scheme, estimated also an upper bound for the loss of orthogonality of their orthogonalization technique. More precisely, the loss of orthogonality of a basis of m vectors produced by the Gram scheme from $\{a_1, \dots, a_m\}$ satisfies the same upper bound as CGS, given in (3.8). Lastly the Householder orthogonalization algorithm is known for being the most stable one. Indeed the loss of orthogonality of a basis of m vectors produced by Householder transformations from $\{a_1, \dots, a_m\}$ is bounded by a constant times the round-off unit, i.e.,

$$||\mathbb{I}_m - Q_m^\top Q_m||_2 \sim \mathcal{O}\Big(u\Big) \tag{3.10}$$

as proved in [145]. When a further orthogonalization step is introduced in the classical and modified Gram-Schmidt, defining CGS2 and MGS2, their loss of orthogonality improves considerably, reaching Householder quality. Indeed, as proved in [51, 129], the loss of orthogonality of CGS2 and MGS2 satisfies the bound given in Equation (3.10), under the hypothesis $\kappa^2(A_m)u \ll 1$ for CGS2, while it holds for MGS2 if $\kappa(A_m)u \ll 1$. A summary of all the loss of orthogonality bounds is presented in Table 3.1.

3.2.5 Numerical tensor experiments

In Sections 3.2.1 - 3.2.3, we describe four orthogonalization methods which produce an orthonormal basis of TT-vectors, given a set of TT-vectors and a rounding accuracy

δ. This section analyses two sets of results obtained from the orthogonalization schemes, highlighting similarities and differences with the known theoretical results in matrix computation. In all the experiments, the input set of TT-vectors $\mathcal{A}_m = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$ is generated with a Krylov process. Let $\mathbf{x}_1 \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be the TT-vector of ones, iteratively we compute $\mathbf{x}_{j+1} = -\mathbf{\Delta}_d \mathbf{a}_j$ where $\mathbf{\Delta}_d$ is the TT-matrix representing the discretization of the Laplacian operator of order d with Dirichlet boundary conditions, Equation (2.37), and $\mathbf{a}_j$ is the normalized output of the `TT-rounding` algorithm applied to $\mathbf{x}_j$. Consequently, $\mathbf{a}_j$ the j-th element of $\mathcal{A}_m$ has TT-rank 1 for every $j \in \{1, \dots, m\}$. This TT-rank constraint facilitates the analysis of the memory requirement. By construction, the elements of $\mathcal{A}$ are generated as a sequence of m normalized (and rounded) Krylov TT-vectors, so that when we vectorize and arrange only the first k as columns of the matrix A_k, its condition number $\kappa(A_k)$ grows for $k \in \{1, \dots, m\}$. Henceforth $\mathcal{A}_k$ denotes the subset of $\mathcal{A}_m$ defined by its first k TT-vectors. The two experiments differ in the dimension of the problem, 3 for the first one and 6 for the second one, but they have the same mode size $n = 15$. We provide further details in the following sections.

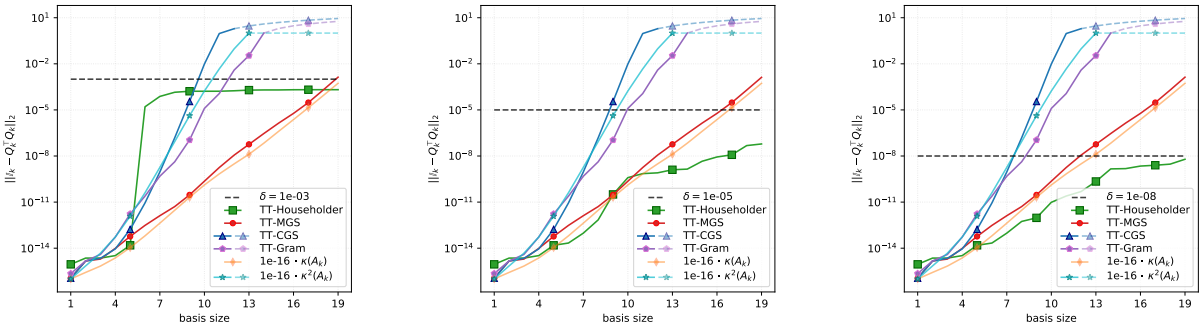
3.2.5.1 Numerical loss of orthogonality

In this section, we examine the numerical results from two set of experiments from the viewpoint of the loss of orthogonality. The purpose is highlighting the similarities with the classical matrix orthogonalization methods. In particular, we investigate the loss of orthogonality $\left\| \mathbb{I}_k - Q_k^\top Q_k \right\|_2$, where the (i, j) element of $Q_k^\top Q_k$ is computed by the scalar product of the i-th and j-th TT-vector of the orthogonal basis, i.e.,

$$\left(Q_k^\top Q_k\right)(i, j) = \langle \mathbf{q}_i, \mathbf{q}_j \rangle$$

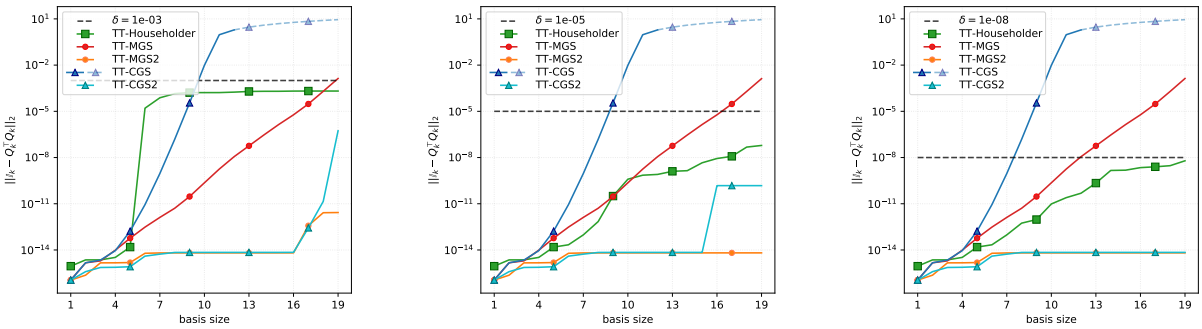
for $\mathbf{q}_i, \mathbf{q}_j \in \mathcal{Q}$ for $i, j \in \{1, \dots, k\}$ for $k \in \{1, \dots, m\}$, where $\mathcal{Q}$ is the set of TT-vectors produced by the considered orthogonalization kernel.

For the first experiment, we set the order $d = 3$, the size mode $n_i = 15$ for $i \in \{1, 2, 3\}$ and the input number of TT-vectors $m = 20$. The results of this first group of experiments for the six different schemes and three different rounding accuracy values $\delta \in \{10^{-3}, 10^{-5}, 10^{-8}\}$ are reported in Figure 3.1. The choice of a low dimensional problem enables us to convert in dense format and vectorize each TT-vector $\mathbf{a}_j$, storing it as the j-th column of $A_m \in \mathbb{R}^{n^3 \times m}$. As consequence, we can estimate the condition number of $A_k \in \mathbb{R}^{n^3 \times k}$, i.e., the submatrix of A_m formed by its first k columns. In all the plots of Figure 3.1, together with the loss of orthogonality with continuous coloured curves, we display the constant rounding accuracy δ with a dashed black line, with coloured continuous lines the condition number $\kappa(A_k)$ and its squared value scaled by $u \approx 10^{-16}$, as long as they are smaller than 1. We scale the condition number curves for the ease of comparison with the slope of the loss of orthogonality of TT-MGS and TT-CGS. All the curves are in function of the basis size k. Indeed, as previously stated, since the TT-vectors are generated as a sequence of normalized Krylov TT-vectors, for increasing values of k, the elements of $\mathcal{A}_k$ get more linearly dependent, leading the associated condition number $\kappa(A_k)$ to increase. For the interpretation ease, we present in the first



(A) $\delta = 10^{-3}$ (B) $\delta = 10^{-5}$ (C) $\delta = 10^{-8}$

Loss of orthogonality with classical algorithms



(D) $\delta = 10^{-3}$ (E) $\delta = 10^{-5}$ (F) $\delta = 10^{-8}$

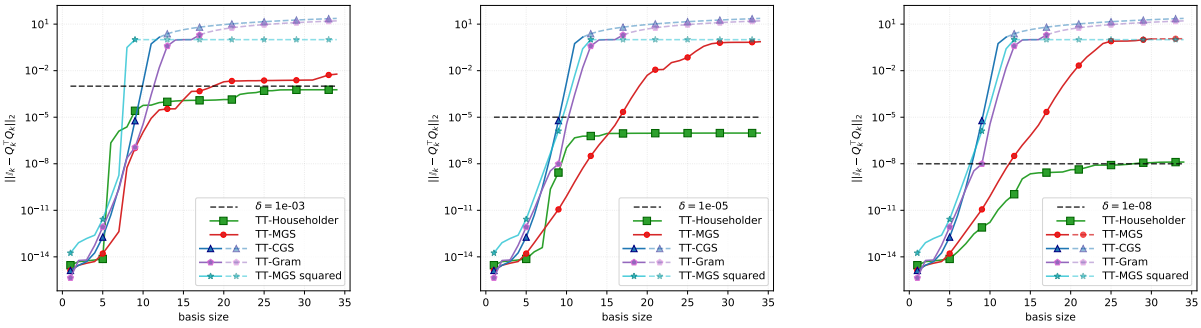
Loss of orthogonality with reorthogonalization comparison

Figure 3.1 – LOO and condition number for $m = 20$ TT-vectors of order $d = 3$ and mode size $n = 15$. The curves get dashed and partially transparent when they get greater than 1.

line three plots, that are Figure 3.1A, 3.1B and 3.1C, with the loss of orthogonality of standard methods in tensor format, i.e., TT-Householder, TT-MGS, TT-CGS, TT-Gram; on the second line, Figure 3.1D, 3.1E and 3.1F display the loss of orthogonality of the methods with re-orthogonalization, that are TT-MGS2 and TT-CGS2, compared with the result of TT-Householder, TT-MGS and TT-CGS. In all the six plots of Figure 3.1, we observe similar behaviours. The TT-Householder loss of orthogonality in green stagnates around the rounding accuracy δ; this phenomenon is extremely clear in Figure 3.1A. This is consistent with the matrix theoretical expectation, stated in Equation (3.10), where the unit round-off u is replaced by the TT-rounding accuracy δ. The TT-MGS loss of orthogonality in red grows with the same slope of the condition number $\kappa(A_k)$ in dashed green, matching the matrix upper bound stated in (3.9). Finally both the TT-CGS and and TT-Gram loss of orthogonality curves cross the rounding accuracy dashed line faster than TT-MGS, following the behaviour of the squared condition number $\kappa^2(A_k)$,

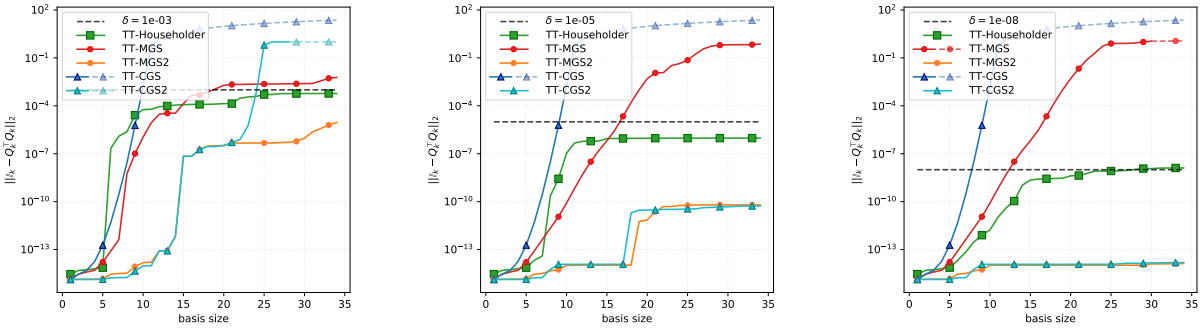
as long as $\kappa^2(A_k) < 10^2$. Indeed TT-CGS and TT-Gram loss of orthogonality curves stagnates below 10^2 for $k > 10$ approximately, while $\kappa^2(A_k)$ continues to grow. Looking at the second line plots, Figure 3.1D, 3.1E and 3.1F show that introducing a second re-orthogonalization loop improves considerably the loss of orthogonality. Indeed as long as $k < 15$ the loss of orthogonality of TT-CGS2 is close to the machine precision, around 10^{-14} for $\delta \in \{10^{-3}, 10^{-5}\}$, increasing after; for $\delta = 10^{-8}$ it remains around 10^{-14}. Consequently as long as the element of $\mathcal{A}_k$ are not too much collinear, TT-CGS2 outcompetes TT-CGS, TT-MGS and TT-Householder, but not TT-MGS2. For the rounding accuracy $\delta \in \{10^{-5}, 10^{-8}\}$, the loss of orthogonalization of TT-MGS2 stagnates around 10^{-14}, while for $\delta = 10^{-3}$ the loss of orthogonality jumps from 10^{-14} to 10^{-11}, where it seems to stagnates, when $k > 16$. Overall TT-MGS2 outperforms all the other algorithms. These results found for TT-CGS2 and TT-MGS2 are consistent with the matrix theory presented in Section 3.2.4. Moreover, we may hypothesise that the jumps arrive when the condition number $\kappa(A_k)$, or its square, times the rounding accuracy is not any more sufficiently smaller than 1.

To further validate our results and to study the applicability to large-scale problems, we consider a second experimental framework. Set the problem order to $d = 6$ with size mode $n_i = 15$ for $i \in \{1, \ldots, 6\}$, we generate $m = 35$ TT-vectors, defining the set $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_{35}\}$. For the rounding accuracy values $\delta \in \{10^{-3}, 10^{-5}, 10^{-8}\}$, we compute the loss of orthogonality for the four orthogonalization schemes, presented in Section 3.2.1 - 3.2.3. The loss of orthogonality of these experiments is displayed in Figure 3.2. Because of the problem order $d = 6$ and size $n = 15$, in this case we do not add the curve of the condition number of the $A_k \in \mathbb{R}^{n^6 \times k}$, whose k-th column would be the TT-vector $\mathbf{a}_k \in \mathcal{A}_m$ uncompressed and vectorized. To compensate the absence of the condition number curve, we display the square of the TT-MGS loss of orthogonality values with a dashed line, that should exhibit the same slop as the CGS loss of orthogonality (and consequently of the squared condition number $\kappa(A_k)$), if the matrix theory extends to the TT-framework. As in the previous case, the first line plots display standard orthogonalization algorithms in TT-format, while on the second line we display results from methods with re-orthogonalization. Figures 3.2A, 3.2B and 3.2C clearly show that the TT-Householder orthogonalization algorithm produces a basis whose loss of orthogonality stagnates around the rounding accuracy δ for every value in $\{10^{-3}, 10^{-5}, 10^{-8}\}$ after the basis size gets greater than approximately 10. This seems to support further our intuition that even in the tensor framework the bound expressed in Equation (3.10) is still holding true, with the unit round-off u replaced by the TT-rounding accuracy δ. In Figure 3.2A and 3.2B, when the basis size is smaller than about 15, the TT-MGS loss of orthogonality is smaller than the Householder one, but when the basis includes more then 15 TT-vectors, the relation reverses. For more accurate computation, that is for $\delta = 10^{-8}$, already when the basis size is around 5, the Householder loss of orthogonality outcompetes the TT-MGS one. Notice that for all the rounding accuracy values, the TT-MGS loss of orthogonality firstly grows linearly and stagnates after a while at different level, at 10^{-2} for $\delta = 10^{-3}$, at 1 for $\delta \in \{10^{-5}, 10^{-8}\}$. The stagnation of the TT-CGS and TT-Gram loss of orthogonality curves arrives almost immediately, when the basis size is greater than 10 for all



(A) $\delta = 10^{-3}$ (B) $\delta = 10^{-5}$ (C) $\delta = 10^{-8}$

Loss of orthogonality with classical algorithms



(D) $\delta = 10^{-3}$ (E) $\delta = 10^{-5}$ (F) $\delta = 10^{-8}$

Loss of orthogonality with reorthogonalization comparison

Figure 3.2 – LOO for $m = 35$ TT-vectors of order $d = 6$ and mode size $n = 15$. The curves get dashed and partially transparent when they get greater than 1.

the rounding accuracy values, i.e., in all the plots of Figure 3.2. This is probably due to the very bad condition number of the basis, so that the assumptions of the matrix theory are not longer valid, that is, condition number times rounding smaller than 1. Moreover mainly Figures 3.2B and 3.2C show that the TT-CGS and TT-Gram loss of orthogonality follow the square of the TT-MGS loss of orthogonality, supporting the idea that even in the TT-format, TT-MGS loss of orthogonality grows as the condition number, while TT-Gram and TT-CGS loss of orthogonality as the squared condition number. Figures 3.2D, 3.2E and 3.2F display the loss of orthogonality of TT-MGS2 and TT-CGS2, compared with the previously analysed results of TT-Householder, TT-MGS and TT-CGS. For all the considered rounding accuracies, TT-MGS2 outperforms all the other methods, with a loss of orthogonality which stagnates around 10^{-5} for $\delta = 10^{-3}$, around 10^{-10} for $\delta = 10^{-5}$ and around 10^{-13} for $\delta = 10^{-8}$. In all the three Figures 3.2D- 3.2F, the TT-MGS2 curve presents a jump larger for greater values of δ, for $15 \leq k \leq 20$ with $\delta \in \{10^{-3}, 10^{-5}\}$ and

for $k \sim 10$ with $\delta = 10^{-8}$. Also TT-CGS2 outcompetes the TT-Householder, TT-CGS and TT-MGS, following the behaviour of TT-MGS2, always for $\delta \in \{10^{-5}, 10^{-8}\}$ and as long as the TT-vectors are not too much collinear, i.e., for $k < 20$ when $\delta = 10^{-3}$. These results support the conclusion stated for the $d = 3$ experiments, that the bounds for the loss of orthogonality of TT-CGS2 and TT-MGS2 proved in the classical matrix framework still apply, probably under revised hypothesis.

3.2.5.2 Memory usage estimation

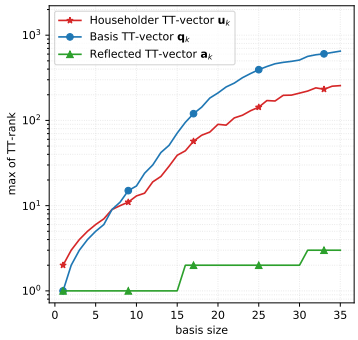
This section aims to study the effect on the TT-rank and consequently on the memory requirement of the orthogonalization process from the numerical results. We investigate the growth of the TT-ranks, of the compression ratio, defined in Equation (1.7), and the compression gain curve, stated in Equation (1.8). We examine the TT-rank, the compression ratio and the compression gain of the new basis produced by the orthogonalization algorithms in the second set of experiments with $d = 6$, $n_i = 15$, $m = 35$ and $\delta \in \{10^{-3}, 10^{-5}, 10^{-8}\}$, since this problem can already be considered a large scale one.

In the Householder algorithm 18 there are three sets of TT-vectors on which the `TT-rounding` is applied: the Householder TT-vector $\mathbf{u}_k$, the TT-vector $\mathbf{a}_k$ to which we apply k Householder transformations and the orthogonal TT-vector $\mathbf{q}_k$ obtained from the canonical basis TT-vectors with i successive Householder transformations. It is worthwhile studying the evolution of the maximum TT-rank, of the compression ratio and gain for each of these three TT-vector groups. In Figure 3.3 we display the maximum TT-rank, the compression ratio and the compression gain of $\mathbf{u}_k$, $\mathbf{a}_k$ after the k-th reflection and $\mathbf{q}_k$ for every $k \in \{1, \ldots, 35\}$ for all the values of the rounding accuracy δ. As expected, the maximum TT-rank and the compression ratio of $\mathbf{u}_k$, $\mathbf{a}_k$ and $\mathbf{q}_k$ grow for increasing basis sizes, because the number of terms in their computation grows with k. Remark that the maximum TT-rank of $\mathbf{q}_k$ becomes larger than one of $\mathbf{u}_k$ for a basis size greater than 10. This property is significant since in the majority of the practical vector computations only the Householder TT-vectors $\mathbf{u}_k$ are stored and the orthogonal basis TT-vector $\mathbf{q}_k$ is usually not explicitly formed.

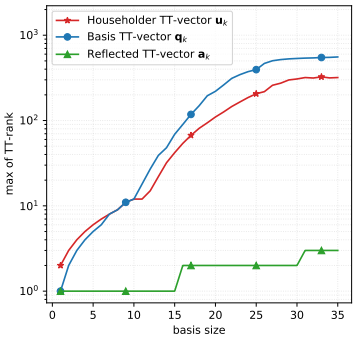
In Figures 3.3A, 3.3B and 3.3C we observe that the maximum TT-rank of $\mathbf{a}_k$ after the k-th Householder reflection is extremely low, mainly if it is compared with those of $\mathbf{q}_k$ and $\mathbf{u}_k$. Moreover, the rise of 1 of the maximum TT-rank of $\mathbf{a}_k$ arrives every time the index k is equal to a multiple of the mode size $n = 15$, as we can see in Figure 3.3A and 3.3B. This behaviour is present also for $\delta = 10^{-8}$ for Figure 3.3C, but it is less regular. We tried to investigate further this phenomenon from a theoretical viewpoint, but we did not arrive to a clear and convincing explanation.

The compression ratio has closely the same shape as maximum TT-rank, but it enables us to observe the memory growth in percentage. Figures 3.3D, 3.3E and 3.3F present for all the values of the rounding accuracy δ a compression ratio smaller than 1, implying that for all these experiments the TT-format is still relevant to reduce the memory footprint. Moreover the compression ratio of $\mathbf{a}_k$ stagnates around 10^{-5} for all the three values of δ. Figures 3.3D and 3.3E show that the compression ratio of $\mathbf{q}_k$ stagnates around 10^{-1},

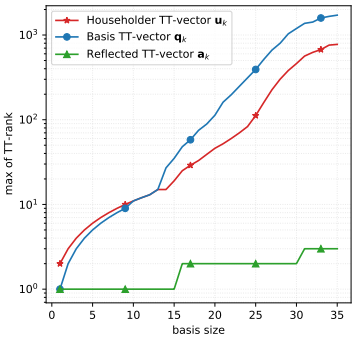
while the one of $\mathbf{u}_k$ around 10^{-2}. In Figure 3.3F, it is not clear if the compression ratio of $\mathbf{q}_k$ and $\mathbf{u}_k$ stagnate around 1 and 10^{-1} respectively. All the Figures 3.3G, 3.3I and 3.3H, the gain curves of $\mathbf{u}_k$, $\mathbf{q}_k$ and $\mathbf{a}_k$ present approximately the same behaviour. In particular the gain of compressing the k-th Householder TT-vector $\mathbf{u}_k$ and the input TT-vector $\mathbf{a}_k$ after the k-th reflection is almost constant through the iterations and minimal, but both the TT-vectors are compressed several times during the previous iterations. On the other side, the compression gain for storing the Householder basis TT-vector $\mathbf{q}_k$ is significantly larger. For all the rounding accuracies, the compression gain curve of $\mathbf{q}_k$ increases during the first 10 iterations, decreases slightly after and seems to grow again during the last 10 iterations. Remark that for the compression gain curve of the Householder basis vector the highest value is around 110 if $\delta = 10^{-5}$. This means that after the compression, it is actually requested slightly less than 1% of the memory used to store the same tensor in TT-format before the compression.



(A) $\delta = 10^{-3}$

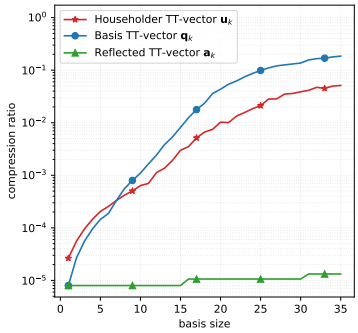


(B) $\delta = 10^{-5}$

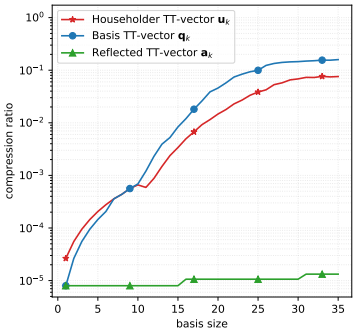


(C) $\delta = 10^{-8}$

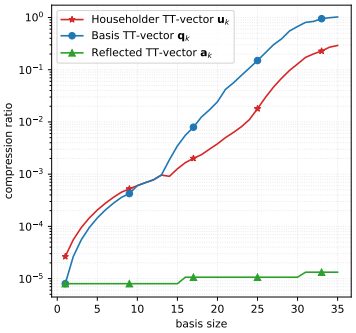
Maximal TT-rank for the TT-vectors in Householder algorithm



(D) $\delta = 10^{-3}$

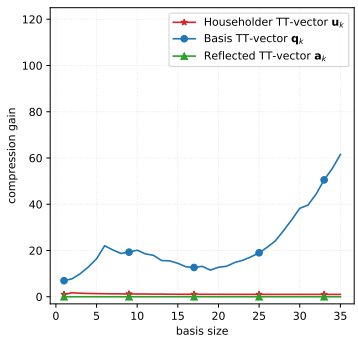


(E) $\delta = 10^{-5}$

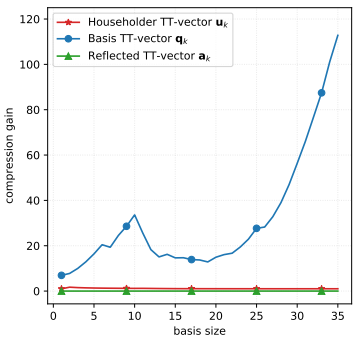


(F) $\delta = 10^{-8}$

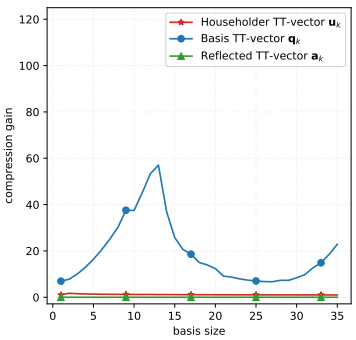
Compression ratio for the TT-vectors in Householder algorithm



(G) $\delta = 10^{-3}$



(H) $\delta = 10^{-5}$

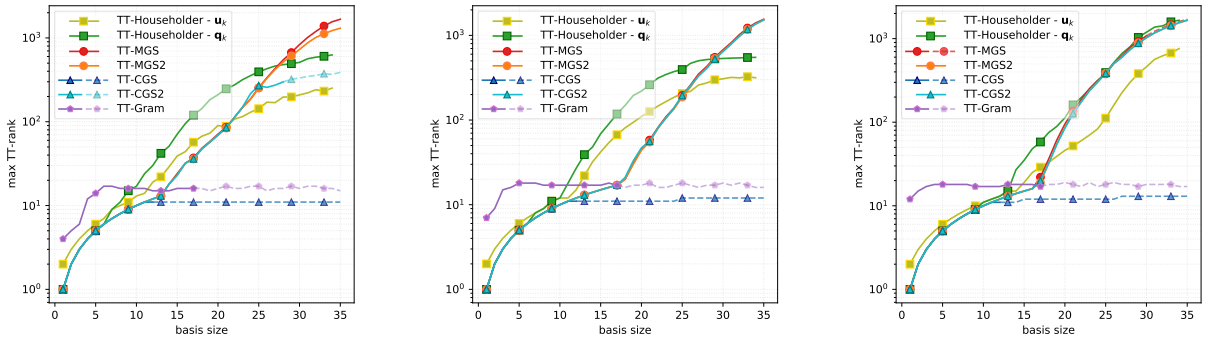


(I) $\delta = 10^{-8}$

Compression gain for the TT-vectors in Householder algorithm

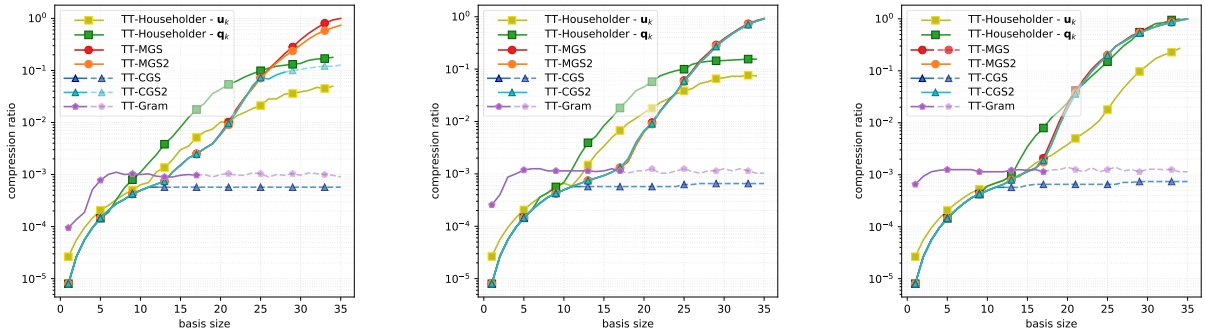
Figure 3.3 – Householder memory requirement for $m = 35$ TT-vectors of order $d = 6$ and mode size $n = 15$.

After describing the memory need of the TT-Householder algorithm, we compare the four orthogonalization schemes from the memory requirement viewpoint. Indeed, to fairly compare the orthogonalization schemes, we have to consider the memory consumptions together with the loss of orthogonality, studied in Section 3.2.5.1. In Figure 3.4 we display, for different accuracies δ, the maximum TT-rank, the compression ratio and the gain for the orthogonal TT-vectors $\mathbf{q}_k$ generated by the orthogonalization schemes plus the Householder TT-vector $\mathbf{u}_k$ ones, since, as already mentioned, in many applications $\mathbf{q}_k$ are not computed. Remark that for every rounding accuracy δ, in the corresponding figure the curves get dashed and partially transparent when the corresponding loss of orthogonality become larger than 10^{-1}. In all Figures 3.4A, 3.4B and 3.4C we see that the maximum TT-rank of the orthogonal TT-vectors computed by TT-CGS and TT-Gram schemes stagnates around 10, but we do not have a clear theoretical justification for this phenomenon. The maximal TT-rank of $\mathbf{q}_k$ from TT-Gram algorithm is theoretically bounded by k times the maximal TT-rank of $\mathbf{a}_j$ for $j \in \{1, \ldots, k\}$. When $\mathbf{a}_j$ are generated, they are rounded by maximal TT-rank equal to 1, then the maximal TT-rank of $\mathbf{q}_k$ is bounded by k in our experimental framework. On the other side, the maximal TT-rank of $\mathbf{q}_k$ from TT-CGS is bounded by $1 + k(k-1)/2$ knowing that the maximal TT-rank of $\mathbf{a}_i$ is bounded by 1 for $i \in \{1, \ldots, m\}$ in our experiments. From the viewpoint of the maximal TT-rank, TT-Gram outcompetes TT-MGS and TT-Householder for basis size greater than 10, while TT-CGS establishes a lower bound in terms of maximal TT-rank. However the loss of orthogonality of TT-CGS and TT-Gram gets greater than 10^{-1} around $k = 10$. On the other side, the TT-MGS maximal TT-rank curve gets dashed around $k = 20$, while the Householder one never, that is the loss of orthogonality arrives much after for TT-MGS or even does not arrive for TT-Householder. In Figure 3.4A, the maximal TT-rank of $\mathbf{q}_k$ from TT-MGS overcomes the maximal TT-rank of the Householder TT-vector $\mathbf{u}_k$ and of the Householder orthogonal TT-vector $\mathbf{q}_k$ when the basis size k reaches 20 and 25 respectively. A similar relation among the maximal TT-rank for Householder and for MGS generated orthogonal TT-vectors appears in Figure 3.4B, but the turning point is set at a different basis size, that is 25 and 30 for the Householder TT-vector $\mathbf{u}_k$ and $\mathbf{q}_k$ respectively. For the last rounding accuracy value $\delta = 10^{-8}$, whose related results are displayed in Figure 3.4C, the maximal TT-rank of the MGS orthogonal TT-vector reaches the Householder $\mathbf{q}_k$ when the basis size gets larger than 15, overcoming the maximal TT-rank of the Householder $\mathbf{u}_k$ approximately at the same basis size.



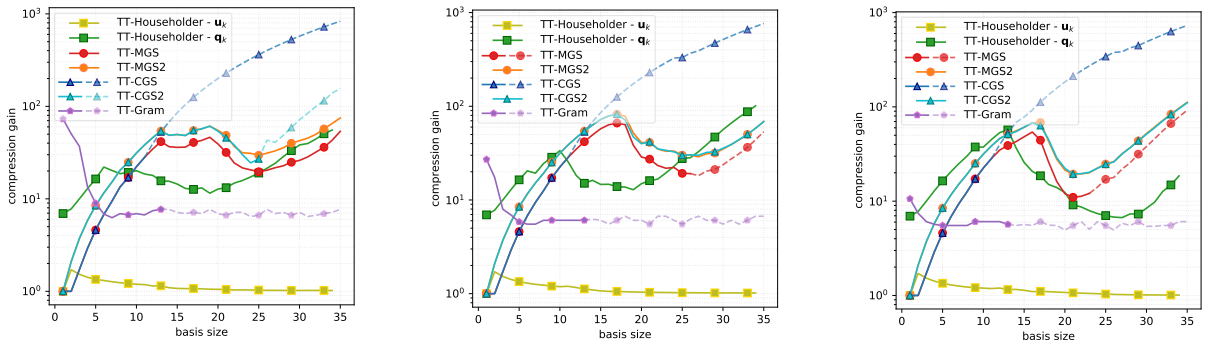
(A) $\delta = 10^{-3}$ (B) $\delta = 10^{-5}$ (C) $\delta = 10^{-8}$

Maximal TT-rank for the orthogonal basis



(D) $\delta = 10^{-3}$ (E) $\delta = 10^{-5}$ (F) $\delta = 10^{-8}$

Compression ratio for the orthogonal basis



(G) $\delta = 10^{-3}$ (H) $\delta = 10^{-5}$ (I) $\delta = 10^{-8}$

Compression gain for the orthogonal basis

Figure 3.4 – Comparison of the orthogonal basis memory requirement for $m = 35$ TT-vectors of order $d = 6$ and mode size $n = 15$. The curves get dashed and partially transparent when their corresponding loss of orthogonality gets greater then the prescribed rounding accuracy δ.

As in analysis of TT-Householder algorithm memory requirement, we investigate also the compression ratio for the TT-vectors forming the orthogonal basis produced by the four orthogonalization methods. The compression ratio curves of Figures 3.4D, 3.4E and 3.4F have the same slopes of their corresponding maximal TT-rank curves, but they display clearly the memory needs. The orthogonal TT-vectors from TT-CGS and TT-Gram schemes demand approximately 1% of the total memory necessary to store the full format tensors, as illustrated in Figure 3.4D, 3.4E and 3.4F. The same figures highlight that when the basis size gets greater than 10 and the TT-vectors start to be more collinear, the curves gets dashed and these scheme resulting basis are very poor in terms of orthogonality. From both Figures 3.4D and 3.4E, we infer that storing the basis TT-vector generated by TT-Householder scheme request approximately 20% of the memory that we would need to store those tensors in full format. Similarly only 10% of the entire memory requested for full format storage is necessary to store the Householder TT-vector $\mathbf{u}_k$. Finally, for $\delta = 10^{-8}$ storing cost for the Householder basis TT-vector $\mathbf{q}_k$ reaches the same amount of memory demanded for storing them in full format, as depicted in Figure 3.4F. However from the same figure, we notice that storing the Householder TT-vectors even for $\delta = 10^{-8}$ needs approximately 30% of the memory necessary to store the same tensor in full format. This property makes the TT-Householder algorithm extremely interesting, since usually it is sufficient to store the Householder TT-vectors. TT-Householder algorithm gets even more convenient when we compare its compression ratio curves with the TT-MGS, TT-MGS2 and TT-CGS2 ones. Indeed, for all the rounding accuracy values, the compression ratio curve of TT-MGS and TT-MGS2 always reaches 1, see Figures 3.4D, 3.4E and 3.4F, implying that the memory requested by the TT-vectors from these schemes reaches the same amount of memory we would use to store in full format the orthogonal basis tensors. This same consideration holds true for the compression ratio of the orthogonal basis produced by TT-CGS2, for $\delta \in \{10^{-5}, 10^{-8}\}$, while for $\delta = 10^{-3}$, the TT-vectors from TT-CGS2 consume just 20% of the memory we would need to store the same tensors in dense format. The compression gain curves in Figures 3.4G, 3.4H and 3.4I present approximately the same behaviour for the different rounding accuracy values δ. The compression gain of TT-Gram has a pick at the very beginning, then it stagnates around 10 starting from $k = 10$, meaning at the compressing the j-th basis TT-vector reduce of 10 times the memory requirement for $j \geq k$. The gain curves of TT-MGS, TT-MGS2 and TT-CGS2 have approximately the same shape, they increase up to $k = 15$, for the successive 5 iterations the gain curves drop down and around $k = 22$, they rise again. Also the TT-Householder basis gain curve has a similar patter, it first grows, reaching a pick, then it decreases and during the last iterations it rises again. The Householder TT-vector gain curve, as previously observed, has a tiny growth at the beginning, then it drops down and stagnates at a very low value from $k > 10$. Finally the TT-CGS gain curve increases for increasing dimension k of the basis, so when the last basis TT-vector is rounded, it leads to use 0.001 of the memory used to store the TT-vector before rounding it. However as highlighted by the dash style, the TT-CGS and TT-Gram basis have almost entirely lost the orthogonality already at $k = 10$.

3.2.6 Summary

From the viewpoint of the memory footprint, thanks to its stability property and the possibility of storing only the TT-vectors $\mathbf{u}_i$, the TT-Householder orthogonalization scheme appears to be the best in many cases, together with TT-CGS2 and TT-MGS2. Figures 3.4D, 3.4E and 3.4F display that when the input TT-vectors are not too much collinear, $k < 15$, TT-MGS2 and TT-CGS2 have a compression ratio approximately equal to TT-Householder. For a basis size between 15 and 25 (or 20 for $\delta = 10^{-8}$) TT-Householder is more expensive than TT-MGS, TT-MGS2 and TT-CGS2 in memory terms. Finally when the input TT-vectors get more and more linearly dependent, the memory need of TT-MGS2 and TT-CGS2 basis is greater or equal than both the TT-Householder basis and Householder TT-vector. However from the viewpoint of the loss of orthogonality, TT-MGS2 performs better than TT-Householder for every rounding accuracy, while TT-CGS2 for $\delta \in \{10^{-5}, 10^{-8}\}$. Finally from the computational cost side, TT-CGS2 and TT-MGS2 are cheaper than TT-Householder, that is $2m$ TT-rounding versus $4m$.

Similarly to the matrix case, the choice of the orthogonalization scheme among TT-Householder, TT-CGS2 and TT-MGS2 depends strongly on the purpose and on the available computing resources. Indeed TT-Householder requires less memory, but it is computationally more expensive and its orthogonality stagnates around the rounding accuracy. On the other side, TT-MGS2 produces a basis of better orthogonality quality, as long as the input TT-vectors are not too collinear, and it is computationally cheaper than TT-Householder, but it is more expensive from the viewpoint of the memory consumption. The same considerations hold also for TT-CGS2, under the same hypothesis. The computational costs in tensor and matrix case are summarized in Table 3.1.

	Matrix		**TT-vectors**	
Algorithm	*Computational cost in fp operations*	$\left\|\mathbb{I}_k - Q_k^\top Q_k\right\|$	*Computational cost in TT-rounding*	$\left\|\mathbb{I}_k - Q_k^\top Q_k\right\|$
Gram	$\mathcal{O}(2nm^2)$	$\mathcal{O}(u\kappa^2(A_k))$	m	$\mathcal{O}(\delta\kappa^2(A_k))$
CGS	$\mathcal{O}(2nm^2)$	$\mathcal{O}(u\kappa^2(A_k))$	m	$\mathcal{O}(\delta\kappa^2(A_k))$
MGS	$\mathcal{O}(2nm^2)$	$\mathcal{O}(u\kappa(A_k))$	m	$\mathcal{O}(\delta\kappa^2(A_k))$
CGS2	$\mathcal{O}(4nm^2)$	$\mathcal{O}(u)$	$2m$	$\mathcal{O}(\delta)$
MGS2	$\mathcal{O}(4nm^2)$	$\mathcal{O}(u)$	$2m$	$\mathcal{O}(\delta)$
Householder	$\mathcal{O}(2nm^2 - 2m^3/3)$	$\mathcal{O}(u)$	$4m$	$\mathcal{O}(\delta)$

Table 3.1 – Computational costs in floating point operations and in TT-rounding operations, and bounds for the loss of orthogonality, theoretical with respect to the unit round-off u and conjectured ones with respect to the rounding accuracy δ, for an input set of m vectors and TT-vectors respectively.

3.3 Eigensolvers

The orthogonalization schemes presented in Section 3.2 are usually kernels of more elaborated algorithms. The purpose of this section is to illustrate an eigensolver as study case where they are applied and where their properties affect the final result. In Section 3.3.1 we describe the chosen eigensolver algorithm, that is the subspace iteration method, while Section 3.3.2 collects the numerical results.

3.3.1 Subspace iteration method

The subspace iteration method is an iterative eigensolver, that can be regarded as a block generalization of the power iteration method. This eigensolver is considered one of the simplest and most reliable, cf. [133, Chapter 6] or [126, Chapter 5], even though it is usually slow and should be accelerated with preconditioning techniques in practical computations. This section aims to describe the subspace iteration method extended to the tensor framework to solve tensor eigenproblems, i.e., finding eigenpairs $(\lambda, \mathbf{v})$ of a tensor multilinear operator $\mathbf{A} \in \mathbb{R}^{(n_1 \times n_1) \times \cdots \times (n_d \times n_d)}$ such that

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v}$$

with $\lambda \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.

We briefly illustrate the subspace iteration method extended to tensor eigenproblems through the TT-formalism, henceforth referred to as TT-SUBSPace ITeration (TT-SUBSPIT), depicted in Algorithm 19. The TT-SUBSPIT is meant to compute `nev` TT-eigenvectors associated with the largest `nev` eigenvalues of a given TT-matrix. Together with the TT-matrix whose TT-eigenpairs are searched, it takes as input parameters $\mathcal{A}$ a set of $m \geq$ `nev` TT-vectors, a rounding accuracy $\delta \in \mathbb{R}_+$, a convergence threshold $\varepsilon \in \mathbb{R}_+$, a maximum number of iterations `max_it` and an integer $p \in \mathbb{N}$, that defines the power of the TT-operator to use. As first step, TT-SUBSPIT orthogonalizes the m input TT-vectors of $\mathcal{A}$ by an orthogonalization kernel, using the prescribed rounding accuracy δ and returning the orthogonal set $\mathcal{Q}$, see line 5 of Algorithm 19. During the first iteration, we compute the j-th elements of $\mathcal{U}$ as the product of the TT-operator at power p and the j-th orthogonal TT-vector of $\mathcal{Q}$, that is $\mathbf{u}_j = \mathbf{A}^p \mathbf{q}_j$. To avoid memory deficiencies, we round at accuracy δ and then normalize each element of $\mathcal{U}$ in line 14 and 15 of Algorithm 19. Then, the TT-vecotrs of $\mathcal{U}$ are orthogonalized with the previously used orthogonalization kernel at accuracy δ, see line 17, defining $\mathcal{Q}$. Remark that at the very first iteration, the set $\mathcal{V}$ is empty and the set $\mathcal{U}$ as cardinality $n = m$. Then, we project the TT-operator $\mathbf{A}$ into the new orthogonal basis formed by the last n elements of $\mathcal{Q}$, defining matrix $G \in \mathbb{R}^{n \times n}$ with $n = m$ since no TT-eigenpair has converged yet. In line 23 of Algorithm 19, the eigenvalue decomposition of matrix G is computed, sorting the eigenvalues, and the correspondent eigenvector, in decreasing order. The n eigenvalues of G are by construction an approximation of the largest n eigenvalues of $\mathbf{A}$. The eigenvectors of G are used to approximate of the first n TT-eigenvectors of $\mathbf{A}$. Indeed, the TT-vector $\mathbf{w}_j$ is obtained as a linear combination of the last n elements of $\mathcal{Q}$, choosing as coefficients

Algorithm 19 $\mathcal{V}, \mathcal{L} =$ TT-SUBSPIT($\mathbf{A}$, $\mathcal{Z}$, p, δ, ε `max_it`)

input: $\mathbf{A} \in \mathbb{R}^{(n_1 \times n_1) \times \cdots \times (n_d \times n_d)}$, $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ with $\mathbf{z}_h \in \mathbb{R}^{n_1 \times \cdots \times n_d}$
input: $p \in \mathbb{N}$, $\delta, \varepsilon \in \mathbb{R}_+$
$\mathcal{V} = \emptyset$; $\mathcal{L} = \emptyset$
$c = 0$; $n = m$; `i` $= 0$; `iter` $= 0$
$\mathcal{Q}, \cdot =$ orth_scheme($\mathcal{Z}$, δ)
while `iter` $<$ `max_it` and $c < m$ **do**
    $n = m - c$ ▷ update the number of TT-eigenpairs to compute
    ▷ `iter` counts the number of contractions performed, while i the number of subspace iterations
    `iter` $=$ `iter` $+ n(p+1)$, $i = i + 1$
    for $j = 1, \ldots, n$ **do**
        $\mathbf{x} = \mathbf{A}\mathbf{q}_{j+c}$
        for $k = 1, \ldots, p$ **do**
            $\mathbf{x} = \mathbf{A}\mathbf{x}$ ▷ compute $\mathbf{A}^p\mathbf{x}$
        end for
        $\mathbf{y} =$ `TT-round`$(\mathbf{x}, \delta)$
        $\mathbf{u}_j = (1/\|\mathbf{y}\|)\mathbf{y}$ with $\mathbf{u}_j \in \mathcal{U}$
    end for
    ▷ orthogonalize the set formed by the locked TT-eigenvector and the TT-vector just updated
    $\mathcal{Q}, \cdot =$ orth_scheme($\mathcal{V} \cup \mathcal{U}$, δ)
    for $j = 1, \ldots, n$ **do**
        for $k = j, \ldots, n$ **do**
            $G(j,k) = G(k,j) = \langle \mathbf{A}\mathbf{q}_{j+c}, \mathbf{q}_{k+c} \rangle$
        end for
    end for
    ▷ compute the eigenvalue decomposition of the matrix G with $\lambda_h \in \mathbb{R}$ sorted with decreasing value and $E \in \mathbb{R}^{n \times n}$ inheriting the same ordering
    $\{\lambda_1, \ldots, \lambda_n\}, E =$ `eig`(G)
    for $j = 1, \ldots, n$ **do**
        $\mathbf{z} = E(1,j)\mathbf{q}_{1+c}$
        for $k = 2, \ldots, n$ **do**
            $\mathbf{z} = \mathbf{z} + E(k,j)\mathbf{q}_{k+c}$ ▷ construct the j-th TT-eigenvector
        end for
        $\mathbf{w}_j =$ `TT-round`$(\mathbf{z}, \delta)$ with $\mathbf{w}_j \in \mathcal{W}$
    end for
    for $j = 1, \ldots, n$ **do**
        $r = (1/\lambda_j)\|\mathbf{A}\mathbf{w}_j - \lambda_j \mathbf{w}_j\|$ ▷ compute the scaled residual
        if $r < \varepsilon$ **then**
            $\mathbf{v}_{c+1} = \mathbf{w}_j$ belongs to $\mathcal{V}$ and λ_j to $\mathcal{L}$ ▷ lock the TT-eigenpair that converged
            $c = c + 1$ ▷ update the number of TT-eigenpairs that converged
        else
            $\mathbf{q}_{j+c} = \mathbf{w}_j$ ▷ otherwise update the set $\mathcal{Q}$
        end if
    end for
end while
return: $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_c\}$, $\mathcal{L} = \{\lambda_1, \ldots, \lambda_c\}$

the entries of the eigenvectors of G, see line 27 of Algorithm 19. Before adding $\mathbf{w}_j$ to the set $\mathcal{W}$, we apply the TT-rounding algorithm at accuracy δ to prevent memory deficiencies, line 29 of Algorithm 19. Finally for every TT-vector of $\mathcal{W}$ we compute their residual, that is $\|\mathbf{A}\mathbf{w}_j - \lambda_j \mathbf{w}_j\|$, scaled by $|\lambda_j|$, where λ_j is the j-th eigenvalue of G for every $j \in \{1, \ldots, n\}$, see line 32. If the scaled residual is smaller than the convergence threshold ε, then the TT-vector $\mathbf{w}_j$ and its eigenvalue λ_j are locked, becoming the 1st converged TT-eigenvector and eigenvalue. Thus, by locking, we include $\mathbf{w}_j$ in the set $\mathcal{V}$ and λ_j in $\mathcal{L}$, and we increase the counter of the number of converged eigenpairs c by one unit, lines 34 and 35 of Algorithm 19. Otherwise, the $\mathbf{w}_j$ is used to update the $(j+c)$-th element of $\mathcal{Q}$, see line 37 of Algorithm 19. During the following iterations, thanks to the locking strategy, we apply the TT-operator $\mathbf{A}$ at power p only to the last $n = m - c$ elements of $\mathcal{Q}$, so that the set $\mathcal{U}$ will have cardinality n. However, to ensure the orthogonality at line see line 17 of Algorithm 19 we apply the orthogonalizations scheme on the set of TT-vectors, whose first c elements are the TT-eigenvectors locked in $\mathcal{V}$and the last $n = m - c$ are those that have not converged yet in $\mathcal{U}$, see line 17.

3.3.2 Numerical Experiments

In this section, we investigate the numerical results obtained with the TT-SUBSPIT with the six different orthogonalization schemes in TT-format, described in Section 3.2. Two sets of experiments are considered: in the first, we compute the TT-eigenpairs of the discrete Laplacian operator $\mathbf{\Delta}_3$ of order $d = 3$ with mode size $[19, 24, 31]$, in the second, we examine the TT-eigenpairs for the discrete order $d = 3$ Laplacian with all mode sizes equal to 24. This operator is chosen since there exists an analytical expression for their eigenpairs. To compute an approximation of the first $m = 7$ TT-eigenpairs, we use the same value for the rounding accuracy δ and the convergence one ε in $\{10^{-3}, 10^{-5}\}$ for both the experiment sets. As for the experiments of Section 3.2.5, the input set $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ of TT-vectors are generated with a Krylov process, that is $\mathbf{z}_1$ is a TT-vector of ones and the $(h+1)$-th one is obtained from the TT-rounding at TT-rank 1 of $\mathbf{\Delta}_d \mathbf{z}_h$ for $h \in \{1, \ldots, m-1\}$. As mentioned in [126, 133], in the classical matrix case the higher the power value, the faster the convergence, but, at the same time, the more the vectors to orthogonalize get linearly dependent. Thus, to simplify the investigation of these preliminary results and highlight as much as possible the orthogonalization kernel effect, we set $p = 1$.

3.3.2.1 TT-Eigenpairs convergence

This section aims to study the convergence of the TT-SUBSPIT algorithm combined with the different orthogonalization schemes, described in Section 3.2. In particular we want to highlight the effect of the orthogonality stability on the quality of the computed TT-eigenpairs.

In Tables 3.2 and 3.3 we report the $m = 7$ computed eigenvalues $\tilde{\lambda}$ sorted with decreasing value, the closest theoretical eigenvalue λ^*, the relative distance between the two, the

convergence order, i.e., in which order the eigenvalues converged and the *spectrum order*, that is which position λ^* occupies in the spectrum. Notice that the 7 largest theoretical eigenvalues of the discrete Laplacian $\mathbf{\Delta}_3$ with mode sizes $[19, 24, 31]$ are

$$[8166.432,\ 8137.127,\ 8137.018,\ 8136.942,\ 8107.712,\ 8107.637,\ 8107.528],$$

that is there the largest one, followed by two clusters of three eigenvalues. Comparing the different orthogonalization kernel results in Table 3.2, we see that the Gram method leads to the worst approximations, missing the largest eigenvalue and getting close to just one out of 7 eigenvalues in the selected region. On the other hand, Householder algorithm provides the best output approximating six eigenvalues out of seven in the expected spectrum region; the other kernels find just two out of seven. Basically, TT-SUBSPIT with Householder approaches the largest one, three from the first cluster and two from the second one. With the other kernels, excluded Gram's one, TT-SUBSPIT leads to an estimate of the largest one and one from the first cluster, while the other estimated eigenvalues are closer to theoretical eigenvalues localized much far in the spectrum. From the convergence order view point, it is worthwhile remarking that MGS, MGS2, CGS and CGS2 computes the approximation in the exact same order. For all the seven approximated eigenvalues and the six orthgonalization kernels, the relative distance is lower than the rounding and convergence accuracy $\delta = \varepsilon = 10^{-3}$, with the largest error realised by Householder for the largest eigenvalue. Setting the rounding accuracy and the convergence one to 10^{-5} changes slightly these observations. As previously, Householder outcompetes the other kernels providing an estimate of five eigenvalues in the chosen spectrum region, while all the other kernels return just four. However, TT-SUBSPIT Householder with accuracy 10^{-5} approximates one eigenvalue in the right region less than with accuracy 10^{-3}, suggesting that the accuracy and the inter-cluster distance may play a role in the computing capabilities of the TT-SUBSPIT algorithm. Notice that for $\delta = 10^{-5}$ the approximation of the eigenvalues outside the right spectrum region are closer to the first theoretical seven than in the $\delta = 10^{-3}$ case. Moreover, the relative distance between the approximated eigenvalues and the theoretical closest ones is significantly lower than the prescribed rounding and converging accuracy 10^{-5}.

		Householder	MGS	MGS2	CGS	CGS2	GRAM
1	$\tilde{\lambda}$	8165.137	8166.140	8166.151	8166.156	8166.138	8136.768
	λ^*	8166.432	8166.432	8166.432	8166.432	8166.432	8136.942
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$1.586e-04$	$3.586e-05$	$3.447e-05$	$3.390e-05$	$3.607e-05$	$2.145e-05$
	convergence order	3	7	7	7	7	2
	spectrum order	1	1	1	1	1	4
2	$\tilde{\lambda}$	8136.560	8136.758	8136.758	8136.758	8136.758	8058.973
	λ^*	8136.942	8136.942	8136.942	8136.942	8136.942	8059.020
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$4.697e-05$	$2.260e-05$	$2.260e-05$	$2.260e-05$	$2.260e-05$	$5.817e-06$
	convergence order	1	1	1	1	1	3
	spectrum order	4	4	4	4	4	15
3	$\tilde{\lambda}$	8136.450	8058.985	8058.985	8058.985	8058.985	8058.601
	λ^*	8136.942	8059.020	8059.020	8059.020	8059.020	8058.693
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$6.051e-05$	$4.330e-06$	$4.331e-06$	$4.330e-06$	$4.331e-06$	$1.145e-05$
	convergence order	5	3	3	3	3	1
	spectrum order	4	15	15	15	15	17
4	$\tilde{\lambda}$	8135.066	8058.559	8058.559	8058.559	8058.559	8021.230
	λ^*	8136.942	8058.693	8058.693	8058.693	8058.693	8021.672
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$2.305e-04$	$1.673e-05$	$1.673e-05$	$1.673e-05$	$1.673e-05$	$5.520e-05$
	convergence order	6	2	2	2	2	4
	spectrum order	4	17	17	17	17	22
5	$\tilde{\lambda}$	8107.330	8021.229	8021.229	8021.229	8021.229	7980.728
	λ^*	8107.528	8021.672	8021.672	8021.672	8021.672	7980.880
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$2.440e-05$	$5.527e-05$	$5.527e-05$	$5.527e-05$	$5.527e-05$	$1.894e-05$
	convergence order	7	4	4	4	4	5
	spectrum order	7	22	22	22	22	35
6	$\tilde{\lambda}$	8106.774	7980.676	7980.676	7980.676	7980.676	7943.274
	λ^*	8107.528	7980.880	7980.880	7980.880	7980.880	7943.348
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$9.300e-05$	$2.546e-05$	$2.546e-05$	$2.546e-05$	$2.546e-05$	$9.334e-06$
	convergence order	2	5	5	5	5	6
	spectrum order	7	35	35	35	35	42
7	$\tilde{\lambda}$	8058.094	7943.245	7943.245	7943.245	7943.245	7942.854
	λ^*	8058.693	7943.348	7943.348	7943.348	7943.348	7943.054
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$7.444e-05$	$1.289e-05$	$1.289e-05$	$1.289e-05$	$1.289e-05$	$2.513e-05$
	convergence order	4	6	6	6	6	7
	spectrum order	17	42	42	42	42	43

Table 3.2 – $m = 7$ eigenvalues for the Laplacian of order $d = 3$ and mode size $n = [19, 24, 31]$ with $\delta = 10^{-3}$.

		Householder	MGS	MGS2	CGS	CGS2	GRAM
1	$\tilde{\lambda}$	8166.432	8166.432	8166.432	8166.432	8166.432	8166.432
	λ^*	8166.432	8166.432	8166.432	8166.432	8166.432	8166.432
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$6.885e-09$	$5.228e-09$	$3.884e-09$	$3.795e-09$	$3.625e-09$	$3.481e-09$
	convergence order	4	5	5	5	5	5
	spectrum order	1	1	1	1	1	1
2	$\tilde{\lambda}$	8137.127	8137.018	8137.018	8137.018	8137.126	8137.018
	λ^*	8137.127	8137.018	8137.018	8137.018	8137.127	8137.018
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$2.561e-08$	$3.193e-09$	$3.193e-09$	$3.193e-09$	$1.051e-07$	$3.204e-09$
	convergence order	6	1	1	1	7	1
	spectrum order	2	3	3	3	2	3
3	$\tilde{\lambda}$	8137.018	8107.710	8107.712	8107.709	8137.018	8107.712
	λ^*	8137.018	8107.712	8107.712	8107.712	8137.018	8107.712
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$3.353e-09$	$3.018e-07$	$2.743e-08$	$4.553e-07$	$3.193e-09$	$1.366e-08$
	convergence order	1	6	6	6	1	6
	spectrum order	3	5	5	5	3	5
4	$\tilde{\lambda}$	8136.942	8107.530	8107.528	8107.531	8107.712	8107.528
	λ^*	8136.942	8107.528	8107.528	8107.528	8107.712	8107.528
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$2.456e-08$	$2.913e-07$	$7.965e-09$	$4.368e-07$	$5.620e-09$	$2.044e-09$
	convergence order	7	7	7	7	6	7
	spectrum order	4	7	7	7	5	7
5	$\tilde{\lambda}$	8107.557	8059.672	8059.672	8059.672	8059.672	8059.672
	λ^*	8107.528	8059.672	8059.672	8059.672	8059.672	8059.672
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$3.534e-06$	$5.534e-09$	$5.534e-09$	$5.534e-09$	$5.534e-09$	$5.398e-09$
	convergence order	5	3	3	3	3	3
	spectrum order	7	12	12	12	12	12
6	$\tilde{\lambda}$	8059.672	8058.693	8058.693	8058.693	8058.693	8058.693
	λ^*	8059.672	8058.693	8058.693	8058.693	8058.693	8058.693
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$5.895e-09$	$5.260e-09$	$5.260e-09$	$5.260e-09$	$5.260e-09$	$5.285e-09$
	convergence order	3	2	2	2	2	2
	spectrum order	12	17	17	17	17	17
7	$\tilde{\lambda}$	8058.693	8021.672	8021.672	8021.672	8021.672	8021.672
	λ^*	8058.693	8021.672	8021.672	8021.672	8021.672	8021.672
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$5.376e-09$	$6.350e-09$	$6.350e-09$	$6.350e-09$	$6.350e-09$	$6.381e-09$
	convergence order	2	4	4	4	4	4
	spectrum order	17	22	22	22	22	22

Table 3.3 – $m = 7$ eigenvalues for the Laplacian of order $d = 3$ and mode size $n = [19, 24, 31]$ with $\delta = 10^{-5}$.

The information about the estimation of the discrete Laplacian of order 3 and mode sizes $[24, 24, 24]$ eigenvalue are depicted in Table 3.4 and 3.5 for accuracy 10^{-3} and 10^{-5} respectively. The analytical value of the 7 largest eigenvalues of this operator are

$$[7470.430,\ 7441.016,\ 7441.016,\ 7441.016,\ 7411.601,\ 7411.601,\ 7411.601\ ].$$

Thus, similarly to the previous experimental case, there is the largest eigenvalue, followed by two eigenvalues with multiplicity 3, that is two degenerate clusters of size 3. Table 3.4 highlights that, as in the previous experimental framework, TT-SUBSPIT with Householder manages to approximate six eigenvalues in the right spectrum region, while all the other kernels lead to eigenvalues located outside the selected region. The relative distance between the approximation and the theoretical eigenvalue is always lower than the rounding accuracy 10^{-3}, with Householder realising the largest relative error. As in the previous case, MGS and CGS with and without re-orthogonalization return the eigenpair approximations in the same order. When the rounding and converging accuracy δ and ε are set equal to 10^{-5}, the situation changes. Indeed, as reported in Table 3.5, Householder leads to four estimations in the right spectrum region, while all the other kernels to three. As in the other experiment set-up, when the accuracy value decreases, all the kernels performances improve, except for Householder, which remains in any case the most effective. Moreover, in this case all the kernels except Householder present the same convergence order. As previously, the relative distance between the numerical eigenvalue and the theoretical one is much lower than the rounding accuracy 10^{-5}.

		Householder	MGS	MGS2	CGS	CGS2	GRAM
1	$\tilde{\lambda}$	7469.291	7382.018	7382.018	7382.018	7382.018	7382.010
	λ^*	7470.430	7382.187	7382.187	7382.187	7382.187	7382.187
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$1.525e-04$	$2.283e-05$	$2.283e-05$	$2.283e-05$	$2.283e-05$	$2.400e-05$
	convergence order	6	1	1	1	1	1
	spectrum order	1	11	11	11	11	11
2	$\tilde{\lambda}$	7440.841	7266.610	7266.610	7266.610	7266.610	7266.611
	λ^*	7441.016	7266.841	7266.841	7266.841	7266.841	7266.841
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$2.349e-05$	$3.179e-05$	$3.179e-05$	$3.179e-05$	$3.179e-05$	$3.164e-05$
	convergence order	3	2	2	2	2	3
	spectrum order	3	36	36	36	36	36
3	$\tilde{\lambda}$	7440.597	7266.598	7266.598	7266.598	7266.598	7266.603
	λ^*	7441.016	7266.841	7266.841	7266.841	7266.841	7266.841
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$5.629e-05$	$3.342e-05$	$3.342e-05$	$3.342e-05$	$3.342e-05$	$3.275e-05$
	convergence order	5	3	3	3	3	2
	spectrum order	3	36	36	36	36	36
4	$\tilde{\lambda}$	7439.689	7266.503	7266.503	7266.503	7266.503	7266.566
	λ^*	7441.016	7266.841	7266.841	7266.841	7266.841	7266.841
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$1.783e-04$	$4.655e-05$	$4.655e-05$	$4.655e-05$	$4.655e-05$	$3.795e-05$
	convergence order	7	4	4	4	4	4
	spectrum order	3	36	36	36	36	36
5	$\tilde{\lambda}$	7411.681	7151.317	7151.317	7151.317	7151.317	7151.279
	λ^*	7411.601	7151.496	7151.496	7151.496	7151.496	7151.496
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$1.070e-05$	$2.496e-05$	$2.496e-05$	$2.496e-05$	$2.496e-05$	$3.025e-05$
	convergence order	4	6	6	6	6	7
	spectrum order	5	76	76	76	76	76
6	$\tilde{\lambda}$	7410.233	7151.150	7151.149	7151.150	7151.150	7151.103
	λ^*	7411.601	7151.496	7151.496	7151.496	7151.496	7151.496
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$1.820e-04$	$4.837e-05$	$4.845e-05$	$4.832e-05$	$4.838e-05$	$5.485e-05$
	convergence order	2	7	7	7	7	5
	spectrum order	5	76	76	76	76	76
7	$\tilde{\lambda}$	7379.785	7151.073	7151.073	7151.073	7151.073	7150.870
	λ^*	7382.187	7151.496	7151.496	7151.496	7151.496	7151.496
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$3.234e-04$	$5.907e-05$	$5.907e-05$	$5.907e-05$	$5.907e-05$	$8.749e-05$
	convergence order	1	5	5	5	5	6
	spectrum order	11	76	76	76	76	76

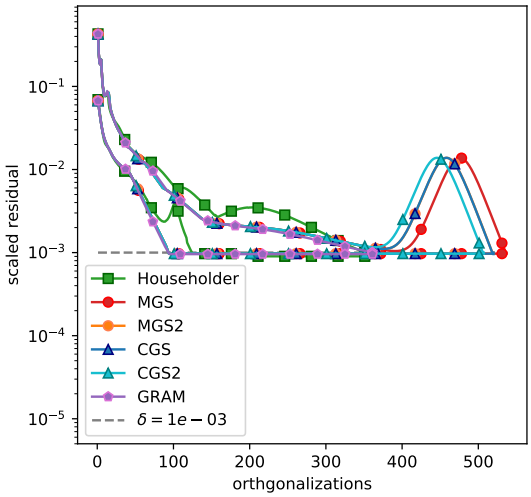
Table 3.4 – $m = 7$ eigenvalues for the Laplacian of order $d = 3$ and mode size $n = [24, 24, 24]$ with $\delta = 10^{-3}$.

		Householder	MGS	MGS2	CGS	CGS2	GRAM
1	$\tilde{\lambda}$	7470.430	7411.601	7411.601	7411.601	7411.601	7411.601
	λ^*	7470.430	7411.601	7411.601	7411.601	7411.601	7411.601
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$2.501e-08$	$2.628e-09$	$2.753e-09$	$2.586e-09$	$2.862e-09$	$2.682e-09$
	convergence order	7	7	7	7	7	6
	spectrum order	1	5	5	5	5	5
2	$\tilde{\lambda}$	7441.016	7411.601	7411.601	7411.601	7411.601	7411.601
	λ^*	7441.016	7411.601	7411.601	7411.601	7411.601	7411.601
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$7.972e-09$	$2.861e-09$	$3.139e-09$	$2.708e-09$	$2.957e-09$	$2.840e-09$
	convergence order	6	6	6	6	6	7
	spectrum order	3	5	5	5	5	5
3	$\tilde{\lambda}$	7411.601	7411.601	7411.601	7411.601	7411.601	7411.601
	λ^*	7411.601	7411.601	7411.601	7411.601	7411.601	7411.601
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$8.870e-09$	$8.889e-09$	$8.520e-09$	$8.886e-09$	$8.787e-09$	$8.175e-09$
	convergence order	4	5	5	5	5	5
	spectrum order	5	5	5	5	5	5
4	$\tilde{\lambda}$	7411.601	7382.187	7382.187	7382.187	7382.187	7382.187
	λ^*	7411.601	7382.187	7382.187	7382.187	7382.187	7382.187
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$9.363e-09$	$2.048e-09$	$2.048e-09$	$2.048e-09$	$2.048e-09$	$1.998e-09$
	convergence order	5	1	1	1	1	1
	spectrum order	5	11	11	11	11	11
5	$\tilde{\lambda}$	7382.187	7266.841	7266.841	7266.841	7266.841	7266.841
	λ^*	7382.187	7266.841	7266.841	7266.841	7266.841	7266.841
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$1.896e-09$	$3.170e-09$	$3.170e-09$	$3.170e-09$	$3.170e-09$	$3.141e-09$
	convergence order	1	2	2	2	2	2
	spectrum order	11	36	36	36	36	36
6	$\tilde{\lambda}$	7266.841	7266.841	7266.841	7266.841	7266.841	7266.841
	λ^*	7266.841	7266.841	7266.841	7266.841	7266.841	7266.841
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$1.877e-09$	$3.759e-09$	$3.759e-09$	$3.759e-09$	$3.759e-09$	$3.801e-09$
	convergence order	2	3	3	3	3	3
	spectrum order	36	36	36	36	36	36
7	$\tilde{\lambda}$	7266.841	7266.841	7266.841	7266.841	7266.841	7266.841
	λ^*	7266.841	7266.841	7266.841	7266.841	7266.841	7266.841
	$\frac{\lvert\lambda^*-\tilde{\lambda}\rvert}{\lvert\lambda^*\rvert}$	$4.857e-09$	$6.020e-09$	$6.020e-09$	$6.018e-09$	$6.018e-09$	$6.011e-09$
	convergence order	3	4	4	4	4	4
	spectrum order	36	36	36	36	36	36

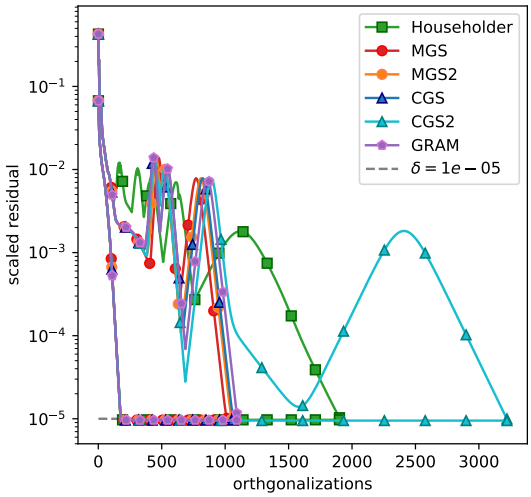
Table 3.5 – $m = 7$ eigenvalues for the Laplacian of order $d = 3$ and mode size $n = [24, 24, 24]$ with $\delta = 10^{-5}$.

To complete the investigation of the convergence quality depending on the chosen orthogonalization kernel, we study the minimum and the maximum value of the stopping criterion, that is the scaled residual; for simplicity we will refer to this by the term *scaled residual envelope*. More precisely, we study the scaled residual envelope in function of the number of orthogonalizations, counted by the `i` variable in Algorithm 19, and of the number of TT-rounding call performed, estimated by the `iter` variable in Algorithm 19. The number of orthogonalizations counts the number of loops, while the number of TT-rounding estimates the algorithm complexity. In Figures 3.5A and 3.5B, we display the scaled residual envelope in function of the number of orthogonalizations and of TT-rounding for the discrete order 3 Laplacian of mode sizes $[19, 24, 31]$. From the orthogonalization counter point of view, Householder seems to be the optimal one, together with Gram for $\delta = 10^{-3}$, as seen in Figure 3.5A, while for $\delta = 10^{-5}$, CGS2 is the slowest one and Householder the second slowest one, see Figure 3.5B. From the number of TT-rounding instead, Householder is always the slowest, as shown in Figure 3.5C and 3.5D, i.e., it is the one with the highest complexity in terms of TT-rounding, the most expensive operation. However, it is important to remark that Householder for both the accuracy values leads to the best results in terms of robustness to compute the m largest eigenvalues. Remark that for $\delta = 10^{-5}$ all the envelopes display many more peaks and valleys compared to the envelopes for $\delta = 10^{-3}$. Notice that also in the classical linear algebra scheme the convergence monotony of the subspace iteration method is not theoretically guaranteed.

The scaled residual envelope from the discrete Laplacian of mode sizes $[24, 24, 24]$ experiments are displayed in Figure 3.6. In this case the Householder kernel is the one requiring more orthogonalizations and more TT-rounding steps to converge, but for both the accuracy values it leads to the best results in terms of robustness to compute the m largest eigenvalues. We highlight that as in the previous case, for $\delta = 10^{-5}$ all the envelopes present peaks and valleys, while for $\delta = 10^{-3}$ all the envelopes except the Householder one are monotonously decreasing.

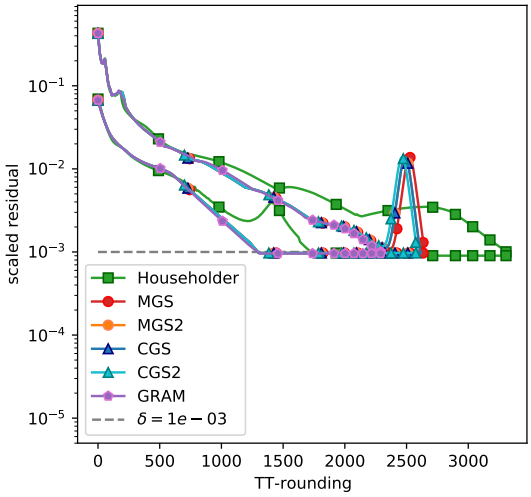


(A) $\delta = 10^{-3}$

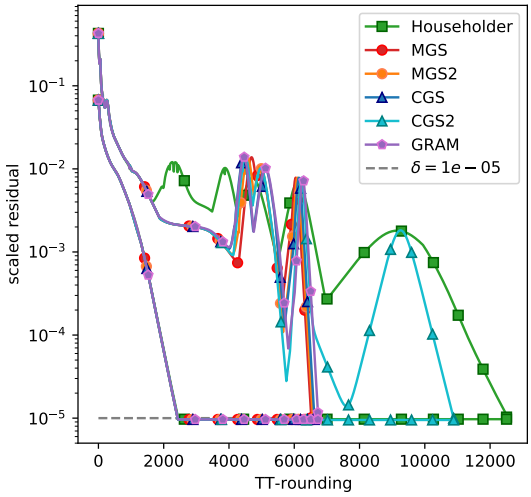


(B) $\delta = 10^{-5}$

Residual envelope in function of the number of orthogonalization



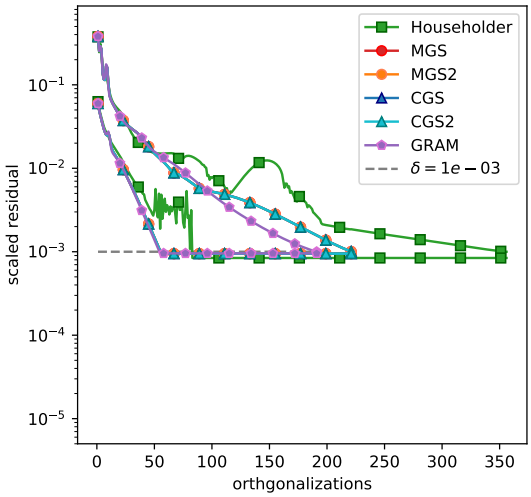
(C) $\delta = 10^{-3}$



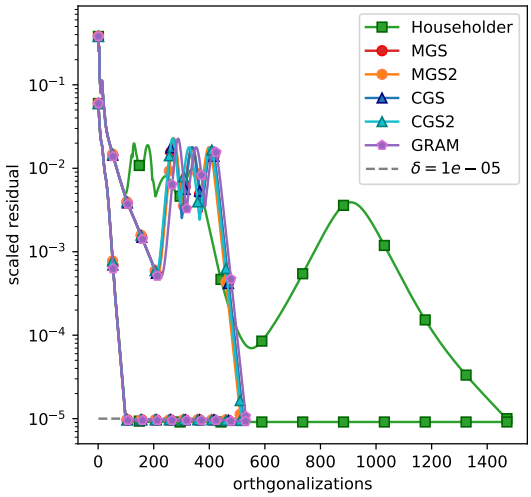
(D) $\delta = 10^{-5}$

Residual envelope in function of the number of TT-rounding

Figure 3.5 – Residual envelope for $m = 7$ TT-eigenpairs of order $d = 3$ and mode size $n = [19, 24, 31]$.

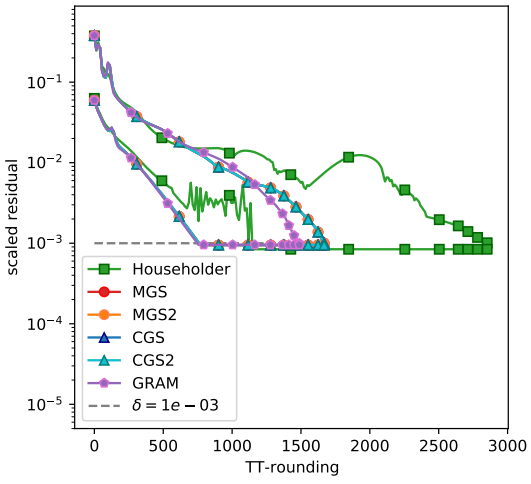


(A) $\delta = 10^{-3}$

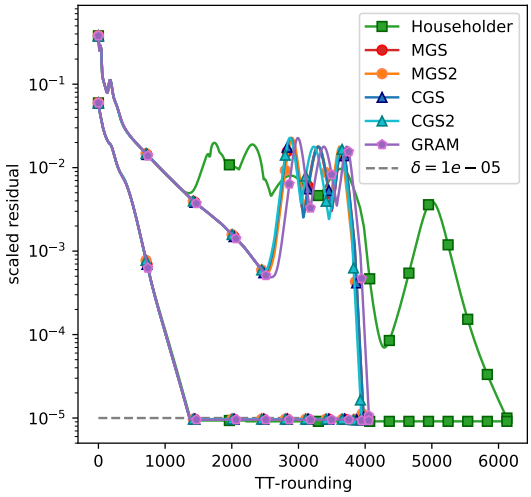


(B) $\delta = 10^{-5}$

Residual envelope in function of the number of orthogonalization



(C) $\delta = 10^{-3}$



(D) $\delta = 10^{-5}$

Residual envelope in function of the number of TT-rounding

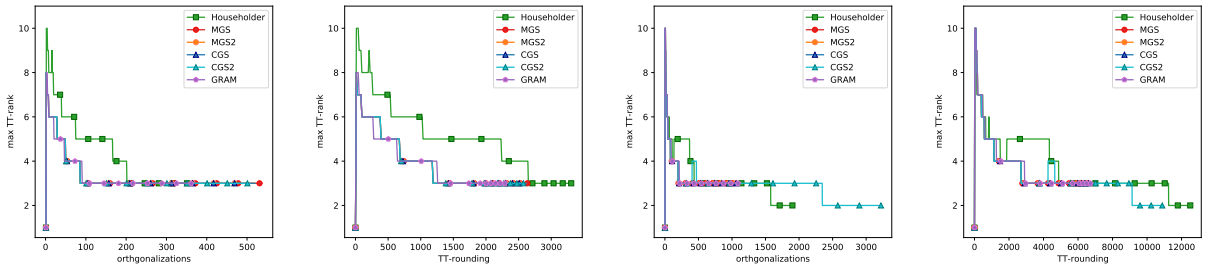
Figure 3.6 – Residual envelope for $m = 7$ TT-eigenpairs of order $d = 3$ and mode size $n = [24, 24, 24]$.

3.3.2.2 Memory requirement

This section is devoted to the analysis of the TT-eigenvector memory requirement through the study of the maximal TT-rank and the maximal compression ratio. In particular we display the maximum, among all the $m = 7$ computed TT-eigenvectors, the maximum TT-rank and compression ratio. Notice that, as in the previous section, we display both the maximal TT-rank and the compression ratio in function of the number of orthogonalization and of TT-rounding steps.

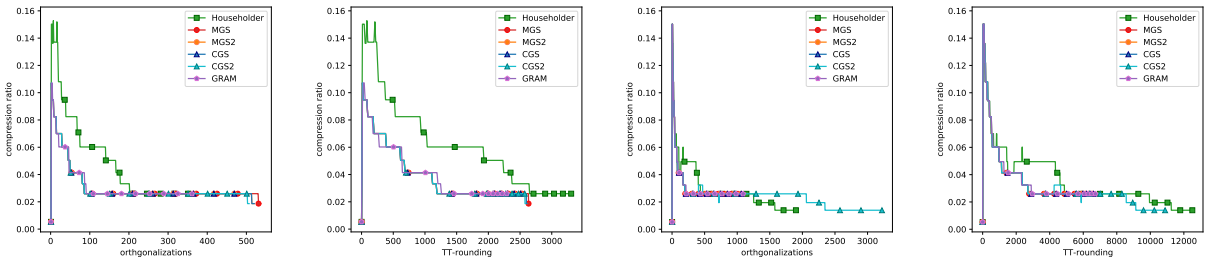
The memory requirement for the approximation of the TT-eigenvectors of the discrete order $d = 3$ Laplacian of mode sizes $[19, 24, 31]$ are shown in Figure 3.7. For $\delta = 10^{-3}$, all the orthogonalization kernels lead to some initial oscillations, higher for Householder, and then to a maximum TT-rank stagnating at 3. For $\delta = 10^{-5}$, the maximum TT-rank of the TT-eigenvectors approximated through TT-SUBSPIT with Householder or CGS2 stagnates at 3, while when the other orthogonalization kernels are used, the maximum TT-rank sets to 4. As comparison term, remark that the theoretical eigenvectors in TT-format have TT-rank equal to 1.
From Figures 3.7E, 3.7F the compression ratio of TT-eigenvector approximations presents some peaks, which hit 16% of the memory needed to store the dense format tensor for Householder and 10% for the other schemes, and finally it stagnates around $2\% - 3\%$. Similarly, in Figures 3.7G and 3.7H, the compression ratio has some peaks reaching, for all kernels, 16% of the memory requested to store the same tensors in dense format and it sets between 2% and 3% at convergence, when the rounding accuracy is 10^{-5}. In this case, the maximal TT-rank and the compression ratio curves in function of the number of TT-roundings or orthogonalizations do not present significantly different behaviours.

The memory requirement of TT-eigenvector approximations for the discrete order 3 Laplacian with all mode size equal to 24 are displayed in Figure 3.8. For the rounding and the converging accuracy equal to 10^{-3}, the maximal TT-rank hits 10 when the Householder kernel is used and 8 when the other orthogonalization schemes are applied, stagnating in any case at 3 at convergence, as displayed in Figure 3.8A and 3.8B. For $\delta = 10^{-5}$, the maximal TT-rank associated with every orthogonalization schemes hits 10 during a first phase and sets to 2 for Householder and to 3 for the other kernels, see Figure 3.8C and 3.8D. From the compression ratio side, Figures 3.8E and 3.8F highlight that the compression ratio of the TT-eigenvectors approximated with the Householder kernel reaches higher peaks, as 0.2, that reads as 20% of the dense format memory requirement, than the other schemes, which do not overcome 0.13. However, at convergence the Householder compression ratio stagnates around 0.02 slightly below the other scheme curves. Similarly for $\delta = 10^{-5}$, all the compression ratio curves reach 0.2 during a first phase and they decrease to less than 0.02 for the Householder one and to 0.02 for the other schemes, as displayed in Figures 3.8G and 3.8H.



(A) $\delta = 10^{-3}$ (B) $\delta = 10^{-3}$ (C) $\delta = 10^{-5}$ (D) $\delta = 10^{-5}$

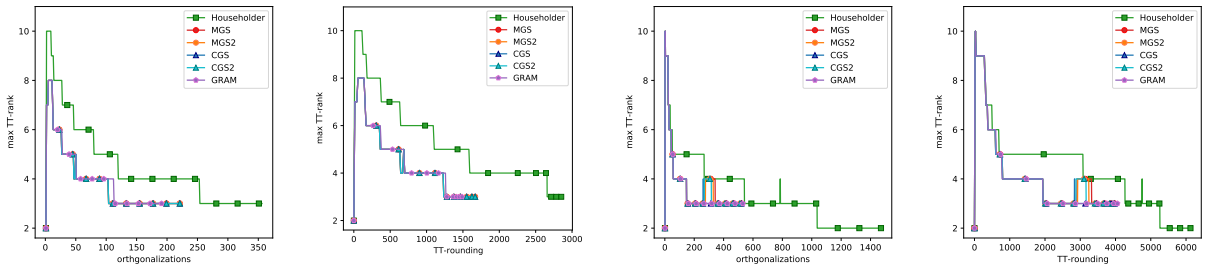
Maximal TT-rank in function of the number of orthogonalization or TT-rounding



(E) $\delta = 10^{-3}$ (F) $\delta = 10^{-3}$ (G) $\delta = 10^{-5}$ (H) $\delta = 10^{-5}$

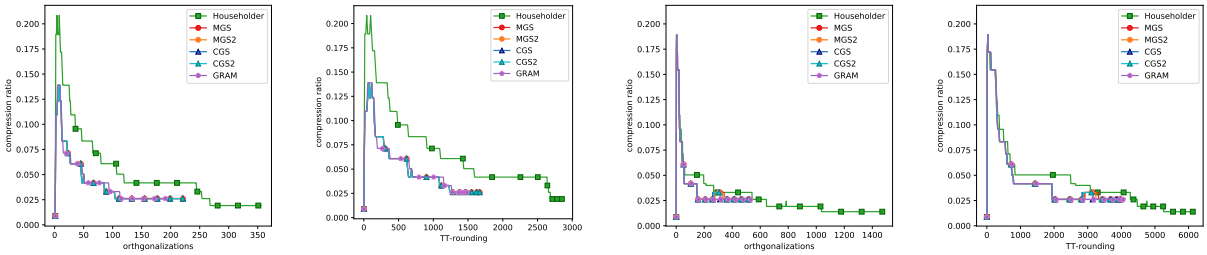
Compression ratio in function of the number of orthogonalization or TT-rounding

Figure 3.7 – Memory requirement for TT-eigenvector for $m = 7$ TT-eigenpairs of order $d = 3$ and mode size $n = [19, 24, 31]$.



(A) $\delta = 10^{-3}$ (B) $\delta = 10^{-3}$ (C) $\delta = 10^{-5}$ (D) $\delta = 10^{-5}$

Maximal TT-rank in function of the number of orthogonalization or TT-rounding



(E) $\delta = 10^{-3}$ (F) $\delta = 10^{-3}$ (G) $\delta = 10^{-5}$ (H) $\delta = 10^{-5}$

Compression ratio in function of the number of orthogonalization or TT-rounding

Figure 3.8 – Memory requirement for TT-eigenvector for $m = 7$ TT-eigenpairs of order $d = 3$ and mode size $n = [24, 24, 24]$.

3.4 Concluding remarks

This chapter aims to study the orthogonalization algorithms implemented with the TT-formalism with a particular focus on the TT-rounding effects. We firstly describe our extension to the tensor framework of six classical orthogonalization schemes through the TT-formalism. Then, we study the quality in terms of orthogonality of the computed basis and we relate our numerical results with the classical linear algebra theoretical ones. For completeness, we present an application of the considered kernels: an iterative eigensolver in TT-format. After describing the chosen eigensolver algorithm and highlighting its tensor specificities, we analyse its behaviour. Through a study case, we investigate how the chosen orthogonalization methods affect the quality of the computed eigenvalues and TT-eigenvectors. In all the numerical experiments, particular attention is paid to the memory requirements, which are known to be a major constraint when tensors are involved.

In the first part, corresponding to Sections 3.2, six orthogonalization kernels are described. Firstly we present the Classical Gram-Schmidt (CGS) and its Modified version (MGS) extended to the tensor framework with the TT-formalism. To be as comprehensive as possible, the CGS and MGS algorithms with re-orthogonalization are addressed as well. After detailing the structure in TT-format of the Gram algorithm, the Householder transformation is introduced. This last scheme requires a more accurate generalization in TT-format, which is completely given in Section 3.2.3. To face the memory requirements, we include in all the orthogonalization schemes further rounding steps, that play a crucial role in maintaining the computations affordable and in determining the quality of the computed orthogonal basis. The theoretical classical linear algebra bounds for the loss of orthogonality of these six orthogonalization kernels are presented in Section 3.2.4 and compared with the numerical experiments reported in Section 3.2.5. We generate with a Krylov process an input set of TT-vectors, which become more and more collinear by construction, and we orthogonalize them through the considered methods. The two sets of experiments of different orders suggest that the theoretical classical bounds can be generalized to the tensor context with the TT-formalism replacing the IEEE unit round-off with the TT-rounding accuracy. We include an analysis of the memory requirement for the basis computation estimated through the maximal TT-rank and the compression ratio of TT-vectors forming the orthogonal basis. Moreover, the complexity of the algorithms is evaluated in terms of TT-rounding operations, which represent the major computational cost.

The second part, coinciding with Section 3.3, presents a possible application of these kernels, that is inside the SUBSPace ITeration eigensolver formulated in TT-format (TT-SUBSPIT). Section 3.3.1 includes the description of the TT-SUBSPIT algorithm, while the numerical results are found in Section 3.3.2. As for the orthogonalization schemes, also for TT -SUBSPIT some TT-rounding steps are required to prevent memory deficiencies. For the numerical experiments, we consider a classical study case in tensor format, i.e., the discrete order 3 Laplacian with different mode sizes. We compare the computed eigenvalues at different rounding accuracies with their analytical values, remarking that the Householder kernel seems to outcompete the other ones approximating more eigenvalues

in the right spectrum region. For the sake of completeness, we analyse the convergence through the stopping criterion, that is the scaled residual, and the memory requirement through the maximal TT-rank and the compression ratio. Those preliminary results are a starting point for the investigation of the TT-SUBSPIT algorithm for solving tensor eigenproblems.

Part II

Data analysis

II.I Introduction

Data science, and every aspect related to this discipline, have been on the crest of a wave since 2010 [17] when media and business popularized it, see [34, 100, 102, 117]. Even if it has impacted economic, social and scientific levels [17], data science as a field of study has not yet a clear and universally accepted definition. The problem of stating what are the topics of interest in data science was firstly addressed in 1962 by Tukey [138], who started the discussion inside the scientific community long before this discipline gained its notoriety. Notice that in 1962 Tuckey was using the term 'data analysis' and the expression 'data science' arrived later, in 1985 at the Chinese Academy of Sciences in Beijing during a talk by Jeff Wu [17], who proposed it as a substitute for 'statistics'. A more wide and more comprehensive definition was proposed in 1998 by Hayashi [67], who highlighted data science's multidisciplinary nature and the three main investigation topics, i.e., how to design, collect, and analyse data. Nowadays, a popular definition [17, 130] highlights the presence of methods and principles, coming from both theoretical areas, such as statistics or computer science, and applicative ones, such as economics, psychology, and biology.

A central topic in data science is analysing the data, extrapolating and visualizing the significant information. The increasing number of large-size datasets, for example, those coming from CERN experiments or Hubble Space Telescope images, from private technology companies such as Google, Amazon, and Facebook, or private sensors or devices, has acted as propulsion for developing new data analysis techniques able to handle huge data. Therefore, in recent years researchers from different fields started modelling and investigating their datasets, relying on tensor theory and algorithms. In the following chapters, we address two data analysis techniques, namely Correspondence Analysis (CA) and climate data analysis popular method referred to by Empirical Orthogonal Function (EOF).

II.I.I Correspondence Analysis background

In 1904 Pearson referred with the expression '*contingency tables*' to datasets whose values are frequencies or counts of a combination of two categories belonging respectively to two different categorical variables, for example, the hair colour or the social class [118]. The standard technique for analysing contingency tables and visualizing their information is Correspondence Analysis (CA), whose mathematical roots appeared during the 1930s in Richardson's and Kuder's [120], Horst's [74] and Hartley's [71] (whose former family name was Hirschfeld) works. During the 1940s, the notable statisticians, Fisher [43] and Guttman [60] separately derived the same technique in different contexts, that is biometrics and psychometrics. The ideas of Guttman were further developed by Hayashi [66] between the 1940s and the 1950s, while in the same period Williams [146] worked on Fisher's proposal. All those contributions, sharing the same background procedure, are meant for returning a numerical result rather than a visual one [57]. Only with the advent of Benzécri's research during the 1960s, the first geometrical interpretations and outputs

arrived together with the French expression 'analyse des correspondances' [12, 14]. Hill's publication of 1974 [70] diffused Benzécri's and his collaborators' results, previously limited to the French audience, suggesting the name correspondence analysis directly from the French translation. His later publication of 1982 [69] defines CA with its geometric aspects.

In many domains, for example, social or biological sciences, frequently count data come from more than two categorical variables; these datasets are usually called *multiway contingency tables*. A common technique for studying multiway contingency tables is Multiple Correspondance Analysis (MCA). The principle of MCA is associating the multiway table with an indicator or Burt table [20], which is mathematically speaking, a matrix of only 0 and 1. The roots of these techniques can be found already in Guttman's paper of 1941 [60]. The expression 'multiple correspondence analysis' was presented by Burt in 1950, but the complete formulation of MCA as analysing and visualizing methods was proposed during the 1970s by Benzécri and Lebart [14, 93, 95]. Starting from the 1970s, several researchers developed MCA; the works of Kroonenberg, Kiers, Marcotorchino, Clausen, Beh, D'Ambra, and Lombardo contributed to the MCA investigation in the last thirty years, and an exhaustive description is presented in [10]. As pointed out in [10], MCA still relies on matrix representations of multiway tables, and consequently, it discards without the possibility of retrieving the information about the interactions among more than two variables. Thus, at the end of the 1990s [10, 88], tensors were introduced to mathematically represent multiway tables, developing the MultiWay Correspondence Analysis (MWCA) technique, even if some previous attempts were already made between the 1970s and the 1980s [88]. Lombardo's [98], Carlier's, and Kroonenberg's [21, 22, 87, 88, 89] results are based on the CP [72] and the Tucker [137] decomposition.

II.I.II Climate data analysis history

The weather and climate have always fascinated and intrigued humankind. Even if the collection of weather information started already during the XIX century, the scientific investigation of climate patterns through mathematical techniques made significant progress only from the 1920s [63]. After the preliminary results of Walker and Ångström obtained between the 1920s and the 1930s, Obukhov and Bagrov introduced linear algebra tools in the study of climate data between 1940 and 1950 [63]. The invention of *Empirical Orthogonal Function* (EOF), today's popular climate data analysis technique [64], is attributed to Lorenz [99], who first proposed the expression, even if other scientists as Obukhov [107] and Fukuoka [45] already described a similar analysis procedure. Since the second half of the XX century, the EOF method has continuously developed with many theoretical and practical contributions; we refer the reader to [63, 64] and to the references therein for an exhaustive list of EOF-related works.

The EOF method is meant for the investigation and the independent visualization of data that depends on time and space, which are the usual variable of climate data. In particular, it works on dimensional reduction and pattern extraction [64]. Since climate datasets are usually of huge dimensions, it would be natural to generalize the EOF method

through tensor compression techniques. We are not aware of any previous contribution that describes EOF in the tensor framework, even if in literature are present some studies of climate data through tensor techniques, as for example [149].

II.II Statistics preliminaries

In this section, we introduce some basic statistical concepts, that we will use in the following chapters

The first key concept is the average or mean of an array, whether it represents a vector, a matrix, or a tensor.

Definition II.II.i. *Let $x \in \mathbb{R}^n$ be a vector, whose i-th component represent the i-th observation of a random variable, the (sample) average or mean of x is $\mu_x \in \mathbb{R}$ defined as*

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n} x(i).$$

The definition of average is generalized mode-wise to matrices and tensors. Given a tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, its (sample) average or mean with respect to mode k is an order $(d-1)$ tensor $\overline{\mathbf{x}}_k \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times n_{k+1} \times \cdots \times n_d}$ such that

$$\overline{\mathbf{x}}_k(i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_d) = \frac{1}{n_k}\sum_{i_k=1}^{n_k} \mathbf{x}(i_1, \ldots, i_k, \ldots, i_d).$$

for every $i_h \in \{1, \ldots, n_h\}$ with $h \in \{1, \ldots, k-1, k+1, \ldots, d\}$ and $k \in \{1, \ldots, d\}$. If $d = 2$, the considered data structure is a matrix with $\overline{x}_1 \in \mathbb{R}^{n_2}$ and $\overline{x}_2 \in \mathbb{R}^{n_1}$ its *column* and *row* average respectively.

The variance is a mathematical object closely related to the average. As previously, we present the definition of variance for a vector and then we generalize it to tensors.

Definition II.II.ii. *Let $x \in \mathbb{R}^n$ be a vector, whose i-th component represent the i-th observation of a random variable, the (sample) unbiased variance of x is $s_x \in \mathbb{R}$ defined as*

$$s_x = \frac{1}{n-1}\sum_{i=1}^{n}(x(i) - \mu_x)^2 = \frac{1}{n}\langle x - \mu_x \mathbb{1}_n, x - \mu_x \mathbb{1}_n\rangle = \frac{1}{n}\|x - \mu_x \mathbb{1}_n\|^2$$

where $\mathbb{1}_n \in \mathbb{R}^n$ is a vector of ones. The standard deviation is $\sigma_x = \sqrt{s_x}$.

The (sample) unbiased variance with respect to mode k of an order d tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is an order $(d-1)$ tensor $\mathbf{s}_k \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times n_{k+1} \times \cdots \times n_d}$ such that

$$\begin{aligned}\mathbf{s}_k(i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_d) &= \frac{1}{n_k - 1}\sum_{i_k=1}^{n_k}\Big(\mathbf{x}(i_1, \ldots, i_k, \ldots, i_d) - \overline{\mathbf{x}}_k(i_1, \ldots, i_{k-1}, i_{k+1} \ldots, i_d)\Big)^2 \\ &= \frac{1}{n_k - 1}\sum_{i_k=1}^{n_k}\left(\Big(\mathbf{x} - \mathbb{1}_{n_k} \otimes \overline{\mathbf{x}}_k\Big)(i_1, \ldots, i_k, \ldots, i_d)\right)^2\end{aligned}$$

where $\mathbb{1}_{n_k} \in \mathbb{R}^{n_k}$ is a vector of ones, for every $i_h \in \{1, \dots, n_h\}$ with $h \in \{1, \dots, k-1, k+1, \dots, d\}$ and $k \in \{1, \dots, d\}$. The standard deviation with respect to mode k of $\mathbf{x}$ is the order $(d-1)$ tensor defined as the square root of the mode k variance, that is $\boldsymbol{\sigma}_k = \sqrt{\mathbf{s}_k}$. Another statistical definition, that we will use later on, is the covariance.

Definition II.II.iii. *Given two vectors $x, y \in \mathbb{R}^n$ whose components are the observations of two different random variables respectively, then $c_{x,y} \in \mathbb{R}$ is the covariance of x and y computed as*

$$c_{x,y} = \langle x - \mu_x \mathbb{1}_n, y - \mu_y \mathbb{1}_n \rangle.$$

More generally if $X \in \mathbb{R}^{n_1 \times n_2}$ is a matrix whose i-th column $x_i \in \mathbb{R}^n$ is the realization of a random variable for every $i \in \{1, \dots, p\}$, then its covariance matrix is $Q_X \in \mathbb{R}^{n_2 \times n_2}$ such that

$$\begin{aligned} Q_X(j,k) &= \sum_{i=1}^{n_1} \Big(X(i,j) - \overline{x}_1(j)\Big)\Big(X(i,k) - \overline{x}_1(k)\Big) \\ &= \Big(X - \mathbb{1}_{n_1} \otimes \overline{x}_1\Big)^\top \Big(X - \mathbb{1}_{n_1} \otimes \overline{x}_1\Big) \end{aligned}$$

where $\mathbb{1}_{n_1} \in \mathbb{R}^{n_1}$ is a vector of ones. The (j,k)-th element of Q_X is actually equal to the covariance of the j-th and k-th column of X, if $j \neq k$, while $Q_X(j,j)$ is the (scaled) variance of the j-th column of X.

II.III Principal Component Analysis

A common technique to explore data is Principal Components Analysis (PCA) [5, 28, 46, 75, 78, 79, 82, 105, 119]. The analysis procedures we investigate in the following chapters, i.e., CA in Chapter 4 and the EOF analysis in Chapter 5, rely on PCA. This method has three different natures, which are statistical, geometrical, and algebraical, that CA and EOF analysis inherit. We briefly present those three facets, under the same assumptions. Let $X \in \mathbb{R}^{n \times p}$ be a matrix whose i-th column $x_i \in \mathbb{R}^n$ is the realization of a random variable for every $i \in \{1, \dots, p\}$. In addition, we assume that every column of X has zero mean, i.e., $\overline{x}_1(j) = 0$ for every $j \in \{1, \dots, p\}$.

II.III.I Statistical viewpoint

The statistical purpose of PCA is to find a linear combination of the random variables that maximize their covariance. Mathematically, this problem seeks the first principal component $y \in \mathbb{R}^n$ such that $y = Xv^*$ where $v^* \in \mathbb{R}^p$ is the said to be the *loading* and it

is obtained as

$$\begin{aligned} v^* &= \arg\max_{||v||=1}(s_{Xv}) \\ &= \frac{1}{p}\arg\max_{||v||=1}\langle Xv, Xv\rangle \\ &= \frac{1}{p}\arg\max_{||v||=1}(v^\top X^\top Xv) \\ &= \frac{1}{p}\arg\max_{||v||=1}(v^\top Q_X v) \end{aligned}$$

with the covariance matrix $Q_X = X^\top X$ thanks to the assumption that the column average of X is zero. Since the matrix Q_X is by construction symmetric, then it can be decomposed into the orthogonal basis formed by its eigenvectors, that is $Q_X = W^\top \Lambda W$ with $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_p)$ where w_i and λ_i are the i-th eigenvector and eigenvalue respectively. We assume that the eigenvalues, and consequently the eigenvectors, are sorted with decreasing absolute values. Thus, the PCA problem is

$$\begin{aligned} v^* &= \frac{1}{p}\arg\max_{||v||=1}(v^\top W^\top \Lambda W v) \\ &= \frac{1}{p}\arg\max_{\substack{Wv=z \\ ||v||=1}}(z^\top \Lambda z) \end{aligned}$$

The solution to this maximization problem is $z^* = e_1$ the first canonical basis vector and v^* equal to w_1 the first eigenvector, associated with the largest eigenvalue. Consequently, the first principal component is $y_1 = Xv^* = Xw_1$. If we want to determine r principal components, we impose that the r loadings have norm 1 and are mutually orthogonal, getting that the i-th loading is the i-th eigenvector of the covariance matrix Q_X and the i-th principal component is $y_i = Xw_i$ for every $i \in \{1, \ldots, r\}$. We refer the reader to [101, Theorem 2.1] for further details about the statistical viewpoint of PCA.

II.III.II Geometrical viewpoint

In this case, it is more convenient to assume that the observations are arranged columnwise, i.e., assuming that $Y \in \mathbb{R}^{p\times n}$ with n the number of observations and p the number of random variables considered. Remark that $Y = X^\top$ used in the previous section.

The geometrical facet of PCA regards the j-th column of Y as the coordinate array of the j-th observation in a space of dimension p. Thus, the aim of PCA from the geometrical viewpoint is finding a subspace of $\mathbb{R}^p$ of dimension r which minimizes the projection error, i.e., the PCA is geometrically speaking a dimensionality reduction method. Mathematically the problem is determining $\mathfrak{S}^* \subseteq \mathbb{R}^p$ with $\dim(\mathfrak{S}^*) = r$ which minimizes the projection error of all the columns of Y, that is

$$\mathfrak{S}^* = \arg\min_{\mathfrak{S}\subseteq\mathbb{R}^p} \sum_{i=1}^{n} \|y_i - \pi_{\mathfrak{S}}(y_i)\|^2$$

where $\pi_{\mathfrak{S}} : \mathbb{R}^p \mapsto \mathfrak{S}$ is the projection map. Each subspace is determined by a basis, thus the projection map can be expressed using the selected basis. If the columns of $U \in \mathbb{O}(p \times r)$ form an orthonormal basis of $\mathfrak{S}$, that is $\mathfrak{S} = \text{span}(U)$, then the projection by $\pi_{\mathfrak{S}}$ of $z \in \mathbb{R}^p$ is expressed as $\pi_{\mathfrak{S}} z = UU^\top z$. Thanks to this result, the geometric PCA problem becomes determining the orthonormal basis $U^* \in \mathbb{O}(p \times r)$ which minimizes the projection error, i.e.,

$$U^* = \arg \min_{U^\top U = \mathbb{I}_r} \sum_{i=1}^{n} \left\| y_i - UU^\top y_i \right\|^2 .$$

This is a classical least squares problem in linear algebra, whose solution is given by $U \in \mathbb{O}(p \times r)$ the left orthogonal matrix of the SVD of $Y = U\Sigma V^\top$ truncated at the r-th column, which is also called loading matrix. The matrix $U^\top Y$, which is equal to $\Sigma V^\top$ by construction, is referred as the *principal components* or *coordinates* of Y. The columns of U define the axis of the low dimensional space where we display the data, while the columns of $\Sigma V^\top$ represent the data coordinates in the new low dimensional subspace.
Notice that the statistical and the geometrical results are coherent. Indeed, in the statistical loading matrix is W the eigenvector matrix of Q_X, which is equal to the right orthogonal matrix of the SVD of X, that is $X = Y^\top = V\Sigma U^\top$ by construction, i.e., the loading matrix found with the geometrical approach, cf. [101, Theorem 2.3].

II.III.III Algebraic viewpoint

In the algebraic context, given $Y \in \mathbb{R}^{n \times p}$, PCA seeks a matrix $Z^* \in \mathbb{R}^{n \times p}$ of rank r which minimizes the approximation error, that is

$$Z^* = \arg \min_{\text{rank}(Z)=r} \|Y - Z\|$$

with $r < \text{rank}(Y)$. Under this formulation, the best solution is guaranteed by the Eckart-Young theorem [41] through the SVD of Y. This algebraic approach is exactly equivalent to the previous two, leading indeed to the same solution, cf [101, Theorem 2.6]. This type of approach is often of interest in face recognition problems.

II.IV Tensor formalism

The generalization of PCA to tensor has been for many years a discussion topic, leading to different results, summarized in [87, 89], relying either on the Tucker model or on the Canonical Polyadic one. The study performed and reported in the following chapters are based on the Tucker model and its HOSDV realization, even if it is known that the HOSVD approximation at a prescribed accuracy or prescribed multilinear does not guarantee the optimality of the approximation as SVD does.

Chapter 4

A geometric framework for multiway correspondence analysis

4.1 Introduction

Correspondence Analysis (CA) [13, 47, 49, 55, 56, 57, 70] is a well suited tool for the study of categorical data, which usually are stored in contingency tables[1]. In particular, this analysis technique makes possible the simultaneous visualization and interpretation of categories of two variables in a low dimension space. CA is the Principal Component Analysis (PCA), see Section II.III, of a contingency table with a specific metric and consequently it inherits the PCA three possible natures. Indeed, the PCA of a matrix has an algebraic, a geometric and a statistical guise simultaneously. A two variables table is algebraically represented by a matrix and from the algebraic viewpoint PCA looks for the best approximation of this matrix at a prescribed rank. As explained in [41], PCA relies on SVD for determining the best matrix approximation at a prescribed rank. Geometrically speaking, a two variable table is associated with two point clouds, i.e., the sets of row and the column vectors of the original matrix. These vectors are the coordinates of the different variables in the full dimensional space. Therefore, geometrically speaking, PCA looks for a subspace of low dimension which minimizes the projection error of the two original point clouds, as described in [119]. Then, in a statistical context, the table entries represent observations of two variables and the statistical purpose of PCA is maximizing the variance, identifying a small amount of linear independent variable combinations. As PCA, also CA looks for the best approximation at a given rank or equivalently for the optimal subspace of low dimension, but taking into account a specific weight metric equivalent to the classical one. Particularly, a barycentric relation links the two point clouds when projected in the low dimension subspace found by CA [94]. This link justifies the simultaneous interpretation and visualization of the two point clouds.

When there are more than two variables in the contingency table, i.e., the data are

[1] In Statistics a contingency table is a table whose (i, j) entry is the frequency of the combination of the i-th category of the first variable with the j-th category of the second variable.

organized in a multiway table, it is convenient to move from matrix analysis techniques to tensor ones. As well underlined in [148], it is still possible to investigate multiway data through matrix methods, matricizing the data, but the matricization introduces necessarly a variable coupling, making some variable interactions harder to identify. On the other side, analysis procedures based on tensor decomposition algorithms have the advantage of considering each variable independently and clearly show all the variable links. For example in the Tucker model, presented in Section 1.3.1, the strength of the variable correlation is expressed by the core tensor coefficients. Classically, PCA is extended to tensor data by the Tucker model and its realization through HOSVD, see Algorithm 1. The algebraic extension of CA to MultiWay Correspondence Analysis (MWCA) follows naturally and it has been studied in [29, 44, 87, 89]. However the further investigation of the geometric nature of tensor MWCA was missing in the literature. The focus of this chapter is investigating the geometric viewpoint in CA in the tensor framework. Indeed we propose to fulfil this gap by proving the algebraic relation and the geometric interpretation of the point clouds associated with each mode of a multiway contingency table.

The remainder of this chapter is organized as follows. The first section presents the classical CA theory and then the MWCA with a focus on the geometric viewpoint, which is our theoretical contribution to the MWCA literature. In conclusion to this part, we compare the CA and the MWCA on two test examples, previously studied in [58]. The second section is centred on the analysis of the Malabar multiway contingency table [6]. We apply the MWCA technique, which is known as the most suitable investigation tool for frequency data, using the theoretical results to interpret the dataset.

4.2 Correspondence Analysis

After recalling briefly the CA theory, we describe the construction of point clouds from multiway data, one per mode, and we link them together. More precisely, we show that each point cloud is associated with each mode of a tensor in MWCA in the same way that a point cloud is associated with either rows or columns of a matrix in CA. This theoretical work justifies the possibility of visualizing and interpreting simultaneously point clouds constructed in MWCA from multiway data. To prove the existence of a barycentric link between point clouds attached to each tensor mode and generalizing the correspondence between rows and columns for contingency tables, we pass through three steps. Firstly we express the link in a classical Euclidean space, after we generalize to any Euclidean space and finally we construct the MWCA metric, stating the barycentric relation.

4.2.1 Matrix case

This section describes briefly the main steps of CA and its main result, i.e., the barycentric relation linking row and columns point clouds, see [94].

Let $F \in \mathbb{R}_+^{n_1 \times n_2}$ be a contingency table, whose entry $F(i,j)$ represents the relative[2] frequency of the i-th category for the first variable and the j-th category for the second one. As already stated, geometrically speaking, the matrix F is associated with two point clouds, i.e., two sets $\mathcal{R} = \{r_1, \dots, r_{n_1}\}$ and $\mathcal{C} = \{c_1, \dots, c_{n_2}\}$ where $r_i \in \mathbb{R}^{n_2}$ is the i-th row of F and similarly $c_j \in \mathbb{R}^{n_1}$ is the j-th column of F. The vectors r_i and c_j represent the components of the i-th category of the first variable in the space $\mathbb{R}^{n_2}$ and the j-th category of the second variable in the space $\mathbb{R}^{n_1}$ respectively. Henceforth we denote row and column marginals, norms and spaces by R and C respectively, and not by 1 and 2, to avoid any confusion with the classical meaning of norm-1 and norm-2, used in the previous chapters.

In the CA framework, we assume that the row space $\mathbb{R}^{n_2}$ and the column one $\mathbb{R}^{n_1}$ are endowed with two norms defined from the row and column marginals. Let $f_{\mathrm{R}} \in \mathbb{R}_+^{n_1}$ be the row marginal vector, whose i-th entry is

$$f_{\mathrm{R}}(i) = \sum_{j=1}^{n_2} r_i(j) = \sum_{j=1}^{n_2} F(i,j).$$

Similarly the column marginal is $f_{\mathrm{C}} \in \mathbb{R}_+^{n_2}$ defined as

$$f_{\mathrm{C}}(j) = \sum_{i=1}^{n_1} c_j(i) = \sum_{i=1}^{n_1} F(i,j).$$

Then the row norm $||\cdot||_{\mathrm{R}} : \mathbb{R}^{n_2} \to \mathbb{R}_+$ and the column norm $||\cdot||_{\mathrm{C}} : \mathbb{R}^{n_2} \to \mathbb{R}_+$ are defined as

$$||x||_{\mathrm{R}} = ||D_{\mathrm{R}}^{-1} x|| \qquad \text{and} \qquad ||y||_{\mathrm{C}} = ||D_{\mathrm{C}}^{-1} y||$$

where $D_{\mathrm{R}} = \mathrm{diag}(\sqrt{f_{\mathrm{R}}})$ and $D_{\mathrm{C}} = \mathrm{diag}(\sqrt{f_{\mathrm{C}}})$. The matrices D_{R}^{-1} and D_{C}^{-1} induce two isometries on the row space $(\mathbb{R}^{n_2}, ||\cdot||_{\mathrm{R}})$ and the column one $(\mathbb{R}^{n_1}, ||\cdot||_{\mathrm{C}})$, since they are both SPD matrices. Indeed, the functions

$$\nu_{\mathrm{R}} : (\mathbb{R}^{n_2}, ||\cdot||_{\mathrm{R}}) \to (\mathbb{R}^{n_2}, ||\cdot||) \quad \text{and} \quad \nu_{\mathrm{C}} : \ (\mathbb{R}^{n_1}, ||\cdot||_{\mathrm{C}}) \to (\mathbb{R}^{n_1}, ||\cdot||)$$

such that $\nu_{\mathrm{R}}(x) = D_{\mathrm{R}}^{-1} x$ and $\nu_{\mathrm{C}}(y) = D_{\mathrm{C}}^{-1} y$ are isometries, as they preserve the norm, that is $||\nu_{\mathrm{R}}(x)|| = ||D_{\mathrm{R}}^{-1} x|| = ||x||_{\mathrm{R}}$ and similarly for ν_{C}. The pair of matrices $(D_{\mathrm{R}}^{-1}, D_{\mathrm{C}}^{-1})$ enables us to endow a metric on the matrix space $\mathbb{R}^{n_1 \times n_2}$. Let $||\cdot||_M : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}_+$ be a metric such that $||X||_M = ||D_{\mathrm{R}}^{-1} X D_{\mathrm{C}}^{-1}||$. We denote by $\mathcal{S}_M$ the matrix space $\mathbb{R}^{n_1 \times n_2}$ endowed with the metric norm $||\cdot||_M$, while by $\mathcal{S}$ the same matrix space with the standard Euclidean norm. The bijection $\nu : \mathcal{S}_M \to \mathcal{S}$ defined as $\nu(X) = D_{\mathrm{R}}^{-1} X D_{\mathrm{C}}^{-1}$ is an isometry among $\mathcal{S}_M$ and $\mathcal{S}$, since it preserve the norm, that is $||\nu(X)|| = ||X||_M$. Remark that the action of ν can be seen as the combined action of ν_{C} and ν_{C} on the row and columns of an element of $\mathcal{S}_M$ respectively.

The geometrical aim of CA is finding a low dimension space $\mathbb{R}^s$ with $s \ll \min\{n_1, n_2\}$ which minimizes the projection error of the two point clouds $\mathcal{R}$and $\mathcal{C}$, so that they can be

[2]any contingency tables with absolute frequencies can be transformed into a relative frequency one

simultaneously visualized and interpreted easily. From the algebraic viewpoint this means finding the best approximation of a given contingency table at a prescribed rank in $\mathcal{S}_M$, thus it is not possible to directly apply the SVD to $F \in \mathcal{S}_M$. To retrieve the best low rank approximation, we transport $F \in \mathcal{S}_M$ through the isometry ν into the Euclidean space $\mathcal{S}$, where we can apply SVD. So given the contingency table F belonging to the space $\mathcal{S}_M$, let $X \in \mathcal{S}$ be its image through the isometry ν, i.e., $X = \nu(F)$, then we consider the reduced SVD of X

$$X = U\Sigma V^\top$$

with $U \in \mathbb{O}(n_1 \times r)$, $V \in \mathbb{O}(n_2 \times r)$ and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ with $r = \mathrm{rank}(X)$. The principal component for the row point clouds are given by the columns of $Y_{\mathrm{R}} = U\Sigma$ in a subspace of dimension r of $\mathbb{R}^{n_2}$ endowed with the canonical Euclidean norm, while the column principal coordinates are $Y_{\mathrm{C}} = V\Sigma$ in a subspace of dimension r of $\mathbb{R}^{n_1}$ still with the canonical Euclidean norm. More precisely from a geometric viewpoint the (i, h) element of Y_{R} represents the coordinate of the i-th category of the row variable along the h-th principal axis. Similarly $Y_{\mathrm{R}}(j, h)$ is the coordinate of the j-th category of the column variable along the h-th principal axis for $h \in \{1, \ldots, r\}$. As consequence the projections of the row and the column point clouds live in the subspaces of the same dimension r. They are usually simultaneously displayed and interpreted, because a barycentric relation links these principal components in the Euclidean space selected by CA. More specifically, let the $W_{\mathrm{R}} = D_{\mathrm{R}} Y_{\mathrm{R}}$ be the scaled principal components of the row variable in the subspace of dimension r of $(\mathbb{R}^{n_2}, ||\cdot||_{\mathrm{R}})$ and similarly $W_{\mathrm{C}} = D_{\mathrm{C}} Y_{\mathrm{C}}$ represents the columns principal components in the subspace of dimension r of $(\mathbb{R}^{n_1}, ||\cdot||_{\mathrm{C}})$, then it holds

$$Z_{\mathrm{R}} = D_{\mathrm{R}}^{-2} F Z_{\mathrm{C}} \Sigma^{-1} \qquad \text{and} \qquad Z_{\mathrm{C}} = D_{\mathrm{C}}^{-2} F^\top Z_{\mathrm{R}} \Sigma^{-1}$$

where the scaled principal components are $Z_i \in \mathbb{R}^{n_i \times r}$ with $Z_i = D_i^{-2} W_i$ for $i \in \{\mathrm{R}, \mathrm{C}\}$. These relations state that the row scaled principal components in $(\mathbb{R}^{n_2}, ||\cdot||_{\mathrm{R}})$ are actually the scaled barycentre of the column scaled principal components in $(\mathbb{R}^{n_1}, ||\cdot||_{\mathrm{C}})$ and vice-versa. Indeed component-wise, the previous relation writes

$$Z_{\mathrm{R}}(i, h) = \frac{1}{\sigma_h} \sum_{j=1}^{n_2} \frac{F(i, j)}{f_{\mathrm{R}}(i)} Z_{\mathrm{C}}(j, h) \qquad \text{and} \qquad Z_{\mathrm{C}}(j, h) = \frac{1}{\sigma_h} \sum_{i=1}^{n_1} \frac{F(i, j)}{f_{\mathrm{C}}(j)} Z_{\mathrm{R}}(i, h), \tag{4.1}$$

which is a barycentric relation scaled by the h-th singular value, since the sum of weights of the principal components sum is equal to 1, i.e.,

$$\sum_{j=1}^{n_2} \frac{F(i, j)}{f_1(i)} = \sum_{i=1}^{n_1} \frac{F(i, j)}{f_2(j)} = 1$$

by the marginal definition. CA relies on this relation for justifying the simultaneous visualization and interpretation of the two contingency table variables. We refer to [94] for further details on the barycentric relation linking the principal components in the CA framework.

4.2.2 Tensor case

Starting from the extension of principal component analysis to tensors with HOSVD, cf. Algorithm 1, we associate a point cloud with each mode, and show the existence of an algebraic link between them. The aim of this section is providing a geometrical interpretation of the point cloud relations algebraically proved in the tensor framework. For sake of simplicity, we first prove a link between the point clouds in the standard Euclidean 3-order tensor space, and then we generalize to d-order tensors. This structure choice is kept throughout this theoretical section. At the end of each section we present the geometric interpretation of the stated results. We naturally attach a point cloud to each matricization of 3-order tensor, see Definition 1.2.3. Then we search the optimal projection of each point cloud defined from the mode matricization in low dimension space. Finally, we show how their coordinates are linked and we extend this result to general Euclidean spaces of d-order tensors.

Let $\mathbf{x} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a tensor of order d and let $(U_\mu)_\mu \in \mathbb{O}(n_\mu \times r_\mu)$ be the Tucker decomposition basis obtained from the HOSVD algorithm at multi-linear rank $r = (r_1, \ldots, r_d)$. The tensor $\mathbf{x}$ is expressed as

$$\mathbf{x} = \sum_{i_1,\ldots,i_d=1}^{r_1,\ldots,r_d} \mathbf{c}(i_1, \ldots, i_d) u^1_{i_1} \otimes \cdots \otimes u^d_{i_d} \tag{4.2}$$

with $\mathbf{c} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ being the HOSVD core tensor and $u^\mu_{i_\mu} \in \mathbb{R}^{n_\mu}$ the i_μ-th column of U_μ. The condition for Equation (4.2) to be a decomposition is $r_\mu = \operatorname{rank}(X^{(\mu)})$ for $\mu \in \{1, \ldots, d\}$, otherwise we have an approximation of the given tensor. To keep the notation plain, we denote by $\mathbf{x}$ both the approximation and the decomposition of the given tensor, when its nature is clear enough from the context. Let Σ_μ be the diagonal singular value matrix of the matricization of $\mathbf{x}$ with respect to mode μ. For simplicity, the i_μ-th diagonal element of Σ_μ is denoted by $\sigma^{(\mu)}_{i_\mu}$ for every $i_\mu \in \{1, \ldots, r_\mu\}$ and for every $\mu \in \{1, \ldots, d\}$. The principal component of mode μ is $Y_\mu \in \mathbb{R}^{n_\mu \times r_\mu}$ defined as $Y_\mu = U_\mu \Sigma_\mu$ for each $\mu \in \{1, \ldots, d\}$.

4.2.2.1 Principal components in the canonical Euclidean space

For sake of simplicity and clarity, we assume d equal to 3. The following proposition states a relation linking the three sets of principal components.

Proposition 4.2.1. *Let $\mathbf{x}$ be a tensor of $\mathbb{R}^{n_1 \times n_2 \times n_3}$ and let $\mathbf{c}$ be its HOSVD core at multi-linear rank r. Let Y_μ be the principal components of mode μ. If $r_\mu = \operatorname{rank}(X^{(\mu)})$ for $\mu \in \{1, 2, 3\}$, then*

$$\begin{aligned} Y_1 &= X^{(1)}(Y_3 \otimes_K Y_2)(B^{(1)})^\top, \\ Y_2 &= X^{(2)}(Y_3 \otimes_K Y_1)(B^{(2)})^\top, \\ Y_3 &= X^{(3)}(Y_2 \otimes_K Y_1)(B^{(3)})^\top, \end{aligned}$$

with $\mathbf{b} = (\Sigma_1^{-1}, \Sigma_2^{-1}, \Sigma_3^{-1})\mathbf{c}$.

Proof. We start the proof for the first mode principal components. Let $\mathbf{x}$ be decomposed in the HOSVD basis as in Equation (4.2), i.e.,

$$\mathbf{x} = \sum_{i,j,k=1}^{r_1,r_2,r_3} \mathbf{c}(i,j,k) u_i^1 \otimes u_j^2 \otimes u_k^3.$$

As previously stated in Corollary 1.2.4 the matricization of $\mathbf{x}$ with respect to mode 1 in the Tucker basis is expressed with the Kronecker product, defined in 1.2.4, as

$$X^{(1)} = \sum_{i=1}^{r_1} u_i^1 \otimes \left(\sum_{j,k=1}^{r_2,r_3} \mathbf{c}(i,j,k) u_k^3 \otimes_{\mathrm{K}} u_j^2 \right). \tag{4.3}$$

The PCA of $X^{(1)}$ is

$$X^{(1)} = Y_1 V_1^\top = \sum_{i=1}^{r_1} \sigma_i^{(1)} u_i^1 \otimes v_i^1 \tag{4.4}$$

with v_i^1 the i-th column of $V_1 \in \mathbb{O}(n_2 n_3 \times r_1)$ and $\sigma_i^{(1)} u_i^1$ the i-th column of $Y_1 \in \mathbb{R}^{n_1 \times r_1}$. By comparing Equations (4.3) and (4.4) for a fixed index i, we get

$$\sigma_i^{(1)} u_i^1 \otimes v_i^1 = u_i^1 \otimes \left(\sum_{j,k=1}^{r_2,r_3} \mathbf{c}(i,j,k) u_k^3 \otimes_{\mathrm{K}} u_j^2 \right).$$

Since Σ_1 is invertible by the hypothesis $r_1 \leq \operatorname{rank}(A^{(1)})$, we identify v_i^1 with a linear combination of the Kronecker product of u_j^2 and u_k^3 scaled by $\sigma_i^{(1)}$ as

$$v_i^1 = \frac{1}{\sigma_i^{(1)}} \sum_{j,k=1}^{r_2,r_3} \mathbf{c}(i,j,k) u_k^3 \otimes_{\mathrm{K}} u_j^2. \tag{4.5}$$

Notice that the j-th and k-th column of $Y_2 = U_2 \Sigma_2$ and $Y_3 = U_3 \Sigma_3$ are $\sigma_j^{(2)} u_j^2$ and $\sigma_k^{(3)} u_k^3$ respectively. So introducing in Equation (4.5) the singular values $\sigma_j^{(2)}$ and $\sigma_k^{(3)}$, we express the i-th column of V_1 as a linear combination of the Kronecker product of the j-th and k-th column of Y_2 and Y_3, i.e.,

$$\begin{aligned} v_i^1 &= \frac{1}{\sigma_i^{(1)}} \sum_{j,k=1}^{r_2,r_3} \mathbf{c}(i,j,k) \frac{\sigma_j^{(2)} \sigma_k^{(3)}}{\sigma_j^{(2)} \sigma_k^{(3)}} u_k^3 \otimes_{\mathrm{K}} u_j^2 \\ &= \sum_{j,k=1}^{r_2,r_3} \frac{\mathbf{c}(i,j,k)}{\sigma_i^{(1)} \sigma_j^{(2)} \sigma_k^{(3)}} y_k^3 \otimes_{\mathrm{K}} y_j^2 \\ &= \sum_{j,k=1}^{r_2,r_3} \mathbf{b}(i,j,k)\, y_k^3 \otimes_{\mathrm{K}} y_j^2 \end{aligned} \tag{4.6}$$

with $\mathbf{b} = (\Sigma_1^{-1}, \Sigma_2^{-1}, \Sigma_3^{-1})\mathbf{c}$, y_j^2 and y_k^3 the j-th and k-th column of Y_2 and Y_3 respectively. Remark that $Y_3 \otimes_{\mathrm{K}} Y_2$ is a matrix of $n_2 n_3$ rows and $r_2 r_3$ columns whose ℓ-th column is $y_k^3 \otimes_{\mathrm{K}} y_j^2$ with $\ell = \overline{jk}$ for every $j \in \{1, \ldots, r_2\}$, $k \in \{1, \ldots, r_3\}$ and $\ell \in \{1, \ldots, r_2 r_3\}$,

see Definition 1.2.3. The tensor $\mathbf{b}$ matricized with respect to mode 1 is a matrix of r_1 rows and $r_2 r_3$ columns, whose $(i, \overline{jk})$-th element is $\mathbf{b}(i, j, k)$ for all $j \in \{1, \ldots, r_2\}$, $k \in \{1, \ldots, r_3\}$. So the sum in the right-hand side of Equation (4.6) can be expressed as the matrix-product between $Y_3 \otimes_{\mathrm{K}} Y_2$ and tensor $\mathbf{b}$ matricized with respect to mode 1 as

$$V_1 = (Y_3 \otimes_{\mathrm{K}} Y_2)(B^{(1)})^\top. \tag{4.7}$$

Multiplying Equation (4.4) on the right by V_1 yields $Y_1 = X^{(1)} V_1$. Therefore multiplying Equation (4.7) by the matricization of $\mathbf{x}$ with respect to mode 1, the principal component Y_1 is expressed as linear combination of the Kronecker product of principal components Y_2 and Y_3, i.e.,

$$Y_1 = X^{(1)} V_1 = X^{(1)}(Y_3 \otimes_{\mathrm{K}} Y_2)(B^{(1)})^\top.$$

The left and right hand side of this last equation proved right the proposition thesis. The other relations follow straightforwardly from this one, permuting the indices coherently. □

From an algebraic view this first proposition shows that the principal components of each mode can be expressed as a linear combination of the principal components of the two other modes. However as pointed out in the preliminary section, principal components can be seen from different viewpoints.

From a geometric viewpoint a point cloud $\mathcal{X}_\mu$ is attached to the matricization with respect to mode μ of tensor $\mathbf{x}$ for $\mu \in \{1, 2, 3\}$. Indeed the i_μ-th row of $X^{(\mu)}$ represents the coordinates of the i_μ-th element of mode μ point cloud living in the space $\mathbb{R}^{n_{\neq\mu}}$ where $n_{\neq\mu} = \frac{n_1 n_2 n_3}{n_\mu}$. Given a multi-linear rank r, we reformulate the problem of the Tucker approximation as a problem of dimension reduction. Indeed, we look for the subspace of $\mathbb{R}^{n_{\neq\mu}}$ of dimension r_μ which minimizes in norm the projection of point cloud $\mathcal{X}_\mu$ on it. This problem is solved with the HOSVD algorithm, which provides three orthogonal basis (U_1, U_2, U_3) of the corresponding subspaces. Therefore the i_μ-th row of $Y_\mu = U_\mu \Sigma_\mu$ represents the coordinates of the i_μ-th element of $\mathcal{X}_\mu$ projected into the subspace of $\mathbb{R}^{n_{\neq\mu}}$ of dimension r_{i_μ} for $\mu \in \{1, 2, 3\}$. The Proposition 4.2.1 result is interpreted geometrically as each point cloud living in the linear subspace built from the Kronecker product of the other two. Proposition 4.2.1 is generalized to the d-case as follows.

Proposition 4.2.2. *Let $\mathbf{x}$ be a tensor of $\mathbb{R}^{n_1 \times \cdots \times n_d}$ and let $\mathbf{c}$ be its HOSVD core at multi-linear rank r. Let Y_μ be the principal components of mode μ. If $r_\mu \leq$ rank$(A^{(\mu)})$ for $\mu \in \{1, \ldots, d\}$, then*

$$Y_\mu = X^{(\mu)}(Y_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} Y_{\mu+1} \otimes_{\mathrm{K}} Y_{\mu-1} \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} Y_1)(B^{(\mu)})^\top$$

with $\mathbf{b} = (\Sigma_1^{-1}, \ldots, \Sigma_d^{-1})\mathbf{c}$ for every $\mu \in \{1, \ldots, d\}$.

Proof. The proof is very similar to that of Proposition 4.2.1, so we give only the main steps. Let start by the first mode principal components. The 1-mode matricization of $\mathbf{x}$ is expressed in the Tucker basis with the Kronecker product, see Corollary 1.2.4, as

$$X^{(1)} = \sum_{i_1=1}^{r_1} u_{i_1}^1 \otimes \left(\sum_{i_2,\ldots,i_d=1}^{r_2,\ldots,r_d} \mathbf{c}(i_1, \ldots, i_d) u_{i_d}^d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} u_{i_2}^2 \right). \tag{4.8}$$

The PCA of $X^{(1)}$ leads to

$$X^{(1)} = Y_1 V_1^\top = \sum_{i_1=1}^{r_1} \sigma_{i_1}^{(1)} u_{i_1}^1 \otimes v_{i_1}^1 \tag{4.9}$$

with $v_{i_1}^1$ the i_1-th unitary column of $V_1 \in \mathbb{O}(n_{\neq 1} \times r_1)$ where $n_{\neq 1} = \prod_{\mu=2}^d n_\mu$ and $\sigma_{i_1}^{(1)} u_{i_1}^1$ the i_1-th column of $Y_1 \in \mathbb{R}^{n_1 \times r_1}$. Comparing Equations (4.8) and (4.9) for a fixed index i_1, we identify $v_{i_1}^1$ with a linear combination of the Kronecker product of $u_{i_\mu}^\mu$ for $\mu \in \{2, \dots, d\}$ scaled by $\sigma_{i_1}^{(1)}$ as

$$v_{i_1}^1 = \frac{1}{\sigma_{i_1}^{(1)}} \sum_{i_2,\dots,i_d=1}^{r_2,\dots,r_d} \mathbf{c}(i_1, \dots, i_d) u_{i_d}^d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} u_{i_2}^2. \tag{4.10}$$

Introducing in Equation (4.10) the singular values $\sigma_{i_\mu}^{(\mu)}$, we express the i_1-th column of V_1 as a linear combination of the Kronecker product of the i_μ-th column of Y_μ for every $i_\mu \in \{2, \dots, d\}$ as

$$\begin{aligned} v_{i_1}^1 &= \sum_{i_2,\dots,i_d=1}^{r_2,\dots,r_d} \frac{\mathbf{c}(i_1, \dots, i_d)}{\sigma_{i_1}^{(1)} \sigma_{i_2}^{(2)} \dots \sigma_{i_d}^{(d)}} \sigma_{i_2}^{(2)} \dots \sigma_{i_d}^{(d)} u_{i_d}^d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} u_{i_2}^2 \\ &= \sum_{i_2,\dots,i_d=1}^{r_2,\dots,r_d} \mathbf{b}(i_1, \dots, i_d) y_{i_d}^d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} y_{i_2}^2 \end{aligned} \tag{4.11}$$

with $\mathbf{b} = (\Sigma_1^{-1}, \dots, \Sigma_d^{-1})\mathbf{c}$ and $y_{i_\mu}^\mu$ the i_μ-th column of Y_μ for $\mu \in \{2, \dots, d\}$. Thanks to the correspondence between Kronecker product and matricization, the right-hand-side of Equation (4.11) is expressed as the matrix-product between $Y_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} Y_2$ and tensor $\mathbf{b}$ matricized with respect to mode 1 and transposed as

$$V_1 = (Y_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} Y_2)(B^{(1)})^\top. \tag{4.12}$$

Multiplying Equation (4.9) on the right by V_1 yields $Y_1 = X^{(1)} V_1$ and replacing V_1 by its expression of Equation (4.12), it finally follows

$$Y_1 = X^{(1)} V_1 = X^{(1)} (Y_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} Y_2)(B^{(1)})^\top.$$

This last equation proves the thesis. The other relations follow straightforwardly from this one, permuting the indices coherently. □

4.2.2.2 Extension to a generic Euclidean space for d-order tensors

As discussed in Section 1.3.1, the minimization problem faced with the HOSVD algorithm is expressed by the Frobenious norm, induced by an inner product. The standard inner product is defined by the identity matrix. However, whatever SPD matrix induces an inner product and the associated metric norm on a vector space, which is therefore isomorphic to the standard Euclidean space. We emphasize in this section the role of

the metric on the relationships between point clouds, using this isomorphic relationship between Euclidean spaces with different inner products [44].

Let $\mathcal{S}$ be the Euclidean tensor space $\mathbb{R}^{n_1 \times \cdots \times n_d}$ endowed with the standard inner product. Given d SPD matrices M_μ of size n_μ, then the space $\mathcal{S}_M$ is the tensor space $\mathbb{R}^{n_1 \times \cdots \times n_d}$ endowed with the metric norm defined by $||\mathbf{x}||_M = ||(M_1, \ldots, M_d)\mathbf{x}||$. As in the matrix case discussed in Section 4.2.1, we move the $\mathbf{f}$ belonging to $\mathcal{S}_M$ into $\mathcal{S}$, thanks to the isometry defined by $\nu : \mathcal{S}_M \to \mathcal{S}$, such that $\mathbf{x} = \nu(\mathbf{f}) = (M_1, \ldots, M_d)\mathbf{f}$. Remark that ν acts independently on each mode by construction.

Let now $\mathbf{x} \in \mathcal{S}$ be the HOSVD approximation of $\nu(\mathbf{f})$ at multi-linear rank r and let $(U_\mu)_\mu \in \mathbb{O}(n_\mu \times r_\mu)$ be the associated basis. Σ_μ is the singular value matrix of the μ-matricization $X^{(\mu)}$ and define $Y_\mu = U_\mu \Sigma_\mu$ the principal components of mode μ for tensor $\mathbf{x}$ for each $\mu \in \{1, \ldots, d\}$. The principal components of $\mathbf{f}$ in the tensor space $\mathcal{S}_M$ are the given by $W_\mu \in \mathbb{R}^{n_\mu \times r_\mu}$ such that

$$W_\mu = M_\mu^{-1} Y_\mu. \tag{4.13}$$

In the following proposition, we link the sets of principal components in the Euclidean space $\mathcal{S}_M$. As previously, for the sake of clarity the result is first proved for $d = 3$ and afterwards generalized to whatever order d.

Proposition 4.2.3. *Let $\mathbf{f}$ be a tensor in $\mathcal{S}_M$ and let $\mathbf{x}$ be its image through the isometry ν in the standard tensor space $\mathcal{S}$. Let $\mathbf{c}$ be the HOSVD core tensor of $\mathbf{x}$ at multi-linear rank r such that $r_\mu \leq rank(X^{(\mu)})$ for $\mu \in \{1, 2, 3\}$. Then for W_μ the principal component of mode μ of $\mathbf{f}$ in the metric tensor space $\mathcal{S}_M$ it holds*

$$\begin{aligned} W_1 &= F^{(1)}(M_3^2 W_3 \otimes_{\mathrm{K}} M_2^2 W_2)(B^{(1)})^\top, \\ W_2 &= F^{(2)}(M_3^2 W_3 \otimes_{\mathrm{K}} M_1^2 W_1)(B^{(2)})^\top, \\ W_3 &= F^{(3)}(M_2^2 W_2 \otimes_{\mathrm{K}} M_1^2 W_1)(B^{(3)})^\top, \end{aligned}$$

with $\mathbf{b} = (\Sigma_1^{-1}, \Sigma_2^{-1}, \Sigma_3^{-1})\mathbf{c}$.

Proof. This result comes straightforwardly from Proposition 4.2.1 proof by introducing the metrics matrices. For completeness, we illustrate the proof focusing on the link for the principal components of the first mode. Let $(U_\mu)_{\mu=1,2,3}$ be the HOSVD basis of $\mathbf{x}$ at multi-linear rank r with $U_\mu \in \mathbb{O}(n_\mu \times r_\mu)$. Let $Y_\mu = U_\mu \Sigma_\mu$ be the principal components of mode μ for tensor $\mathbf{x}$, where Σ_μ is the singular values matrix of $X^{(\mu)}$. Proposition 4.2.1 yields

$$Y_1 = X^{(1)}(Y_3 \otimes_{\mathrm{K}} Y_2)(B^{(1)})^\top \tag{4.14}$$

with $\mathbf{b} = (\Sigma_1^{-1}, \Sigma_2^{-1}, \Sigma_3^{-1})\mathbf{c}$. Notice that $\mathbf{x} = \nu(\mathbf{f}) = (M_1, M_2, M_3)\mathbf{f}$. So thanks to the thesis of Corollary 1.2.4, we express $X^{(1)}$ in function of $F^{(1)}$ as

$$X^{(1)} = M_1 F^{(1)}(M_3^\top \otimes_{\mathrm{K}} M_2^\top)$$

and replacing it into (4.14), it gets

$$Y_1 = M_1 F^{(1)}(M_3 \otimes_{\mathrm{K}} M_2)(Y_3 \otimes_{\mathrm{K}} Y_2)(B^{(1)})^\top \tag{4.15}$$

where the transposition symbol is discarded on M_μ since they are SPD matrices. Remarking that $Y_\mu = M_\mu W_\mu$ from the W_μ definition in Equation (4.13), substituting it in the Equation (4.15) we obtain

$$M_1 W_1 = M_1 F^{(1)}(M_3 \otimes_{\mathrm{K}} M_2)(M_3 W_3 \otimes_{\mathrm{K}} M_2 W_2)(B^{(1)})^\top. \tag{4.16}$$

Since M_1 is invertible, from the previous equation it follows the thesis. The other relations follow straightforwardly from this proof, permuting the indices coherently. □

This result is easily generalized to d-order tensors as follows.

Proposition 4.2.4. *Let* $\mathbf{f}$ *be a tensor in* $\mathcal{S}_M$ *and let* $\mathbf{x}$ *be its image through the isometry* ν *in the standard tensor space* $\mathcal{S}$*. Let* $\mathbf{c}$ *be the HOSVD core tensor of* $\mathbf{x}$ *at multi-linear rank* r *such that* $r_\mu \leq rank(X^{(\mu)})$ *for for* $\mu \in \{1, \ldots, d\}$*. Then for* W_μ *the principal component of mode* μ *of* $\mathbf{f}$ *in the metric tensor space* $\mathcal{S}_M$ *it holds*

$$W_\mu = F^{(1)}(M_d^2 W_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_{\mu+1}^2 W_{\mu+1} \otimes_{\mathrm{K}} M_{\mu-1}^2 W_{\mu-1} \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_1^2 W_1)(B^{(\mu)})^\top$$

with $\mathbf{b} = (\Sigma_1^{-1}, \ldots, \Sigma_d^{-1})\mathbf{c}$ *for every* $\mu \in \{1, \ldots, d\}$*.*

Proof. The proof is a direct a generalization of Proposition 4.2.3 proof. Let $Y_\mu \in \mathbb{R}^{n_\mu \times r_\mu}$ be the principal components of $\mathbf{x}$, then from Proposition 4.2.2, it writes

$$Y_\mu = X^{(\mu)}(Y_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} Y_{\mu+1} \otimes_{\mathrm{K}} Y_{\mu-1} \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} Y_1)(B^{(\mu)})^\top \tag{4.17}$$

with $\mathbf{b} = (\Sigma_1^{-1}, \ldots, \Sigma_d^{-1})\mathbf{c}$. Since $\mathbf{x}$ is the image of $\mathbf{f}$ through the isometry ν, that is $\mathbf{x} = \nu(\mathbf{f}) = (M_1, \ldots, M_d)\mathbf{f}$, the mode μ matricization of $\mathbf{X}$ is expressed in function of the SPD matrices M_μ as

$$X^{(\mu)} = M_\mu F^{(\mu)}(M_d^\top \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_{\mu+1}^\top \otimes_{\mathrm{K}} M_{\mu-1}^\top \cdots M_1^\top). \tag{4.18}$$

Replacing $X^{(\mu)}$ in Equation (4.17) by this last equation, it follows that

$$Y_\mu = M_\mu F^{(\mu)}(M_d^\top Y_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_{\mu+1}^\top Y_{\mu+1} \otimes_{\mathrm{K}} M_{\mu-1}^\top Y_{\mu-1} \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_1^\top Y_1)(B^{(\mu)})^\top$$

thanks to the distributive property of the Kronecker product. Remarking that M_μ are SPD matrices, i.e., they coincide with their transpose, the previous equation gets

$$Y_\mu = M_\mu F^{(\mu)}(M_d Y_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_{\mu+1} Y_{\mu+1} \otimes_{\mathrm{K}} M_{\mu-1} Y_{\mu-1} \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_1 Y_1)(B^{(\mu)})^\top$$

and because of the relation between classical principal components and metric ones, that is $Y_\mu = M_\mu W_\mu$, we have

$$M_\mu W_\mu = M_\mu F^{(\mu)}(M_d^2 W_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_{\mu+1}^2 W_{\mu+1} \otimes_{\mathrm{K}} M_{\mu-1}^2 W_{\mu-1} \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_1^2 W_1)(B^{(\mu)})^\top.$$

Since M_μ is invertible, from this last equation the thesis follows. □

4.2.2.3 Geometric view for the MultiWay Correspondence Analysis

In this conclusive section, we transport the previous results in the MultiWay Correspondence Analysis context. We firstly clarify the metric of the Euclidean space where we set our problem. Then we make explicit the point cloud relation in this particular framework. As final outcome, we are able to prove the correspondence between the point clouds attached to each mode.

In accordance with the correspondence analysis framework, we consider a d-way contingency table $\mathbf{t} \in \mathbb{N}^{n_1 \times \cdots \times n_d}$. The first step for performing CA is scaling $\mathbf{t}$ by the sum of all its components setting a new relative frequency tensor $\mathbf{f} \in \mathbb{R}_+^{n_1 \times \cdots \times n_d}$

$$\mathbf{f} = \frac{1}{\sum_{i_1,\ldots,i_d=1}^{n_1,\ldots,n_d} \mathbf{t}(i_1,\ldots,i_d)} \mathbf{t}.$$

We first clarify the tensor space we will work with. Let f_μ be the marginal of mode μ, i.e., the vector whose components are the sums of the slices of mode μ for all $\mu \in \{1,\ldots,d\}$. For example the i_1-th element of f_1 is

$$f_1(i_1) = \sum_{i_2,\ldots,i_d=1}^{n_2,\ldots,n_d} \mathbf{f}(i_1,\ldots,i_d) \qquad \text{for all} \qquad i_1 \in \{1,\ldots,n_1\}.$$

There is no loss of generality in assuming that $f_\mu(i_\mu) > 0$ for every $\mu \in \{1,\ldots,d\}$. Indeed the i_μ-th component of f_μ is equal to zero if and only if all the entries of the i_μ slice of $\mathbf{f}$ with respect to mode μ are zeros, i.e., $\mathbf{f}(i_1,\ldots,i_\mu,\ldots,i_d) = 0$ for every $i_k \in \{1,\ldots,n_k\}$ and $k \in \{1,\ldots,\mu-1,\mu+1,\ldots,d\}$. If there exists a mode μ and an index i_μ such that its marginal component is zero, that is $f_\mu(i_\mu) = 0$, then the i_μ category of the μ variable has zero frequency, or differently said, this variable case never arrives. Consequently it does not impact the global analysis and it is discarded.

Define $D_\mu = \operatorname{diag}(\sqrt{f_\mu}) \in \mathbb{R}^{n_\mu \times n_\mu}$ for each $\mu \in \{1,\ldots,d\}$ and assume that $\mathbf{f}$ belongs to $\mathbb{R}^{n_1 \times \cdots \times n_d}$ endowed by the metric induced by the matrices $(D_1^{-1},\ldots,D_d^{-1})$, since D_μ^{-1} is SPD for every $\mu \in \{1,\ldots,d\}$. We denote by $\mathcal{S}_M$ this Euclidean space and by $\mathcal{S}$ the tensor space $\mathbb{R}^{n_1 \times \cdots \times n_d}$ endowed with the standard inner product. Under this assumption, let ν be the isometry between the spaces $\mathcal{S}_M$ and $\mathcal{S}$ and let $\mathbf{x} = \nu(\mathbf{f}) = (D_1^{-1},\ldots,D_d^{-1})\mathbf{f}$. The general element of tensor $\mathbf{x}$ is written as

$$\mathbf{x}(i_1,\ldots,i_d) = \frac{\mathbf{f}(i_1,\ldots,i_d)}{\sqrt{f_1(i_1)\ldots f_d(i_d)}}.$$

Performing the HOSVD over tensor $\mathbf{x}$ at multi-linear rank r leads to a new orthogonal basis $(U_\mu)_{\mu=1,\ldots,d}$, a core tensor $\mathbf{c}$ and principal components $Y_\mu = U_\mu \Sigma_\mu$ for every $\mu \in \{1,\ldots,d\}$ in the standard tensor space. Focusing on the principal components $W_\mu = D_\mu Y_\mu$ of tensor $\mathbf{f}$ in $\mathcal{S}_M$, Proposition 4.2.4 entails

$$W_\mu = F^{(1)}(M_d^2 W_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_{\mu+1}^2 W_{\mu+1} \otimes_{\mathrm{K}} M_{\mu-1}^2 W_{\mu-1} \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} M_1^2 W_1)(B^{(\mu)})^\top$$

where $B^{(\mu)}$ is the matricization of $\mathbf{b} = (\Sigma_1^{-1}, \ldots, \Sigma_d^{-1})\mathbf{c}$ and $M_\mu = D_\mu^{-1}$ for $\mu \in \{1, \ldots, d\}$. Let $Z_\mu = D_\mu^{-2} W_\mu$ be the *principal components scaled* by the marginal inverse for $\mu \in \{1, \ldots, d\}$ in tensor space $\mathcal{S}_M$. Henceforth, we denote by $z_{i_\mu}^{\mu}$ the i_μ-th row of Z_μ. Now we prove that each component of vector $z_{i_\mu}^{\mu}$ can be expressed as a scaling factor times the barycenter of the linear combinations of the other scaled principal component rows. We assume d equal to 3 to facilitate the comprehension of the following proof.

Proposition 4.2.5. *Let* $\mathbf{f}$ *be a tensor in the tensor space* $\mathcal{S}_M$ *endowed with the norm induced by the inner product matrices* $D_\mu = diag(\sqrt{f_\mu})$ *with* f_μ *the* μ *mode marginal of* $\mathbf{f}$ *for* $\mu \in \{1, 2, 3\}$. *Let* $Z_\mu \in \mathbb{R}^{n_\mu \times r_\mu}$ *be the scaled principal components for tensor* $\mathbf{f}$ *of mode* μ *in* $\mathcal{S}_M$. *If* $r_\mu = rank(F^{(\mu)})$ *for every* $\mu \in \{1, 2, 3\}$, *then*

$$
\begin{aligned}
z_i^1(\ell) &= \frac{1}{\sigma_\ell^{(1)}} \sum_{j,k=1}^{n_2,n_3} \sum_{m,p=1}^{r_2,r_3} \frac{\mathbf{f}(i,j,k)}{f_1(i)} z_k^3(p) z_j^2(m) \mathbf{b}_1(\ell, m, p), \\
z_j^2(m) &= \frac{1}{\sigma_m^{(2)}} \sum_{i,k=1}^{n_1,n_3} \sum_{\ell,p=1}^{r_1,r_3} \frac{\mathbf{f}(i,j,k)}{f_2(j)} z_k^3(p) z_i^1(\ell) \mathbf{b}_2(\ell, m, p), \\
z_k^3(p) &= \frac{1}{\sigma_p^{(3)}} \sum_{i,j=1}^{n_1,n_2} \sum_{\ell,m=1}^{r_1,r_2} \frac{\mathbf{f}(i,j,k)}{f_3(k)} z_i^1(\ell) z_j^2(m) \mathbf{b}_3(\ell, m, p),
\end{aligned}
$$

where $\mathbf{b}_\mu = \Sigma_\mu \times_\mu \mathbf{b}$ *from Proposition 4.2.3.*

Proof. We describe the proof for the i-th row of Z_1 with $i \in \{1, \ldots, n_1\}$. From Proposition 4.2.3, under the CA metric choice, it follows that the principal components of $\mathbf{f}$ in the tensor space $\mathcal{S}_M$ satisfy the relation

$$
W_1 = F^{(1)}(D_3^{-2} W_3 \otimes_{\mathrm{K}} D_2^{-2} W_2)(B^{(1)})^\top = F^{(1)}(Z_3 \otimes_{\mathrm{K}} Z_2)(B^{(1)})^\top.
$$

Multiplying on the left this last equation by D_1^{-2}, we obtain

$$
Z_1 = D_1^{-2} W_1 = D_1^{-2} F^{(1)}(Z_3 \otimes_{\mathrm{K}} Z_2)(B^{(1)})^\top
$$

and since $B^{(1)} = \Sigma_1^{-1} B_1^{(1)}$, it gets

$$
Z_1 = D_1^{-2} F^{(1)}(Z_3 \otimes_{\mathrm{K}} Z_2)(\Sigma_1^{-1} B_1^{(1)})^\top. \tag{4.19}
$$

Making explicit the ℓ-th component of z_i^1, the i-th row of Z_1, from Equation (4.19), we have

$$
\begin{aligned}
z_i^1(\ell) = Z_1(i,\ell) &= \sum_{\overline{jk}=1}^{n_2 n_3} \sum_{m,p=1}^{r_2,r_3} \Big(D_1^{-2} F^{(1)}\Big)(i, \overline{jk}) \Big(Z_3 \otimes_{\mathrm{K}} Z_2\Big)(\overline{jk}, \overline{mp}) \Big(\Sigma_1^{-1} B_1^{(1)}\Big)(\ell, \overline{mp}) \\
&= \sum_{j,k=1}^{n_2,n_3} \sum_{m,p=1}^{r_2,r_3} \frac{\mathbf{f}(i,j,k)}{f_1(i)} Z_3(k,p) Z_2(j,m) \frac{\mathbf{b}_1(\ell, m, p)}{\sigma_\ell^{(1)}} \\
&= \frac{1}{\sigma_\ell^{(1)}} \sum_{j,k=1}^{n_2,n_3} \sum_{m,p=1}^{r_2,r_3} \frac{\mathbf{f}(i,j,k)}{f_1(i)} z_k^3(p) z_j^2(m) \mathbf{b}_1(\ell, m, p)
\end{aligned}
$$

by the definition of z_j^2 and z_k^3 for every $i \in \{1,\dots,n_1\}$, $j \in \{1,\dots,n_2\}$, $k \in \{1,\dots,n_3\}$ and $\ell \in \{1,\dots,r_1\}$. This final equation can be read as a mutual barycenter relation scaled by the inverse of the corresponding singular value. Indeed there is a list of weight terms which sum to 1, i.e.,

$$\sum_{j,k=1}^{n_2,n_3} \frac{\mathbf{f}(i,j,k)}{f_1(i)} = \frac{f_1(i)}{f_1(i)} = 1$$

times a linear combination expressed through $\mathbf{b}_1$ of z_j^2 and z_k^3. Moving back to the geometric perspective, the Proposition 4.2.5 states that the scaled coordinates of a point cloud correspond to the barycenter of the other two point cloud scaled coordinates.

The two remaining barycentric relations follow from this proof, permuting coherently the indices. □

Correspondence in CA refers to the correspondence of point cloud coordinates through the scaled barycentric relation. We proved that this well known relation in matrix framework is holding also in the tensor one, through HOSVD. Therefore we propose to refer to it as *MultiWay Correspondence Analysis from HOSVD*.

This final result is extended and verified straightforwardly for d-order tensors.

Proposition 4.2.6. *Let $\mathbf{f}$ be a tensor in the tensor space $\mathcal{S}_M$ endowed with the norm induced by the inner product matrices $D_\mu = diag(\sqrt{f_\mu})$ with f_μ the μ mode marginal of $\mathbf{f}$ for every $\mu \in \{1,\dots,d\}$. Let $Z_\mu \in \mathbb{R}^{n_\mu \times r_\mu}$ be the scaled principal components for tensor $\mathbf{f}$ of mode μ in $\mathcal{S}_M$. If $r_\mu = rank(F^{(\mu)})$ for $\mu \in \{1,\dots,d\}$, then*

$$z_{i_\mu}^{\mu}(\ell_\mu) = \frac{1}{\sigma_{\ell_\mu}^{(\mu)}} \sum_{\substack{\eta=1 \\ \eta\neq\mu}}^{d} \sum_{i_\eta=1}^{n_\eta} \sum_{\ell_\eta=1}^{r_\eta} \frac{\mathbf{f}(i_1,\dots,i_d)}{f_\mu(i_\mu)} z_{i_d}^{d}(\ell_d)\cdots z_{i_{\mu+1}}^{\mu+1}(\ell_{\mu+1}) z_{i_{\mu-1}}^{\mu-1}(\ell_{\mu-1})\cdots z_{i_1}^{1}(\ell_1)\mathbf{b}_\mu(\ell_1,\dots,\ell_d)$$

where $\mathbf{b}_\mu = \Sigma_\mu \times_\mu \mathbf{b}$ from Proposition 4.2.4.

Proof. From Proposition 4.2.4, under the CA metric choice, the principal components of the first mode of $\mathbf{f}$ in $\mathcal{S}_M$ are expressed as

$$W_1 = F^{(1)}(Z_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} Z_2)(B^{(1)})^\top.$$

If the previous equation is multiplied by equation by D_1^{-2} and it has $B^{(1)}$ replaced by the equivalent $\Sigma_1^{-1}B_1^{(1)}$, it gets

$$Z_1 = D_1^{-2}F^{(1)}(Z_d \otimes_{\mathrm{K}} \cdots \otimes_{\mathrm{K}} Z_2)(\Sigma_1^{-1}B_1^{(1)})^\top. \tag{4.20}$$

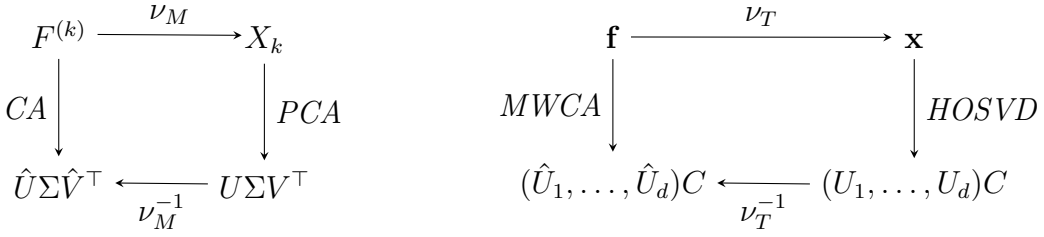
Making explicit the ℓ_1-th component of $z_{i_1}^1$, the i_1-th row of Z_1, from Equation (4.20), it follows the thesis, i.e.,

$$z_{i_1}^{1}(\ell_1) = \frac{1}{\sigma_{\ell_1}^{(1)}} \sum_{i_2,\dots,i_d=1}^{n_2,\dots,n_d} \sum_{\ell_2,\dots,\ell_d=1}^{r_2,\dots,r_d} \frac{\mathbf{f}(i_1,\dots,i_d)}{f_1(i_1)} z_{i_d}^{d}(\ell_d)\cdots z_{i_2}^{2}(\ell_2)\mathbf{b}_1(\ell_1,\dots,\ell_d).$$

The proof for the other modes follows directly from this one permuting coherently the indices. □

4.2.2.4 Examples

To validate our theoretical results and to highlight the benefit of a tensor approach, MWCA is compared with the classical CA, applied to the same data reorganized as a matrix. Since the SPD matrices defining the isometry of CA and MWCA are different, there exist a discrepancy between CA and MWCA. We rely on Figure 4.1 and the elements therein to compare the two approaches. Let $\mathbf{f} \in \mathbb{R}_+^{n_1 \times \cdots \times n_d}$ store the data relative frequencies as a tensor, then their matrix representation is given by $F^{(k)}$ the k-matricization mode of $\mathbf{f}$, once mode k has been selected. The MWCA is performed on $\mathbf{f}$, while CA is performed on $F^{(k)}$. Let $X^{(k)}$ be the k-matricization of $\nu_T(\mathbf{f})$ where ν_T is the MWCA isometry, as depicted in Figure 4.1B, similarly X_k is the outcome of CA isometry ν_M applied to $F^{(k)}$, as in Figure 4.1A.



(A) Scheme for CA of k-th matricization

(B) Scheme for MWCA

Figure 4.1 – Correspondence analysis: matrix versus tensor.

Since the two approaches are different, the SPD matrices defining the isometries ν_T and ν_M transporting $\mathbf{f}$ and $F^{(k)}$ differ and so do $X^{(k)}$ and X_k. To estimate this discrepancy between the two decomposed objects we define the relative error $e(X^{(k)}, X_k)$ as

$$e(X^{(k)}, X_k) = \frac{||X^{(k)} - X_k||}{||X^{(k)}||}. \tag{4.21}$$

If the value of $e(X^{(k)}, X_k)$ is small, we expect the two approach to lead to similar interpretations. Conversely, if the error value is significantly large, the outcome may suggest a different interpretation for the variable category interactions.

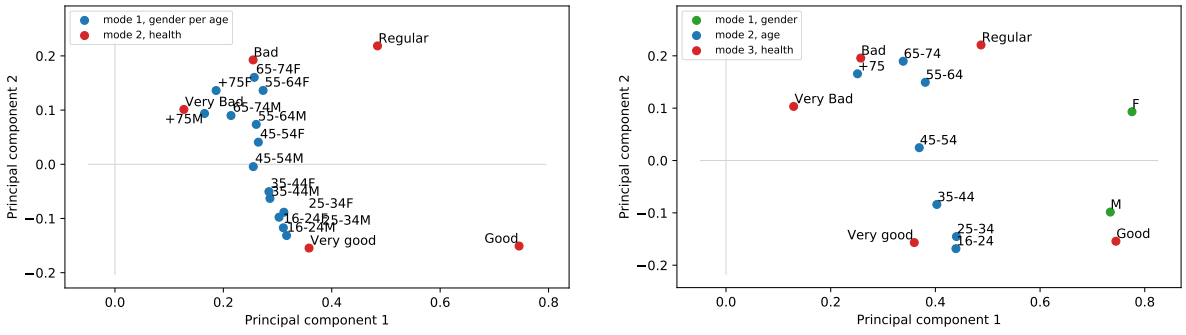
The SVD and HOSVD, over which CA and MWCA relay respectively, provide an orthogonal basis for the matrix or the decomposed tensor, which is unique up to an orthogonal rotation. As proposed in [18] and [19], we orient these new basis selecting as leading direction the one where the majority of the data points out. To perform CA and MWCA we used `python 3.6.9` and the library `TensorLy 0.6.0`, see [85].

The following example is based on [24]. We select this dataset, which is already analysed with CA in [24], to show that the multiway method results are coherent with CA ones.

	Males					Females				
Age group	**Very good**	**Good**	**Regular**	**Bad**	**Very bad**	**Very good**	**Good**	**Regular**	**Bad**	**Very bad**
16-24	145	402	84	5	3	98	387	83	13	3
25-34	112	414	74	13	2	108	395	90	22	4
35-44	80	331	82	24	4	67	327	99	17	4
45-54	54	231	102	22	6	36	238	134	28	10
55-64	30	219	119	53	12	23	195	187	53	18
65-74	18	125	110	35	4	26	142	174	63	16
+75	9	67	65	25	8	11	69	92	41	9

Table 4.1 – Data from the Spanish National Health Survey of 1997, see [24].

Example 4.2.7. In [58] data are reported from a survey over 6731 people of both genders, from 16 to over 75 years old, who were asked to evaluate their health status. Then the answers were organized in a 3-way table with dimensions $n_1 = 2$, $n_2 = 7$ and $n_3 = 5$, as in Table 4.1. On mode 1 we have the two genders: male, 'M', and female, 'F' and on the second mode there are 7 age groups. On the last mode we set 5 health grades, from 'Very Good' to 'Very bad'. The $(1, 1, 1)$ entry of the multiway table is the number of men between 16 and 24 years old who judge 'Very good' their health status. Let $\mathbf{f} \in \mathbb{R}_+^{n_1 \times n_2 \times n_3}$ represent the data relative frequencies in tensor format for MWCA. CA is realized over A_3 the matricization of $\mathbf{f}$ with respect to mode 3, i.e., 'health grade', where on the row we set the health categories and on the columns all the possible combinations of ages and gender. In Figure 4.2A we recover the correspondence analysis of [58], while in 4.2B we display multiway correspondence analysis with the first two columns of $Y_\mu = U_\mu \Sigma_\mu$, defined in Section 4.2.2.3 for $\mu \in \{1, 2, 3\}$.



(A) Mode 1 correspondence analysis.

(B) Multiway correspondence analysis.

Figure 4.2 – CA versus MWCA for data of [24].

As in [58], both plots in Figure 4.2 display a gradient for the age categories: they distribute from the youngest in the bottom to the oldest near the top. The 5 health categories are at the extremes of this gradient, the better health levels at the bottom, worst

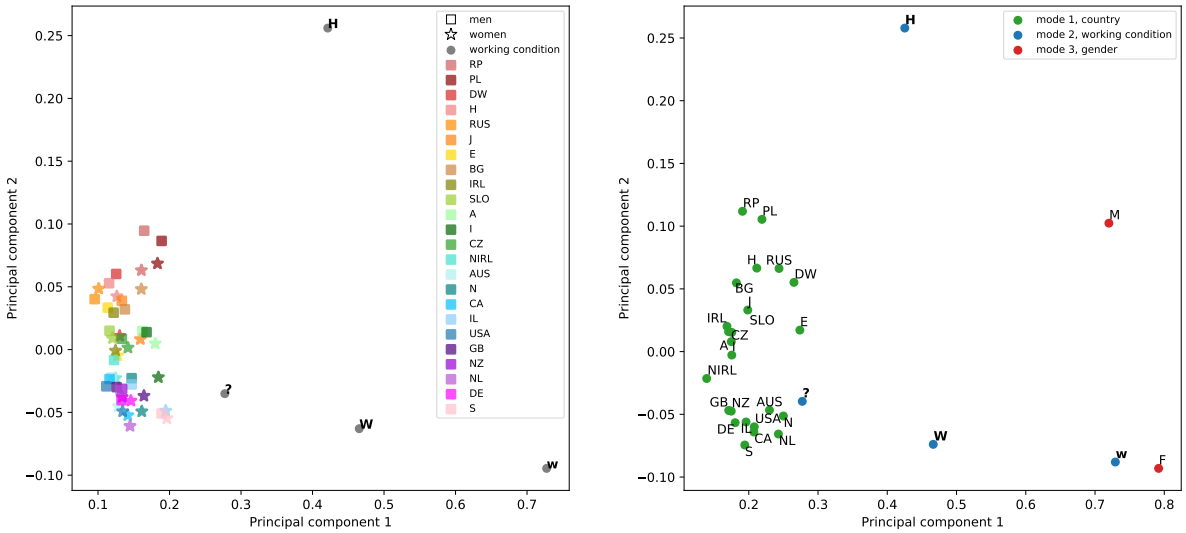
ones at the top. From this we infer that the health evaluation decreases for increasing ages. This phenomenon is highlighted by both the techniques, even if it is clearer in Figure 4.2B, thanks to the possibility of analysing the variables separately. Taking into account the gender in Figure 4.2B, let us remark that the male dot is close to the 'Good' one. Thanks to the barycentric relation, we know that the closer the points the more correlated they are. So men appear to provide optimistic evaluations of their health status. Women dot stands aside from the other, slightly closer to worst health status, suggesting that female health judging is balanced between the age classes, with a minor inclination toward pessimistic evaluation. In Figure 4.2A for increasing ages, male and female dots tend to increase their distances, suggesting that when men and women age, they tend respectively to be more optimistic and pessimistic in evaluating their health status. MWCA plot seems to point out that men are more optimistic than how much pessimistic women are. In contrast from Figure 4.2B we cannot conclude easily which age category presents the biggest difference in health evaluation for the two genders, information that can be inferred from 4.2A. Lastly comparing the health point cloud in Figures 4.2A and 4.2B, we observe that their principal coordinates are almost the same and we explain it in terms of relative error. Indeed for this data-set the relative error defined in Equation (4.21) is

$$e(X^{(3)}, X_3) = \frac{||X^{(3)} - X_3||}{||X^{(3)}||} \approx 0.035$$

is quite small.

To further investigate the tensor approach in comparison with the matrix one, we analyse the dataset available on [23]. In [58], these data are fully studied and described in their interaction through a matrix approach.

Example 4.2.8. In [58] a case study is given by the data coming from the Internation Social Survey Programme of 1994. The authors focus on the survey question if a woman with a schoolchild should work full-time, part-time or stay at home. The participants were also given the possibility of expressing a doubtful position. In total 33590 people of both genders took part in the survey, from 24 countries, among which East and West Germany, Great Britain and Northern Ireland were considered respectively separated. The results relative to the working conditions of a schoolchild mother were organized in a 3-way contingency table [23]. On mode 1 we set the $n_1 = 24$ countries ('***AUS***' for Australia, '***DW***' for West Germany, '***DE***' for East Germany, '***GB***' for Great Britain, '***NIRL***' for Northern Ireland, '***USA***' for United States of America, '***A***' for Austria, '***H***' for Hungary, '***I***' for Italy, '***IRL***' for Ireland, '***NL***' for Netherlands, '***N***' for Norway, '***S***' for Sweden, '***CZ***' for Czechoslovakia, '***SLO***' for Slovenia, '***PL***' for Poland, '***BG***' for Bulgaria, '***Rus***' for Russia, '***NZ***' for New Zealand, '***CA***' for Canada, '***RP***' for Philippines, '***IL***' for Israel, '***J***' for Japan and '***E***' for Spain). On the second mode, there are the four possible answers to the specific question about the working condition, i.e., full time working denoted by 'W', part-time working by 'w', stay at home by 'H' and unsure by question mark, that is $n_2 = 4$. The last mode has size $n_3 = 2$ for the two genders: male, 'M', and female, 'F'. As



(A) Mode 2 correspondence analysis.

(B) Multiway correspondence analysis.

Figure 4.3 – CA versus MWCA for data of [24].

consequence, the $(1,1,1)$-th entry of the multiway table is the number of Australian men who judge that a woman with a schoolchild should work full-time. As in the previous example, we assume that $\mathbf{f} \in \mathbb{R}_+^{n_1 \times n_2 \times n_3}$ represents the data relative frequencies in tensor format for MWCA, while we perform CA over $F^{(2)}$ the matricization of $\mathbf{f}$ with respect to the working condition mode, i.e., the second one. Differently said, the columns of $F^{(2)}$ corresponds to men and women from different country, while the row to the four possible survey answers. The result of CA is displayed in Figure 4.3A, matching the outcome of the correspondence analysis performed in [58]. Figure 4.2B shows multiway correspondence analysis with the first two columns of $Y_\mu = U_\mu \Sigma_\mu$, defined in Section 4.2.2.3 for $\mu \in \{1,2,3\}$.

Comparing Figure 4.3A and 4.3B, it is clearly evident that the four survey answers about the working conditions have the same layout. The staying at home 'H' direction is almost orthogonal to the working one, outlined by part-time 'w' and full-time 'W' work. On the intersection among these two directions there is the question mark for the unsure answer. This strong similarity may be explained by the relative error among mode 2 MCA and MWCA. Indeed for this data-set the relative error defined in Equation (4.21) is

$$e(X^{(2)}, X_2) = \frac{||X^{(2)} - X_2||}{||X^{(2)}||} \approx 0.04$$

which is quite small. This working condition pattern is similar, but not identical to the one found in [58]. Indeed in [58], staying at home and working, in both the full and part time possibilities, form orthogonal direction, but in [58] the unsure reply dot is in the middle between 'w' and 'W'. We may believe that this difference is due to the pretreatment (particular scaling and centring) considered in [58] and not in our analysis. From the

viewpoint of the countries, in both the Figure 4.2 plots a bottom-up gradient emerges for the countries. On the bottom, with cold colours for Figure 4.3A, we find liberal nations as Sweden, West Germany, Netherlands, New Zealand or Great Britain, while on the top there are more conservative states as Philippines, Poland or West Germany, according to the social structure of 1994. This gradient appears clearly in the analysis of [58]. As in Example 4.2.7, the separation among of country and gender in Figure 4.3B eases the perception and interpretation of the liberal to conservative bottom-up gradient. Moreover in Figure 4.3A, it is not as immediately clear as in Figure 4.3B the global influence of the gender in answering to the considered survey question. Indeed in the MWCA output, the women dot is clearly aligned along the working line, while men dot is set in the midway between the staying at home and working direction. As previously stated, the barycentric relation implies that the nearer the dots the more correlated they are likely to be. As consequence, since the women dot is closer to the part-time 'w' dot, we may infer that on a global scale women in 1994 believed on average that a schoolchild mother should also have a job, with a preference for a part-time one. On the other side, the midway position of the men dot may suggest that male opinions uniformly distribute among the two extrema possible answers, working or staying at home. This potential conclusion, stated in [58], is not as conveniently achieved from MCA output in Figure 4.3A. On the other side, if the analysis is focused on the gender effect per country, i.e., how much different are on average men and women answers to the survey, the results are visible only in Figure 4.3A. For example, checking the pink square and star in Figure 4.3A, we may conclude on average Philippines people reply similarly to the survey neglecting the gender. On the contrary, Russian men, displayed by the light orange square, seem to favour on average the stay at home answer, if they are compared with the Russian women, i.e., the light orange star in Figure 4.3A.

Exactly as in Example 4.2.7, the choice of the matrix or tensor approach for the multiway table analysis strongly depends on the interpretation aims. If the final purpose is highlighting specific differences among variable category combinations, then the matrix approach should be privileged. However if more global variable category interactions are searched, then the tensor approach is probably the most convenient, thanks to the complete separation of variable categories and the consequent displaying clarity.

4.3 Application: the Malabar dataset

This second main section is devoted to the original analysis of the Malabar dataset [6] a contingency multiway table. We briefly describe the data trying to get hints of a possible behaviour. Since the classical tensor data analysis literature [87] suggests to perform MCA or MWCA on categorical variables data, this dataset represents the ideal opportunity for applying the theory developed in Section 4.2. To highlight the advantages of the tensor approach, we interpret the dataset through MWCA and MCA, constructing and analysing the point clouds as in sections 4.3.3 and 4.2.1, as it was done for Example 4.2.7.

4.3.1 Data description

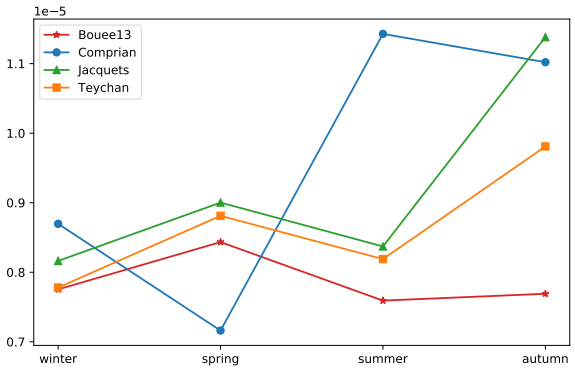
The data-set provided by the Malabar project (IFREMER, CNRS, INRAE, Labex COTE) [6] is a multiway contingency table of order 4. In this metabarcoding project, 32 water samples have been collected in Arcachon Bay at four locations (Bouee13, Comprian, Jacquets, Teychan), during four seasons, and at two positions in water column (pelagic and benthic). DNA has been extracted, and Operational Taxonomic Units (OTUs) have been built. OTU are expected to correspond to species and, without entering too much into details here, the question which motivates Malabar project is to understand or quantify the role of locations, seasons and water column positions in the diversity of protists. To address this question, a four way contingency table $\mathbf{t}$ has been built, where $\mathbf{t}(i, j, k, \ell)$ is the number of sequences collected at location j, in season ℓ and position k which belong to OTU i. The size of the contingency tensor is $n_1 = 3539$, $n_2 = 4$ $n_3 = 2$ and $n_4 = 4$. Henceforth we assume to work with the relative frequency data, i.e., with $\mathbf{f} = (1/t)\mathbf{t}$ where $t = \sum_{i,j,k,\ell=1}^{n_1,n_2,n_3,n_4} \mathbf{t}(i, j, k, \ell)$.

4.3.2 Average comparison and data preprocessing

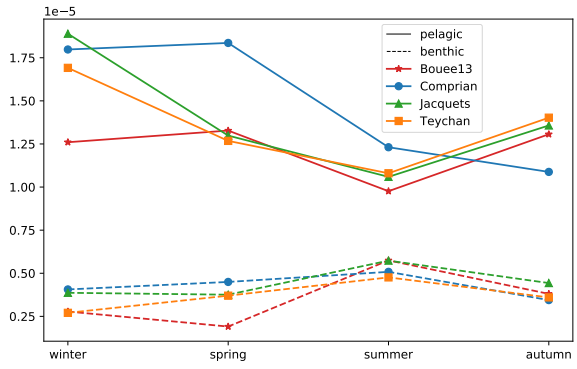
Before proceeding with the analysis of the dataset with tensor techniques, we investigate and eventually preprocess them. Since the focus of the data exploration is on the OTUs, we compute the average number of OTU sequences per location and per season, summing over the first OTU mode and third water column one. The average matrix is $\Xi \in \mathbb{R}^{n_2 \times n_4}$ such that

$$\Xi(j, \ell) = \frac{1}{n_1 n_3} \sum_{i,k=1}^{n_1,n_3} \mathbf{f}(i, j, k, \ell)$$

for every $j \in \{1, \ldots, n_2\}$ and $k \in \{1, \ldots, n_4\}$.



(A) Seasonal variation of the OTU average per location.



(B) Seasonal variation of the OTU average per location and water column position.

Figure 4.4 – Seasonal variation of the OTU average.

Figure 4.4A shows the variation of the number of OTU sequences in function of the

season for the four different locations. We observe that Jacquets and Teychan present the same average behaviour, with a significant increase in the average number of OTU between summer and autumn and a huge decrease from autumn to winter. In Comprian the average number of OTU grows between spring and summer and similarly to the previous two locations, it decreases significantly from autumn to winter. Finally, in Bouee13 the average number of OTU shows much lighter average variation, in a way similar to Jacquets and Teychan.

Since the water column position variable has only two categories, 'pelagic', i.e., close to the ocean surface, and 'benthic', i.e., close to the seabed, we study the average number of OTU in function of location, water column position and season, defining, $\boldsymbol{\xi} \in \mathbb{R}^{n_2 \times n_3 \times n_4}$ as

$$\boldsymbol{\xi}(j,k,\ell) = \frac{1}{n_1} \sum_{i=1}^{n_1} \mathbf{f}(i,j,k,\ell).$$

The variation of the average number of OTU in function of the season is displayed in Figure 4.4B for each location and for both the water column positions. Remarkably, the averages for the 'pelagic' and 'benthic' position are clearly clustered and present slightly opposite tendencies. Indeed for all the locations, in the 'benthic' position the average number of OTUs seems to stay almost constant throughout the year, with a perceptible increase only for Bouee13 from spring to summer and a minor decrease for all the locations from summer to autumn. On the 'pelagic' side, the average number of OTU decreases significantly from spring to summer and increases from summer to winter for all the location except Bouee13. This last location presents even in the 'pelagic' case lower fluctuations in the average number of OTU during the season cycle. Before drawing a conclusion, it is worthwhile computing the the standard deviation of the OTU defined as $\boldsymbol{\lambda} \in \mathbb{R}^{n_2 \times n_3 \times n_4}$ such that

$$\boldsymbol{\lambda}(j,k,\ell) = \sqrt{\frac{1}{n_1} \sum_{i=1}^{n_1} \Big(\mathbf{f}(i,j,k,\ell) - \boldsymbol{\xi}(j,k,\ell)\Big)^2}.$$

The mean $\Xi_P = \boldsymbol{\xi}_P(:,1,:)$ and the standard deviation $\Lambda_P = \boldsymbol{\lambda}_P(:,1,:)$ for the pelagic category are of order 10^{-5} and 10^{-4} respectively. On the other side, the orders of the benthic mean, $\Xi_B = \boldsymbol{\xi}(:,2,:)$, and of the benthic standard deviation, $\Lambda_B = \boldsymbol{\lambda}_B(:,2,:)$, are respectively 10^{-6} and 10^{-5}. Since the standard deviation is of one order greater than the mean for both the pelagic and benthic subdataset, an independent study of the pelagic and benthic subtensor of order 3 is not motivated.

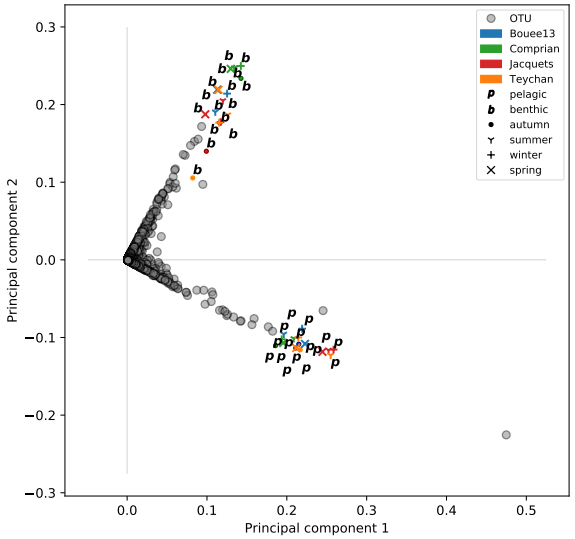
4.3.3 MultiWay Correspondence Analysis

The Malabar dataset is a multiway contingency table and thus according to the literature [87] the most suitable interpretation technique is MultiWay Correspondence Analysis or Multiple Correspondence Analysis, as previously mention in Section 4.3. On the basis of the result of Section 4.2, we perform MWCA on the Malabar dataset, constructing the point clouds associated with each mode of the multiway table and visualizing the first

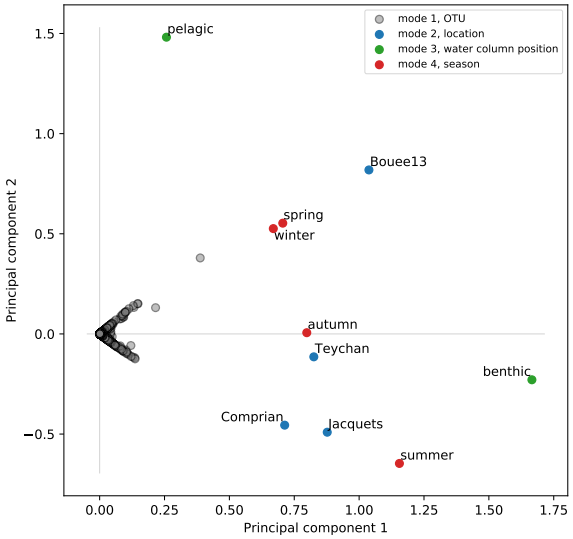
two columns of each mode principal components together in Figure 4.5 to obtain a more reliable interpretation. As for Examples 4.2.7 and 4.2.8, we perform also the canonical CA on the matricization with respect to mode 1 of the Malabar dataset, to emphasize which aspects each of the techniques best grasps.

In Figure 4.5A the result of CA performed over A_1, the matricization of frequency tensor data $\mathbf{f}$ with respect to mode 1, where the rows represent the different OTUs and the columns the locations, water column positions and seasons combined. Next to it, Figure 4.5B displays the result of MWCA over the tensor data. In the CA case OTU point cloud spreads over two orthogonal directions, led by the two positions in the water column. Similarly in MWCA plot the water column point cloud seem to organize and to affect the position and clustering of the others. Because of mode combination in Figure 4.5A, it is not evident which position in the water column affects which location or season. However thanks to the barycentric relation and the multiway approach, we can infer some relations among seasons, locations and water column positions from Figure 4.5B, where they are independently displayed. Indeed pelagic point appears to affect the OTU present during spring and winter time. Similarly benthic position probably drives the distribution of OTU in autumn and summer. A similar argument can be repeated for location point cloud. Indeed the barycentric relation suggests that the benthic position leads the distribution of OTU in Comprian, Jacquets and Teychan, while the pelagic dot influences Bouee13 dot. The barycentric relation enables us to interpret the interaction among season and location point clouds. In Figure 4.5B winter and spring seem to be more correlated with Bouee13 and similarly summer and autumn are probably more correlated with the remaining three locations. Analysing the OTU and the season point clouds, we observe that winter and spring pull out one orthogonal direction over which part of the OTU points spread. Similarly summer pulls out the other OTU orthogonal direction. The autumn point appears to be neutral with respect to the OTU point cloud distribution. Similarly if we study the OTU and location point cloud interaction, one orthogonal direction is driven by Bouee13, one by Comprian and Jaquets, while Teychan seems to have a small effect on the OTU point distribution.

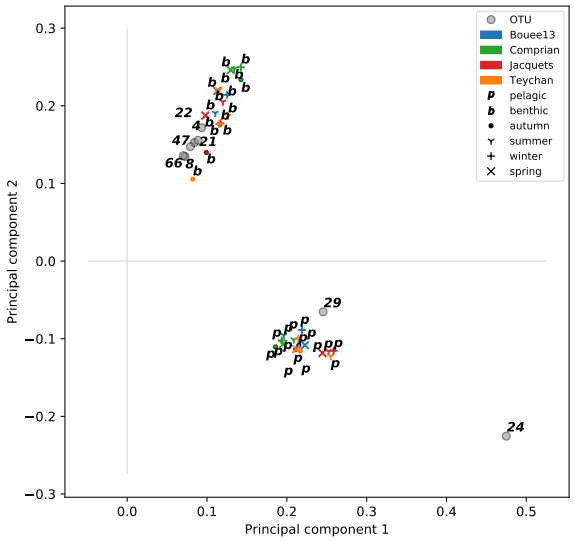
In Figures 4.5C and 4.5D, we filter the OTU by size, keeping just those with first principal components greater than 0.2 and the second greater than 0.12. Figure 4.5C confirms the idea of water-column position driving the OTU point cloud distribution. However in Figure 4.5B most of the OTU points are close to the origin. Indeed in this left plot we filtered 8 OTU which are probably more correlated with benthic or pelagic condition. Comparing the two filtered figures, we see that only 3 of the 8 selected OTU are common between them, i.e., OTU '22', '24' and '29'. These common OTU share the same correlations between the two methods. Indeed in both case, '22' is more related to benthic position, while '24' and '29' with pelagic one. Moreover most of the OTU in Figure 4.5D are concentrated around the origin. We may infer that the action of the water column point cloud is softened by the other modes, here independent, leading to a more balanced distribution of OTU. In conclusion notice that in MWCA the different point clouds are more spread than in the CA case, enabling a deeper analysis of the variables



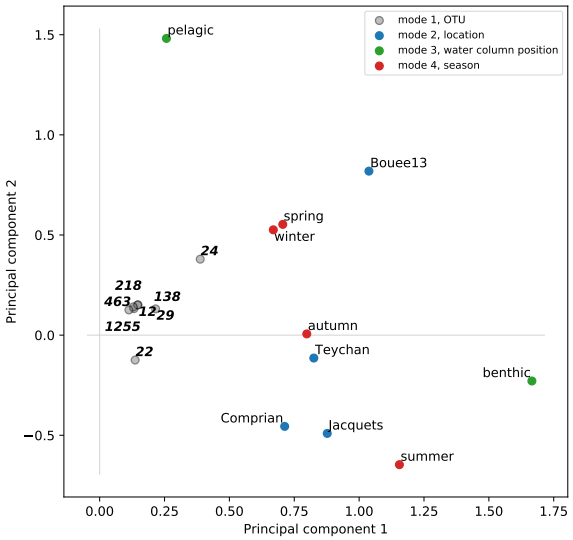
(A) Correspondence Analysis of mode 1.



(B) Multiway Correspondence Analysis.



(C) CA with filtered OTU.



(D) MWCA with filtered OTU.

Figure 4.5 – CA vs MWCA biplot of Arcachon Bay data-set.

interactions. It could be linked to the significant relative error from Equation (4.21)

$$e(X^{(1)}, X_1) = \frac{||X^{(1)} - X_1||}{||X^{(1)}||} \approx 0.14,$$

which may also explain why CA and MWCA figures present a flipped y-axis, even though the SVD and HOSVD basis have been oriented according to the same criterion. Indeed the mode combinations including benthic have positive y-components in Figure 4.5A, while benthic point has negative y-component in Figure 4.5B. However since it affects in the same way all the point clouds, the two methods lead to a coherent interpretation.

4.4 Concluding remarks

We devoted this chapter to the study of multiway contingency tables through tensor techniques. The focus is set on the MultiWay Correspondance Analysis (MWCA), that associates multiway contingency tables with point clouds. After describing the point clouds construction process, we proved the algebraic relations linking them and we proposed a geometric interpretation. To highlight the benefits and the potential drawbacks of a tensor approach compared to a classical matrix one, we presented two examples. Then we applied this technique together with other classical ones for analysing and interpreting an original multiway contingency table.

The first part, i.e., Sections 4.2.1 and 4.2.2, collects the theoretical content of the entire chapter. We started describing the classical theory of Correspondence Analysis (CA) in the usual matrix context. We showed how points clouds are attached to a contingency table and how the geometrical spaces, where these point clouds live, are defined in CA. As conclusive point, we stated in Equation (4.1) the known barycentric relation linking point cloud associated with rows and columns of a contingency table. Next we moved to the tensor framework. We recalled the classical way to associate a point clouds with each mode of a multiway contingency table and to construct their principal components. Our contribution was investigating the link of point cloud principal components. First we proved the existence of this relation in the real order-d tensor space endowed with the classical Euclidean metric. Then we extended this relation to the same space endowed with a whatever metric, induced by a set of d SPD matrices. Once set these preliminary results, we derived the barycentric relation in the MWCA case, introducing the MWCA metric. We have taken care to supplement each algebraic result with its geometric interpretation. In conclusion to Section 4.2 we presented two examples, both from [58], with the aim of confirming the consistency of MWCA with CA and of highlighting the main advantages of a tensor technique over a matrix one.

The second part focuses entirely on the original analysis of the Malabar dataset [6], a 4-way contingency table. We introduced initially the dataset, remarking that each of its element represents the number of DNA sequences extracted from a sample of water collected in a specific location inside the Arcachon Bay during a certain season and at a certain level in the water column. We proceeded with a preliminary data analysis,

investigating the variation over the season of the average number of microorganism DNA sequences per location, observing that each location seems to have a preferred seasonal pattern. Next we studied the seasonal average change in the number of microorganisms per location and per position in the water column, noticing that a clear distinction exists among seabed and surface microorganism numbers, for all the considered locations. This step already suggested that the position in the water column could play a key role in the analysis. The conclusive part of this second section is dedicated to the analysis of Malabar dataset with the MWCA. Especially we compare the MWCA interpretation with the mode one MCA of the tensor data. We remark here that MWCA simplifies significantly the data interpretation, justifying it as the preferred tool for this type of dataset.

Chapter 5

Tensor techniques and climate data

5.1 Introduction

Climate data are measurements of complex phenomena involving a huge number of interactions of many different variables [63, 64], among which two crucial ones are time and space. The Empirical Orthogonal Function (EOF) analysis is a widely used tool originally meant to separate and visualize time and space-related information from a climate dataset. In practice, the EOF analysis is a variation of the Principal Components Analysis, see II.III, suitably tuned for climate data. Thus, as PCA, the EOF analysis is also used as an exploratory technique to investigate the pattern captured in the data and as a dimensional reduction method. The EOF analysis is applied also to forecast, to compare models and observations [63].

The classical EOF analysis aims to capture the maximum variance of the data, represented by a matrix, expressing it as the product of two matrices, an orthogonal one, whose columns are said Empirical Orthogonal Functions (EOF) and an un-correlated one, whose columns are referred to as Principal Components (PC). The EOF matrix collects the space information, while the PC matrix the time one. The basic version of the EOF analysis has been improved and modified over the years, for example relaxing the constraint of the orthogonality; we refer the reader to [63, 64] for variations of the EOF analysis and other climate data exploration methods.

Climate data can also be mathematically modelled by tensors, as presented in [149] for example. Thus, also the two variants of the EOF analysis can be readily formulated in the tensor framework, but without a significant benefit from the computational viewpoint. However, if the climate data are high dimensional and made available in Tucker format 1.3.1, the tensor approach becomes more interesting. Under this strong assumption, we investigate how to retrieve the EOF analysis results from Tucker format data and when it could be beneficial. The results we present constitute a preliminary study of the EOF analysis.

The remainder of this chapter is organized as follows. Section 5.2 presents two variants of the classical EOF analysis: one based on the covariance matrix and one on the correlation matrix, both computed from matrix climate data. We underline the computational

possible simplifications for both these two mathematical procedures. In Section 5.3, we represent the climate data through the Tucker model and we show how to perform the EOF analysis benefiting from this compressed format. This tensor approach for EOF analysis is applied and compared with the classical matrix one in Section 5.4. More in detail, we briefly study the first EOF and PC obtained from the classical matrix analysis of a real climate dataset, namely `HadCRUT4_RecAI_tas_mon_188001-201512.nc`. This dataset is approximated also at different accuracies in Tucker format. Following the approach described in Section 5.3, we compute the EOF and the PC matrices, studying the relative error in function of the different accuracies. For completeness, we display and investigate also the first column of the Tucker decomposition of the dataset, relating it to the classical EOF results.

5.2 Empirical Orthogonal Function Analysis

This section introduces the classical EOF analysis. As already mentioned, the EOF analysis aims to separate the information carried in the data due to the time variables from those linked to the space variables. After describing in Section 5.2.1 the data structure and the pretreatments applied, we present two mathematical processes considered to compute the PC and the EOF matrices. For completeness, the computational aspects of the EOF analysis are briefly described.

To highlight the nature of the different mathematical objects, we add a subscript c to denote the centred data, a subscript s for the scaled data, and a subscript sc for the scaled and centred data to the bold Latin letter indicating the tensor storing the data. As it is conventional, when an array stores a sample average, the letter denoting it is overlined.

5.2.1 Data description and pretreatments

In this section, we present the general structure of data we will analyse henceforth. Let $\mathcal{S} \subset \mathbb{R}^3$ denote the Earth surface in this chapter. The function $f : \mathbb{R}_+ \times \mathcal{S} \to \mathbb{R}$ associates with each time value and each point of the Earth's surface the temperature or the pressure measured in that time-space point. In the climatology framework, the function f is usually referred to as *space-time field*. As the next step, the field is discretized into a three-dimensional array. To achieve this aim, we define a grid of size $p_1 \times p_2$ (i.e., $p = p_1 p_2$ points), over the Earth surface denoting the discrete latitude variable by θ_j for $j \in \{1, \dots, p_1\}$ and the discrete longitude by ϕ_k for $k \in \{1, \dots, p_2\}$. Let t_i be the discrete-time variable for $i \in \{1, \dots, n\}$, then $\{(t_i, \theta_j, \phi_k)\}_{(i,j,k)=(1,1,1)}^{(n,p_1,p_2)}$ is the time-space grid used to discretize the function f. Indeed, the discretization of f over the time-space chosen grid is the tensor $\mathbf{f} \in \mathbb{R}^{n \times p_1 \times p_2}$ of order 3 such that

$$\mathbf{f}(i, j, k) = f(t_i, \theta_j, \phi_k) \tag{5.1}$$

for all $i \in \{1, \dots, n\}, j \in \{1, \dots, p_1\}$ and $k \in \{1, \dots, p_2\}$.

Henceforth, we assume data to be purged by the seasonal (or external) cyclic patterns. Let $X \in \mathbb{R}^{n\times p}$ with $p = p_1p_2$ denote the tensor $\mathbf{f}$ matricized with respect to mode one, i.e., $X = \mathbf{F}^{(1)}$. This choice couples together the two space variables. Before proceeding with the classical EOFs analysis, the data are preprocessed, removing from X the average in time. The time average $\overline{x} \in \mathbb{R}^p$ is

$$\overline{x}(h) = \frac{1}{n}\sum_{i=1}^{n} X(i,h) \qquad \text{for every} \qquad h \in \{1,\dots,p\}.$$

We compute $X_c \in \mathbb{R}^{n\times p}$ centring each column of X by its average, i.e.,

$$X_c = X - \mathbb{1}_n \otimes \overline{x} \tag{5.2}$$

where $\mathbb{1}_n$ denotes the vector of ones in $\mathbb{R}^n$. In details each element of X_c is of the form $X_c(i,h) = X(i,h) - \overline{x}(h)$ for every $i \in \{1,\dots,n\}$ and $h \in \{1,\dots,p\}$. In climatology matrix X_c is called the *anomaly field*.

5.2.2 EOFs and PCs computation

This section describes how to compute the Empirical Orthogonal Functions (EOFs) and the Principal Components (PCs) to perform the EOFs analysis [63, 64].

As previously mentioned, the purpose of the EOFs analysis is decomposing the anomaly field, dividing the time and the space contributions, i.e., expressing the factorization of X_c as

$$X_c(t_i, s_k) = \sum_{k=1} v_k(s_j)c_k(t_i)$$

for every discrete time t_i and coupled space s_j point. The EOFs analysis states the problem as finding a linear combination of coupled grid points that maximizes the variance, see Definition II.II.ii. This linear combination will prove how to split time and space discrete variables. Mathematically, we seek for a vector $a^* \in \mathbb{R}^p$ such that $X_c a^*$ has the maximal variance, i.e.

$$a^* = \underset{||a||=1}{\operatorname{argmax}} \mathbb{V}\mathrm{ar}(X_c a) = \underset{||a||=1}{\operatorname{argmax}} \frac{1}{n-1}||X_c a||^2 \tag{5.3}$$

since X_c columns have zero mean, see Equation (5.2). Let $Y_c \in \mathbb{R}^{n\times p}$ be a matrix defined as

$$Y_c = \frac{1}{\sqrt{n-1}} X_c, \tag{5.4}$$

then the maximum of (5.3) is actually a solution of the Karush–Kuhn–Tucker equation

$$(Y_c^\top Y_c)a - \lambda a = 0 \qquad \text{with} \qquad ||a|| = 1$$

which is solved by the eigenvalue decomposition of $(Y_c^\top Y_c)$, i.e.,

$$(Y_c^\top Y_c) = A\Lambda A^\top$$

with $A \in \mathbb{O}(p \times p)$ and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_p)$. The eigenvalues are positive since $(Y_c^\top Y_c)$ is symmetric semipostive definite, see Equation (5.2), and for convenience we assume them to be sorted in decreasing order. The ℓ-th eigenvalue $\lambda_\ell \in \mathbb{R}^p$ of $(Y_c^\top Y_c)$ is a measure of the variance expressed by a_ℓ the ℓ-th eigenvector, i.e., the ℓ-th column of A. In climatology the eigenvectors a_ℓ are called *Empirical Orthogonal Functions*, or EOFs. Once the EOFs are computed, the anomaly field X_c is projected in the subspace spanned by the eigenvectors, defining the so-called *Principal Component* (PC) matrix $C \in \mathbb{R}^{n \times p}$ as $C = X_c A$. Thanks to this construction and the orthogonality of A, multiplying the equation defining the PC matrix on both sides by A^T leads to a decomposition of the anomaly filed X_c, that is

$$X_c = CA^T. \tag{5.5}$$

The components of the anomaly field X_c are expressed as a linear combination of the EOF matrix column a_ℓ and of the PC matrix column c_ℓ, i.e.,

$$X_c(i,h) = \sum_{\ell=1}^{p} c_\ell(i) a_\ell(h).$$

In climatology, EOFs and PCs are expressed as a function of the discrete space variable and of the time variable respectively. So, let s_h be the discrete coupled space variables running over the space grid for $h \in \{1, \dots p\}$ and let express the EOFs as $a_\ell(h) = a_\ell(s_h)$. Similarly PCs can be written as a function of the discrete-time variable, that is the i-th component of c_ℓ is denoted $c_\ell(t_i)$ for every $i \in \{1, \dots, n\}$. If now we reintroduce the discrete-time and coupled space variables, we obtain an expression of X_c where these two discrete variables are separated in EOFs and PCs, i.e.,

$$X_c(t_i, s_h) = X_c(i,h) = \sum_{\ell=1}^{p} c_\ell(t_i) a_\ell(s_h).$$

Remark 5.2.1. During the interpretation step, the ℓ-th EOF $a_\ell \in \mathbb{R}^p$ is reshaped into a matrix of size $(p_1 \times p_2)$ with $p = p_1 p_2$, that is the space grid size, and visualized over a Earth surface map.

To perform the EOF analysis, literature [64] numerical advice is to compute the SVD of X_c instead of the eigenvalue decomposition of $(X_c^\top X_c)$. The reduced SVD of X_c is

$$X_c = U \Sigma V^\top$$

with $U \in \mathbb{O}(n \times p)$, $V \in \mathbb{O}(p \times p)$ and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_p)$. Then the eigenvalues of $(X_c^\top X_c)$ are the squared singular values of X_c, i.e., $\lambda_\ell = \sigma_\ell^2$. By construction, the EOFs are equal to the eigenvectors of $(X_c^\top X_c)$, which correspond to the columns of V, the right orthogonal matrix from the SVD of X_c. Comparing Equation (5.5) and the SVD of X_c, the PC matrix writes $C = \frac{1}{\sqrt{n-1}} U \Sigma$. In particular, the ℓ-th PC is

$$c_\ell = \frac{\sigma_\ell}{\sqrt{n-1}} u_\ell$$

with u_ℓ the ℓ-th column of the left orthogonal matrix U. The scaling factor $\sqrt{n-1}$ is due to the definition of covariance matrix from Y_c, see Equation (5.4).

5.2.3 Covariance and correlation for EOFs

In climatology literature, see [7, 63], it is common to perform the EOFs analysis, not on the covariance matrix, defined in Equation (5.4), but on the correlation matrix, obtained by scaling the data by a symmetric matrix. Indeed, as suggested in [7, 63], this choice compensates for the non-uniform distribution of the grid point over the Earth's surface. So, we scale the input tensor data $\mathbf{f}$ by the diagonal matrix $D = \operatorname{diag}(\sqrt{\cos\theta_1}, \dots, \sqrt{\cos\theta_{p_1}})$ along the latitude mode, that is the second one, defining the scaled tensor $\mathbf{f}_s \in \mathbb{R}^{n\times p_1\times p_2}$ as

$$\mathbf{f}_s = D \times_2 \mathbf{f}.$$

The procedure to obtain the EOFs and the PCs requests now some tiny changes. Let $X_s \in \mathbb{R}^{n\times p}$ denotes the matricization with respect to mode 1 of $\mathbf{f}_s$, then, as previously, the matrix data is centred in time defining $X_{sc} \in \mathbb{R}^{n\times p}$ as

$$X_{sc} = X_s - \mathbb{1}_n \otimes \overline{x}_s \qquad \text{with} \qquad \overline{x}_s(h) = \frac{1}{n}\sum_{i=1}^{n} X_s(i,h)$$

for every $h \in \{1, \dots, p\}$. Then instead of computing the eigenvalue decomposition of the *correlation matrix* $(X_{sc}^\top X_{sc})$, we directly compute the SVD of X_{sc} as

$$X_{sc} = U\Sigma V^\top \tag{5.6}$$

where $U \in \mathbb{O}(n\times p)$, $V \in \mathbb{O}(p\times p)$ and $\Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_p)$. In this particular context, the PC matrix is commonly chosen as $C = (\sqrt{n-1})\,U$, i.e., with a different scaling. Since the EOFs are affected by the latitude scaling, as well explained in [7], the EOF matrix $A \in \mathbb{R}^{p\times p}$ is then constructed projecting the anomaly field not scaled X_c into the space spanned by the PCs, that is

$$A = \frac{1}{n}(X_c^\top C). \tag{5.7}$$

Remark 5.2.2. As stated in Remark 5.2.1, for the ease of the interpretation a_ℓ the ℓ-th column of the EOF matrix A is usually reshaped into a matrix of shape $(p_1 \times p_2)$. However in the correlation EOF analysis, instead of computing the EOF matrix, we can compute a tensor whose ℓ-th slice with respect to the first mode is the ℓ-th EOF already in matrix format. To do it, we firstly define $\mathbf{f}_c \in \mathbb{R}^{n\times p_1\times p_2}$, i.e., we centre in time the input tensor data

$$\mathbf{f}_c = \mathbf{f} - \mathbb{1}_n \otimes \overline{F} \qquad \text{where} \qquad \overline{F}(j,k) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{f}(i,j,k).$$

By construction $F_c^{(1)} = X_c$ since $\operatorname{vec}(\overline{F}) = \overline{x}$. Thus, the *EOF tensor* $\mathbf{a} \in \mathbb{R}^{p\times p_1\times p_2}$ is defined as

$$\mathbf{a} = \frac{1}{n}(C^\top \times_1 \mathbf{f}_c) \tag{5.8}$$

and by construction the matricization with respect to the first mode of $\mathbf{a}$ is equal to the EOF matrix A, that is $A^{(1)} = A$.

It is common to centre in time the PC matrix, i.e., defining $C_c = (\sqrt{n-1})U_c$ with

$$U_c = U - \mathbb{1}_n \otimes \overline{u} \qquad \text{with} \qquad \overline{u}(h) = \frac{1}{n}\sum_{i=1}^{n} U(i,h)$$

and using it to define the EOF matrix, that is $A_c = \frac{1}{\sqrt{n}}(X_c^\top C_c)$. In all the following sections, we assume to compute the EOFs and PCs following this approach frequently referred to as *correlation EOFs analysis*, centring in time both the EOF and the PC matrix.

5.3 EOF Analysis with Tucker model

In this section assuming that the input data are given in the Tucker compressed format, we derive the EOFs and PCs relying on this formalism. For ease of reading, the core tensor is denoted by the same letter of the decomposed (or approximated) tensor with a prime as superscript.

Let the tensor $\mathbf{f} \in \mathbb{R}^{n \times p_1 \times p_2}$ defined in Equation (5.1) be expressed in the Tucker compressed format, see Definition 1.3.1, that is

$$\mathbf{f} = (U, W, Z)\mathbf{f}'$$

where $U \in \mathbb{O}(n \times r_1)$, $W \in \mathbb{O}(p_1 \times r_2)$ and $Z \in \mathbb{O}(p_2 \times r_3)$, while $\mathbf{f}' \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is the core tensor. Equivalently this last equation can be written as

$$\mathbf{f} = \sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3} \mathbf{f}'(\alpha,\beta,\gamma) u_\alpha \otimes w_\beta \otimes z_\gamma$$

where u_α, w_β and z_γ are the α-th, β-th and γ-th column of U, W and Z respectively.

We follow the approach described in Section 5.2.3 to compute the EOFs and PCs. Firstly we scale the input data by the diagonal matrix $D = \mathrm{diag}(\sqrt{\cos\theta_1}, \ldots, \sqrt{\cos\theta_{p_1}})$ along the latitude mode, defining $\mathbf{f}_s \in \mathbb{R}^{n \times p_1 \times p_2}$

$$\begin{aligned}
\mathbf{f}_s &= D \times_2 \mathbf{f} = (U, DW, Z)\mathbf{f}' \\
&= \sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3} \mathbf{f}'(\alpha,\beta,\gamma) u_\alpha \otimes Dw_\beta \otimes z_\gamma \\
&= \sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3} \mathbf{f}'(\alpha,\beta,\gamma) u_\alpha \otimes v_\beta \otimes z_\gamma
\end{aligned}$$

where $v_\beta = Dw_\beta$ by construction for every $\beta \in \{1, \ldots, r_2\}$.

Remark 5.3.1. The matrix $V \in \mathbb{R}^{p_1 \times r_2}$ defined as $V = DW$ is not orthogonal any more.

The next step is centring the scaled data in time. Still keeping the data in Tucker format, we compute the average in time $\overline{F}_s \in \mathbb{R}^{p_1 \times p_2}$ as

$$\begin{aligned}\overline{F}_s(j,k) &= \frac{1}{n}\sum_{i=1}^{n} \mathbf{f}_s(i,j,k) \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3} \mathbf{f}'(\alpha,\beta,\gamma) u_\alpha(i)\, v_\beta(j)\, z_\gamma(k).\end{aligned}$$

In a more compact format, the average in time writes

$$\overline{F}_s = \sum_{\beta,\gamma=1}^{r_2,r_3} F'_s(\beta,\gamma) v_\beta \otimes z_\gamma \qquad \text{with} \qquad F'_s(\beta,\gamma) = \frac{1}{n}\sum_{i,\alpha=1}^{n,r_1} \mathbf{f}'(\alpha,\beta,\gamma) u_\alpha(i).$$

The scaled data centred in time are expressed by $\mathbf{f}_{sc} \in \mathbb{R}^{n \times p_1 \times p_2}$, given by

$$\begin{aligned}\mathbf{f}_{sc} &= \mathbf{f}_s - \sum_{\alpha=1}^{r_1} u_\alpha \otimes \overline{F}_s \\ &= \sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3} \Big(\mathbf{f}'(\alpha,\beta,\gamma) - F'_s(\beta,\gamma)\Big) u_\alpha \otimes v_\beta \otimes z_\gamma \\ &= \sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3} \mathbf{f}'_{sc}(\alpha,\beta,\gamma) u_\alpha \otimes v_\beta \otimes z_\gamma\end{aligned}$$

where $\mathbf{f}'_{sc}(\alpha,\beta,\gamma) = \Big(\mathbf{f}'(\alpha,\beta,\gamma) - F'_s(\beta,\gamma)\Big)$ for every $\alpha \in \{1,\dots,r_1\}$, $\beta \in \{1,\dots,r_2\}$ and $\gamma \in \{1,\dots,r_3\}$.

Denoted by $X_{sc} \in \mathbb{R}^{n \times p}$ the matricization of $\mathbf{f}_{sc}$ with respect to mode 1, then X_{sc} gets

$$X_{sc} = F^{(1)}_{sc} = \sum_{\alpha=1}^{r_1} u_\alpha \otimes \Big(\sum_{\beta,\gamma=1}^{r_2,r_3} \mathbf{f}'_{sc}(\alpha,\beta,\gamma)\, z_\gamma \otimes_{\mathrm{K}} v_\beta\Big), \tag{5.9}$$

thanks to Lemma 1.2.3. Thanks to Lemma 1.2.4, this previous equation writes

$$X_{sc} = U\Upsilon(Z \otimes_{\mathrm{K}} V)^\top \tag{5.10}$$

where $\Upsilon \in \mathbb{R}^{r_1 \times r_2 r_3}$ is the matricization with respect to mode 1 of the core tensor $\mathbf{f}'_{sc}$, while $U \in \mathbb{O}(n \times r_1)$, $V \in \mathbb{R}^{p_1 \times r_2}$ and $Z \in \mathbb{O}(p_2 \times r_3)$ have the α-th, β-th and γ-th column equal to u_α, v_β and z_γ respectively. In Section 5.2.3, the PCs are computed through the SVD of X_{sc}, but it is not correct to identify directly Equation (5.6) and (5.10). Indeed, the right factor of Equation (5.10) is not orthogonal because of the linear transformation applied over the second mode, see Remark 5.3.1. Consequently the orthogonal matrix U in Equation (5.10) is not the left orthogonal factor from the SVD of X_{sc}, i.e., it is different from the matrix U in Equation (5.6) by construction. However, we can still

benefit from the Tucker format, applying some further algebraic transformations. We proceed computing the QR-factorization of $(Z \otimes_{\mathrm{K}} V)$, getting $Q \in \mathbb{O}(p \times r)$ and the upper triangular matrix $R \in \mathbb{R}^{r \times r}$ with $p = p_1 p_2$ and $r = r_2 r_3$. Introducing the QR-factors into Equation (5.10), it writes

$$X_{sc} = U\Upsilon(QR)^\top = U(\Upsilon R^\top)Q^\top. \tag{5.11}$$

Let now compute the SVD of the $(r_1 \times r)$ matrix $(\Upsilon R^\top)$, as $(\Upsilon R^\top) = \hat{U}\hat{\Sigma}\hat{V}^\top$ with $\hat{U} \in \mathbb{O}(r_1 \times q)$ $\hat{V} \in \mathbb{O}(q \times r)$ and $\hat{\Sigma}$ a $(q \times q)$ diagonal matrix with q equal to the rank of $(\Upsilon R^\top)$. If the matrix product $(\Upsilon R^\top)$ is replaced by its SVD in Equation (5.11), it becomes

$$X_{sc} = U(\Upsilon R^\top\Big)Q^\top = (U\hat{U})\hat{\Sigma}(Q\hat{V})^\top.$$

Thanks to this last equation, we can compute the PC matrix from $(U\hat{U})$, which is equal up to a rotation, to the orthogonal matrix U of Equation (5.6). Thus, let the centred in time PC matrix be $C_c = (\sqrt{n-1})U_c$ where

$$U_c = (U\hat{U}) - \mathbb{1}_n \otimes \overline{u} \qquad \text{with} \qquad \overline{u}(h) = \frac{1}{n}\sum_{i=1}^{n}(U\hat{U})(i,h). \tag{5.12}$$

As described in Section 5.2.3, the EOF matrix is computed from the product of the PC matrix and X_c, the matricization with respect to mode 1 of $\mathbf{f}$ centred in time. As explained in Remark 5.2.2, instead of computing the EOF matrix, we can construct the EOF tensor from the input data centred on time and the PC matrix. We start computing $\overline{F} \in \mathbb{R}^{p_1 \times p_2}$ the average in time of the input data as

$$\begin{aligned}\overline{F}(j,k) &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{f}(i,j,k) \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3}\mathbf{f}'(\alpha,\beta,\gamma)u_\alpha(i)\,w_\beta(j)\,z_\gamma(k),\end{aligned}$$

which more compactly is expressed as

$$\overline{F} = \sum_{\beta,\gamma=1}^{r_2,r_3} F'(\beta,\gamma)w_\beta \otimes z_\gamma \qquad \text{with} \qquad F'(\beta,\gamma) = \frac{1}{n}\sum_{i,\alpha=1}^{n,r_1}\mathbf{f}'(\alpha,\beta,\gamma)u_\alpha(i).$$

Then the tensor data centred in time $\mathbf{f}_c \in \mathbb{R}^{n \times p_1 \times p_2}$ is such that

$$\begin{aligned}\mathbf{f}_c &= \mathbf{f} - \sum_{\alpha=1}^{r_1} u_\alpha \otimes \overline{F} \\ &= \sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3}\Big(\mathbf{f}'(\alpha,\beta,\gamma) - F'(\beta,\gamma)\Big)u_\alpha \otimes w_\beta \otimes z_\gamma \\ &= \sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3}\mathbf{f}'_c(\alpha,\beta,\gamma)u_\alpha \otimes w_\beta \otimes z_\gamma\end{aligned}$$

where $\mathbf{f}'_c(\alpha,\beta,\gamma) = \Big(\mathbf{f}'(\alpha,\beta,\gamma) - F'(\beta,\gamma)\Big)$ for every $\alpha \in \{1,\dots,r_1\}$, $\beta \in \{1,\dots,r_2\}$ and $\gamma \in \{1,\dots,r_3\}$. Finally the tensor EOF $\mathbf{a}_c \in \mathbb{R}^{r_1 \times p_1 \times p_2}$ writes

$$\begin{aligned}\mathbf{a}_c &= \frac{1}{n}(C_c^\top \times_1 \mathbf{f}_c) \\ &= \frac{\sqrt{n-1}}{n}(U_c^\top \times_1 \mathbf{f}_c) \\ &= \frac{\sqrt{n-1}}{n}\sum_{\alpha,\beta,\gamma=1}^{r_1,r_2,r_3} \mathbf{f}'_c(\alpha,\beta,\gamma) q_\alpha \otimes w_\beta \otimes z_\gamma,\end{aligned} \tag{5.13}$$

where $q_\alpha \in \mathbb{R}^{r_1}$ is equal to $U_c^\top u_\alpha$ for every $\alpha \in \{1,\dots,r_1\}$.

From the computational point of view, assuming the data to be already in Tucker format, this approach requests two main operation: the QR-factorization of a $(p \times r)$ matrix and the SVD of a $(r_1 \times r)$ matrix with $r = r_2 r_3$ and $p = p_1 p_2$. On the other side, the classical matrix approach performs only one SVD over a $(n_1 \times p)$ matrix. Consequently, if the multilinear rank components are considerably small, the tensor technique may be of interest.

5.4 Numerical results

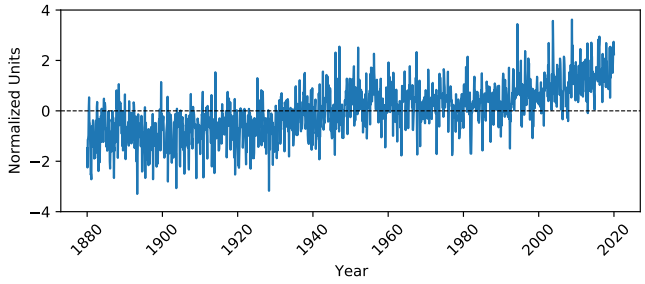
The purpose of this section is to investigate through the Tucker model a temperature time series. The dataset `HadCRUT4_RecAI_tas_mon_188001-201512.nc`, provided by L. Terray, research scientist in Climate, Environment, Coupling, and Uncertainties (CECI), a joint research team between Cerfacs and CNRS, contains a month temperature value in Kelvin for each point of the Earth's surface grid and 136 years. We mathematically represent the data as a tensor $\mathbf{f} \in \mathbb{R}^{n \times p_1 \times p_2}$ where $n = 1632$ is the number of time points, that is 12 months for 136 years, $p_1 = 36$ the number of latitude discrete points and $p_2 = 72$ of longitude ones.

We first present the first EOF and PC, which are by construction the most significant ones, obtained from the matrix correlation EOF analysis, see Section 5.2.3. Then a tensor approach is considered. To set a baseline, we study the Tucker basis of $\mathbf{f}_c$ the data after centring in time. In the conclusive section, we approximate $\mathbf{f}_c$ through the Tucker model prescribing an accuracy and we study the errors in computing the PC and EOF.

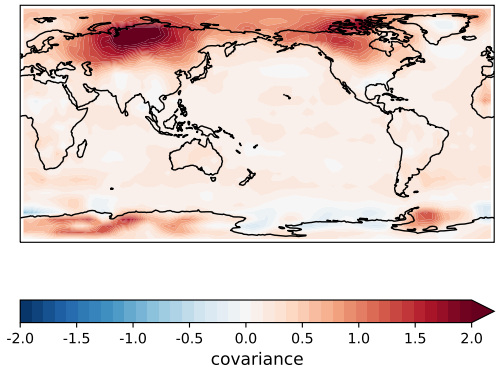
5.4.1 EOF analysis

To perform the correlation EOF analysis, as described in 5.2.3, we rely on the `python3` library `eofs` applied to $F^{(1)}$ the matricization with respect to the first mode of the tensor $\mathbf{f}$ storing the raw temperature time series. By construction, the first PC displayed in Figure 5.1A and the first EOF in Figure 5.1B are the most significant ones. The first PC in Figure 5.1A shows a time trend over the years for the temperature. Neglecting the many oscillations, we observe that the curve general behaviour is growing up to 1950, has a

drop-down point between the 60s and the 80s, and a final steep growth from the 90s. This general pattern reflects the atmospheric concentration of carbon dioxide, cf. [132]. The EOF plot 5.1B highlights the north hemisphere and the south pole as the ones with the higher covariance values. This result suggests that the temperature rise is more significant in some areas, the northern hemisphere, and the Antarctic pole, while other areas, such as the oceans, are lower affected. Those results are very classical in climatology, see for example [30].



(A) First PC.



(B) First EOF.

Figure 5.1 – PC and EOF obtained from the correlation EOF analysis.

5.4.2 Baseline Tucker model

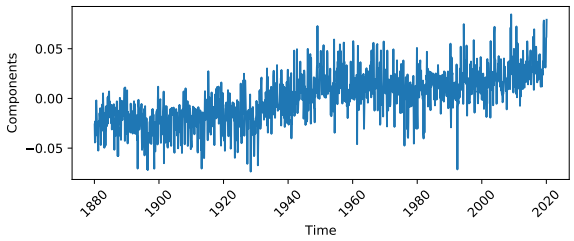
As a first step in the analysis of the temperature time series data through tensor methods, we decompose it by the HOSVD algorithm and study the properties of the Tucker basis so computed.

Remark 5.4.1. During the computational phase, we observe that the relative error among the raw data $\mathbf{f}$ and its HOSVD decomposition is of the order 10^{-7}, which is significantly larger than the machine precision of the working arithmetic. However, if we centre the data in time, defining $\mathbf{f}_c$, then the relative error among $\mathbf{f}_c$ and its HOSVD decomposition drops down to 10^{-15}, as expected. A clear explanation for this phenomenon is not present, but it may be linked to the high-rank nature of the time series. We choose to perform all the analysis on the Tucker decomposition by the HOSVD algorithm applied to $\mathbf{f}_c$.

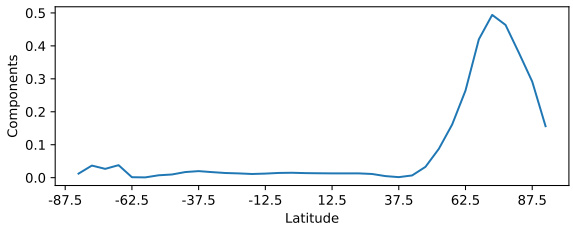
The data centred in time by $\mathbf{f}_c$ writes in Tucker format 1.3.1 as

$$\mathbf{f}_c = (U, W, Z)\mathbf{f}'_c$$

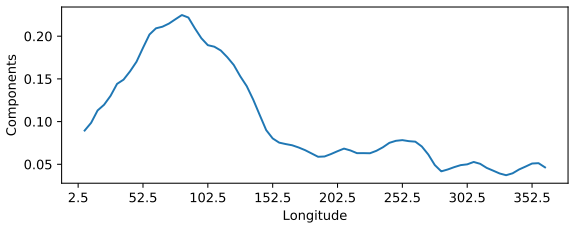
with $U \in \mathbb{O}(n \times r_1)$ the time basis, $W \in \mathbb{O}(p_1 \times r_2)$ the latitude one and $Z \in \mathbb{O}(p_2 \times r_3)$ the longitude one, while $\mathbf{f}'_c \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is the core tensor.



(A) 1st column of the time basis.



(B) 1st column of the latitude basis.



(C) 1st column of the longitude basis.

Figure 5.2 – Tucker basis of $\mathbf{f}_c$.

In Figure 5.2, we display the first column of the time, latitude and longitude basis. The profile of the first column of the time basis in Figure 5.2A presents a shape similar to the one of the 1st PC displayed in Figure 5.1A, even if on a different scale. The first column of the latitude basis in Figure 5.2B presents two peaks, a tiny one between the 90° and 60° south parallel and a major one between the 50° and 90° north parallel. The 90° and 60° south parallel isolate the South pole area, while the 50° and 90° the north hemisphere. On the other side, the longitude basis first column in Figure 5.2C has the main crest between the Greenwich meridian and the 150th one east, a minor one between the 90th meridian and the 150th one west. The area underlying the main crest goes from the east of Europe and Africa to the east of Asia and Australia, while the area included between the 90th and the 150th meridian west covers North and South America. Crossing the information coming from the latitude and longitude plots, the areas carrying the most significant information are the northern hemisphere and the Antarctic pole, as the first EOF shows in 5.1B.

5.4.3 Tucker model and EOF analysis

In this section, we investigate the error in constructing the PC and EOF from a Tucker approximation of $\mathbf{f}_c$ the data centred in time given an approximation accuracy.

Given $\mathbf{f}_c$ the data centred in time and the accuracy δ, we compute its Tucker approximation such that

$$\frac{\|\mathbf{f}_c - (U, W, Z)\mathbf{f}'_c\|}{\|\mathbf{f}_c\|} \leq \delta,$$

with $\mathbf{f}'_c \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ the core tensor, $U \in \mathbb{O}(n \times r_1)$, $W \in \mathbb{O}(p_1 \times r_2)$ and $Z \in \mathbb{O}(p_2 \times r_3)$. Notice that the multilinear rank (r_1, r_2, r_3) depends on the prescribed accuracy. The

accuracy values we consider belong to $\{0.5, 0.1, 10^{-2}, 10^{-3}, 10^{-8}\}$. To obtain the PC and the EOF from the Tucker approximation, we follow the approach described in Section 5.3. The $(n \times r_1)$ PC matrix centred in time is defined in Equation (5.12) from the QR of $(Z \otimes_{\mathrm{K}} DV)$ with D the scaling diagonal matrix and the SVD of $(\Upsilon R^\top)$ with Υ the matricization of the core tensor $\mathbf{f}'_c$ with respect to mode 1. To highlight that in this case an approximation of the data at accuracy δ is used instead of the decomposition, we denote the PC matrix centred in time by $C_{\delta c}$; a similar notation is used for the EOF tensor. The EOF tensor $\mathbf{a}_{\delta c} \in \mathbb{R}^{r_1 \times p_1 \times p_2}$ is computed from the PC matrix and the approximation at accuracy δ of $\mathbf{f}_c$.

The PC and EOF so computed are compared with those obtained from the classical matrix analysis. In particular we estimate the relative error for each PC as $e_\delta \in \mathbb{R}^{r_1}$ defined as

$$e_\delta(h) = \frac{\|C_c(:,h) - C_{\delta c}(:,h)\|}{\|C_c(:,h)\|}$$

while for the EOF tensor the relative error is $g_\delta \in \mathbb{R}^{r_1}$ such that

$$g_\delta(h) = \frac{\|\mathbf{a}_c(h,:,:) - \mathbf{a}_{\delta c}(h,:,:)\|}{\|\mathbf{a}_c(h,:,:)\|}$$

for every $h \in \{1, \ldots, r_1\}$. Before the analysis of the PC and EOF relative errors, we focus on the multilinear rank of the approximation for a given accuracy δ, reported in Table 5.1. Notice indeed that already for $\delta = 10^{-2}$, the second and third mode are full rank, since $r_2 = p_1$ and $r_3 = p_2$. For $\delta = 10^{-3}$, even the first mode is almost full rank, with $r_1 = n - 1$, while for $\delta = 10^{-8}$ we obtain the data decomposition in Tucker format. From the point of view of the relative approximation error, when $\delta = 0.5$, we are collecting approximately 60% of the total information, for $\delta = 0.1$ we keep 90% of it. Remark that for $\delta \in \{10^{-2}, 10^{-3}, 10^{-8}\}$ the relative error is at least one order less than δ. Those peculiarities seem to be related to the high-rank nature of the time series data. For completeness, we include the compression ratio defined as the number of memory units needed to store the tensor in Tucker format over the memory units requested for storing the tensor in full format, i.e.,

$$\frac{\sum_{h=1}^{d} n_h r_h + \prod_{h=1} r_h}{\prod_{j=1}^{d} n_j}$$

where r_h and n_h are the h-th component of the multilinear rank and the mode size respectively. Checking the third row of Table 5.1, we conclude that a beneficial compression arrives only when δ is equal to 0.5 with a request of only 9% of the memory used to store the entire tensor. For all the other accuracy values, the memory request to store the tensor in the Tucker format is more than the amount necessary to store the data in full format.

δ	0.5	0.1	10^{-2}	10^{-3}	10^{-8}
ml-rank	$(201, 16, 17)$	$(1130, 34, 67)$	$(1604, 36, 72)$	$(1631, 36, 72)$	$(1632, 36, 72)$
approximation error	0.376	0.094	0.005	10^{-7}	10^{-15}
compression ratio	0.091	1.046	1.603	1.630	1.631

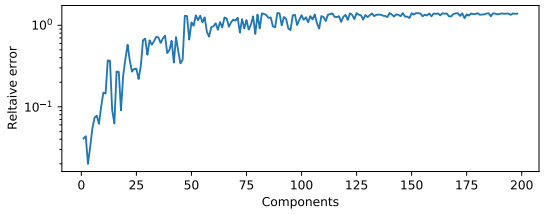
Table 5.1 – Multilinear rank depending on the prescribed accuracy.

In Figure 5.3 we display the PC and EOF relative error e_δ and g_δ on the left and right column respectively for every $\delta \in \{0.5, 0.1, 10^{-2}, 10^{-3}, 10^{-8}\}$. In all the plots of Figure 5.3, the relative error e_δ and g_δ have increasing component values, which implies that the first and most important PC and EOF are better approximated than the last ones. As we could expect, the more precise the approximations, i.e., the lower the values of δ, the smaller the PC and EOF relative errors are in general. As long as $\delta < 10^{-2}$, the PC relative error components are smaller than the EOF ones, as shown in Figures 5.3A, 5.3B, 5.3C and 5.3D. Moreover the h-th component of both e_δ and g_δ reaches and stagnates around 1, the relative error maximum, for $h > 100$ if $\delta = 0.5$ and for $h > 400$ if $\delta = 0.1$. When $\delta = 10^{-2}$, the first 200 components of g_δ appear, in Figures 5.3E and 5.3F, smaller than the correspondent ones of e_δ, that is the first EOF are better approximated than the first PC. Moreover $\delta = 10^{-2}$ is the only case where the components of e_δ and g_δ appear to be always increasing. The Figures 5.3G and 5.3I, 5.3H and 5.3J look similar, as we could expect since the two multilinear rank differ of one unit only on the first component. In these last two cases, we see that for $h \geq 400$, the components of the PC and EOF relative error stagnate around 10^{-6}.

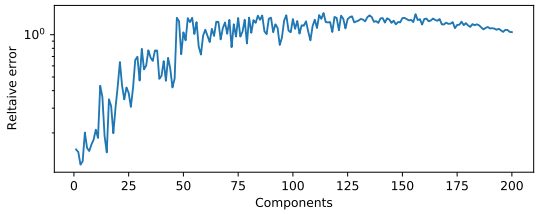
5.5 Concluding remarks

The focus of this chapter is on the investigation of climate datasets through tensor methods. In particular, the Tucker format is introduced in the EOF analysis, studying the benefit and the flaws of this choice. We show theoretically that the EOF and PC matrix can be computed from the Tucker factors and core tensor taking advantage of the compressed format, under the strong assumption that climate data are made available in Tucker format. To complete our study, we approximate a temperature time series through the HOSVD algorithm prescribing the accuracy and we study the effects of compression in the computation of the PC and EOF matrix.

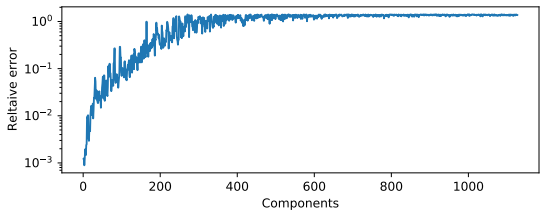
The first part, which includes Sections 5.2 and Section 5.3, collects the theoretical results. In Section 5.2, after describing the mathematical data modelling and the considered pretreatments, we present the EOF analysis. The purpose of this technique is to separate the space and time information of the data into the PC and the EOF matrix respectively. For completeness, Section 5.2 includes the EOF analysis based on the correlation and the covariance matrix associated with the data. The computational aspects of this



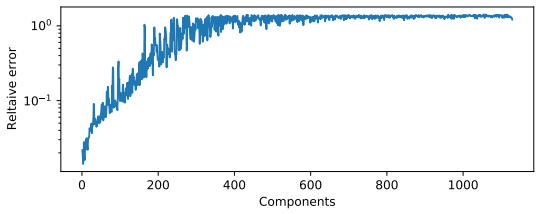
(A) PC, $\delta = 0.5$.



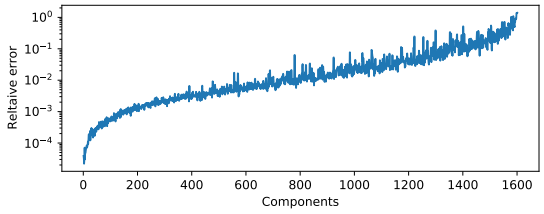
(B) EOF, $\delta = 0.5$.



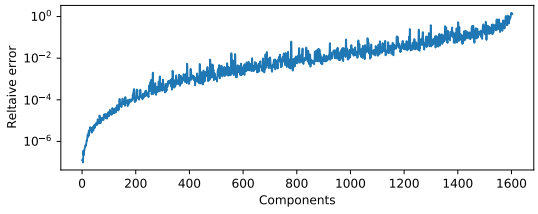
(C) PC, $\delta = 0.1$.



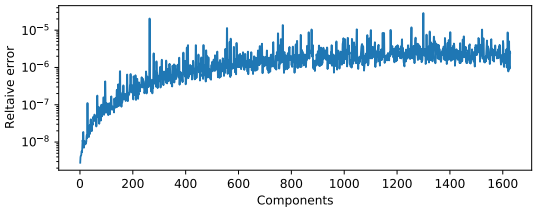
(D) EOF, $\delta = 0.1$.



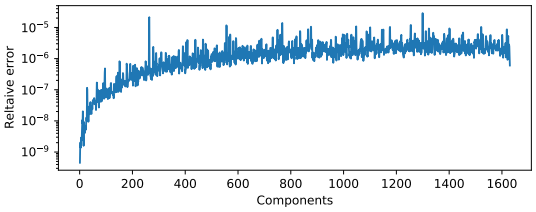
(E) PC, $\delta = 10^{-2}$.



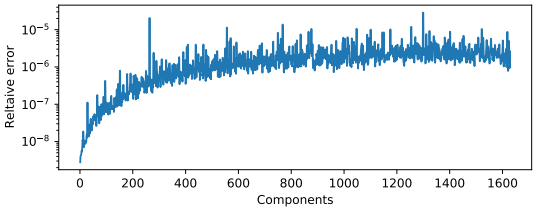
(F) EOF, $\delta = 10^{-2}$.



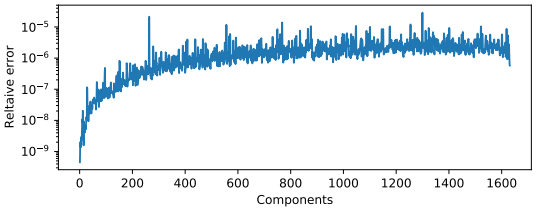
(G) PC, $\delta = 10^{-3}$.



(H) EOF, $\delta = 10^{-3}$.



(I) PC, $\delta = 10^{-8}$.



(J) EOF, $\delta = 10^{-8}$.

Figure 5.3 – Relative error for PC and EOF from Tucker approximation given an accuracy threshold δ.

technique are briefly introduced in Section 5.2. Section 5.3 focuses on the construction of the EOF and PC matrix from data mathematically expressed in Tucker format. Under this assumption, we show that the correlation EOF analysis can be performed keeping the data in compressed Tucker format, using the Tucker factors and core. Thus, if the multilinear rank of the considered data is low, the tensor approach appears beneficial from the computational viewpoint.

Section 5.4 constitutes the second part of the chapter. It is devoted to the numerical experiments performed on a single dataset through `python3` and the `eof` library. As the first step, we plot the first PC and EOF computed by the `eof` library; they are used as comparison terms for the other experiments. Then, we decompose the dataset by the HOSVD algorithm and analyse the components of the first column of each orthogonal basis. We illustrate the similarities among the first column of the time orthogonal basis and the first PC and the relations among the first latitude and longitude orthogonal basis column and the first EOF. In the last Section 5.4.3, the data are approximated by the HOSVD at a prescribed accuracy δ, then the core tensor and the orthogonal basis are used to compute an approximation of the PC and EOF matrix. As a final step, we compare the approximations of the EOF and PC matrices with those computed by the `eof` library. We display and investigate the relative errors for each PC and EOF depending on the approximation accuracy δ. Our study shows that the first and more important EOF and PC are better approximated than the last ones.

Conclusions and prospectives

The current fast technological development brings new mathematical and computational challenges while opening new research directions. A clear example of this is represented by linear algebra, which in the years has turned into multilinear algebra. Indeed, different scientific and social disciplines, that used to rely on linear algebra techniques, present nowadays problems that are naturally addressed in multilinear algebra, moving from matrix approaches to tensor ones.

The purpose of this work is to investigate the properties of some classical linear algebra methods, coming from numerical linear algebra and data analysis, once they are extended to the tensor framework through the classical tensor decomposition algorithms, recalled in Chapter 1. In the following sections, we summarize the main results and contributions, provided by our work, together with the main open questions and research directions that it has opened.

Numerical linear algebra

The topic of Part I is numerical linear algebra methods generalized to tensors though the Tensor-Train format [108], presented in Section 1.3.2.

More in detail, Chapter 2 focuses on the Generalised Minimal RESidual (GMRES) iterative solver from the backward stability viewpoint. Firstly we describe the variable accuracy approach, which consists in decoupling the computational and storage accuracies. This choice describes mathematically those situations where memory constraints make working with the classical IEEE floating point format not affordable. Once the representation hypotheses are fixed, we study experimentally the norm-wise backward error of GMRES, introducing storage perturbations firstly controlled component-wise and secondly norm-wise. The TT realization of GMRES in TT-format (TT-GMRES) represents a study case for the variable accuracy approach. So, we report on the backward error of TT-GMRES for solving several classical tensor linear systems. In particular, we address theoretically and numerically the problem of solving many tensor linear systems at once increasing the order of the space.

In Chapter 3, we describe and investigate experimentally six orthogonalization schemes, namely Classical Gram-Schmidt (CGS), Modified Gram-Schmidt (MGS) [52, 127], their version with re-orthogonalization (CGS2, MGS2) [51], the Gram procedure from [131] and the Householder transformation [76], implemented with the TT-format. Their TT-version

includes additional compression steps to prevent memory issues. Firstly, we report on their behavior in terms of loss of orthogonality of the computed basis. Then we plug them into the TT-version of the SUBSPace ITeration eigensolver (TT-SUBSPIT), analysing their effect in terms of eigenvalue computation.

Concluding remarks

The variable accuracy approach allows us to show experimentally in Chapter 2 that the backward stability of GMRES theoretically proved in [40, 113], still holds when the computational and the storage accuracies are distinguished. This means that the attainable accuracy of the backward error of GMRES appears of the order of the maximum between the computational and the storage accuracy, even when the assumptions of the theoretical proofs of [40, 113] are violated. This attainable accuracy property is observed numerically also for TT-GMRES. Our experiments lead us to conclude that our TT-GMRES algorithm is more robust than its previous relaxed version proposed in [39]. Finally, the theoretical bounds, which we formulate when many tensor linear systems are solved at once increasing the space order, appear from our numerical experiments sharp enough to guarantee some practical applicability.

The first contribution of Chapter 3 is the realization of the Householder orthogonalization kernel in TT-format, which has never been presented previously to the best of our knowledge. Additionally, the numerical experiments of Chapter 3 suggest that the loss of orthogonality in each orthogonalization scheme satisfies the same bounds [15, 51, 129, 131, 145] theoretically proved in the matrix case, where the unit round-off of the current working arithmetic is replaced by the compression accuracy. Lastly, the experimental results from the TT-SUBSPIT algorithm appear to recommend the TT-Householder transformation as the most robust one for every computation.

Perspectives

A central key point to address in the future is the theoretical validation of the backward stability of GMRES in the variable accuracy approach. Indeed, when the storage perturbations are controlled only norm-wise the backward stability theorems of [40, 113] do not apply anymore, since the underlying floating point representation is not holding anymore. However, the analysis presented in [113] is based on the worst-case. So, other possible proving techniques could be considered, such as those presented in [26, 114]. Similarly, the loss of orthogonality bounds for the six kernels should be studied theoretically. Moreover, studying the quality of the TT-eigenvectors as much as the relationship between the cluster distance and the rounding accuracy would complete the analysis of the TT-SUBSPIT algorithm. Finally, to link the two chapters, we could try to implement the flexibility described in [115] with the basis re-orthogonalization, to keep the memory advantages of the relaxed TT-GMRES from [39] without losing the attainable accuracy of our version.

Data Analysis

The data analysis techniques investigated in Part II are meant for contingency tables and climate data. Their extension to the tensor format relies on the Tucker model [135], described in Section 1.3.1.

The central subject of Chapter 4 is Correspondence Analysis (CA) from the geometric point of view. This technique is a variation of Principal Component Analysis (PCA) meant for investigating and visualizing frequency tables, referred to as contingency tables. The purpose of CA is to find the subspace that minimizes the projection error of the cloud points, whose coordinates are the row and the column entries of the contingency table. The projection of the two point clouds can be visualized and interpreted together, thanks to the barycentric relation that links them. The results of Chapter 4 integrate the MultiWay Correspondence Analysis (MWCA) theory, which generalizes the CA through the Tucker format. To stress the benefits and the flaws of a tensor approach two examples of MWCA in comparison with the matrix CA over literature data [57, 87] are presented. In the second half of Chapter 4, we apply the MWCA to investigate the Malabar dataset [6] relying on the barycentric relation. As for the other examples, the MWCA results are compared with the CA ones.

The Empirical Orthogonal Function (EOF) analysis is the subject studied in Chapter 5. This technique is a variation of the PCA tuned for climate data. The EOF analysis aims to separate time and space information, analysing them independently. After studying how to perform the EOF analysis when the data are available in Tucker format, we investigate numerically the effect of the Tucker compression on the EOF outcome, comparing them with the classical EOF analysis results.

Concluding remarks

In Chapter 4, we show through the Tucker model that the CA barycentric link holds theoretically also when tensors data are considered, i.e., in the MWCA. We stress mainly how to interpret geometrically this relation and how to use it in the visualization and interpretation of a multiway contingency table. So, we fulfil the gap in the MWCA, which has not been studied geometrically as far as we are aware. Moreover, we perform the analysis of an original dataset. In Chapter 5, we describe theoretically how to perform the EOF analysis starting from data expressed in Tucker format, without decompressing. In addition, our numerical studies suggest that the Tucker compression affects less the most important components of the final EOF outcome.

Perspectives

Both the CA and the EOF analysis could be regarded as visualization techniques relying on PCA. Nowadays, it is common to replace PCA with the Tucker model, which preserves some of the PCA properties. However, the optimality of the projection error is not among these. So a possible alternative research direction is represented by the

use of Canonical Polyadic decomposition [72] (CP) instead of the Tucker one. For the contingency table, for example, we could study if a similar barycentric relation may be inferred using the CP decomposition, or combining it with the Tucker model, to benefit also with the usual low dimension of the core tensor. On the other side, for the EOF analysis, the orthogonality of the factors does not represent a crucial constraint. However, climate data are known for being high-rank data, so tensor decomposition techniques beneficial when the low-rank property is present may not be a suitable tool. A much more general and open question is the search for new visualization techniques relying on tensor formats, but it may request knowledge and to be co-designed with other disciplines.

Bibliography

[1] N. N. Abdelmalek. "Round off error analysis for Gram-Schmidt method and solution of linear least squares problems". In: *BIT Numerical Mathematics* 11.4 (Dec. 1971), pp. 345–367. DOI: 10.1007/BF01939404.

[2] I. Affleck, T. Kennedy, E. H. Lieb, and H. Tasaki. "Rigorous results on valence-bond ground states in antiferromagnets". In: *Phys. Rev. Lett.* 59 (7 Aug. 1987), pp. 799–802. DOI: 10.1103/PhysRevLett.59.799.

[3] E. Agullo, O. Coulaud, L. Giraud, M. Iannacito, G. Marait, and N. Schenkels. *The backward stable variants of GMRES in variable accuracy*. Research Report RR-9483. Inria, Sept. 2022, pp. 1–77.

[4] W. E. Arnoldi. "The principle of minimized iterations in the solution of the matrix eigenvalue problem". In: *Quarterly of Applied Mathematics* 9 (1951), pp. 17–29.

[5] W. R. Atchley and E. H. Bryant. *Multivariate Statistical Methods: Among-Groups Covariation*. Benchmark Papers in Systematic and Evolutionary Biology. Elsevier Science & Technology Books, 1975.

[6] I. Auby, C. Meteigner, M. Rumebe, E. Chancerel, F. Salin, C. Aluome, F. Barraquand, L. Carassou, Y. Del Amo, V. Meleder, A. Petit, C. Picoche, j.-M. Frigerio, and A. Franc. *Malabar project: samples versus otus contingency tables*. Version V1. 2022. DOI: 10.57745/9QOSDY.

[7] M. P. Baldwin, D. B. Stephenson, and I. T. Jolliffe. "Spatial Weighting and Iterative Projection Methods for EOFs". In: *Journal of Climate* 22.2 (2009), pp. 234–243. DOI: 10.2307/26259635.

[8] J. Ballani and L. Grasedyck. "A projection method to solve linear systems in tensor format". In: *Numerical Linear Algebra with Applications* 20.1 (2013), pp. 27–43. DOI: 10.1002/nla.1818.

[9] F. L. Bauer. "Das Verfahren der Treppeniteration und verwandte Verfahren zur Lösung algebraischer Eigenwertprobleme". In: *Zeitschrift für angewandte Mathematik und Physik ZAMP* 8.3 (May 1957), pp. 214–235.

[10] E. J. Beh and R. Lombardo. *Correspondence Analysis*. Ed. by E. J. Beh and R. Lombardo. Wiley Series in Probability and Statistics. Wiley & Sons, Oct. 2014, pp. 1–560. DOI: 10.1002/9781118762875.

[11] E. J. Beh and R. Lombardo. "Multiple and multiway correspondence analysis". In: *WIREs Computational Statistics* 11.5 (2019), e1464. DOI: 10.1002/wics.1464.

[12] J.-P. Benzécri. "Statistical analysis as a tool to make patterns emerge from data". In: *Methodologies of Pattern Recognition*. Ed. by S. Watanabe. Academic Press, 1969, pp. 35–74. DOI: 10.1016/B978-1-4832-3093-1.50009-2.

[13] J.-P. Benzécri and L. Bellier. *L'analyse des données: Benzécri, J.-P. et al. L'analyse des correspondances*. L'analyse des données: leçons sur l'analyse factorielle et la reconnaissance des formes, et travaux du Laboratoire de statistique de l'Université de Paris VI. Dunod, 1973.

[14] J.-P. Benzécri. "Sur l'analyse des tableaux binaires associés à une correspondance multiple". In: *Cahiers de l'analyse des données* 2.1 (1977), pp. 55–71.

[15] Å. Björck. "Solving linear least squares problems by Gram-Schmidt orthogonalization". In: *BIT Numerical Mathematics* 7.1 (Mar. 1967), pp. 1–21. DOI: 10.1007/BF01934122.

[16] A. Bouras and V. Frayssé. "Inexact matrix-vector products in Krylov methods for solving linear systems: a relaxation strategy". In: *SIAM Journal on Matrix Analysis and Applications* 26.3 (2005), pp. 660–678. DOI: 10.1137/S0895479801384743.

[17] M. Braschler, T. Stadelmann, and K. Stockinger. *Applied Data Science: Lessons Learned for the Data-Driven Business*. Cham: Springer International Publishing, 2019, pp. 17–29. DOI: 10.1007/978-3-030-11821-1_2.

[18] R. Bro, E. Acar, and T. G. Kolda. "Resolving the sign ambiguity in the singular value decomposition". In: *Journal of Chemometrics* 22.2 (2008), pp. 135–140. DOI: 10.1002/cem.1122.

[19] R. Bro, R. Leardi, and L. G. Johnsen. "Solving the sign indeterminacy for multiway models". In: *Journal of Chemometrics* 27.3-4 (2013), pp. 70–75. DOI: 10.1002/cem.2493.

[20] C. Burt. "The factorial analysis of qualitative data". In: *British Journal of Statistical Psychology* 3.3 (1950), pp. 166–185. DOI: 10.1111/j.2044-8317.1950.tb00296.x.

[21] A. Carlier and P. M. Kroonenberg. "Decompositions and biplots in three-way correspondence analysis". In: *Psychometrika* 61.2 (June 1996), pp. 355–373. DOI: 10.1007/BF02294344.

[22] A. Carlier and P. M. Kroonenberg. "Chapter 19 - The Case of the French Cantons: An Application of Three-Way Correspondence Analysis". In: *Visualization of Categorical Data*. Ed. by J. Blasius and M. J. Greenacre. San Diego, USA: Academic Press, 1998, pp. 253–275. DOI: 10.1016/B978-012299045-8/50021-8.

[23] CARME-N. *Opinions about working women*. http://www.carme-n.org/?sec=data2. [Data-set 9]. 2007.

[24] CARME-N. *Spanish National Health Survey*. `http://www.carme-n.org/?sec=data2`. [Data-set 3]. 2007.

[25] J. D. Carroll and J.-J. Chang. "Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition". In: *Psychometrika* 35.3 (Sept. 1970), pp. 283–319. DOI: `10.1007/BF02310791`.

[26] E. Carson and N. J. Higham. "A New Analysis of Iterative Refinement and Its Application to Accurate Solution of Ill-Conditioned Sparse Linear Systems". In: *SIAM Journal on Scientific Computing* 39.6 (2017), A2834–A2856. DOI: `10.1137/17M1122918`.

[27] R. B. Cattell. ""Parallel proportional profiles" and other principles for determining the choice of factors by rotation". In: *Psychometrika* 9 (1944), pp. 267–283.

[28] J. Chambers. *Computational Methods for Data Analysis*. Wiley Series in Probability and Mathematical Statistics. Wiley & Sons, 1977.

[29] R. Coppi and S. Bolasco, eds. *Multiway Data Analysis*. North-Holland Publishing Co., 1989.

[30] Core Writing Team, R. Pachauri, and A. Reisinger. *Climate Change 2007: Synthesis Report. Contribution of Working Groups I, II and III to the Fourth Assessment*. Tech. rep. Geneva, Switzerland: Intergovernmental Panel on Climate Change, 2007, p. 104.

[31] O. Coulaud, L. Giraud, and M. Iannacito. *A note on GMRES in TT-format*. Research Report RR-9384. Inria Bordeaux Sud-Ouest, 2022.

[32] O. Coulaud, L. Giraud, and M. Iannacito. *On some orthogonalization schemes in TT-format*. Research Report RR-9491. Inria Bordeaux Sud-Ouest, 2022.

[33] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. "Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization". In: *Mathematics of Computation* 30.136 (1976), pp. 772–795.

[34] T. H. Davenport and D. Patil. *Data scientist: The sexiest job of the 21st century*. `http://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century/ar/1`. 2012.

[35] T. A. Davis and Y. Hu. "The university of Florida sparse matrix collection". In: *ACM Transactions on Mathematical Software* 38.1 (Nov. 2011), pp. 1–25. DOI: `10.1145/2049662.2049663`.

[36] L. De Lathauwer, B. De Moor, and J. Vandewalle. "A Multilinear Singular Value Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1253–1278. DOI: `10.1137/S0895479896305696`.

[37] L. De Lathauwer, B. De Moor, and J. Vandewalle. "On the Best Rank-1 and Rank-$(R_1, R_2, \ldots, R_N)$ approximation of high order tensors". In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1324–1342. DOI: `10.1137/S0895479898346995`.

[38] S. Di and F. Cappello. "Fast error-bounded lossy HPC data compression with SZ". In: *2016 IEEE international parallel and distributed processing symposium (IPDPS)*. IEEE. 2016, pp. 730–739.

[39] S. V. Dolgov. "TT-GMRES: solution to a linear system in the structured tensor format". In: *Russian Journal of Numerical Analysis and Mathematical Modelling* 28.2 (2013), pp. 149–172. DOI: 10.1515/rnam-2013-0009.

[40] J. Drkosova, A. Greenbaum, M. Rozloznik, and Z. Strakoš. "Numerical stability of GMRES". In: *BIT Numerical Mathematics* 35.February 1994 (1995), pp. 309–330.

[41] C. Eckart and G. Young. "The approximation of one matrix by another of lower rank". In: *Psychometrika* 1.3 (Sept. 1936), pp. 211–218. DOI: 10.1007/BF02288367.

[42] G. Favier. "Historical Elements of Matrices and Tensors". In: *From Algebraic Structures to Tensors*. Ed. by G. Favier. Wiley & Sons, 2019. Chap. 1, pp. 1–7. DOI: 10.1002/9781119681137.ch1.

[43] R. A. Fisher. "The precision of discriminant functions". In: *Annals of Human Genetics* 10 (1940), pp. 422–429.

[44] A. Franc. "Etude Algébrique des multitableaux: apports de l'algèbre tensorielle". PhD thesis. University of Montpellier, 1992.

[45] A Fukuoka. "The central meteorological observatory, a study on 10-day forecast (a synthetic report)". In: *Geophysical Magazine* 22.3 (1951), pp. 177–208.

[46] K. R. Gabriel. "The biplot graphic display of matrices with application to principal component analysis". In: *Biometrika* 58.3 (Dec. 1971), pp. 453–467. DOI: 10.1093/biomet/58.3.453.

[47] H. G. Gauch. *Multivariate Analysis in Community Ecology*. Cambridge Studies in Ecology. Cambridge University Press, 1982. DOI: 10.1017/CBO9780511623332.

[48] P. Gelß. "The Tensor-Train Format and Its Applications". PhD thesis. 2017. DOI: 10.17169/refubium-7566.

[49] A. Gifi. *Nonlinear Multivariate Analysis*. Wiley Series in Probability and Statistics. Wiley & Sons, 1990.

[50] L. Giraud, S. Gratton, and J. Langou. "Convergence in Backward Error of Relaxed GMRES". In: *SIAM Journal Scientific Computing* 29.2 (2007), pp. 710–728. DOI: 10.1137/040608416.

[51] L. Giraud, J. Langou, M. Rozložník, and J. v. d. Eshof. "Rounding error analysis of the classical Gram-Schmidt orthogonalization process". In: *Numerische Mathematik* 101.1 (July 2005), pp. 87–100. DOI: 10.1007/s00211-005-0615-4.

[52] J. P. Gram. "Ueber die Entwickelung reeller Functionen in Reihen mittelst der Methode der kleinsten Quadrate". In: *Journal für die reine und angewandte Mathematik (Crelles Journal)* 1883.94 (1883), pp. 41 –73.

[53] L. Grasedyck. "Hierarchical Singular Value Decomposition of Tensors". In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (2010), pp. 2029–2054. DOI: 10.1137/090764189.

[54] L. Grasedyck. "Hierarchical Singular Value Decomposition of Tensors". In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (2010), pp. 2029–2054. DOI: 10.1137/090764189.

[55] M. J. Greenacre and L. Degos. "Correspondence Analysis of HLA Gene Frequency Data from 124 Population Samples". In: *American journal of human genetics* 29 (Feb. 1977), pp. 60–75.

[56] M. J. Greenacre. "Some objective methods of graphical display of a data matrix". In: *UNISA, Pretoria* (1978).

[57] M. J. Greenacre. *Theory and Applications of Correspondence Analysis*. Academic Press, 1984.

[58] M. Greenacre. *Correspondence Analysis in Practice*. Data-sets are available at. Academic Press, 1993.

[59] A. Greenbaum. *Iterative methods for solving linear systems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997.

[60] L. Guttman. "The Quantification of a class of attributes: A theory and method of scale construction". In: Social Science Research Council, 1941, pp. 319–348.

[61] W. Hackbusch and B. N. Khoromskij. "Low-rank Kronecker-product Approximation to Multi-dimensional Nonlocal Operators. Part I. Separable Approximation of Multi-variate Functions". In: *Computing* 76.3 (Jan. 2006), pp. 177–202. DOI: 10.1007/s00607-005-0144-0.

[62] W. Hackbusch and B. N. Khoromskij. "Low-rank Kronecker-product Approximation to Multi-dimensional Nonlocal Operators. Part II. HKT Representation of Certain Operators". In: *Computing* 76.3 (Jan. 2006), pp. 203–225. DOI: 10.1007/s00607-005-0145-z.

[63] A. Hannachi. *Patterns Identification and Data Mining in Weather and Climate*. Springer Atmospheric Sciences. Springer International Publishing, 2021.

[64] A. Hannachi, I. T. Jolliffe, and D. B. Stephenson. "Empirical orthogonal functions and related techniques in atmospheric science: A review". In: *International Journal of Climatology* 27.9 (2007), pp. 1119–1152. DOI: 10.1002/joc.1499.

[65] R. A. Harshman. "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-model factor analysis". In: vol. 16. UCLA Working Papers in Phonetics. University Microfilms, Ann Arbor, Michigan, 1970, pp. 1–84.

[66] C. Hayashi. "On the quantification of qualitative data from the mathematico-statistical point of view". In: *Annals of the Institute of Statistical Mathematics* 2 (1950), pp. 35–47.

[67] C. Hayashi. "What is Data Science ? Fundamental Concepts and a Heuristic Example". In: *Data Science, Classification, and Related Methods*. Ed. by C. Hayashi, K. Yajima, H.-H. Bock, N. Ohsumi, Y. Tanaka, and Y. Baba. Tokyo: Springer Japan, 1998, pp. 40–51.

[68] N. Higham. *Accuracy and Stability of Numerical Algorithms: Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2002.

[69] M. O. Hill. "Correspondence analysis". In: *Encyclopedia of Statistical Sciences*. Ed. by N. L. J. Samuel Kotz. New York, NY: Wiley & Sons, 1982, pp. 204–210.

[70] M. O. Hill. "Correspondence Analysis: A Neglected Multivariate Method". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 23.3 (1974), pp. 340–354. DOI: https://doi.org/10.2307/2347127.

[71] H. O. Hirschfeld. "A Connection between Correlation and Contingency". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 31 (1935), pp. 520 – 524.

[72] F. L. Hitchcock. "The Expression of a Tensor or a Polyadic as a Sum of Products". In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189. DOI: 10.1002/sapm192761164.

[73] W. Hoffmann. "Iterative algorithms for Gram-Schmidt orthogonalization". In: *Computing* 41.4 (Dec. 1989), pp. 335–348. DOI: 10.1007/BF02241222.

[74] P. Horst. "Measuring Complex Attitudes". In: *Journal of Social Psychology* 6 (1935), pp. 369–374.

[75] H. Hotelling. "Analysis of a complex of statistical variables into principal components." In: *Journal of Educational Psychology* 24.6 (1933), pp. 417–441. DOI: 10.1037/h0071325.

[76] A. S. Householder. "Unitary Triangularization of a Nonsymmetric Matrix". In: *J. ACM* 5.4 (Oct. 1958), 339–342. DOI: 10.1145/320941.320947.

[77] "IEEE Standard for Binary Floating-Point Arithmetic". In: *ANSI/IEEE Std 754-1985* (1985), pp. 1–20. DOI: 10.1109/IEEESTD.1985.82928.

[78] I. T. Jolliffe. *Principal Component Analysis*. Second. Springer Series in Statistics. Springer-Verlag New York, 2002.

[79] I. T. Jolliffe and J. Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202. DOI: 10.1098/rsta.2015.0202.

[80] A. Kapteyn, H. Neudecker, and T. Wansbeek. "An approach to $n-$mode components analysis". In: *Psychometrika* 51.2 (1986), pp. 269–275. DOI: 10.1007/BF02293984.

[81] V. A. Kazeev and B. N. Khoromskij. "Low-Rank Explicit QTT Representation of the Laplace Operator and Its Inverse". In: *SIAM Journal on Matrix Analysis and Applications* 33.3 (2012), pp. 742–758. DOI: 10.1137/100820479.

[82] M. G. Kendall. *Multivariate analysis*. English. Griffin London, 1975, p. 210.

[83] T. G. Kolda. *Multilinear Operators for Higher-order Decompositions*. Tech. rep. SAND2006-2081. Sandia National Laboratories, Apr. 2006. DOI: 10.2172/923081.

[84] T. G. Kolda and B. W. Bader. "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (Aug. 2009), pp. 455–500. DOI: 10.1137/07070111x.

[85] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic. "TensorLy: Tensor Learning in Python". In: *Journal of Machine Learning Research* 20.26 (2019), pp. 1–6.

[86] D. Kressner and C. Tobler. "Low-Rank Tensor Krylov Subspace Methods for Parametrized Linear Systems". In: *SIAM Journal on Matrix Analysis and Applications* 32.4 (2011), pp. 1288–1316. DOI: 10.1137/100799010.

[87] P. M. Kroonenberg. *Applied Multiway Data Analysis*. Wiley & Sons, 2008.

[88] P. M. Kroonenberg. "History of multiway component analysis and three-way correspondence analysis". In: *Visualization and Verbalization of Data*. Ed. by G. M. Blasius J. CRC Press, 2014. Chap. 6, pp. 77–94.

[89] P. M. Kroonenberg. *Three-mode principal component analysis: Theory and applications*. DSWO Press, Leiden, 1983.

[90] P. M. Kroonenberg and J. de Leeuw. "Principal Component Analysis of three modes data by means of Alternating Least Square algorithms". In: *Psychometrika* 45.1 (1980), pp. 69–97. DOI: 10.1007/BF02293599.

[91] A. Krylov. "On the Numerical Solution of Equation by Which are Determined in Technical Problems the Frequencies of Small Vibrations of Material Systems". In: *Izvestiia Akademii nauk SSSR* 7.4 (1931), 491–539.

[92] J. Landsberg. *Tensors: Geometry and Applications: Geometry and Applications*. Graduate studies in mathematics. American Mathematical Society, 2011.

[93] L. Lebart. *Validité des résultats en analyse des données*. Tech. rep. SOU1975-2140. Paris, FR: CREDOC-DGRST, Nov. 1975.

[94] L. Lebart, A. Morineau, and J. Fénelon. *Traitement des données statistiques: méthodes et programmes*. Dunod, 1982.

[95] L. Lebart, A. Morineau, and N. Tabard. *Techniques de la Description Statistique: méthodes et logiciels pour l'analyse des grands tableaux*. Paris, FR: Dunod, 1977.

[96] S. J. Leon, Å. Björck, and W. Gander. "Gram-Schmidt orthogonalization: 100 years and more". In: *Numerical Linear Algebra with Applications* 20.3 (2013), pp. 492–532. DOI: 10.1002/nla.1839.

[97] J. Levin. "Three-Mode Factor Analysis". PhD thesis. University of Illinois, Urbana, 1963.

[98] R. Lombardo, A. Carlier, and L. D'Ambra. "Nonsymmetric correspondence analysis for three-way contingency tables". In: *Methodologica* 4 (1996), pp. 59–80.

[99] E. N. Lorenz. *Empirical Orthogonal Functions and Statistical Weather Prediction*. Tech. rep. 1. Statistical Forecasting Project. Department of Meteorology, MIT, 1956, p. 52.

[100] M. Loukides. *What is data science?* `https://www.oreilly.com/ideas/what-is-data-science`. 2010.

[101] Y. Ma, S. Sastry, and R. Vidal. *Generalized Principal Component Analysis*. Interdisciplinary Applied Mathematics. Springer New York, 2015.

[102] J. Manyika. *Hal Varian on how the Web challenges managers*. McKinsey Quarterly, `https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/hal-varian-on-how-the-web-challenges-managers`. 2009.

[103] Martinsson, P. Gunnar, V. Rokhlin, and M. Tygert. "A randomized algorithm for the decomposition of matrices". In: *Applied and Computational Harmonic Analysis* 30.1 (2011), pp. 47–68. DOI: `10.1016/j.acha.2010.02.003`.

[104] G. Meurant and J. D. Tebbens. *Krylov Methods for Nonsymmetric Linear Systems*. Springer International Publishing, 2020. DOI: `10.1007/978-3-030-55251-0`.

[105] D. Morrison. *Multivariate Statistical Methods*. Annals of the New York Academy of Sciences. New York: McGraw-Hill, 1976.

[106] J. von Neumann and H. H. Goldstine. "Numerical inverting of matrices of high order". In: *Bulletin of the American Mathematical Society* 53.11 (1947), pp. 1021–1099. DOI: `bams/1183511222`.

[107] A. M. Obukhov. "Statistically homogeneous fields on a sphere". In: *Usp. Mat. Nauk* 2.2 (1947), pp. 196–198.

[108] I. V. Oseledets. "Tensor-Train Decomposition". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317. DOI: `10.1137/090752286`.

[109] I. V. Oseledets. *ttpy*. Version 1.2.0. https://github.com/oseledets/ttpy. 2015.

[110] I. V. Oseledets and E. E. Tyrtyshnikov. "Breaking the Curse of Dimensionality, Or How to Use SVD in Many Dimensions". In: *SIAM Journal on Scientific Computing* 31.5 (2009), pp. 3744–3759. DOI: `10.1137/090748330`.

[111] I. V. Oseledets. "DMRG Approach to Fast Linear Algebra in the TT-Format". In: *Computational Methods in Applied Mathematics* 11.3 (2011), pp. 382–393. DOI: `10.2478/cmam-2011-0021`.

[112] C. C. Paige and M. A. Saunders. "Solution of Sparse Indefinite Systems of Linear Equations". In: *SIAM Journal on Numerical Analysis* 12.4 (1975), pp. 617–629. DOI: `10.1137/0712047`.

[113] C. C. Paige, M. Rozloznik, and Z. Strakoš. "Modified Gram–Schmidt (MGS), least squares, and backward stability of MGS-GMRES". In: *SIAM Journal on Matrix Analysis and Applications* 28.1 (2006), pp. 264–284. DOI: 10.1137/050630416.

[114] C. C. Paige and Z. Strakoš. "Residual and Backward Error Bounds in Minimum Residual Krylov Subspace Methods". In: *SIAM Journal on Scientific Computing* 23.6 (Jan. 2002), pp. 1898–1923. DOI: 10.1137/s1064827500381239.

[115] D. Palitta and P. Kürschner. "On the convergence of Krylov methods with low-rank truncations". In: *Numerical Algorithms* 88.3 (2021), pp. 1383–1417. DOI: 10.1007/s11075-021-01080-2.

[116] B. Parlett. *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1980.

[117] D. Patil. *Building data science teams*. http://radar.oreilly.com/2011/09/building-data-science-teams.html. Available March 23, 2018. 2011.

[118] K. Pearson. *On the Theory of Contingency and Its Relation to Association and Normal Correlation*. London, UK: Dulau and Co., 1904.

[119] K. Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720.

[120] M. W. Richardson and G. F. Kuder. "Making a rating scale that measures." In: *The Personnel journal* 12 (1933), pp. 36–40.

[121] J. L. Rigal and J. Gaches. "On the Compatibility of a Given Solution With the Data of a Linear System". In: *Journal of the ACM* 14.3 (June 1967), pp. 543–548. DOI: 10.1145/321406.321416.

[122] V. Rokhlin, A. Szlam, and M. Tygert. "A Randomized Algorithm for Principal Component Analysis". In: *SIAM Journal on Matrix Analysis and Applications* 31.3 (Jan. 2010), pp. 1100–1124. ISSN: 0895-4798. DOI: 10.1137/080736417.

[123] Y. Saad. "ILUT: A dual threshold incomplete ILU factorization". In: *Numerical Linear Algebra with Applications* 1 (1994), pp. 387–402. DOI: 10.1002/nla.1680010405.

[124] Y. Saad and M. H. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (1986), pp. 856–869. DOI: 10.1137/0907058.

[125] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003.

[126] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Society for Industrial and Applied Mathematics, 2011. DOI: 10.1137/1.9781611970739.

[127] E. Schmidt. "Zur Theorie der linearen und nichtlinearen Integralgleichungen". In: *Mathematische Annalen* 63.4 (Dec. 1907), pp. 433–476. DOI: 10.1007/BF01449770.

[128] V. Simoncini and D. B. Szyld. "Theory of Inexact Krylov Subspace Methods and Applications to Scientific Computing". In: *SIAM Journal on Scientific Computing* 25.2 (2003), pp. 454–477. DOI: 10.1137/S1064827502406415.

[129] A. Smoktunowicz, J. L. Barlow, and J. Langou. "A note on the error analysis of classical Gram–Schmidt". In: *Numerische Mathematik* 105.2 (Dec. 2006), pp. 299–313. DOI: 10.1007/s00211-006-0042-1.

[130] T. Stadelmann, K. Stockinger, M. Braschler, M. Cieliebak, G. Baudinot, O. Dürr, and A. Ruckstuhl. "Applied data science in Europe : challenges for academia in keeping up with a highly demanded topic". In: *9th European Computer Science Summit*. 2013.

[131] A. Stathopoulos and K. Wu. "A Block Orthogonalization Procedure with Constant Synchronization Requirements". In: *SIAM Journal on Scientific Computing* 23.6 (2002), pp. 2165–2182. DOI: 10.1137/S1064827500370883.

[132] R. N. Stavins. "The Problem of the Commons: Still Unsettled after 100 Years". In: *American Economic Review* 101.1 (Feb. 2011), pp. 81–108. DOI: 10.1257/aer.101.1.81.

[133] G. W. Stewart. *Matrix Algorithms*. Society for Industrial and Applied Mathematics, 2001. DOI: 10.1137/1.9780898718058.

[134] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.

[135] L. R. Tucker. "Implications of factor analysis of three way matrices for measurements of change". In: *Problems in measuring change*. Ed. by C. Harris. Madison: University of Wisconsin Press, 1963, pp. 122–137.

[136] L. R. Tucker. "The extension of factor analysis to three-dimensional matrices". In: *Contributions to mathematical psychology*. Ed. by H. Gulliksen and N. Frederiksen. New York: Holt, Rinehart and Winston, 1964, pp. 110–127.

[137] L. R. Tucker. "Some mathematical notes on three-modes factor analysis". In: *Psychometrika* 31.3 (1966), pp. 279–311. DOI: 10.1007/BF02289464.

[138] J. W. Tukey. "The Future of Data Analysis". In: *The Annals of Mathematical Statistics* 33.1 (1962), pp. 1 –67. DOI: 10.1214/aoms/1177704711.

[139] J. van den Eshof and G. L. G. Sleijpen. "Inexact Krylov subspace methods for linear systems". In: *SIAM Journal on Matrix Analysis and Applications* 26.1 (2004), pp. 125–153. DOI: 10.1137/S0895479802403459.

[140] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen. "A New Truncation Strategy for the Higher-Order Singular Value Decomposition". In: *SIAM Journal on Scientific Computing* 34.2 (2012), A1027–A1052. DOI: 10.1137/110836067.

[141] M. A. O. Vasilescu and D. Terzopoulos. "Multilinear Analysis of Image Ensembles: TensorFaces". In: *Computer Vision — ECCV 2002*. Ed. by A. Heyden, G. Sparr, M. Nielsen, and P. Johansen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 447–460.

[142] H. F. Walker. "Implementation of the GMRES method using Householder transformations". In: *SIAM Journal on Scientific Computing* 9.1 (1988), pp. 152–163. DOI: 10.1137/0909010.

[143] J. H. Wilkinson. "Modern Error Analysis". In: *SIAM Review* 13.4 (Oct. 1971), pp. 548–568. DOI: 10.1137/1013095.

[144] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Vol. 32. Notes on Applied Science. Also published by Prentice-Hall, Englewood Cliffs, NJ, USA, 1964, translated into Polish as *Bledy Zaokragleń w Procesach Algebraicznych* by PWW, Warsaw, Poland, 1967 and translated into German as *Rundungsfehler* by Springer-Verlag, Berlin, Germany, 1969. Reprinted by Dover Publications, New York, 1994. London, UK: HMSO, 1963, pp. vi + 161.

[145] J. H. Wilkinson. *The algebraic eigenvalue problem*. en. Numerical Mathematics and Scientific Computation. Oxford, England: Clarendon Press, 1965.

[146] E. J. Williams. "Use of scores for the analysis of association in contingency table". In: *Biometrika* 39 (1952), pp. 274–289.

[147] Y. K. Wong. "An Application of Orthogonalization Process to the Theory of Least Squares". In: *The Annals of Mathematical Statistics* 6.2 (1935), pp. 53 –75. DOI: 10.1214/aoms/1177732609.

[148] A. Zare, A. Ozdemir, M. A. Iwen, and S. Aviyente. "Extension of PCA to Higher Order Data Structures: An Introduction to Tensors, Tensor Decompositions, and Tensor PCA". In: *Proceedings of the IEEE* 106.8 (2018), pp. 1341–1358. DOI: 10.1109/JPROC.2018.2848209.

[149] Q. Zhang, M. W. Berry, B. T. Lamb, and T. Samuel. "A Parallel Nonnegative Tensor Factorization Algorithm for Mining Global Climate Data". In: *Computational Science – ICCS 2009*. Ed. by G. Allen, J. Nabrzyski, E. Seidel, G. D. van Albada, J. Dongarra, and P. M. A. Sloot. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 405–415. DOI: 10.1007/978-3-642-01973-9_45.