KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Computing tensor decompositions from incomplete and implicit data

Stijn Hendrikx

Supervisor:
Prof. dr. ir. Lieven De Lathauwer

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

January 2024

Computing tensor decompositions from incomplete and implicit data

Stijn HENDRIKX

Examination committee:
Prof. dr. ir. Omer Van Der Biest, chair
Prof. dr. ir. Lieven De Lathauwer, supervisor
Prof. dr. ir. Alexander Bertrand
Prof. dr. ir. Stijn Van Cleuvenbergen
Prof. dr. ir. Karim Abed-Meraim
(University of Orléans, France)
Prof. dr. ir. Borbála Hunyadi
(TU Delft, The Netherlands)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

January 2024

© 2024 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Stijn Hendrikx, Kasteelpark Arenberg 10 - box 2446, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Preface

First of all, I would like to thank my promotor Lieven De Lathauwer, not only for his expert guidance during my PhD but also for being the reason I started a PhD in the first place. Near the end of the first year of my master's, I made a selection of my preferred topics for my master's thesis. Surprisingly, I actually received my first choice. The topic concerned tensors, which was, to be honest, something I had never heard of before. I selected it as my preferred option because it was placed at the intersection of my two main interests, namely mathematical engineering and machine learning. It turned out that I had made a good choice because I found tensors and tensor decompositions very interesting indeed. It must have been noticeable to Lieven as well, as he offered me a position as a PhD researcher in his team after I finished my degree.

Before that moment, I had never even really considered a PhD as an option after my master's. Frankly, I did not have a concrete idea of what I wanted to do afterward at all, but I was probably thinking something like "I am going to do a master's after master's because I do not feel like working yet". Very ambitious indeed. Lieven's proposition forced me to consider my options more seriously, and I increasingly began to see a PhD as a very attractive prospect. Because of my master's thesis, I was already familiar with the topic. Furthermore, being offered a position meant that I did not have to apply and was guaranteed to get it if I accepted. So thank you Lieven for offering me a position in your team. I am very grateful for your guidance and the insightful discussions we had regarding my research. Your comments and feedback have vastly improved the quality of my work and my publications. I truly feel like I have become a better researcher during my PhD thanks to you.

Next, I would like to thank the members of my examination committee, Chairman Prof. Omer Van Der Biest, Prof. Alexander Bertrand, Prof. Stijn Van Cleuvenbergen, Prof. Karim Abed-Meraim and Prof. Borbála Hunyadi, for the feedback at my preliminary defense and the comments on my manuscript. Alexander and Stijn, thank you for the feedback on the intermediate presentations and reports during the course of my PhD.

My research received funding from the Flemish Government (AI Research Program). Stijn Hendrikx is affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium. This work was supported by the Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no. G0F6718N (SeLMA). My research received funding from KU Leuven Internal Funds: iBOF/23/064, C14/22/096, C16/15/059 and IDN/19/014.

I have also had the pleasure of being able to travel to various interesting conferences and a winter school, allowing me to meet many other researchers and talk about my research. Not all of my conversations were about my research though and some might have even involved a few drinks. For my first conference, I got to travel to a very chic hotel in Guadeloupe, setting the bar impossibly high for future conferences. Indeed, later conferences did not involve fancy buffets including lobster on the beach of an exotic island, but they too were great in their own right. For the low-rank winter school, I got to travel to Switzerland. Thoughtfully, the organizers left one afternoon free, allowing the attendees to go skiing. However, renting skiing equipment is rather expensive and I still had some work to do so... I went skiing of course. If skiing was the high point of the winter school, then traveling back was the low point as I forgot my backpack, with my work laptop inside, on a bus, only realizing what had happened after the train I just boarded left the station. Luckily my backpack was found and sent back to Belgium.

As a consequence of the pandemic, my next conferences took place online, which made them ever so slightly less enjoyable, but at least I could not lose my laptop anymore. Still, I would certainly have preferred to travel to the USA for Asilomar and actually talk to people rather than sitting silently in front of my webcam listening to my own pre-recorded presentation. Despite the pandemic, I still got to travel a lot, so I cannot complain too much. During the Tricap conference, almost half of the planned talks had to be canceled, meaning we had extra time to spend at the pool surrounded by an amazing landscape. Research-wise, these cancellations were a shame, but they benefitted my sun tan tremendously. My last conference was the most interesting of them all, both in terms of my presentation and the destination itself. There were people at my poster during the entirety of the session and I got to talk about my work non-stop. That certainly beats previous occasions spent standing next to my poster in silence, looking around desperately for someone who pretends to be somewhat interested, only to then trap them into a fifteen-minute conversation about all the smallest details of my poster that they will forget almost instantly. That last part might be slightly exaggerated, but presenting a poster to many people who truly are interested is very satisfying. The conference took place in Hanoi, which allowed me to explore the city and try the local cuisine, unlike previous conferences that were usually in more remote locations. We also got to do a boat trip in the stunning Ha Long Bay. Even though this trip was falsely advertised as being included in the price of the conference, I still gladly paid for it myself as it was easily worth it.

Next, I would like to thank all of my colleagues at ESAT, especially Nico and Martijn who were my supervisors during my master's thesis and the targets of many of my stupid questions during the beginning of my PhD. When Martijn

responded to one of them by sending me a “let me google that for you” link, I started to get the point. Nico and Martijn would gladly fill any document I sent them with annotations full of thorough feedback, which both vastly improved the quality of my work and prepared me for Lieven’s feedback. Your help and guidance was vital to my development as a researcher, so thank you very much. Thank you to all the Jean-luc Doumont evangelists who forced me to reconsider how I designed posters and presentations and made me realize the importance of adapting a presentation to your audience. The lessons I learned from you, and the messiah Jean-luc himself, of course, will be useful to me for the rest of my career. I would also like to thank all of my team members, Nico, Martijn, Rob, Raphaël, Nithin, Michiel, Ignat, Patrick, Eric, Muzaffer, Shakir and Martina for all the help you offered me, the interesting discussions we had and our less formal conversations during coffee breaks or team dinners. Having so much knowledge in the team (and so many targets for my questions) genuinely helped me during my PhD. I would like to thank everyone in Biomed for inviting the tensor team to all of your events, even after we had been exiled from our shared hallway to a new office in the renovated tower. The coffee breaks, board game sessions, nonsensical lunch conversations, receptions and parties we shared were a truly great aspect of my time at ESAT. Thank you to everyone in our very professional futsal team. I have always really enjoyed our games, even though I think you should sometimes just give me a bit more space so that I can shoot with my left from a distance. Simon, I personally blame you for always shouting “left” as soon as I get somewhat close to the goal with the ball at my feet.

I would like to thank all of my friends for their support and for allowing me to relax and forget my troubles when I was close to a deadline. Jullie zijn allemaal gunners en QQ is en blijft van ons. Thank you to my family, for your support, encouragement and the candles you lit during my preliminary defense. Mama en papa, thank you for always being there for me, for all your support and for the opportunities you have given me. Your upbringing made me into the person I am today. Thank you to Stef, I feel like we still would be best friends even if you were not my brother. Last but not least, thank you to Stephanie, the love of my life and, as I am proud to say, my future wife. Given how you have carried me through the difficult parts of my PhD with your support, love, care and encouragement, I am certain that we can manage everything that life throws at us in the future. I would not have been able to complete this journey without you, nor do I want to start any new journey without you. Thank you, “G”.

Stijn Hendrikx
January 2024

Abstract

The prevalence of data in a vast array of domains calls for methods for compression and information extraction. While a dataset is often represented as a matrix, it is more natural to represent a dataset as a tensor if it is higher-order, i.e., it has more than two dimensions. The matrix representation enables the use of linear algebra for purposes such as compression and the extraction of information. However, such linear techniques might not be appropriate for capturing the more complex underlying structure of higher-order data. The tensor representation enables the use of tools from multilinear algebra, namely tensor decompositions. These tools offer possibilities that are not present in their matrix counterparts, such as uniqueness under mild conditions and the ability to break the curse of dimensionality.

Real data often comes with limitations, for example, in an incomplete dataset only a part of the entries is available. In this setting, a tensor decomposition can be fit to the available part of the tensor and then serve as a model for estimating the missing entries, which is referred to as tensor completion. Furthermore, in the case of implicit data, linear combinations of entries are available instead of several individual entries. In compressed sensing, linear combinations of entries are intentionally used for reconstructing a large dataset from a small number of measurements. This is possible if the original data exhibits some underlying structure and tensor decompositions offer one possible way for compactly expressing this structure.

Tensor decompositions have been successfully used for incomplete and implicit data. However, most relevant algorithms from the literature rely on optimization. Algebraic algorithms, which consist of only standard linear algebra operations, present an alternative approach. This type of algorithm has proven useful for computing decompositions of complete tensors, since they yield the exact solution if no noise is present and are often computationally inexpensive. Additionally, the combined approach of initializing an optimization-based algorithm with the solution obtained by an algebraic one tackles the difficult problem of choosing a starting point, while retaining the improved accuracy offered by optimization-based algorithms in the presence of noise. The first main objective of this thesis is to extend these advantages to the incomplete and implicit settings by developing algebraic tensor decomposition algorithms.

The scientific claims that are made in research on tensor decomposition algorithms are underpinned by numerical experiments with synthetic and real data. Therefore, the reproducibility of these experiments is of great importance. For this reason, reproducibility in the field of tensor-related research is the second main objective of this thesis. To this end, a framework for releasing the code and data associated with publications from our team was developed.

Beknopte samenvatting

De alomtegenwoordigheid van data vraagt om methoden voor compressie en de extractie van informatie. Hoewel datasets vaak voorgesteld worden als een matrix, is het natuurlijker om een hogere-orde dataset, dat wil zeggen met meer dan twee dimensies, voor te stellen als een tensor. Een voorstelling als matrix laat toe om alle hulpmiddelen uit de lineaire algebra toe te passen. Maar dergelijke lineaire technieken zijn mogelijk inadequaat om de complexere onderliggende structuur van hogere-orde data te beschrijven. Een voorstelling als tensor laat toe om alle hulpmiddelen uit de multilineaire algebra, namelijk tensorontbindingen, toe te passen. Deze hulpmiddelen bieden mogelijkheden die ontbreken in het lineaire geval, zoals de uniciteit van de ontbinding onder milde voorwaarden en het doorbreken van de vloek van dimensionaliteit.

Echte data heeft vaak beperkingen, zoals een onvolledige dataset die slechts deels beschikbaar is. In tensorvervolledigingwordt het beschikbare deel benaderd met een tensor van lage rang, waarna deze benadering fungeert als een model waarmee het ontbrekende deel wordt ingevuld. Een ander soort beperking is impliciete data, waarbij enkel lineaire combinaties van datapunten uit de dataset beschikbaar zijn in de plaats van individuele datapunten. In gecomprimeerd bemonsteren worden bewust lineaire combinaties opgemeten om de volledige dataset te modelleren aan de hand van een beperkt aantal metingen, gebruik makende van onderliggende structuur in de data. Tensorontbindingen kunnen gebruikt worden om dergelijke structuur op een compacte manier te beschrijven.

Tensorontbindingen zijn succesvol toegepast op zowel onvolledige als impliciete data. De meeste algoritmes uit de literatuur voor beide soorten data maken gebruik van optimalisatie. Algebraïsche technieken, die enkel bestaan uit standaardbewerkingen uit de lineaire algebra, bieden een alternatieve aanpak. Dergelijke algoritmes vinden de exacte oplossing bij ruisloze data en hebben doorgaans een lage rekenkost. Daarnaast is een algebraïsche oplossing ook een goed startpunt voor een optimalisatiealgoritme. Deze gecombineerde aanpak lost het moeilijke probleem van het bepalen van een startpunt op en behoudt daarnaast de betere nauwkeurigheid van een optimalisatiealgoritme in het geval van ruizige data. De eerste doelstelling van deze thesis is om de voordelen van algebraïsche algoritmes voor het berekenen van tensorontbindingen beschikbaar te maken voor onvolledige en impliciete data.

De wetenschappelijke stellingen in publicaties over tensorontbindingen worden onderbouwd door numerieke experimenten met synthetische en echte data. Om die reden is het cruciaal dat deze experimenten reproduceerbaar zijn. De tweede doelstelling in deze thesis betreft de reproduceerbaarheid van onderzoek gerelateerd aan tensoren. Hiervoor werd een raamwerk ontwikkeld om code en data, die horen bij publicaties van ons team, beschikbaar te maken.

Contents

Abstract v

Beknopte samenvatting vii

Contents ix

List of Figures xiii

List of Tables xix

List of Abbreviations xxi

List of Symbols xxiv

1 Introduction 1
- 1.1 Tensor decompositions . . . 4
- 1.2 Incomplete and implicit data . . . 8
  - 1.2.1 Incomplete data and tensor completion . . . 8
  - 1.2.2 Implicit data and compressed sensing . . . 10
- 1.3 Algebraic and optimization-based algorithms . . . 12
- 1.4 Reproducible research . . . 14
- 1.5 Research objectives and chapter overview . . . 15
  - 1.5.1 Objective 1: An algebraic algorithm for computing the MLSVD of a fiber-wise observed tensor . . . 15
  - 1.5.2 Objective 2: Computing tensor decompositions from a block row Kronecker-structured linear system. . . . 17
  - 1.5.3 Objective 3: A framework for reproducible numerical experiments with tensor decompositions . . . 18
  - 1.5.4 Chapter overview . . . 19

2 Tensor decompositions 21
- 2.1 Notation and definitions . . . 21
  - 2.1.1 Indexing, reshaping and permuting tensors . . . 22
  - 2.1.2 Tensor and matrix products . . . 23
- 2.2 Tensor decompositions . . . 24
  - 2.2.1 Canonical polyadic decomposition . . . 24
  - 2.2.2 Multilinear singular value decomposition . . . 26
  - 2.2.3 Tensor train . . . 29

3 Multilinear singular value decomposition based completion with fibers observed in a single mode 31
- 3.1 Introduction . . . 32

3.1.1 Additional notation . . . 34
3.2 The multilinear singular value decomposition . . . 34
3.3 Matrix completion: algebraic conditions for uniqueness . . . 36
3.3.1 Problem definition . . . 38
3.3.2 Uniqueness conditions . . . 39
3.4 Matrix completion: algebraic algorithms . . . 48
3.4.1 An SVD-based kernel subspace method . . . 48
3.4.2 An SVD-based principal subspace method . . . 50
3.5 MLSVD of a fiber-wise observed tensor . . . 53
3.5.1 Algorithm . . . 53
3.5.2 Uniqueness conditions . . . 54
3.6 Numerical experiments . . . 55
3.6.1 Synthetic data experiments . . . 55
3.6.2 Completion of traffic speed data . . . 61
3.7 Conclusion . . . 63
3.8 Proof of Lemma 3.3 . . . 64

4 Block row Kronecker-structured linear systems with a low-rank tensor solution 67
4.1 Introduction . . . 68
4.1.1 Additional notation and definitions . . . 71
4.2 Computing an MLSVD from a BRKS linear system . . . 71
4.2.1 Computing the factor matrices . . . 73
4.2.2 Computing the core tensor . . . 73
4.2.3 Algorithm . . . 76
4.2.4 Conditions for MLSVD retrieval . . . 77
4.2.5 Noisy data . . . 79
4.3 Computing an orthogonally compressed CPD from a BRKS linear system . . . 80
4.3.1 Computing the factor matrices . . . 81
4.3.2 Algorithm . . . 82
4.3.3 Conditions for CPD retrieval . . . 83
4.4 Computing a TT from a BRKS linear system . . . 86
4.4.1 Computing the TT cores . . . 86
4.4.2 Algorithm . . . 88
4.4.3 Conditions for TT retrieval . . . 88
4.5 Experiments . . . 91
4.5.1 Randomized MLSVD . . . 92
4.5.2 Randomized CPD . . . 93
4.5.3 Randomized TT . . . 95
4.5.4 Compressed sensing hyperspectral imaging . . . 97
4.6 Conclusion and further work . . . 99

5 Tensorlab$^+$: A case study on reproducibility in tensor research 101
5.1 Introduction . . . 102
5.2 Tensorlab$^+$ . . . 103
5.2.1 Development guidelines . . . 105
5.2.2 Quality control . . . 106
5.2.3 Lessons learned . . . 107
5.3 Introduction to tensors and decompositions . . . 107
5.4 Example applications of tensor methods . . . 112
5.5 Conclusion . . . 114

6 Conclusion 115
6.1 Overview of contributions . . . 116
6.2 Perspectives on further work . . . 120

A Algebraic and optimization based algorithms for multivariate regression using symmetric tensor decomposition 123
A.1 Introduction . . . 124
A.1.1 Additional notation and basic definitions . . . 124
A.1.2 Polynomials and symmetric tensors . . . 125
A.2 Multivariate regression using PD . . . 125
A.3 Gauss–Newton algorithm . . . 126
A.3.1 Expressions . . . 127
A.3.2 Computational complexity . . . 128
A.4 Algebraic algorithm . . . 129
A.5 Experiments . . . 130
A.6 Conclusion and further work . . . 131

Bibliography 133

List of publications 149

List of Figures

1.1 A tensor is a natural generalization of a vector and matrix to multiway arrays with more than two modes. 1
1.2 A grayscale image corresponds to a matrix of pixel values. . . . 2
1.3 A color image has three values for each pixel, namely for the colors red, green and blue, and thus consists of three matrices. Therefore, a color image can be represented as a third-order tensor by stacking these matrices along the third mode. 3
1.4 A spreadsheet with more than two columns has a multi-way character. Using the table at the top with example data from interactions between people on a social medium, a tensor can be constructed that contains the number of (one-way) interactions between two people on a single day. 3
1.5 Reshaping a tensor into a matrix partially destroys the multi-way structure present in the data. Whereas the same pixel in the red, green and blue channels were adjacent in the tensor along its third mode, they are far apart in the matrix representation. . . 4
1.6 In BSS, the matrix containing the observed signals admits a matrix factorization, in which one of the factors corresponds to the source signals. By factorizing the former matrix, the source signals can be recovered if the factorization is unique. However, a matrix factorization is not unique in general, unless restrictive constraints are placed on the factor matrices. Under such constraints, the source signals correspond to the columns of the first factor matrices. However, this also means that this approach is only viable for applications in which the source signals and mixing vectors can be expected to satisfy these constraints, which limits the use of matrix factorizations in BSS. 5
1.7 Top: Low-rank matrices can be represented as a sum R of rank-1 matrices, where R is the rank of the matrix. A rank-1 matrix can be expressed as the outer product of two nonzero vectors. Bottom: Similarly, a rank-1 Nth order tensor can be expressed as the outer product of N nonzero vectors. The CPD expresses a tensor as a minimal sum of rank-1 tensors, and its mild uniqueness conditions can be leveraged to retrieve the source signals in BSS, without requiring that the source signals satisfy restrictive constraints. . 6

1.8 In both decompositions, the factors matrices $\mathbf{U}, \mathbf{V}$ (and $\mathbf{W}$) are column-wise orthonormal and form a basis for the subspace spanned by the vectors along their corresponding mode. Therefore, the MLSVD can be used for subspace analysis. Like $\mathbf{\Sigma}$ contains the singular values of $\mathbf{M}$, the multilinear singular values are determined by $\boldsymbol{\mathcal{S}}$, which will be explained in more detail in Chapter 2. The multilinear singular values enable the use of the MLSVD for compression and higher-order principal component analysis. 7
1.9 Completing a dataset with inherently missing entries allows one to make predictions for unobserved combinations of variables. In the Netflix Prize example, these predictions can be used to make movie recommendations to a user. 10
1.10 In the CS setting, the data is compressed through multiplication with the measurement basis to a smaller vector of more informative samples. With these samples, the original data can be closely approximated in a compact format if the data contains some underlying structure. For example, a compact representation can be obtained if the data is sparse when expressed in a proper basis or if it is approximately low-rank. In this thesis, low-rank tensors are used as a compact representation for a dataset. 13
1.11 The first objective concerns a specific pattern of available tensor entries. Instead of the available entries being spread throughout the tensor (left), the tensor has been observed in vectors along the same mode (right). This observation pattern enables the algebraic computation of the MLSVD. 17
1.12 Compressing each mode of the tensor individually corresponds to a linear system with a Kronecker-structured coefficient matrix. Note that $\otimes$ denotes the Kronecker product. 18
1.13 Vertically stacking multiple Kronecker-structured matrices yields a BRKS matrix. An underdetermined system with a BRKS coefficient matrix can be solved for a solution expressed as an MLSVD or TT in an algebraic manner. 18

2.1 Selecting a part of a tensor is denoted with the indices as subscripts. A colon : as an index indicates that no selection is performed along the mode that corresponds to the position of the colon. 22
2.2 The CPD generalizes the sum of rank-1 terms representation of a matrix decomposition to tensors of higher order. 25

2.3 Top: The MLSVD generalizes some core concepts of the SVD to higher-order tensors. Bottom: A good, but not necessarily optimal, LMLRA can be obtained by truncating the MLSVD, that is, retaining only the dark blue parts of the factor matrices and core tensor. 28
2.4 The TT consists of second- and third-order tensors, subsequently joined together by contractions over a common index, and therefore does not suffer from the curse of dimensionality. . . . 30

3.1 Example of a fiber-wise observed tensor and the corresponding mask. 36
3.2 The mode-2 unfolding of the fiber-wise observed tensor in Fig. 3.1 consists of fully observed submatrices that are potentially overlapping. There are five fully observed submatrices that are formed by the observed rows of the mode-1 slices $\mathbf{X}_{i::}$ for $i = 1, \ldots, 5$. There are also two larger fully observed submatrices that span across multiple slices, namely the submatrices marked by horizontal and vertical lines. 37
3.3 Accuracy for the fiber-wise observed tensor with a block diagonal mask $\mathbf{W}$ in the first experiment. The algorithm `mlsvd_fs`, as it is algebraic, obtains less accurate results than the optimization-based `geomCG` and `tucker_opt` algorithms; however, the difference in accuracy is limited (left). Algorithm `HaLRTC` is very inaccurate for this observation pattern. However, for an incomplete tensor with the same sampling ratio, the accuracy improves if the observed part consists of randomly selected fibers or randomly selected entries (right). Similarly, `tucker_opt` is more accurate at higher values of the SNR if the observed fibers are selected randomly. 57
3.4 Accuracy for the fiber-wise observed tensor with a random mask $\mathbf{W}$ in the second experiment. At a sampling ratio of 10% the working conditions of `mlsvd_fs` are not yet satisfied. For higher sampling ratios, the accuracy obtained by `mlsvd_fs` is close to that obtained by the optimization-based algorithms. 58

3.5 Comparison of random and algebraic initialization of `geomCG` in terms of accuracy (left) and number of iterations (right). In the case where the multilinear rank is overestimated, the optimization-based algorithm has difficulty finding good solutions, starting from 100 different random initializations. In the case where it is correctly estimated or underestimated, both initialization strategies yield equally good results in terms of accuracy. However, the algebraic initialization is still useful, since it significantly reduces the number of iterations required to reach this level of accuracy. 59
3.6 The computation time required by `geomCG` and `tucker_opt` to reach the same level of accuracy as `mlsvd_fs` is respectively about two and three orders of magnitude larger than the computation time of `mlsvd_fs` for the tensor dimensions I that were tested in this experiment. 61

4.1 The mth block row of the linear system in Eq. (4.3) produces a version of $\boldsymbol{\mathcal{X}}$ that is compressed in each mode except the mth mode. Each compressed tensor can be used to retrieve one factor matrix of the MLSVD, which is similar to the RTD algorithm in [122]. 74
4.2 Algorithm `mlsvd_rsi` is by far the most accurate because it used the full-sized matrix unfolding in the subspace iteration step. Algorithm `lsmlsvd_brks` is more accurate than `lmlra_aca` and `rand_tucker` and less accurate than `rand_tucker_kron`. If the sequential truncation step in `rand_tucker_kron` is omitted, it achieves the same accuracy as `lsmlsvd_brks`. 94
4.3 Increasing the oversampling factor q of `lscpd_brks`, for which the value is shown in the figure, greatly improves its accuracy while having a negligible effect on the computation time. Algorithm `lscpd_brks` is faster than `cp_arls` and `cp_rbs`, for a range of tensor dimensions that are relevant for applications, with a limited loss of accuracy. 96
4.4 Algorithm `lstt_brks` enables us to achieve accuracy similar to that of `rand_tt` in CS applications. If a method for obtaining any entry of $\boldsymbol{\mathcal{X}}$ is available, then `cross_tt` can be used to recover a more accurate approximation of the tensor, in exchange for a longer computation time. 97

5.1 In the full archive, the code for each publication is located in a separate folder. The name of this folder consists of the last name of the first author, the publication year and a keyword for convenience. 105

5.2 Graphical representations of basic tensor decompositions. . . . 109
5.3 The comparison between the accuracy of both algorithms is not representative of the actual difference in accuracy when the results are aggregated (using the median) over one or five problems. Therefore, the experimental results in Tensorlab$^+$ are aggregated over plenty of problems to ensure robustness and to give a representative estimate of the actual accuracy of the algorithms. 114

A.1 By applying a process called homogenization, symmetric tensors can also represent the coefficients of non–homogeneous polynomials. For example, by stacking the coefficients $t^{(0)}, \mathbf{t}^{(1)}, \mathbf{T}^{(2)}$ and $\boldsymbol{\mathcal{T}}^{(3)}$ of a degree-3 polynomial into a tensor as shown above, we can represent it with a symmetric third-order tensor. Image reproduced from [11]. 125
A.2 A high accuracy is achieved by both methods when modeling G_{p}, even for few samples. The algebraic method fails for $M < 35$. Median over 50 experiments. 131
A.3 Biased sampling improves accuracy when modeling G. The algebraic method is a bit more accurate, but it fails for $M < 120$. Median over 50 experiments. 131

List of Tables

3.1 Unlike in the experiments with synthetic data, filling in the missing fibers of this traffic speed dataset yields a far smaller difference in accuracy between `mlsvd_fs` and `geomCG`. Additionally, the accuracy achieved by `mlsvd_fs` is comparable to that of the baseline algorithms. 63

4.1 This table shows the reconstruction quality, quantified in PSNR (dB), of a hyperspectral image for a range of sampling ratios, obtained with different algorithms. Algorithm `lsmlsvd_brks` performs approximately equally well as `GAP_TV`, which is an algorithm specifically suited for image reconstruction. The lower reconstruction quality for `KCS` indicates that the bases estimated by `lsmlsvd_brks` suit the data better than the a priori determined bases used in `KCS`. 100

A.1 By carefully exploiting the Khatri–Rao structure and the symmetry in Eq. (A.4), we can obtain a significant improvement in the computational complexity. 128

List of Abbreviations

ALS	alternating least squares
BRKS	block row Kronecker-structured
BSS	blind source separation
BTD	block term decomposition
CASSI	coded aperture snapshot spectral imager
CPD	canonical polyadic decomposition
CS	compressed sensing
GEVD	generalized eigenvalue decomposition
GN	Gauss–Newton
JLT	Johnson-Lindenstrauss transform
KCS	Kronecker compressed sensing
LMLRA	low multilinear rank approximation
LS-CPD	linear system with a polyadic decomposition constrained solution
MLSVD	multilinear singular value decomposition
PD	polyadic decomposition
PSNR	peak signal-to-noise ratio
RTD	randomized tensor decomposition
SNR	signal-to-noise ratio
SVD	singular value decomposition
TR	trust region
TT	tensor train

List of Symbols

Unary

$a, b, \ldots$	scalar
$\mathbf{a}, \mathbf{b}, \ldots$	vector
$\mathbf{A}, \mathbf{B}, \ldots$	matrix
$\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \ldots$	tensor
N	order of a tensor
$I \times J \times K$	dimensions of a third-order tensor
$I_1 \times I_2 \times \cdots \times I_N$	dimensions of an Nth order tensor
$.^{(n)}$	nth element in a sequence of vectors, matrices or tensors
a_{ijk}	entry at position (i, j, k) of a third-order tensor $\boldsymbol{\mathcal{A}}$
:	colon as subscript indicates no selection is performed along the corresponding mode
$\mathbf{a}_j$	shorthand for $\mathbf{a}_{:j}$, the jth column of $\mathbf{A}$
$\boldsymbol{\mathcal{A}}_{[2;4,1;3]}$	example of a tensor obtained by permuting and combining modes of the original tensor $\boldsymbol{\mathcal{A}}$, where ; indicates a new mode and the indices determine the new order of the modes of the original tensor
$\mathbf{A}_{[n]}$	shorthand notation for the mode-n unfolding of a tensor $\mathbf{A}_{[n;1,\ldots,n-1,n+1,\ldots,N]}$
$\text{vec}\,(\boldsymbol{\mathcal{A}})$	reshapes $\boldsymbol{\mathcal{A}}$ into the vector $\mathbf{a}_{[N,\ldots,2,1]}$
$\mathbb{R}$	real field
$\mathbb{C}$	complex field
$\|\|\cdot\|\|_{\text{F}}$	Frobenius norm
$.^{\text{T}}$	transpose
$.^{*}$	conjugate
$.^{\text{H}}$	conjugate or Hermitian transpose
$.^{-1}$	inverse

$.^{\dagger}$	Moore–Penrose pseudoinverse
$\mathbf{I}_I$	identity matrix of dimensions $I \times I$
$\mathbf{1}_I$	vector of ones with length I
$\mathrm{r}(\cdot)$	matrix rank
$\mathrm{range}(\cdot)$	column space or range
$\ker(\cdot)$	nullspace or kernel
$\alpha, \beta, \ldots$	set
$.^{\mathrm{c}}$	set complement
$\mathrm{card}(\cdot)$	cardinality of a set

Binary

$\otimes$	Kronecker product
$\odot$	Khatri–Rao product
$\odot^{\mathrm{T}}$	transposed Khatri–Rao product
$\cdot_n$	mode-n tensor matrix product
$\otimes$	outer product
$*$	element-wise/Hadamard product

Other

$[\![\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}]\!]$	polyadic decomposition with factor matrices $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}$
$[\![\boldsymbol{\mathcal{S}}; \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}]\!]$	low multilinear rank approximation with factor matrices $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}$ and core tensor $\boldsymbol{\mathcal{S}}$
$(\!\|\boldsymbol{\mathcal{G}}^{(1)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)}\|\!)$	tensor train with cores $\boldsymbol{\mathcal{G}}^{(1)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)}$

1 Introduction

Tensors are multi-way arrays of numerical values and can be found everywhere if you know what to look for. They are generalizations of vectors and matrices, i.e., one- and two-way arrays, as illustrated in Fig. 1.1. The term tensor is generally used for multiway arrays of order three or more, where the order corresponds to the number of modes N of an N-way array[1]. A matrix is a common way of representing data, since every table and spreadsheet is a matrix. A grayscale image is a rectangle of pixels and each pixel corresponds to one value, meaning that it corresponds to a matrix, see Fig. 1.2. Representing data as a matrix enables the use of all the tools that linear algebra has to offer. For example, a large matrix can be compressed while maximizing the amount of preserved information using principal component analysis. With linear regression, a linear dependency between independent and dependent variables can be modeled. With factorizations such as the singular value decomposition or the eigenvalue decomposition, patterns can be extracted from a data matrix.

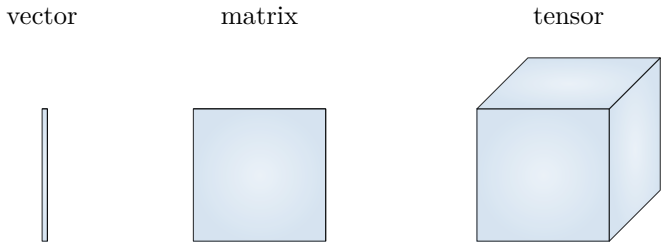


Figure 1.1: A tensor is a natural generalization of a vector and matrix to multiway arrays with more than two modes.

However, nowadays a grayscale image is outdated. A color picture represented in the RGB format, as illustrated in Fig. 1.3, has three values for each pixel, meaning that it has an additional mode and thus is a third-order tensor. A color is quantified as a wavelength of an electromagnetic wave, so this third mode adds a spectral dimension to the tensor. While there are only three wavelengths

[1] An N-way array has N modes and the dimensions of a tensor correspond to the size of the tensor along each mode.

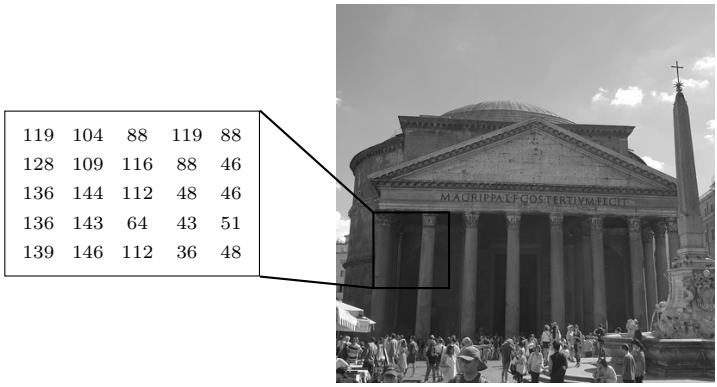


Figure 1.2: A grayscale image corresponds to a matrix of pixel values.

in a color image, a hyperspectral image consists of many more wavelengths. A video is a sequence of color images, so the corresponding tensor has an extra temporal dimension. Even spreadsheets, although they are two-way arrays, also have a multi-way character if they contain more than two columns, because each column corresponds to a different variable. If each variable is used as a separate mode, then the data naturally corresponds to a tensor. For example, consider a table that consists of interactions between persons on a social medium. Each row corresponds to one interaction and the columns are "Person 1" (the person who initiates the interaction, "Interaction Type", "Person 2" and "Date". This table can also be represented using a third-order tensor, in which the modes correspond to the columns of the table, as illustrated in Fig. 1.4. An entry of the tensor equals the number of interactions between two persons on a specific day. Very similar tensors can be constructed from, for instance, academic paper citations, internet traffic data or financial transactions [1, Table 2]. In general, tensors are prevalent in a wide range of domains and applications [2]–[5].



Figure 1.3: A color image has three values for each pixel, namely for the colors red, green and blue, and thus consists of three matrices. Therefore, a color image can be represented as a third-order tensor by stacking these matrices along the third mode.

Person 1	Interaction type	Person 2	Date
Stijn	Like	Stephanie	10/10
Stijn	Comment	Stephanie	10/10
Stef	Comment	Stephanie	10/10
Stephanie	Comment	Stijn	10/10
Stijn	Like	Stef	11/10
Stephanie	Like	Stef	11/10
Stephanie	Comment	Stef	11/10
Stephanie	Comment	Stef	11/10
Stephanie	Like	Stijn	11/10

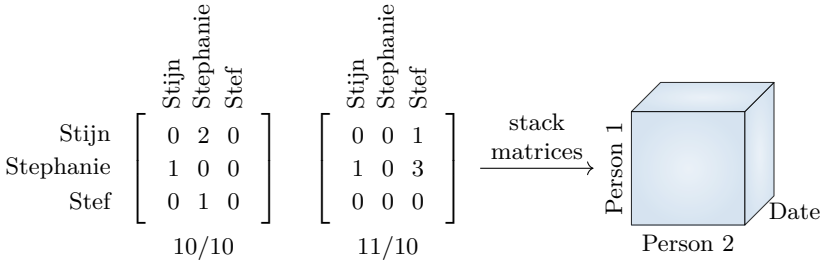


Figure 1.4: A spreadsheet with more than two columns has a multi-way character. Using the table at the top with example data from interactions between people on a social medium, a tensor can be constructed that contains the number of (one-way) interactions between two people on a single day.

To extract useful information, linear algebra tools can be used if the tensor is first reshaped into a matrix. However, this destroys the multi-way structure present in the data, as illustrated in Fig. 1.5. Similarly, consider a grayscale video that is reshaped from a third-order tensor into a matrix such that subsequent frames are placed next to each other along the second mode. Therefore, the second mode of the resulting matrix has both a spatial and a temporal component. Whereas values for the same pixel at subsequent timestamps were adjacent to each other, this is no longer the case after reshaping. It is thus difficult to extract temporal patterns using matrix tools and this is disadvantageous since video has significant temporal redundancy, which is exploited by video compression algorithms. To keep the multi-way structure of data intact, the data should be processed as a tensor.



Figure 1.5: Reshaping a tensor into a matrix partially destroys the multi-way structure present in the data. Whereas the same pixel in the red, green and blue channels were adjacent in the tensor along its third mode, they are far apart in the matrix representation.

1.1 Tensor decompositions

With the multi-way structure of data remaining intact by representing it as a tensor, this structure can then be modeled or approximated using tensor decompositions. As tensor decompositions can be considered as generalizations of matrix decompositions to higher orders, the former can be used for similar purposes as the latter. However, tensor decompositions offer possibilities that are not present in their matrix counterparts, which can make them even more powerful. In blind source separation (BSS), the goal is to identify source signals given a set of observed signals, which are mixtures of these source signals. A possible method for recovering the source signals and the mixing coefficients is by factorizing the matrix containing the observed signals, as illustrated in Fig. 1.6. To guarantee that the factorization has indeed recovered the original source signals, it must be unique. However, without imposing constraints on the factor matrices, a matrix factorization is generally not unique. This means

that the matrix factorization approach can yield infinitely many possible sets of source signals, with no way of identifying which set corresponds to the original source signals. Constraints such as orthogonality or nonnegativity can be used to make the components of a matrix decomposition interpretable. For example, the singular value decomposition (SVD) imposes orthogonality on the factor matrices and is unique[2], but such constraints are often too restrictive for the source signals in real BSS applications.

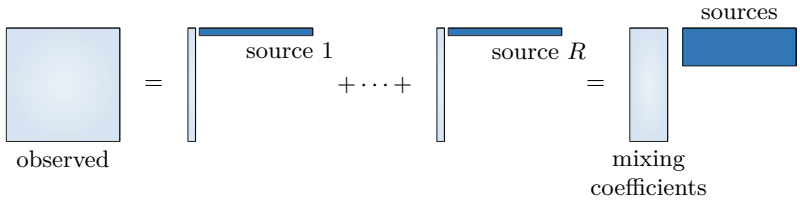


Figure 1.6: In BSS, the matrix containing the observed signals admits a matrix factorization, in which one of the factors corresponds to the source signals. By factorizing the former matrix, the source signals can be recovered if the factorization is unique. However, a matrix factorization is not unique in general, unless restrictive constraints are placed on the factor matrices. Under such constraints, the source signals correspond to the columns of the first factor matrices. However, this also means that this approach is only viable for applications in which the source signals and mixing vectors can be expected to satisfy these constraints, which limits the use of matrix factorizations in BSS.

A tensor decomposition such as the canonical polyadic decomposition (CPD) offers uniqueness under mild conditions, that is, uniqueness of the decomposition is possible without requiring restrictive constraints on the factor matrices like in the matrix case. As illustrated in Fig. 1.7, this decomposition expresses a tensor as a minimal sum of rank-1 tensors. For example, a particular third-order CPD is unique if the first two factor matrices have full column rank and the third one has no colinear columns [4] and conditions that are even less restrictive are available in literature [6], [7]. For a general tensor, given that it satisfies some assumptions[3], there are uniqueness conditions for the CPD that simply reduce to an inequality condition involving only the dimensions and the rank of the tensor [8]. Therefore, CPD-based techniques can be used in BSS for recovering source signals that do not satisfy restrictive constraints such as orthogonality or nonnegativity, which makes the decomposition a successful tool in this field.

[2]The SVD is unique up to rotational invariances of the singular vectors corresponding to repeated singular values.

[3]The assumption is that the factor matrices are generic. An example of a generic uniqueness condition for the CPD will be given in Section 2.2.1.

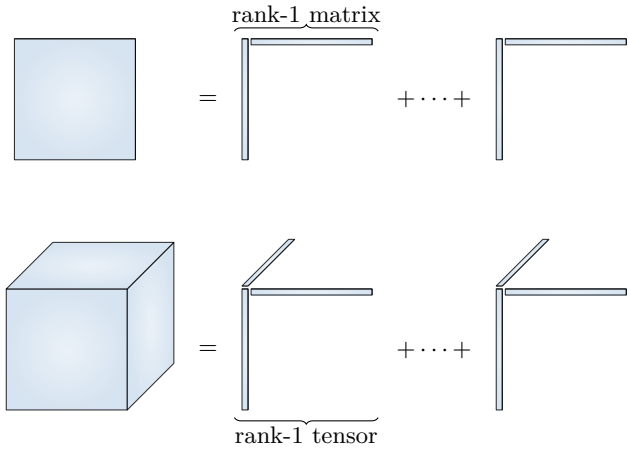


Figure 1.7: Top: Low-rank matrices can be represented as a sum R of rank-1 matrices, where R is the rank of the matrix. A rank-1 matrix can be expressed as the outer product of two nonzero vectors. Bottom: Similarly, a rank-1 Nth order tensor can be expressed as the outer product of N nonzero vectors. The CPD expresses a tensor as a minimal sum of rank-1 tensors, and its mild uniqueness conditions can be leveraged to retrieve the source signals in BSS, without requiring that the source signals satisfy restrictive constraints.

As the number of modes of data increases, the size of the corresponding tensor grows exponentially. Namely, if each dimension of a tensor is equal to I and its order is N, then it contains I^N entries. Because of this exponential dependency, storing and processing a tensor quickly becomes prohibitively expensive in terms of memory and computing power requirements as the order increases. This problem is commonly referred to as the "curse of dimensionality" and tensor decompositions offer a possible solution. Tensor decompositions can alleviate the curse, if they can be expressed using a number of variables that is far smaller than I^N, or even completely break the curse if, for example, the number of variables scales only linearly with N. This is possible because data often possesses some inherent structure, which can be exploited to obtain compact representations of the data. If a matrix has a low rank, meaning that the rank is far smaller than the dimensions of the matrix, a matrix decomposition serves as such a compact representation. As illustrated in Fig. 1.7, a low-rank matrix can be represented using only $2R$ vectors. Similarly, a tensor decomposition is a compact representation of a tensor with low rank[4], for example, the CPD

[4]Note that the definition of rank is different depending on the considered tensor decomposition.

consists of NR vectors, meaning that the storage requirements scale linearly with the order N and the curse is indeed broken.

While real data is often not exactly low-rank, it is often approximately low-rank because it has underlying low-dimensional structure [9]. In the matrix case, this can be observed from the singular values: if they decay to zero quickly, then the optimal (in the least squares sense) low-rank approximation can be obtained by truncating the SVD such that only the significant singular values are retained. Similarly, a pattern of decaying multilinear singular values indicates that a tensor could be closely approximated by a low multilinear rank tensor. The multilinear singular values are determined by the multilinear singular value decomposition (MLSVD) of a tensor. As illustrated in Fig. 1.8, this decomposition generalizes some properties of the SVD to higher-order tensors. In some applications, the rank has physical meaning and is known to be small compared to the dimensions of the tensor. For example, in BSS the rank corresponds to the number of source signals. Because of the prevalence of approximately low-rank data, tensor decompositions have been used for purposes such as compression or recovering different components in a large variety of fields, for example, signal processing, machine learning and data analysis [2]–[4]. A detailed introduction to the tensor decompositions that are relevant in this thesis follows in Chapter 2.

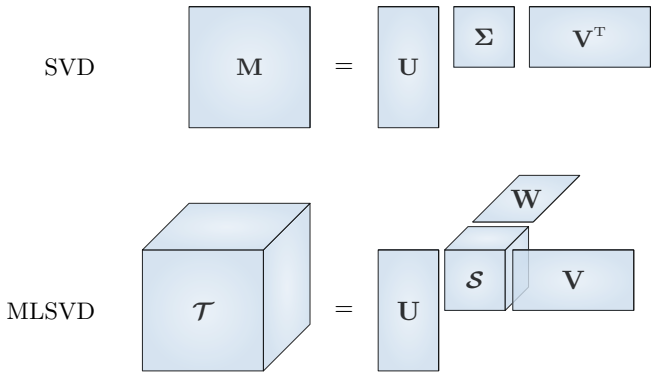


Figure 1.8: In both decompositions, the factors matrices $\mathbf{U}, \mathbf{V}$ (and $\mathbf{W}$) are column-wise orthonormal and form a basis for the subspace spanned by the vectors along their corresponding mode. Therefore, the MLSVD can be used for subspace analysis. Like $\mathbf{\Sigma}$ contains the singular values of $\mathbf{M}$, the multilinear singular values are determined by $\boldsymbol{\mathcal{S}}$, which will be explained in more detail in Chapter 2. The multilinear singular values enable the use of the MLSVD for compression and higher-order principal component analysis.

In order to benefit from the possibilities offered by tensor decompositions, data that only has two modes may be tensorized. This means that the data is

reshaped or transformed into a higher-order tensor in a meaningful way such that the resulting tensor is approximately low rank. For example, in BSS the observed mixtures of source signals correspond to a matrix. This matrix can be tensorized in a way such that the resulting tensor can be expressed as a sum of low multilinear rank components, where each component corresponds to one source signal. The source signals can then be retrieved by leveraging the mild uniqueness conditions offered by some tensor decompositions[5] [10]. A general overview of tensorization techniques and their applications can be found in [11].

1.2 Incomplete and implicit data

In the most straightforward case, a data tensor is explicitly given and complete. In this context, explicit means that individual entries[6] of the tensor are given and complete means that all of its entries are available. While many applications fall into the straightforward case, data is often incomplete or available implicitly. In an incomplete tensor only a subset of the entries is available. A tensor that is implicitly available means that only linear combinations of the entries of the tensor are available, as opposed to individual entries being available. In this thesis, the emphasis lies on incomplete and implicitly available data.

1.2.1 Incomplete data and tensor completion

There are two main reasons for the incompleteness of data:

1. A dataset is inherently incomplete if some combinations of the independent variables are not available. A well-known example is the movie rating dataset that was used in the Netflix Prize [12]. Here, the independent variables are the users and the movies. Since users generally have not watched and rated all the available movies, there are no ratings available for some combinations of users and movies and the data is incomplete. In the case of incomplete tensor data, tensor completion, that is, completing a dataset by estimating the missing entries, is very useful for making predictions for new combinations of independent variables. Indeed, predicting the ratings of users for movies they have not yet seen was the main purpose of the Netflix Prize, since this allows one to recommend movies that they will likely enjoy.

[5]The decomposition in question is the block term decomposition, which expresses a tensor as a sum of low multilinear rank components. Similar to the CPD, it is unique under mild conditions.

[6]As opposed to linear combinations of entries in the implicit case.

2. A dataset can be incomplete because it is too expensive or impractical to obtain every entry. Datasets that are extremely large, either because the order is high or because its dimensions are large, might be too expensive in terms of storage requirements. In this case, only obtaining a subset of its entries and completing the missing entries serves as a proxy for the entire dataset. As a result, this proxy can be obtained at a cost that is far lower than the cost of obtaining the complete dataset. Alternatively, such a large dataset might actually be complete, but processing it in its entirety might be computationally prohibitive. For example, in stochastic gradient descent, which is a very popular algorithm for training models in machine learning, the gradient is estimated in each iteration using a subset of the full dataset. While the number of variables of some tensor decompositions does not suffer from the curse of dimensionality, the curse is still present if the full tensor is used during the computation of this decomposition. As a result, the curse can only be broken by computing the decomposition using only a subset of the entries of the tensor [13]–[17]. Even for a dataset with a manageable size, it can still be advantageous to only measure some of its entries if measurements are expensive.

Aside from these two main reasons, there can be other reasons for incompleteness. For example, parts of data that are heavily distorted by noise can be handled by considering these parts to be missing.

In both cases, tensor completion is a very powerful tool, namely for making predictions or for efficiently computing a compact approximation of a large dataset. In general, filling in missing values for making recommendations is common in applications such as video or music streaming services, e-commerce and social networks and the general class of techniques for this purpose are recommender systems [18]. Additionally, data might be inherently incomplete because of sensor failures or other types of distortion, such as dead pixels in visual data [19] or cloud coverage in hyperspectral imaging [20]. A comprehensive survey of tensor completion algorithms and applications can be found in [21].

Tensor completion is made possible by the same property that makes tensor decompositions so useful, namely that most data possesses structure and therefore might be approximately low rank. In tensor completion, an (approximately) low-rank tensor is computed that closely approximates the available entries. This low-rank tensor can then be used as a model for estimating the missing entries. Intuitively, this can be explained by the fact that a low-rank tensor can be compactly represented by a tensor decomposition, which can be described by a number of variables that is far smaller than the total number of entries in the tensor. Therefore, a subset of the entries of the tensor is sufficient to estimate these variables. The two main approaches to tensor completion are:

1. Minimizing the rank, while keeping the approximation error between the completed tensor and the available entries below a tolerance [22]–[24]. However, since minimizing the rank of a matrix or tensor is an NP-hard problem [24], a convex relaxation of rank is minimized instead.
2. Minimizing the approximation error while fitting a tensor decomposition with a fixed rank to the incomplete tensor [13], [16], [25], [26].

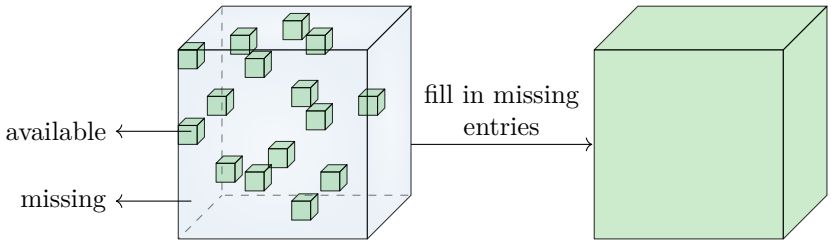


Figure 1.9: Completing a dataset with inherently missing entries allows one to make predictions for unobserved combinations of variables. In the Netflix Prize example, these predictions can be used to make movie recommendations to a user.

1.2.2 Implicit data and compressed sensing

Linear combinations of the entries of a dataset are an important part of compressed sensing (CS). CS techniques allow one to reconstruct a dataset using a number of measurements that is far smaller than the required number of measurements in traditional methods [27]–[29]. In signal processing, CS techniques facilitate the reconstruction of signals that are sampled below the Nyquist rate [30]. In general, that is, in fields other than signal processing, CS allows one to solve underdetermined linear systems. This means that the number of linear combinations on the right-hand side of the system is far smaller than the number of entries in its solution. This powerful property can, for example, be used to reconstruct images that have more pixels than the number of measurements obtained by the imaging device [31]–[34]. By replacing the lens in a standard camera with an optical encoder, the device can obtain more informative measurements that facilitate CS [35]. Furthermore, CS has been applied in, for example, radar, remote sensing and channel estimation [36], [37].

CS rests on two key assumptions:

1. The dataset has an underlying low-dimensional structure and therefore has a compact representation that closely approximates it [38]. In the

early work on CS, sparsity is the standard way for compactly representing low-dimensional structure [27], [28]. In this context, sparsity means that the coefficients of the data are approximately sparse when expressed in the proper basis, that is, only a fraction of the coefficients is significantly large. The basis that renders the data approximately sparse is referred to as the sparsifying basis. A compact representation that closely approximates the original data can be obtained by retaining only the significant coefficients of the representation in the sparsifying basis and discarding the rest. In later work on CS, the notion of sparsity was extended to more general compact representations of data. For example, low-rank tensors, expressed as tensor decompositions, are used as compact representations in this thesis.

2. The measurements, that is, the linear combinations of entries, are sufficiently informative for reconstructing the original data. This means that the matrix containing the coefficients of the linear combinations of entries, referred to as the measurement basis, should be incoherent with respect to the sparsifying basis. This means that the measurement basis vectors should not be highly correlated to any sparsifying basis vectors. If a highly correlated pair of measurement and sparsifying basis vectors does exist, then that measurement basis vector produces a measurement that is dominated by the coefficient corresponding to the sparsifying basis vector it is highly correlated with. As these coefficients are sparse, the information content of this measurement is likely to be low. Incoherence thus guarantees that each linear combination contains information on the coefficients belonging to (almost) all the sparsifying basis vectors.

If both assumptions hold, then a compact and accurate representation of the dataset can be obtained from a relatively small number of measurements, as illustrated in Fig. 1.10. The intuition behind CS is that, using informative measurements, one can directly estimate a compact representation of the dataset, without acquiring the dataset in its entirety. This compact representation is described by a number of variables that is far smaller than the total number of entries in the dataset, which is why a small number of informative measurements is sufficient. The number of measurements must at least equal the degrees of freedom of the underlying structure of the data. A standard approach for processing large datasets is to compress them before performing any other steps, which reduces the computational cost down the line. Using CS, it is possible to skip the step in which the full dataset is constructed and then compressed because this approach immediately yields a compressed version of the original data.

Tensor decompositions are useful for CS in applications in which the underlying structure of the data has a higher-order character, which cannot be captured adequately as the linear combination of the basis vectors of some (sparsifying) basis. They have been successfully applied in this context in magnetic resonance imaging, hyperspectral imaging, compressive imaging, video compression and channel estimation [39]–[42]. A common approach for higher-order data in CS is to use the Kronecker product of multiple bases, namely one for each mode of the tensor, as the sparsifying basis [39], [41], [43]. This approach corresponds to using a Tucker decomposition, which is like the MLSVD in Fig. 1.8 but without constraints imposed upon the factor matrices and core tensor, as a compact representation of the data. Each factor matrix of this decomposition then corresponds to one of the bases in the Kronecker product and the core tensor functions as the coefficients. Therefore, tensor decompositions are a better compact representation of data with a higher-order character.

Note that tensor completion can also be regarded as a CS technique. A low-rank tensor expressed as a tensor decomposition plays the part of the compact representation of the data. Instead of linear combinations of entries, the measurements in tensor completion consist of a subset of the entries of the dataset. Incoherence assumptions are also present to guarantee that the available entries are sufficiently informative [44]. This ensures that the information content of the matrix is not concentrated in a few significant entries, while the other entries are negligibly small in comparison. Otherwise, the majority of the entries are not informative.

Many problems in scientific computing can be expressed as a linear system with a tensor solution, in which the tensor follows from the discretization of a multivariate function on a higher-order grid. For example, such linear systems emerge when solving high-dimensional parameter-dependent partial differential equations for discretized values of these parameters [45] or evaluating integrals on a grid of interpolation points [46]. As the solution of these large systems can suffer from the curse of dimensionality, they are often solved for a compact representation of the solution instead. Tensor decompositions may be used as a compact approximation of the solution if the latter possesses some inherent structure. A general overview of applications in scientific computing that involve solving a large linear system for a structured solution is given in [47].

1.3 Algebraic and optimization-based algorithms

A standard way for computing tensor decompositions is through the minimization of a cost function, which consists of the approximation error between

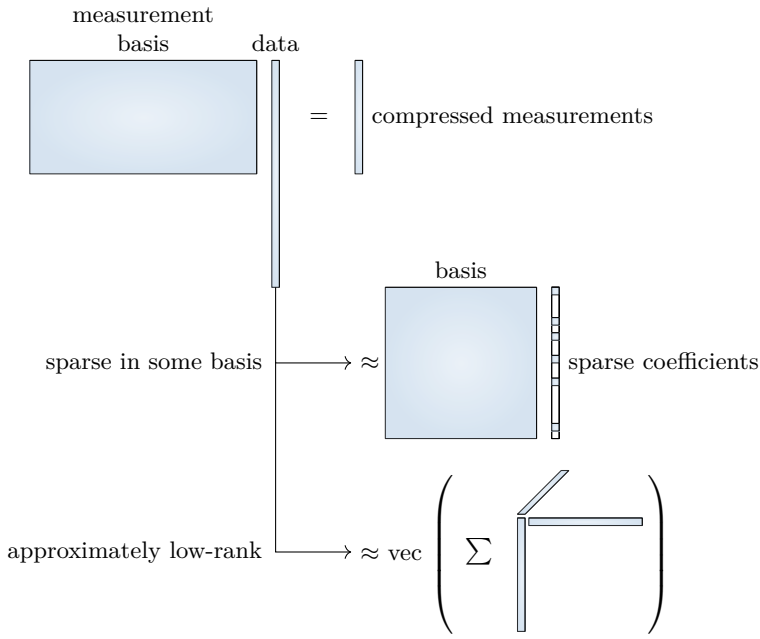


Figure 1.10: In the CS setting, the data is compressed through multiplication with the measurement basis to a smaller vector of more informative samples. With these samples, the original data can be closely approximated in a compact format if the data contains some underlying structure. For example, a compact representation can be obtained if the data is sparse when expressed in a proper basis or if it is approximately low-rank. In this thesis, low-rank tensors are used as a compact representation for a dataset.

a tensor and the decomposition, using an optimization algorithm. Such an algorithm requires a starting point, which is then iteratively improved. The number of iterations until convergence depends on the quality of the starting point, that is, fewer iterations are needed if the starting point is closer to the global minimum of the cost function. The cost functions that arise when computing tensor decompositions are generally non-convex, meaning that the cost function might have multiple minima that are not necessarily global minima. The choice of the starting point is even more important in the case of non-convex problems because it might also determine whether the algorithm converges to a global or local minimum.

If a tensor is completely available, then other approaches for computing some tensor decompositions are possible. These algorithms are referred to as algebraic,

because they only consist of standard linear algebra operations, such as solving linear systems of equations or computing matrix decompositions like the SVD. In contrast to optimization-based algorithms, an algebraic algorithm does not need to be provided with a starting point. For example, the CPD can be computed using a generalized eigenvalue decomposition or the MLSVD can be computed using multiple SVDs. Algebraic tensor decomposition algorithms yield the exact solution if the tensor is of low rank and contains no noise, making these algorithms already very useful on their own. However, data from real applications often does contain noise. In this case, the solution obtained by an algebraic algorithm can be further improved using an optimization algorithm. This approach combines the advantages of both: the starting point is obtained using the algebraic algorithm and the optimization algorithm requires fewer iterations and is less likely to yield a local minimum. The reduced number of iterations achieved through algebraic initialization will generally result in a net gain in terms of computation time because fast and stable algorithms are available for most standard linear algebra operations. Furthermore, a common approach to finding a good starting point is to try many different random ones and keeping the best solution. By using an algebraic algorithm as a better initialization strategy, one starting point often suffices, meaning that the reduction in computation time compared to the random strategy is even more significant.

While many algebraic algorithms are available for complete and explicitly given tensors, this is not the case for incomplete or implicitly given tensors. The work in this thesis aims to provide algebraic approaches for the incomplete and implicit setting, in order to make the advantages offered by this type of algorithm more easily available in this context.

1.4 Reproducible research

Most publications on tensor decomposition algorithms include numerical experiments in order to illustrate the performance of the algorithm, to support the claims made in the publication and to highlight its relevance in applications. For instance, Chapters 3 to 4 contain experiments that compare the performance of an algorithm to baseline methods from literature and that illustrate its advantages. As numerical experiments are an important part of research on tensor decompositions, the reproducibility of these experiments is equally important.

Methods reproducibility, which will be referred to as reproducibility for brevity, is obtaining the same results as published in a paper by using the exact same

computational procedures on the same data [48]. In the context of numerical experiments, this simply means that the code and data used by the original authors of a paper should be available in order to be reproducible. The importance of reproducibility has come to light in different fields, such as psychology and biology, in which there were concerns that many published results are not reproducible [49]–[52]. In some cases, this has even been referred to as a "reproducibility crisis". On the other hand, it has also been argued that the term crisis is unjustified as it does not affect most of the literature [53].

While the presence of a reproducibility crisis is up for debate [54], [55], the resulting emphasis on reproducibility in research is beneficial regardless. It leads to higher quality research since the claims made in a publication can be verified more easily. Additionally, the availability of the code accompanying a publication enhances knowledge transfer, as it allows anyone to get a very detailed look at the algorithms and experiments. Besides psychology and biology, reproducibility has also received attention in fields closer to the topic of this thesis, namely in machine learning [56] and more specifically in tensor decompositions [57]. Because of its importance and the advantages it offers, reproducibility is a major objective of this thesis.

1.5 Research objectives and chapter overview

The first two main objectives lie at the intersection of tensor decompositions, incomplete and implicit data and algebraic algorithms. Namely, specific cases of incomplete and implicit tensors that enable the algebraic computation of tensor decompositions are considered. This way, the useful advantages of algebraic tensor decomposition algorithms for complete and explicitly given tensors are extended to the tensor completion and CS setting. The algebraic algorithms in this work are useful on their own and also as an initialization strategy for optimization algorithms. Given the importance of reproducibility in research, the third main objective in this thesis is to ensure that the reported results obtained with tensor decomposition algorithms are reproducible. To this end, guidelines for reproducible experiment design have been employed and a framework for sharing the relevant code and data has been developed.

1.5.1 Objective 1: An algebraic algorithm for computing the MLSVD of a fiber-wise observed tensor

In the general case of an incomplete tensor, the available entries are spread throughout the tensor. In [58], a specific pattern of the available entries is

considered, namely a tensor in which complete vectors or fibers along one mode are either entirely available or entirely missing. This fiber-wise observation pattern, as illustrated in Fig. 1.11, enables the algebraic computation of the CPD. The first objective of this thesis is to show that the same pattern[7] also enables the algebraic computation of an MLSVD and to develop the corresponding algorithm. These algorithms enable the algebraic completion of a fiber-wise observed tensor. Since matrix completion requires that the observed part of the data satisfies more restrictive constraints than a fiber-wise pattern, tensors have a key advantage over matrices for completion. The algorithm for algebraically computing an MLSVD of a fiber-wise observed tensor can be used for tensor completion and falls into the second main category in Section 1.2.1. This means that if this pattern is present, the algorithm enables an algebraic approach to tensor completion, whereas most algorithms in the literature are optimization-based [16], [22], [23], [25]. It can either be used by itself or as an initialization strategy for an optimization-based algorithm.

Additionally, this fiber-wise observation pattern arises in applications such as the traffic speed data example in Chapter 3. In general, applications in which obtaining samples along one mode is easier than along the others may lead to this pattern. For example, more samples along a temporal mode simply means collecting data for a longer time frame, whereas additional samples along another mode might mean that more sensors or experiments are required.

For completion algorithms of the first main category in Section 1.2.1, that is, those that minimize a convex relaxation of rank, there are theoretical guarantees regarding the recovery of the original low-rank matrix or tensor. Namely, under assumptions similar to those in CS exact recovery is possible with high probability if the number of available entries exceeds some bound [44], [59], [60]. Similarly, theoretical guarantees are available in the matrix case for the second main category of algorithms. In particular, any local minimum of the cost function is also a global minimum with high probability [61]. To the best of my knowledge, these guarantees do not extend to the tensor completion case. However, if no noise is present and certain uniqueness conditions are satisfied, then the algebraic algorithm exactly retrieves the original low multilinear rank tensor. The derivation of these uniqueness conditions is also part of the first objective.

[7] The conditions on the available fibers in a fiber-wise observed tensor under which a unique solution can be retrieved differs depending on whether a CPD or an MLSVD is computed.

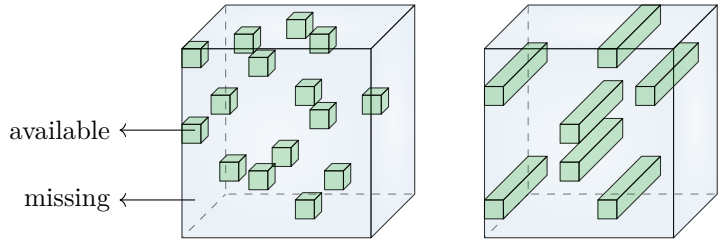


Figure 1.11: The first objective concerns a specific pattern of available tensor entries. Instead of the available entries being spread throughout the tensor (left), the tensor has been observed in vectors along the same mode (right). This observation pattern enables the algebraic computation of the MLSVD.

1.5.2 Objective 2: Computing tensor decompositions from a block row Kronecker-structured linear system.

Solving an underdetermined linear system of equations for a low-rank tensor is a problem that is very relevant in CS and scientific computing. In the standard CS setting, the coefficient matrix of the linear system is unstructured, as illustrated in Fig. 1.10. As mentioned before in Section 1.2.2, a separate (sparsifying) basis for each mode of the tensor might work well for higher-order data, which corresponds to representing the solution to the system as a Tucker decomposition. For the same reason, separate coefficient matrices can be used to compress each mode of the tensor individually. As illustrated in Fig. 1.12, this separate compression per mode corresponds to a linear system with a Kronecker-structured coefficient matrix. Coefficient matrices with this structure have already been used for CS [39], [41], [43], [62].

The second objective concerns a linear system with coefficient matrix that consists of multiple vertically stacked Kronecker-structured matrices, as illustrated in Fig. 1.13. This results in a block row Kronecker-structured (BRKS) linear system and each block row of this produces a compressed version of the tensor. This specific structure is considered because it allows one to solve the linear system in an algebraic manner for a tensor decomposition such as the MLSVD or the tensor train (TT). When computing an MLSVD, each compressed version of the tensor is used to compute one of its factor matrices. The algorithm for computing an MLSVD of a BRKS system can also be used as a precompression step before computing a CPD. This MLSVD precompression step is also common for computing the CPD of an explicitly given tensor [63] and it is extended to the implicit setting for BRKS linear systems. The second objective thus concerns algorithms for solving a BRKS system for an MLSVD,

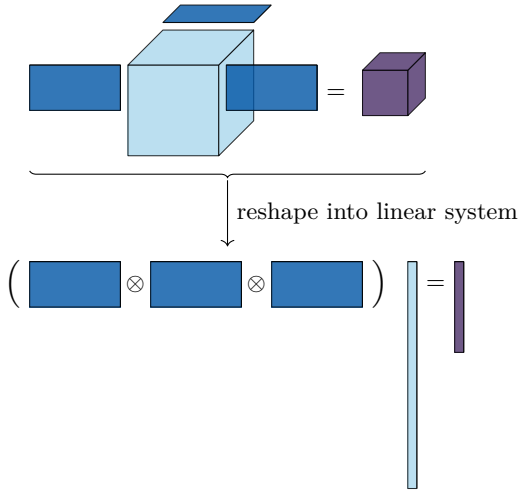


Figure 1.12: Compressing each mode of the tensor individually corresponds to a linear system with a Kronecker-structured coefficient matrix. Note that $\otimes$ denotes the Kronecker product.

CPD or TT and the corresponding conditions that guarantee that these tensor decompositions can be retrieved from the system.

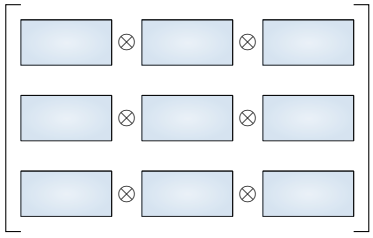


Figure 1.13: Vertically stacking multiple Kronecker-structured matrices yields a BRKS matrix. An underdetermined system with a BRKS coefficient matrix can be solved for a solution expressed as an MLSVD or TT in an algebraic manner.

1.5.3 Objective 3: A framework for reproducible numerical experiments with tensor decompositions

In publications on algorithms for computing tensor decompositions, the performance of the algorithms is evaluated and compared to other related

algorithms in numerical experiments. Given the importance of reproducibility and the advantages it offers, as discussed in Section 1.4, this objective concerns the reproducibility of tensor research. In particular, a software framework Tensorlab$^+$ is developed that ensures the reproducibility of such experiments and enables the dissemination of the corresponding code and data in a straightforward manner. The framework is not a software package, but instead, it is a repository containing experiment scripts and the code they depend on, organized per publication. This way, future work can continuously be added to the repository. The software framework is accompanied by coding guidelines and procedures for testing and reviewing that help to achieve reproducibility. The scope of this framework expands beyond the research in this thesis, as it also includes code belonging to other publications from our research group.

1.5.4 Chapter overview

The next chapter provides a basic overview of tensor decompositions. It also introduces the relevant tensor notation and definitions. Chapters 3 to 5 respectively correspond to research objectives 1 to 3. Each of these chapters is based on a paper that has been submitted to or published in a journal. As a result, this thesis is a collection of papers and Chapters 3 to 5 follow the general structure of a paper, which includes an introduction, literature overview, methods and numerical experiments.

Appendix A introduces a tensor-based approach to multivariate polynomial regression, which is relevant in machine learning and statistics. Since multivariate regression does not fit in with the three main objectives of this thesis, it is not a chapter. However, the tensor-based aspects of this appendix are very relevant to this thesis. Additionally, the approach can be used for tensor completion, as illustrated in Appendix A on a materials science dataset. The number of coefficients of a multivariate polynomial increases very rapidly as the degree of the polynomial increases, which can render the problem intractable. Since a multivariate polynomial can be represented by a tensor containing its coefficients, a very compact representation can be obtained under the assumption that this coefficient tensor is approximately low-rank. An efficient optimization-based algorithm and an algebraic algorithm are derived for computing the low-rank coefficients.

2 Tensor decompositions

Tensors are the generalization of vectors (first-order) and matrices (second-order) to higher orders and offer a natural way of representing higher-order data, while keeping its multidimensional character intact. Tensor decompositions provide tools for compressing or extracting information from tensors, as matrix decompositions do for matrices. However, tensor decompositions offer possibilities that go beyond their matrix counterparts, making them even more powerful. For example, mild uniqueness conditions enable the recovery of independent components from data, without requiring that these components satisfy strict constraints. Additionally, tensor decompositions can be used to break the curse of dimensionality and turn it into a blessing by compactly representing higher-order tensors. Because of the numerous advantages offered by tensor decompositions, they have been successfully applied in fields such as machine learning, (biomedical) signal processing, (hyperspectral) image and video processing, scientific computing, psychometrics and chemometrics [2]–[4], [47], [64]–[70].

In this chapter, the required tensor notation and definitions are introduced first. Next, the tensor decompositions that are relevant for the following chapters are described in detail.

2.1 Notation and definitions

This section concerns basic tensor notation that is used in the following chapters. Some chapters use additional notation that is specific to that chapter. This additional notation is defined in those chapters. For simplicity, all vectors, matrices and tensors are defined over the real field $\mathbb{R}$ in this chapter. For simplicity, the real field is used in all chapters except Chapter 3, in which the complex field $\mathbb{C}$ will be used.

Scalars, vectors, matrices and tensors are denoted by lower case (a), lower case boldface ($\mathbf{a}$), upper case boldface ($\mathbf{A}$) and upper case calligraphic letters ($\boldsymbol{\mathcal{A}}$), respectively. The nth element in a sequence of vectors, matrices or tensors is denoted by a superscript in parentheses $\cdot^{(n)}$. The order of a tensor is denoted

by N. An N-way tensor has N modes. The dimensions of a tensor describe its size along each mode and can be denoted by $I \times J \times K$ for a third-order tensor. In general, the dimensions of a tensor of order N can be denoted by $I_1 \times I_2 \times \cdots \times I_N$, which is more convenient for tensors with order greater than three. The conjugate, transpose, conjugate or Hermitian transpose, inverse, pseudoinverse, rank, column space or range, nullspace or kernel and Frobenius norm of a matrix $\mathbf{A}$ are denoted by $\mathbf{A}^*$, $\mathbf{A}^\mathrm{T}$, $\mathbf{A}^\mathrm{H}$, $\mathbf{A}^{-1}$, $\mathbf{A}^\dagger$, $\mathrm{r}(\mathbf{A})$, $\mathrm{range}(\mathbf{A})$, $\ker(\mathbf{A})$ and $||\mathbf{A}||_\mathrm{F}$ respectively. The identity matrix with dimensions $I \times I$ is denoted by $\mathbf{I}_I$. A vector of dimension I containing only ones is denoted by $\mathbf{1}_I$. A set is denoted by a lower case greek letter α. The cardinality of a set α and the complement of a set are respectively denoted by $\mathrm{card}(\alpha)$ and α^c.

2.1.1 Indexing, reshaping and permuting tensors

Indexing, i.e., selecting a part of a tensor, is denoted using indices as subscripts, for example, a_{ijk} is the entry at position (i, j, k) in the third-order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I \times J \times K}$. A colon as a subscript indicates that no selection is performed along the corresponding mode of the tensor. For example, $\mathbf{a}_{i:k} \in \mathbb{R}^J$ selects an entire vector that lies along the second mode of $\boldsymbol{\mathcal{A}}$. Figure 2.1 illustrates how selection is performed for some examples of indices. In general, a vector that lies along the nth mode of a tensor is referred to as a mode-n vector. A subtensor of order $N-1$ obtained by performing selection only along the nth mode of a tensor, such as $\mathbf{A}_{i::} \in \mathbb{R}^{J \times K}$, is a mode-$n$ slice. Selecting the jth column of a matrix $\mathbf{B}$ is denoted by $\mathbf{b}_{:j}$, but $\mathbf{b}_j$ can also be used as a shorthand.

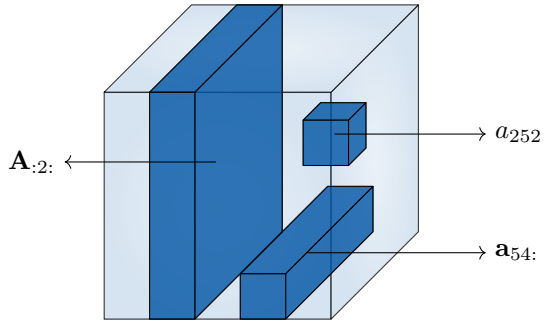


Figure 2.1: Selecting a part of a tensor is denoted with the indices as subscripts. A colon : as an index indicates that no selection is performed along the mode that corresponds to the position of the colon.

Combining and permuting the modes of a tensor is denoted by, for example, $\mathbf{A}_{[2;1,3]} \in \mathbb{R}^{J \times IK}$. The indices refer to the modes of the original tensor and the

";" separates the modes of the result. Within one mode of the result, the indices determine the order in which the entries are stacked in that mode, namely by iterating over the rightmost index first. For example, entry a_{ijk} of $\boldsymbol{\mathcal{A}}$ is located at position $(j, (i-1)I + k)$ in $\mathbf{A}_{[2;1,3]}$. The columns of the matrix $\mathbf{A}_{[2;1,3]}$ are the mode-2 vectors of $\boldsymbol{\mathcal{A}}$. In general, a matrix that has the mode-n vectors of a tensor as its columns is the mode-n unfolding of the tensor. In a mode-n unfolding, the modes other than n are ordered ascendingly in the rows of the unfolding: $\mathbf{A}_{[n;N,\ldots,n+1,n-1,\ldots,1]}$. A shorthand notation for this unfolding is $\mathbf{A}_{[n]}$. Stacking the entries of a tensor into a vector is denoted by $\operatorname{vec}(\boldsymbol{\mathcal{A}}) = \mathbf{a}_{[N,\ldots,2,1]}$.

2.1.2 Tensor and matrix products

The element-wise or Hadamard product is denoted by $*$. The Kronecker product $\otimes$ of matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ yields a matrix of dimensions $IK \times JL$ that equals:

$$\mathbf{A} \otimes \mathbf{B} \coloneqq \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \ldots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \ldots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \ldots & a_{IJ}\mathbf{B} \end{bmatrix}.$$

The Khatri–Rao product $\odot$ of matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$ is the column-wise Kronecker product of these matrices and yields a matrix of dimensions $IJ \times K$ that equals:

$$\mathbf{A} \odot \mathbf{B} \coloneqq \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \ldots & \mathbf{a}_K \otimes \mathbf{b}_K \end{bmatrix}.$$

The transposed Khatri–Rao product $\odot^{\mathrm{T}}$ of matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{I \times K}$ is the row-wise Kronecker product of these matrices:

$$\mathbf{A} \odot^{\mathrm{T}} \mathbf{B} = (\mathbf{A}^{\mathrm{T}} \odot \mathbf{B}^{\mathrm{T}})^{\mathrm{T}} \coloneqq \begin{bmatrix} \mathbf{a}_{1:} \otimes \mathbf{b}_{1:} \\ \mathbf{a}_{2:} \otimes \mathbf{b}_{2:} \\ \vdots \\ \mathbf{a}_{I:} \otimes \mathbf{b}_{I:} \end{bmatrix}.$$

A shorthand notation for a sequence of products is:

$$\bigotimes_{n=1}^{N} \mathbf{U}^{(n)} \coloneqq \mathbf{U}^{(1)} \otimes \cdots \otimes \mathbf{U}^{(N)}, \qquad \bigodot_{n=1}^{N} \mathbf{U}^{(n)} \coloneqq \mathbf{U}^{(1)} \odot \cdots \odot \mathbf{U}^{(N)}.$$

The outer product $\otimes$ of the vectors $\mathbf{a} \in \mathbb{R}^I$, $\mathbf{b} \in \mathbb{R}^J$ and $\mathbf{c} \in \mathbb{R}^K$ yields a tensor of dimensions $I \times J \times K$ and an entry of this tensor is equal to

$$(\mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c})_{ijk} \coloneqq a_i b_j c_k.$$

The mode-n product $\cdot_n$ of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ yields a tensor of dimensions $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$ and an entry of this tensor is equal to

$$(\boldsymbol{\mathcal{A}} \cdot_n \mathbf{U})_{i_1 i_2 \ldots j \ldots i_N} := \sum_{i_n=1}^{I_n} a_{i_1 i_2 \ldots i_n \ldots i_N} u_{j i_n}. \tag{2.1}$$

This means that $\mathbf{U}$ is multiplied with each mode-n vector of $\boldsymbol{\mathcal{A}}$. Let $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}} \cdot_n \mathbf{U}$, then $\mathbf{B}_{[n]} = \mathbf{U}\mathbf{A}_{[n]}$ holds.

2.2 Tensor decompositions

2.2.1 Canonical polyadic decomposition

A decomposition of a rank-R matrix $\mathbf{M}$ can be expressed as a matrix product

$$\mathbf{M} = \mathbf{A}\mathbf{B}^{\mathrm{T}} \quad \text{with} \quad \mathbf{A} \in \mathbb{R}^{I \times R}, \mathbf{B} \in \mathbb{R}^{J \times R}$$

or, equivalently, as a sum of rank-1 matrices, which are the outer product of two nonzero vectors

$$\mathbf{M} = \sum_{r=1}^{R} \mathbf{a}_r \mathbf{b}_r^{\mathrm{T}} = \sum_{r=1}^{R} \mathbf{a}_r \otimes \mathbf{b}_r.$$

A polyadic decomposition (PD) generalizes the latter representation of a matrix decomposition, since it expresses a tensor as a sum of rank-1 tensors, which is shown in Fig. 1.7. The generalization from a rank-1 matrix to a rank-1 tensor of order N is straightforward since the latter is the outer product of N nonzero vectors. A PD can thus be expressed as

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \mathbf{u}_r^{(1)} \otimes \mathbf{u}^{(2)} \otimes \cdots \otimes \mathbf{u}^{(N)} =: \left[\!\!\left[\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)} \right]\!\!\right], \tag{2.2}$$

where $\mathbf{U}^{(n)}$ for $n = 1, \ldots, N$ are the factor matrices of the PD. If the number of rank-1 components R in the sum is minimal, then R is the rank of the tensor and the polyadic decomposition is canonical (CPD).

The CPD is unique under mild conditions, whereas strict conditions are required for uniqueness in the matrix case. The latter is due to the fact that, given a matrix factorization $\mathbf{M} = \mathbf{A}\mathbf{B}^{\mathrm{T}}$, infinitely many alternative factorizations $\mathbf{M} = (\mathbf{A}\mathbf{W})\left(\mathbf{W}^{-1}\mathbf{B}^{\mathrm{T}}\right)$ for any nonsingular matrix $\mathbf{W} \in \mathbb{R}^{R \times R}$ are possible[1].

[1] Except in the $R = 1$ case, which is unique up to scaling and counterscaling of the vectors $\mathbf{a}$ and $\mathbf{b}$.

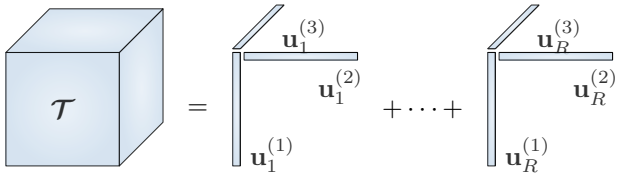


Figure 2.2: The CPD generalizes the sum of rank-1 terms representation of a matrix decomposition to tensors of higher order.

The SVD resolves this indeterminacy by factorizing $\mathbf{M}$ as $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{H}}$ and requiring that $\mathbf{U}$ and $\mathbf{V}$ contain orthonormal columns and $\mathbf{\Sigma}$ is a diagonal matrix with positive entries that are in monotonically non-increasing order. However, in the CPD the high degree of non-uniqueness of matrix decompositions, caused by the nonsingular matrix$\mathbf{W}$, is not present. Intuitively, this becomes clear by looking at a matrix unfolding of a tensor $\boldsymbol{\mathcal{T}}$ that admits a rank-R CPD:

$$\mathbf{T}_{[1]} = \mathbf{U}^{(1)} \left(\mathbf{U}^{(3)} \odot \mathbf{U}^{(2)}\right)^{\mathrm{T}}.$$

In general, it is not possible to construct an alternative CPD of $\boldsymbol{\mathcal{T}}$

$$\left(\mathbf{U}^{(1)}\mathbf{W}\right) \left(\mathbf{W}^{-1} \left(\mathbf{U}^{(3)} \odot \mathbf{U}^{(2)}\right)^{\mathrm{T}}\right). \tag{2.3}$$

for any nonsingular matrix $\mathbf{W} \in \mathbb{R}^{R\times R}$, since $\mathbf{W}^{-1}\left(\mathbf{U}^{(3)} \odot \mathbf{U}^{(2)}\right)$ no longer has Khatri–Rao structure and therefore Eq. (2.3) is not (the mode-1 unfolding of) a CPD[2]. Two ambiguities in the uniqueness of the CPD remain, namely the permutation of the rank-1 tensors in the sum in Eq. (2.2) and the relative scaling and counterscaling between the different vectors in one rank-1 tensor. Therefore, a CPD is unique up to these trivial ambiguities.

Deterministic uniqueness conditions guarantee that one specific CPD is unique if its factor matrices satisfy a set of conditions. A generic variant of a uniqueness condition is more general, as it guarantees uniqueness with a probability equal to one under the assumption that the factor matrices are generic matrices [8]. A generic matrix means that its entries have been sampled from an absolutely continuous probability distribution. As a result of this assumption, generic uniqueness conditions only depend on the dimensions and rank of a tensor. A traditional uniqueness condition is Kruskal's condition [71], which was generalized to tensors of order N in [72]. Its generic counterpart is

$$\sum_{n=1}^{N} \min(I_n, R) \geq (N-1)R + N - 1, \tag{2.4}$$

[2]Note that this is only an intuitive explanation. Alternative factorizations may exist, but this possibility is ruled out when the CPD satisfies conditions that are relatively mild.

which shows that a CPD is indeed easily unique. There are many more uniqueness conditions for the CPD and some are even less restrictive than Eq. (2.4) [4], [6], [7].

Computing a CPD is a difficult problem: the best rank-R approximation problem is ill-posed [73] and the problem might be ill-conditioned [74], [75]. The ill-posedness stems from the fact that the set of tensors with rank $\leq R$ is not closed and therefore a best rank-R approximation might not exist. However, it has recently been shown that in a region around the true rank-R tensor, a best rank-R approximation is guaranteed to exist and the problem is well-posed [76]. This fact might explain why a low-rank approximation often works well in practice for real data that has some inherent low-dimensional structure. The most straightforward approach for the algebraic computation of a CPD reduces the problem to the generalized eigenvalue decomposition (GEVD) of two (linear combinations of) matrix slices of the tensor [77]. However, the condition number of the GEVD problem can be arbitrarily larger than the condition number of the original CPD problem. This problem can be mitigated using a more advanced GEVD-based algorithm that utilizes many different pairs of matrix slices [78]. Whereas the initial GEVD-based algorithms require that two factor matrices of a third-order CPD have full column rank, there are other algebraic algorithms that have less restrictive requirements [79]–[81].

Optimization-based CPD algorithms can be divided into two types. The first type are alternating least squares (ALS) algorithms that update the factor matrices in each iteration in an alternating fashion. The update is computed by solving a least squares problem for one factor matrix while considering the other factor matrices to be known. The second type of optimization-based algorithms differs from the first type, as all factor matrices are updated simultaneously in each iteration [82]–[85]. While ALS approaches are common due to their simplicity, the second type allows the use of optimization algorithms with better convergence properties, such as second-order methods, and offers improved performance in ill-conditioned cases [86].

2.2.2 Multilinear singular value decomposition

The MLSVD generalizes some core properties of the singular values and vectors of the SVD to higher order tensors [87]. It decomposes a tensor as

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{S}} \cdot_1 \mathbf{U}^{(1)} \cdot_2 \mathbf{U}^{(2)} \cdots \cdot_N \mathbf{U}^{(N)} =: \left[\!\!\left[\boldsymbol{\mathcal{S}}; \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)} \right]\!\!\right], \tag{2.5}$$

in which $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ is the "all-orthogonal" and "ordered" core tensor and the factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ contain the multilinear singular vectors,

which are mutually orthonormal. Each factor matrix serves as a basis for the vectors along the corresponding mode of the tensor and the core captures the interactions between the basis vectors. The column-wise orthonormality constraint on the factor matrices is the same as the constraint placed upon $\mathbf{U}$ and $\mathbf{V}$ in a matrix SVD $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{H}}$. "All-orthogonality" of the core tensor, which means that all the mode-n slices are mutually orthogonal for each mode n, is a higher-order extension of the diagonal constraint on $\mathbf{\Sigma}$. Namely, it is straightforward that both the rows and columns of a diagonal matrix are mutually orthogonal. A higher-order tensor has a set of multilinear singular values for each mode n, which are equal to the Frobenius norm of the mode-n slices. Just like the singular values in an SVD are monotonically non-increasing, the mode-n slices of the core tensor are ordered such that the same holds true for the multilinear singular values along each mode n. When omitting the constraints placed upon the core tensor and factor matrices, the decomposition in Eq. (2.5) is a Tucker decomposition [88].

The minimal tuple $(R_1, R_2, \ldots, R_N)$ is the multilinear rank of a tensor. Since the nth factor matrix is a basis for the mode-n vectors of $\boldsymbol{\mathcal{T}}$, the dimension of the subspace spanned by this basis must be equal to the rank of the mode-n unfolding: $R_n = \mathrm{r}\left(\mathbf{T}_{[n]}\right)$. In general, the multilinear ranks are equal to the dimensions of the tensor, i.e., $R_n = I_n$ for $n = 1, \ldots, N$. In the case that $R_n < I_n$ for $n = 1, \ldots, N$, the tensor has a low multilinear rank. In real data with inherent structure, often only a fraction of the multilinear singular values along each mode is significant. As a result, such data can be closely approximated by a low multilinear rank tensor, that is, a Tucker decomposition with $R_n < I_n$ for $n = 1, \ldots, N$. Such a low multilinear rank approximation (LMLRA) can be obtained by truncating the MLSVD, as illustrated in Fig. 2.3. However, this approximation is not necessarily optimal in the least squares sense, unlike a low-rank matrix decomposition obtained through truncation of the SVD. Although suboptimal, the square of the approximation error of an LMLRA obtained by truncating the MLSVD is upper bounded by the sum of the squared multilinear singular values that were omitted [87]. Therefore, the resulting LMLRA still serves as a good approximation of the original tensor.

The factor matrices $\mathbf{U}^{(n)}$ for $n = 1, \ldots, N$ of the MLSVD can be obtained by computing the SVD of the mode-n unfoldings of the tensor. The core then follows from the mode-n multiplication of the original tensor with the Hermitian transpose of the computed factor matrices for each mode n:

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{S}} \cdot_1 \mathbf{U}^{(1)^{\mathrm{H}}} \cdot_2 \mathbf{U}^{(2)^{\mathrm{H}}} \cdots \cdot_N \mathbf{U}^{(N)^{\mathrm{H}}}.$$

As these steps only consist of standard linear algebra operations, we say that the standard MLSVD algorithm is algebraic. A possible purpose of computing an MLSVD is to obtain an LMLRA by truncating this MLSVD. In this case, an

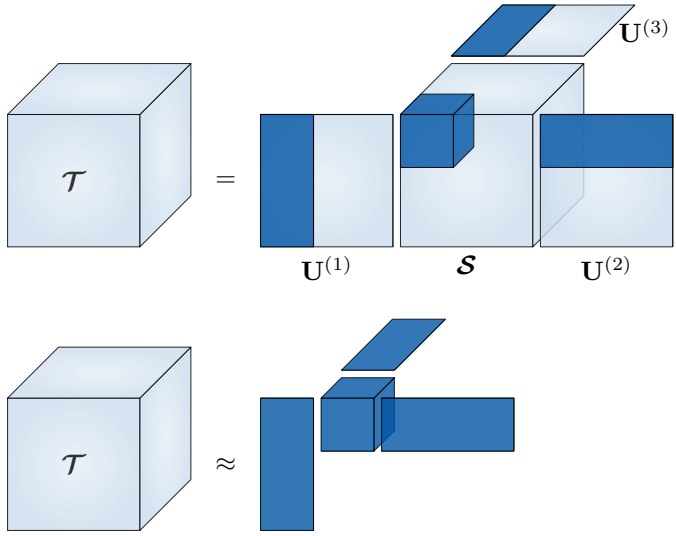


Figure 2.3: Top: The MLSVD generalizes some core concepts of the SVD to higher-order tensors. Bottom: A good, but not necessarily optimal, LMLRA can be obtained by truncating the MLSVD, that is, retaining only the dark blue parts of the factor matrices and core tensor.

LMLRA can be obtained at a lower computational cost compared to computing the MLSVD first and truncating it afterward. This cost reduction can be achieved by sequentially truncating each mode of the tensor through compression with the factor matrix for that mode immediately after it has been computed [89]. Another approach for speeding up the computation of an LMLRA through truncation is to compress the mode-n unfoldings through right multiplication with a random matrix before computing the SVD [90].

To further improve an LMLRA obtained through truncation of the MLSVD, various optimization-based algorithms can be employed. Like the CPD, an LMLRA can be computed using an ALS approach by separately updating the factor matrices during each iteration [91]. Given an LMLRA, an alternative factorization

$$\mathcal{T} = \hat{\mathcal{S}} \cdot_1 \left(\mathbf{U}^{(1)}\mathbf{W}^{(1)}\right) \cdot_2 \left(\mathbf{U}^{(2)}\mathbf{W}^{(2)}\right) \cdot_3 \left(\mathbf{U}^{(3)}\mathbf{W}^{(3)}\right)$$

$$\text{with } \hat{\mathcal{S}} = \mathcal{S} \cdot_1 \mathbf{W}^{(1)^{-1}} \cdot_2 \mathbf{W}^{(2)^{-1}} \cdot_3 \mathbf{W}^{(3)^{-1}}.$$

can be obtained by multiplying each factor matrix on the right with a nonsingular matrix and multiplying the vectors along each mode of the core with the

inverse of the corresponding nonsingular matrix. The MLSVD resolves these indeterminacies by imposing constraints on the factor matrices and core. During the computation of an LMLRA, these indeterminacies can be accomodated by optimizing on the manifold of tensors with fixed multilinear rank [92], [93].

2.2.3 Tensor train

The tensor train (TT) decomposition, which is also known as matrix product states, allows one to use the SVD along multiple modes, thus inheriting its numerical stability [94], [95]. The MLSVD also enjoys this numerical stability, but the core tensor has the same order as the original tensor, meaning that this decomposition is unable to break the curse of dimensionality. On the other hand, the TT is composed of a sequence of N tensors with order at most equal to three. Therefore the number of variables required to express a TT does not scale exponentially with N. The entry-wise definition of a TT is composed of a sequence of matrix products:

$$t_{i_1 i_2 \ldots i_N} = \mathbf{G}^{(1)}_{:i_1:} \mathbf{G}^{(2)}_{:i_2:} \cdots \mathbf{G}^{(N)}_{:i_N:}.$$

The matrices are the lateral or mode-2 matrix slices of the cores $\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ for $n = 1, \ldots, N$ of the TT. The values $R_0, R_1, \ldots, R_N$ are the TT-ranks of $\boldsymbol{\mathcal{T}}$. Since $R_0 = R_N = 1$, the first and last core are matrices, while the others are all third-order tensors, as illustrated in Fig. 2.4.

Two subsequent cores are joined together by a tensor contraction[3]

$$\left(\boldsymbol{\mathcal{G}}^{(n-1)} \cdot_{3,1} \boldsymbol{\mathcal{G}}^{(n)}\right)_{r_{n-2} i_{n-1} i_n r_n} = \sum_{r_{n-1}=1}^{R_{n-1}} g^{(n-1)}_{r_{n-2} i_{n-1} r_{n-1}} g^{(n)}_{r_{n-1} i_n r_n}.$$

Similar to the mode-n product in Eq. (2.1), a contraction sums products over a common index. The TT gets its name from the cores being interpreted as the carriages of a train that are connected by the common index in the contractions. A shorthand notation for the TT is:

$$\boldsymbol{\mathcal{X}} \coloneqq (\!| \boldsymbol{\mathcal{G}}^{(1)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)} |\!).$$

A TT can be computed using N SVDs, one for each core. The first core is obtained in the same manner as the first factor matrix of the MLSVD, namely as the left singular vectors of the mode-1 unfolding of the tensor. The

[3]The indices $3, 1$ underneath the symbol for contraction $\cdot$ indicate the modes of the cores that correspond to the common index.

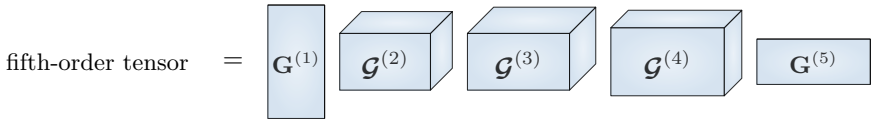


Figure 2.4: The TT consists of second- and third-order tensors, subsequently joined together by contractions over a common index, and therefore does not suffer from the curse of dimensionality.

algorithm then proceeds with the matrix that consists of the singular values and right singular vectors of the previous step. This matrix can be considered as a matricization of a compressed version of the original tensor. After a permutation of the modes in the matricization, the second core can be computed as its left singular vectors. The permutation and SVD steps are repeated until the last core is computed.

Tensor networks provide a general framework for decomposing a tensor into multiple tensors of lower order that are linked together using contractions. In fact, all of the tensor decompositions introduced in this section can be expressed as a tensor network. The framework can also represent more complicated decompositions, such as the hierarchical Tucker decomposition [96] or the tensor networks that are used for quantum systems [97].

3 Multilinear singular value decomposition based completion with fibers observed in a single mode

ABSTRACT

The multilinear singular value decomposition (MLSVD) extends the properties of the singular value decomposition to the tensor case and is a basic tool for working with tensors that are (approximately) of low multilinear rank. In the context of incomplete data, low multilinear rank tensor completion algorithms are usually optimization-based. In this chapter, we introduce an algorithm for computing the MLSVD of tensors with fibers that are observed along a single mode. This manner of observation allows one to formulate an algebraic algorithm, i.e., involving only standard linear algebra operations. The algorithm is guaranteed to exactly recover a low multilinear rank tensor in the noiseless case if some very reasonable conditions are satisfied. Part of this chapter consists of the derivation of algebraic algorithms for completing a matrix that is observed through a set of fully available submatrices. Experiments with synthetic data show that the algorithm comes close to algorithms from the literature in terms of accuracy while requiring far less computation time. As a possible use, we obtain a "completion" counterpart of initializing optimization-based low multilinear rank approximation algorithms with a truncated MLSVD. In a completion experiment with traffic speed data, our algorithm achieves a level of accuracy that is comparable to baseline algorithms.

This chapter is based on S. Hendrikx, M. Sørensen, and L. De Lathauwer, "Multilinear singular value decomposition based completion with fibers observed in a single mode", *Technical Report 23-142, ESAT-STADIUS, KU Leuven, Leuven, Belgium*, 2023, Submitted to SIAM Journal on Matrix Analysis and Applications.

3.1 Introduction

The multilinear singular value decomposition (MLSVD) is a fundamental tensor tool for uses such as higher-order principal component analysis, orthogonal compression or signal subspace estimation. The MLSVD has been used in fields such as machine learning, signal processing and scientific computing [3], [4], [65], [99]. It is also often used in a preprocessing step to speed up the computation of other tensor decompositions [100].

In tensor completion, the goal is to fill in the missing entries of an incomplete tensor, assuming that the tensor is (approximately) low rank. Datasets can be incomplete for various reasons. For example, a matrix containing movie ratings by different people is incomplete since one cannot have watched all movies [12]. Alternatively, a very large dataset might be considered to be incomplete because using all entries might be too computationally expensive [15]. In both cases, filling in the missing entries is very useful. In the former case, it allows one to make predictions and in the latter case it offers a compact model at a feasible cost.

In literature, there are two main approaches to tensor completion. The first one minimizes the rank of the completed tensor while keeping the error between the completed tensor and the incomplete data below some tolerance. Since minimizing the rank of a matrix or tensor is a difficult and non-convex problem, convex relaxations of the rank are used instead [22]–[24]. In both the matrix and tensor case, the complete dataset can be exactly recovered with high probability by minimizing a relaxed problem under some assumptions regarding incoherence [44], [59], [60]. The second approach works the opposite way, namely by minimizing the error between the incomplete data and a tensor (decomposition) with a fixed rank [13], [16], [25]. This approach also has theoretical guarantees in the matrix case, namely that any local minimum of the non-convex cost function is also a global minimum with high probability [61]. However, to the best of the authors' knowledge, there are no such theoretical guarantees for the second approach in the tensor case.

The algorithm introduced in this chapter uses the second tensor completion approach and approximates an incomplete tensor with a tensor of fixed multilinear rank. Algorithms of this type are usually optimization-based. Some of these algorithms use Riemannian optimization [25], [101]. Alternatively, there are algorithms that optimize over the variables of a Tucker decomposition, of which the MLSVD is a specific case with additional constraints [102], [103]. For a more general overview of tensor completion algorithms, we refer the reader to [21]. Properties such as existence, uniqueness and recoverability in the absence of noise are studied using algebraic geometry in [104].

An important aspect of optimization algorithms is the selection of the starting point or initialization. In the case of complete tensors, a starting point may be obtained in an algebraic manner, that is, using only standard linear algebra operations. For example, the canonical polyadic decomposition (CPD) can be computed using a generalized eigenvalue decomposition and the basic MLSVD algorithm consists of a few matrix singular value decompositions (SVDs). Such algebraic algorithms are not available for general incomplete tensors. A new point of view was developed in [58] for a specific type of incomplete tensors, in which the observed part of the tensor consists of entire vectors or fibers along a single mode (as opposed to consisting of entries that are spread randomly throughout the tensor). It turned out that in such cases, the CPD can again be computed in an algebraic manner. The low multilinear rank counterpart, i.e., an algebraic algorithm for computing the MLSVD of a fiber-wise observed tensor and the corresponding uniqueness conditions are introduced in this chapter. In the noiseless case, the algorithm exactly recovers the original low multilinear rank tensor if these conditions are satisfied.

To compute the factor matrices (except the factor matrix of the mode along which the fibers are observed), we derive algorithms for computing a column-wise orthonormal basis for the column space of an incomplete matrix. Additionally, we derive conditions under which this column space can be uniquely retrieved. The completion algorithms and corresponding uniqueness conditions introduced in this chapter are not specific for computing an MLSVD and can already be used for matrix completion as such.

The performance of our algorithm is evaluated in experiments with synthetic and real data. The experiments with synthetic data show that our algorithm achieves a level of accuracy that comes close to that achieved by optimization-based tensor completion algorithms from the literature and that it requires far less computation time. A solution obtained by our algorithm can be used as a good starting point for optimization-based algorithms, especially when the multilinear rank is overestimated. In real-life applications, a fiber-wise observation pattern can occur if obtaining entries along one mode is easier than along the others. For example, sampling along a temporal mode may be more practical than varying other problem parameters, such as concentrations or pH values. The traffic speed data [105] used in the final experiment is one example of real-life data exhibiting a fiber-wise observation pattern.

In the remainder of this section, the required tensor notation is introduced. Section 3.2 concerns the basics of the MLSVD and introduces the fiber-wise observation pattern. Next, uniqueness conditions for matrix completion are derived in Section 3.3. In Section 3.4, algebraic matrix completion algorithms are introduced. The algorithm for computing the overall MLSVD of a fiber-wise observed tensor and the corresponding uniqueness conditions are introduced in

Section 3.5. Last, the performance of the algorithm is evaluated in experiments with synthetic data and real-life traffic speed data.

3.1.1 Additional notation

The dimension of a subspace S is denoted by $\dim(S)$. The element-wise "logical or" operation of two $(0,1)$-binary matrices $\mathbf{A}$ and $\mathbf{B}$ is denoted by $\mathbf{A} \vee \mathbf{B}$, e.g., $\left[\begin{smallmatrix}1 & 0\\ 1 & 1\end{smallmatrix}\right] \vee \left[\begin{smallmatrix}1 & 0\\ 0 & 0\end{smallmatrix}\right] = \left[\begin{smallmatrix}1 & 0\\ 1 & 1\end{smallmatrix}\right]$. Let $\mathbf{e}_i^{(I)} \in \{0,1\}^I$ denote the I-dimensional unit vector with a one at position i and zeros elsewhere. The union of a sequence of sets is denoted by $\cup_{\ell \in \{1,\ldots,L\}} \alpha_\ell$. The matrix that selects the rows of a $(J \times K)$ matrix with indices in the set $\alpha_\ell = \{j_1, \ldots, j_{J_\ell}\} \subseteq \{1, \ldots, J\}$ through left multiplication is

$$\mathbf{S}_{\text{row}}^{(\ell)} \in \{0,1\}^{J_\ell \times J} = \begin{bmatrix} \mathbf{e}_{j_1}^{(J)} & \ldots & \mathbf{e}_{j_{J_\ell}}^{(J)} \end{bmatrix}^{\mathrm{T}}.$$

The matrix that selects the columns of a $(J \times K)$ matrix with indices in the set $\beta_\ell = \{k_1, \ldots, k_{K_\ell}\} \subseteq \{1, \ldots, K\}$ through right multiplication is

$$\mathbf{S}_{\text{col}}^{(\ell)} \in \{0,1\}^{K \times K_\ell} = \begin{bmatrix} \mathbf{e}_{k_1}^{(K)} & \ldots & \mathbf{e}_{k_{K_\ell}}^{(K)} \end{bmatrix}.$$

For example, selecting the submatrix of $\mathbf{A}$ at the intersection of the row indices $\alpha_1 = \{1,3\}$ and column indices $\beta_1 = \{2,4\}$ corresponds to

$$\mathbf{S}_{\text{row}}^{(1)} \mathbf{A} \mathbf{S}_{\text{col}}^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 10 & 12 \end{bmatrix}.$$

3.2 The multilinear singular value decomposition

The MLSVD is a special case of the Tucker decomposition, which factorizes a third-order[1] tensor as

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}} \cdot_1 \mathbf{A} \cdot_2 \mathbf{B} \cdot_3 \mathbf{C}, \tag{3.1}$$

with $\mathbf{A} \in \mathbb{C}^{I \times R_1}$, $\mathbf{B} \in \mathbb{C}^{J \times R_2}$, $\mathbf{C} \in \mathbb{C}^{K \times R_3}$, $\boldsymbol{\mathcal{G}} \in \mathbb{C}^{R_1 \times R_2 \times R_3}$. The minimal tuple (R_1, R_2, R_3) is the multilinear rank of $\boldsymbol{\mathcal{X}}$. The additional constraints that the MLSVD imposes on top of the Tucker decomposition are that the factor matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are column-wise orthonormal, like $\mathbf{U}$ and $\mathbf{V}$ in the matrix

[1] A third-order tensor is considered in this chapter to simplify the notation of the equations. The algorithm is generalized to tensors of order N in Section 3.5.1.

SVD $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{H}}$, and that the core tensor $\mathcal{G}$ is "all-orthogonal" and "ordered". The mode-n slices of an "all-orthogonal" tensor are mutually orthogonal for $n = 1, \ldots, N$. The "all-orthogonality" of the core can be seen as a generalization of $\mathbf{\Sigma}$ in the matrix SVD being diagonal, since both the rows and columns of a diagonal matrix are mutually orthogonal. Along each mode n of the core, the mode-n slices are ordered such that their Frobenius norms, which correspond to the multilinear singular values, decrease monotonically. Therefore, a low multilinear rank approximation of $\mathcal{X}$ can be obtained by truncating the core. In real-life applications, the multilinear singular values of the data tensor often exhibit a decaying trend, meaning that the tensor can be well approximated by a low multilinear rank model. Unlike in the matrix case, this approximation is not necessarily optimal in the least squares sense. However, the approximation error is bounded by the multilinear singular values that were omitted by truncating and such a low multilinear rank approximation might be sufficient in practice.

The columns of the factor matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ form bases for the subspaces spanned by the mode-1, 2 and 3 vectors of $\mathcal{X}$ respectively. A straightforward way to compute these factor matrices then is to obtain them from the dominant left singular vectors of their corresponding matrix unfolding. This is the standard MLSVD algorithm and is described in Algorithm 3.1, in which $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}$ and $\mathbf{U}^{(3)}$ correspond to $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ respectively.

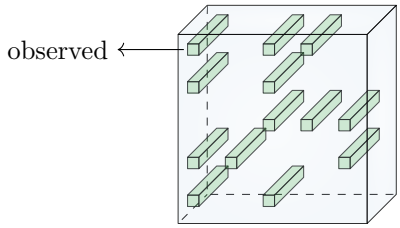
Algorithm 3.1: The basic MLSVD method.

Input: $\mathcal{X}$
Output: $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}, \mathcal{G}$
begin
  for $n = 1, 2, 3$ **do**
    Compute truncated SVD $\mathbf{X}_{[n;\mathcal{I}]} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{H}}$ with $\mathcal{I} = \{1, 2, 3\} \setminus \{n\}$
    Set $R_n \leftarrow \mathrm{r}\left(\mathbf{X}_{[n;\mathcal{I}]}\right)$, $\mathbf{U}^{(n)} \leftarrow \begin{bmatrix}\mathbf{u}_1 & \ldots & \mathbf{u}_{R_n}\end{bmatrix}$
  Compute core $\mathcal{G} = \mathcal{X} \cdot_1 \mathbf{U}^{(1)^{\mathrm{H}}} \cdot_2 \mathbf{U}^{(2)^{\mathrm{H}}} \cdot_3 \mathbf{U}^{(3)^{\mathrm{H}}}$

In this chapter, we develop an algorithm to compute the MLSVD of an incomplete tensor which consists of fibers observed along the last mode. The observed fibers are indicated by a binary mask $\mathbf{W} \in \{0, 1\}^{I \times J}$, meaning that $\mathbf{x}_{ij:}$ has been observed if $w_{ij} = 1$. For example, a mask and the corresponding fiber-wise observed tensor could look like in Fig. 3.1. Since complete mode-3 vectors are observed, computing the third factor matrix is as straightforward as in the complete tensor case. Namely, it can be obtained as the dominant left singular vectors of a compact version of the mode-3 unfolding of $\mathcal{X}$, namely $\mathbf{X}_{[3;2,1]}\mathbf{S}_{\mathrm{fib}}^{\mathrm{T}}$, in which all unobserved fibers have been dropped. Right multiplication with $\mathbf{S}_{\mathrm{fib}}^{\mathrm{T}}$ selects the columns with indices in the set $\{i \in \{1, \ldots, IJ\} | w_i = 1\}$, with

$\mathbf{w} = \mathrm{vec}\,(\mathbf{W})$.

Due to the pattern of observed fibers, this straightforward way of computing the factor matrices no longer works for modes 1 and 2. For example, the mode-2 unfolding of the tensor in Fig. 3.1 exhibits the pattern shown in Fig. 3.2. This pattern consists of submatrices of the unfolding that have been completely observed. In the following section, uniqueness conditions are derived that guarantee that the column space of an incomplete matrix can be computed and algebraic algorithms for this purpose are introduced in Section 3.4. These algorithms work with fully observed submatrices of the incomplete matrix and are used for the computation of the mode-1 and 2 factor matrices of the MLSVD of a fiber-wise observed tensor in Section 3.5.



$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 3.1: Example of a fiber-wise observed tensor and the corresponding mask.

3.3 Matrix completion: algebraic conditions for uniqueness

For a rank-R matrix $\mathbf{M} \in \mathbb{C}^{J\times F} = \mathbf{B}\mathbf{F}^{\mathrm{T}}$ with $\mathbf{B} \in \mathbb{C}^{J\times R}$ and $\mathbf{F} \in \mathbb{C}^{F\times R}$, we have that $(\mathbf{B}\mathbf{Z})\left(\mathbf{Z}^{-1}\mathbf{F}^{\mathrm{T}}\right)$ is an alternative factorization for any nonsingular matrix $\mathbf{Z} \in \mathbb{C}^{R\times R}$. Therefore, $\mathbf{B}$ and $\mathbf{F}$ can only be recovered up to consistent change-of-basis transformations. In this section, we will only be concerned with the unique retrieval of the subspace range($\mathbf{B}$). In the setting of matrix completion, in which the observed entries of $\mathbf{M}$ are indicated by the ones in $\mathbf{V} \in \{0,1\}^{J\times F}$, we will consider the subspace range($\mathbf{B}$) to be unique if every alternative factorization $\tilde{\mathbf{B}}\tilde{\mathbf{F}}^{\mathrm{T}}$, for which $\mathbf{V} * \mathbf{M} = \mathbf{V} * \left(\tilde{\mathbf{B}}\tilde{\mathbf{F}}^{\mathrm{T}}\right)$ holds, also satisfies the relation range($\mathbf{B}$) = range($\tilde{\mathbf{B}}$). The identifiability of the subspace range($\mathbf{B}$) is a necessary, but not sufficient, condition for the identifiability of the low-rank matrix completion of $\mathbf{M}$ [106]. The latter problem, which requires that the missing entries of $\mathbf{M}$ can be exactly recovered in the noiseless case,

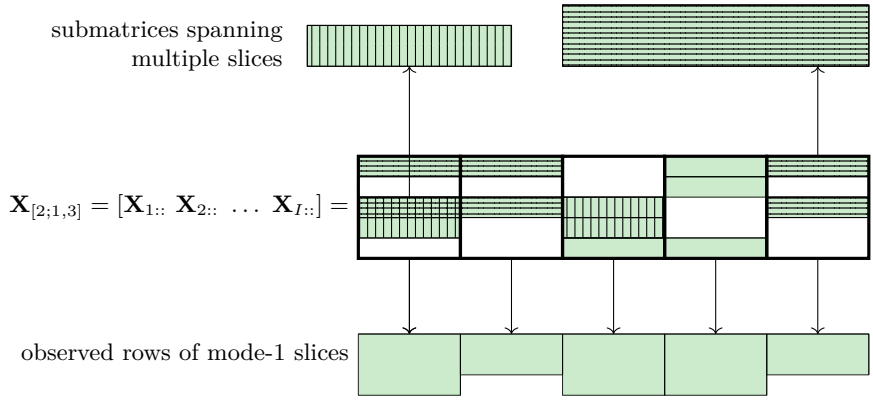


Figure 3.2: The mode-2 unfolding of the fiber-wise observed tensor in Fig. 3.1 consists of fully observed submatrices that are potentially overlapping. There are five fully observed submatrices that are formed by the observed rows of the mode-1 slices $\mathbf{X}_{i::}$ for $i = 1, \ldots, 5$. There are also two larger fully observed submatrices that span across multiple slices, namely the submatrices marked by horizontal and vertical lines.

is therefore more general than the former problem that is considered in this section.

For the completion of low-rank matrices, two types of identifiability conditions can be distinguished, namely statistical and deterministic conditions. Statistical conditions provide a bound on the number of randomly sampled observed entries of a low-rank matrix, such that it can be completed with a certain (high) probability [24]. On the other hand, deterministic conditions guarantee that a low-rank matrix can be completed for a specific mask, which is the pattern of observed entries. The conditions introduced in this chapter will be of the deterministic type. In literature, there are graph-based deterministic conditions that guarantee that the matrix can be completed if a bipartite graph, which has the mask as its biadjacency matrix, satisfies some constraints. In [107], the constraint that guarantees that a rank-R matrix can be completed is that this graph must be "R-closable". Similar graph-based matrix completion identifiability conditions can be found in [108], [109]. Furthermore, there are deterministic conditions that characterize which observation patterns are completable [110] or that follow from the connection between matrix completion and rigidity theory [111]. The former conditions are extended to low multilinear rank completion in [112].

Deterministic conditions in literature generally rely on assumptions on the

low-rank matrix $\mathbf{M}$, such as incoherence or the assumption that the matrices $\mathbf{B}$ and $\mathbf{F}$ are generic[2], so that identifiability only depends on the observation pattern. In this chapter, the identifiability conditions do not rely on such assumptions and therefore depend on both the observation pattern and the low-rank matrix $\mathbf{M}$ itself. The approach for checking whether our conditions are satisfied is also different from the literature, as it involves checking whether a certain contrived matrix has full column rank. We also provide a generic counterpart of a particular set of uniqueness conditions that is very easy to check given the mask, rank and matrix dimensions. Under the conditions presented in this section, factor matrices of the MLSVD of a fiber-wise observed tensor can be computed using the algorithms in Section 3.4. However, the conditions can also be used for matrix completion problems as such.

3.3.1 Problem definition

Consider the factorization of the incomplete version, in which missing entries have been replaced by zeros, of a low-rank matrix $\mathbf{M} \in \mathbb{C}^{J\times F}$:

$$\mathbf{V} * \mathbf{M} = \mathbf{V} *(\mathbf{B}\mathbf{F}^{\mathrm{T}}) \in \mathbb{C}^{J\times F}, \tag{3.2}$$

where $\mathbf{B} \in \mathbb{C}^{J\times R}$ and $\mathbf{F} \in \mathbb{C}^{F\times R}$ are rank-R matrices with $R \leq \min(J, F)$ and $\mathbf{V} \in \{0, 1\}^{J\times F}$ is a binary mask with property $v_{jk} = 1$ when entry m_{jk} is observed and $v_{jk} = 0$ otherwise. In this section, we think of $\mathbf{M}$ as a general incomplete matrix. It does not necessarily correspond to a matrix unfolding of a fiber-wise observed tensor. Whereas the mask $\mathbf{W}$ from the previous section indicates observed mode-3 fibers in a tensor $\boldsymbol{\mathcal{X}}$, the mask $\mathbf{V}$ indicates observed entries in a matrix $\mathbf{M}$. The mask $\mathbf{V}$ can always be expressed as

$$\mathbf{V} = \bigvee_{\ell=1}^{L} \mathbf{V}^{(\ell)} = \mathbf{V}^{(1)} \vee \cdots \vee \mathbf{V}^{(L)} \qquad \text{with } v_{ij}^{(\ell)} = \begin{cases} 1 & \text{if } i \in \alpha_\ell \text{ and } j \in \beta_\ell \\ 0 & \text{otherwise} \end{cases}, \tag{3.3}$$

with the sets $\alpha_\ell \subseteq \{1, \ldots, J\}$ (with $\operatorname{card}(\alpha_\ell) = J_\ell$) and $\beta_\ell \subseteq \{1, \ldots, F\}$ (with $\operatorname{card}(\beta_\ell) = F_\ell$) for $\ell = 1, \ldots, L$.

Due to the element-wise multiplication in Eq. (3.2), each $\mathbf{V}^{(\ell)}$ selects a submatrix of $\mathbf{M}$, namely the fully observed submatrix[3] at the intersection of the rows with index in α_ℓ and the columns with index in β_ℓ. For example, for $\alpha_1 = \{2, 3, 5\}$

[2]The entries of a generic matrix have been drawn from an absolutely continuous joint probability distribution.

[3]In case $\mathbf{M}$ is a matrix unfolding of a fiber-wise observed tensor, $\mathbf{M}$ contains fully observed submatrices that are similar to those in the example in Fig. 3.2.

and $\beta_1 = \{1, 3\}$ we have that

$$\mathbf{V}^{(1)} * \mathbf{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 8 & 0 & 0 \\ 11 & 0 & 13 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 21 & 0 & 23 & 0 & 0 \end{bmatrix},$$

for which the corresponding fully observed submatrix of $\mathbf{M}$ then is

$$\mathbf{M}_{\text{obs}}^{(1)} = \begin{bmatrix} 6 & 8 \\ 11 & 13 \\ 21 & 23 \end{bmatrix}.$$

These fully observed submatrices of $\mathbf{M}$ can be expressed as

$$\mathbf{M}_{\text{obs}}^{(\ell)} = \mathbf{S}_{\text{row}}^{(\ell)} \mathbf{M} \mathbf{S}_{\text{col}}^{(\ell)} = \mathbf{B}_{\text{obs}}^{(\ell)} {\mathbf{F}_{\text{obs}}^{(\ell)}}^{\mathrm{T}} \in \mathbb{C}^{J_\ell \times F_\ell} \quad \text{for } \ell = 1, \ldots, L, \tag{3.4}$$

where $\mathbf{B}_{\text{obs}}^{(\ell)} = \mathbf{S}_{\text{row}}^{(\ell)} \mathbf{B} \in \mathbb{C}^{J_\ell \times R}$ and $\mathbf{F}_{\text{obs}}^{(\ell)} = {\mathbf{S}_{\text{col}}^{(\ell)}}^{\mathrm{T}} \mathbf{F} \in \mathbb{C}^{F_\ell \times R}$.

The expression for the mask in Eq. (3.3) is not restrictive, as any $\mathbf{V}$ can be written in this form. However, at this point we make the working assumption that $\mathbf{B}_{\text{obs}}^{(\ell)}$ and $\mathbf{F}_{\text{obs}}^{(\ell)}$ have full column rank for $\ell = 1, \ldots, L$, i.e., each fully observed submatrix $\mathbf{M}_{\text{obs}}^{(\ell)}$ has rank R. Note that the working assumption implies that $\min(J_\ell, F_\ell) \geq R$, $1 \leq \ell \leq L$. In the specific case where $\mathbf{M}$ corresponds to a matrix unfolding of $\boldsymbol{\mathcal{X}}$, $\mathbf{V}$ is related to $\mathbf{W}$ and will be structured similarly to the example in Fig. 3.2. The mask $\mathbf{V}$ can also represent more general patterns that allow overlap between the columns of the fully observed submatrices.

3.3.2 Uniqueness conditions

We will now develop a condition that enables us to obtain the overall range($\mathbf{B}$) from the submatrix relations Eq. (3.4). Combination of the working assumption that the fully observed submatrices $\mathbf{M}_{\text{obs}}^{(\ell)}$ of $\mathbf{M}$ have rank R with Eq. (3.4) yields that range $\left(\mathbf{M}_{\text{obs}}^{(\ell)}\right) = \text{range}\left(\mathbf{B}_{\text{obs}}^{(\ell)}\right)$. As a result, the matrices $\mathbf{M}_{\text{obs}}^{(\ell)}$ and $\mathbf{B}_{\text{obs}}^{(\ell)}$ are equal up to transformation matrices $\boldsymbol{\Gamma}^{(\ell)}$:

$$\mathbf{M}_{\text{obs}}^{(\ell)} \boldsymbol{\Gamma}^{(\ell)} = \mathbf{S}_{\text{row}}^{(\ell)} \mathbf{B} = \mathbf{B}_{\text{obs}}^{(\ell)} \quad \text{for } \ell = 1, \ldots, L. \tag{3.5}$$

This is a set of linear systems with the a priori unknown matrices $\boldsymbol{\Gamma}^{(\ell)} \in \mathbb{C}^{F_\ell \times R}$ for $\ell = 1, \ldots, L$ and $\mathbf{B}$, all of which have full column rank. Note that a solution

to Eq. (3.5) can only be unique up to a change-of-basis transformational ambiguity. To be more specific, two solutions $[\mathbf{\Gamma}^{(1)^{\mathrm{T}}}, \ldots, \mathbf{\Gamma}^{(L)^{\mathrm{T}}}, \mathbf{B}^{\mathrm{T}}]^{\mathrm{T}}$ and $[\mathbf{\Delta}^{(1)^{\mathrm{T}}}, \ldots, \mathbf{\Delta}^{(L)^{\mathrm{T}}}, \widehat{\mathbf{B}}^{\mathrm{T}}]^{\mathrm{T}}$ to Eq. (3.5) are equivalent if there exists a nonsingular matrix $\mathbf{N} \in \mathbb{C}^{R \times R}$ such that

$$\begin{bmatrix} \mathbf{\Gamma}^{(1)^{\mathrm{T}}} & \ldots & \mathbf{\Gamma}^{(L)^{\mathrm{T}}} & \mathbf{B}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \mathbf{\Delta}^{(1)^{\mathrm{T}}} & \ldots & \mathbf{\Delta}^{(L)^{\mathrm{T}}} & \widehat{\mathbf{B}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \mathbf{N}.$$

The linear system in Eq. (3.5) can be written in matrix form as

$$\begin{bmatrix} \mathbf{M}_{\mathrm{obs}}^{(1)} & & & -\mathbf{S}_{\mathrm{row}}^{(1)} \\ & \ddots & & \vdots \\ & & \mathbf{M}_{\mathrm{obs}}^{(L)} & -\mathbf{S}_{\mathrm{row}}^{(L)} \end{bmatrix} \begin{bmatrix} \mathbf{\Gamma}^{(1)} \\ \vdots \\ \mathbf{\Gamma}^{(L)} \\ \mathbf{B} \end{bmatrix} = \mathbf{0}. \tag{3.6}$$

If the solution to this system of linear equations is unique up to the intrinsic change-of-basis transformation, then range($\mathbf{B}$) can be computed by solving it. However, more efficient algorithms can be developed by eliminating variables, which will be explained in Section 3.4.1 in the context of algebraic matrix completion algorithms.

A condition for the uniqueness of the solution of Eq. (3.6) is stated in Proposition 3.1 below, which makes use of the matrix

$$\mathbf{\Upsilon} := \begin{bmatrix} \mathbf{U}_{\mathrm{obs}}^{(1)} & & & -\mathbf{S}_{\mathrm{row}}^{(1)} \\ & \ddots & & \vdots \\ & & \mathbf{U}_{\mathrm{obs}}^{(L)} & -\mathbf{S}_{\mathrm{row}}^{(L)} \end{bmatrix},$$

where the columns of $\mathbf{U}_{\mathrm{obs}}^{(\ell)} \in \mathbb{C}^{J_\ell \times R}$ form a basis for range($\mathbf{M}_{\mathrm{obs}}^{(\ell)}$).

Proposition 3.1. *Consider the factorization of* $\mathbf{M}_{obs}^{(\ell)} \in \mathbb{C}^{J_\ell \times F_\ell}$ *for* $\ell = 1, \ldots, L$ *given by Eq.* (3.4)*. If*

$$\begin{cases} \mathbf{F}_{obs}^{(\ell)} \text{ has full column rank for } \ell = 1, \ldots, L, \\ \mathbf{B}_{obs}^{(\ell)} \text{ has full column rank for } \ell = 1, \ldots, L, \\ dim(ker(\mathbf{\Upsilon})) = R, \end{cases} \tag{3.7}$$

then range($\mathbf{B}$) *is unique and can be explicitly obtained from solving the set of linear equations Eq.* (3.5)*.*

Proof. Since $\mathbf{F}_{\mathrm{obs}}^{(\ell)}$ and $\mathbf{B}_{\mathrm{obs}}^{(\ell)}$ have full column rank, we can from Eq. (3.4) obtain a matrix $\mathbf{U}_{\mathrm{obs}}^{(\ell)}$ with property range($\mathbf{U}_{\mathrm{obs}}^{(\ell)}$) = range($\mathbf{M}_{\mathrm{obs}}^{(\ell)}$) = range($\mathbf{B}_{\mathrm{obs}}^{(\ell)}$).

This also means that there exists a nonsingular matrix $\mathbf{G}_{\text{obs}}^{(\ell)} \in \mathbb{C}^{R\times R}$ such that $\mathbf{B}_{\text{obs}}^{(\ell)} = \mathbf{U}_{\text{obs}}^{(\ell)}\mathbf{G}_{\text{obs}}^{(\ell)}$. Inserting the latter factorization into Eq. (3.4) yields $\mathbf{M}_{\text{obs}}^{(\ell)} = \mathbf{B}_{\text{obs}}^{(\ell)}\mathbf{F}_{\text{obs}}^{(\ell)T} = \mathbf{U}_{\text{obs}}^{(\ell)}\mathbf{G}_{\text{obs}}^{(\ell)}\mathbf{F}_{\text{obs}}^{(\ell)T}$. Inserting the latter factorization of $\mathbf{M}_{\text{obs}}^{(\ell)}$ into Eq. (3.5) in turn yields $\mathbf{M}_{\text{obs}}^{(\ell)}\mathbf{\Gamma}^{(\ell)} = \mathbf{U}_{\text{obs}}^{(\ell)}\mathbf{G}_{\text{obs}}^{(\ell)}\mathbf{F}_{\text{obs}}^{(\ell)T}\mathbf{\Gamma}^{(\ell)} = \mathbf{U}_{\text{obs}}^{(\ell)}\mathbf{V}_{\text{obs}}^{(\ell)} = \mathbf{B}_{\text{obs}}^{(\ell)}$, where $\mathbf{V}_{\text{obs}}^{(\ell)} := \mathbf{G}_{\text{obs}}^{(\ell)}\mathbf{F}_{\text{obs}}^{(\ell)T}\mathbf{\Gamma}^{(\ell)} \in \mathbb{C}^{R\times R}$ is nonsingular. From Eq. (3.6), we now obtain the relation $\mathbf{\Upsilon}[\mathbf{V}_{\text{obs}}^{(1)T}, \ldots, \mathbf{V}_{\text{obs}}^{(L)T}, \mathbf{B}]^{\text{T}} = \mathbf{0}$. Since $\dim(\ker(\mathbf{\Upsilon})) = R$, we can conclude that range($\mathbf{B}$) is unique and can be explicitly obtained from solving the set of linear equations Eq. (3.5). □

In Proposition 3.1 we have assumed that $\dim(\ker(\mathbf{\Upsilon})) = R$. We will now develop a condition on the factor matrices $\mathbf{B}$ and $\mathbf{F}$ under which $\dim(\ker(\mathbf{\Upsilon})) = R$ is guaranteed, implying in particular the uniqueness of range($\mathbf{B}$). This condition will be obtained by first eliminating $\mathbf{B}$ from Eq. (3.6). To achieve this, a linear system similar to Eq. (3.6) is formulated which utilizes the overlap between the available rows of different submatrices $\mathbf{M}_{\text{obs}}^{(\ell)}$. The set of indices of the overlapping rows that are available in both $\mathbf{M}_{\text{obs}}^{(\ell_1)}$ and $\mathbf{M}_{\text{obs}}^{(\ell_2)}$ is $\alpha_{\ell_1,\ell_2} = \alpha_{\ell_1} \cap \alpha_{\ell_2}$ with $J_{\ell_1,\ell_2} = \text{card}\,(\alpha_{\ell_1,\ell_2})$. The row selection matrix that selects these overlapping rows from $\mathbf{M}$ can be constructed in two ways, namely

$$\mathbf{S}_{\text{row}}^{(\ell_1,\ell_2)} = \mathbf{S}_{\text{row}}^{(\ell_1|\ell_2)}\mathbf{S}_{\text{row}}^{(\ell_2)} \text{ or } \mathbf{S}_{\text{row}}^{(\ell_1,\ell_2)} = \mathbf{S}_{\text{row}}^{(\ell_2,\ell_1)} = \mathbf{S}_{\text{row}}^{(\ell_2|\ell_1)}\mathbf{S}_{\text{row}}^{(\ell_1)}. \tag{3.8}$$

The "conditional" row selection matrix $\mathbf{S}_{\text{row}}^{(\ell_2|\ell_1)}$, omits the rows of $\mathbf{M}_{\text{obs}}^{(\ell_1)}$ that are not available in $\mathbf{M}_{\text{obs}}^{(\ell_2)}$. Namely, it selects the rows with indices in the set

$$\alpha_{\ell_2|\ell_1} = \{i \in \{1, \ldots, \text{card}\,(\alpha_{\ell_1})\} | \alpha_{\ell_1}(i) \in \alpha_{\ell_2}\},$$

in which $\alpha_{\ell_1}(i)$ is the ith element listed for the set α_{ℓ_1}. For example, the sets $\alpha_1 = \{1, 3, 5\}$ and $\alpha_2 = \{3, 4, 5\}$ have the indices $\{3, 5\}$ in common. The conditional set $\alpha_{2|1} = \{2, 3\}$ contains the positions of these common indices in α_1.

Combination of the two expressions in Eq. (3.8) with Eq. (3.5) yields

$$\mathbf{S}_{\text{row}}^{(\ell_2|\ell_1)}\mathbf{M}_{\text{obs}}^{(\ell_1)}\mathbf{\Gamma}^{(\ell_1)} = \mathbf{S}_{\text{row}}^{(\ell_1|\ell_2)}\mathbf{M}_{\text{obs}}^{(\ell_2)}\mathbf{\Gamma}^{(\ell_2)} = \mathbf{S}_{\text{row}}^{(\ell_1,\ell_2)}\mathbf{B} \quad \text{for } 1 \le \ell_1 < \ell_2 \le L,$$

so that

$$\mathbf{S}_{\text{row}}^{(\ell_2|\ell_1)}\mathbf{M}_{\text{obs}}^{(\ell_1)}\mathbf{\Gamma}^{(\ell_1)} - \mathbf{S}_{\text{row}}^{(\ell_1|\ell_2)}\mathbf{M}_{\text{obs}}^{(\ell_2)}\mathbf{\Gamma}^{(\ell_2)} = \mathbf{0} \quad \text{for } 1 \le \ell_1 < \ell_2 \le L. \tag{3.9}$$

In words this means that the matrices, obtained by omitting the rows of $\mathbf{M}_{\text{obs}}^{(\ell_1)}$ that are not in $\mathbf{M}_{\text{obs}}^{(\ell_2)}$ and by omitting the rows of $\mathbf{M}_{\text{obs}}^{(\ell_2)}$ that are not in $\mathbf{M}_{\text{obs}}^{(\ell_1)}$

respectively, are equal up to a change-of-basis transformation. Equation (3.9) can be written in matrix form, similar to Eq. (3.6), as

$$\mathbf{\Xi}\,\mathbf{\Gamma} = \mathbf{0}, \tag{3.10}$$

with

$$\mathbf{\Xi} = \begin{bmatrix} \mathbf{S}_{\text{row}}^{(2|1)}\mathbf{M}_{\text{obs}}^{(1)} & -\mathbf{S}_{\text{row}}^{(1|2)}\mathbf{M}_{\text{obs}}^{(2)} & \mathbf{0} & & \\ \mathbf{S}_{\text{row}}^{(3|1)}\mathbf{M}_{\text{obs}}^{(1)} & \mathbf{0} & -\mathbf{S}_{\text{row}}^{(1|3)}\mathbf{M}_{\text{obs}}^{(3)} & & \\ \vdots & & & \ddots & \\ & & & \mathbf{S}_{\text{row}}^{(L|L-1)}\mathbf{M}_{\text{obs}}^{(L-1)} & -\mathbf{S}_{\text{row}}^{(L-1|L)}\mathbf{M}_{\text{obs}}^{(L)} \end{bmatrix}$$

and

$$\mathbf{\Gamma} = \begin{bmatrix} \mathbf{\Gamma}^{(1)} \\ \vdots \\ \mathbf{\Gamma}^{(L)} \end{bmatrix}.$$

From the homogeneous matrix equation Eq. (3.10) we obtain an alternative to Proposition 3.1. More precisely, by replacing $\mathbf{\Upsilon}$ in condition Eq. (3.7) with

$$\mathbf{\Xi}_{\mathbf{U}} = \begin{bmatrix} \mathbf{S}_{\text{row}}^{(2|1)}\mathbf{U}_{\text{obs}}^{(1)} & -\mathbf{S}_{\text{row}}^{(1|2)}\mathbf{U}_{\text{obs}}^{(2)} & \mathbf{0} & & \\ \mathbf{S}_{\text{row}}^{(3|1)}\mathbf{U}_{\text{obs}}^{(1)} & \mathbf{0} & -\mathbf{S}_{\text{row}}^{(1|3)}\mathbf{U}_{\text{obs}}^{(3)} & & \\ \vdots & & & \ddots & \\ & & & \mathbf{S}_{\text{row}}^{(L|L-1)}\mathbf{U}_{\text{obs}}^{(L-1)} & -\mathbf{S}_{\text{row}}^{(L-1|L)}\mathbf{U}_{\text{obs}}^{(L)} \end{bmatrix}$$

where the columns of $\mathbf{U}_{\text{obs}}^{(\ell)} \in \mathbb{C}^{J_\ell \times R}$ form a basis for range$(\mathbf{M}_{\text{obs}}^{(\ell)})$, we obtain the result stated in Proposition 3.2.

Proposition 3.2. *Consider the factorization of* $\mathbf{M}_{obs}^{(\ell)} \in \mathbb{C}^{J_\ell \times F_\ell}$ *for* $\ell = 1, \ldots, L$ *given by Eq.* (3.4)*. If*

$$\begin{cases} \mathbf{F}_{obs}^{(\ell)} \text{ has full column rank for } \ell = 1, \ldots, L, \\ \mathbf{B}_{obs}^{(\ell)} \text{ has full column rank for } \ell = 1, \ldots, L, \\ dim(ker(\ \mathbf{\Xi}_{\mathbf{U}})) = R, \end{cases}$$

then range$(\mathbf{B})$ *is unique and can be explicitly obtained from solving the set of linear equations Eq.* (3.5)*.*

Proof. The proof is analogous to the proof of Proposition 3.1. □

If the uniqueness condition stated in Proposition 3.2 is satisfied, then this implies that the uniqueness condition stated in Proposition 3.1 must also be satisfied. This follows from the fact that the matrix equations Eq. (3.9) used in Proposition 3.2 are built from (a subset of) the matrix equations Eq. (3.5)

used in Proposition 3.1. That is, if the solution to the former set of equations is unique up to an intrinsic change-of-basis transformation ambiguity, then the same holds for the solution to the latter set of equations.

Based on Proposition 3.2 and relation Eq. (3.9) we will derive a third uniqueness condition that is equivalent to the condition stated in Proposition 3.2, but formulated only in terms of the factor matrices $\mathbf{B}$ and $\mathbf{F}$. In more detail, relation Eq. (3.9) can also be expressed as

$$\mathbf{S}_{\text{row}}^{(\ell_2|\ell_1)}\mathbf{S}_{\text{row}}^{(\ell_1)}\mathbf{B}\mathbf{F}_{\text{obs}}^{(\ell_1)^{\mathrm{T}}}\mathbf{\Gamma}^{(\ell_1)} - \mathbf{S}_{\text{row}}^{(\ell_1|\ell_2)}\mathbf{S}_{\text{row}}^{(\ell_2)}\mathbf{B}\mathbf{F}_{\text{obs}}^{(\ell_2)^{\mathrm{T}}}\mathbf{\Gamma}^{(\ell_2)}\mathbf{0},\ 1 \leq \ell_1 < \ell_2 \leq L,$$

Because of Eq. (3.8), this becomes

$$\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_2)}\mathbf{F}_{\text{obs}}^{(\ell_1)^{\mathrm{T}}}\mathbf{\Gamma}^{(\ell_1)} - \mathbf{B}_{\text{obs}}^{(\ell_1,\ell_2)}\mathbf{F}_{\text{obs}}^{(\ell_2)^{\mathrm{T}}}\mathbf{\Gamma}^{(\ell_2)} = \mathbf{0},\ 1 \leq \ell_1 < \ell_2 \leq L, \tag{3.11}$$

in which $\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_2)} = \mathbf{S}_{\text{row}}^{(\ell_1,\ell_2)}\mathbf{B} \in \mathbb{C}^{\text{card}\left(\alpha_{\ell_1,\ell_2}\right)\times R}$ collects the rows of $\mathbf{B}$ that are observed in both $\mathbf{M}_{\text{obs}}^{(\ell_1)}$ and $\mathbf{M}_{\text{obs}}^{(\ell_2)}$. Based on relations Eq. (3.10) and Eq. (3.11), Proposition 3.4 below presents a sufficient condition that ensures the identifiability of the subspace range($\mathbf{B}$). First, additional notation will be defined. Proposition 3.4 will make use of the matrix

$$\mathbf{\nabla} := \begin{bmatrix} \mathbf{\nabla}^{(1)} \\ \vdots \\ \mathbf{\nabla}^{(L-1)} \end{bmatrix} \in \mathbb{C}^{(\sum_{1\leq\ell_1<\ell_2\leq L} J_{\ell_1,\ell_2})\times(L-1)R} \tag{3.12}$$

with the submatrices $\mathbf{\nabla}^{(\ell_1)} \in \mathbb{C}^{(\sum_{\ell_2=\ell_1+1}^{L} J_{\ell_1,\ell_2})\times(L-1)R}$

$$\mathbf{\nabla}^{(\ell_1)} = \begin{bmatrix} \mathbf{0}_{J_{\ell_1,\ell_1+1}\times(\ell_1-1)R} & \mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+1)} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0}_{J_{\ell_1,\ell_1+2}\times(\ell_1-1)R} & \mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+2)} & \mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+2)} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{J_{\ell_1,L}\times(\ell_1-1)R} & \mathbf{B}_{\text{obs}}^{(\ell_1,L)} & \mathbf{B}_{\text{obs}}^{(\ell_1,L)} & \cdots & \cdots & \mathbf{B}_{\text{obs}}^{(\ell_1,L)} \end{bmatrix} \tag{3.13}$$

for $1 \leq \ell_1 < L-1$. The matrix $\mathbf{\nabla}$ is block upper triangular. To make this more clear, let us consider an example with $L = 4$:

$$\mathbf{\nabla} = \begin{bmatrix} \mathbf{\nabla}^{(1)} \\ \mathbf{\nabla}^{(2)} \\ \mathbf{\nabla}^{(3)} \end{bmatrix} = \left[\begin{array}{c|c|c} \mathbf{B}_{\text{obs}}^{(1,2)} & \mathbf{0} & \mathbf{0} \\ \mathbf{B}_{\text{obs}}^{(1,3)} & \mathbf{B}_{\text{obs}}^{(1,3)} & \mathbf{0} \\ \mathbf{B}_{\text{obs}}^{(1,4)} & \mathbf{B}_{\text{obs}}^{(1,4)} & \mathbf{B}_{\text{obs}}^{(1,4)} \\ \hline \mathbf{0} & \mathbf{B}_{\text{obs}}^{(2,3)} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{\text{obs}}^{(2,4)} & \mathbf{B}_{\text{obs}}^{(2,4)} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{B}_{\text{obs}}^{(3,4)} \end{array}\right]. \tag{3.14}$$

Additionally, Proposition 3.4 makes use of Lemma 3.3 below, which in turn makes use of the matrices $\boldsymbol{\Theta}^{(\ell_1)} \in \mathbb{C}^{(\sum_{\ell_2=\ell_1+1}^{L} J_{\ell_1,\ell_2})\times LR} =$

$$\begin{bmatrix} \mathbf{0}_{J_{\ell_1,\ell_1+1}\times(\ell_1-1)R} & \mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+1)} & -\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+1)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0}_{J_{\ell_1,\ell_1+2}\times(\ell_1-1)R} & \mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+2)} & \mathbf{0} & -\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+2)} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{J_{\ell_1,L}\times(\ell_1-1)R} & \mathbf{B}_{\text{obs}}^{(\ell_1,L)} & \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{B}_{\text{obs}}^{(\ell_1,L)} \end{bmatrix}$$

for $1 \leq \ell_1 < L-1$,

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\Theta}^{(1)} \\ \vdots \\ \boldsymbol{\Theta}^{(L-1)} \end{bmatrix} \in \mathbb{C}^{(\sum_{1\leq\ell_1<\ell_2\leq L} J_{\ell_1,\ell_2})\times LR} \tag{3.15}$$

and

$$\mathbf{L} = \begin{bmatrix} \mathbf{I}_R & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{I}_R & \mathbf{I}_R & \cdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{I}_R & \mathbf{I}_R & \cdots & \mathbf{I}_R \end{bmatrix} \in \mathbb{C}^{LR\times LR}. \tag{3.16}$$

To illustrate the structure of $\boldsymbol{\Theta}$, let us again consider an example with $L = 4$:

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\Theta}^{(1)} \\ \boldsymbol{\Theta}^{(2)} \\ \boldsymbol{\Theta}^{(3)} \end{bmatrix} = \left[\begin{array}{cccc} \mathbf{B}_{\text{obs}}^{(1,2)} & -\mathbf{B}_{\text{obs}}^{(1,2)} & \mathbf{0} & \mathbf{0} \\ \mathbf{B}_{\text{obs}}^{(1,3)} & \mathbf{0} & -\mathbf{B}_{\text{obs}}^{(1,3)} & \mathbf{0} \\ \mathbf{B}_{\text{obs}}^{(1,4)} & \mathbf{0} & \mathbf{0} & -\mathbf{B}_{\text{obs}}^{(1,4)} \\ \hline \mathbf{0} & \mathbf{B}_{\text{obs}}^{(2,3)} & -\mathbf{B}_{\text{obs}}^{(2,3)} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{\text{obs}}^{(2,4)} & \mathbf{0} & -\mathbf{B}_{\text{obs}}^{(2,4)} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{B}_{\text{obs}}^{(3,4)} & -\mathbf{B}_{\text{obs}}^{(3,4)} \end{array}\right].$$

The proof of Proposition 3.4 is constructive. It guarantees that $\boldsymbol{\Gamma}^{(1)}, \ldots, \boldsymbol{\Gamma}^{(L)}$ can be obtained via the system of linear of equations Eq. (3.9)/Eq. (3.10). To guarantee that not only $\boldsymbol{\Gamma}^{(1)}, \ldots, \boldsymbol{\Gamma}^{(L)}$ but also the full subspace range($\mathbf{B}$) can be obtained, each row of $\mathbf{M}$ must be observed in at least one of the submatrices $\mathbf{M}_{\text{obs}}^{(l)}$ for $\ell = 1, \ldots, L$[4]. Specifically, the necessary condition

$$\cup_{\ell=1}^{L} \alpha_\ell = \{1, \ldots, J\}$$

must hold. Finally, Proposition 3.4 makes use of the following lemma.

[4]This means that every row of $\mathbf{B}$ needs to appear in Eq. (3.5), because rows that do not appear cannot be retrieved. Note also that $\dim(\ker(\boldsymbol{\Upsilon})) = R$ implies that this basic condition is satisfied. Indeed, if $\cup_{\ell=1}^{L} \alpha_\ell \neq \{1, \ldots, J\}$, then $\boldsymbol{\Upsilon}$ will contain all-zero column vectors, implying that $\dim(\ker(\boldsymbol{\Upsilon})) > R$.

Lemma 3.3. *Consider the matrices* $\boldsymbol{\nabla}$, $\boldsymbol{\Theta}$ *and* $\mathbf{L}$ *given by Eqs.* (3.12), (3.15) *and* (3.16)*, respectively. Then* $\boldsymbol{\Theta}\mathbf{L} = [\mathbf{0}_{(\sum_{1\leq \ell_1<\ell_2\leq L} J_{\ell_1,\ell_2})\times R}, -\boldsymbol{\nabla}]$.

Proof. The lemma follows with a simple technical derivation from equations Eq. (3.12), Eq. (3.15) and Eq. (3.16). A formal proof based on induction is given in Section 3.8. □

Proposition 3.4. *Consider the factorization of* $\mathbf{M}_{obs}^{(\ell)} \in \mathbb{C}^{J_\ell \times F_\ell}$ *for* $\ell = 1, \ldots, L$ *given by Eq.* (3.4)*. If*

$$\begin{cases} \mathbf{F}_{obs}^{(\ell)} \text{ has full column rank for } \ell = 1, \ldots, L, \\ \mathbf{B}_{obs}^{(\ell)} \text{ has full column rank for } \ell = 1, \ldots, L, \\ \boldsymbol{\nabla} \text{ has full column rank}, \\ \cup_{\ell=1}^{L} \alpha_\ell = \{1, \ldots, J\}, \end{cases} \tag{3.17}$$

then range$(\mathbf{B})$ *is unique and can be explicitly obtained from solving the set of linear equations Eq.* (3.5)*.*

Proof. Given Eq. (3.8), we have that $\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_2)}{\mathbf{F}_{\text{obs}}^{(\ell_1)}}^{\mathrm{T}} = \mathbf{S}_{\text{row}}^{(\ell_2|\ell_1)}\mathbf{B}_{\text{obs}}^{(\ell_1)}{\mathbf{F}_{\text{obs}}^{(\ell_1)}}^{\mathrm{T}} = \mathbf{S}_{\text{row}}^{(\ell_2|\ell_1)}\mathbf{M}_{\text{obs}}^{(\ell_1)}$ and similarly that $\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_2)}{\mathbf{F}_{\text{obs}}^{(\ell_2)}}^{\mathrm{T}} = \mathbf{S}_{\text{row}}^{(\ell_1|\ell_2)}\mathbf{M}_{\text{obs}}^{(\ell_2)}$. As a result, Eq. (3.10) can be expressed as

$$\boldsymbol{\Xi}\boldsymbol{\Gamma} = \mathbf{0} \Leftrightarrow \boldsymbol{\Theta}\boldsymbol{\Lambda}\boldsymbol{\Gamma} = \mathbf{0}, \tag{3.18}$$

where $\boldsymbol{\Theta} \in \mathbb{C}^{(\sum_{1\leq \ell_1<\ell_2\leq L} J_{\ell_1,\ell_2})\times LR}$ is defined as indicated in Eq. (3.15) and

$$\boldsymbol{\Lambda} = \begin{bmatrix} {\mathbf{F}_{\text{obs}}^{(1)}}^{\mathrm{T}} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & {\mathbf{F}_{\text{obs}}^{(L)}}^{\mathrm{T}} \end{bmatrix}.$$

Next, we will argue that the conditions in Eq. (3.17) imply that $\boldsymbol{\Gamma}^{(1)}, \ldots, \boldsymbol{\Gamma}^{(L)}$ can be obtained via the system of linear of equations Eq. (3.9)/Eq. (3.10), which further implies that range$(\mathbf{B})$ can be uniquely obtained from Eq. (3.5)/Eq. (3.6) under the richness-of-sampling assumption $\cup_{\ell=1}^{L} \alpha_\ell = \{1, \ldots, J\}$. First, $\boldsymbol{\Gamma}^{(\ell)}$ has full column rank since Eq. (3.5) holds and $\mathbf{B}_{\text{obs}}^{(\ell)}$ and $\mathbf{F}_{\text{obs}}^{(\ell)}$ are assumed to have full column rank. Second, under the same assumption, range $\left({\mathbf{M}_{\text{obs}}^{(\ell)}}^{\mathrm{T}}\right) =$ range $\left(\mathbf{F}_{\text{obs}}^{(\ell)}\right)$ follows from Eq. (3.4), which implies that a basis for the latter can be obtained from the former. Let the columns of $\mathbf{V}^{(\ell)} \in \mathbb{C}^{F_\ell \times R}$ form a basis for range$(\mathbf{F}_{\text{obs}}^{(\ell)})$. Third, note that in Eq. (3.9) we can w.l.o.g. choose

$\mathbf{\Gamma}^{(\ell)}$ such that range($\mathbf{\Gamma}^{(\ell)}$) $\subseteq$ range($\mathbf{F}^{(\ell)}_{\text{obs}}$), as components of the columns of $\mathbf{\Gamma}^{(\ell)}$ that are orthogonal to range($\mathbf{F}^{(\ell)}_{\text{obs}}$) disappear after multiplication by $\mathbf{F}^{(\ell)\,\mathrm{T}}_{\text{obs}}$ in Eq. (3.9). Since range($\mathbf{F}^{(\ell)}_{\text{obs}}$) = range($\mathbf{V}^{(\ell)}$), there exists a nonsingular matrix $\mathbf{G}^{(\ell)} \in \mathbb{C}^{R\times R}$ such that $\mathbf{\Gamma}^{(\ell)} = \mathbf{V}^{(\ell)}\mathbf{G}^{(\ell)}$. This means that $\mathbf{\Gamma}$ in Eq. (3.18) admits the factorization

$$\mathbf{\Gamma} = \mathbf{\Upsilon}\mathbf{G}, \tag{3.19}$$

in which

$$\mathbf{\Upsilon} = \begin{bmatrix} \mathbf{V}^{(1)} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{V}^{(L)} \end{bmatrix} \in \mathbb{C}^{(\sum_{\ell=1}^{L} F_\ell)\times LR} \quad \text{and } \mathbf{G} = \begin{bmatrix} \mathbf{G}^{(1)} \\ \vdots \\ \mathbf{G}^{(L)} \end{bmatrix} \in \mathbb{C}^{LR\times R}.$$

Using factorization Eq. (3.19), relation Eq. (3.18) can be expressed as

$$\mathbf{\Theta\Lambda\Gamma} = \mathbf{\Theta\Lambda\Upsilon G} = \mathbf{0}.$$

Since $\mathbf{\Lambda\Upsilon} \in \mathbb{C}^{LR\times LR}$ is a nonsingular block-diagonal matrix with nonsingular matrices $\mathbf{F}^{(\ell)\,\mathrm{T}}_{\text{obs}}\mathbf{V}^{(\ell)} \in \mathbb{C}^{R\times R}$ for $\ell = 1,\ldots,L$ on its block-diagonal, we know that $\ker(\mathbf{\Theta}) = \ker(\mathbf{\Theta\Lambda\Upsilon})$. This means that if $\dim(\ker(\mathbf{\Theta})) = R$, then matrix $\mathbf{G}$, and in turn also $\mathbf{\Gamma}$, can be found from $\ker(\mathbf{\Theta})$ up to an irrelevant change-of-basis transformation. Due to relation Eq. (3.5)/Eq. (3.6) and the condition $\cup_{\ell=1}^{L}\alpha_\ell = \{1,\ldots,J\}$, this implies the uniqueness of range($\mathbf{B}$).

Finally, we will explain that condition Eq. (3.17) indeed implies that $\dim(\ker(\mathbf{\Theta})) = R$. From Lemma 3.3 we know that

$$\mathbf{\Theta L} = [\mathbf{0}_{(\sum_{\ell_2=\ell_1+1}^{L} J_{\ell_1,\ell_2})\times R}, -\mathbf{\nabla}],$$

where $\mathbf{L} \in \mathbb{C}^{LR\times LR}$ is a lower block-triangular matrix defined in accordance with Eq. (3.16). The full column rank condition on $\mathbf{\nabla}$ in Eq. (3.17) and the fact that $\mathbf{L}$ is nonsingular imply that $\dim(\ker(\mathbf{\Theta})) = R$. □

The number of rows of $\mathbf{\nabla}$, assuming that the number of overlapping rows between any combination of two fully observed submatrices $\mathbf{M}^{(\ell_1)}_{\text{obs}}$ and $\mathbf{M}^{(\ell_2)}_{\text{obs}}$ is cJ with $0 < c \leq 1$, is equal to $\frac{L(L-1)}{2}cJ$. Therefore, this matrix can become very large as L increases, making it computationally expensive to check whether it has full column rank and, in turn, making it impractical to verify the conditions in Proposition 3.4. The following corollary presents an easier way to guarantee that $\mathbf{\nabla}$ has full column rank, but the downside is that the resulting uniqueness conditions are more restrictive. This is achieved by checking whether the $L-1$ blocks on the block diagonal of $\mathbf{\nabla}$ have full column rank, which in turn guarantees that $\mathbf{\nabla}$ itself has full column rank. So checking the rank of a large

matrix has been replaced by checking the rank of multiple smaller matrices. These blocks on the block diagonal are denoted by

$$\boldsymbol{\nabla}^{(\ell_1 \to L)} = \begin{bmatrix} \mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+1)} \\ \mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+2)} \\ \vdots \\ \mathbf{B}_{\text{obs}}^{(\ell_1,L)} \end{bmatrix} \in \mathbb{C}^{(\sum_{\ell_2=\ell_1+1}^{L} J_{\ell_1,\ell_2}) \times R} \quad \text{for } \ell_1 = 1, \ldots, L-1.$$

In the example in Eq. (3.14), these blocks are

$$\begin{bmatrix} \mathbf{B}_{\text{obs}}^{(1,2)} \\ \mathbf{B}_{\text{obs}}^{(1,3)} \\ \mathbf{B}_{\text{obs}}^{(1,4)} \end{bmatrix}, \begin{bmatrix} \mathbf{B}_{\text{obs}}^{(2,3)} \\ \mathbf{B}_{\text{obs}}^{(2,4)} \end{bmatrix} \text{ and } \mathbf{B}_{\text{obs}}^{(3,4)}.$$

Corollary 3.5. *Consider the factorization of* $\mathbf{M}_{obs}^{(\ell)} \in \mathbb{C}^{J_\ell \times F_\ell}$ *for* $\ell = 1, \ldots, L$ *given by Eq.* (3.4)*. If*

$$\begin{cases} \mathbf{F}_{obs}^{(\ell)} \text{ has full column rank for } \ell = 1, \ldots, L, \\ \mathbf{B}_{obs}^{(\ell)} \text{ has full column rank for } \ell = 1, \ldots, L, \\ \boldsymbol{\nabla}^{(\ell_1 \to L)} \text{ has full column rank for } \ell_1 = 1, \ldots, L-1, \\ \cup_{\ell=1}^{L} \alpha_\ell = \{1, \ldots, J\}, \end{cases} \tag{3.20}$$

then range$(\mathbf{B})$ *is unique and can be explicitly obtained by solving the set of linear equations Eq.* (3.5)*.*

Proof. Observe that $\boldsymbol{\nabla}^{(\ell_1 \to L)}$ corresponds to the leftmost nonzero block column of $\boldsymbol{\nabla}^{(\ell_1)}$ given in Eq. (3.13). Since we assume that $\boldsymbol{\nabla}^{(\ell_1 \to L)}$ has full column rank for $\ell_1 = 1, \ldots, L-1$, the block-triangular structure of $\boldsymbol{\nabla}$ (see example Eq. (3.14)) implies that the overall matrix $\boldsymbol{\nabla}$ has full column rank. The result follows now directly from Proposition 3.4. □

The most restrictive part of the third set of conditions in Corollary 3.5 is that $\boldsymbol{\nabla}^{(L-1 \to L)} = \mathbf{B}_{\text{obs}}^{(L-1,L)}$ must have full column rank. As the order of the fully observed submatrices $\mathbf{M}_{\text{obs}}^{(\ell)}$ for $\ell = 1, \ldots, L$ does not matter, they can be reordered such that $\mathbf{B}_{\text{obs}}^{(L-1,L)}$ does have full column rank. The rank constraint on $\boldsymbol{\nabla}^{(\ell_1 \to L)}$ gradually relaxes as ℓ_1 decreases, because the number of vertically stacked matrices $\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_2)}$ for $\ell_2 = \ell_1 + 1, \ldots, L$ in $\boldsymbol{\nabla}^{(\ell_1 \to L)}$ increases. Therefore, none of the individual submatrices $\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_2)}$ for $\ell_2 = \ell_1 + 1, \ldots, L$ of $\boldsymbol{\nabla}^{(\ell_1 \to L)}$ need to have full column rank, only the overall matrix does.

Next, we present the generic variant of the uniqueness conditions in Corollary 3.5. The number of distinct rows in $\boldsymbol{\nabla}^{(\ell_1 \to L)}$, as there might be repeated rows, is denoted by $J_{\ell_1 \to L}$. The number of rows and columns in $\mathbf{M}_{\text{obs}}^{(\ell)} = \mathbf{B}_{\text{obs}}^{(\ell)} \mathbf{F}_{\text{obs}}^{(\ell)T}$ are respectively denoted by J_ℓ and F_ℓ. A generic matrix $\mathbf{A} \in \mathbb{C}^{I \times J}$ has full column rank if $I \geq J$. As a result, condition Eq. (3.20) is generically satisfied if

$$\begin{cases} J_\ell \geq R \text{ for } \ell = 1, \ldots, L, \\ F_\ell \geq R \text{ for } \ell = 1, \ldots, L, \\ J_{\ell_1 \to L} \geq R \text{ for } \ell_1 = 1, \ldots, L-1, \\ \cup_{\ell=1}^{L} \alpha_\ell = \{1, \ldots, J\}. \end{cases}$$

3.4 Matrix completion: algebraic algorithms

In this section, three algorithms are introduced for the computation of range $(\mathbf{B})$ from the fully observed submatrices $\mathbf{M}_{\text{obs}}^{(\ell)}$ for $\ell = 1, \ldots, L$ of the incomplete matrix $\mathbf{M}$. These algorithms can be used to compute a column-wise orthonormal basis for range $(\mathbf{B})$ using the set of equations Eq. (3.5) if condition Eq. (3.17) in Proposition 3.4 is satisfied.

3.4.1 An SVD-based kernel subspace method

If Eq. (3.5) is inexact, which is the case if noise is present, the linear systems can be solved in the least squares sense by minimizing the cost function

$$\min_{\boldsymbol{\Gamma}^{(1)}, \ldots, \boldsymbol{\Gamma}^{(L)}, \mathbf{B}} \sum_{\ell=1}^{L} \left|\left| \mathbf{M}_{\text{obs}}^{(\ell)} \boldsymbol{\Gamma}^{(\ell)} - \mathbf{B}_{\text{obs}}^{(\ell)} \right|\right|_{\text{F}}^2, \tag{3.21}$$

subject to $\mathbf{B}^{\text{H}}\mathbf{B} = \mathbf{I}_R$. Condition Eq. (3.17) states that $\mathbf{B}_{\text{obs}}^{(\ell)}$ and $\mathbf{F}_{\text{obs}}^{(\ell)}$ have full column rank, which in turn implies that $\mathbf{M}_{\text{obs}}^{(\ell)}$ has rank R for $\ell = 1, \ldots, L$. This allows the elimination of the variables $\boldsymbol{\Gamma}^{(\ell)}$ by substituting $\boldsymbol{\Gamma}^{(\ell)} = \mathbf{M}_{\text{obs}}^{(\ell)^\dagger} \mathbf{B}_{\text{obs}}^{(\ell)}$ into Eq. (3.21), which results in the optimization problem

$$\min_{\mathbf{B}^{\text{H}}\mathbf{B} = \mathbf{I}_R} \sum_{\ell=1}^{L} \left|\left| \left(\mathbf{M}_{\text{obs}}^{(\ell)} \mathbf{M}_{\text{obs}}^{(\ell)^\dagger} - \mathbf{I}_{J_\ell} \right) \mathbf{S}_{\text{row}}^{(\ell)} \mathbf{B} \right|\right|_{\text{F}}^2. \tag{3.22}$$

The matrix $\left(\mathbf{M}_{\text{obs}}^{(\ell)} \mathbf{M}_{\text{obs}}^{(\ell)^\dagger} - \mathbf{I}_{J_\ell} \right)$ is an orthogonal projector onto the orthogonal complement of range $\left(\mathbf{M}_{\text{obs}}^{(\ell)} \right)$. A column-wise orthonormal basis $\mathbf{Q}^{(\ell)}$ for this

complement can be obtained from the SVD $\mathbf{M}_{\text{obs}}^{(\ell)} = \mathbf{U}^{(\ell)}\mathbf{\Sigma}^{(\ell)}\mathbf{V}^{(\ell)^{\text{H}}}$, namely

$$\mathbf{Q}^{(\ell)} = \begin{bmatrix} \mathbf{u}_{R+1}^{(\ell)} & \dots & \mathbf{u}_{J_\ell}^{(\ell)} \end{bmatrix} \quad \text{for } \ell = 1, \dots, L.$$

The orthogonal projector in Eq. (3.22) can then be replaced by $\mathbf{Q}^{(\ell)}\mathbf{Q}^{(\ell)^{\text{H}}}$, which gives

$$\min_{\mathbf{B}^{\text{H}}\mathbf{B}=\mathbf{I}_R} \sum_{\ell=1}^{L} \left|\left| \mathbf{Q}^{(\ell)}\mathbf{Q}^{(\ell)^{\text{H}}}\mathbf{S}_{\text{row}}^{(\ell)}\mathbf{B} \right|\right|_{\text{F}}^2.$$

Here, the multiplication from the left by $\mathbf{Q}^{(\ell)}$ can be omitted because $\mathbf{Q}^{(\ell)}$ is column-wise orthonormal:

$$\min_{\mathbf{B}^{\text{H}}\mathbf{B}=\mathbf{I}_R} \sum_{\ell=1}^{L} \left|\left| \mathbf{Q}^{(\ell)^{\text{H}}}\mathbf{S}_{\text{row}}^{(\ell)}\mathbf{B} \right|\right|_{\text{F}}^2.$$

This minimization problem can further be rewritten as

$$\min_{\mathbf{B}^{\text{H}}\mathbf{B}=\mathbf{I}_R} ||\mathbf{B}^{\text{H}}\mathbf{Q}||_{\text{F}}^2 \quad \text{with } \mathbf{Q} = \begin{bmatrix} \mathbf{S}_{\text{row}}^{(1)^{\text{T}}}\mathbf{Q}^{(1)} & \dots & \mathbf{S}_{\text{row}}^{(L)^{\text{T}}}\mathbf{Q}^{(L)} \end{bmatrix}.$$

This shows that range $(\mathbf{B})$ is generated by the left singular vectors of $\mathbf{Q}$ that correspond to the R smallest singular values. The overall approach for the computation of a column-wise orthonormal basis for range $(\mathbf{B})$ is described in Algorithm 3.2.

Algorithm 3.2: SVD-based kernel subspace method for computing range$(\mathbf{B})$.

Input: $\mathbf{M}, \mathbf{V}$
Output: $\mathbf{B}$
begin
  for $\ell = 1, \dots, L$ **do**
    Compute SVD $\mathbf{M}_{\text{obs}}^{(\ell)} = \mathbf{U}^{(\ell)}\mathbf{\Sigma}^{(\ell)}\mathbf{V}^{(\ell)^{\text{H}}}$
    Set $\mathbf{Q}^{(\ell)} = \begin{bmatrix} \mathbf{u}_{R+1}^{(\ell)} & \dots & \mathbf{u}_{J_\ell}^{(\ell)} \end{bmatrix}$
  Compute SVD $\mathbf{Q} = \begin{bmatrix} \mathbf{S}_{\text{row}}^{(1)^{\text{T}}}\mathbf{Q}^{(1)} & \dots & \mathbf{S}_{\text{row}}^{(L)^{\text{T}}}\mathbf{Q}^{(L)} \end{bmatrix} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\text{H}}$
  Set $\mathbf{B} = \begin{bmatrix} \mathbf{u}_{J-R+1} & \dots & \mathbf{u}_J \end{bmatrix}$

Algorithm 3.2 is similar to Algorithm 1 in [113]. The latter algorithm was developed in the setting in which the observed entries can be adaptively selected. It selects entries such that for each column of the matrix, there is a fully observed submatrix with $R+1$ rows and at least R columns. As a result, the dimension of the left nullspace of each submatrix is equal to one, meaning that the equivalent

of the matrix $\mathbf{Q}^{(\ell)}$ in that algorithm only consists of one column. In contrast, in Algorithm 3.2 the dimension of the nullspace of the observed rows of the submatrix $\mathbf{M}_{\text{obs}}^{(\ell)}$, with J_ℓ observed rows, equals $J_\ell - R$ and $\mathbf{Q}^{(\ell)}$ has $J_\ell - R$ columns. This is an extension of the algorithm in [114] for the specific case $R = 3$, which was developed for the structure-from-motion problem in computer vision. In the next section, we introduce the principal subspace counterpart of this kernel subspace approach.

3.4.2 An SVD-based principal subspace method

In the previous section, a column-wise orthonormal basis for range $(\mathbf{B})$ was computed by minimizing the projection of $\mathbf{B}$ onto the orthogonal complement of range $\left(\mathbf{M}_{\text{obs}}^{(\ell)}\right)$ for $\ell = 1, \ldots, L$ simultaneously. In contrast, here we compute range $(\mathbf{B})$ by maximizing the projection onto range $\left(\mathbf{M}_{\text{obs}}^{(\ell)}\right)$ for $\ell = 1, \ldots, L$. The optimization problem in Eq. (3.22) can be rewritten as

$$\min_{\mathbf{B}^{\text{H}}\mathbf{B}=\mathbf{I}_R} \sum_{\ell=1}^{L} \left|\left| \left(\mathbf{M}_{\text{obs}}^{(\ell)} \mathbf{M}_{\text{obs}}^{(\ell)\dagger} \mathbf{S}_{\text{row}}^{(\ell)} - \mathbf{S}_{\text{row}}^{(\ell)} \right) \mathbf{B} \right|\right|_{\text{F}}^2 . \tag{3.23}$$

Each term in the sum can be multiplied from the left by $\mathbf{S}_{\text{row}}^{(\ell)^{\text{T}}}$ since this only adds zero rows and therefore does not change the Frobenius norm of that term. For example, $\mathbf{S}_{\text{row}}^{(\ell)^{\text{T}}} \mathbf{M}_{\text{obs}}^{(\ell)}$ yields a matrix with J rows, in which the rows with indices in α_ℓ are the rows of $\mathbf{M}_{\text{obs}}^{(\ell)}$ and the other rows are zero. After multiplication with $\mathbf{S}_{\text{row}}^{(\ell)^{\text{T}}}$, the optimization problem Eq. (3.23) becomes

$$\min_{\mathbf{B}^{\text{H}}\mathbf{B}=\mathbf{I}_R} \sum_{\ell=1}^{L} \left|\left| \left(\mathbf{S}_{\text{row}}^{(\ell)^{\text{T}}} \mathbf{M}_{\text{obs}}^{(\ell)} \mathbf{M}_{\text{obs}}^{(\ell)\dagger} \mathbf{S}_{\text{row}}^{(\ell)} - \mathbf{S}_{\text{row}}^{(\ell)^{\text{T}}} \mathbf{S}_{\text{row}}^{(\ell)} \right) \mathbf{B} \right|\right|_{\text{F}}^2 ,$$

or equivalently,

$$\min_{\mathbf{B}^{\text{H}}\mathbf{B}=\mathbf{I}_R} \sum_{\ell=1}^{L} \left|\left| \mathbf{Z}^{(\ell)} \mathbf{B} \right|\right|_{\text{F}}^2 , \tag{3.24}$$

with $\mathbf{Z}^{(\ell)} = \mathbf{S}_{\text{row}}^{(\ell)^{\text{T}}} \mathbf{M}_{\text{obs}}^{(\ell)} \mathbf{M}_{\text{obs}}^{(\ell)\dagger} \mathbf{S}_{\text{row}}^{(\ell)} - \mathbf{S}_{\text{row}}^{(\ell)^{\text{T}}} \mathbf{S}_{\text{row}}^{(\ell)}$. The matrix $\mathbf{Z}^{(\ell)}$ is an orthogonal projector, since the relations $\mathbf{Z}^{(\ell)^{\text{H}}} = \mathbf{Z}^{(\ell)}$ and $\mathbf{Z}^{(\ell)}\mathbf{Z}^{(\ell)} = \mathbf{Z}^{(\ell)}$ hold, onto the orthogonal complement of

$$\text{range}\left(\begin{bmatrix} \mathbf{S}_{\text{row}}^{(\ell)^{\text{T}}} \mathbf{M}_{\text{obs}}^{(\ell)} & \mathbf{S}_{\text{row}}^{(\ell^{\text{c}})^{\text{T}}} \end{bmatrix} \right). \tag{3.25}$$

The matrix $\mathbf{S}_{\text{row}}^{(\ell^{\text{c}})^{\text{T}}}$ contains unit columns with ones at the rows with indices that are in α_ℓ^{c}, that is, at the rows that have not been observed in $\mathbf{M}_{\text{obs}}^{(\ell)}$. A

matrix $\mathbf{P}^{(\ell)}$ with orthonormal columns that span the subspace in Eq. (3.25) can be computed from the SVD $\mathbf{M}_{\text{obs}}^{(\ell)} = \mathbf{U}^{(\ell)}\mathbf{\Sigma}^{(\ell)}\mathbf{V}^{(\ell)\text{H}}$, namely

$$\mathbf{P}^{(\ell)} = \begin{bmatrix} \mathbf{S}_{\text{row}}^{(\ell)\text{T}} \begin{bmatrix} \mathbf{u}_1^{(\ell)} \dots \mathbf{u}_R^{(\ell)} \end{bmatrix} & \mathbf{S}_{\text{row}}^{(\ell^{\text{c}})\text{T}} \end{bmatrix} \in \mathbb{C}^{J \times (R+J-J_\ell)}.$$

The leftmost part of $\mathbf{P}^{(\ell)}$, without $\mathbf{S}_{\text{row}}^{(\ell)\text{T}}$, spans a part of range $(\mathbf{B})$, namely range $\left(\begin{bmatrix} \mathbf{u}_1^{(\ell)} \dots \mathbf{u}_R^{(\ell)} \end{bmatrix}\right) = \text{range}\left(\mathbf{S}_{\text{row}}^{(\ell)}\mathbf{B}\right)$. The rightmost part $\mathbf{S}_{\text{row}}^{(\ell^{\text{c}})\text{T}}$ then adds unit columns for the rows that have not been observed in $\mathbf{M}_{\text{obs}}^{(\ell)}$, which ensures that range $(\mathbf{B}) \subseteq$ range $\left(\mathbf{P}^{(\ell)}\right)$. According to the orthogonal decomposition theorem, the minimization problem in Eq. (3.24) is equivalent to the following maximization problem:

$$\max_{\mathbf{B}^{\text{H}}\mathbf{B}=\mathbf{I}_R} \sum_{\ell=1}^{L} \left|\left|\mathbf{P}^{(\ell)}\mathbf{P}^{(\ell)\text{H}}\mathbf{B}\right|\right|_{\text{F}}^2.$$

Here, the multiplication from the left by $\mathbf{P}^{(\ell)}$ can be omitted because $\mathbf{P}^{(\ell)}$ is column-wise orthonormal:

$$\max_{\mathbf{B}^{\text{H}}\mathbf{B}=\mathbf{I}_R} \sum_{\ell=1}^{L} \left|\left|\mathbf{P}^{(\ell)\text{H}}\mathbf{B}\right|\right|_{\text{F}}^2.$$

The maximization problem can further be rewritten as

$$\max_{\mathbf{B}^{\text{H}}\mathbf{B}=\mathbf{I}_R} ||\mathbf{B}^{\text{H}}\mathbf{P}||_{\text{F}}^2 \quad \text{with } \mathbf{P} = \begin{bmatrix} \mathbf{P}^{(1)} & \dots & \mathbf{P}^{(L)} \end{bmatrix}.$$

This shows that range $(\mathbf{B})$ is generated by the left singular vectors of $\mathbf{P}$ that correspond to the R largest singular values. Given that range $(\mathbf{B})$ is a subspace of range $\left(\mathbf{P}^{(\ell)}\right)$ for $\ell = 1, \dots, L$, computing range $(\mathbf{B})$ as the dominant column space of $\mathbf{P}$ corresponds to finding the intersection of the subspaces range $\left(\mathbf{P}^{(\ell)}\right)$ for $\ell = 1, \dots, L$. The overall computation of a column-wise orthonormal basis for range $(\mathbf{B})$ is described in Algorithm 3.3. Matrix $\mathbf{P}$ is of size $J \times \left(L(R+J) - \sum_{\ell=1}^{L} J_\ell\right)$ and matrix $\mathbf{Q}$ is of size $J \times \left(\left(\sum_{\ell=1}^{L} J_\ell\right) - LR\right)$. This means that Algorithm 3.3 is less computationally expensive than Algorithm 3.2 if $\frac{1}{L}\sum_{\ell=1}^{L} J_\ell > \frac{J}{2} + R$ for $\ell = 1, \dots, L$, that is, if the number of observed rows in the fully observed submatrices is relatively large.

Algorithm 3.3: SVD-based principal subspace method for computing range($\mathbf{B}$).

Input: $\mathbf{M}, \mathbf{V}$
Output: $\mathbf{B}$
begin
  for $\ell = 1, \ldots, L$ **do**
    Compute SVD $\mathbf{M}_{\text{obs}}^{(\ell)} = \mathbf{U}^{(\ell)}\mathbf{\Sigma}^{(\ell)}\mathbf{V}^{(\ell)\text{H}}$
    Set $\mathbf{P}^{(\ell)} = \begin{bmatrix}\mathbf{S}_{\text{row}}^{(\ell)\text{T}} \begin{bmatrix}\mathbf{u}_1^{(\ell)} \ldots \mathbf{u}_R^{(\ell)}\end{bmatrix} & \mathbf{S}_{\text{row}}^{(\ell^c)\text{T}}\end{bmatrix}$
  Compute SVD $\mathbf{P} = \begin{bmatrix}\mathbf{P}^{(1)} & \ldots & \mathbf{P}^{(L)}\end{bmatrix} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\text{H}}$
  Set $\mathbf{B} = \begin{bmatrix}\mathbf{u}_1 & \ldots & \mathbf{u}_R\end{bmatrix}$

If $J_\ell \ll J$ for $\ell = 1, \ldots, L$, then $\mathbf{P}$ contains $\mathcal{O}(LJ)$ columns and the computational cost for finding the dominant column space of $\mathbf{P}$ increases. Since we have that range $(\mathbf{P})$ = range $(\mathbf{P}\mathbf{P}^{\text{T}})$, finding the dominant column space of range $(\mathbf{P}\mathbf{P}^{\text{T}})$ instead provides an alternative approach with a lower cost. This is because matrix $\mathbf{P}_{\text{tot}} = \mathbf{P}\mathbf{P}^{\text{T}}$ contains only J columns and can be computed without constructing $\mathbf{P}$ first. This alternative approach to computing a column-wise orthonormal basis for range $(\mathbf{B})$ is described in Algorithm 3.4. The tradeoff for the reduction in computational cost compared to Algorithm 3.3 is that the condition number of $\mathbf{P}_{\text{tot}}$ is the square of the condition number of $\mathbf{P}$.

Algorithm 3.4: SVD-based large-scale principal subspace method for computing range($\mathbf{B}$).

Input: $\mathbf{M}, \mathbf{V}$
Output: $\mathbf{B}$
begin
  $\mathbf{P}_{\text{tot}} = \mathbf{0}$
  for $\ell = 1, \ldots, L$ **do**
    Compute SVD $\mathbf{M}_{\text{obs}}^{(\ell)} = \mathbf{U}^{(\ell)}\mathbf{\Sigma}^{(\ell)}\mathbf{V}^{(\ell)\text{H}}$
    Keep the dominant left singular vectors $\mathbf{U}^{(\ell)} \leftarrow \begin{bmatrix}\mathbf{u}_1^{(\ell)} & \ldots & \mathbf{u}_R^{(\ell)}\end{bmatrix}$
    Update $\mathbf{P}_{\text{tot}} \leftarrow \mathbf{P}_{\text{tot}} + \mathbf{S}_{\text{row}}^{(\ell)\text{T}}\mathbf{U}^{(\ell)}\mathbf{U}^{(\ell)\text{H}}\mathbf{S}_{\text{row}}^{(\ell)} + \mathbf{S}_{\text{row}}^{(\ell^c)\text{T}}\mathbf{S}_{\text{row}}^{(\ell^c)}$
  Compute SVD $\mathbf{P}_{\text{tot}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\text{H}}$
  Set $\mathbf{B} = \begin{bmatrix}\mathbf{u}_1 & \ldots & \mathbf{u}_R\end{bmatrix}$

3.5 MLSVD of a fiber-wise observed tensor

In this section, we introduce the algorithm for computing an MLSVD from a fiber-wise observed tensor and the corresponding uniqueness conditions. The first and second factor matrix of the MLSVD are computed using the matrix completion algorithms from the previous section.

3.5.1 Algorithm

As mentioned in Section 3.2, $\mathbf{C}$ can be obtained from the dominant left singular vectors of $\mathbf{X}_{[3;2,1]}\mathbf{S}_{\text{fib}}^{\text{T}}$. To reduce the computational complexity of the next steps of the algorithm, the mode-3 fibers can be compressed using $\mathbf{C}$. In this way, the third dimension of the compressed tensor[5] $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \cdot_3 \mathbf{C}^{\text{H}}$ is reduced from K to R_3. Note that the product $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \cdot_3 \mathbf{C}^{\text{H}}$ does not need to be computed, since a matricized version of $\boldsymbol{\mathcal{Y}}$ can be obtained from the SVD of $\mathbf{X}_{[3;2,1]}\mathbf{S}_{\text{fib}}^{\text{T}}$, which has already been computed for obtaining $\mathbf{C}$.

The remaining factor matrices $\mathbf{A}$ and $\mathbf{B}$ can be computed by applying either Algorithm 3.2, Algorithm 3.3 or Algorithm 3.4 to the incomplete matrix unfoldings $\mathbf{Y}_{[1;2,3]}$ and $\mathbf{Y}_{[2;1,3]}$ respectively.

Last, the core can be computed from a linear system of equations. Given that the tensor $\boldsymbol{\mathcal{X}}$ admits the MLSVD Eq. (3.1), the observed part of its mode-3 unfolding is equal to

$$\mathbf{S}_{\text{fib}}\mathbf{X}_{[3;2,1]}^{\text{T}} = \mathbf{S}_{\text{fib}}(\mathbf{B} \otimes \mathbf{A})\mathbf{G}_{[3;2,1]}\mathbf{C}^{\text{T}},$$

with $\mathbf{S}_{\text{fib}}$ as defined in Section 3.2. After the initial compression of the mode-3 vectors with $\mathbf{C}$, this becomes

$$\mathbf{S}_{\text{fib}}\mathbf{Y}_{[3;2,1]}^{\text{T}} = \mathbf{S}_{\text{fib}}(\mathbf{B} \otimes \mathbf{A})\mathbf{G}_{[3;2,1]}. \tag{3.26}$$

This is a linear system that can be solved for $\mathbf{G}_{[3;2,1]}$. The overall algorithm for computing the MLSVD of $\boldsymbol{\mathcal{X}}$ is described in Algorithm 3.5. It can easily be generalized to a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{C}^{I_1 \times \cdots \times I_N}$ of order N that admits an MLSVD with factor matrices $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}$ and core $\boldsymbol{\mathcal{G}}$. The last factor matrix $\mathbf{U}^{(N)}$ corresponds to the dominant left singular vectors of $\mathbf{X}_{[N;N-1,\ldots,1]}\mathbf{S}_{\text{fib}}^{\text{T}}$. After compression of the mode-N vectors, we have that $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \cdot_N \mathbf{U}^{(N)^{\text{H}}}$. Factor matrix $\mathbf{U}^{(n)}$ for $n = 1, \ldots, N-1$ can be obtained by applying any algorithm

[5] Since $\boldsymbol{\mathcal{X}}$ is fiber-wise observed, the mode-3 product is only computed for the observed fibers of $\boldsymbol{\mathcal{X}}$.

from Section 3.4 to the matrix unfolding $\mathbf{Y}_{[1,\ldots,N-2;N-1;N]}$. The core can be found as the solution of the linear system

$$\mathbf{S}_{\text{fib}}\mathbf{Y}^{\text{T}}_{[N;N-1,\ldots,1]} = \mathbf{S}_{\text{fib}}(\mathbf{U}^{(N-1)} \otimes \cdots \otimes \mathbf{U}^{(1)})\mathbf{G}_{[N;N-1,\ldots,1]}$$

for $\mathbf{G}_{[N;N-1,\ldots,1]}$, which is the generalization of the system in Eq. (3.26) to order N.

Algorithm 3.5: (`mlsvd_fs`) Computation of an MLSVD of a fiber-wise observed tensor.

Input: $\boldsymbol{\mathcal{X}}, \mathbf{W}$
Output: $\mathbf{A}, \mathbf{B}, \mathbf{C}, \boldsymbol{\mathcal{G}}$
begin
- Compute SVD $\mathbf{X}_{[3;2,1]}\mathbf{S}^{\text{T}}_{\text{fib}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\text{H}}$
- Set $\mathbf{C} = \begin{bmatrix}\mathbf{u}_1 & \ldots & \mathbf{u}_{R_3}\end{bmatrix}$
- Compress mode-3 vectors $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \cdot_3 \mathbf{C}^{\text{H}}$
- Compute $\mathbf{A}$ from $\boldsymbol{\mathcal{Y}}$ using Algorithm 3.2, 3.3 or 3.4
- Compute $\mathbf{B}$ from $\boldsymbol{\mathcal{Y}}$ using Algorithm 3.2, 3.3 or 3.4
- Solve $\mathbf{S}_{\text{fib}}\mathbf{Y}^{\text{T}}_{[3;2,1]} = \mathbf{S}_{\text{fib}}(\mathbf{B} \otimes \mathbf{A})\mathbf{G}_{[3;2,1]}$ for $\mathbf{G}_{[3;2,1]}$

3.5.2 Uniqueness conditions

In order to be able to uniquely retrieve range $(\mathbf{C})$, matrix $\mathbf{X}_{[3;2,1]}\mathbf{S}^{\text{T}}_{\text{fib}}$ containing the observed mode-3 vectors must have rank R_3. The core tensor can be retrieved if the solution to the linear system in Eq. (3.26) is unique, which is the case if $\text{r}\left(\mathbf{S}_{\text{fib}}(\mathbf{B} \otimes \mathbf{A})\right) = R_1R_2$. The conditions in Proposition 3.4 can be used to guarantee the unique retrieval of range $(\mathbf{A})$ and range $(\mathbf{B})$ from the incomplete matrix unfoldings $\mathbf{X}_{[1;2,3]}$ and $\mathbf{X}_{[2;1,3]}$, respectively. The first two sets of conditions in Eq. (3.17) can be satisfied by only selecting fully observed submatrices of $\mathbf{X}_{[1;2,3]}$ ($\mathbf{X}_{[2;1,3]}$) that are of rank R_1 (R_2). In order to differentiate between modes 1 and 2, matrix $\boldsymbol{\nabla}^{(\mathbf{A})}$ denotes the equivalent of the matrix in Eq. (3.12) that is constructed for $\mathbf{A}$ and matrix $\boldsymbol{\nabla}^{(\mathbf{B})}$ denotes its counterpart for $\mathbf{B}$. For the same reason, the sets α_ℓ are replaced by $\alpha^{(1)}_{\ell_1}$ and $\alpha^{(2)}_{\ell_2}$, in which the superscript corresponds to the mode of the unfolding. The overall conditions that guarantee that the factor matrices of the MLSVD of a fiber-wise observed tensor can be uniquely retrieved, up to orthogonal transformation, are listed in Proposition 3.6.

Proposition 3.6. *Consider a fiber-wise observed tensor* $\boldsymbol{\mathcal{X}} \in \mathbb{C}^{I\times J\times K}$ *of multilinear rank* (R_1, R_2, R_3)*, in which the observed mode-*3 *vectors are indicated by the ones in* $\mathbf{W} \in \{0,1\}^{I\times J}$*. Given a set of fully observed submatrices of*

$\mathbf{X}_{[1;3,2]}$ of rank R_1 and a set of fully observed submatrices of $\mathbf{X}_{[2;3,1]}$ of rank R_2, the factor matrices and core of the MLSVD of $\boldsymbol{\mathcal{X}}$ can be retrieved if

$$\begin{cases} \mathrm{r}\left(\mathbf{X}_{[3;2,1]}\mathbf{S}_{\mathrm{fib}}^{\mathrm{T}}\right) = R_3 & [\text{factor } \mathbf{C}] \\ \boldsymbol{\nabla}^{(\mathbf{A})} \text{ has full column rank} & [\text{factor } \mathbf{A}] \\ \cup_{\ell_1 \in \{1,\ldots,L_1\}} \alpha_j^{(1)} = \{1,\ldots,J\} & [\text{factor } \mathbf{A}] \\ \boldsymbol{\nabla}^{(\mathbf{B})} \text{ has full column rank} & [\text{factor } \mathbf{B}] \\ \cup_{i \in \Omega^{(2)}} \alpha_i^{(2)} = \{1,\ldots,I\} & [\text{factor } \mathbf{B}] \\ \mathrm{r}\left(\mathbf{S}_{\mathrm{fib}}(\mathbf{B} \otimes \mathbf{A})\right) = R_1 R_2 & [\text{core } \boldsymbol{\mathcal{G}}]. \end{cases} \tag{3.27}$$

In the generic case, the first and last conditions of Proposition 3.6 are satisfied if $F \geq R_3$ and $F \geq R_1 R_2$ respectively, in which $F = \sum_{i,j} w_{ij}$ equals the total number of observed mode-3 vectors. Since R_3 is upper bounded by $R_1 R_2$, which is one of the properties of multilinear rank, $F \geq R_1 R_2$ implies $F \geq R_3$. A generic variant of Corollary 3.5 can be used to guarantee that $\boldsymbol{\nabla}^{(\mathbf{A})}$ and $\boldsymbol{\nabla}^{(\mathbf{B})}$ have full column rank except in a set of measure zero. The third and fifth conditions in Proposition 3.6 remain the same in the generic case.

3.6 Numerical experiments

In this section, we evaluate the performance of Algorithm 3.5 on synthetic and real datasets. Algorithm 3.4 is used for computing $\mathbf{A}$ and $\mathbf{B}$ since it is less computationally expensive than the other algorithms in Section 3.4.

3.6.1 Synthetic data experiments

A synthetic tensor with a given multilinear rank is generated by sampling the entries of the core tensor and factor matrices from the standard normal distribution. The level of additive Gaussian noise $\boldsymbol{\mathcal{N}}$ added to this tensor $\boldsymbol{\mathcal{X}}$ is quantified by the signal-to-noise ratio (SNR), which is defined as $20 \log_{10} \frac{||\boldsymbol{\mathcal{X}}||}{||\boldsymbol{\mathcal{N}}||}$. Since this chapter considers fiber-wise observed incomplete tensors, only a fraction of the mode-3 fibers of the tensor are available to the algorithm. The available mode-3 fibers are indicated by the mask $\mathbf{W}$. The sampling ratio is the ratio of the number of available entries in a tensor to the total number of entries. The approximation error is expressed as the ratio of the Frobenius norm of the residual to the Frobenius norm of the tensor

$$E_{\mathrm{Frob}} = \frac{||\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} \cdot_1 \mathbf{A} \cdot_2 \mathbf{B} \cdot_3 \mathbf{C}||_{\mathrm{F}}}{||\boldsymbol{\mathcal{X}}||_{\mathrm{F}}}.$$

Note that this error quantifies how closely the true, noiseless and complete tensor $\boldsymbol{\mathcal{X}}$ is approximated instead of the noisy tensor $\boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{N}}$. The results reported in figures and tables are obtained by generating multiple random tensors (for each value of the parameter that is varied in that experiment) and then taking the median of the results for each tensor.

In the experiments with synthetic data, the performance of our algorithm is compared to that of the following optimization-based baseline algorithms:

- `geomCG` is a variant of nonlinear conjugated gradients that uses Riemannian optimization techniques on the manifold of tensors of fixed multilinear rank [25]. In the experiments, this optimization algorithm starts from a random starting point, unless stated otherwise.
- `tucker_opt` updates the variables of a low multilinear rank tensor in blocks, that is, the factor matrices and core are updated at each iteration in an alternating fashion. The update for each block of variables is performed using the nonlinear conjugate gradient method [103].
- `HaLRTC` relaxes the problem of minimizing the multilinear rank to minimizing the sum of the nuclear norms of the mode-n unfoldings of the tensor, namely $\sum_{n=1}^{N} ||\mathbf{X}_{[n]}||_*$. The resulting convex problem is solved using the alternating direction method of multipliers [19].

Note that `geomCG` and `tucker_opt` compute a low multilinear rank approximation (LMLRA) instead of an MLSVD. This means that no constraints are imposed on the core tensor $\boldsymbol{\mathcal{G}}$ and factor matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$.

Accuracy comparison

Figure 3.3 compares the accuracy obtained by the algorithms on $100 \times 100 \times 100$ tensors of multilinear rank $(8, 8, 8)$ with varying levels of noise. The mask $\mathbf{W}$ has a block diagonal structure with 9 blocks of size 20×20, in which each block has an overlap of size 10×10 with the next block. This overlap must at least be greater than the multilinear rank along the corresponding mode to satisfy the second and fourth conditions in Eq. (3.27)[6]. This mask results in a sampling ratio of 28%. The median results over 20 repetitions are shown. This

[6]For this mask, only subsequent fully observed submatrices ($\mathbf{M}_{\text{obs}}^{(\ell)}$ and $\mathbf{M}_{\text{obs}}^{(\ell+1)}$) have rows in common. As a result, among all the submatrices $\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_2)}$ in $\boldsymbol{\nabla}^{(\mathbf{B})}$, only $\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+1)}$ for $1 \leq \ell_1 < L$ have a nonzero number of rows. The size of the overlap guarantees that each $\mathbf{B}_{\text{obs}}^{(\ell_1,\ell_1+1)}$ has full column rank, which in turn guarantees that $\boldsymbol{\nabla}^{(\mathbf{B})}$ has full column rank. The reasoning for $\boldsymbol{\nabla}^{(\mathbf{A})}$ is analogous.

observation pattern does not suit `HaLRTC` well, as it achieves better accuracy at the same sampling ratio if the observed entries are spread randomly throughout the tensor, as can be seen in the right subplot. At higher values of the SNR, the accuracy obtained by `tucker_opt` flattens off for the block diagonal mask. Algorithm `geomCG` achieves the highest level of accuracy, while `mlsvd_fs` is a bit less accurate. This is to be expected since optimization-based algorithms generally are more accurate than algebraic algorithms for noisy data. On the other hand, the following experiments will show that `mlsvd_fs` requires much less computation time.

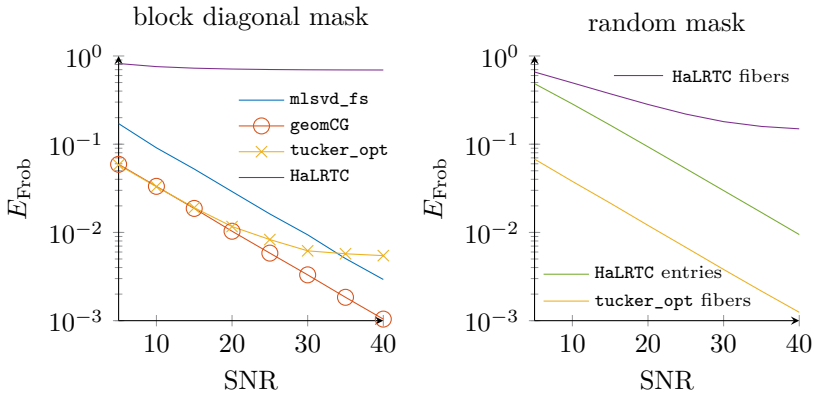


Figure 3.3: Accuracy for the fiber-wise observed tensor with a block diagonal mask $\mathbf{W}$ in the first experiment. The algorithm `mlsvd_fs`, as it is algebraic, obtains less accurate results than the optimization-based `geomCG` and `tucker_opt` algorithms; however, the difference in accuracy is limited (left). Algorithm `HaLRTC` is very inaccurate for this observation pattern. However, for an incomplete tensor with the same sampling ratio, the accuracy improves if the observed part consists of randomly selected fibers or randomly selected entries (right). Similarly, `tucker_opt` is more accurate at higher values of the SNR if the observed fibers are selected randomly.

Figure 3.4 concerns $100 \times 100 \times 100$ tensors of multilinear rank $(10, 10, 10)$ at varying sampling ratios and an SNR of 20 dB. The positions of the ones in the mask $\mathbf{W}$ are randomly selected. To compute $\mathbf{A}$ (resp. $\mathbf{B}$), fully observed submatrices of $\mathbf{X}_{[1;2,3]}$ (resp. $\mathbf{X}_{[2;1,3]}$) are now selected using a simple heuristic. Namely, we use the observed parts of the mode-2 (resp. 1) slices (Fig. 3.2 bottom) and also the largest fully observed submatrix of each pair of mode-2 (resp. 1) slices (Fig. 3.2 top). This figure shows that `HaLRTC` performs badly at lower sampling ratios for fiber-wise observed tensors. Furthermore, `mlsvd_fs` performs badly at the low sampling ratio of 10% because the uniqueness conditions in

Eq. (3.27) are not yet satisfied. This is because there are not enough fully observed submatrices of the mode-1 and mode-2 unfoldings of the tensor with the required rank to guarantee that $\nabla^{(\mathbf{A})}$ and $\nabla^{(\mathbf{B})}$ have full column rank. The optimization-based algorithms `geomCG` and `tucker_opt` achieve the highest level of accuracy. The gap between the accuracy obtained by `mlsvd_fs` and the optimization-based algorithms becomes very small at sampling ratios of 40% and higher. Because the selected fully observed submatrices are at most two slices wide, it might be possible to get the accuracy even closer to that of the optimization-based methods by selecting submatrices that span across more than two slices.

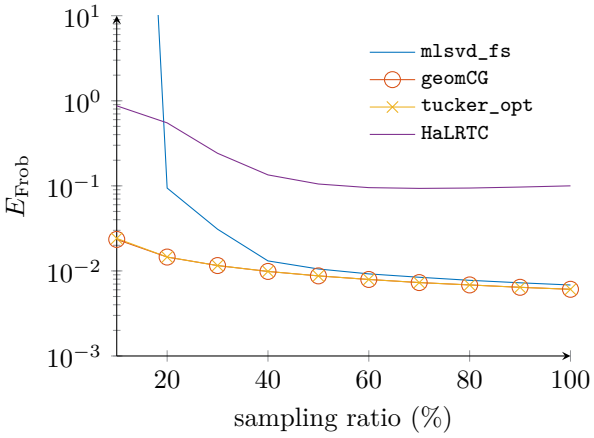


Figure 3.4: Accuracy for the fiber-wise observed tensor with a random mask $\mathbf{W}$ in the second experiment. At a sampling ratio of 10% the working conditions of `mlsvd_fs` are not yet satisfied. For higher sampling ratios, the accuracy obtained by `mlsvd_fs` is close to that obtained by the optimization-based algorithms.

Initialization of optimization algorithms

A standard approach to computing an LMLRA of a complete tensor is to first truncate the MLSVD, which can be computed algebraically, and further improve it using, for example, higher-order orthogonal iteration [91] or an optimization algorithm on manifolds [92], [93]. Overall, this reduces the number of iterations required until convergence and it also decreases the risk of convergence to a local, non-global optimum.

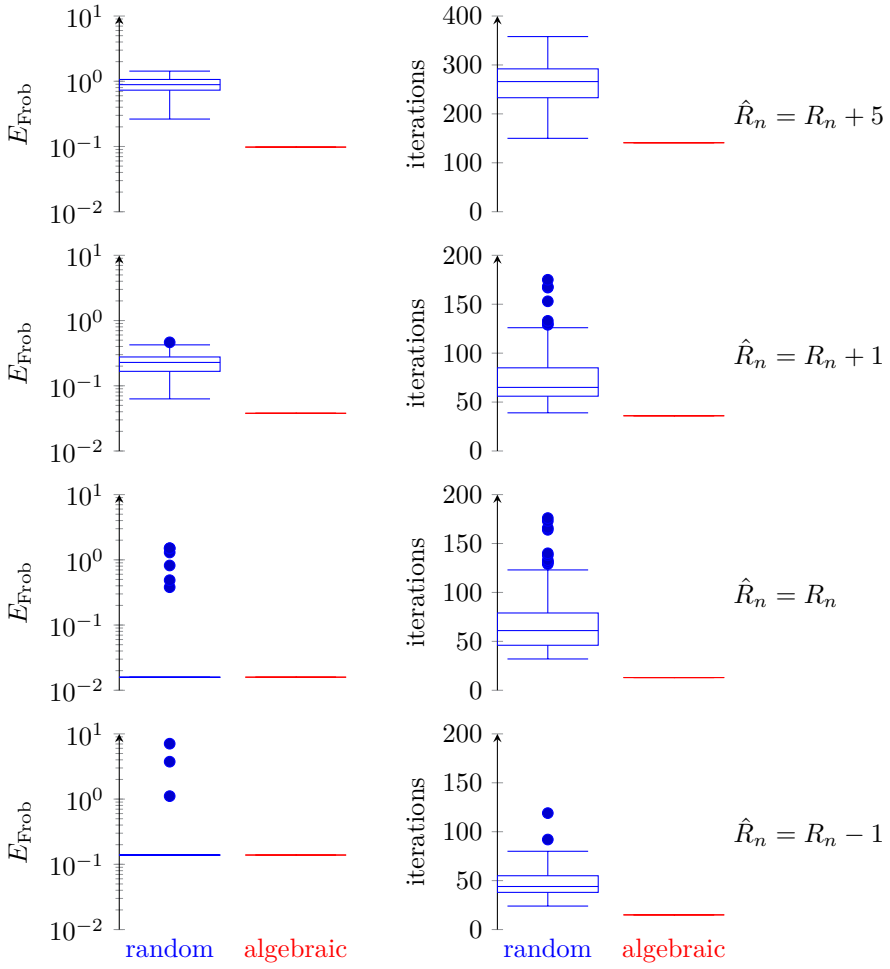


Figure 3.5: Comparison of random and algebraic initialization of `geomCG` in terms of accuracy (left) and number of iterations (right). In the case where the multilinear rank is overestimated, the optimization-based algorithm has difficulty finding good solutions, starting from 100 different random initializations. In the case where it is correctly estimated or underestimated, both initialization strategies yield equally good results in terms of accuracy. However, the algebraic initialization is still useful, since it significantly reduces the number of iterations required to reach this level of accuracy.

Our proposed algorithm offers a similar algebraic initialization strategy in the case in which an LMLRA is computed from a fiber-wise observed tensor. In a new experiment, an LMLRA of multilinear rank $\hat{R}$ is computed of a $50 \times 50 \times 50$ tensor of multilinear rank $R = (5, 5, 5)$ with a sampling ratio of 30% and the positions of the observed fibers are selected at random. An LMLRA is computed with multilinear rank $(\hat{R}_1, \hat{R}_2, \hat{R}_3)$, with $\hat{R}_n$ chosen equal to $R_n + 5, R_n + 1, R_n, R_n - 1$ for $n = 1, 2, 3$. The LMLRA is computed using `geomCG`. The purpose of the experiment is to compare 100 different random initializations of `geomCG` with one run of `geomCG` initialized using our algorithm. For the selection of the fully observed submatrices of $\mathbf{X}_{[1;2,3]}$ (resp. $\mathbf{X}_{[2;1,3]}$) in the computation of $\mathbf{A}$ (resp. $\mathbf{B}$), we took the simplest possible approach and used only the matrices of the type in Fig. 3.2 (bottom).

Figure 3.5 compares the two initialization strategies in terms of accuracy and the required number of iterations. In case the multilinear rank is overestimated, random initialization yields much less accurate results than algebraic initialization. The accuracy worsens as the overestimation of the multilinear rank increases, indicating that overfitting occurs since E_{Frob} is the error with respect to the noiseless and complete tensor $\boldsymbol{\mathcal{X}}$. In case the multilinear rank is correctly estimated or underestimated, both the random and algebraic initialization strategies yield approximately equally accurate results, except for a few outliers in the former case. However, the algebraic initialization still proves useful, as it reduces the number of iterations required to reach a solution with this level of accuracy significantly and avoids outliers.

Computation time

In this experiment, we compare the computation time of `mlsvd_fs`, `geomCG` and `tucker_opt`. It was performed on a laptop with an AMD Ryzen 7 PRO 3700U CPU and 30GB RAM. A random tensor of dimensions $I \times I \times I$ with multilinear rank $(5, 5, 5)$ and a sampling ratio of 20% is generated for several values of I. First, a solution is computed using `mlsvd_fs` using the slices approach. The accuracy of this solution is used to set the tolerance of the residual stopping criterion of the optimization algorithms to ensure a fair comparison. In other words, the optimization algorithms stop once they achieve the same level of accuracy as `mlsvd_fs`.

Figure 3.6 shows the total time required for each algorithm for increasing I. In this experiment, the computation time of each algorithm scales similarly as I increases. However, the required computation time for `mlsvd_fs` is respectively about two and three orders of magnitude less than for `geomCG` and `tucker_opt`. Another way to interpret this figure is that the difference between the `mlsvd_fs`

curve and the curve of an optimization algorithm is the computation time that can be saved by initializing using `mlsvd_fs`. This shows that `mlsvd_fs` can significantly speed up the computation of an LMLRA for a fiber-wise observed tensor.

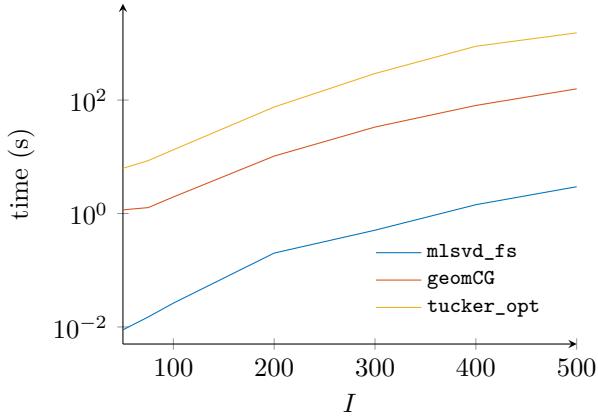


Figure 3.6: The computation time required by `geomCG` and `tucker_opt` to reach the same level of accuracy as `mlsvd_fs` is respectively about two and three orders of magnitude larger than the computation time of `mlsvd_fs` for the tensor dimensions I that were tested in this experiment.

3.6.2 Completion of traffic speed data

Traffic speed data is spatiotemporal, as it provides the speed of traffic at a particular road segment during an interval of time. It can either be collected by stationary sensors, located at different road segments, or it can be floating car data, that is, data collected by vehicles. However, the resulting datasets can have missing elements in both cases. In the former case, missing data might be caused by a sensor failure that persists for a period of multiple days until it is fixed. In the latter case, it could be that no car that collects data passes through a road segment during some intervals. These missing elements are often first filled in before using the data for other purposes, such as modeling traffic, estimating travel times and navigation.

The Guangzhou traffic dataset, which is used in this experiment, is floating car data collected from a navigation application. It contains the speed of traffic, aggregated over vehicles passing by a road segment in 10-minute intervals (144 intervals in each day), from the period from 1 August 2016 to 30 September 2016 (61 days) on 214 road segments [105]. The dataset thus corresponds to

a third-order tensor of dimensions $214 \times 61 \times 144$. Data that is missing for some road segments during one or more days correspond to completely missing mode-3 fibers. The data does indeed exhibit such a fiber-wise observation pattern since 1.28% out of the 1.29% missing data corresponds to completely missing fibers. The remaining 0.01% of missing entries are concentrated in three fibers that are partially missing and will be treated as completely missing for convenience.

In order to evaluate the capability of our algorithm to fill in the missing fibers, the available fibers are split into a training and a test set. Here, the sampling ratio refers to the ratio between the elements in the training set and the available elements in the entire dataset. The approximation error on the test set is quantified using the relative Frobenius norm of the residual and the relative absolute error, namely

$$E_{\text{Frob},\alpha} = \frac{\sqrt{\sum_{i\in\alpha} r_i^2}}{\sqrt{\sum_{i\in\alpha} x_i^2}} \qquad \text{and} \qquad E_{\text{MAE},\alpha} = \frac{\sum_{i\in\alpha} |r_i|}{\sum_{i\in\alpha} |x_i|},$$

in which $\mathbf{r} = \operatorname{vec}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} \cdot_1 \mathbf{A} \cdot_2 \mathbf{B} \cdot_3 \mathbf{C})$ and α contains the indices of the elements of the vectorized traffic data tensor $\mathbf{x} = \operatorname{vec}(\boldsymbol{\mathcal{X}})$ that belong to the test set. These error metrics are computed for the results obtained by `mlsvd_fs`, using only the matrices as in Fig. 3.2 (bottom), and also for the results obtained after further refinement with `geomCG`. We then compare the achieved accuracy with two related tensor-based algorithms, namely `HaLRTC` and `BGCP`. In `BGCP` the data is modeled as a low-rank tensor, i.e. a CPD, subject to additive independent Gaussian noise [115]. The factor matrices and the noise level are assumed to follow certain prior distributions, which allows the factor matrices to be estimated using Bayesian inference through Gibbs sampling. A multilinear rank of $(20, 7, 20)$ was selected for `mlsvd_fs` using a validation set and hyperparameters for `HaLRTC` were chosen in the same manner. For `BGCP`, the rank was set to 20 as this provided the best results in the case of fiber-wise missing data [115].

For each sampling ratio, 10 repetitions are performed in which the data is randomly split into different training and test sets. Table 3.1 shows the median errors of these repetitions for the different algorithms. In the current experiment, the further optimization-based refinement of the solution obtained by `mlsvd_fs` yields only a small improvement in accuracy. As a matter of fact, the accuracy achieved by `mlsvd_fs` is already comparable to `BGCP`, which was specifically designed for completing traffic data. After a little refinement, `mlsvd_fs` yields the most accurate results in most cases.

Sampling ratio (%)	50	60	70	80	90
	$E_{\mathrm{MAE},\alpha}$				
`mlsvd_fs`	0.0756	0.0744	0.0737	0.0731	0.0721
`mlsvd_fs` + `geomCG`	**0.0730**	**0.0717**	**0.0710**	0.0705	0.0699
`BGCP`	0.0743	0.0723	0.0712	**0.0704**	**0.0697**
`HaLRTC`	0.0823	0.0794	0.0770	0.0753	0.0734
	$E_{\mathrm{Frob},\alpha}$				
`mlsvd_fs`	0.1160	0.1122	0.1102	0.1094	0.1080
`mlsvd_fs` + `geomCG`	**0.1099**	**0.1068**	**0.1060**	0.1051	0.1044
`BGCP`	0.1122	0.1078	0.1062	**0.1045**	**0.1036**
`HaLRTC`	0.1121	0.1095	0.1073	0.1057	0.1041

Table 3.1: Unlike in the experiments with synthetic data, filling in the missing fibers of this traffic speed dataset yields a far smaller difference in accuracy between `mlsvd_fs` and `geomCG`. Additionally, the accuracy achieved by `mlsvd_fs` is comparable to that of the baseline algorithms.

3.7 Conclusion

In this chapter, we derive an algebraic algorithm for computing the MLSVD of a fiber-wise observed tensor and conditions that guarantee that the algorithm can uniquely retrieve the factor matrices and core tensor. The algorithm is similar in spirit to the basic MLSVD algorithm for fully observed tensors, but with some of the steps consisting of a structured matrix completion problem. This chapter included the development of algorithms for computing a column-wise orthonormal basis for the column space of incomplete matrices. Conditions for the unique retrieval of this column space are provided.

The performance of our algorithm is compared to related algorithms in experiments with synthetic and real-life data. The observations are analogous to the performance of the MLSVD of fully observed tensors. The accuracy of the algebraic procedure is often close to the accuracy achieved by optimization-based LMLRA algorithms while requiring much less computation time. Like in the case of fully observed tensors, the algebraic algorithm is a good initialization strategy for optimization-based algorithms. This strategy is especially useful if the multilinear rank of the underlying tensor is overestimated, as an optimization algorithm may struggle to find accurate solutions in this case. Additionally, the algorithm was applied to an incomplete traffic speed dataset. In this

application, the level of accuracy achieved by the algebraic algorithm, after further refinement with an optimization-based algorithm, is similar to or better than that achieved by relevant algorithms from the literature.

3.8 Proof of Lemma 3.3

Proof. The lemma follows with a simple technical derivation from equations Eq. (3.12), Eq. (3.15) and Eq. (3.16). We give a proof based on induction on n. The value of n is added to the notation of the following matrices: $\boldsymbol{\nabla}, \boldsymbol{\nabla}^{(\ell_1)}, \boldsymbol{\Theta}, \boldsymbol{\Theta}^{(\ell_1)}, \mathbf{L}$; these matrices are now denoted as $\boldsymbol{\nabla}^{(n)}, \boldsymbol{\nabla}^{(\ell_1,n)}, \boldsymbol{\Theta}^{(n)}, \boldsymbol{\Theta}^{(\ell_1,n)}, \mathbf{L}^{(n)}$ Consider the base case $n = 2$. We have

$$\boldsymbol{\Theta}^{(2)}\mathbf{L}^{(2)} = [\mathbf{B}^{(1,2)}, -\mathbf{B}^{(1,2)}] \begin{bmatrix} \mathbf{I}_R & \mathbf{0} \\ \mathbf{I}_R & \mathbf{I}_R \end{bmatrix} = [\mathbf{0}, -\mathbf{B}^{(1,2)}] = [\mathbf{0}_{J_{1,2}\times R}, -\boldsymbol{\nabla}^{(2)}],$$

which proves that the statement holds true for $n = 2$. We will now prove that if the statement holds true for $n = L - 1$, i.e.,

$$\boldsymbol{\Theta}^{(L-1)}\mathbf{L}^{(L-1)} = [\mathbf{0}_{(\sum_{1\le\ell_1<\ell_2\le L-1} J_{\ell_1,\ell_2})\times R}, -\boldsymbol{\nabla}^{(L-1)}],$$

then it also holds true for $n = L$, i.e.,

$$\boldsymbol{\Theta}^{(L)}\mathbf{L}^{(L)} = [\mathbf{0}_{(\sum_{1\le\ell_1<\ell_2\le L} J_{\ell_1,\ell_2})\times R}, -\boldsymbol{\nabla}^{(L)}].$$

Observe that

$$\mathbf{L}^{(L)} = \left[\begin{array}{c|c} \mathbf{L}^{(L-1)} & \mathbf{1}_{L-1} \otimes \mathbf{0}_{R\times R} \\ \hline \mathbf{1}_{L-1}^{\mathrm{T}} \otimes \mathbf{I}_R & \mathbf{I}_R \end{array}\right],$$

$$\boldsymbol{\Theta}^{(m,n)} = \left[\begin{array}{ccc|c} & \boldsymbol{\Theta}^{(m,n-1)} & & \mathbf{0}_{\sum_{i=m+1}^{n-1} J_{m,i}\times R} \\ \hline \mathbf{1}_{m-1}^{\mathrm{T}} \otimes \mathbf{0}_{J_{m,n}\times R} & \mathbf{B}^{(m,n)} & \mathbf{1}_{n-m-1}^{\mathrm{T}} \otimes \mathbf{0}_{J_{m,n}\times R} & -\mathbf{B}^{(m,n)} \end{array}\right].$$

Note also that the rightmost block column of $\boldsymbol{\Theta}^{(L)}$ is given by

$$\left[\begin{array}{c} \mathbf{0}_{\sum_{\ell=2}^{L-1} J_{1,\ell}\times R} \\ -\mathbf{B}^{(1,I)} \\ \hline \mathbf{0}_{\sum_{\ell=3}^{L-1} J_{2,\ell}\times R} \\ -\mathbf{B}^{(2,L)} \\ \hline \vdots \\ \hline \mathbf{0}_{J_{L-1,L}\times R} \\ -\mathbf{B}^{(L-2,L)} \\ \hline -\mathbf{B}^{(L-1,L)} \end{array}\right].$$

The zero pattern of the above block column of $\boldsymbol{\Theta}^{(L)}$ and the induction hypothesis $\boldsymbol{\Theta}^{(L-1)}\mathbf{L}^{(L-1)} = [\mathbf{0}_{(\sum_{1\leq \ell_1 < \ell_2 \leq L-1} J_{\ell_1,\ell_2})\times R}, -\boldsymbol{\nabla}^{(L-1)}]$ implies that we only need to investigate how the block rows in $\boldsymbol{\Theta}^{(L)}$ that are introduced when going from $n = L-1$ to $n = L$ are affected by $\mathbf{L}^{(L)}$. The introduced block rows in $\boldsymbol{\Theta}^{(L)}$, when going from $n = L-1$ to $n = L$, are of the form

$$\left[\mathbf{1}_{\ell-1}^{\mathrm{T}} \otimes \mathbf{0}_{J_{\ell,L}\times R}, \mathbf{B}^{(\ell,L)}, \mathbf{1}_{L-\ell+1}^{\mathrm{T}} \otimes \mathbf{0}_{J_{\ell,L}\times R}, -\mathbf{B}^{(\ell,L)}\right], \quad \ell = 1,\ldots,L-1.$$

Observe that the above block row vectors that are introduced when going from $n = L-1$ to $n = L$ only involve the newly introduced matrices $\mathbf{B}^{(1,L)},\ldots,\mathbf{B}^{(L-1,L)}$. The right multiplication of the above block row vectors with $\mathbf{L}^{(L)}$ yield

$$\left[\mathbf{1}_{\ell-1}^{\mathrm{T}} \otimes \mathbf{0}_{J_{\ell,L}\times R}, \mathbf{1}_{L-\ell}^{\mathrm{T}} \otimes -\mathbf{B}^{(\ell,L)}\right], \quad \ell = 1,\ldots,L-1.$$

Note that the above block row vector corresponds to the bottom block row vector of $\boldsymbol{\nabla}^{(\ell,L)}$ in $\boldsymbol{\nabla}^{(L)}$, i.e.,

$$\boldsymbol{\nabla}^{(\ell_1,L)} = \left[\begin{array}{c|c} \boldsymbol{\nabla}^{(\ell_1,L-1)} & \mathbf{0}_{(\sum_{\ell_2=\ell_1+1}^{L-1} J_{\ell_1,\ell_2})\times(L-\ell_1)R} \\ \hline \mathbf{1}_{\ell_1-1}^{\mathrm{T}} \otimes \mathbf{0}_{J_{\ell_1,L}\times R} & \mathbf{1}_{L-\ell_1}^{\mathrm{T}} \otimes -\mathbf{B}^{(\ell_1,L)} \end{array}\right].$$

We can now conclude that if the induction hypothesis holds true for $n = L-1$, i.e., $\boldsymbol{\Theta}^{(L-1)}\mathbf{L}^{(L-1)} = [\mathbf{0}_{(\sum_{1\leq \ell_1 < \ell_2 \leq L-1} J_{\ell_1,\ell_2})\times R}, -\boldsymbol{\nabla}^{(L-1)}]$, then it also holds true for $n = L$. □

4 Block row Kronecker-structured linear systems with a low-rank tensor solution

ABSTRACT

Several problems in compressed sensing and randomized tensor decomposition can be formulated as a structured linear system with a constrained tensor as the solution. In particular, we consider block row Kronecker-structured linear systems with a low multilinear rank multilinear singular value decomposition, a low-rank canonical polyadic decomposition or a low tensor train rank tensor train constrained solution. In this chapter, we provide algorithms that serve as tools for finding such solutions for a large, higher-order data tensor, given Kronecker-structured linear combinations of its entries. Consistent with the literature on compressed sensing, the number of linear combinations of entries needed to find a constrained solution is far smaller than the corresponding total number of entries in the original tensor. We derive conditions under which a multilinear singular value decomposition, canonical polyadic decomposition or tensor train solution can be retrieved from this type of structured linear systems and also derive the corresponding generic conditions. Finally, we validate our algorithms by comparing them to related randomized tensor decomposition algorithms and by reconstructing a hyperspectral image from compressed measurements.

This chapter is based on S. Hendrikx and L. De Lathauwer, "Block row Kronecker-structured linear systems with a low-rank tensor solution", *Frontiers in Applied Mathematics and Statistics*, vol. 8, pp. 41–57, Mar. 2022, Special Issue on High-Performance Tensor Computations in Scientific Computing and Data Science. DOI: 10.3389/fams.2022.832883.

4.1 Introduction

In a wide array of applications within signal processing, machine learning, and data analysis, sampling all entries of a dataset is infeasible. Datasets can be infeasibly large either because their dimensions are huge, like a matrix with millions of rows and columns, or because they are higher-order. In several cases, a relatively limited set of indirectly sampled datapoints, i.e. linear combinations $\mathbf{Ax} = \mathbf{b}$ of the datapoints $\mathbf{x}$, suffices for recovering an accurate approximation of the full dataset, making the problem tractable again. For example in compressed sensing [27], [30] and randomized tensor decomposition algorithms [117], random measurement matrices are used to compress the data $\mathbf{x}$. Directly sampling a subset of the datapoints can also be written in the format $\mathbf{Ax} = \mathbf{b}$, in which the measurement matrix $\mathbf{A}$ now consists of a subset of the rows of the identity matrix. Hence, problems such as incomplete tensor decomposition [16], non-uniform sampling [118] and cross-approximation [14] can also be formulated in this manner.

Retrieving the original data from such a linear system is generally only possible if it is overdetermined. This would mean that the number of indirectly sampled datapoints equals at least the total number of dataset entries, which is the opposite of what is needed in the compressed sensing (CS) setting. This requirement becomes especially restrictive for higher-order datasets. "Higher-order" means that the dataset consists of more than two dimensions or modes, in which case the number of entries increases exponentially with the number of modes. This phenomenon is commonly known as the curse of dimensionality.

Real data often allows compact representations thanks to some intrinsic structure, such as the data being generated by an underlying lower-dimensional process [9], [30]. In this case, $\mathbf{x}$ can be well approximated by a sparsifying basis $\mathbf{\Phi}$ and a sparse coefficient vector $\boldsymbol{\theta}$, namely $\mathbf{x} \approx \mathbf{\Phi}\boldsymbol{\theta}$. The literature on compressed sensing (CS) shows that for a measurement matrix and sparsifying basis pair with low coherence, the linear system can be solved in the underdetermined case [27], [30]. This means that $\mathbf{x}$ can be recovered using far fewer compressed measurements $\mathbf{Ax}$ than the total number of entries in $\mathbf{x}$, breaking the curse of dimensionality in the case of a higher-order dataset. An appropriate sparsifying basis is known a priori for some types of data, for example a wavelet basis for images, or can be obtained through dictionary learning [119]. If the sparsifying basis is known, then the measurement matrix can be chosen such that the coherence between the sparsifying basis and the measurement matrix is low. If no sparsifying basis for $\mathbf{x}$ is known a priori, one often chooses a random measurement matrix, as they are largely incoherent with any fixed basis [30].

In this chapter, we exploit intrinsic structure that is common in real data by

compactly approximating $\mathbf{x}$ using tensor decompositions, which in turn allows us to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ in the underdetermined case. This means that we will solve

$$\mathbf{A}\text{vec}\left(\boldsymbol{\mathcal{X}}\right) = \mathbf{b} \quad \text{with} \quad \boldsymbol{\mathcal{X}} \text{ of low rank} \quad \text{and} \quad \text{vec}\left(\boldsymbol{\mathcal{X}}\right) = \mathbf{x}. \tag{4.1}$$

Concretely, we will consider $\boldsymbol{\mathcal{X}}$ constrained to a multilinear singular value decomposition (MLSVD), a canonical polyadic decomposition (CPD) or a tensor train (TT) of low rank[1]. Note that at this point we have addressed the dimensionality issue only partially: on one hand, $\mathbf{x}$ is compactly modeled by a tensor decomposition; however, on the other hand, the number of columns of $\mathbf{A}$ remains equal to the total number of entries in $\mathbf{x}$ and thus still suffers from the curse of dimensionality. Therefore, we will employ a block row Kronecker-structured (BRKS) measurement matrix $\mathbf{A}$. Efficient algebraic algorithms for solving this linear system will be obtained by combining this Kronecker structure with the low-rank constraints on $\mathbf{x}$ such that $\mathbf{A}$ and $\mathbf{x}$ do not need to be fully constructed. A standard approach for computing the CPD of a tensor is to first orthogonally compress this tensor, for example using the MLSVD, and then compute the CPD of the compressed tensor [63]. In this chapter, we generalize this approach to the CS-setting.

Unstructured measurement matrices compress all modes of $\boldsymbol{\mathcal{X}}$ simultaneously. On the contrary, Kronecker-structured measurement matrices produce compressed versions of $\boldsymbol{\mathcal{X}}$ by compressing each mode individually, making them useful for higher-order datasets. Therefore, these Kronecker-structured measurement matrices are used in the CS-setting [39], [43], [62]. In [43] the CPD of a tensor is computed by first decomposing multiple compressed versions of $\boldsymbol{\mathcal{X}}$ and then retrieving the factor matrices of the full tensor under the assumption that their columns are sparse. There is some similarity between the Kronecker compressive sampling (KCS) approach [39] and ours, because it uses a Kronecker-structured measurement matrix and assumes that $\mathbf{x}$ is sparse in a Kronecker-structured basis. However, in KCS this basis is assumed to be known a priori, while in our approach it is estimated as well. In [120], a low-rank approximation to the solution of a parametrized set of linear systems, which can be rewritten as a large linear system in which the coefficient matrix consists of a sum of Kronecker products, is computed. This approach is suited towards applications such as solving partial differential equations rather than CS, as the requirement that the smaller individual systems should be overdetermined makes it infeasible for the latter purpose. Additionally, this approach utilizes the hierarchical Tucker decomposition to constrain the solution, as opposed to the MLSVD, CPD and TT constraints in this chapter.

[1] In this context, rank pertains to the definition of rank that corresponds to the respective tensor decomposition.

Algorithms similar to the ones in this chapter appear in the literature on randomized tensor decomposition (RTD) [117], [121]–[123]. The main difference is that the full tensor is available in such a randomized algorithm, while our algorithms can also be applied when only compressed measurements are available. In a randomized algorithm, the tensor is compressed in multiple modes to speed up further computations. In [123] the factor matrices of an MLSVD are computed by randomly compressing the tensor. However, this compression is carried out simultaneously in multiple modes in a manner that is not Kronecker-structured. Also, the full tensor is needed to retrieve the core $\boldsymbol{\mathcal{S}}$, as opposed to only compressed measurements like in our approach. A similar randomized approach for computing the MLSVD is introduced in [122], with the difference that the compression is carried out independently in different modes. An overview of RTD algorithms for computing an MLSVD is given in [117, Section 5.A]. In [62], a CPD is computed by decomposing multiple randomly compressed versions of the tensor in parallel and then combining the results. The algorithm in [124] improves upon this by replacing the dense random matrices with sketching matrices in the compression step to reduce the computational complexity. In [121], sketching matrices are used to speed up the least squares subproblems in the alternating least squares approach for computing the CPD. Multiplication with a sketching matrix is a Johnson-Lindenstrauss Transform (JLT), which transforms points in a high dimensional subspace to a lower dimensional subspace while preserving the distance between points up to a certain bound. In [125], it is proven that applying a JLT along each mode of a tensor is also a JLT. On the other hand, in cross approximation, CUR decompositions and pseudo-skeleton decompositions, a tensor is decomposed using a subset of directly sampled vectors along each mode [14], [17], [126], [127]. These subsets are determined on the basis of heuristics, for which algebraic results on the obtained quality of the approximation are available [128].

In the next part of this section, we introduce notations and definitions for further use in this chapter. In Section 4.2, we propose an algorithm for computing an MLSVD from a BRKS linear system and derive conditions under which a solution can be found. In Section 4.3, we generalize the standard approach for computing a CPD, in which the tensor is first compressed using its MLSVD, to computing a CPD from a BRKS linear system. We also derive conditions under which the CPD can be retrieved. In Section 4.4, we compute a TT from a BRKS linear system and derive conditions under which the TT can be retrieved. Finally, in Section 4.5, we validate our algorithms by computing tensor decompositions in a randomized approach using synthetic data and by applying them to a hyperspectral imaging application.

4.1.1 Additional notation and definitions

The mixed product property of the Kronecker product is

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

with $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{B} \in \mathbb{R}^{L \times M}$, $\mathbf{C} \in \mathbb{R}^{J \times K}$ and $\mathbf{D} \in \mathbb{R}^{M \times N}$. Similarly, the mixed product property of the Khatri–Rao product is

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

with $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{B} \in \mathbb{R}^{L \times M}$, $\mathbf{C} \in \mathbb{R}^{J \times K}$ and $\mathbf{D} \in \mathbb{R}^{M \times K}$. The rank of the Kronecker product of matrices equals the product of the ranks of those matrices, i.e.

$$\mathrm{r}(\mathbf{A} \otimes \mathbf{B}) = \mathrm{r}(\mathbf{A})\,\mathrm{r}(\mathbf{B})$$

In vectorized form, the MLSVD in Eq. (2.5) equals

$$\mathrm{vec}(\boldsymbol{\mathcal{X}}) = \left(\bigotimes_{n=1}^{N} \mathbf{U}^{(n)} \right) \mathrm{vec}(\boldsymbol{\mathcal{S}}) \tag{4.2}$$

and the CPD in Eq. (2.2) equals

$$\mathrm{vec}(\boldsymbol{\mathcal{X}}) = \left(\bigodot_{n=1}^{N} \mathbf{U}^{(n)} \right) \mathbf{1}_R.$$

In these equations, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is used instead of $\boldsymbol{\mathcal{T}}$ in Chapter 2. The reshaped version of a tensor $\boldsymbol{\mathcal{X}}$ is denoted by $\mathrm{reshape}(\boldsymbol{\mathcal{X}}, [J_1, \ldots, J_M])$, in which $J_1 \times \cdots \times J_M$ are the dimensions of the reshaped tensor.

4.2 Computing an MLSVD from a BRKS linear system

Using a BRKS linear system avoids the need for constructing and storing the full measurement matrix $\mathbf{A}$ and results in efficient algorithms for retrieving a low-rank constrained $\mathbf{x}$. With this structure, Eq. (4.1) becomes

$$\begin{bmatrix} \otimes_{n=1}^{N} \mathbf{A}^{(1,n)} \\ \otimes_{n=1}^{N} \mathbf{A}^{(2,n)} \\ \vdots \\ \otimes_{n=1}^{N} \mathbf{A}^{(M,n)} \end{bmatrix} \mathrm{vec}(\boldsymbol{\mathcal{X}}) = \begin{bmatrix} \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \\ \vdots \\ \mathbf{b}^{(M)} \end{bmatrix}, \tag{4.3}$$

with generating matrices $\mathbf{A}^{(m,n)} \in \mathbb{R}^{P_{mn} \times I_n}$ and compressed measurements $\mathbf{b}^{(m)} \in \mathbb{R}^{\prod_{n=1}^{N} P_{mn}}$ for $m = 1, \ldots, M$ and $n = 1, \ldots, N$. This type of linear system appears, for instance, in RTD and hyperspectral imaging, as illustrated in Section 4.5. Each block row of this linear system corresponds to a linear subsystem that produces a compressed version of $\boldsymbol{\mathcal{X}}$. This can be seen by tensorizing the mth block row as

$$\boldsymbol{\mathcal{B}}^{(m)} = \boldsymbol{\mathcal{X}} \cdot_1 \mathbf{A}^{(m,1)} \cdots \cdot_N \mathbf{A}^{(m,N)}, \tag{4.4}$$

in which $\boldsymbol{\mathcal{B}}^{(m)}$ is $\mathbf{b}^{(m)}$ reshaped into a tensor of dimensions $P_{m1} \times \cdots \times P_{mN}$. Each mode of $\boldsymbol{\mathcal{B}}^{(m)}$ has been compressed by mode-n multiplication with $\mathbf{A}^{(m,n)}$ for $n = 1, \ldots, N$.

If $\boldsymbol{\mathcal{X}}$ is approximately of low multilinear rank $(R_1, \ldots, R_N)$, reflecting some inherent structure, then the vectorized MLSVD in Eq. (4.2) can be substituted into Eq. (4.3):

$$\mathbf{A} \left(\bigotimes_{n=1}^{N} \mathbf{U}^{(n)} \right) \operatorname{vec}(\boldsymbol{\mathcal{S}}) = \mathbf{b}. \tag{4.5}$$

Note that the factor matrices would be in the reverse order when vectorizing conventionally, meaning $\otimes_{n=N}^{1} \mathbf{U}^{(n)}$. However, by vectorizing like in Eq. (4.5), the index n of the generating matrices $\mathbf{A}^{(m,n)}$ for $m = 1, \ldots, M$ and $n = 1, \ldots, N$ corresponds nicely to the mode it operates on.

In order to solve the linear system in Eq. (4.5) in the underdetermined case, it will not be directly solved for $\left(\otimes_{n=1}^{N} \mathbf{U}^{(n)}\right) \operatorname{vec}(\boldsymbol{\mathcal{S}})$. Instead, the factor matrices $\mathbf{U}^{(n)}$ for $n = 1, \ldots, N$ will be retrieved individually using the linear subsystems

$$\left(\bigotimes_{n=1}^{N} \mathbf{A}^{(m,n)} \right) \left(\bigotimes_{n=1}^{N} \mathbf{U}^{(n)} \right) \operatorname{vec}(\boldsymbol{\mathcal{S}}) = \mathbf{b}^{(m)} \quad \text{for } m = 1, \ldots, M \tag{4.6}$$

in Eq. (4.5). If the BRKS linear system consists of N linear subsystems, then all factor matrices can be computed. Therefore, we assume from this point on that $M = N$. The case where $M < N$ is of use when not all factor matrices need to be retrieved. Next, the core tensor will be retrieved using the computed factor matrices. The linear system in Eq. (4.6) is similar to the problem that is solved in KCS, namely $\otimes_{n=1}^{N} \mathbf{A}^{(m,n)}$, $\otimes_{n=1}^{N} \mathbf{U}^{(n)}$ and $\operatorname{vec}(\boldsymbol{\mathcal{S}})$, respectively, correspond to the Kronecker-structured measurement matrix, the Kronecker-structured sparsifying basis and the sparse coefficients. In CS, choosing a good basis that sparsifies the data [119] and determining the sparse coefficients of the data in this basis [27], [30] are separate problems. In this chapter, both problems are solved simultaneously using one BRKS linear system. Furthermore, unlike in the CS-setting, the coefficients $\operatorname{vec}(\boldsymbol{\mathcal{S}})$ are not necessarily sparse. In the remainder of this section, we discuss the methods for retrieving the factor matrices and the core tensor.

4.2.1 Computing the factor matrices

The mth linear subsystem in Eq. (4.6) can be simplified to

$$\left(\bigotimes_{n=1}^{N}\left(\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}\right)\right)\operatorname{vec}\left(\boldsymbol{\mathcal{S}}\right)=\mathbf{b}^{(m)}\quad\text{for } m=1,\ldots,N \tag{4.7}$$

using the mixed product property of the Kronecker product. This corresponds to a vectorized MLSVD with factor matrices $\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}$ for $n=1,\ldots,N$ and core $\boldsymbol{\mathcal{S}}$. Rearranging Eq. (4.7) as the mode-m matrix unfolding of this MLSVD

$$\begin{aligned}\mathbf{A}^{(m,m)}\mathbf{U}^{(m)}\mathbf{S}^{(m)}=\mathbf{B}^{(m)}_{[m]}\quad\text{with}\quad\mathbf{S}^{(m)}&=\left(\prod_{n\neq m}\boldsymbol{\mathcal{S}}\cdot_n\left(\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}\right)\right)_{[m]}\\&=\mathbf{S}_{[m]}\left(\bigotimes_{n\neq m}\left(\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}\right)\right)^{\mathrm{T}},\end{aligned} \tag{4.8}$$

for $m=1,\ldots,N$ shows that solving the linear system

$$\mathbf{A}^{(m,m)}\mathbf{F}^{(m)}=\mathbf{B}^{(m)}_{[m]}\quad\text{with}\quad\mathbf{F}^{(m)}=\mathbf{U}^{(m)}\mathbf{S}^{(m)}\quad\text{for } m=1,\ldots,N \tag{4.9}$$

for the unknown matrix $\mathbf{F}^{(m)}$ yields linear combinations of the columns of $\mathbf{U}^{(m)}$. Therefore, the dominant column space of $\mathbf{F}^{(m)}$ is the same as the subspace spanned by the columns of $\mathbf{U}^{(m)}$ if $\mathbf{S}^{(m)}$ is of full column rank. This means that we can find the column space of the mth factor matrix by computing an orthonormal basis that spans the dominant column space of $\mathbf{F}^{(m)}$. In applications that allow the generating matrices to be chosen, setting $\mathbf{A}^{(m,m)}=\mathbf{I}_{I_m}$ for $m=1,\ldots,N$ avoids the need for solving this linear system altogether. In this case, $\boldsymbol{\mathcal{B}}^{(m)}$ in Eq. (4.4) equals $\boldsymbol{\mathcal{X}}$ multiplied in every mode except the mth with a generating matrix, as illustrated in Fig. 4.1. This situation is similar to RTD methods for computing an MLSVD, where the tensor is compressed in all modes except the mth in order to retrieve $\mathbf{U}^{(m)}$ [122], [123]. Generically, the dimensions P_{mn}, for $m,n=1,\ldots,N$ and $n\neq m$, of the generating matrices can be chosen much smaller than the rank R_n of the mode-n unfolding of $\boldsymbol{\mathcal{X}}$, as we will further explain in Section 4.2.4. As a first indication, note that for the factorization in the right-hand side of Eq. (4.9), we expect $\prod_{n\neq m}P_{mn}=R_m$ to be sufficient, allowing $P_{mn}\ll R_m$ for $m,n=1,\ldots,N$ and $n\neq m$.

4.2.2 Computing the core tensor

In some applications, not only the factor matrices $\mathbf{U}^{(n)}$ for $n=1,\ldots,N$ but also the core $\boldsymbol{\mathcal{S}}$ are needed. At first sight, it looks like a data efficient way to

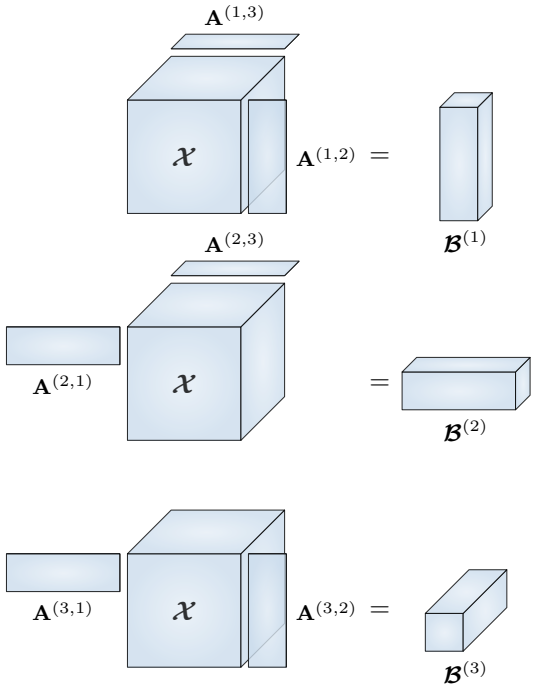


Figure 4.1: The mth block row of the linear system in Eq. (4.3) produces a version of $\boldsymbol{\mathcal{X}}$ that is compressed in each mode except the mth mode. Each compressed tensor can be used to retrieve one factor matrix of the MLSVD, which is similar to the RTD algorithm in [122].

compute the core is reusing the linear combinations with which the column space of the factor matrices has been computed. The core tensor that corresponds to the retrieved column spaces of the factor matrices is then obtained by solving the linear system in Eq. (4.5) for $\operatorname{vec}(\boldsymbol{\mathcal{S}})$, with the column space of $\mathbf{U}^{(n)}$ already available for $n = 1, \ldots, N$, which is

$$\begin{bmatrix} \otimes_{n=1}^{N} \left(\mathbf{A}^{(1,n)}\mathbf{U}^{(n)}\right) \\ \otimes_{n=1}^{N} \left(\mathbf{A}^{(2,n)}\mathbf{U}^{(n)}\right) \\ \vdots \\ \otimes_{n=1}^{N} \left(\mathbf{A}^{(N,n)}\mathbf{U}^{(n)}\right) \end{bmatrix} \operatorname{vec}(\boldsymbol{\mathcal{S}}) = \mathbf{b}. \tag{4.10}$$

The core can be retrieved if the coefficient matrix of this system is of full column rank. However, the requirement that this coefficient matrix must be of full column rank imposes much more severe constraints on the dimensions P_{mn}

than the constraints for computing the factor matrices in Section 4.2.4. Under these harder constraints, the right hand sides $\mathbf{b}^{(n)}$ would have to contain far more compressed measurements than actually needed to retrieve the nth factor matrix $\mathbf{U}^{(n)}$ for $n = 1, \ldots, N$. The reason is that in the coefficient matrix in Eq. (4.10) there are many linear dependencies between the rows, because the same factor matrices $\mathbf{U}^{(n)}$ appear in each block row. Therefore, it is not practical to compute the core by solving the system in Eq. (4.10) for $\operatorname{vec}(\boldsymbol{\mathcal{S}})$.

Instead, the core can be computed from a separate linear system, independent from the N block rows of the BRKS system used for computing the factor matrices, namely

$$\left(\bigotimes_{n=1}^{N} \mathbf{C}^{(n)}\right) \operatorname{vec}(\boldsymbol{\mathcal{X}}) = \mathbf{d},$$

with extra generating matrices $\mathbf{C}^{(n)} \in \mathbb{R}^{Q_n \times I_n}$ for $n = 1, \ldots, N$ and compressed measurements $\mathbf{d} \in \mathbb{R}^{Q_1 Q_2 \cdots Q_N}$. By substituting the vectorized MLSVD in Eq. (4.2) for $\boldsymbol{\mathcal{X}}$ and using the mixed product property of the Kronecker product, we obtain a linear system

$$\left(\bigotimes_{n=1}^{N} \left(\mathbf{C}^{(n)}\mathbf{U}^{(n)}\right)\right) \operatorname{vec}(\boldsymbol{\mathcal{S}}) = \mathbf{d} \tag{4.11}$$

that can be solved for the vectorized core $\operatorname{vec}(\boldsymbol{\mathcal{S}})$.

Solving a separate linear system for $\boldsymbol{\mathcal{S}}$ allows us to choose the dimensions Q_n for $n = 1, \ldots, N$ such that the core can be retrieved, without the need to increase the dimensions P_{mn} for $m, n = 1, \ldots, N$. Furthermore, this additional system can be solved efficiently by subsequently solving smaller linear systems for $\tilde{\mathbf{D}}^{(1)}, \ldots, \tilde{\mathbf{D}}^{(N)}$

$$\begin{aligned}
&\left(\mathbf{C}^{(1)}\mathbf{U}^{(1)}\right)\tilde{\mathbf{D}}^{(1)} = \mathbf{D}^{(1)} && \mathbf{D}^{(1)} = \mathbf{D}_{[1]} \quad \text{with } \operatorname{vec}(\boldsymbol{\mathcal{D}}) = \mathbf{d} \\
&\left(\mathbf{C}^{(2)}\mathbf{U}^{(2)}\right)\tilde{\mathbf{D}}^{(2)} = \mathbf{D}^{(2)} && \mathbf{D}^{(2)} = \operatorname{reshape}\left(\tilde{\mathbf{D}}^{(1)}, [Q_2, R_1 Q_3 \cdots Q_N]\right) \\
&\qquad\vdots && \qquad\vdots \\
&\left(\mathbf{C}^{(N)}\mathbf{U}^{(N)}\right)\tilde{\mathbf{D}}^{(N)} = \mathbf{D}^{(N)} && \mathbf{D}^{(N)} = \operatorname{reshape}\left(\tilde{\mathbf{D}}^{(N-1)}, [Q_N, R_1 \cdots R_{N-1}]\right),
\end{aligned}$$

with $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{Q_1 \times \cdots \times Q_N}$. The solution $\tilde{\mathbf{D}}^{(N)}$ of the last system is equal to $\mathbf{S}_{[N]}$, the mode-N matrix unfolding of $\boldsymbol{\mathcal{S}}$. These linear systems, with coefficient matrices of size $Q_n \times R_n$ for $n = 1, \ldots, N$, are much smaller than the linear system in Eq. (4.10), consisting of all compressed measurements used to estimate the factor matrices, with a coefficient matrix of size $\sum_{m=1}^{N} \left(\prod_{n=1}^{N} P_{mn}\right) \times \prod_{n=1}^{N} R_n$. This coefficient matrix is so large because of the Kronecker products. In Section 4.2.4, we show that generically the core can be retrieved if the dimensions Q_n are

greater than or equal to R_n for $n = 1, \ldots, N$. Therefore, subsequently solving the N smaller systems is a much less computationally expensive approach for computing $\boldsymbol{\mathcal{S}}$ than solving the system in Eq. (4.10) is. This additional linear system can be added to the BRKS linear system in Eq. (4.3) by renaming $\mathbf{C}^{(n)}$ for $n = 1, \ldots, N$ and $\mathbf{d}$ to $\mathbf{A}^{(N+1,n)}$ and $\mathbf{b}^{(N+1)}$ respectively. The complete BRKS linear system now consists of $N+1$ block rows with generating matrices $\mathbf{A}^{(m,n)} \in \mathbb{R}^{P_{mn} \times I_n}$ and compressed measurements $\mathbf{b}^{(m)} \in \mathbb{R}^{\prod_{n=1}^{N} P_{mn}}$ for $m = 1, \ldots, N+1$ and for $n = 1, \ldots, N$.

If we estimate the core tensor using a separate linear system, we do not use all available compressed measurements, since we disregard the first N block rows of the BRKS linear system. This can be resolved by solving the full BRKS linear system for $\operatorname{vec}(\boldsymbol{\mathcal{S}})$ with a numerical algorithm such as conjugate gradients, using the matrix-vector product

$$\begin{bmatrix} \otimes_{n=1}^{N} \left(\mathbf{A}^{(1,n)} \mathbf{U}^{(n)} \right) \\ \vdots \\ \otimes_{n=1}^{N} \left(\mathbf{A}^{(N+1,n)} \mathbf{U}^{(n)} \right) \end{bmatrix} \operatorname{vec}(\boldsymbol{\mathcal{S}}),$$

which can be computed efficiently by exploiting the block rowwise Kronecker structure. Alternatively, all block rows of the system can also be used if the full BRKS system is solved for a sparse core $\boldsymbol{\mathcal{S}}$, which we will illustrate in an experiment in Section 4.5.

4.2.3 Algorithm

By combining the steps in Section 4.2.1 and Section 4.2.2, we obtain Algorithm 4.1 for computing the MLSVD of $\boldsymbol{\mathcal{X}}$ from a BRKS linear system. In the outlined version of the algorithm, the additional linear system in Eq. (4.11) is used to estimate the core tensor. The computation of each factor matrix depends only on one compressed tensor $\boldsymbol{\mathcal{B}}^{(m)}$ for $m = 1, \ldots, N$, allowing the different factor matrices to be computed in parallel. In the RTD-setting, the full tensor is available and is then randomly compressed in order to efficiently compute a tensor decomposition. On the other hand, compressed measurements can be obtained directly in the data acquisition stage in the CS-setting. To accommodate for both settings, Algorithm 4.1 accepts either the generating matrices and the compressed measurements or the generating matrices and the full tensor as inputs.

Algorithm 4.1: MLSVD from a BRKS linear system (`lsmlsvd_brks`).

Input: (CS-setting) $\mathbf{A}^{(m,n)}$, $\mathbf{b}^{(m)}$ for $m = 1, \ldots, N+1; n = 1, \ldots, N$
Input: (RTD-setting) $\mathbf{A}^{(m,n)}$ for $m = 1, \ldots, N+1; n = 1, \ldots, N$ and $\boldsymbol{\mathcal{X}}$
Output: $\boldsymbol{\mathcal{S}}$, $\mathbf{U}^{(m)}$ for $m = 1, \ldots, N$
begin
  if *RTD-setting* **then**
    Compute $\mathbf{b}^{(m)}$ for $m = 1, \ldots, N$ using Eq. (4.3)
  `/* Estimate factor matrices */`
  for $m = 1$ **to** N **do**
    Solve $\mathbf{A}^{(m,m)}\mathbf{F}^{(m)} = \mathbf{B}^{(m)}_{[m]}$ for $\mathbf{F}^{(m)}$
    Find $\mathbf{U}^{(m)}$ by computing an orthonormal basis that spans the dominant column space of $\mathbf{F}^{(m)}$
  `/* Estimate core tensor */`
  $\tilde{\mathbf{S}} = \text{reshape}\left(\mathbf{b}^{(N+1)}, [P_{N+1,1}, \prod_{i=2}^{N} P_{N+1,i}]\right)$
  for $n = 1$ **to** N **do**
    Solve $\left(\mathbf{A}^{(N+1,n)}\mathbf{U}^{(n)}\right)\tilde{\mathbf{S}}_{\text{new}} = \tilde{\mathbf{S}}$ for $\tilde{\mathbf{S}}_{\text{new}}$
    $\tilde{\mathbf{S}} = \text{reshape}\left(\tilde{\mathbf{S}}_{\text{new}}, [P_{N+1,n}, \prod_{i=1}^{n-1} R_i \prod_{i=n+1}^{N} P_{N+1,i}]\right)$

4.2.4 Conditions for MLSVD retrieval

In this section, we derive the conditions in Theorem 4.1, under which the retrieval of the MLSVD of $\boldsymbol{\mathcal{X}}$ from a BRKS system is guaranteed. Since the factor matrices and the core are intrinsic to the data $\boldsymbol{\mathcal{X}}$, the conditions in this section are to be seen as constraints on the generating matrices.

Theorem 4.1. *Consider a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ of multilinear rank $(R_1, \ldots, R_N)$, admitting an MLSVD $[\![\boldsymbol{\mathcal{S}}; \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}]\!]$ with factor matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ for $n = 1, \ldots, N$ and core tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$. Given linear combinations $\mathbf{b}$ of $vec(\boldsymbol{\mathcal{X}})$ obtained from a BRKS linear system with generating matrices $\mathbf{A}^{(m,n)} \in \mathbb{R}^{P_{mn} \times I_n}$ for $m = 1, \ldots, N+1$ and $n = 1, \ldots, N$, the factor matrices $\mathbf{U}^{(n)}$ for $n = 1, \ldots, N$ can be retrieved if and only if*

$$1) \quad \text{r}\left(\mathbf{A}^{(m,m)}\right) = I_m \quad \text{for } m = 1, \ldots, N;$$

$$2) \quad \prod_{n \neq m}^{N} \text{r}\left(\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}\right) \geq R_m \quad \text{for } m = 1, \ldots, N.$$

The core $\boldsymbol{\mathcal{S}}$ *can be retrieved if and only if*

$$3) \prod_{n=1}^{N} \mathrm{r}\left(\mathbf{A}^{(N+1,n)}\mathbf{U}^{(n)}\right) = \prod_{n=1}^{N} R_n.$$

Proof. Condition 1): The linear system in Eq. (4.9) can be uniquely solved for $\mathbf{F}^{(m)}$ if and only if $\mathbf{A}^{(m,m)}$ is of full column rank I_m for $m = 1, \ldots, N$.
Condition 2): An orthonormal basis of dimension R_m for the column space of $\mathbf{F}^{(m)}$ can be retrieved if and only if

$$\mathrm{r}\left(\mathbf{U}^{(m)}\mathbf{S}^{(m)}\right) = \mathrm{r}\left(\mathbf{U}^{(m)}\mathbf{S}_{[m]}\left(\bigotimes_{n\neq m}\left(\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}\right)\right)\right) = R_m \quad \text{for } m = 1, \ldots, N$$

holds. Since $\mathrm{r}\left(\mathbf{U}^{(m)}\right) = \mathrm{r}\left(\boldsymbol{\mathcal{S}}_{[m]}\right) = R_m$, which follows from the definition of the MLSVD, this condition reduces to

$$\mathrm{r}\left(\bigotimes_{n\neq m}\left(\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}\right)\right) = \prod_{n\neq m}^{N} \mathrm{r}\left(\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}\right) \geq R_m \quad \text{for } m = 1, \ldots, N.$$

Condition 3): The linear system in Eq. (4.11) can be uniquely solved for $\mathrm{vec}\left(\boldsymbol{\mathcal{S}}\right)$ if and only if the coefficient matrix $\otimes_{n=1}^{N}\left(\mathbf{A}^{(N+1,n)}\mathbf{U}^{(n)}\right)$ is of full column rank:

$$\mathrm{r}\left(\bigotimes_{n=1}^{N}\mathbf{A}^{(N+1,n)}\mathbf{U}^{(n)}\right) = \prod_{n=1}^{N}\mathrm{r}\left(\mathbf{A}^{(N+1,n)}\mathbf{U}^{(n)}\right) = \prod_{n=1}^{N} R_n.$$

□

The conditions in Theorem 4.1 are satisfied in the generic case, in which the generating matrices are sampled from a continuous probability distribution and thus are of full rank with probability 1, if and only if the conditions in Theorem 4.2 hold. The conditions in the latter theorem allow us to determine the dimensions P_{mn} for $m = 1, \ldots, N+1$ and $n = 1, \ldots, N$ of the generating matrices such that generically the MLSVD of $\boldsymbol{\mathcal{X}}$ can be retrieved.

Theorem 4.2. *With generic generating matrices* $\mathbf{A}^{(m,n)}$ for $m = 1, \ldots, N+1$ *and* $n = 1, \ldots, N$*, the conditions in Theorem 4.1 hold if and only if*

1) $P_{mm} \geq I_m \quad \text{for } m = 1, \ldots, N;$

2) $\prod_{n\neq m} \min\left(P_{mn}, R_n\right) \geq R_m \quad \text{for } m = 1, \ldots, N;$

3) $P_{N+1,n} \geq R_n \quad \text{for } n = 1, \ldots, N.$

Proof. The proof of these conditions follows from the conditions in Theorem 4.1 and the properties of generic matrices.
Condition 1): Since a generic matrix is of full rank, $\mathrm{r}\left(\mathbf{A}^{(m,m)}\right) = \min(P_{mm}, I_m)$ for $m = 1, \ldots, N$.
Condition 2): The matrix product $\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}$ is of full rank $\min\left(P_{mn}, R_n\right)$ with probability 1 for a generic matrix $\mathbf{A}^{(m,n)}$ and $\mathbf{U}^{(n)}$ of full column rank (the latter following from the definition of an MLSVD) with $P_{mn}, R_n \leq I_n$ for $m = 1, \ldots, N+1$ and $n = 1, \ldots, N$ [43, Lemma 1].
Condition 3): Similar to the proof of condition 2), generically $\mathrm{r}\left(\mathbf{A}^{(N+1,n)}\mathbf{U}^{(n)}\right) = \min\left(P_{N+1,n}, R_n\right)$ for $n = 1, \ldots, N$ according to [43, Lemma 1]. Condition 3) in Theorem 4.1 then reduces to $\prod_{n=1}^{N} \min(P_{N+1,n}, R_n) \geq \prod_{n=1}^{N} R_n$, which holds if and only if condition 3) in this theorem is satisfied. □

In practice, the dimensions P_{mn} can easily be chosen such that the second generic condition is satisfied with $P_{mn} < R_n$ for $m, n = 1, \ldots, N$ and $n \neq m$. The second condition then reduces to $\prod_{n \neq m}^{N} P_{mn} \geq R_m$ for $m = 1, \ldots, N$. Assuming that P_{mn} for $m = 1, \ldots, N$ and $n \neq m$ are approximately equally large, we obtain the condition $P_{mn} \geq \sqrt[N-1]{R_m}$, meaning that the generating matrices $\mathbf{A}^{(m,n)} \in \mathbb{R}^{P_{mn} \times I_n}$ for $m, n = 1, \ldots, N$ and $n \neq m$ can be chosen as fat matrices. Therefore, the mth block row of Eq. (4.3) compresses all modes of $\boldsymbol{\mathcal{X}}$ except the mth, as illustrated in Fig. 4.1, similar to an RTD algorithm for the MLSVD. Additionally, this implies that $\mathbf{B}_{[m]}^{(m)} \in \mathbb{R}^{P_{mm} \times \prod_{n \neq m} P_{mn}}$ holds at least R_m columns, which is indeed the minimum required number for estimating the R_m-dimensional mode-m subspace of $\boldsymbol{\mathcal{X}}$. Algorithm 4.1 uses an oversampling factor $q \geq 1$ such that this matrix holds more than the minimum required number of columns, namely $\prod_{n \neq m} P_{mn} = qR_m$ columns for $m = 1, \ldots, N$. These additional compressed measurements allow a better estimation of the factor matrices if the data is noisy.

4.2.5 Noisy data

In applications, noise can be present on the compressed measurements and/or on the entries of the tensor. In the former case, the linear system becomes $\mathbf{A}\mathrm{vec}\left(\boldsymbol{\mathcal{X}}\right) = \mathbf{b} + \mathbf{n}$, in which the noise vector $\mathbf{n}$ is partitioned into subvectors $\mathbf{n}^{(m)}$ for $m = 1, \ldots, N$, consistent with the partitioning of $\mathbf{b}$ in Eq. (4.3). The linear system in Eq. (4.8), with a noise term $\mathbf{n}$, becomes

$$\mathbf{A}^{(m,m)}\mathbf{F}^{(m)} = \mathbf{B}_{[m]}^{(m)} + \mathbf{N}_{[m]}^{(m)} \quad \text{for } m = 1, \ldots, N,$$

in which $\boldsymbol{\mathcal{N}}^{(m)} \in \mathbb{R}^{P_{m1} \times \cdots \times P_{mN}}$ is the tensor representation of $\mathbf{n}^{(m)}$ and $\mathbf{N}_{[m]}^{(m)}$ its mode-m matrix unfolding. The rank of the matrix, obtained by solving

this subsystem for $\mathbf{F}^{(m)}$, generically equals $\min\left(I_m, \prod_{n\neq m}^{N} P_{mn}\right)$ due to the presence of the noise and thus exceeds R_m if $q > 1$ for $m = 1, \ldots, N$. If the signal-to-noise ratio $\frac{||\mathbf{b}||_2}{||\mathbf{n}||_2}$ is sufficiently high, then the dominant column space of $\mathbf{F}^{(m)}$ is nevertheless expected to be a good approximation for the column space of $\mathbf{U}^{(m)}$. A basis for this dominant column space can be computed with the SVD and the approximation is optimal in least squares sense if $\mathbf{n}^{(m)}$ is white Gaussian noise.

In the case of noisy tensor entries, the linear system becomes $\mathbf{A}\text{vec}\left(\boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{N}}\right) = \mathbf{b}$, with $\boldsymbol{\mathcal{N}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. In contrast with the former case, the coefficient matrix now also operates on the noise. Equation (4.8) then becomes

$$\mathbf{A}^{(m,m)}\left(\mathbf{F}^{(m)} + \mathbf{N}^{(m)}\right) = \mathbf{B}_{[m]}^{(m)} \quad \text{with} \quad \mathbf{N}^{(m)} = \left(\prod_{n\neq m}\left(\boldsymbol{\mathcal{N}} \cdot_i \mathbf{A}^{(m,n)}\right)\right)_{[m]}$$

for $m = 1, \ldots, N$. Because of the noise, the rank of $\mathbf{F}^{(m)} + \mathbf{N}^{(m)}$ generically exceeds R_m for $m = 1, \ldots, N$. The least squares optimal approximation of the column space of $\mathbf{U}^{(m)}$ can again be retrieved using the SVD if $\left(\otimes_{n=1}^{N} \mathbf{A}^{(m,n)}\right)\text{vec}\left(\boldsymbol{\mathcal{N}}\right)$ is white Gaussian noise. As derived in [62], this holds true if $\text{vec}\left(\boldsymbol{\mathcal{N}}\right)$ is white Gaussian noise and $\mathbf{A}^{(m,n)}\mathbf{A}^{(m,n)^{\mathrm{T}}} = \mathbf{I}_{P_{mn}}$ for $m, n = 1, \ldots, N$. The latter condition is approximately satisfied for large tensor dimensions if the generating matrices are sampled from a zero-mean uncorrelated distribution.

4.3 Computing an orthogonally compressed CPD from a BRKS linear system

Since the core of the MLSVD of an Nth order tensor is also an Nth order tensor, the MLSVD suffers from the curse of dimensionality. On the contrary, the number of parameters of the CPD scales linearly with the order of the tensor. Additionally, the CPD is unique under mild conditions. For applications that are more suited to these properties, such as blind signal separation, we introduce an algorithm for computing a CPD from a BRKS linear system in this section. An efficient, three-step approach for computing the CPD of a tensor is: 1) compressing the tensor, 2) decomposing the compressed tensor and 3) expanding the factor matrices to the original dimensions. In [63], the tensor is orthogonally compressed using the factor matrices of its MLSVD, i.e. the orthogonally compressed tensor corresponds to the core tensor of the MLSVD. In this section, we generalize this popular approach to computing a CPD from

the BRKS linear system in Eq. (4.3). Following this approach, we can find the CPD of a large tensor of order N by computing the CPDs of N small tensors.

4.3.1 Computing the factor matrices

If $\boldsymbol{\mathcal{X}}$ is approximately of rank R, it admits a CPD

$$\boldsymbol{\mathcal{X}} = \left[\!\left[\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(N)} \right]\!\right] \tag{4.12}$$

with $\mathbf{W}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$. Since the dimension of the subspace spanned by the mode-n vectors of $\boldsymbol{\mathcal{X}}$ is at most of dimension R for $n = 1, \ldots, N$, as $\boldsymbol{\mathcal{X}}$ is of rank R, $\boldsymbol{\mathcal{X}}$ also admits an MLSVD

$$\boldsymbol{\mathcal{X}} = \left[\!\left[\boldsymbol{\mathcal{S}}; \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)} \right]\!\right] \tag{4.13}$$

with $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$ and $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{R \times \cdots \times R}$. Note that the rank R_n of the mode-n vectors of $\boldsymbol{\mathcal{X}}$ can be smaller than R for any $n = 1, \ldots, N$, namely when $\mathbf{W}^{(n)}$ is not of full column rank. In this case, an orthogonal compression matrix $\mathbf{U}^{(n)}$ of dimensions $I_n \times R_n$ can still be used. The combination of Eq. (4.12) and Eq. (4.13) implies that the core tensor $\boldsymbol{\mathcal{S}}$ admits the following CPD:

$$\boldsymbol{\mathcal{S}} = \left[\!\left[\mathbf{U}^{(1)^{\mathrm{T}}} \mathbf{W}^{(1)}, \ldots, \mathbf{U}^{(N)^{\mathrm{T}}} \mathbf{W}^{(N)} \right]\!\right] = \left[\!\left[\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(N)} \right]\!\right]. \tag{4.14}$$

Alternatively, the CPD of $\boldsymbol{\mathcal{X}}$ can be written as

$$\boldsymbol{\mathcal{X}} = \left[\!\left[\mathbf{U}^{(1)} \mathbf{V}^{(1)}, \ldots, \mathbf{U}^{(N)} \mathbf{V}^{(N)} \right]\!\right]$$

by substituting Eq. (4.14) into Eq. (4.13). This CPD can be substituted for $\boldsymbol{\mathcal{S}}$ in the BRKS system in Eq. (4.3). Using the mixed product property of the Khatri–Rao product, the mth block row of the BRKS system becomes

$$\left(\bigodot_{n=1}^{N} \left(\mathbf{A}^{(m,n)} \mathbf{U}^{(n)} \mathbf{V}^{(n)} \right) \right) \mathbf{1}_R = \mathbf{b}^{(m)} \quad \text{for } m = 1, \ldots, N.$$

Tensorizing this equation yields a polyadic decomposition (PD) expression for each block row of the BRKS system:

$$\left[\!\left[\mathbf{A}^{(m,1)} \mathbf{U}^{(1)} \mathbf{V}^{(1)}, \ldots, \mathbf{A}^{(m,N)} \mathbf{U}^{(N)} \mathbf{V}^{(N)} \right]\!\right] = \boldsymbol{\mathcal{B}}^{(m)} \quad \text{for } m = 1, \ldots, N.$$

(Note that, since the number of rank-1 terms in this PD is not necessarily minimal, it is a priori not necessarily canonical. However, we will assume further

on that the m-th factor matrix in the PD of $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ is unique for $m = 1, \ldots, N$. As that implies that a decomposition in fewer terms is impossible, the PDs are CPDs by our assumption.) After solving Eq. (4.9) for $\mathbf{F}^{(m)}$ and estimating $\mathbf{U}^{(m)}$ for $m = 1, \ldots, N$ as described in Section 4.2.1, the tensorization $\boldsymbol{\mathcal{F}}^{(m)} \in \mathbb{R}^{P_{m1} \times \cdots \times I_m \times \cdots \times P_{mN}}$ of $\mathbf{F}^{(m)}$ is orthogonally compressed

$$\tilde{\boldsymbol{\mathcal{B}}}^{(m)} = \boldsymbol{\mathcal{F}}^{(m)} \cdot_m \mathbf{U}^{(m)^{\mathrm{T}}} = \left[\!\!\left[\mathbf{A}^{(m,1)}\mathbf{U}^{(1)}\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(m)}, \ldots, \mathbf{A}^{(m,N)}\mathbf{U}^{(N)}\mathbf{V}^{(N)} \right]\!\!\right] \tag{4.15}$$

for $m = 1, \ldots, N$. The CPD of the compressed tensor $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ shares the factor matrix $\mathbf{V}^{(m)}$ with the CPD of the core tensor $\boldsymbol{\mathcal{S}}$. The factor matrices $\mathbf{V}^{(n)}$ for $n = 1, \ldots, N$ of the CPD of the core can thus be found by computing the CPD of each compressed tensor if the mth factor matrix of the CPD of $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ is unique for $m = 1, \ldots, N$. If the CPD of the full tensor $\boldsymbol{\mathcal{X}}$ is also unique, its factor matrices can be retrieved by expanding the factor matrices $\mathbf{W}^{(n)} = \mathbf{U}^{(n)}\mathbf{V}^{(n)}$ for $n = 1, \ldots, N$. Since a CPD can only be unique up to the factor scaling and permutation indeterminacies and each factor matrix $\mathbf{V}^{(n)}$ for $n = 1, \ldots, N$ is retrieved from a different compressed tensor, the indeterminacies must be addressed. To this end, the fact that the N tensors $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ all have the same CPD, up to the (known) compression matrices $\mathbf{A}^{(m,n)}$ for $m, n = 1, \ldots, N$, can be exploited. In Algorithm 4.2, this is done by scaling and permuting the factor matrices of $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ for $m = 2, \ldots, N$ such that they match the factor matrices of $\tilde{\boldsymbol{\mathcal{B}}}^{(1)}$.

4.3.2 Algorithm

The steps in Section 4.3.1 mimic the popular approach in [63] for computing the CPD of a fully given tensor, which is: 1) compute the MLSVD of $\boldsymbol{\mathcal{X}}$, 2) compute the CPD of the core $\boldsymbol{\mathcal{S}}$ and 3) expand the factor matrices $\mathbf{W}^{(n)} = \mathbf{U}^{(n)}\mathbf{V}^{(n)}$ for $n = 1, \ldots, N$. The corresponding steps in the approach in this chapter are: 1) compute the MLSVD factor matrices $\mathbf{U}^{(n)}$ for $n = 1, \ldots, N$ and orthogonally compress the tensors $\boldsymbol{\mathcal{F}}^{(m)}$ for $m = 1, \ldots, N$, 2) compute the CPD of the compressed tensors and 3) expand the factor matrices $\mathbf{W}^{(n)} = \mathbf{U}^{(n)}\mathbf{V}^{(n)}$ for $n = 1, \ldots, N$. Instead of computing the CPD of a core tensor of dimensions $R \times \cdots \times R$, this approach computes the CPD of the N tensors in Eq. (4.15) which are of dimensions $P_{m1} \times \cdots \times P_{m,m-1} \times R \times P_{m,m+1} \times \cdots \times P_{m,N}$ for $m = 1, \ldots, N$. As will be shown in Section 4.3.3, these tensors can be far smaller than the core tensor. Algorithm 4.2 outlines all steps needed to compute a CPD with orthogonal compression from a BRKS linear system. All steps of this algorithm can be computed in parallel. Like Algorithm 4.1, Algorithm 4.2 accommodates both the RTD- and CS-setting.

Instead of computing the CPDs of the tensors $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ for $m = 1, \ldots, N$ separately, they can also be computed simultaneously as a set of coupled CPDs. These CPDs are coupled since their factor matrices all depend linearly, with coefficients $\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}$, on the same factors $\mathbf{V}^{(n)}$ for $m, n = 1, \ldots, N$ with $n \neq m$. This set of coupled CPDs can be computed in Tensorlab [129] through structured data fusion [130].

Algorithm 4.2: Orthogonally compressed CPD from a BRKS linear system (`lscpd_brks`).

Input: (CS-setting) $\mathbf{A}^{(m,n)}$, $\mathbf{b}^{(m)}$ for $m, n = 1, \ldots, N$
Input: (RTD-setting) $\mathbf{A}^{(m,n)}$ for $m, n = 1, \ldots, N$ and $\boldsymbol{\mathcal{X}}$
Output: $\mathbf{W}^{(m)}$ for $m = 1, \ldots, N$
begin
  if *RTD-setting* **then**
    Compute $\mathbf{b}^{(m)}$ for $m = 1, \ldots, N$ using Eq. (4.3)
  for $m = 1$ **to** N **do**
    Solve $\mathbf{A}^{(m,m)}\mathbf{F}^{(m)} = \mathbf{B}^{(m)}_{[m]}$ for $\mathbf{F}^{(m)}$
    Find $\mathbf{U}^{(m)}$ by computing an orthonormal basis that spans the dominant column space of $\mathbf{F}^{(m)}$
    Orthogonally compress $\tilde{\boldsymbol{\mathcal{B}}}^{(m)} = \boldsymbol{\mathcal{F}}^{(m)} \cdot_m \mathbf{U}^{(m)^{\mathrm{T}}}$
    Compute the CPD of $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ and set $\mathbf{V}^{(m)}$ equal to the mth factor matrix
    Scale and permute $\mathbf{V}^{(m)} \leftarrow \mathbf{D}^{(m)^{-1}}\mathbf{P}^{(m)^{\mathrm{T}}}\mathbf{V}^{(m)}$ to fix CPD indeterminacies
    Expand $\mathbf{W}^{(m)} = \mathbf{U}^{(m)}\mathbf{V}^{(m)}$

4.3.3 Conditions for CPD retrieval

In this section, we derive conditions for the identifiability of the CPD of $\boldsymbol{\mathcal{X}}$ from a BRKS system. The CPD can be identified if the conditions in Theorem 4.3 hold. In [131], conditions are provided to guarantee that one factor matrix of the CPD is unique.

Theorem 4.3. *Consider a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ of rank R, admitting a CPD $[\![\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(N)}]\!]$ with factor matrices $\mathbf{W}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$. Given linear combinations $\mathbf{b}$ of $\mathrm{vec}(\boldsymbol{\mathcal{X}})$, obtained from a BRKS linear system with generating matrices $\mathbf{A}^{(m,n)} \in \mathbb{R}^{P_{mn} \times I_n}$ for $m, n = 1, \ldots, N$, the factor*

matrices $\mathbf{W}^{(n)}$ *for* $n = 1, \ldots, N$ *can be retrieved if*

1) $\mathrm{r}\left(\mathbf{A}^{(m,m)}\right) = I_m \quad \text{for } m = 1, \ldots, N;$

2) $\prod_{n \neq m}^{N} \mathrm{r}\left(\mathbf{A}^{(m,n)}\mathbf{U}^{(n)}\right) \geq R \quad \text{for } m = 1, \ldots, N;$

3) The mth factor matrix of $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ is unique for $m = 1, \ldots, N;$

4) The CPD of $\boldsymbol{\mathcal{X}}$ is unique.

Proof. Conditions 1) and 2): For the compression matrices $\mathbf{U}^{(n)}$ for $n = 1, \ldots, N$, the factor matrices of the MLSVD of $\boldsymbol{\mathcal{X}}$ must be retrievable. These conditions are the same as the first two conditions in Theorem 4.1.
Condition 3): The CPD of the compressed tensor $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ for $m = 1, \ldots, N$ in Eq. (4.15) shares the mth factor matrix with the core of the MLSVD of $\boldsymbol{\mathcal{X}}$ if the mth factor matrix is unique for $m = 1, \ldots, N$.
Condition 4): While condition 3) ensures that the factor matrices $\mathbf{W}^{(n)}$ for $n = 1, \ldots, N$ are unique, condition 4) is needed to ensure that there is only one set of rank-1 tensors, consisting of the columns of $\mathbf{W}^{(n)}$ for $n = 1, \ldots, N$, that forms a CPD of $\boldsymbol{\mathcal{X}}$, i.e. to exclude different ways of pairing. □

In the generic case, in which the generating matrices and the factor matrices of the CPD are sampled from a continuous probability distribution, the CPD of $\boldsymbol{\mathcal{X}}$ can be identified if the conditions in Theorem 4.4 hold. Here we used a generic condition to prove the uniqueness of the full CPD of $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ for $m = 1, \ldots, N$, which a fortiori guarantees the uniqueness of its mth factor matrix. The conditions in Theorem 4.4 can be used to determine the dimensions of the generating matrices that are generically required for the identifiability of the CPD of $\boldsymbol{\mathcal{X}}$.

Theorem 4.4. *With generic generating matrices* $\mathbf{A}^{(m,n)}$ *for* $m, n = 1, \ldots, N$, *the conditions in Theorem 4.3 hold if*

1) $P_{mm} \geq I_m \quad \text{for } m = 1, \ldots, N;$

2) $\prod_{n \neq m}^{N} \min(P_{mn}, R) \geq R \quad \text{for } m = 1, \ldots, N;$

3) $I_m \geq R \quad \text{for } m = 1, \ldots, N;$

4) $\left\{ \exists \alpha \subset \{\{1, \ldots, N\} \setminus m\} : \min(J_2, J_3) \geq 3 \text{ and } (J_2 - 1)(J_3 - 1) \geq R \right\}$

$$\text{for } m = 1, \ldots, N \text{ with } J_2 = \prod_{n \in \alpha} P_{mn} \text{ and } J_3 = \prod_{n \in \alpha^c} P_{mn}$$

Proof. Conditions 1) and 2): These conditions are the same as the first two conditions in Theorem 4.2.
Condition 3) and 4): The mth factor matrix of $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ is unique for $m = 1, \ldots, N$ if the CPD of these tensors is unique. The Nth order tensor $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ can be reshaped to a third-order tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$, with $J_1 = I_m$, $J_2 = \prod_{n \in \alpha} P_{mn}$, $J_3 = \prod_{n \in \alpha^c} P_{mn}$ and $\alpha \subset \{[1, \ldots, N] \setminus m\}$, of which the first mode corresponds to the uncompressed mth mode of $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$. The second and third mode respectively correspond to a subset α of the remaining $N - 1$ modes and its complementary subset α^{c}. The rank R CPD of $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ is unique if there exists a subset α such that the rank R CPD of the reshaped third-order tensor $\boldsymbol{\mathcal{Y}}$ is unique. Generically, a third-order tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ of rank R is unique if $J_1 \geq R$, $\min(J_2, J_3) \geq 3$ and $(J_2 - 1)(J_3 - 1) \geq R$ [132]. Condition 3) guarantees that the former condition is satisfied and condition 4) guarantees that the latter two conditions are satisfied for $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ for $m = 1, \ldots, N$. Additionally, condition 3) guarantees generic uniqueness of the CPD of $\boldsymbol{\mathcal{X}}$ since for each reshaped, third-order version of $\boldsymbol{\mathcal{X}}$, all dimensions exceed R [4, Theorem 3]. □

As explained in Section 4.2.4, the second condition in Theorem 4.4 implies that $P_{mn} \geq \sqrt[N-1]{R}$ if the values P_{mn} are approximately equally large for $m, n = 1, \ldots, N$ and $n \neq m$. Similarly, the fourth condition in Theorem 4.4 implies that

$$P_{mn} \geq \sqrt[N-1]{\left(1 + \sqrt{R}\right)^2} \quad \text{for } m, n = 1, \ldots, N \text{ and } n \neq m.$$

(Note that this bound is derived for tensors of uneven order N. The bound for tensors of even order is similar, but does not have a simple expression.)

The latter constraint poses only slightly more restrictive bounds than the former, meaning that the dimensions P_{mn} for $m, n = 1, \ldots, N$ and $n \neq m$ can still be chosen such that they are much smaller than R. In real applications, tensor dimensions often exceed the tensor rank, satisfying the third condition in Theorem 4.4.

Remark: Note that the rank of a tensor can exceed some of its dimensions, in which case Theorem 4.4 cannot be satisfied. This can be resolved by using a different uniqueness condition to guarantee the uniqueness of the CPDs of $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ for $m = 1, \ldots, N$, such as the generic version of Kruskal's condition [71]. However, using this condition also results in stricter bounds on P_{mn} for $m, n = 1, \ldots, N$ and $n \neq m$.

4.4 Computing a TT from a BRKS linear system

Since the TT also does not suffer from the curse of dimensionality, we derive an algorithm for computing a TT from a BRKS linear system in this section. The TT-SVD algorithm in [95] computes the cores using sequential SVDs. Unlike in TT-SVD, for a BRKS linear system the SVDs can be computed in parallel by processing the compressed tensors $\boldsymbol{\mathcal{B}}^{(m)}$ for $m = 1, \ldots, N$ of $\boldsymbol{\mathcal{X}}$ separately.

4.4.1 Computing the TT cores

If $\boldsymbol{\mathcal{X}}$ is approximately of TT-rank $(R_0, \ldots, R_N)$, it admits a TT $(\!|\boldsymbol{\mathcal{G}}^{(1)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)}|\!)$ with cores $\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ for $n = 1, \ldots, N$. Substituting this TT for $\boldsymbol{\mathcal{X}}$ into the BRKS linear system in Eq. (4.3) leads to

$$\mathbf{A}\text{vec}\left((\!|\boldsymbol{\mathcal{G}}^{(1)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)}|\!)\right) = \mathbf{b}.$$

It follows that the tensorized mth block row of this BRKS system corresponds to the TT of $\boldsymbol{\mathcal{X}}$ transformed through mode-n multiplication with the generating matrices $\mathbf{A}^{(m,n)}$:

$$\boldsymbol{\mathcal{B}}^{(m)} = (\!|\boldsymbol{\mathcal{H}}^{(m,1)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,N)}|\!)$$

with $\boldsymbol{\mathcal{H}}^{(m,n)} = \boldsymbol{\mathcal{G}}^{(n)} \cdot_2 \mathbf{A}^{(m,n)}$ for $n = 1, \ldots, N$. Rearranging this transformed TT into its mode-m unfolding leads to

$$\mathbf{A}^{(m,m)}\left((\!|\boldsymbol{\mathcal{H}}^{(m,1)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,m-1)}, \boldsymbol{\mathcal{G}}^{(m)}, \boldsymbol{\mathcal{H}}^{(m,m+1)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,N)}|\!)\right)_{[m]} = \mathbf{B}^{(m)}_{[m]}.$$

This unfolding corresponds to a linear system

$$\mathbf{A}^{(m,m)}\tilde{\mathbf{B}}^{(m)}_{[m]} = \mathbf{B}^{(m)}_{[m]} \tag{4.16}$$

that can be solved for $\tilde{\mathbf{B}}^{(m)}_{[m]}$. The tensor $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ is a transformed TT of $\boldsymbol{\mathcal{X}}$ that shares the mth core $\boldsymbol{\mathcal{G}}^{(m)}$ with the TT of the full tensor $\boldsymbol{\mathcal{X}}$. Following the definition of a TT, this core can be retrieved from the column space of the following matrix unfolding

$$\tilde{\mathbf{B}}^{(m)}_{[1,\ldots,m;m+1,\ldots,N]} = \left(\langle\!\langle \boldsymbol{\mathcal{H}}^{(m,1)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,m-1)}, \boldsymbol{\mathcal{G}}^{(m)} \rangle\!\rangle\right)^{\mathrm{T}}_{[m+1]} \left(\langle\!\langle \boldsymbol{\mathcal{H}}^{(m,m+1)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,N)} \rangle\!\rangle\right)_{[1]}. \tag{4.17}$$

(W.r.t. the matricization, note that while $\langle\!\langle \boldsymbol{\mathcal{H}}^{(m,1)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,m-1)}, \boldsymbol{\mathcal{G}}^{(m)} \rangle\!\rangle$ consists of m cores, it is a tensor of order $m+1$ since R_m is not necessarily equal to one.) First, we retrieve $\boldsymbol{\mathcal{G}}^{(1)}$ by computing an orthonormal basis of dimension R_1 for the dominant column space of the matrix unfolding in Eq. (4.17) for $m = 1$. For $m = 2, \ldots, N$, the column space of this unfolding also involves the preceding cores $\boldsymbol{\mathcal{H}}^{(m,1)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,m-1)}$. If the cores are computed in order, then these preceding cores are known and can be compensated for by subsequently solving smaller linear systems

$$\mathbf{H}^{(m,n)}_{[1,2;3]} \mathbf{B}^{(n)}_{\mathrm{new}} = \mathbf{B}^{(n)} \quad \text{for } n = 1, \ldots, m-1, \tag{4.18}$$

for $\mathbf{B}^{(n)}_{\mathrm{new}}$, with $\mathbf{B}^{(1)} = \tilde{\mathbf{B}}^{(m)}_{[1]}$ and

$$\mathbf{B}^{(n)} = \mathrm{reshape}\left(\mathbf{B}^{(n-1)}_{\mathrm{new}}, [R_{n-1}P_{mn}, I_m \prod_{k=n+1}^{N} P_{mk}]\right) \text{ for } n = 2, \ldots, m-1.$$

After solving these systems, $\boldsymbol{\mathcal{G}}^{(m)}$ is retrieved by computing an orthonormal basis of dimension R_m for the dominant column space of

$$\mathrm{reshape}\left(\mathbf{B}^{(n)}_{\mathrm{new}}, [R_{m-1}I_m, \prod_{k=m+1}^{N} P_{mk}]\right) \text{for } m = 1, \ldots, N.$$

Alternatively, we can immediately compute an orthonormal basis of dimension R_m for the dominant column space of the matrix unfolding in Eq. (4.17) for $m = 1, \ldots, N$, without compensating for preceding cores first. Computing these bases is more expensive, since they are obtained from a larger matrix than in the case where the preceding cores have already been compensated for. On the other hand, the orthonormal bases can be computed in parallel, as there is no dependency on any preceding core. If after the orthonormal bases have been found, also the cores of the TT are desired, compensation of preceding cores can be done in a similar manner as described above, namely by subsequently solving linear systems. These linear systems have the same coefficient matrix as the linear systems in Eq. (4.18).

4.4.2 Algorithm

Algorithm 4.3 outlines all steps needed to compute a TT from a BRKS linear system for the approach in which the orthonormal bases are computed first and the preceding cores are compensated for second. Note that only $N-1$ SVDs need to be computed, just like in the standard TT-SVD algorithm, in which the final SVD reveals both cores $N-1$ and N. Like Algorithm 4.1 and Algorithm 4.2, Algorithm 4.3 also accommodates both the RTD- and CS-setting.

Remark: When computing the mth core in Algorithm 4.3, the $m-1$ preceding compressed cores $\boldsymbol{\mathcal{H}}^{(m,1)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,m-1)}$ in $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ are compensated for from left to right, i.e. for $m = 1, \ldots, n-1$. If $\tilde{\boldsymbol{\mathcal{B}}}^{(m)}$ is unfolded in reverse, i.e.

$$\tilde{\mathbf{B}}^{(m)}_{[N,\ldots,m;m-1,\ldots,1]} = \left(\left(\boldsymbol{\mathcal{H}}^{(m,N)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,m+1)}, \boldsymbol{\mathcal{G}}^{(m)} \right) \right)^{\mathrm{T}}_{[N-m+2]} \left(\left(\boldsymbol{\mathcal{H}}^{(m,m-1)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,1)} \right) \right)_{[1]}, \tag{4.19}$$

then the column space of the unfolding contains, besides $\boldsymbol{\mathcal{G}}^{(m)}$, the $N-m$ next compressed cores $\boldsymbol{\mathcal{H}}^{(m,N)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,m+1)}$. This way, it is possible to compute the mth core by compensating for these next compressed cores from right to left, i.e. $\boldsymbol{\mathcal{H}}^{(m,N)}, \ldots, \boldsymbol{\mathcal{H}}^{(m,m+1)}$. The efficiency of Algorithm 4.3 can be improved by computing the first half of the cores using the unfolding in Eq. (4.17) and compensating for the preceding cores from left to right, and the second half of the cores using the unfolding in Eq. (4.19) and compensating for the next cores from right to left. This halves the number of cores that need to be compensated for compared to Algorithm 4.3, in which cores are only compensated for from left to right to simplify the pseudocode.

4.4.3 Conditions for TT retrieval

In this section, we derive the conditions in Theorem 4.5, under which retrieval of the TT of $\boldsymbol{\mathcal{X}}$ from a BRKS system is guaranteed. The compensation for preceding cores occurs in both approaches for computing the TT in Section 4.4.1. Therefore, the linear systems that are solved to compensate for them must have a unique solution. Since these linear systems have the same coefficient matrices in both approaches, the condition under which they have a unique solution is the same, regardless of the chosen approach.

Theorem 4.5. *Consider a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ of TT-rank $(R_0, \ldots, R_N)$, admitting a TT $(\!|\boldsymbol{\mathcal{G}}^{(1)}, \ldots, \boldsymbol{\mathcal{G}}^{(N)}|\!)$ with cores $\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ for $n =$*

Algorithm 4.3: TT from a BRKS linear system (`lstt_brks`).

Input: (CS-setting) $\mathbf{A}^{(m,n)}$, $\mathbf{b}^{(m)}$ for $m, n = 1, \ldots, N$
Input: (RTD-setting) $\mathbf{A}^{(m,n)}$ for $m, n = 1, \ldots, N$ and $\boldsymbol{\mathcal{X}}$
Output: $\boldsymbol{\mathcal{G}}^{(m)}$ for $m = 1, \ldots, N$
begin
  if *RTD-setting* **then**
    Compute $\mathbf{b}^{(m)}$ for $m = 1, \ldots, N$ using Eq. (4.3)
  for $m = 1$ **to** N **do**
    Solve $\mathbf{A}^{(m,m)}\tilde{\mathbf{B}}^{(m)}_{[m]} = \mathbf{B}^{(m)}_{[m]}$ for $\tilde{\mathbf{B}}^{(m)}_{[m]}$
    Estimate $\tilde{\mathbf{G}}^{(m)}$ as an orthonormal basis for the dominant column space of $\tilde{\mathbf{B}}^{(m)}_{[1,\ldots,m;m+1,\ldots,N]}$
    for $n = 1$ **to** $m - 1$ **do**
      Reshape $\tilde{\mathbf{G}}^{(m)}$ to dimensions $R_{n-1}P_{mn} \times P_{m,n+1} \cdots P_{m,m-1} I_m R_m$
      Solve the system $\mathbf{H}^{(m,n)}_{[1,2;3]}\mathbf{G}_{\text{new}} = \tilde{\mathbf{G}}^{(m)}$ for $\mathbf{G}_{\text{new}}$
      Set $\tilde{\mathbf{G}}^{(m)} = \mathbf{G}_{\text{new}}$
    $\boldsymbol{\mathcal{G}}^{(m)} = \text{reshape}\left(\tilde{\mathbf{G}}^{(m)}, [R_{m-1}, I_m, R_m]\right)$

$1, \ldots, N$. *Given linear combinations* $\mathbf{b}$ *of vec*$(\boldsymbol{\mathcal{X}})$*, obtained from a BRKS linear system with generating matrices* $\mathbf{A}^{(m,n)} \in \mathbb{R}^{P_{mn} \times I_n}$ *for* $m, n = 1, \ldots, N$*, the cores* $\boldsymbol{\mathcal{G}}^{(n)}$ *for* $n = 1, \ldots, N$ *can be retrieved if and only if*

1) $\text{r}\left(\mathbf{A}^{(m,m)}\right) = I_m \quad \text{for } m = 1, \ldots, N;$

2) $\text{r}\left(\tilde{\mathbf{B}}^{(m)}_{[1,\ldots,m;m+1,\ldots,N]}\right) = R_m \quad \text{for } m = 1, \ldots, N;$

3) $\text{r}\left(\mathbf{H}^{(m,n)}_{[1,2;3]}\right) = R_n \quad \text{for } m = 1, \ldots, N \text{ and } n = 1, \ldots, m-1.$

Proof. Condition 1): The linear system in Eq. (4.16) can be uniquely solved for $\tilde{\mathbf{B}}^{(m)}_{[m]}$ if and only if $\mathbf{A}^{(m,m)}$ is of full column rank I_m for $m = 1, \ldots, N$. Condition 2): An orthonormal basis of dimension R_m for the column space of the matricization $\tilde{\mathbf{B}}^{(m)}_{[1,\ldots,m;m+1,\ldots,N]}$ in Eq. (4.17) can be retrieved if and only if the rank of this matricization equals R_m. Condition 3): The linear systems that are solved to compensate for compressed cores have a unique solution if and only if

$$\text{r}\left(\mathbf{H}^{(m,n)}_{[1,2;3]}\right) = R_n \quad \text{for } m = 1, \ldots, N \text{ and } n = 1, \ldots, m-1$$

holds. □

In the generic case, in which the generating matrices are sampled from a continuous probability distribution, the TT of $\boldsymbol{\mathcal{X}}$ can be retrieved if the conditions in Theorem 4.6 hold. These conditions can be used to determine the dimensions of the generating matrices such that the TT of $\boldsymbol{\mathcal{X}}$ can generically be retrieved.

Theorem 4.6. *With generic generating matrices* $\mathbf{A}^{(m,n)}$ *for* $m, n = 1, \ldots, N$, *the conditions in Theorem 4.1 hold if and only if*

1) $P_{mm} \geq I_m \quad \text{for } m = 1, \ldots, N;$

2) $P_{mn} \geq \dfrac{R_m}{R_{n-1} I_m \prod_{k=n+1}^{m-1} P_{mk}} \quad \text{for } m = 1, \ldots, N \text{ and } n = 1, \ldots, n-1;$

3) $P_{mn} \geq \dfrac{R_m}{R_n \prod_{k=m+1}^{n-1} P_{mk}} \quad \text{for } m = 1, \ldots, N \text{ and } n = m+1, \ldots, N;$

4) $P_{mn} \geq \dfrac{R_n}{R_{n-1}} \quad \text{for } m = 1, \ldots, N \text{ and } n = 1, \ldots, m-1.$

Proof. Condition 1): This condition is the same as condition 1) in Theorem 4.2. Condition 2): The matricization $\tilde{\mathbf{B}}^{(m)}_{[1,\ldots,m;m+1,\ldots,N]}$ in Eq. (4.17) equals

$$\tilde{\mathbf{B}}^{(m)}_{[1,\ldots,m;m+1,\ldots,N]} = \tag{4.20}$$

$$\left(\mathbf{G}^{(m)}_{[3]}\left(\mathbf{H}^{(m,m-1)}_{[3]}\left(\cdots \mathbf{H}^{(m,2)}_{[3]}\left(\mathbf{H}^{(m,1)}_{[3]} \otimes \mathbf{I}_{P_{m2}}\right)\cdots \otimes \mathbf{I}_{P_{m,m-1}}\right) \otimes \mathbf{I}_{I_m}\right)\right)^{\mathrm{T}}$$

$$\left(\mathbf{H}^{(m,m+1)}_{[1]}\left(\mathbf{I}_{P_{m,m+1}} \otimes \cdots \mathbf{H}^{(m,N-1)}_{[1]}\left(\mathbf{I}_{P_{m,N-1}} \otimes \mathbf{H}^{(m,N)}_{[1]}\right)\right)\right) =: \mathbf{D}^{\mathrm{T}}\mathbf{E}.$$

Generically, the rank of $\tilde{\mathbf{B}}^{(m)}_{[1,\ldots,m;m+1,\ldots,N]}$ equals R_m if and only if the rank of both $\mathbf{D}$ and $\mathbf{E}$ equals R_m. Condition 2) relates to $\mathbf{D}$ and condition 3) to $\mathbf{E}$. Since $\mathbf{G}^{(m)}_{[3]}$ is of full rank, which follows from the definition of a TT, and $\mathbf{H}^{(m,n)}_{[3]}$ for $n \neq m$ is generically of full rank, each matrix product in Eq. (4.20)

is also of full rank [43, Lemma 1]. Therefore, the rank of $\mathbf{D}$ generically equals

$$\min\left(\mathrm{r}\left(\mathbf{G}^{(m)}_{[3]}\right), \min\left(\mathrm{r}\left(\mathbf{H}^{(m,m-1)}_{[3]}\right), \ldots \right.\right.$$

$$\left.\left.\min\left(\mathrm{r}\big(\mathbf{H}^{(m,2)}_{[3]}\big), \mathrm{r}\big(\mathbf{H}^{(m,1)}_{[3]}\big)P_{m2}\right)P_{m3}\cdots P_{m,m-1}\right)I_m\right).$$

Both arguments of the leftmost $\min(\cdot)$ must at least equal R_m such that $\mathrm{r}(\mathbf{D}) \geq R_m$ holds. Following the definition of a TT, this always holds true for the first argument $\mathrm{r}\left(\mathbf{G}^{(m)}_{[3]}\right)$. The second argument is $\min(\cdot)I_m$, so both arguments of this second $\min(\cdot)$ must at least equal $\frac{R_m}{I_m}$. Generically $\mathrm{r}\left(\mathbf{H}^{(m,m-1)}_{[3]}\right) = \min(P_{m,m-1}R_{m-2}, R_{m-1})$, leading to the conditions $P_{m,m-1} \geq \frac{R_m}{R_{m-2}I_m}$ and $R_{m-1}I_m \geq R_m$. The latter condition is satisfied by the definition of a TT. Repeating the same steps for each subsequent $\min(\cdot)$ leads to the conditions

$$P_{mn} \geq \frac{R_m}{R_{n-1}I_m \prod_{k=n+1}^{m-1} P_{mk}} \quad \text{for } m = 1, \ldots, N \text{ and } n = 1, \ldots, m-1.$$

Condition 3): In a similar fashion, it can be proven that $\mathrm{r}(\mathbf{E}) \geq R_m$ holds if and only if

$$P_{mn} \geq \frac{R_m}{R_n \prod_{k=m+1}^{n-1} P_{mk}} \quad \text{for } m = 1, \ldots, N \text{ and } n = m+1, \ldots, N.$$

Condition 4): Since a generic matrix is of full rank, $\mathrm{r}\left(\mathbf{H}^{(m,n)}_{[1,2;3]}\right) = \min(R_{n-1}P_{mn}, R_n)$ for $m, n = 1, \ldots, N$. □

4.5 Experiments

In this section, we validate our algorithms using synthetic and real data. The algorithms are implemented in MATLAB and are available at `https://www.tensorlabplus.net`. In the practical implementation of the algorithms, column spaces are estimated using the SVD and linear systems are solved using the MATLAB backslash operator. All experiments are performed on a laptop with an AMD Ryzen 7 PRO 3700U processor and 32GB RAM. The algorithms are run sequentially even though each algorithm can (partly) be executed in parallel.

Since it is possible to choose the generating matrices in the experiments in this section, we set $\mathbf{A}_{mm} = \mathbf{I}_{I_m}$ for $m = 1, \ldots, N$. This means that the

first step in each algorithm, namely solving a linear system with $\mathbf{A}_{mm}$ as the coefficient matrix, can be skipped. First, we use synthetic data to compare the accuracy and computation time of these algorithms to related algorithms. All synthetic problems are constructed by sampling the entries of the factor matrices and/or core(s) of a tensor decomposition from the standard normal distribution. Additive Gaussian noise $\boldsymbol{\mathcal{N}}$ is added to these randomly generated tensors $\boldsymbol{\mathcal{X}}$ and the noise level is quantified using the signal-to-noise ratio (SNR):

$$\text{SNR} = 10 \log_{10} \left(\frac{||\boldsymbol{\mathcal{X}}||_{\text{F}}^2}{||\boldsymbol{\mathcal{N}}||_{\text{F}}^2} \right).$$

4.5.1 Randomized MLSVD

In the first experiment, a random third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times I \times I}$ of low multilinear rank (R, R, R), with $I = 200$ and $R = 10$, is generated with varying levels of noise. The factor matrices of the MLSVD of $\boldsymbol{\mathcal{X}}$ are then retrieved from a BRKS linear system using Algorithm 4.1. The entries of the generating matrices are sampled from the standard normal distribution. This means that we are using Algorithm 4.1 in the RTD-setting in this experiment. The dimensions of the generating matrices are determined using the oversampling factor $q = 5$ and the multilinear rank of $\boldsymbol{\mathcal{X}}$. For this value of q, the sampling ratio equals 0.005. This ratio is defined as the number of compressed measurements in $\mathbf{b}$ divided by the number of entries in $\boldsymbol{\mathcal{X}}$. We compare the accuracy of our algorithm with related RTD algorithms and a cross-approximation algorithm for low multilinear rank approximation. These algorithms are:

- `rand_tucker` [123, Algorithm 2]: The factor matrices $\mathbf{U}^{(n)}$ for $n = 1, \ldots, N$ are retrieved as follows: 1) the mode-n matrix unfolding of $\boldsymbol{\mathcal{X}}$ is compressed through multiplication with a random Gaussian matrix and 2) an orthonormal basis for this compressed matrix unfolding is computed using the QR decomposition. After computing each factor matrix, $\boldsymbol{\mathcal{X}}$ is orthogonally compressed using this factor matrix like in the sequentially truncated higher-order SVD algorithm [89]. The oversampling factor is set to $p = 5$, which means that this algorithm actually estimates an MLSVD of multilinear rank $(R + p, R + p, R + p)$.
- `rand_tucker_kron` [122, Algorithm 4.2]: Similar to `rand_tucker`, but the algorithm uses a Kronecker-structured matrix for compression and the SVD for computing an orthonormal basis. The oversampling factor for this algorithm is chosen such that it is the same as our oversampling factor q in Section 4.2.4.

- `mlsvd_rsi` [133]: Computes an MLSVD using sequential truncation [89] and uses randomized compression and subspace iteration for estimating the SVD [90]. In the randomized compression step, the mode-n unfolding $\mathbf{X}_{[n]}$ is compressed through matrix multiplication with a random matrix of dimensions $\prod_{i \neq n} I_i \times R_n + p$, with an oversampling factor $p = 5$. Next, the nth factor matrix is estimated by computing an SVD of this randomly compressed matrix and further refined using two subspace iteration steps with the full mode-n unfolding $\mathbf{X}_{[n]}$.
- `lmlra_aca` [17]: Cross approximation approach for low multilinear rank approximation.

The last two algorithms are available in Tensorlab [129]. As the full tensor $\boldsymbol{\mathcal{X}}$ is usually available in the RTD-setting, the core is computed as

$$\boldsymbol{\mathcal{S}} = \boldsymbol{\mathcal{X}} \cdot_1 \mathbf{U}^{(1)^{\mathrm{T}}} \cdots \cdot_N \mathbf{U}^{(N)^{\mathrm{T}}}.$$

Fig. 4.2 compares the accuracy, quantified as a relative error

$$E_{\mathrm{rel}} = \frac{||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{S}}} \cdot_1 \hat{\mathbf{U}}^{(1)} \cdots \cdot_N \hat{\mathbf{U}}^{(N)}||_{\mathrm{F}}}{||\boldsymbol{\mathcal{X}}||_{\mathrm{F}}}$$

with $\hat{\boldsymbol{\mathcal{S}}}$ and $\hat{\mathbf{U}}^{(n)}$ for $n = 1, \ldots, N$ the estimated core and factor matrices, of these algorithms. The error shown in Fig. 4.2 is the average relative error over 10 trials. Algorithm `lsmlsvd_brks` is more accurate than `rand_tucker` and `lmlra_aca`. Algorithm `mlsvd_rsi` is far more accurate than all other algorithms because it uses subspace iteration with the full mode-n matrix unfolding of $\boldsymbol{\mathcal{X}}$ for computing the nth factor matrix. For this reason it also has the longest computation time of all algorithms. Algorithm `rand_tucker_kron` is more accurate than `lsmlsvd_brks` due to the sequential truncation step in `rand_tucker_kron` after each factor matrix is computed. If this step is omitted, which corresponds to [122, Algorithm 4.1], it achieves the same accuracy as `lsmlsvd_brks`. This sequential truncation step is not possible in `lsmlsvd_brks` since this algorithm only uses the compressed measurements $\mathbf{b}$ instead of the full tensor $\boldsymbol{\mathcal{X}}$. Note that increasing the oversampling factors of the algorithms results in higher accuracy in exchange for a longer computation time and requiring more compressed datapoints.

4.5.2 Randomized CPD

In this experiment, we generate a CPD with random factor matrices of a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times I \times I}$, with $I = 100$, of rank $R = 10$. Varying levels of noise

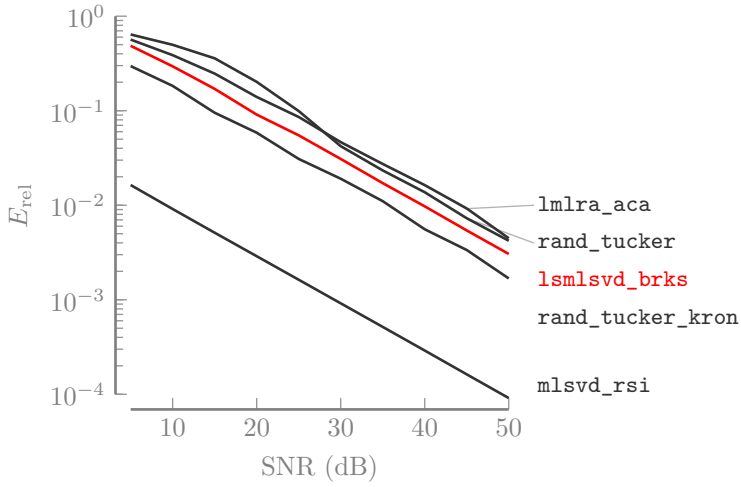


Figure 4.2: Algorithm `mlsvd_rsi` is by far the most accurate because it used the full-sized matrix unfolding in the subspace iteration step. Algorithm `lsmlsvd_brks` is more accurate than `lmlra_aca` and `rand_tucker` and less accurate than `rand_tucker_kron`. If the sequential truncation step in `rand_tucker_kron` is omitted, it achieves the same accuracy as `lsmlsvd_brks`.

are added to this tensor. Algorithm 4.2 is used in the RTD-setting to estimate the factor matrices of the CPD of $\boldsymbol{\mathcal{X}}$. To compute the CPDs of the compressed tensors, we use the `cpd` function in Tensorlab [129]. This function initializes the factor matrices with a generalized eigenvalue decomposition if possible and further improves them using (second-order) optimization algorithms. The accuracy of the factor matrices estimated by this algorithm are compared to results obtained with related RTD algorithms:

- `cpd_rbs` [15]: In each iteration, a random subtensor of $\boldsymbol{\mathcal{X}}$ is sampled and the corresponding rows of the factor matrices are updated. These updates are computed using a Gauss–Newton algorithm. In this experiment, the algorithm starts with random initial factor matrices.
- `cp_arls` [121]: Alternating least squares with random sketching for solving the least squares subproblems.

The accuracy of the estimated factor matrices is quantified as a relative error

$$E_{\text{CPD}} = \max_n \frac{||\hat{\mathbf{U}}^{(n)} - \mathbf{U}^{(n)}||_{\text{F}}}{||\mathbf{U}^{(n)}||_{\text{F}}},$$

in which the scaling and permutation ambiguities between the true $\mathbf{U}^{(n)}$ and estimated $\hat{\mathbf{U}}^{(n)}$ factor matrix have been resolved for $n = 1, \ldots, N$. Figure 4.3 shows the average relative error on the left and the average computation time for tensors with increasing dimensions on the right. Both averages are computed over 10 trials. For `lscpd_brks`, sampling the randomly compressed tensors, i.e. evaluating Eq. (4.3), is included in the computation time. The oversampling factor q of `lscpd_brks` is set to $5, 10$ or 50. For a larger value of q, $\mathbf{U}^{(n)}$ better captures the mode-n subspace of $\boldsymbol{\mathcal{X}}$ and less information is lost during the orthogonal compression step. Figure 4.3 illustrates that the algorithm is much more accurate for $q = 50$, while the increase in computation time compared to $q = 5$ is negligible. If q is set such that the size of the compressed tensors in Eq. (4.4) is of the same order as the full tensor $\boldsymbol{\mathcal{X}}$, then the computation time will of course increase significantly. For $q = 50$, the number of compressed datapoints in $\mathbf{b}$ equals just 15% of the total number of datapoints in $\boldsymbol{\mathcal{X}}$. This sampling ratio can be even lower for tensors with order greater than three. Algorithm `cp_arls` is slightly more accurate than `lscpd_brks` with $q = 50$ while being much slower in terms of computation time. Algorithm `cpd_rbs` is even more accurate and is situated in between `lscpd_brks` and `cp_arls` for smaller values of I. The computation time of `lscpd_brks`is dominated by sampling the randomly compressed tensors, since computing the CPD of these small tensors is very fast. Therefore, the computation time of `cpd_rbs` scales better for higher values of I since sampling random subtensors of $\boldsymbol{\mathcal{X}}$ is less time consuming than the random compression of $\boldsymbol{\mathcal{X}}$ in `lsmlsvd_brks`, which requires multiple large matrix products. In contrast to `lscpd_brks`, which uses a fixed amount of compressed datapoints determined by the size of the problem and the oversampling factor, `cpd_rbs` and `cp_arls` continue randomly sampling from $\boldsymbol{\mathcal{X}}$ every iteration until a stopping criterion is met.

4.5.3 Randomized TT

In this experiment, we generate a TT with random core tensors of a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times I \times I}$, with $I = 500$, of TT-rank $(1, 10, 10, 1)$. We add varying levels of noise to this tensor and estimate the cores using Algorithm 4.3 in the RTD-setting. For this experiment, we use the version of this algorithm that first computes orthonormal bases and then compensates for preceding cores. Additionally, preceding cores are compensated for from left to right for the first half of the cores and from right to left for the second half, as explained in Section 4.4.2. The dimensions of the generating matrices are chosen such that the sampling ratio equals 0.005.

The results of `lstt_brks` are compared with the results of a related RTD algorithm and a TT cross-approximation approach:

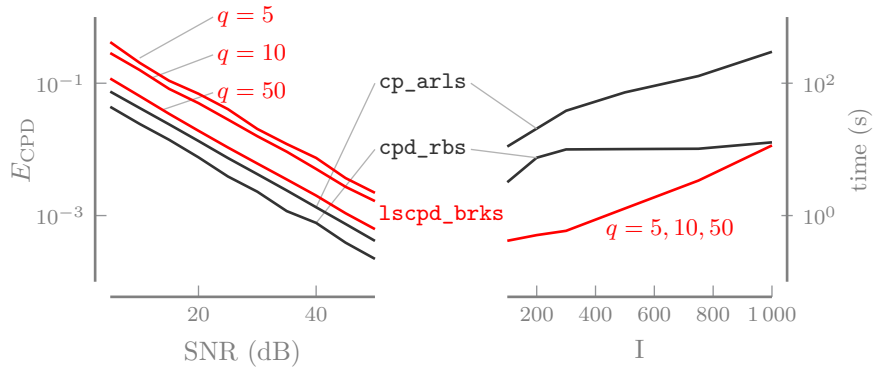


Figure 4.3: Increasing the oversampling factor q of `lscpd_brks`, for which the value is shown in the figure, greatly improves its accuracy while having a negligible effect on the computation time. Algorithm `lscpd_brks` is faster than `cp_arls` and `cp_rbs`, for a range of tensor dimensions that are relevant for applications, with a limited loss of accuracy.

- `rand_tt` [134, Algorithm 5.1]: This algorithm is a randomized version of the standard TT-SVD algorithm. The TT-SVD algorithm consists of three steps that are performed for the first $N-1$ cores: 1) tensor $\boldsymbol{\mathcal{X}}$ is matricized, 2) a core is estimated by computing a basis for the column space of this matricization using the SVD and 3) tensor $\boldsymbol{\mathcal{X}}$ is compressed using this basis. In `rand_tt`, the matricization in the first step is compressed by multiplying it with a random matrix from the right in order to speed up the computation of the SVD in the next step.
- `cross_tt` [135]: This algorithm reduces the size of the matricizations of $\boldsymbol{\mathcal{X}}$ using a maximal volume cross-approximation approach. The matricizations used in this algorithm allow the TT-ranks to be determined adaptively.

The accuracy of the results of these algorithms are quantified as a relative error

$$E_{\text{rel}} = \frac{||\boldsymbol{\mathcal{X}} - (\!|\hat{\boldsymbol{\mathcal{G}}}^{(1)}, \ldots, \hat{\boldsymbol{\mathcal{G}}}^{(N)}|\!)||_{\text{F}}}{||\boldsymbol{\mathcal{X}}||_{\text{F}}},$$

in which $\hat{\boldsymbol{\mathcal{G}}}^{(n)}$ for $n = 1, \ldots, N$ are the estimated cores. Figure 4.4 shows the relative error, averaged over 50 trials, and the computation time, averaged over 10 trials, for all algorithms. Tensors of increasing size are used to estimate the

computation time. The relative errors achieved by the algorithms with these tensors are in proportion with the errors shown in Fig. 4.4. The relative error is approximately the same for both `rand_tt` and `lstt_brks`. The former is faster thanks to the compression step that is performed after computing each core, which causes the tensor to get progressively smaller as the algorithm progresses. However, this algorithm requires the full tensor to be available, whereas the latter works with a limited amount of Kronecker-structured linear combinations of the entries of $\boldsymbol{\mathcal{X}}$. Therefore, `lstt_brks` enables us to achieve accuracy comparable to `rand_tt` in CS applications. If a method for obtaining any entry of $\boldsymbol{\mathcal{X}}$ is available, then `cross_tt` can be used to recover a more accurate approximation of the tensor, in exchange for a longer computation time. However, this algorithm is not applicable in the CS-setting either.

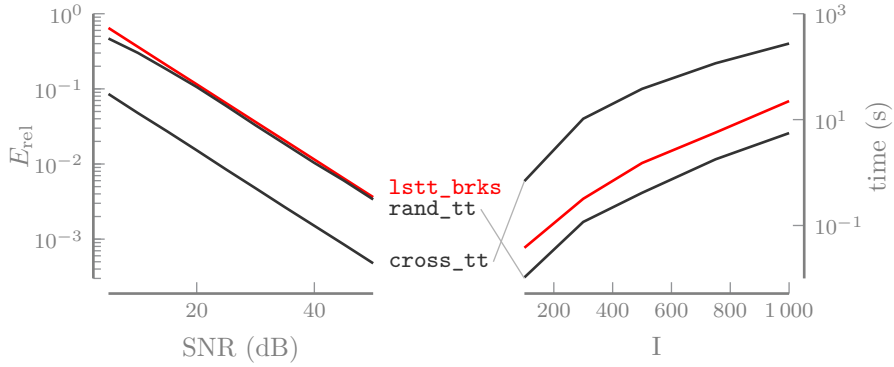


Figure 4.4: Algorithm `lstt_brks` enables us to achieve accuracy similar to that of `rand_tt` in CS applications. If a method for obtaining any entry of $\boldsymbol{\mathcal{X}}$ is available, then `cross_tt` can be used to recover a more accurate approximation of the tensor, in exchange for a longer computation time.

4.5.4 Compressed sensing hyperspectral imaging

In the RTD-setting, the tensor is fully acquired first and then randomly compressed. In the CS-setting, the data can in some applications be compressed during the data acquisition step. For example in hyperspectral imaging, compressed measurements can be obtained using the single-pixel camera [34] or the coded aperture snapshot spectral imager (CASSI) [136]. A hyperspectral image is a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, consisting of images of I_1 by I_2 pixels taken at I_3 different wavelengths. The single-pixel camera randomly

compresses the spatial dimensions of a hyperspectral image. In CASSI, a mask is applied to each I_1 by I_2 image for different wavelengths and these masked images are then aggregated into a single one, i.e. a snapshot. In general, compressed measurements are often obtained in (hyperspectral) imaging by compressing along each dimension separately, meaning that the measurement matrix is Kronecker-structured [39], [137]. With a small number of compressed measurements, a large hyperspectral image can often be accurately reconstructed. Whereas the random compression step dominated the computation time of our algorithms in the previous experiments, it is missing altogether in this experiment since it is inherent to the application. Therefore, our algorithms enable fast reconstruction of hyperspectral images.

We use `lsmlsvd_brks` to reconstruct a hyperspectral image using compressed measurements sampled at different sampling ratios. Similar to KCS, this approach expresses $\boldsymbol{\mathcal{X}}$ in a Kronecker-structured basis, namely the Kronecker product of $\mathbf{U}^{(n)}$ for $n = 1, 2, 3$, in which the core tensor $\boldsymbol{\mathcal{S}}$ contains the coefficients. In KCS, the bases are chosen a priori such that these coefficients are sparse, which is not necessarily true for the core of an MLSVD. Also, in `lsmlsvd_brks`, the bases do not have to be chosen a priori as this algorithm also estimates them using compressed measurements. Additionally, in KCS the measurement matrix is Kronecker-structured, whereas it consists of multiple Kronecker-structured block rows in our approach.

Instead of using just the final, $(N + 1)$th block row of the BRKS system to estimate the core, it makes more sense to use all available compressed measurements to improve the quality of the estimation, since the number of measurements is limited. Therefore we solve the full BRKS system, with all $N + 1$ block rows, for a sparse $\text{vec}\,(\boldsymbol{\mathcal{S}})$ by optimizing:

$$\min_{\text{vec}(\boldsymbol{\mathcal{S}})} ||\text{vec}\,(\boldsymbol{\mathcal{S}})\,||_1 \quad \text{subject to } ||\mathbf{A}\text{vec}\,(\boldsymbol{\mathcal{S}}) - \mathbf{b}||_2 \leq \sigma.$$

This is a basis pursuit denoising problem, which can be solved using the SPGL1 solver [138], [139]. The reason for imposing sparsity is as follows. If we were to estimate a dense core tensor using `lsmlsvd_brks` in this experiment, the number of compressed measurements $\mathbf{b}^{(N+1)}$ certainly cannot be less than the number of entries in the core if we want to be able to retrieve it. However, it turns out that the mode-n vectors of the hyperspectral imaging data $\boldsymbol{\mathcal{X}}$ cannot be well approximated as linear combinations of a small number of multilinear singular vectors, indicating that the multilinear rank is not small. Estimating a dense core tensor would then require a large number of measurements. On the other hand, it also turns out that in this data only a relatively small number of the entries in a relatively large core is important. We exploit this by computing a sparse approximation of the core, which results in reconstructions of better

quality than with the dense core approach, while requiring far less compressed measurements.

The reconstruction results of our algorithm are compared with two other approaches for reconstructing a hyperspectral image from compressed measurements:

- `GAP_TV` [140]: A generalized alternating projection algorithm that solves the total variation minimization problem. Minimizing total variation leads to accurate image reconstruction because images are generally locally self-similar.
- `KCS` [39]: We used a two-dimensional Daubechies wavelet basis to sparsify the spatial dimensions and the Fourier basis for the spectral dimension. The sparse coefficients are computed using the SPGL1 solver.

The quality of the reconstructed images is quantified using the peak signal-to-noise ratio (PSNR)

$$\text{PSNR} = \log_{10}\left(\frac{\max(\boldsymbol{\mathcal{X}})}{\sqrt{\frac{1}{I_1 I_2 I_3}||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||_{\text{F}}}}\right),$$

in which $\hat{\boldsymbol{\mathcal{X}}}$ is the reconstructed hyperspectral image. In this experiment, we used the corrected Indian Pines dataset, in which some very noisy wavelengths have been left out [141]. This results in a hyperspectral image of dimensions $145 \times 145 \times 200$. For `lsmlsvd_brks`, we compute an MLSVD of multilinear rank $(70, 70, 30)$ with a sparse core. This multilinear rank was obtained by trying a wide range of multilinear ranks and assessing the quality of their corresponding reconstructions.

Table 4.1 shows the quality of the reconstructed images for a range of sampling ratios. Whereas `GAP_TV` is specifically designed for imaging applications, `lsmlsvd_brks` can be applied to a wide range of problems. Regardless, `lsmlsvd_brks` performs approximately equally well in terms of reconstruction quality. The reconstruction quality achieved by `lsmlsvd_brks` is higher than for `KCS`, indicating that the bases estimated by the former suit the data better than the a priori determined bases used in the latter.

4.6 Conclusion and further work

We have considered a general framework of BRKS linear systems with a compact solution, which suits a wide variety of problems. We developed

Table 4.1: This table shows the reconstruction quality, quantified in PSNR (dB), of a hyperspectral image for a range of sampling ratios, obtained with different algorithms. Algorithm `lsmlsvd_brks` performs approximately equally well as `GAP_TV`, which is an algorithm specifically suited for image reconstruction. The lower reconstruction quality for `KCS` indicates that the bases estimated by `lsmlsvd_brks` suit the data better than the a priori determined bases used in `KCS`.

Algorithm	Sampling ratio		
	0.02	0.05	0.1
`lsmlsvd_brks`	25.58	28.67	28.25
`GAP_TV`	25.68	28.18	30.28
`KCS`	10.94	16.40	22.83

efficient algorithms for computing an MLSVD, CPD or TT constrained solution from a BRKS system, allowing the user to choose the decomposition that best matches their specific application. The efficiency of these algorithms is enabled on one hand by the BRKS linear system, since such a system produces multiple compressed versions of the tensor and thus splits the problem into a number of smaller ones, and on the other hand by the low (multilinear-/TT-)rank constrained solution. With these algorithms, real data can be accurately reconstructed using far fewer compressed measurements than the total number of entries in the dataset. We have derived conditions under which an MLSVD, CPD or TT can be retrieved from a BRKS system. The corresponding generic versions of these conditions allow us to choose the dimensions of the generating matrices such that a solution can generically be found. Through numerical experiments, we have shown that these algorithms can be used for computing tensor decompositions in a randomized approach. In the case of the CPD, our algorithm needs less computation time than the alternative algorithms. Additionally, we have illustrated the good performance of the algorithms for reconstructing compressed hyperspectral images, despite not being specifically developed for this application. In further work, we will look into parallel implementations for the algorithms in this chapter.

5 Tensorlab$^+$: A case study on reproducibility in tensor research

ABSTRACT

Tensor methods emerge as an important class of basic techniques, generalizing matrix methods to multi-way data and models. We have recently released Tensorlab$^+$, which is a downloadable archive of code and data that allows peers to reproduce the experiments reported in our publications on tensor decompositions and applications. We briefly discuss the basic tensor tools and give an introduction to the contents of Tensorlab$^+$. We elaborate on the steps that were taken to ensure the reproducibility of the experiments and the quality of the code.

This chapter is based on S. Hendrikx, R. Widdershoven, N. Vervliet, and L. De Lathauwer, "Tensorlab$^+$: A case study on reproducibility in tensor research", *Technical Report 22–116, ESAT-STADIUS, KU Leuven, Leuven, Belgium*, 2022, Accepted for publication in IEEE Computing in Science and Engineering.

5.1 Introduction

Replication is the gold standard for guaranteeing the validity of scientific claims in research. However, performing an independent replication of a study, especially if it is a large study, can be expensive and time-consuming. Furthermore, replication studies are not well incentivized, as journals often prefer original research. Many experiments in research are numerical, for which reproduction is an attractive alternative to replication for assessing scientific claims, as it requires far less time and resources than the latter.

Reproducibility in research is defined as "obtaining consistent results using the same input data; computational steps, methods, and code; and conditions of analysis" [143]. In the context of numerical experiments, this boils down to running the code of the original authors on the data they collected or used. As a result, numerical experiments are reproducible when the code and data are publicly available. While any code that produces the figures and tables from the corresponding paper technically suffices, high quality code is much more useful to the broader research community. Naturally, reproducing the results of numerical experiments is much easier than replicating them through an independent study, since the latter requires collecting a new dataset and performing the analysis again. The former allows anyone to inspect the data, numerical analysis and results of the original authors in order to assess the validity of their claims. The only cost of reproduction is the computational effort required to rerun the experiments.

Besides enabling reproducibility, sharing code and data has other major advantages. First, the different steps of the numerical analysis are briefly discussed in a paper, but many details are left out as it is often infeasible to present every single one. By sharing the code, every parameter and every step of the analysis is made available, making it much easier for other researchers to gain a deep knowledge of and to build upon the original research. Second, in numerical experiments, results are often compared to results obtained by similar algorithms in terms of both accuracy and computation time. If the code for these algorithms is available, such a comparison can easily be performed. If this is not the case, one might have to write one's own implementation using the limited details in the paper. A possible skill gap in terms of code development with the original authors may result in incorrect conclusions being drawn from this comparison. For example, an increase in performance could be interpreted as an algorithmic improvement, while the actual cause is that it is being compared to an implementation of an algorithm that is worse than the original, unavailable one. Given the advantages, sharing code should be the standard in research involving computational experiments.

A considerable amount of research on reproducibility has been published in literature [144], [145]. Some articles indicate concern about the current state of reproducibility in research. A survey conducted by *Nature* revealed that 90% of the respondents think there is a reproducibility crisis and 70% tried and failed to reproduce research performed by others [146]. Reproducibility has also received attention in machine learning in general and even in research on tensor decompositions [57]. The latter work is focused on the retrieval of the true factor matrices of tensor decompositions that are unique, whereas this chapter is about achieving reproducibility in tensor research in general.

Despite the attention given to reproducibility in literature and all the advantages of sharing code, in practice and in our personal experience, code and data are often not available. The largest barrier withholding authors from publishing their code is the time and effort it takes to clean up and document the code [147]. This is understandable, as software development is not a researcher's main focus. Still, sharing code that is lacking in quality is already more helpful than not sharing any code at all.

Since the code for some of our own publications was not available either, we have recently released[1] Tensorlab$^+$ [148]. It is a downloadable archive of high-quality code and data. Its main purpose is to enable others to easily reproduce the experiments in the corresponding publications. It also allows peers to study the algorithms in detail and facilitates their use as a baseline for comparison with other algorithms in future research. The approach that we adopted for the development of this archive may be useful for anyone interested in releasing reproducible code related to their own work.

In the next section, we elaborate in more detail on the contents and objectives of Tensorlab$^+$ and how these objectives have been achieved. Tensors and tensor decompositions form the foundation of our research and therefore are also at the core of Tensorlab$^+$. These concepts are introduced in Section 5.3. In Section 5.4, we showcase some example applications of tensors, for which the results can be reproduced through Tensorlab$^+$. We specifically focus on some key elements of these experiments that guarantee their reproducibility.

5.2 Tensorlab$^+$

In 2016, we released the latest version of Tensorlab[2], which is a MATLAB software package for working with tensors, computing tensor decompositions

[1] https://www.tensorlabplus.net
[2] https://www.tensorlab.net

and performing complex optimization. It consists mainly of tensor decomposition algorithms and utility and visualization functions, which are designed in a user-friendly way. Tensorlab$^+$, on the other hand, is a downloadable archive of code and data specifically aimed at reproducing experiments from our publications. Tensorlab$^+$ is an extension of Tensorlab, which is where it gets its name, since the former uses the functionality of the latter. In this section, we elaborate on the contents of Tensorlab$^+$ and our objectives for this archive besides reproducibility. These objectives have been achieved by translating them into a set of concrete guidelines for code development. Adherence to these guidelines was checked by employing quality control procedures and tools.

Concretely, Tensorlab$^+$ is a collection of scripts, algorithms and utility functions, which are grouped per publication, as illustrated in Fig. 5.1. Currently, it contains code for reproducing the results of 43 publications. We will continue to add code for our future publications. Depending on the needs of the user, the code for each publication can be downloaded separately or the entire archive can be downloaded at once. First and foremost, the archive contains the algorithms that were introduced in their respective publications in the case that those algorithms were not yet available in Tensorlab. Second, tutorials have been included that illustrate how the algorithms can be applied, which makes them easy to use. Third, the archive contains experiment scripts that can be used to reproduce the figures and tables from the corresponding paper. These scripts contain all the correct parameter values and can thus be run without needing to be altered. Finally, for each paper, a README file is included, which gives an overview of the included files and lists code and data dependencies.

On top of the main objective of reproducibility, we aim to ensure that Tensorlab$^+$ is of high quality and easy to use. Specifically, the former means that the code should be readable, well-documented, and efficient and that it functions correctly and consistently. The latter means that it should be easy to reproduce an experiment or apply an algorithm without a need to study all the details in the paper or code. These additional objectives both improve the reproducibility of the results and make the code more valuable in its own right. The latter enables one to, for example, apply an algorithm to another problem or use an algorithm as a benchmark for one's own.

While Tensorlab$^+$ aims to facilitate reproducibility in tensor research, it is important to note that the software is subject to specific licensing terms set forth by KU Leuven. The license grants a non-transferable, time-limited use for academic, non-commercial purposes only. Commercial use requires a separate license agreement. Any redistribution, modification, or improvement of the software necessitates prior written approval from KU Leuven. Users are advised to review the full license agreement on the website for a comprehensive understanding of these terms and their implications for research use and

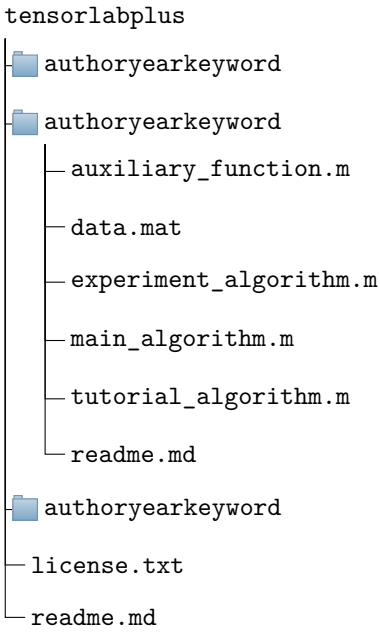


Figure 5.1: In the full archive, the code for each publication is located in a separate folder. The name of this folder consists of the last name of the first author, the publication year and a keyword for convenience.

reproducibility.

5.2.1 Development guidelines

The general objectives were translated into a concrete set of guidelines. We wrote documentation that outlines those, to streamline the process of adding new code to Tensorlab$^+$. Most guidelines correspond to good coding practices that are common in software development but less so in research. They can be summarized as:

1. Ensure easy readability of code by adding line comments and by using a proper naming convention for variables and functions. All files have documentation in the style of built-in Matlab functions. Documentation of a function consists of a one-line summary on top, followed by a longer summary that includes a description of input and output arguments, and names of related functions. Additionally, the names of the authors, the

version history and references are given. The format of an experiment script is similar, but the longer summary is a description of the experiment itself and it references the reproduced figure or table. Readability is a key aspect of reproducibility since it makes the different steps of the experiment, the algorithmic choices made by the authors and the parameters settings more clear.

2. Simplify and shorten code, without affecting its readability. Optimize code with a profiler and use built-in vectorized functions when possible.

3. Add tests for important functions and algorithms. In this way, changes to the code that introduce errors or hamper the reproducibility of experiments can be detected soon after their introduction.

4. Use version control software, e.g. GIT, for code development. Even small changes to code can cause an experiment to no longer produce the same results. By going back into the version history, the cause can be found and addressed. Since Tensorlab+ is a large project involving the code for many papers, changes were often made simultaneously by multiple contributors and for different papers. Version control software makes combining these, possibly conflicting, changes far easier than doing so manually.

5.2.2 Quality control

To guarantee the reproducibility of code in different software environments, two different approaches are possible. One approach is to ensure that the environment in which the code is executed is always exactly the same, which in turn ensures the reproducibility of the results. To this end, one may use a package manager such as Conda or, taking it a step further, a virtual container like Docker. Another approach is to allow the environment to change and check whether the results remain reproducible. We opted for the second approach and created an automated tool for checking the reproducibility of the code. Specifically, this software tool runs all tests to ensure the correctness of the algorithms and also runs all experiment and tutorial scripts to ensure that they complete without errors. Additionally, the tool bundles the relevant files for each individual paper and for the entire collection, while resolving internal code dependencies in which functions are used for multiple papers. The resulting packages are then made available on the Tensorlab+ website in compressed form.

The last step in the process of adding code to Tensorlab+ is to have the new code reviewed by team members to check whether it adheres to the guidelines mentioned above. The review stage consists of multiple rounds, performed by

at least two different reviewers. During these rounds, the quality of the code is further improved according to the reviewers' comments. Additionally, the reproducibility is checked by comparing the results of the experiments to those reported in the corresponding paper. These reviews were facilitated by GIT merge requests. These ensure that the review process is fully completed before the new code is added because approval is required for performing the merge.

5.2.3 Lessons learned

In our opinion, the most valuable lesson we learned when developing Tensorlab$^+$ was to keep reproducibility in mind from the very start of writing code. This may seem obvious, but a considerable amount of work on this project involved retroactively improving the code for experiments in older publications. For example, for some experiments it was difficult to retrieve the script that produced the reported results. In other cases, available older code produced results that did not exactly match those reported in the corresponding paper. Resolving these issues long after publication is of course much more time-consuming than putting a bit more effort into the code from the beginning. In the meantime, we have cultivated a mindset for incorporating the guidelines for reproducibility from the beginning of any new research project.

Initially, the guidelines were not written down and they evolved as the Tensorlab$^+$ project progressed. This means that code that was developed in the early stages, did not necessarily meet the later criteria. It is beneficial to set the standards for one's code as early as possible. Note that good coding practices and documentation styles to start from are already available for different programming languages.

For a large project, we also advise the use of automated tools for tasks like testing, building and formatting code. In this way, relatively simple issues are resolved before the code is reviewed by team members so that the reviews can focus on more important aspects like reproducibility.

5.3 Introduction to tensors and decompositions

The experiments in Tensorlab$^+$ involve tensors and tensor decompositions. A tensor is a multi-way array of numerical values and the order of a tensor corresponds to the number of dimensions. Therefore, vectors and matrices are first- and second-order tensors, respectively. Representing higher-order data by tensors is inherently advantageous since the multidimensional structure

is preserved, while this structure would be lost if the data is flattened and represented by a matrix. Although the generalization from matrix to tensor is straightforward in a sense, that is, stacking multiple matrices of the same size in a third mode yields a third-order tensor, the properties of a tensor are far from simple generalizations of matrix properties. For example, the column rank, row rank and rank of a matrix are all equal and can easily be computed using a rank-revealing decomposition, for instance, the QR factorization. In a tensor, the dimensions of the subspaces spanned by the vectors along each mode are not necessarily equal to each other, nor equal to the rank of the tensor, which is in general NP-hard to compute. The different properties of tensors give tensor decompositions certain advantages over their matrix counterparts. For example, tensor decompositions generally offer greater compression ratios and some are unique under mild conditions. Thanks to these advantages, tensor methods have been applied in a wide range of fields, such as speech and audio processing, biomedical engineering, machine learning and chemometrics [4]. An overview of the tensor decompositions discussed in this section is shown in Fig. 5.2.

A matrix of rank R can be written as the matrix product of two matrices, with R columns and R rows, respectively, and also as the sum of R rank-1 matrices

$$\mathbf{M} = \mathbf{A}\mathbf{B}^{\mathrm{T}} = \sum_{r=1}^{R} \mathbf{a}_r \mathbf{b}_r^{\mathrm{T}} = \sum_{r=1}^{R} \mathbf{a}_r \otimes \mathbf{b}_r, \tag{5.1}$$

with $\mathbf{M} \in \mathbb{R}^{I \times J}$, $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_R \end{bmatrix} \in \mathbb{R}^{I \times R}$ and $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \dots & \mathbf{b}_R \end{bmatrix} \in \mathbb{R}^{J \times R}$ and $\otimes$ denoting the outer product. This decomposition is not unique, since

$$\mathbf{M} = (\mathbf{A}\mathbf{W}) \left(\mathbf{B}\mathbf{W}^{-\mathrm{T}}\right)^{\mathrm{T}} \tag{5.2}$$

is also a rank-R decomposition of $\mathbf{M}$ for any invertible matrix $\mathbf{W} \in \mathbb{R}^{R \times R}$. Note that tensors and their decompositions are defined over the real field for simplicity; it is also possible to define them over the complex field.

The rightmost expression in Eq. (5.1) is straightforward to generalize to higher-order tensors. Namely, a rank-1 tensor of order N equals the outer product of N nonzero vectors. A tensor can be decomposed as

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{\widetilde{R}} \mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r, \tag{5.3}$$

with $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I \times J \times K}$ and factor matrices $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_{\widetilde{R}} \end{bmatrix} \in \mathbb{R}^{I \times \widetilde{R}}$, $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \dots & \mathbf{b}_{\widetilde{R}} \end{bmatrix} \in \mathbb{R}^{J \times \widetilde{R}}$ and $\mathbf{C} = \begin{bmatrix} \mathbf{c}_1 & \dots & \mathbf{c}_{\widetilde{R}} \end{bmatrix} \in \mathbb{R}^{K \times \widetilde{R}}$, which is called a polyadic decomposition. The rank R of a tensor is defined in terms of this decomposition, namely it is the minimum value of $\widetilde{R}$ for which a decomposition

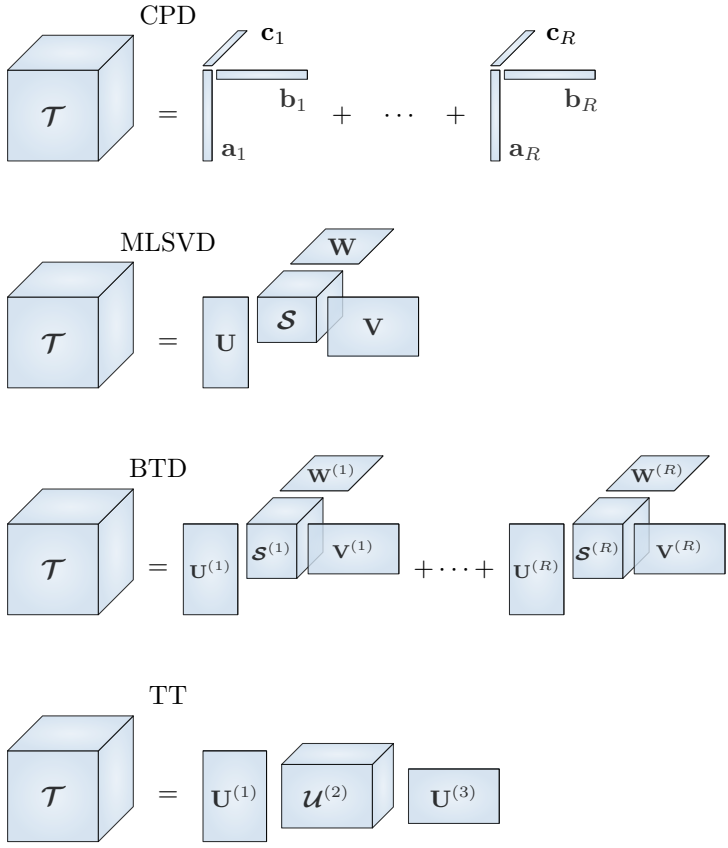


Figure 5.2: Graphical representations of basic tensor decompositions.

as in Eq. (5.3) is possible or, in words, the minimal number of rank-1 tensors that sum up to $\mathcal{T}$. If the number of terms is minimal, then the decomposition is called the canonical polyadic decomposition (CPD). Although computing the exact rank of a tensor is NP-hard in general, computing a low-rank approximation of a tensor often works well in engineering [76].

As previously mentioned, the CPD is unique under mild conditions, making it a very useful tool in applications such as blind signal separation (BSS). In BSS, the goal is to recover a set of unobserved source signals given multiple observed signals that are mixtures of these source signals. The mixing coefficients are unknown as well. Many types of BSS problems can be translated into the computation of the CPD of a tensor $\mathcal{T}$ that is derived from the observed signals. The source signals and mixing vectors can then be retrieved if the

decomposition in Eq. (5.3) is unique. The most famous CPD uniqueness condition is Kruskal's condition. The generic variant of this condition, which ignores linear dependencies of factors that can only occur in a set of measure zero, is a simple relation between the dimensions of the tensor and its rank:

$$\min(I, R) + \min(J, R) + \min(K, R) \geq 2R + 2.$$

More conditions for the uniqueness of the CPD are available in literature; some are even less restrictive than Kruskal's condition and also have simple generic counterparts [4]. In contrast, a specific matrix decomposition like the singular value decomposition (SVD) owes its uniqueness (up to elementary indeterminacies) to much more restrictive constraints, including orthonormality of the singular vectors. Such constraints may not be very useful in BSS, since neither mixing vectors nor sources are orthonormal in general.

Additionally, tensor decompositions generally offer greater compression ratios for low-rank approximation than matrix decompositions. Consider a matrix and an Nth order tensor, in which each dimension is equal to I. The compression ratios for a matrix decomposition, as formulated in Eq. (5.1), and the CPD are $\mathcal{O}\left(\frac{I^2}{2IR}\right)$ and $\mathcal{O}\left(\frac{I^N}{NIR}\right)$, respectively. The latter ratio even increases exponentially for increasing N. One might expect that R also increases as the order increases; however, this is not the case in applications where R has a physical meaning, such as the number of sources in BSS.

The multilinear singular value decomposition (MLSVD) aims to extend some of the very useful properties of the SVD to higher-order tensors. It can be expressed in the form

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{S}} \cdot_1 \mathbf{U} \cdot_2 \mathbf{V} \cdot_3 \mathbf{W},$$

with factor matrices $\mathbf{U} \in \mathbb{R}^{I \times R_1}$, $\mathbf{V} \in \mathbb{R}^{J \times R_2}$ and $\mathbf{W} \in \mathbb{R}^{K \times R_3}$, core tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ and $\cdot_n$ denoting the mode-n product. The generalization from SVD to MLSVD becomes more apparent if the SVD is also formulated using the mode-n product:

$$\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} = \boldsymbol{\Sigma} \cdot_1 \mathbf{U} \cdot_2 \mathbf{V}.$$

Like the factors $\mathbf{U}$ and $\mathbf{V}$ in the SVD, factors $\mathbf{U}, \mathbf{V}$ and $\mathbf{W}$ are column-wise orthonormal.

However, there are also differences between the properties of the SVD and the MLSVD. First, the dimensions of the subspaces spanned by the vectors along each mode of a tensor are not necessarily equal. The multilinear rank $(R_1, R_2, \ldots, R_N)$ of a tensor captures the dimensions of these subspaces. Second, whereas the matrix singular values appear directly on the diagonal of $\boldsymbol{\Sigma}$, the

multilinear singular values equal the Frobenius norms of the slices of the core along each mode. A slice is a subtensor of order $N-1$, obtained by fixing all but one index. Third, the core tensor $\boldsymbol{\mathcal{S}}$ possesses no diagonal structure, unlike its counterpart $\boldsymbol{\Sigma}$ in the SVD. Instead, the slices of the core are mutually orthogonal along each mode. In fact, this property also holds true for $\boldsymbol{\Sigma}$, since the rows (and the columns) of a diagonal matrix are mutually orthogonal. Fourth, while truncating the SVD to R components yields the best rank-R approximation of a matrix, truncating the MLSVD does not necessarily yield an optimal low multilinear rank approximation in the least squares sense.

Although truncation typically yields a suboptimal low multilinear rank approximation, it is relatively close to the optimum in terms of approximation error, making the MLSVD a powerful tool for compression. The square of the approximation error is upper bounded by the sum of the squared multilinear singular values that have been discarded. Next, in applications with a sufficiently large signal-to-noise ratio, the dominant multilinear singular values correspond to the signal, while the other, less significant values correspond to noise. In this case, truncation serves as a method for signal denoising. The dominant multilinear singular vectors also constitute a basis for the "signal subspace" along each mode. Additionally, the MLSVD may be used for an extension of principal component analysis to the higher-order setting, since the multilinear singular vectors correspond to the principal components in each mode. Finally, the MLSVD can be computed as a sequence of N SVDs; note that here the orthogonality is desired, as it is numerically advantageous.

As a rank-1 tensor is rather simple, the CPD might not be the right tool in applications where the components of interest possess a more complex structure. In this context, the block term decomposition (BTD) may be a more suitable tool. Each term in a BTD has low multilinear rank instead of rank equal to one. This gives the model the flexibility to capture more complex components while retaining mild uniqueness properties.

Tensors suffer from the curse of dimensionality, that is, the number of entries in a tensor scales exponentially with the order; for a tensor in which each dimension is equal to I, the number of entries equals I^N. This makes it infeasible in terms of computation time and storage to work directly with tensors of very high order. The MLSVD is not suitable in the high-order setting, since the core has the same order as the original tensor. In this case, decompositions such as the tensor train (TT) or hierarchical Tucker decomposition are more appropriate, which are both specific types of the broader class of tensor networks. They are able to break the curse of dimensionality because they consist of tensor factors of order three at most. Moreover, thanks to orthogonality, they can be computed in a numerically reliable manner, similar to the MLSVD. Tensor networks are of great interest in fields such as scientific computing and quantum

many-body physics [149]. In fact, tensor trains have allowed one to solve very large linear systems, even involving more unknowns than the number of atoms in the universe, while requiring a limited amount of computation time.

5.4 Example applications of tensor methods

Tensorlab$^+$ contains many experiments that showcase applications of tensor tools. Some of these experiments are discussed in this section, with a special focus on how they were designed to be reproducible.

For example, script `experiment_figure4.m` in folder `bousse2017bss` reproduces a proof-of-concept of a tensor-based method for BSS [10]. Consider the basic linear signal model

$$\mathbf{X} = \mathbf{MS} + \mathbf{N}, \tag{5.4}$$

in which the rows of $\mathbf{X}$ are the observed signals, the rows of $\mathbf{S}$ are the source signals, $\mathbf{M}$ contains the mixing vectors and $\mathbf{N}$ is a perturbation caused by noise.

The matrix factorization in Eq. (5.4) is generally not unique, as shown in Eq. (5.2). Therefore, the source signals and mixing vectors cannot be uniquely retrieved by computing a matrix decomposition of $\mathbf{X}$. This means that some assumptions are required to ensure their retrieval. For example, the source signals may be assumed to exhibit low-rank structure after segmentation, i.e., after stacking consecutive segments in a matrix [10]. This assumption is reasonable for signals that can be closely approximated by, for instance, a sum of exponentials, since segmenting a sum of exponentials produces a low-rank matrix. In the script, this assumption is exploited by segmenting the observed signals in $\mathbf{X}$ using the `segmentize` function from Tensorlab. It can be shown that segmenting the rows of $\mathbf{X}$ transforms the problem in Eq. (5.4) into a BTD. The rth term of this BTD is a segmented representation of the contribution of the rth source signal to the observed data. This term has multilinear rank $(L_r, L_r, 1)$ if the signal can be expressed as a sum of L_r exponentials. The BTD is computed using the `lp_nls` algorithm that is included along with this script. The source signals can be retrieved, without the need of restrictive constraints like orthogonality, thanks to the uniqueness of the BTD under mild conditions. In this particular experiment, no noise $\mathbf{N}$ is present and the original source signals can be retrieved exactly.

The BTD in the reformulated BSS problem is computed iteratively using an optimization algorithm. The choice of the starting point is important for reproducibility, since it affects the number of iterations required to get to the final solution as well as the final solution itself. Tensor problems are generally

nonconvex, which means that the algorithm might end up in a local optimum, depending on the choice of starting point. This variability in the final solutions can be mitigated by starting from a fixed point. For example, under reasonable conditions the CPD of a noise-free tensor can be computed deterministically by means of a generalized eigenvalue decomposition and this solution may provide a good initial point for the noisy case. However, it is not always possible to find a good initialization for more complicated decompositions, such as the BTD, so a random starting point is often used instead. As a result, running an optimization algorithm multiple times may result in different solutions of varying quality. In experiments such as `experiment_figure4.m` the goal is to show that the algorithm can solve a specific instance of the BSS problem. In this case, the reproducibility of the results can be guaranteed by using a fixed seed for the random number generator.

Besides experiments for a single concrete problem, papers typically also contain experiments that show the general performance of an algorithm, such as in `experiment_figure6.m` in folder `bousse2017bss`. Here, fixing the seed of a random number generator is not the appropriate way to guarantee reproducibility. Instead, the experiment should be designed in a way that reveals the average performance in a robust manner. To this end, one should perform a Monte Carlo experiment and aggregate the results using a robust statistic. For example, 100 Monte Carlo trials are performed in `experiment_figure6.m` and they are aggregated using the median. Depending on the number of trials, running the experiment can be computationally expensive. For some scripts, including this one, precomputed results are included such that figures in the corresponding publication can be created without effort. In other scripts, the number of trials is set to a lower default value that still allows one to verify the main conclusions, while the number that was used for the publication is mentioned in a comment. This gives a user the choice between reproducing the original results or quickly producing results that might be slightly different.

Figure 5.3 shows that a badly designed experiment might show results that give a skewed view of the difference in accuracy between two algorithms. When only one problem is used in the experiment, the accuracy of the first algorithm is affected by an outlier at a signal-to-noise ratio (SNR) of 30 dB. Computing the median over five problems seems to indicate that the difference in accuracy is rather small, however, aggregating over 50 shows that the difference is actually larger, indicating that doing so over five is not enough.

Tensorlab$^+$ further includes a variety of methods for computing a CPD or BTD with constraints imposed upon the factor matrices. Constraints can be useful for multiple reasons, including the incorporation of prior knowledge, for example, Vandermonde structure in harmonic retrieval or nonnegative values summing to one for concentrations. Besides offering algorithms for computing

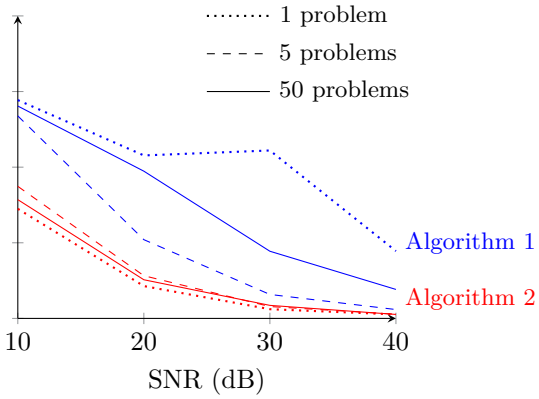


Figure 5.3: The comparison between the accuracy of both algorithms is not representative of the actual difference in accuracy when the results are aggregated (using the median) over one or five problems. Therefore, the experimental results in Tensorlab$^+$ are aggregated over plenty of problems to ensure robustness and to give a representative estimate of the actual accuracy of the algorithms.

decompositions with specific constraints, Tensorlab$^+$ also includes experiments that use the structured data fusion framework [130]. This framework allows one to impose various constraints at once through parameterization, and to couple the decompositions of multiple tensors. Here, coupling means that the decompositions share some factor(s) or underlying generator(s). Structured data fusion is very versatile and can be used to quickly try out different models and constraints on the problem at hand.

5.5 Conclusion

Reproduction is an attractive alternative to replication, especially in tensor-based science and engineering, since many experiments mainly consist of running code. Sharing code offers additional advantages besides making research reproducible. We have recently released a comprehensive downloadable archive of code and data for reproducing experiments in our publications. Our approach to ensuring the reproducibility of experiments might be useful to other researchers. We have given an overview of what a user can expect of Tensorlab$^+$, with an emphasis on the underlying motivations and insights of broader interest.

6 Conclusion

A dataset may be incomplete due to reasons that are inherent to the application or because of the prohibitive cost of acquiring every entry. Implicit data is present in problems that can be expressed as the solution to a linear system. The latter situation occurs in particular in CS, namely to estimate a compact but global approximation of a dataset using a small number of compressed measurements. In both the incomplete and implicit settings, tensor decompositions have been successfully used for tensor completion, extracting underlying components, compression or compactly representing data. Using tensor decompositions is advantageous since they offer possibilities that are not present in their matrix counterparts. For example, the former have mild uniqueness conditions, the ability to break the curse of dimensionality and are better at capturing higher-order characteristics of data. Since most tensor decomposition algorithms for incomplete and implicit data are optimization-based, the first main objective of this thesis is to enable the algebraic computation of tensor decompositions in both settings.

Algebraic algorithms can, under some conditions, exactly retrieve the original low-rank tensor if no noise is present in the data. Optimization-based algorithms can find better solutions in the noisy case, but the quality is highly dependent on the starting point of the algorithm. Selecting a starting point for an optimization problem associated with the computation of a tensor decomposition is nontrivial, as the objective function is usually non-convex. The solution obtained with an algebraic algorithm serves as a good starting point for an optimization-based algorithm. This is because algebraic algorithms are generally computationally efficient since they consist of standard linear algebra operations, and the resulting starting point is likely to be close to a minimum of the objective function. Further refinement of the algebraic solution with an optimization-based algorithm then yields an even more accurate solution at a reduced computational cost, thus combining the advantages of both types of algorithms.

In order to allow for the algebraic computation of tensor decompositions, instances of incomplete and implicit data with specific structure are considered. Besides enabling algebraic algorithms, the types of structure considered in this thesis are also relevant for applications. Chapter 3 considers incomplete data that consists of completely observed or completely missing fibers along one

mode of the tensor. Fiber-wise observation patterns may occur in applications in which collecting data along one of the modes is easier than along the others, such as in the traffic speed data example. If this pattern is present, then it is possible to compute an MLSVD of an incomplete tensor in an algebraic manner. Furthermore, Chapter 4 concerns linear systems with a BRKS coefficient matrix. In CS, Kronecker-structured bases are used for higher-order data, as this corresponds to handling each mode independently. This specific structure allows the algebraic computation of the MLSVD and TT and the computation of the CPD with an MLSVD precompression step in the implicit setting.

Reproducible experiments were carried out to highlight and validate the capabilities of the algorithms introduced in this thesis. Reproducibility is an important aspect of numerical experiments, and therefore it is the second main objective of this thesis. To this end, Tensorlab$^+$ was developed as a framework for sharing the code and data associated with publications from our team. Currently, Tensorlab$^+$ contains code for many of our older and current publications, and we aim to continue to use this framework for releasing future work. The lessons learned during the development may be useful to others looking to ensure the reproducibility of their numerical experiments.

6.1 Overview of contributions

This section provides an overview of the contributions that were made in this thesis on a chapter-by-chapter basis. Some chapters involve work that resulted from collaborations with team members. If a contribution was made by a collaborator, then their name is mentioned. If no name is mentioned, then the contribution was provided by the doctoral candidate.

Chapter 3

- *Algebraic algorithm for computing the MLSVD of a fiber-wise observed tensor.* An incomplete tensor with a specific observation pattern, namely one in which the fibers along one mode are either completely observed or completely missing, enables the algebraic computation of the MLSVD. The factor matrix corresponding to the mode along which the fibers lie can simply be computed through the SVD of the observed fibers. The remaining factor matrices are obtained by estimating the column space of an incomplete matrix, that is, a matrix containing multiple completely observed submatrices. The core tensor is computed by solving

a linear system. The algorithm was derived by Mikael Sørensen and it was implemented by the doctoral candidate.

- *Kernel method for estimating the column space of an incomplete matrix.* Solving this matrix completion problem is the most challenging part of computing the MLSVD of a fiber-wise observed tensor. The kernel method for solving this problem uses the fact that the nullspace of each completely observed submatrix is orthogonal to the column space of interest. In the first step, a basis for the nullspace of each completely observed submatrix is computed. As the column space of the incomplete matrix is orthogonal to all of these bases, it corresponds to the left nullspace of the matrix that consists of these bases concatenated together. The idea for this method was contributed by Mikael Sørensen.

- *Principal subspace method for estimating the column space of an incomplete matrix.* The column space of each completely observed submatrix corresponds to a part of the column space of the incomplete matrix. Unit vectors with a one in the position of an unobserved row can be concatenated to a basis for the column space of a completely observed submatrix. As a result, the column space of interest is a subspace of the space spanned by this augmented basis. The column space of the incomplete matrix then corresponds to the intersection of the spaces spanned by the augmented basis of each completely observed submatrix. Additionally, a more efficient principal subspace method is introduced, which reduces the computational cost at the expense of squaring the condition number of the problem.

- *Deterministic and generic uniqueness conditions for estimating the column space of an incomplete matrix.* The deterministic and generic uniqueness conditions for the column space estimation problem were contributed by Mikael Sørensen. The deterministic conditions can be interpreted as a minimum required amount of overlap between the observed rows in each pair of completely observed submatrices. If this is satisfied, then the intersection of the spaces spanned by the augmented bases corresponds to the column space of the incomplete matrix. The uniqueness conditions of the matrix completion problem are the most important part of the uniqueness conditions for computing the MLSVD of a fiber-wise observed tensor.

- *Traffic speed data completion application.* Traffic speed data can have missing data, which is usually filled in before using it for other purposes. Since this type of data can exhibit a fiber-wise observation pattern, this problem suits the algorithm introduced in Chapter 3 well.

Chapter 4

- *Connection between tensor-based CS and RTD algorithms.* In CS a dataset is compressed through multiplication with a (possibly structured) coefficient matrix, i.e., by taking linear combinations of its entries. In RTD, a tensor is first compressed through the multiplication with one or multiple random matrices to speed up the following steps of the tensor decomposition algorithm. While the compression matrices can be chosen in RTD, they may depend on the application in CS. Regardless, compression is performed in both CS and RTD through matrix multiplication. As a result of this similarity, efficient RTD-like algorithms can be used in the CS setting.

- *Algebraic algorithms for computing an MLSVD or a TT from a BRKS system.* In CS, Kronecker-structured measurement matrices can be used for higher-order data such that each mode is compressed individually. By vertically stacking multiple block rows of Kronecker-structured matrices, a BRKS linear system is obtained. This specific structure allows one to algebraically compute an MLSVD or a TT solution to this system.

- *Computing a CPD from a BRKS system with MLSVD precompression.* The computation of the CPD can be sped up by first orthogonally compressing the tensor by means of an MLSVD. This precompression step can be extended to the implicit setting, namely for BRKS linear systems. This approach yields an efficient algorithm in which each CPD factor matrix is computed from a different compressed version of the tensor.

- *Deterministic and generic uniqueness conditions for the retrieval of an MLSVD, CPD or a TT from a BRKS linear system.*

- *Compressed hyperspectral imaging application.* When computing an MLSVD solution to a BRKS system, the factor matrices play the role of the Kronecker-structured sparsifying basis in KCS. The sparse coefficients in the KCS setting then correspond to an MLSVD with a sparse core tensor. Therefore, the algorithm for computing an MLSVD of a BRKS system can be used to estimate both the sparsifying basis and the sparse coefficients from the compressed measurements simultaneously. In standard CS, only the sparse coefficients are computed from the compressed measurements, whereas the sparsifying basis is provided a priori. The MLSVD approach that estimates both the sparsifying basis and the coefficients at the same time can be used to reconstruct a hyperspectral image from compressed measurements.

Chapter 5

- *A framework enabling the reproduction of tensor research.* Tensorlab$^+$ is a repository that contains the scripts for reproducing experiments from tensor research and includes all other required code that is not available elsewhere (such as in Tensorlab [129]). The code is organized per publication and also includes a set of auxiliary functions that are used in the code for multiple publications. A procedure, implemented by Nico Vervliet, runs all tests in the repository, verifies that all experiments complete without errors and packages the code into downloadable archives.
- *Guidelines for writing reproducible code and for the code review procedure.* These guidelines involve the robust design of experiments and good coding practices, such as simplifying and optimizing code and adding clear documentation and comments. Additionally, the code files in the repository are categorized into different types with their own naming and documentation conventions. Tests are added for the algorithms and important functions, which are used to automatically verify that they function correctly each time new or modified code is uploaded to the website. Next, an internal peer review by team members follows, in which adherence to the guidelines is checked.
- *Adding code for older publications to Tensorlab$^+$.* Besides recent publications, the first release of the repository also includes code for many older publications that was previously unavailable. As a result, a lot of work during the development of Tensorlab$^+$ was done to bring this older code up to the standards of the guidelines. A considerable part of this work was done by the doctoral candidate.
- *Development of the website for Tensorlab$^+$.* The Tensorlab$^+$ repository can be accessed through its website. A user can either download the entire repository or only the code belonging to one publication.
- *Accessible introduction to reproducibility and tensor decompositions.* The publication that Chapter 5 is based on is meant for a broad audience. For this reason, it contains an accessible introduction to tensor decompositions for readers who are not familiar with the topic.

Appendix A

This appendix is based on the master's thesis of the doctoral candidate [150].

- *Using a low-rank tensor to efficiently represent a multivariate polynomial.* A multivariate polynomial can be represented using a symmetric tensor containing its coefficients. Since the number of coefficients scales rapidly with the degree of the polynomial, using a low-rank tensor for the coefficients makes the representation much more compact. This representation enables efficient algorithms for multivariate regression.

- *Efficient Gauss–Newton algorithm for computing the low-rank coefficients.* The multivariate regression problem can be expressed as a linear system with a Khatri–Rao structured coefficient matrix and the low-rank coefficients as its solution. During the computation of the factor matrices of the low-rank coefficient tensor, the structure of both the coefficient matrix and the solution can be simultaneously exploited, which results in an efficient optimization-based algorithm for multivariate regression.

- *Algebraic algorithm for computing the low-rank coefficients.* The algebraic algorithm first solves the linear system for the coefficient tensor without a low-rank constraint. This algorithm uses the symmetry of the coefficient tensor by only computing its unique entries. In the next step, a low-rank approximation of the coefficient tensor is computed. This algorithm provides a useful alternative to the optimization-based one in case the degree is low, which limits the number of polynomial coefficients, and enough data points are available. The algebraic algorithm was contributed by Nico Vervliet.

- *Materials science application.* Evaluating the properties of a metal alloy on a grid of different concentrations of its components yields a tensor, which can be very large depending on the number of components and the coarseness of the grid. Using multivariate regression, the Gibbs free energy of a particular allow can be accurately modeled using a relatively low number of data points. This application was contributed by Nico Vervliet.

6.2 Perspectives on further work

- *Algebraic computation of a TT from a fiber-wise observed tensor.* An approach similar to the one introduced in Chapter 3 may also be used to compute a TT from a fiber-wise observed tensor. The standard TT algorithm involves the computation of the SVD of multiple matricizations of the tensor. In the case of a fiber-wise observed tensor, the matrix completion approaches in Chapter 3 can be used instead of the SVD for computing the cores of the TT.

- *Algebraic computation of a hierarchical SVD from a fiber-wise observed tensor.* Similar to the TT, computing the hierarchical SVD [96] involves the computation of the SVD of matricizations of the tensor. Therefore, the matrix completion approaches in Chapter 3 can also be used for this decomposition.

- *Updating tensor decompositions as new fibers become available.* In an updating setting, new parts of a data tensor continuously arrive and should be processed. This tensor can be modeled by a tensor decomposition that is updated as each part becomes available. In real-time applications, these updates must be fast. This can be achieved by efficiently updating only the parts of the previous decomposition that are affected by the new data. There are algorithms in the literature that can efficiently update decompositions as new slices of the tensor arrive [151], [152]. This approach can be extended to the case in which the new data consists of individual fibers instead of entire slices.

- *Computing constrained and coupled decompositions from implicit data.* Tensorlab, a software package developed in our team [129], has a framework for simultaneously computing tensor decompositions of multiple tensors, in which constraints can be imposed on the factors and the factors can be shared between decompositions [130]. This allows one to extract information that is shared across multiple datasets and to enhance the interpretability of the factors through the imposition of constraints. The extension of this framework to the implicit data brings these possibilities to the implicit setting.

- *Extension to other decompositions in the implicit setting.* The algorithms in Chapter 4 concern the CPD, MLSVD and the TT. Algebraic algorithms for computing other tensor decompositions, such as the block term decomposition [153] or the hierarchical SVD [96], in the implicit setting could be useful for the same reasons that make the algorithms in this thesis useful.

A Algebraic and optimization based algorithms for multivariate regression using symmetric tensor decomposition

ABSTRACT

Multivariate regression is an important task in domains such as machine learning and statistics. We cast this regression problem as a linear system with a solution that is a vectorized symmetric tensor, which is assumed to be of low rank. We show that the structure of the data and the decomposition can be exploited to obtain efficient optimization methods. Furthermore, we show that an algebraic algorithm can be derived even if the number of given data points is low. We demonstrate good performance of our regression model using a real-life dataset from materials science.

This chapter is based on S. Hendrikx, M. Boussé, N. Vervliet, and L. De Lathauwer, "Algebraic and optimization based algorithms for multivariate regression using symmetric tensor decomposition", in *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Le gosier, Guadeloupe, Dec. 2019, pp. 475–479.

A.1 Introduction

Nowadays large amounts of data are available in various domains, resulting in an increased interest in machine learning methods to extract useful information from this data. Data often also has a higher-order structure, which can be represented using higher-order tensors in a straightforward way. Tensor decompositions are important tools in machine learning and signal processing for extracting hidden components and for compact representations [2], [3], [47]. In this chapter, we combine the advantages of both tensors and machine learning methods by applying tensors to the regression problem. This combination has been made before in the literature [4], [155].

In this chapter we use a polyadic decomposition (PD) to compactly represent the coefficients of the multivariate polynomial regression problem. Other polynomial models that are represented by a tensor decomposition are exponential machines [156], tensor train polynomial classifiers [157], higher-order factorization machines [158], [159] and tensor machines [160]. The optimization problem of our model is formulated as a linear system with a polyadic decomposition constrained solution (LS-CPD) [161]. The coefficient matrix of this system has a Khatri–Rao structure, which we exploit to allow efficient optimization of the variables of the model. We discuss an algebraic and an optimization-based algorithm to solve this problem and perform a computational complexity analysis of the latter. We test our model on a materials science dataset, demonstrating high accuracy while only requiring few data points.

In the remainder of this section we discuss notation and definitions and the connection between multivariate polynomials and symmetric tensors. In Appendix A.2, we explain how a polyadic decomposition is used to obtain a compact polynomial model. In Appendix A.3, we discuss the Gauss–Newton optimization algorithm and in Appendix A.4 the algebraic algorithm that can be used to calculate the variables of the model. In Appendix A.5, the model is tested on a dataset from materials science.

A.1.1 Additional notation and basic definitions

The element-wise Nth power is denoted by $\cdot^{N}$. The factor matrices of a PD are denoted by $\mathbf{B}^{(n)}$ for $n = 1, \ldots, N$ in this chapter. As a multivariate polynomial can be represented by a symmetric tensor, as will be shown next, we focus on the symmetric PD, in which each factor matrix is the same: $\mathbf{B}^{(n)} = \mathbf{B}$ for $n = 1, \ldots, N$.

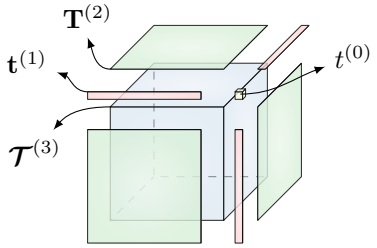


Figure A.1: By applying a process called homogenization, symmetric tensors can also represent the coefficients of non–homogeneous polynomials. For example, by stacking the coefficients $t^{(0)}, \mathbf{t}^{(1)}, \mathbf{T}^{(2)}$ and $\mathcal{T}^{(3)}$ of a degree-3 polynomial into a tensor as shown above, we can represent it with a symmetric third-order tensor. Image reproduced from [11].

A.1.2 Polynomials and symmetric tensors

A multivariate polynomial can be represented by a tensor. For example, consider a *homogeneous* polynomial $p(\mathbf{x})$ of degree $N = 2$ with variables $\mathbf{x} \in \mathbb{R}^I$. The polynomial can then be written using a *symmetric* matrix $\mathbf{T} \in \mathbb{R}^{I \times I}$ as follows[1]:

$$p(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{T}\mathbf{x} = \sum_{i=1}^{I} \left(t_{ii}x_i^2 + \sum_{j=i+1}^{I} 2t_{ij}x_i x_j \right). \tag{A.1}$$

Note that Eq. (A.1) is equivalent to $p(\mathbf{x}) = \mathbf{T} \cdot_1 \mathbf{x}^{\mathrm{T}} \cdot_2 \mathbf{x}^{\mathrm{T}}$, allowing us to generalize in a straightforward way to homogeneous polynomials of degree N using the mode-n product and a symmetric tensor $\mathcal{T} \in \mathbb{R}^{I \times \cdots \times I}$ of order N:

$$p(\mathbf{x}) = \mathcal{T} \cdot_1 \mathbf{x}^{\mathrm{T}} \cdot_2 \mathbf{x}^{\mathrm{T}} \cdots \cdot_N \mathbf{x}^{\mathrm{T}}. \tag{A.2}$$

We can also represent non-homogeneous polynomials by a symmetric tensor by means of homogenization and extending the variables $\mathbf{x}$ to $\mathbf{v} = \left[1, \mathbf{x}^{\mathrm{T}}\right]^{\mathrm{T}}$; see [11] and Fig. A.1.

A.2 Multivariate regression using PD

Given the points $\{\mathbf{x}_m, y_m\}$ for $m = 1, \ldots, M$ of a dataset, fitting model Eq. (A.2) to this data, i.e. $y_m \approx p(\mathbf{x}_m)$, can be reformulated as the following linear system

[1]Note that only the symmetric part of $\mathbf{T}$ is relevant in order to represent the polynomial $p(\mathbf{x})$. This can be proven by substituting $\mathbf{T}_s + \mathbf{T}_a$ for $\mathbf{T}$ in Eq. (A.1), where $\mathbf{T}_s$ is its symmetric and $\mathbf{T}_a$ its asymmetric part.

of equations:

$$\mathbf{y} \approx \mathbf{A}\mathrm{vec}\left(\boldsymbol{\mathcal{T}}\right) \quad \text{with} \quad \mathbf{A} = \underbrace{(\mathbf{X} \odot^{\mathrm{T}} \mathbf{X} \odot^{\mathrm{T}} \cdots \odot^{\mathrm{T}} \mathbf{X})}_{N}, \tag{A.3}$$

in which $\mathbf{y} \in \mathbb{R}^M$ contains the data points and $\mathbf{X} \in \mathbb{R}^{M \times I}$ contains $\mathbf{x}_m^{\mathrm{T}}$ for $m = 1, \ldots, M$ as its rows. In this paper, we assume that the coefficients of the polynomial that approximates the data can be represented by a symmetric tensor that admits a low-rank (symmetric) PD:

$$\mathbf{y} \approx (\mathbf{X} \odot^{\mathrm{T}} \mathbf{X} \odot^{\mathrm{T}} \cdots \odot^{\mathrm{T}} \mathbf{X})\, \mathrm{vec}\left(\llbracket \mathbf{B}, \mathbf{B}, \cdots, \mathbf{B}, \mathbf{c}^{\mathrm{T}} \rrbracket\right). \tag{A.4}$$

Additional weights $\mathbf{c} \in \mathbb{R}^R$ are included for each rank-1 term, allowing negative coefficients for even-degree polynomials. Equation (A.4) is a linear system with a symmetric PD constrained solution *and* a (row-wise) Khatri–Rao structured coefficient matrix $\mathbf{A}$, resulting in an LS-CPD [161] where $\mathbf{A}$ has this specific structure. While a multivariate polynomial has $\binom{I+N-1}{N} = \frac{(I+N-1)!}{N!(I-1)!}$ coefficients, which equals the number of unique entries in $\boldsymbol{\mathcal{T}}$, we need only $R(I+1)$ variables thanks to the compact representation by a symmetric PD, allowing us to efficiently fit a polynomial.

A.3 Gauss–Newton algorithm

Given data $\{\mathbf{x}_m, y_m\}$ for $m = 1, \ldots, M$, a multivariate polynomial approximating this data is found by solving the optimization problem

$$\min_{\mathbf{B},\mathbf{c}} f \qquad \text{with} \quad f = \frac{1}{2} ||\mathbf{r}||_{\mathrm{F}}^2 \tag{A.5}$$

in which the residual vector $\mathbf{r}$ is

$$\mathbf{r} = (\mathbf{X} \odot^{\mathrm{T}} \cdots \odot^{\mathrm{T}} \mathbf{X})\, \mathrm{vec}\left(\llbracket \mathbf{B}, \cdots, \mathbf{B}, \mathbf{c}^{\mathrm{T}} \rrbracket\right) - \mathbf{y}.$$

We solve Eq. (A.5) for $\mathbf{B}$ and $\mathbf{c}$ via a standard Gauss–Newton (GN) algorithm with dogleg trust region (TR) [162]. While this problem can be written and solved as an unstructured LS-CPD Eq. (A.4), the key to deriving an efficient algorithm is to explicitly exploit the Khatri–Rao and multilinear structure when computing the objective function, gradient, and Gramian. These expressions can also be used for other quasi-Newton and nonlinear least-squares algorithms.

The GN algorithm solves Eq. (A.5) by approximating the Hessian with $\mathbf{H}_k$ and solving a linear system [162] in each iteration k. This linear system is $\mathbf{H}_k \mathbf{p}_k = -\mathbf{g}_k$ with $\mathbf{H}_k = \mathbf{J}_k^{\mathrm{T}} \mathbf{J}_k$ the Gramian of the Jacobian $\mathbf{J}_k = \partial \mathbf{r} / \partial \mathbf{z}_k$, GN

step $\mathbf{p}_k$, and the gradient $\mathbf{g}_k = (\partial f/\partial \mathbf{z}_k)^{\mathrm{T}}$ with $\mathbf{z}_k = \left[\mathrm{vec}(\mathbf{B}_k)^{\mathrm{T}}, \mathbf{c}_k{}^{\mathrm{T}}\right]^{\mathrm{T}}$ [162]. This system can be solved exactly using the pseudoinverse of $\mathbf{H}_k$ or inexactly using conjugated gradients. The variables $\mathbf{z}_k$ can then be updated using dogleg TR with $\mathbf{p}_k$ and $\mathbf{g}_k$, or using line search [162]. While a random factor matrix $\mathbf{B}_0$ and coefficients $\mathbf{c}_0$ can be used to initialize the algorithm, starting from an algebraic solution often provides a better initialization; see Appendix A.4.

In the remainder of this section, we derive efficient expressions for the ingredients of the GN algorithm as well as their computational complexities. The complete algorithm and its computational complexity are shown in Algorithm A.1 and Table A.1, respectively.

Algorithm A.1: Symmetric PD-constrained multivariate regression using GN and dogleg trust region.

Input: $\mathbf{X}$, $\mathbf{y}$, and initial values for $\mathbf{B}_0$ and $\mathbf{c}_0$
Output: $\mathbf{B}_{\text{final}}$ and $\mathbf{c}_{\text{final}}$
while *not converged* **do**
  Compute Jacobian $\mathbf{J}_k$ and gradient $\mathbf{g}_k$ using Eq. (A.7)-Eq. (A.8) and Eq. (A.10), respectively.
  Solve $\mathbf{J}_k{}^{\mathrm{T}}\mathbf{J}_k\mathbf{p}_k = -\mathbf{g}_k$ for $\mathbf{p}_k$.
  Update $\mathbf{B}_k$ and $\mathbf{c}_k$, using dogleg trust region with $\mathbf{p}_k$, $\mathbf{g}_k$ and the objective function evaluation via Eq. (A.6).

A.3.1 Expressions

For simplicity, we drop the underscript k. By taking the sum of squared entries of the residuals $\mathbf{r} \in \mathbb{R}^M$, we obtain the value of the objective function f. The residuals can be computed more efficiently by using multilinear identities [86]:

$$\begin{aligned}\mathbf{r} &= \left(\mathbf{X} \odot^{\mathrm{T}} \cdots \odot^{\mathrm{T}} \mathbf{X}\right)\left(\mathbf{B} \odot \cdots \odot \mathbf{B}\right)\mathbf{c} - \mathbf{y} \\ &= \left(\overset{N}{\underset{n=1}{*}} \mathbf{X}\mathbf{B}\right)\mathbf{c} - \mathbf{y} = (\mathbf{X}\mathbf{B})^{\cdot N}\,\mathbf{c} - \mathbf{y}. \end{aligned} \tag{A.6}$$

The Jacobian $\mathbf{J} \in \mathbb{R}^{M \times R(I+1)}$ can be partitioned into two parts $\left[\mathbf{J}_{\mathbf{B}}, \mathbf{J}_{\mathbf{c}}\right]$ corresponding to the variables $\mathbf{B}$ and $\mathbf{c}$ with:

$$\mathbf{J}_{\mathbf{B}} = \frac{\partial \mathbf{r}}{\partial \mathrm{vec}(\mathbf{B})} = \left(N(\mathbf{X}\mathbf{B})^{\cdot(N-1)} \odot \mathbf{c}^{\mathrm{T}}\right) \odot^{\mathrm{T}} \mathbf{X}, \tag{A.7}$$

$$\mathbf{J}_{\mathbf{c}} = \frac{\partial \mathbf{r}}{\partial \mathbf{c}} = (\mathbf{X}\mathbf{B})^{\cdot N}. \tag{A.8}$$

Table A.1: By carefully exploiting the Khatri–Rao structure and the symmetry in Eq. (A.4), we can obtain a significant improvement in the computational complexity.

		Complexity	
	Calls/iteration	Our algorithm	LS-CPD [161]
f	it_{TR}	$\mathcal{O}(MR(I+N))$	$\mathcal{O}(MRI^N)$
$\mathbf{J}$	1	$\mathcal{O}(MR(I+N))$	$\mathcal{O}\left(MRNI^N\right)$
$\mathbf{g}$	1	$\mathcal{O}(MRI)$	$\mathcal{O}\left(MRIN\right)$
$\mathbf{J}^{\text{T}}\mathbf{J}$	1	$\mathcal{O}(MR^2I^2)$	$\mathcal{O}\left(MRI^2N\right)$

Proof. We obtain Eq. (A.7) by computing the derivative row-wise:

$$\begin{aligned}(\mathbf{J}_{\mathbf{B}})_m &= \frac{\partial r_m}{\partial \text{vec}(\mathbf{B})} \\ &= \left(N\left(\mathbf{x}_m^{\text{T}}\mathbf{B}\right)^{\cdot(N-1)} * \mathbf{c}^{\text{T}}\right)\frac{\partial\left(\mathbf{x}_m^{\text{T}}\mathbf{B}\right)}{\partial \text{vec}(\mathbf{B})} \\ &= \left(N\left(\mathbf{x}_m^{\text{T}}\mathbf{B}\right)^{\cdot(N-1)} * \mathbf{c}^{\text{T}}\right)\left(\mathbf{I}\otimes\mathbf{x}_m^{\text{T}}\right) \qquad \text{(A.9)}\end{aligned}$$

using the chain rule and a multilinear identity. By applying Eq. (A.9) for every row, we obtain Eq. (A.7). □

The gradient $\mathbf{g}$ and the Gramian $\mathbf{H}$ are given by

$$\begin{aligned}\mathbf{g} &= \mathbf{J}^{\text{T}}\mathbf{r} \in \mathbb{R}^{R(I+1)} \qquad \text{(A.10)} \\ \mathbf{H} &= \mathbf{J}^{\text{T}}\mathbf{J} \in \mathbb{R}^{R(I+1)\times R(I+1)}.\end{aligned}$$

A.3.2 Computational complexity

If the system $\mathbf{H}_k\mathbf{p}_k = -\mathbf{g}_k$ is solved using the pseudoinverse of the Gramian, the GN algorithm has a computational complexity of $\mathcal{O}\left(\text{it}_{\text{GN}}(MR^2I^2 + R^3I^3)\right)$ where it_{GN} is the number of GN iterations. In Table A.1, we compare the computational complexity of the expressions of our dedicated algorithm with the expressions of the LS-CPD algorithm of [161] that ignores the Khatri–Rao structure of $\mathbf{A}$. The number of TR iterations is denoted by it_{TR}, which is typically 1. Exploiting the structure clearly reduces the complexity.

A.4 Algebraic algorithm

If $\mathbf{A}$ has full column rank, a naive approach for finding a low-rank solution $\mathbf{t} = \text{vec}(\boldsymbol{\mathcal{T}})$ from $\mathbf{A}\mathbf{t} = \mathbf{y}$, is to first solve for $\mathbf{t}$, then reshape $\mathbf{t}$ into tensor $\boldsymbol{\mathcal{T}}$ and finally decompose $\boldsymbol{\mathcal{T}}$ using standard algorithms for computing a CPD [161]. Because of the transposed Kathri–Rao products of $\mathbf{X}$ Eq. (A.3), $\mathbf{A}$ has repeated columns and does not have full column rank: $\text{rank}(\mathbf{A}) \leq F < I^N$, in which $F = \binom{I+N-1}{N}$ which coincides with the number of unique coefficients. However, by restricting the solution $\boldsymbol{\mathcal{T}}$ to the set of symmetric tensors, we show that $\boldsymbol{\mathcal{T}}$ can be found if $\text{rank}(\mathbf{A}) = F$ and therefore M can be as low as F.

Let $\mathbf{U} \in \mathbb{R}^{M \times F}$ be the matrix containing the unique columns of $\mathbf{A} \in \mathbb{R}^{M \times I^N}$ and $\mathbf{P} \in \{0,1\}^{F \times I^N}$ the binary matrix placing the columns of $\mathbf{U}$ at the right position, then

$$\mathbf{A} = \mathbf{U}\mathbf{P}. \tag{A.11}$$

More formally, let $\pi(i_1, i_2, \ldots, i_N)$ be the set of permutations of elements of tuple $(i_1, i_2, \ldots, i_N)$. The column-wise, resp., element-wise definitions of $\mathbf{U}$ and $\mathbf{P}$ are given by

$$\mathbf{u}_j = \mathbf{x}_{j_1} * \mathbf{x}_{j_2} * \cdots * \mathbf{x}_{j_N}$$

$$p_{ji} = \begin{cases} 1 & \text{if } (i_1, i_2, \ldots, i_N) \in \pi(j_1, j_2, \ldots, j_N) \\ 0 & \text{otherwise,} \end{cases}$$

where j is the index corresponding to entry $(j_1, j_2, \ldots, j_N)$ in the colexicographically ordered set $\{(j_1, j_2, \ldots, j_N) \,|\, 1 \leq j_1 \leq j_2 \leq \cdots \leq j_N \leq I\}$ and $i = \sum_{n=1}^{N} (i_n - 1) I^{(n-1)} + 1$.

The solution $\text{vec}(\boldsymbol{\mathcal{T}})$ can be computed using the pseudoinverse, denoted by $\cdot^\dagger$, as

$$\text{vec}(\boldsymbol{\mathcal{T}}) = \mathbf{A}^\dagger \mathbf{y} = \mathbf{P}^\dagger \mathbf{U}^\dagger \mathbf{y} = \mathbf{P}^\dagger (\mathbf{U}^\mathsf{T}\mathbf{U})^{-1} \mathbf{U}^\mathsf{T} \mathbf{y}.$$

$\boldsymbol{\mathcal{T}}$ is unique if $\mathbf{U}^\mathsf{T}\mathbf{U}$ has full rank, hence at least $M \geq F$. Using the definition of symmetric tensors, it is easy to show that $\text{range}(\mathbf{P}^\dagger)$ is the set of (vectorized) symmetric tensors. Therefore, $\boldsymbol{\mathcal{T}}$ is a symmetric tensor and its decomposition can be recovered using standard CPD algorithms. (Note that uniqueness of $\boldsymbol{\mathcal{T}}$ does not imply uniqueness of its decomposition.)

A.5 Experiments

We illustrate that few samples are required for multivariate regression using symmetric tensor decompositions on a materials science dataset. Both algorithms are implemented in Matlab 2016b using Tensorlab 3.0 [129].

The Gibbs free energy of the liquid phase of an Ag-Cu-Ni-Sn alloy can be modeled by

$$G(x_1, x_2, x_3) = G_{\mathrm{p}}(x_1, x_2, x_3) + \bar{R}T \sum_{i=1}^{4} x_i \log x_i,$$

in which G_{p} is an unknown polynomial, $x_i \in [0, 1]$ for $i = 1, \ldots, I$ are $I = 3$ independent molar fractions, $x_4 = 1 - x_1 - x_2 - x_3$, and $\bar{R}$ and T are known constants [163]–[165]. While only G_{p} is modeled in [163] as the logarithmic terms are known and can be subtracted first, we model both G_{p} and G.

First, to model G_{p} we use the feature vector $\mathbf{x} = [1, x_1, x_2, x_3]^{\mathrm{T}}$. The degree $N = 4$ and rank $R = 12$ have been determined by grid search and a validation set. Using these parameters, we perform 50 experiments in which we randomly sample M data points to compute the $\mathbf{B}$ and $\mathbf{c}$ using the algebraic and optimization algorithms. For the latter, we keep only the best result in terms of objective function value for five random initializations. We then compute the 99% quantile of the point-wise relative error for 166K validation points. In Fig. A.2, we report the median over all experiments. Both methods achieve a very high accuracy, even for a very low number of samples M. (There are $IR = 48$ free variables.) For $M < 35$, the algebraic method fails as $\mathbf{U}$ Eq. (A.11) is not of full column rank. If $M > 35$, the algebraic method is typically more accurate, but also more expensive.

Second, we also model G without subtracting the logarithmic terms and use $\mathbf{x} = [1, x_1, x_2, x_3, \log x_1, \log x_2, \log x_3, \log(1 - x_1 - x_2 - x_3)]^{\mathrm{T}}$ as feature vector. As before, a maximal degree $N = 3$ and $R = 20$ are determined via validation sets. In Fig. A.3, we show the error for two sampling strategies: unbiased and biased. In both cases points are sampled uniformly, but in the biased case 25% of the samples are 'boundary' points satisfying $x_i < 0.01$ for at least one $i = 1, 2, 3, 4$. Biased sampling typically improves the accuracy, as can be seen in Fig. A.3, which is expected as the logarithmic terms are large near $x = 0$. Compared to Fig. A.2, the accuracy is worse, which is expected, as $x \log x$ can be well approximated by low-degree polynomials in $x \in (0, 1)$, resulting in an ill-conditioned problem.

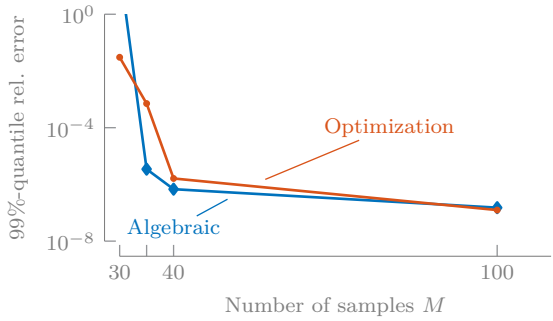


Figure A.2: A high accuracy is achieved by both methods when modeling G_p, even for few samples. The algebraic method fails for $M < 35$. Median over 50 experiments.

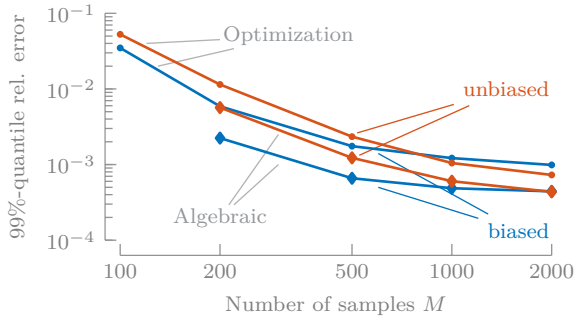


Figure A.3: Biased sampling improves accuracy when modeling G. The algebraic method is a bit more accurate, but it fails for $M < 120$. Median over 50 experiments.

A.6 Conclusion and further work

A multivariate polynomial can be represented by a *low-rank* tensor. This low-rank assumption allows efficient multivariate regression using few samples as illustrated for a materials science problem. By casting the problem as a linear system with a low-rank tensor constrained solution and exploiting the available structure, an efficient Gauss–Newton algorithm with a low time complexity can be derived. By exploiting the structure, the problem can be solved algebraically while lowering the number of required data points. While more expensive, it is often a good starting point for the optimization algorithm.

Future work includes the derivation of lower bounds on the number of samples via generic uniqueness conditions building upon [161]. Representing the coefficient tensor by other tensor decompositions which are often more suitable for compression, e.g., the multilinear singular value decomposition, is another interesting future direction.

Bibliography

[1] H. Fanaee-T and J. Gama, "Tensor-based anomaly detection: An interdisciplinary survey", *Knowledge-Based Systems*, vol. 98, pp. 130–147, 2016. DOI: https://doi.org/10.1016/j.knosys.2016.01.027.

[2] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications", *SIAM Review*, vol. 51, no. 3, pp. 455–500, Aug. 2009.

[3] A. Cichocki, D. P. Mandic, L. De Lathauwer, *et al.*, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis", *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, Mar. 2015.

[4] N. Sidiropoulos, L. De Lathauwer, X. Fu, *et al.*, "Tensor decomposition for signal processing and machine learning", *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, Jul. 2017.

[5] L.-H. Lim, "Tensors in computations", *Acta Numerica*, vol. 30, pp. 555–764, 2021. DOI: 10.1017/S0962492921000076.

[6] I. Domanov and L. De Lathauwer, "On the uniqueness of the canonical polyadic decomposition of third-order tensors — Part II: Uniqueness of the overall decomposition", *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 3, pp. 876–903, Jul. 2013. DOI: 10.1137/120877258.

[7] I. Domanov and L. De Lathauwer, "Canonical polyadic decomposition of third-order tensors: Relaxed uniqueness conditions and algebraic algorithm", *Linear Algebra and its Applications*, vol. 513, pp. 342–375, Jan. 2017.

[8] I. Domanov and L. De Lathauwer, "Generic uniqueness conditions for the canonical polyadic decomposition and INDSCAL", *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 4, pp. 1567–1589, Nov. 2015.

[9] M. Udell and A. Townsend, "Why are big data matrices approximately low rank?", *SIAM Journal on Mathematics of Data Science*, vol. 1, no. 1, pp. 144–160, 2019. DOI: 10.1137/18M1183480.

[10] M. Boussé, O. Debals, and L. De Lathauwer, "A tensor-based method for large-scale blind source separation using segmentation", *IEEE Transactions on Signal Processing*, vol. 65, no. 2, pp. 346–358, Jan. 2017.

[11] O. Debals, "Tensorization and applications in blind source separation", Ph.D. dissertation, KU Leuven, Aug. 2017.

[12] J. Bennett, S. Lanning, *et al.*, "The Netflix prize", in *Proceedings of KDD cup and workshop*, New York, vol. 2007, 2007, p. 35.

[13] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer, "Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis", *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 71–79, Sep. 2014.

[14] I. Oseledets and E. Tyrtyshnikov, "TT-cross approximation for multidimensional arrays", *Linear Algebra and its Applications*, vol. 432, no. 1, pp. 70–88, Jan. 2010.

[15] N. Vervliet and L. De Lathauwer, "A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors", *IEEE Journal on Selected Topics in Signal Processing*, vol. 10, pp. 284–295, 2 2016.

[16] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Morup, "Scalable tensor factorizations for incomplete data", *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41–56, Mar. 2011.

[17] C. F. Caiafa and A. Cichocki, "Generalizing the column–row matrix decomposition to multi-way arrays", *Linear Algebra and its Applications*, vol. 433, no. 3, pp. 557–573, Sep. 2010. DOI: 10.1016/j.laa.2010.03.020.

[18] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Techniques, applications, and challenges", in *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira, Eds. New York, NY: Springer US, 2022, pp. 1–35. DOI: 10.1007/978-1-0716-2197-4_1.

[19] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 208–220, Jan. 2013.

[20] H. Shen, X. Li, Q. Cheng, *et al.*, "Missing information reconstruction of remote sensing data: A technical review", *IEEE Geoscience and Remote Sensing Magazine*, vol. 3, no. 3, pp. 61–85, 2015. DOI: 10.1109/MGRS.2015.2441912.

[21] Q. Song, H. Ge, J. Caverlee, and X. Hu, "Tensor completion algorithms in big data analytics", *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 1, Jan. 2019. DOI: 10.1145/3278607.

[22] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization", *Inverse Problems*, vol. 27, no. 2, 025010 (19pp), 2011.

[23] M. Signoretto, Q. Tran Dinh, L. De Lathauwer, and J. A. Suykens, "Learning with tensors: A framework based on convex optimization and spectral regularization", *Machine Learning*, vol. 94, pp. 303–351, 2014.

[24] E. J. Candés and Y. Plan, "Matrix completion with noise", *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, Jun. 2010.

[25] D. Kressner, M. Steinlechner, and B. Vandereycken, "Low-rank tensor completion by Riemannian optimization", *BIT Numerical Mathematics*, vol. 54, no. 2, pp. 447–468, Nov. 2013.

[26] G. Tomasi and R. Bro, "PARAFAC and missing values", *Chemometrics and Intelligent Laboratory Systems*, vol. 75, no. 2, pp. 163–180, Feb. 2005.

[27] D. L. Donoho, "Compressed sensing", *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[28] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information", *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006. DOI: 10.1109/TIT.2005.862083.

[29] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?", *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006. DOI: 10.1109/TIT.2006.885507.

[30] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling", *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, Mar. 2008.

[31] G. Huang, H. Jiang, K. Matthews, and P. Wilford, "Lensless imaging by compressive sensing", in *2013 IEEE International Conference on Image Processing*, 2013, pp. 2101–2105. DOI: 10.1109/ICIP.2013.6738433.

[32] G. Satat, M. Tancik, and R. Raskar, "Lensless imaging with compressive ultrafast sensing", *IEEE Transactions on Computational Imaging*, vol. 3, no. 3, pp. 398–407, 2017. DOI: 10.1109/TCI.2017.2684624.

[33] J. Wu, H. Zhang, W. Zhang, *et al.*, "Single-shot lensless imaging with fresnel zone aperture and incoherent illumination", *Light: Science & Applications*, vol. 9, no. 1, p. 53, Apr. 2020. DOI: 10.1038/s41377-020-0289-9.

[34] M. F. Duarte, M. A. Davenport, D. Takhar, *et al.*, "Single-pixel imaging via compressive sampling", *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83–91, Mar. 2008. DOI: 10.1109/msp.2007.914730.

[35] V. Boominathan, J. T. Robinson, L. Waller, and A. Veeraraghavan, "Recent advances in lensless imaging", *Optica*, vol. 9, no. 1, pp. 1–16, Jan. 2022. DOI: 10.1364/OPTICA.431361.

[36] S. Qaisar, R. M. Bilal, W. Iqbal, M. Naureen, and S. Lee, "Compressive sensing: From theory to applications, a survey", *Journal of Communications and Networks*, vol. 15, no. 5, pp. 443–456, 2013. DOI: 10.1109/JCN.2013.000083.

[37] A. C. Fannjiang, T. Strohmer, and P. Yan, "Compressed remote sensing of sparse objects", *SIAM Journal on Imaging Sciences*, vol. 3, no. 3, pp. 595–618, 2010. DOI: 10.1137/090757034. eprint: https://doi.org/10.1137/090757034.

[38] M. F. Duarte and Y. C. Eldar, "Structured compressed sensing: From theory to applications", *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4053–4085, 2011. DOI: 10.1109/TSP.2011.2161982.

[39] M. F. Duarte and R. G. Baraniuk, "Kronecker compressive sensing", *IEEE Transactions on Image Processing*, vol. 21, no. 2, pp. 494–504, 2012. DOI: 10.1109/TIP.2011.2165289.

[40] C. F. Caiafa and A. Cichocki, "Multidimensional compressed sensing and their applications", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 6, pp. 355–380, Nov. 2013. DOI: 10.1002/widm.1108.

[41] C. F. Caiafa and A. Cichocki, "Computing sparse representations of multidimensional signals using kronecker bases", *Neural Computation*, vol. 25, no. 1, pp. 186–220, 2013. DOI: 10.1162/NECO_a_00385.

[42] D. C. Araújo, A. L. F. de Almeida, J. P. C. L. Da Costa, and R. T. de Sousa, "Tensor-based channel estimation for massive MIMO-OFDM systems", *IEEE Access*, vol. 7, pp. 42 133–42 147, 2019. DOI: 10.1109/ACCESS.2019.2908207.

[43] N. D. Sidiropoulos and A. Kyrillidis, "Multi-way compressed sensing for sparse low-rank tensors", *IEEE Signal Processing Letters*, vol. 19, no. 11, pp. 757–760, Jan. 2012.

[44] B. Huang, C. Mu, D. Goldfarb, and J. Wright, "Provable low-rank tensor recovery", *Optimization-Online*, vol. 4252, no. 2, pp. 455–500, 2014.

[45] D. Kressner and C. Tobler, "Low-rank tensor krylov subspace methods for parametrized linear systems", *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 4, pp. 1288–1316, 2011. DOI: 10.1137/100799010. eprint: https://doi.org/10.1137/100799010.

[46] J. Ballani, "Fast evaluation of singular bem integrals based on tensor approximations", *Numerische Mathematik*, vol. 121, no. 3, pp. 433–460, Jul. 2012. DOI: 10.1007/s00211-011-0436-6.

[47] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques", *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, Feb. 2013.

[48] S. N. Goodman, D. Fanelli, and J. P. A. Ioannidis, "What does research reproducibility mean?", *Science Translational Medicine*, vol. 8, no. 341, 341ps12–341ps12, 2016. DOI: 10.1126/scitranslmed.aaf5027.

[49] O. S. Collaboration, "Estimating the reproducibility of psychological science", *Science*, vol. 349, no. 6251, aac4716, 2015. DOI: 10.1126/science.aac4716.

[50] T. M. Errington, M. Mathur, C. K. Soderberg, *et al.*, "Investigating the replicability of preclinical cancer biology", *eLife*, vol. 10, R. Pasqualini and E. Franco, Eds., e71601, Dec. 2021. DOI: 10.7554/eLife.71601.

[51] L. D. Nelson, J. Simmons, and U. Simonsohn, "Psychology's renaissance", *Annual Review of Psychology*, vol. 69, no. 1, pp. 511–534, 2018, PMID: 29068778. DOI: 10.1146/annurev-psych-122216-011836.

[52] M. Baker, "Reproducibility crisis: Blame it on the antibodies", *Nature*, vol. 521, no. 7552, pp. 274–276, May 2015. DOI: 10.1038/521274a.

[53] D. Fanelli, "Is science really facing a reproducibility crisis, and do we need it to?", *Proceedings of the National Academy of Sciences*, vol. 115, no. 11, pp. 2628–2631, 2018. DOI: 10.1073/pnas.1708272114.

[54] D. T. Gilbert, G. King, S. Pettigrew, and T. D. Wilson, "Comment on "estimating the reproducibility of psychological science"", *Science*, vol. 351, no. 6277, pp. 1037–1037, 2016. DOI: 10.1126/science.aad7243. eprint: https://www.science.org/doi/pdf/10.1126/science.aad7243.

[55] C. J. Anderson, Š. Bahník, M. Barnett-Cowan, *et al.*, "Response to comment on "estimating the reproducibility of psychological science"", *Science*, vol. 351, no. 6277, pp. 1037–1037, 2016. DOI: 10.1126/science.aad9163. eprint: https://www.science.org/doi/pdf/10.1126/science.aad9163.

[56] J. Pineau, P. Vincent-Lamarre, K. Sinha, *et al.*, "Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program)", *Journal of Machine Learning Research*, vol. 22, no. 1, Jan. 2021.

[57] T. Adali, F. Kantar, M. A. B. S. Akhonda, *et al.*, "Reproducibility in matrix and tensor decompositions: Focus on model match, interpretability, and uniqueness", *IEEE Signal Processing Magazine*, vol. 39, no. 4, pp. 8–24, 2022. DOI: 10.1109/MSP.2022.3163870.

[58] M. Sørensen and L. De Lathauwer, "Fiber sampling approach to canonical polyadic decomposition and application to tensor completion", *SIAM Journal on Matrix Analysis and Applications*, vol. 40, no. 3, pp. 888–917, 2019.

[59] E. Candès and B. Recht, "Exact matrix completion via convex optimization", *Communications of the ACM*, vol. 55, no. 6, pp. 111–119, Jun. 2012. DOI: 10.1145/2184319.2184343.

[60] C. Mu, B. Huang, J. Wright, and D. Goldfarb, "Square deal lower bounds and improved relaxations for tensor recovery.", in *Proceedings of the 31st International Conference on Machine Learning (ICML 2014, Beijing, China)*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Jun. 2014, pp. 73–81.

[61] R. Ge, J. D. Lee, and T. Ma, "Matrix completion has no spurious local minimum", in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, Curran Associates, Inc., 2016.

[62] N. Sidiropoulos, E. E. Papalexakis, and C. Faloutsos, "Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition", *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 57–70, Sep. 2014. DOI: 10.1109/MSP.2014.2329196.

[63] R. Bro and C. Andersson, "Improving the speed of multiway algorithms: Part II: Compression", *Chemometrics and Intelligent Laboratory Systems*, vol. 42, no. 12, pp. 105–113, 1998.

[64] P. M. Kroonenberg, "Three-mode principal component analysis theory and applications", Ph.D. dissertation, Leiden University, 1983.

[65] P. Kroonenberg, *Applied Multiway Data Analysis*. John Wiley & Sons, 2008.

[66] W. Hackbusch, *Tensor spaces and numerical tensor calculus* (Springer Series in Computational Mathematics). Heidelberg: Springer, 2012, vol. 42, pp. xxiv, 500. DOI: 10.1007/978-3-642-28027-6.

[67] A. Smilde, R. Bro, P. Geladi, and J. Wiley, *Multi-way analysis with applications in the chemical sciences*. Wiley Chichester, UK, 2004.

[68] F. Cong, Q.-H. Lin, L.-D. Kuang, *et al.*, "Tensor decomposition of eeg signals: A brief review", *Journal of Neuroscience Methods*, vol. 248, pp. 59–69, 2015. DOI: https://doi.org/10.1016/j.jneumeth.2015.03.018.

[69] B. Hunyadi, S. Van Huffel, and M. De Vos, "The power of tensor decompositions in biomedical applications", in *Machine Learning for Healthcare Technologies*, D. A. Clifton, Ed., The Institution of Engineering and Technology (IET, London, UK), 2016.

[70] Y. Panagakis, J. Kossaifi, G. G. Chrysos, *et al.*, "Tensor methods in computer vision and deep learning", *Proceedings of the IEEE*, vol. 109, no. 5, pp. 863–890, 2021. DOI: 10.1109/JPROC.2021.3074329.

[71] J. B. Kruskal, "Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics", *Linear Algebra and its Applications*, vol. 18, no. 2, pp. 95–138, 1977. DOI: 10.1016/0024-3795(77)90069-6.

[72] N. D. Sidiropoulos and R. Bro, "On the uniqueness of multilinear decomposition of N-way arrays", *Journal of Chemometrics*, vol. 14, no. 3, pp. 229–239, May 2000. DOI: 10.1002/1099-128X(200005/06)14:3<229::AID-CEM587>3.0.CO;2-N.

[73] V. de Silva and L.-H. Lim, "Tensor rank and the ill-posedness of the best low-rank approximation problem", *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1084–1127, Sep. 2008. DOI: 10.1137/06066518X.

[74] C. Beltrán, P. Breiding, and N. Vannieuwenhoven, "Pencil-based algorithms for tensor rank decomposition are not stable", *SIAM Journal on Matrix Analysis and Applications*, vol. 40, no. 2, pp. 739–773, Jan. 2019. DOI: 10.1137/18m1200531.

[75] P. Breiding and N. Vannieuwenhoven, "The condition number of join decompositions", *SIAM Journal on Matrix Analysis and Applications*, vol. 39, no. 1, pp. 287–309, 2018. DOI: 10.1137/17M1142880.

[76] E. Evert and L. De Lathauwer, "Guarantees for existence of a best canonical polyadic approximation of a noisy low-rank tensor", *SIAM Journal On Matrix Analysis And Applications*, vol. 43, no. 1, pp. 328–369, Mar. 2022. DOI: 10.1137/20M1381046.

[77] R. A. Harshman, "Determination and proof of minimum uniqueness conditions for PARAFAC1", *UCLA Working Papers in Phonetics*, vol. 22, pp. 111–117, 1972.

[78] E. Evert, M. Vandecappelle, and L. De Lathauwer, "A recursive eigenspace computation for the canonical polyadic decomposition", *SIAM Journal On Matrix Analysis And Applications*, vol. 43, no. 1, pp. 274–300, Feb. 2022. DOI: 10.1137/21M1423026.

[79] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization", *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 642–666, Sep. 2006. DOI: 10.1137/040608830.

[80] I. Domanov and L. De Lathauwer, "Canonical polyadic decomposition of third-order tensors: Reduction to generalized eigenvalue decomposition", *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 2, pp. 636–660, May 2014. DOI: 10.1137/130916084.

[81] I. Domanov and L. De Lathauwer, "Canonical polyadic decomposition of third-order tensors Relaxed uniqueness conditions and algebraic algorithm", *Linear Algebra and its Applications*, vol. 513, pp. 342–375, 2017. DOI: httpdx.doi.org10.1016j.laa.2016.10.019.

[82] G. Tomasi and R. Bro, "A comparison of algorithms for fitting the PARAFAC model", *Computational Statistics and Data Analysis*, vol. 50, no. 7, pp. 1700–1734, Apr. 2006.

[83] E. Acar, D. Dunlavy, and T. Kolda, "A scalable optimization approach for fitting canonical tensor decompositions", *Journal of Chemometrics*, vol. 25, no. 2, pp. 67–86, 2011.

[84] A.-H. Phan, P. Tichavský, and A. Cichocki, "Low complexity damped Gauss–Newton algorithms for CANDECOMP/PARAFAC", *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 126–147, Jan. 2013. DOI: 10.1137/100808034.

[85] E. Acar, T. G. Kolda, and D. M. Dunlavy, *All-at-once optimization for coupled matrix and tensor factorizations*, 2011. arXiv: 1105.3422 [math.NA].

[86] N. Vervliet and L. De Lathauwer, "Numerical optimization based algorithms for data fusion", in *Data Fusion Methodology and Applications*, M. Cocchi, Ed., vol. 31, Elsevier, 2019, ch. 4, pp. 81–128.

[87] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition", *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, Apr. 2000.

[88] L. R. Tucker, "The extension of factor analysis to three-dimensional matrices", in *Contributions to mathematical psychology*, H. Gulliksen and N. Frederiksen, Eds., New York: Holt, Rinehart & Winston, 1964, pp. 110–127.

[89] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, "A new truncation strategy for the higher-order singular value decomposition", *SIAM Journal on Scientific Computing*, vol. 34, no. 2, A1027–A1052, Jan. 2012.

[90] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions", *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011. DOI: 10.1137/090771806.

[91] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-$(R_1, R_2, \cdots, R_N)$ approximation of higher-order tensors", *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, Jul. 2000.

[92] M. Ishteva, P.-A. Absil, S. Van Huffel, and L. De Lathauwer, "Best low multilinear rank approximation of higher-order tensors, based on Riemannian trust-region scheme", *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 1, pp. 115–135, Jan. 2011.

[93] B. Savas and L.-H. Lim, "Quasi-Newton methods on Grassmannians and multilinear approximations of tensors", *SIAM Journal on Scientific Computing*, vol. 32, no. 6, pp. 3352–3393, 2010. DOI: 10.1137/090763172. eprint: http://dx.doi.org/10.1137/090763172.

[94] I. V. Oseledets and E. E. Tyrtyshnikov, "Breaking the curse of dimensionality, or how to use SVD in many dimensions", *SIAM Journal on Scientific Computing*, vol. 31, no. 5, pp. 3744–3759, 2009. DOI: 10.1137/090748330. eprint: https://doi.org/10.1137/090748330.

[95] I. Oseledets, "Tensor-train decomposition", *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, Sep. 2011.

[96] L. Grasedyck, "Hierarchical singular value decomposition of tensors", *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 4, pp. 2029–2054, Jan. 2010. DOI: 10.1137/090764189.

[97] R. Orús, "Tensor networks for complex quantum systems", *Nature Reviews Physics*, vol. 1, no. 9, pp. 538–550, Sep. 2019. DOI: 10.1038/s42254-019-0086-7.

[98] S. Hendrikx, M. Sørensen, and L. De Lathauwer, "Multilinear singular value decomposition based completion with fibers observed in a single mode", *Technical Report 23-142, ESAT-STADIUS, KU Leuven, Leuven, Belgium*, 2023, Submitted to SIAM Journal on Matrix Analysis and Applications.

[99] V. Khoromskaia and B. N. Khoromskij, *Tensor Numerical Methods in Quantum Chemistry*. Berlin, Boston: De Gruyter, 2018. DOI: doi:10.1515/9783110365832.

[100] C. A. Andersson and R. Bro, "Improving the speed of multi-way algorithms: Part I: Tucker3", *Chemometrics and Intelligent Laboratory Systems*, vol. 42, no. 1–2, pp. 93–103, 1998.

[101] H. Kasai and B. Mishra, "Low-rank tensor completion: A riemannian manifold preconditioning approach", in *Proceedings of The 33rd International Conference on Machine Learning*, New York, New York, USA: PMLR, 2016, pp. 1012–1021.

[102] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering", in *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys 2010, Barcelona, Spain)*, ACM, Sep. 2010, pp. 79–86.

[103] M. Filipović and A. Jukić, "Tucker factorization with missing data with application to low-n-rank tensor completion", *Multidimensional Systems and Signal Processing*, vol. 26, pp. 677–692, 3 Jul. 2015. DOI: 10.1007/s11045-013-0269-9.

[104] P. Breiding, F. Gesmundo, M. Michałek, and N. Vannieuwenhoven, "Algebraic compressed sensing", *Applied and Computational Harmonic Analysis*, vol. 65, pp. 374–406, 2023. DOI: 10.1016/j.acha.2023.03.006.

[105] X. Chen, Y. Chen, and Z. He, *Urban Traffic Speed Dataset of Guangzhou, China*, Mar. 2018. DOI: 10.5281/zenodo.1205229.

[106] D. L. Pimentel-Alarcón, N. Boston, and R. D. Nowak, "Deterministic conditions for subspace identifiability from incomplete sampling", in *2015 IEEE International Symposium on Information Theory (ISIT)*, 2015, pp. 2191–2195. DOI: 10.1109/ISIT.2015.7282844.

[107] F. J. Király and R. Tomioka, "A combinatorial algebraic approach for the identifiability of low-rank matrix completion", in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, icml.cc / Omnipress, 2012.

[108] S. Bhojanapalli and P. Jain, "Universal matrix completion", in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: PMLR, Jul. 2014, pp. 1881–1889.

[109] D. Cheng, N. Ruchansky, and Y. Liu, "Matrix completability analysis via graph k-connectivity", in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, A. Storkey and F. Perez-Cruz, Eds., ser. Proceedings of Machine Learning Research, vol. 84, PMLR, Apr. 2018, pp. 395–403.

[110] D. L. Pimentel-Alarcón, N. Boston, and R. D. Nowak, "A characterization of deterministic sampling patterns for low-rank matrix completion", *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 623–636, 2016. DOI: 10.1109/JSTSP.2016.2537145.

[111] A. Singer and M. Cucuringu, "Uniqueness of low-rank matrix completion by rigidity theory", *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 4, pp. 1621–1641, 2010. DOI: 10.1137/090750688.

[112] M. Ashraphijuo, V. Aggarwal, and X. Wang, "Deterministic and probabilistic conditions for finite completability of low-tucker-rank tensor", *IEEE Transactions on Information Theory*, vol. 65, no. 9, pp. 5380–5400, 2019. DOI: 10.1109/TIT.2019.2919568.

[113] D. L. Pimentel-Alarcón and R. D. Nowak, "Adaptive strategy for restricted-sampling noisy low-rank matrix completion", in *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, IEEE, 2015, pp. 429–432. DOI: 10.1109/CAMSAP.2015.7383828.

[114] D. W. Jacobs, "Linear fitting with missing data for structure-from-motion", *Computer Vision and Image Understanding*, vol. 82, no. 1, pp. 57–81, 2001. DOI: https://doi.org/10.1006/cviu.2001.0906.

[115] X. Chen, Z. He, and L. Sun, "A bayesian tensor decomposition approach for spatiotemporal traffic data imputation", *Transportation Research Part C: Emerging Technologies*, vol. 98, pp. 73–84, 2019. DOI: 10.1016/j.trc.2018.11.003.

[116] S. Hendrikx and L. De Lathauwer, "Block row Kronecker-structured linear systems with a low-rank tensor solution", *Frontiers in Applied Mathematics and Statistics*, vol. 8, pp. 41–57, Mar. 2022, Special Issue on High-Performance Tensor Computations in Scientific Computing and Data Science. DOI: 10.3389/fams.2022.832883.

[117] S. Ahmadi-Asl, S. Abukhovich, M. G. Asante-Mensah, *et al.*, "Randomized algorithms for computation of Tucker decomposition and higher order SVD (HOSVD)", *IEEE Access*, vol. 9, pp. 28 684–28 706, 2021. DOI: 10.1109/ACCESS.2021.3058103.

[118] A. Aldroubi and K. Gröchenig, "Nonuniform sampling and reconstruction in shift-invariant spaces", *SIAM Review*, vol. 43, no. 4, pp. 585–620, 2001. DOI: 10.1137/S0036144501386986.

[119] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling", *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010. DOI: 10.1109/JPROC.2010.2040551.

[120] D. Kressner and C. Tobler, "Low-rank tensor Krylov subspace methods for parametrized linear systems", *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 4, pp. 1288–1316, Oct. 2011. DOI: 10.1137/100799010.

[121] C. Battaglino, G. Ballard, and T. G. Kolda, "A practical randomized CP tensor decomposition", *SIAM Journal on Matrix Analysis and Applications*, vol. 39, no. 2, pp. 876–901, 2018. DOI: 10 . 1137 / 17M1112303.

[122] M. Che, Y. Wei, and H. Yan, "Randomized algorithms for the low multilinear rank approximations of tensors", *Journal of Computational and Applied Mathematics*, vol. 390, p. 113 380, 2021. DOI: https://doi.org/10.1016/j.cam.2020.113380.

[123] G. Zhou, A. Cichocki, and S. Xie, "Decomposition of big tensors with low multilinear rank", *CoRR*, vol. abs/1412.1885, 2014. arXiv: 1412.1885.

[124] B. Yang, A. Zamzam, and N. D. Sidiropoulos, "Parasketch: Parallel tensor factorization via sketching", in *Proceedings of the 2018 SIAM International Conference on Data Mining (SDM)*, 2018, pp. 396–404. DOI: 10.1137/1.9781611975321.45.

[125] R. Jin, T. G. Kolda, and R. Ward, "Faster johnson–lindenstrauss transforms via kronecker products", *Information and Inference: A Journal of the IMA*, vol. 10, no. 4, pp. 1533–1562, Oct. 2020. DOI: 10.1093/imaiai/iaaa028.

[126] M. Mahoney, M. Maggioni, and P. Drineas, "Tensor-CUR decompositions for tensor-based data", *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 957–987, 2008. DOI: 10.1137/060665336.

[127] I. Oseledets, D. V. Savostianov, and E. Tyrtyshnikov, "Tucker dimensionality reduction of three-dimensional arrays in linear time", *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 939–956, 2008.

[128] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, "A theory of pseudoskeleton approximations", *Linear Algebra and its Applications*, vol. 261, no. 1, pp. 1–21, 1997. DOI: 10.1016/S0024-3795(96)00301-1.

[129] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, *Tensorlab 3.0*, Available online at https://www.tensorlab.net, Mar. 2016.

[130] L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion", *IEEE Journal on Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 586–600, Jun. 2015. DOI: 10.1109/JSTSP.2015.2400415.

[131] I. Domanov and L. De Lathauwer, "On the uniqueness of the canonical polyadic decomposition of third-order tensors — Part I: Basic results and uniqueness of one factor matrix", *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 3, pp. 855–875, Jul. 2013. DOI: 10.1137/120877234.

[132] L. Chiantini and G. Ottaviani, "On generic identifiability of 3-tensors of small rank", *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 3, pp. 1018–1037, 2012.

[133] N. Vervliet, O. Debals, and L. De Lathauwer, "Tensorlab 3.0 — Numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization", in *Proceedings of the 50th Asilomar Conference on Signals, Systems and Computers (Pacific Grove, California, USA)*, Nov. 2016, pp. 1733–1738.

[134] M. Che and Y. Wei, "Randomized algorithms for the approximations of tucker and the tensor train decompositions", *Advances in Computational Mathematics*, vol. 45, no. 1, pp. 395–428, Feb. 2019. DOI: 10.1007/s10444-018-9622-8.

[135] D. Savostyanov and I. Oseledets, "Fast adaptive interpolation of multi-dimensional arrays in tensor train format", in *The 2011 International Workshop on Multidimensional (nD) Systems*, 2011, pp. 1–8. DOI: 10.1109/nDS.2011.6076873.

[136] A. A. Wagadarikar, N. P. Pitsianis, X. Sun, and D. J. Brady, "Video rate spectral imaging using a coded aperture snapshot spectral imager", *Optics Express*, vol. 17, no. 8, pp. 6368–6388, Apr. 2009. DOI: 10.1364/OE.17.006368.

[137] Y. Rivenson and A. Stern, "Compressed imaging with a separable sensing operator", *IEEE Signal Processing Letters*, vol. 16, no. 6, pp. 449–452, 2009. DOI: 10.1109/LSP.2009.2017817.

[138] E. V. den berg and M. P. Friedlander, "Probing the pareto frontier for basis pursuit solutions", *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008. DOI: 10.1137/080714488.

[139] E. V. den berg and M. P. Friedlander, "SPGL1: A solver for large-scale sparse reconstruction", Dec. 2019, Available online at https://friedlander.io/spgl1.

[140] X. Yuan, "Generalized alternating projection based total variation minimization for compressive sensing", in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 2539–2543. DOI: 10.1109/ICIP.2016.7532817.

[141] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, *220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3*, Sep. 2015. DOI: 10.4231/R7RX991C.

[142] S. Hendrikx, R. Widdershoven, N. Vervliet, and L. De Lathauwer, "Tensorlab$^+$: A case study on reproducibility in tensor research", *Technical Report 22–116, ESAT-STADIUS, KU Leuven, Leuven, Belgium*, 2022, Accepted for publication in IEEE Computing in Science and Engineering.

[143] National Academies of Sciences, Engineering, and Medicine, *Reproducibility and Replicability in Science*. Washington, DC: The National Academies Press, 2019. DOI: 10.17226/25303.

[144] S. Fomel and J. F. Claerbout, "Guest editors' introduction: Reproducible research", *Computing in Science & Engineering*, vol. 11, no. 1, pp. 5–7, 2009. DOI: 10.1109/MCSE.2009.14.

[145] M. Schwab, N. Karrenbach, and J. Claerbout, "Making scientific computations reproducible", *Computing in Science & Engineering*, vol. 2, no. 6, pp. 61–67, 2000. DOI: 10.1109/5992.881708.

[146] M. Baker, "Reproducibility crisis", *Nature*, vol. 533, no. 26, pp. 353–66, 2016. DOI: 10.1038/533452a.

[147] V. Stodden, "The scientific method in practice: Reproducibility in the computational sciences", Feb. 2010, MIT Sloan Research Paper No. 4773-10. DOI: 10.2139/ssrn.1550193.

[148] S. Hendrikx, M. Boussé, N. Vervliet, *et al.*, *Tensorlab$^+$*, Available online, Version of Feb 2022 downloaded from `https://www.tensorlabplus.net`.

[149] B. N. Khoromskij, *Tensor Numerical Methods in Scientific Computing*. Berlin/Boston: De Gruyter, Jun. 2018.

[150] S. Hendrikx, N. Vervliet, M. Boussé, and L. De Lathauwer, "Tensorgebaseerde patroonherkenning, data-analyse en machinaal leren", (in Dutch), M.S. thesis, Department of Electrical Engineering, KU Leuven, 2019.

[151] M. Vandecappelle, N. Vervliet, and L. De Lathauwer, "Nonlinear least squares updating of the canonical polyadic decomposition", in *Proceedings of the 25th European Signal Processing bousse2017weightedConference (EUSIPCO 2017, Kos, Greece)*, Sep. 2017, pp. 693–697.

[152] M. Vandecappelle and L. De Lathauwer, "Low multilinear rank updating", in *Proceedings of the 53rd Asilomar Conference on Signals, Systems and Computers (Pacific Grove, California, USA)*, Nov. 2019, pp. 437–441.

[153] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms — Part II: Definitions and uniqueness", *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1033–1066, Sep. 2008.

[154] S. Hendrikx, M. Boussé, N. Vervliet, and L. De Lathauwer, "Algebraic and optimization based algorithms for multivariate regression using symmetric tensor decomposition", in *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Le gosier, Guadeloupe, Dec. 2019, pp. 475–479.

[155] H. Zhou, L. Li, and H. Zhu, "Tensor regression with applications in neuroimaging data analysis", *Journal of the American Statistical Association*, vol. 108, no. 502, 2013.

[156] A. Novikov, M. Trofimov, and I. Oseledets, "Exponential machines", *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 66, no. 6, pp. 789–797, 2018.

[157] Z. Chen, K. Batselier, J. A. K. Suykens, and N. Wong, "Parallelized tensor train learning of polynomial classifiers", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4621–4632, 2018.

[158] S. Rendle, "Factorization machines", in *ICDM '10 Proceedings of the 2010 IEEE International Conference on Data Mining*, 2010, pp. 995–1000.

[159] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata, "Higher-order factorization machines", in *Advances in Neural Information Processing Systems*, 2016, pp. 3351–3359.

[160] J. Yang and A. Gittens, "Tensor machines for learning target-specific polynomial features", 2015.

[161] M. Boussé, N. Vervliet, I. Domanov, O. Debals, and L. De Lathauwer, "Linear systems with a canonical polyadic decomposition constrained solution: Algorithms and applications", *Numerical Linear Algebra with Applications*, vol. 25, no. 6, e2190, Aug. 2018.

[162] J. Nocedal and S. Wright, *Numerical optimization*. Springer New York, Jul. 2006.

[163] N. Vervliet, O. Debals, and L. De Lathauwer, "Canonical polyadic decomposition of incomplete tensors with linearly constrained factors", *Technical Report 16—172, ESAT-STADIUS, KU Leuven, Leuven, Belgium*, 2017.

[164] A. Kroupa, "Modelling of phase diagrams and thermodynamic properties using Calphad method — Development of thermodynamic databases", *Computational Materials Science*, vol. 66, pp. 3–13, 2013.

[165] H. L. Lukas, S. G. Fries, and B. Sundman, *Computational thermodynamics: The Calphad method*. Cambridge university press, 2007.

[166] S. Hendrikx, N. Vervliet, M. Boussé, and L. De Lathauwer, "Thesis abstract: Tensor-based pattern recognition, data analysis and learning", in *BNAIC/BeneLearn 2020: Proceedings*, Leiden, The Netherlands, Nov. 2020, pp. 416–417.

[167] S. Hendrikx, M. Sørensen, and L. De Lathauwer, "Multilinear singular value decomposition of a tensor with fibers observed along one mode", in *2023 IEEE Statistical Signal Processing Workshop (SSP)*, Hanoi, Vietnam, Jul. 2023, pp. 566–570. DOI: 10.1109/SSP53291.2023.10207967.

List of publications

Journal papers

1. S. Hendrikx and L. De Lathauwer, "Block row Kronecker-structured linear systems with a low-rank tensor solution", *Frontiers in Applied Mathematics and Statistics*, vol. 8, pp. 41–57, Mar. 2022, Special Issue on High-Performance Tensor Computations in Scientific Computing and Data Science. DOI: 10.3389/fams.2022.832883

2. S. Hendrikx, R. Widdershoven, N. Vervliet, and L. De Lathauwer, "Tensorlab$^+$: A case study on reproducibility in tensor research", *Technical Report 22–116, ESAT-STADIUS, KU Leuven, Leuven, Belgium*, 2022, Accepted for publication in IEEE Computing in Science and Engineering

Reports

1. S. Hendrikx, M. Sørensen, and L. De Lathauwer, "Multilinear singular value decomposition based completion with fibers observed in a single mode", *Technical Report 23-142, ESAT-STADIUS, KU Leuven, Leuven, Belgium*, 2023, Submitted to SIAM Journal on Matrix Analysis and Applications

Conference papers

1. S. Hendrikx, M. Boussé, N. Vervliet, and L. De Lathauwer, "Algebraic and optimization based algorithms for multivariate regression using symmetric tensor decomposition", in *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Le gosier, Guadeloupe, Dec. 2019, pp. 475–479

2. S. Hendrikx, N. Vervliet, M. Boussé, and L. De Lathauwer, "Thesis abstract: Tensor-based pattern recognition, data analysis and learning", in *BNAIC/BeneLearn 2020: Proceedings*, Leiden, The Netherlands, Nov. 2020, pp. 416–417

3. S. Hendrikx, M. Sørensen, and L. De Lathauwer, "Multilinear singular value decomposition of a tensor with fibers observed along one mode", in

2023 IEEE Statistical Signal Processing Workshop (SSP), Hanoi, Vietnam, Jul. 2023, pp. 566–570. DOI: 10.1109/SSP53291.2023.10207967

Other

1. S. Hendrikx, M. Boussé, N. Vervliet, M. Vandecappelle, R. Kenis, and L. De Lathauwer, *Tensorlab$^+$*, Available online, Version of Feb 2022 downloaded from https://www.tensorlabplus.net



FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
STADIUS CENTER FOR DYNAMICAL SYSTEMS, SIGNAL PROCESSING AND DATA ANALYTICS
Kasteelpark Arenberg 10 - box 2446
B-3001 Leuven
stijn.hendrikx@kuleuven.be
www.esat.kuleuven.be/stadius