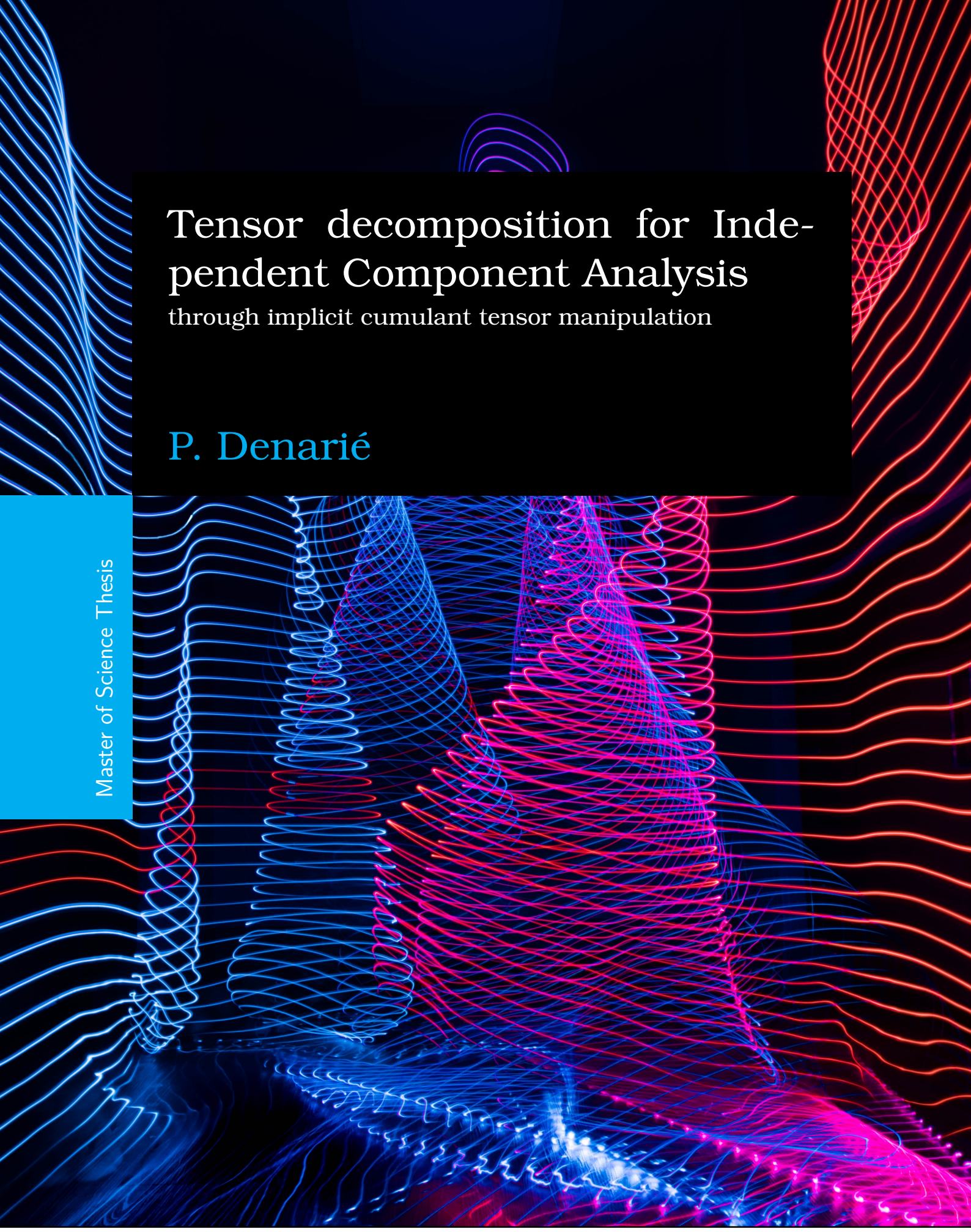
Tensor decomposition for Independent Component Analysis

through implicit cumulant tensor manipulation

P. Denarié

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Tensor decomposition for Independent Component Analysis

through implicit cumulant tensor manipulation

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

P. Denarié

November 25, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Blind Source Separation (BSS), the separation of latent source components from observed mixtures, is relevant to many fields of expertise such as neuro-imaging, economics and machine learning. Reliable estimates of the sources can be obtained through diagonalization of the cumulant tensor, i.e., a fourth-order symmetric multi-linear array containing the cross-kurtosis values of observed mixtures. The downside of such diagonalization methods is that they scale quartically with the increase of the amount of source components to estimate due to the tensor's quartic size. Tensor decomposition can simultaneously diagonalize the cumulant tensor and address its size. However, it does not resolve the scalability issue due to the restriction of having to first explicitly compute the tensor.

It is studied how decomposing the cumulant tensor in implicit fashion can be used to solve the BSS problem while simultaneously addressing its scalablity issue. A class of implicit cumulant tensor decomposition algorithms is derived which scale more favorably than their explicit counterparts in terms of either computational cost, storage cost or both. Firstly, a novel QR-Tensor algorithm (QRT) is introduced which allows for the simultaneous diagonalization of a tensor's outer-slices. It is theoretically shown how an implicit version of the QRT algorithm can be used to solve the BSS problem at a linearly scaling computational cost. Secondly, a fixed-point Canonical Polyadic Decomposition (CPD) iteration method is presented. It reduces the computational complexity from a quartic dependence to a linear dependence on the amount of signals to estimate. The source estimation performance of the devised implicit decomposition methods is compared to that of the state-of-the-art FastICA for an artificial linear BSS problem.

Results show that both fixed-point CPD and QRT are superior to FastICA when it comes to the computation time needed to reach convergence, while producing estimated sources of similar quality. It is shown that when the amount of sources to estimate is increased both QRT and FastICA struggle to converge. In contrast, the fixed-point CPD method converges within a consistent amount of iterations, suggesting a method more suitable for the estimation of a large amount of sources.

Table of Contents

List of Figures

List of Tables

Preface

The photo shown on the cover of this thesis is shot at a low shutter speed while swinging an LED-cord in a dark room. Halfway during the shot the blue part of the LED-cord is changed to the color red. The result is a mixture of colorful lines which seem partially separated and partially mixed with each other. To us, the impression it leaves is akin to that of Blind Source Separation where we think we can see separate signals but we do not know for sure how many there are and what they truly look like.

Acknowledgements

I would like to thank my supervisors for their assistance during the writing of this thesis.

Delft, University of Technology
November 25, 2022

P. Denarié

Chapter 1

Introduction

Advancements in computational technologies stimulate the use of increasingly larger quantities of data together with more mainstream use of multi-dimensionality [4]. This allows for better estimation of Higher Order Statistics (HOS) which are used to solve complex problems in many fields such as machine learning [5][6][7], economics [8][9], signal processing [10][11], general engineering practices [12][13][14] and neuro-imaging [15][5][16].

One such HOS related problem relevant to all aforementioned fields is the extraction of latent information from observed data. From a signal processing perspective, this comes down to unmixing measured data into separate signals such that components of interest can be identified. This challenge of unmixing latent signals is called Blind Source Separation (BSS). After having been studied extensively the past 3 decades [17], the BSS problem remains relevant with new solutions still being found [18][19][20].

We explain BSS through the example of functional neuro-imaging where the objective is to extract genuine functional brain activity from the measured mixtures. At any given moment an unknown amount of processes take place simultaneously within the nervous system and sensors can measure mixtures of these processes as a single signal. The mixing of these source components $\mathbf{s}$ into measured signals $\mathbf{x}$ is illustrated in Figure 1-1. For example, assume that source s_2 represents the activity in the brain responsible for an epileptic seizure of a patient and source s_1 and s_3 represent the activity responsible for heart rate and eye movement. When researching epileptic seizures source component s_2 is clearly of interest. However, s_2 is not measured directly but only indirectly as it is mixed with s_1 and s_2 to form the mixtures x_1, x_2 and x_3. With little to no prior knowledge available about the source signals and how they are mixed this problem extracting s_2 is ill-posed.

This is what defines the BSS problem, where blind indicates that nothing or very little is known about the sources and the mixing model. By defining a mathematical framework for BSS, useful estimations of the sources can be derived that rely on varying assumptions based on statistical and mathematical properties of the underlying processes.

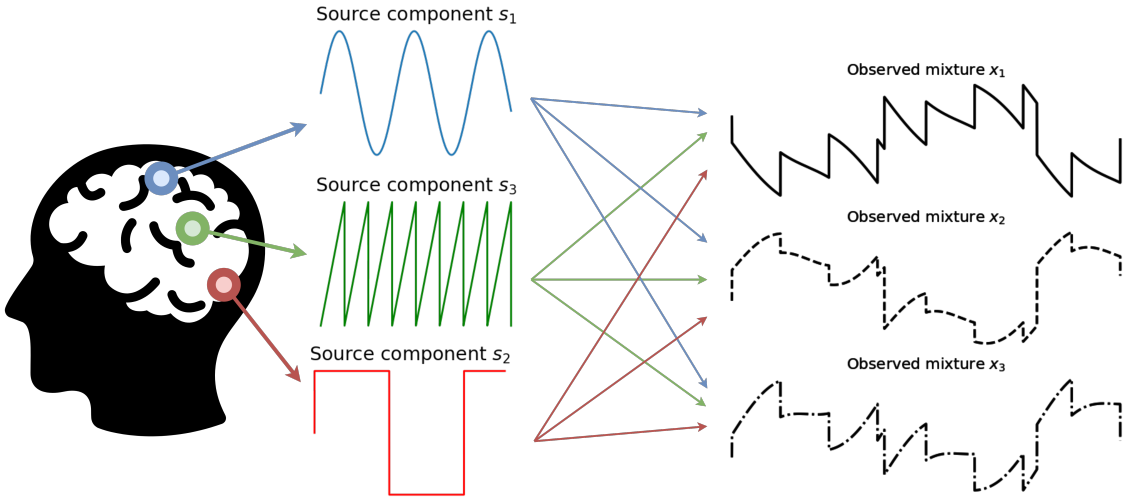


Figure 1-1: The mixing of source signals $\mathbf{s}$ into measured mixtures $\mathbf{x}$. *Brain by Marek Polakovic from NounProject.com*

Throughout this thesis we consider the case where the mixtures $\mathbf{x} = (x_1, x_2, \ldots, x_P)^T \in \mathbb{R}^P$ are assumed to be a linear combination of the source signals $\mathbf{s} = (s_1, s_2, \ldots, s_N)^T \in \mathbb{R}^N$ which has already been widely covered in literature [17][21][22] with a plethora of existing solutions [21][23][24][11][25][15][26][27]. Furthermore, we assume that the amount of mixtures is at the very least equivalent to the amount of source components present $P \geq N$ and that the mixing model $\mathbf{A}$ is stationary, meaning it does not change over time.

Definition 1.1: Linear Mixture model $\mathbf{A}$ [17][21]

For a **linear mixture model** the set of **output signals** $\mathbf{x} = (x_1, x_2, \ldots, x_P)^T \in \mathbb{R}^P$ consist of a weighted sum of the source signals $\mathbf{s} = (s_1, s_2, \ldots, s_N)^T \in \mathbb{R}^N$:

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \tag{1-1}$$

where the mapping $\mathbf{A}$ is a $P \times N$ matrix of which its entries $\{a_{i,j} \quad \forall i \in P, j \in N\}$ are scalar values denoting the weight coefficients of the mixture.

A more in-depth classification of the various existing mixing models and model properties is presented in appendix B-1.

Solving the BSS problem amounts to identifying the inverse mapping of the mixture process such that the source components can be found. In order to do so an additional assumption on the source components $\mathbf{s}$ has to be made.

1-1 Introduction to Independent Component Analysis

Two of the most commonly used assumptions are that the original source components are statistically independent of each other and the assumption that the multidimensional observed data is low in rank. Both can be exploited with tensor decomposition [28][29][5][7] but the former only uses HOS tensors. The latter has seen a recent rise in research, especially in the field of neuro-imaging [15][30][31] due to the growing interest in neuro-imaging technologies combined with the natural representation of multi-way biomedical data as higher-dimensional arrays [5][15]. However, the statistical independence assumption which forms the core principle in Independent Component Analysis (ICA) still remains relevant for functional neuro-imaging [16][32] and other fields such as speech recognition [18], portfolio-diversification [8] and more [13][14]. In all of these cases ICA has proven to be a versatile mathematical tool for finding a solution to the BSS problem.

One of the pioneering ICA methods devised by Hyvarinen called FastICA [33] is still widely used and has seen many alterations [21][2][34]. It is an optimization method that finds the source components by optimizing over a space defined by negentropy, some HOS measure such as kurtosis or a set of nonlinear functions which approximate either of the two at a lower computational cost. For the estimation of many source components it is desirable to keep computational cost as low as possible. For this reason, FastICA is still considered as the *state-of-the-art* method due to its low computational and storage cost.

Opposed to said optimization algorithms are methods which find the solution algebraically [21][17][35][10][23][36]. Similarly to how the covariance matrix is diagonalized through some transformation in Principal Component Analysis (PCA), algebraic methods diagonalize a set of delayed second-order covariance matrices [37] or some HOS tensors [23][11][35] which are classified as tensorial methods [21].

Generally speaking ICA can be separated into these 2 categories. Fundamentally the above mentioned methods all rely on the principle of non-Gaussianity which is inherently intertwined with the statistical independence assumption through the Central Limit Theorem (CLT) [38]. Hyvarinen showed [21] that certain algebraic HOS methods lead to exactly the same algorithm as the FastICA kurtosis method. Tensorial methods have been demonstrated to deliver consistent and reliable results, e.g., with functional Magnetic Resonance Imaging (fMRI) data [39][40]. However, the best results are in general obtained through the use of group methods [41][39][42] where multiple algorithms are used with varying initializations. As such, the need for efficient and characteristically unique new algorithms persists. Nevertheless, algebraic methods definitely have importance amongst all ICA algorithms. Key to understanding algebraic methods are higher-order cumulants and tensor diagonalization which are both explained below.

Fourth order cumulant: kurtosis Most algebraic methods diagonalize the fourth order cumulant tensor [21][17] which is a 4-way symmetric array containing the cross-kurtosis values of the data. A diagonal kurtosis tensor implies statistical independence amongst the components. Algebraic methods are alternatively referred to as tensorial methods. A 4-way symmetric tensor denotes an array of size $\mathbb{R}^{P\times P\times P\times P}$ which is a generalization of symmetric matrices to higher orders. The word "dimension" is purposefully avoided as often in multilinear algebra and data science it denotes the size P.

Kurtosis has been shown to be a good measure of non-Gaussianity but it is not without its drawbacks. Firstly, kurtosis has been critiqued [43][44] for its sensitivity to outliers which makes it not a robust measure of non-Gaussianity. This can be mitigated to some extent by applying pre-processing techniques [45] that detect and remove outliers. For fMRI data it is common practice to spatially smooth the data by convolving the functional images with a Gaussian kernel [46] prior to analysis which helps in eliminating outliers. Secondly, without any knowledge of a variable's probability density function, many samples I are required to be able to get a proper estimate of kurtosis, usually through k-statistics. Mccullagh states in [44] that roughly 40000 to half a million observations are needed to accurately estimate the fourth-order cumulant of a variable. Accurate is defined by Mccullagh as correctly estimating kurtosis up to its second decimal. However, kurtosis based algorithms have been shown to deliver good and reliable results when properly used [26][47].

Cumulant tensor diagonalization Reliable solutions to the BSS problem can be found through diagonalization of the cumulant tensor, i.e., a fourth-order symmetric multi-linear array containing the cross-kurtosis values of observed mixtures. Pointed out by Comon [10], tensor diagonalization for ICA is in general best done by looking for an approximate diagonalization. In reality the ICA model does not hold exactly and sampling errors exist in the data which in most cases is also corrupted by noise. Different cumulant tensor approximate diagonalization algorithms (FOBI[35], COM2[10], JADE[23], STOTD[48], FOOBI2[24]) have been compared [49][26][50] with methods such as FastICA. For low-dimensional spaces these methods provide a competitive alternative. However, they suffer from high computational requirements when dimensionality of the problem is increased [21][50][26][17] due to the quartic scaling of the cumulant tensor $O(P^4)$. This poor scalability is known as *the curse of dimensionality*.

Curse of dimensionality of HOS in ICA The curse of dimensionality is not unique to the cumulant tensor. As a matter of fact, any higher-order statistical tensor suffers from poor scalibility. This fact alone is often attributed to be the cause why HOS tensor methods are avoided. However, this is not a valid reason why HOS tensors and HOS tensorial ICA methods should not be further studied. HOS tensorial methods for ICA have been shown to enjoy useful properties such as robustness and convergence[17][21].

Tensor decomposition for ICA A common method for addressing the poor scalability of tensorial problems is through clever exploitation of the problem's multidimensional structure and breaking it down into elementary pieces, better known as tensor decomposition [28][5]. Existing cumulant tensor ICA methods have been shown to be closely related to a tensor decomposition format known as Canonical Polyadic Decomposition (CPD) [51][52] due to its capability of diagonalizing a tensor. Various formats have properties useful for ICA such as tensor compression and introducing structure. However, the issue that all tensor decomposition formats share when it comes to ICA is that the cumulant tensor has to be first explicitly computed. This means that the scalability issue cannot be properly addressed by tensor decomposition alone.

1-2 Research objective: lifting the curse

The scalablity issue of tensorial ICA methods is based on the premise that these HOS tensors have to be manipulated explicitly. However, existing research already suggests [53] that this issue can be bypassed through exploitation of a HOS tensor's structure. This results in manipulation and decomposition in implicit fashion. In ICA the implicit manipulation of parts of the cumulant tensor is already done by the renowned FastICA algorithm.

The question then arises whether an implicit cumulant tensor manipulation scheme can be combined with tensor decomposition such that new ICA methods can be developed which enjoy characteristics of both optimization based and algebraic methods. This introduces the following research objective of this thesis.

1.1: Research objective

Cumulant tensor based ICA methods for solving the BSS problem suffer in general from the **curse of dimensionality**. The **main research objective** of this thesis is to present how the cost of storage and decomposition of the cumulant tensor for ICA can be reduced through the use of its implicit manipulation. This is formulated into the following research question:

Can implicit decomposition of the cumulant tensor for ICA provide a competitive alternative to FastICA in terms of convergence, solution quality and speed?

The following sub-questions are devised to aid in answering the main question:

- *Can the storage cost and computational cost of manipulating the cumulant tensor be reduced through the use of an implicit manipulation scheme?*
- *Can tensor decomposition methods which have already existing uses for ICA benefit from this implicit manipulation scheme?*

The **secondary objectives** are to **verify** these theoretical cost reductions and to **compare the performances** of the implicit algorithms with that of the widely used FastICA on a BSS problem. Both are done in a controlled test environment using a self designed artificial BSS problem of which the solution is known.

Building on the philosophy of [53], the implicit decomposition of the cumulant tensor forms the cornerstone of the class of implicit algorithms this thesis presents. Meant for but not restricted to, the cumulant tensor with the purpose of solving the BSS problem. All algorithms are an implicit (approximate) version of their explicit counterparts. One novel tensor decomposition algorithm is presented. Another algorithm shows the similarity between a first-order canonical polyadic decomposition problem of the cumulant tensor and FastICA. In all cases a

theoretical decrease in either the computation cost, the storage cost, or both is presented. For several algorithms the quartic dependency on the amount of mixtures is reduced to a linear dependency on the amount of components to be estimated. The theoretical cost reductions are verified through a numerical experiment. Moreover, the experiment shows how two of the presented methods can be considered as new competitors to FastICA when it comes to estimation quality, convergence and computation time. The presentation of it all is structured in such a way that newcomers to the topics of BSS, ICA and tensor decomposition can find all necessary information needed to comprehend the end result.

1-3 Contributions

This section gives the outline of this thesis by chapter. Each chapter is written with a specific purpose. As such, the chapters purpose is explained and the resulting contributions of the chapter to this thesis are listed.

Chapter 2: Multi-linear algebra This chapter presents the core definition of tensors together with higher-order equivalents of notions such as diagonality and symmetry. Furthermore, we present multi-linear algebraic operations together with their respective notations which are used extensively throughout this thesis. Even when knowledgeable on the topic of tensors and multi-linear algebra, the reader is advised to read through this chapter as important nomenclature is presented.

1.2: Contributions chapter 2

- The **base knowledge** concerning **tensors** and **multi-linear algebra** for ICA is presented such that all information is present necessary to comprehend later discussed matter.

Chapter 3: Independent Component Analysis Independent Components Analysis is the framework used in this thesis for solving the BSS problem. In order to fully understand the choices and reasoning made in later chapters it is of great importance that the general notion of ICA is clear to the reader. This means that its working principle and its implications to the BSS problem must be understood together with its accompanied indeterminacies. The importance of whitening is theoretically shown. Furthermore, an example of an existing algebraic ICA algorithm known as COM2 is shown as it has similarities to our later presented QR-Tensor algorithm (QRT). Besides this the renowned optimization based FastICA algorithm is presented. This algorithm plays an extremely important part in practical ICA due to its speed and overall simplicity. On top of that, a fixed-point based CPD algorithm presented later on shows a lot of similarity to the FastICA algorithm and can be considered as a constrained version of it. Lastly, advantages and disadvantages of both the COM2 and FastICA algorithms are discussed. The algorithms are compared to each other to illustrate the detrimental effect of the scalability of the COM2 algorithm and algebraic methods in general.

The reader is advised to read through this chapter if not knowledgeable on the topic of ICA.

1.3: Contributions chapter 3

- All necessary **basic knowledge** concerning ICA is written in compact and logically structured fashion such that newcomers to the topic may understand its basic principles.
- Through the use of examples the difference between **optimization based** and **algebraic** ICA methods is explained.
- The benefits and drawbacks of both optimization based and algebraic ICA methods in general are listed and explained through examples.
- It is shown how through the **curse of dimensionality** algebraic methods scale generally speaking in computation cost with $O(P^4)$ and in storage cost with $O(P^4)$.

Chapter 4: Implicit tensor decomposition for ICA This chapter presents the new implicit algorithms which are derived to address the scalability issue presented in the previous chapter. An implicit manipulation scheme for the cumulant tensor is presented, which together with whitening allows for fast computation of selective values, fibers and slices of the tensor. On top of that, it is shown how the cumulant tensor's symmetry and the indeterminacies of ICA can be exploited for more efficient handling. Each of the 4 algorithms is based on Tucker decomposition. The general use of each implicit decomposition for ICA is shown. For canonical polyadic decomposition it is shown how the behavior of tensor ranks and diagonality of the cumulant tensor are related.

1.4: Contributions chapter 4

- A scheme is presented based on the cumulant tensor's matricization which allows for simple yet **efficient computation** of a single **element**, a **fiber** or entire **slices** in both whitened and non-whitened case.
- It is shown that the storage cost of Higher-Order Singular Value Decomposition (HOSVD) of the cumulant tensor is decreased from $O(P^4)$ to $O(P2)$ using an implicit scheme.
- A **novel** algorithm (QRT) based on the QR-algorithm for matrices is presented which simultaneously **diagonalizes** the outer slices of a symmetric tensor.
- An **implicit** version of the QRT algorithm for ICA is presented which has an iteration cost of $O(IRP)$ and a storage cost of $O(RP)$.
- A small study into the behavior of tensor ranks for ICA is performed which is related to the diagonality of the cumulant tensor. It is identified that the measure of diagonality adequately reflects a solutions quality.
- A **first-order** optimization problem for **implicitly** computing the CPD of the cumulant tensor is presented which has an iteration cost of $O(IRP)$ and a storage cost of $O(RP)$.
- It is shown how rewriting the first-order CPD optimization problem as a **fixed-point** iteration results in FastICA with a CPD constraint on the cumulant tensor which too has an iteration cost of $O(IRP)$ and a storage cost of $O(RP)$.

Chapter 5: A numerical experiment In this chapter the previously presented implicit algorithms are tested on an artificial BSS problem consisting of 4 latent source signals of which the solution is known. The point of this chapter is 2-fold: to verify the theoretical complexities of the derived algorithms and to inspect their performance for ICA through comparison with that of FastICA. The used artificial test-set is relatively simple as this experiment serves as a proof of concept. The algorithms are divided into 2 categories: non-iterative algorithms and iterative algorithms. Both are tested on their performance of separating mixed source components. However, the former and their variants are tested too for their performance of providing initial estimates for the iterative algorithms, as is suggested by literature. The resulting estimations are tested for correctness using a self-designed classification algorithm. Moreover, they are compared on the diagonality of the resulting cumulant tensor, their estimation errors and computation time. The analysis of the performances is done in a qualitative manner as only relative differences are of interest.

1.5: Contributions chapter 5

- It is shown that the implicit HOSVD and implicit Generalized EigenValue Decomposition (GEVD) algorithms do **not scale favorably** with the amount of mixtures P due to the total SVD-update cost.
- It is shown that computing the implicit HOSVD of the cumulant tensor with only its **unique slices** or only its **core slices** can result in similar performance for ICA as computing the full HOSVD.
- It is shown that for a linearly mixed BSS problem with additive Gaussian-White-Noise (GWN) the source component estimation performance of QRT and fixed-point CPD is on par with that of kurtosis-based parallel FastICA in terms of estimation error and solution correctness.
- It is shown that the *QRT* and *I-CPD-FF* algorithms can achieve better results than FastICA when combined with the HOSVD based algorithms as initial estimates in terms of correctness and estimation error.
- It is shown that the **computation time** of the QRT algorithm to reach a certain convergence tolerance is lower than that of FastICA by nearly an order of magnitude due to its lower per iteration cost.
- It is shown that a **fixed-point** iteration of the implicit CPD first order optimization method **converges in fewer iterations** towards a solution than FastICA. As a result the computational time needed by the implicit CPD first order optimization method scales much more favorably with P as it does for FastICA.
- It is shown that for the case when FastICA and QRT struggle to reach convergence, the fixed-point implicit CPD first order optimization method manages to converge in a consistent and much smaller amount of iterations.

The thesis is concluded in chapter 6 in which a conclusion and recommendations for future research are given. A complete overview of the thesis structure is shown in Figure 1-2. All general theoretical contributions of this thesis are presented in the green boxes and build

upon the already existing knowledge shown in the blue boxes.

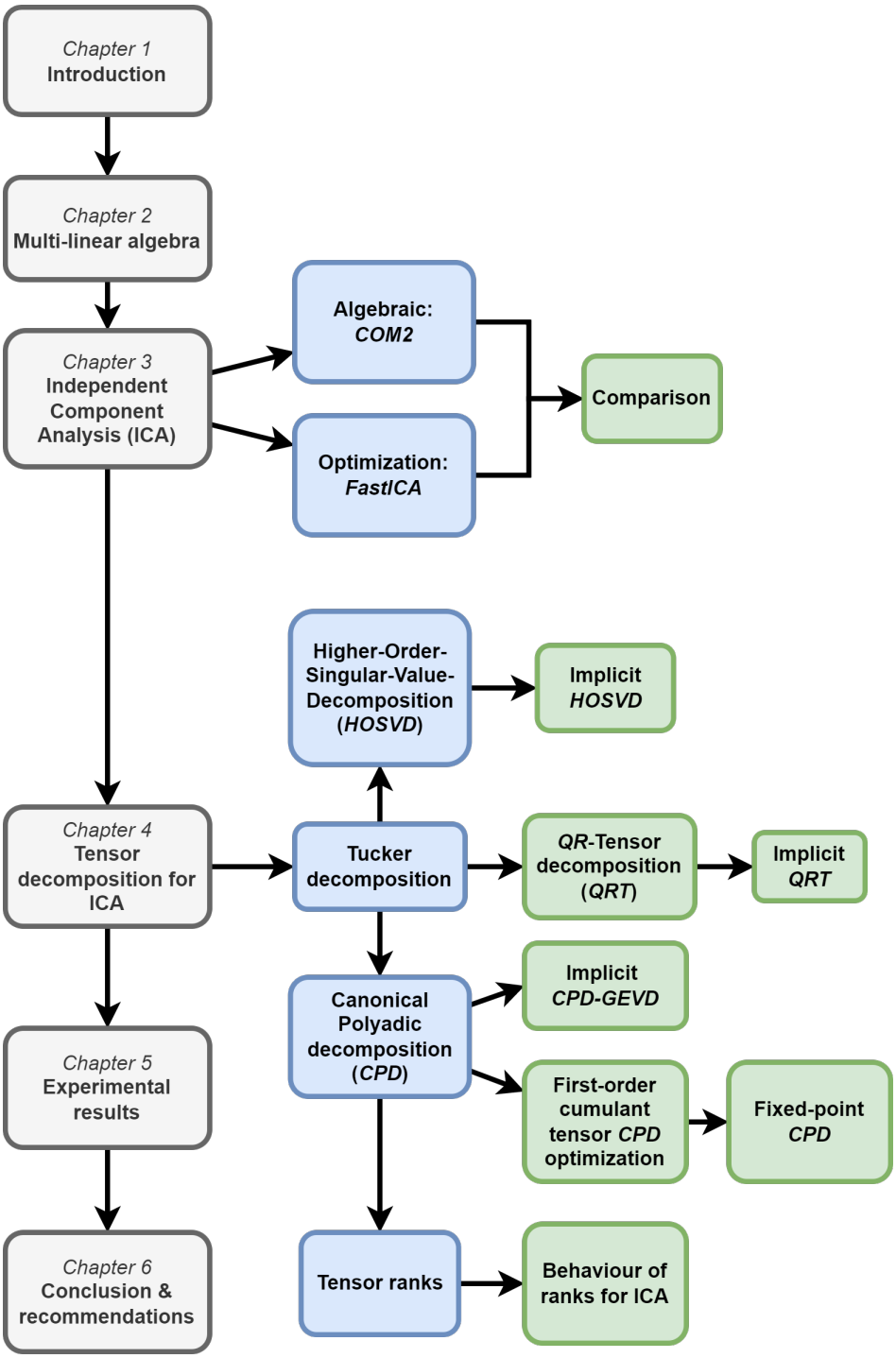


Figure 1-2: Overview of the theoretical topics within this thesis. Chapters are colored in gray, theoretical concepts which already exist are colored in blue and the contributions of this thesis are presented in green.

Chapter 2

An overview of multilinear algebra

This chapter covers the preliminary notation on multi-linear algebra used throughout this thesis. Many of the definitions build on other multi-linear definitions, so they are kept together in this separate chapter in logical order primarily for the report's readability. First the basic definitions and properties of higher-order tensors are explained. This is followed by multilinear operations such as matricization and the mode-N product which are often used in the chapters that follow.

2-1 Tensor preliminaries

Notation of tensors A tensor is any type of data structure which represents multilinear relations between mathematical objects. The values of a tensor are denoted using its *indices* which are interchangeably called *ways* or *modes*. Each tensor i'th index varies from 1 to its maximal value P_i. The indexing of a tensor is denoted using parentheses. The colon format '$s:e$' is used to denote all values starting at s up to and included e along that particular index. No s present means the index starts at 1 and no e present means the index stops at the last value of that mode. The amount of modes a tensor has is known as the tensor's *order*. For example, a tensor of order 0 represents a scalar denoted with lower case regular font $x \in \mathbb{R}$. A first-order tensor is more commonly referred to as a vector, denoted using a bold letter $\mathbf{x} \in \mathbb{R}^{P_1}$. A matrix denoted by a bold capital letter $\mathbf{X} \in \mathbb{R}^{P_1 \times P_2}$ is a tensor of order 2. Tensors of order 3 and above are referred to as Higher-order tensors and are denoted using capital calligraphic letters $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \ldots P_N}$. Often a tensor with N modes is referred to as an N-way tensor. Any tensor which has the property that all of its modes are of equal size can be called 'cubical'.

Definition 2.1: Cubical tensor [28]

A tensor is **cubical** if all modes are of equivalent size, e.g. $\mathcal{X} \in \mathbb{R}^{P \times \ldots \times P}$. The shorthand notation for a N'th-order cubical tensor $\mathcal{X}$ with each mode of size P is: $\mathcal{X} \in \mathbb{R}^{[N,P]}$.

Tensor fibers and slices Fibers and slices are commonly used in tensor algebra as modes represent different types of variability of data.

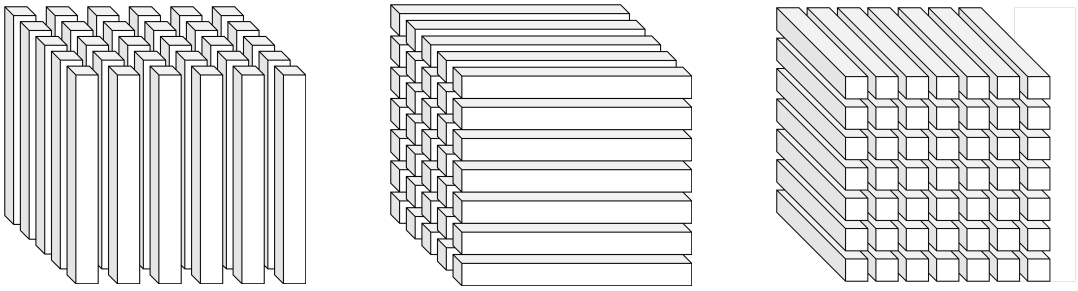
Definition 2.2: Tensor fibers [28]

A **tensor fiber** is a vector that consists of the sequential elements when fixing every index but one. The n-th mode fiber of a tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times \ldots \times P_N}$ is denoted as $\mathbf{x}(p_1, \ldots, p_{n-1}, :, p_{n+1}, \ldots, p_N)$ where the colon indicates the non-fixed dimension.

Definition 2.3: Tensor slices [28]

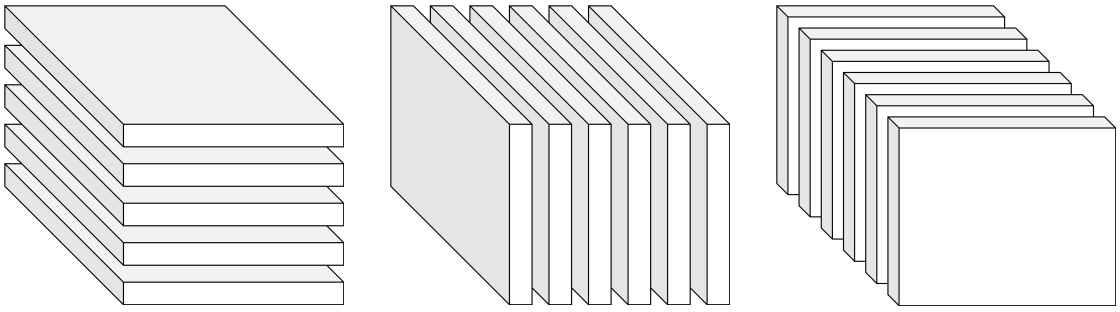
A **slice** of a tensor is a matrix defined by fixing all but two of its indices. For example, the horizontal, lateral and frontal slices of a 3-dimensional tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$ are denoted as $\mathcal{X}(p_1, :, :)$, $\mathcal{X}(:, p_2, :)$ and $\mathcal{X}(:, :, p_3)$ where the colon denotes the non-fixed modes.

Figure 2-1 illustrates the 3 possible fiber configurations of a third-order tensor and Figure 2-2 illustrates its possible slices.



(a) Mode-1 fibers of a 3-way tensor. **(b)** Mode-2 fibers of a 3-way tensor **(c)** Mode-3 fibers of a 3-way tensor

Figure 2-1: The 3 different mode fibers of a 3-way tensor.



(a) Horizontal slices of a 3-dimensional tensor. **(b)** Lateral slices of a 3-dimensional tensor **(c)** Frontal slices of a 3-dimensional tensor

Figure 2-2: The 3 different slice types of a 3-way tensor.

Index permutation

Definition 2.4: Index permutation[54]

Let S be the finite integer set consisting of N elements and let S_N be the set consisting of all permutations of S. Reordering the indices of a tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$ results in the tensor $\mathcal{Y}$ whose indices are a permutation $\sigma \in S_N$ of the indices $S = \{1, 2, \ldots, N\}$ of $\mathcal{X}$. This is denoted by:

$$\mathcal{Y} = \mathcal{X}^{T\sigma} \quad \text{for} \quad \sigma \in S_N. \tag{2-1}$$

Symmetric tensor Similar to matrices, tensors can show symmetry too. Tensors can be fully symmetric or only partially [55] with respect to a selection of modes. However, only full symmetry is of importance throughout the report so only its definition is given below.

Definition 2.5: Symmetric tensor

A cubical tensor $\mathcal{X} \in \mathbb{R}^{[N,P]}$ is symmetrical, if the value of its elements remain constant under any permutation from definition 2.4 of its indices:

$$\mathcal{X} = \mathcal{X}^{T\sigma} \quad \forall \sigma \in S_N. \tag{2-2}$$

Diagonal tensor The general definition of a diagonal tensor given below extends the notion of a diagonal matrix to higher-orders.

Definition 2.6: Diagonal tensor [28]

The **superdiagonal** is the higher-order equivalent of the diagonal of a square matrix. The superdiagonal of a cubic tensor $\mathcal{X}^{[N,P]}$ consists of all elements for which all indices are identical:

$$\mathcal{X}(p_1, p_2, \ldots, p_N) \quad \forall p_1 = p_2 = \ldots = p_N. \tag{2-3}$$

A cubical tensor $\mathcal{X} \in \mathbb{R}^{[N,P]}$ is **diagonal** if and only if all non-zero elements are on the **superdiagonal**. Meaning that all off-diagonal elements equal zero:

$$\mathcal{X}(p_1, p_2, \ldots, p_N) = 0 \quad \forall \neg(p_1 = p_2 = \ldots = p_N) \quad \forall p_1, \ldots, p_N = 1, \ldots, P, \tag{2-4}$$

where the negation symbol $\neg$ means not equal to the statement that follows.
Taking only the values on the superdiagonal of a tensor $\mathcal{X}$ is denoted as $\text{diag}(\mathcal{X})$ and taking only the values that are not on the superdiagonal is denoted as $\text{offdiag}(\mathcal{X})$. The reshaping of a vector $\mathbf{x} \in \mathbb{R}^N$ into a diagonal N-way tensor by placing its elements on the superdiagonal is denoted as $\text{diag}_N(\mathbf{x})$.

2-2 Multilinear operations

Kronecker, Khatri-Rhao and Hadamard product

Definition 2.7: Kronecker product [28]

The **Kronecker product** of matrices $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_J \end{bmatrix} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \dots & \mathbf{b}_L \end{bmatrix} \in \mathbb{R}^{K \times L}$ is denoted by $\mathbf{A} \otimes \mathbf{B}$ and its resulting matrix of size $IK \times JL$ is defined as:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} \mathbf{a}_1(1) \cdot \mathbf{B} & \mathbf{a}_2(1) \cdot \mathbf{B} & \cdots & \mathbf{a}_J(1) \cdot \mathbf{B} \\ \mathbf{a}_1(2) \cdot \mathbf{B} & \mathbf{a}_2(2) \cdot \mathbf{B} & \cdots & \mathbf{a}_J(2) \cdot \mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_1(I) \cdot \mathbf{B} & \mathbf{a}_2(I) \cdot \mathbf{B} & \cdots & \mathbf{a}_J(I) \cdot \mathbf{B} \end{bmatrix}, \tag{2-5}$$

with a computational cost of $O(IJKL)$.

Definition 2.8: Khatri-Rhao product [28]

The **Khatri-Rhao product** is the matching **columnwise** Kronecker product of matrices $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_K \end{bmatrix} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \dots & \mathbf{b}_K \end{bmatrix} \in \mathbb{R}^{J \times K}$. It is denoted by $\mathbf{A} \odot \mathbf{B}$ and its resulting matrix of size $IJ \times K$ is defined as:

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_K \otimes \mathbf{b}_K \end{bmatrix}, \tag{2-6}$$

with a computational cost of $O(KIJ)$.

Definition 2.9: Hadamard product product [28]

The **Hadamard product** is the elementwise product of matrices $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_J \end{bmatrix} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \dots & \mathbf{b}_J \end{bmatrix} \in \mathbb{R}^{I \times J}$. It is denoted by $\mathbf{A} * \mathbf{B}$ and its resulting matrix of size $I \times J$ is defined as:

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} \mathbf{a}_1(1) \cdot \mathbf{b}_1(1) & \mathbf{a}_2(1) \cdot \mathbf{b}_2(1) & \cdots & \mathbf{a}_J(1) \cdot \mathbf{b}_J(1) \\ \mathbf{a}_1(2) \cdot \mathbf{b}_1(2) & \mathbf{a}_2(2) \cdot \mathbf{b}_2(2) & \cdots & \mathbf{a}_J(2) \cdot \mathbf{b}_J(2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_1(I) \cdot \mathbf{b}_1(I) & \mathbf{a}_2(I) \cdot \mathbf{b}_2(I) & \cdots & \mathbf{a}_J I1) \cdot \mathbf{b}_J(I) \end{bmatrix}. \tag{2-7}$$

The computational cost of performing the above described Hadamard product is $O(IJ)$.

The **elementwise power** operation of a matrix $\mathbf{M}$ where each element is raised to the power N is denoted as $[\mathbf{M}]^N$ and equals the following sequence of Hadamard products:

$$[\mathbf{M}]^N = \underbrace{\mathbf{M} * \mathbf{M} * \cdots * \mathbf{M}}_{N-1 \text{ products}}, \tag{2-8}$$

which is performed at a computational cost of $O\left((N-1)IJ\right)$.

Outer-product

Definition 2.10: Outer Product [28]

The **outer-product** $\mathcal{X}$ of N vectors $\{\mathbf{u}_1 \in \mathbb{R}^{P_1}, \dots \mathbf{u}_N \in \mathbb{R}^{P_N}\}$ is the N-dimensional object that exists in the outer-product space of the individual vector spaces $\mathcal{X} = \mathbf{u}_1 \circ \dots \circ \mathbf{u}_N \in \mathbb{R}^{P_1} \circ \dots \circ \mathbb{R}^{P_N}$ where $\circ$ denotes the outer product. Element-wise this is defined as:

$$\mathcal{X}(p_1, p_2, \dots, p_N) = \mathbf{u}_1(p_1)\mathbf{u}_2(p_2) \cdots \mathbf{u}_N(p_N). \tag{2-9}$$

The outer-product operation is applicable to tensors of any order. The outer-product of 2 tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times \dots \times J_M}$ results in a new tensor which is of order $N + M$:

$$\mathcal{X} \circ \mathcal{Y} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}. \tag{2-10}$$

Vectorization Vectorization is the transformation of any tensor of order 2 and higher into a single vector which is a structured concatenation of the tensor's elements.

Definition 2.11: Vectorization

Vectorizing a given tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times \dots \times P_{N-1} \times P_N}$ consists of stacking its elements row-wise into a single vector of size $\prod_{i=1}^{N} P_i$ as follows:

$$\text{vec}\left(\mathcal{X}\right) = \begin{bmatrix} \mathcal{X}(1,\dots,1,1) & \dots & \mathcal{X}(1,\dots,1,P_N) & \mathcal{X}(1,\dots,2,1) & \dots & \mathcal{X}(P_1,\dots,P_{N-1},P_N) \end{bmatrix}^T. \tag{2-11}$$

Mode-N matricization Matricization, defined in [28] as: *'the process of reordering the elements of an N-way array into a matrix'* is an important tensor operation in tensor algebra. Its definition is given below. It helps to visualize the mode-N-matricization of a tensor. A graphical example of mode-3 matricization of a third-order tensor can be found in figure Figure 2-3. The figure shows how the indexing of the tri-linear tensor is translated into a bi-linear format.

Definition 2.12: Mode-N matricization [28]

A tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times \dots \times P_N}$ can be **matricized** along a mode n by stacking the mode-n fibers in **columnwise** fashion:

$$\begin{aligned} \mathbf{X}_{(n)}\left(p_n, j\right) &= \mathbf{X}\left(p_1, p_2, \dots, p_N\right) \\ j &= 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} \left(p_k - 1\right) \prod_{\substack{m=1 \\ m \neq n}}^{k-1} P_m, \end{aligned} \tag{2-12}$$

where $\mathbf{X}_{(n)}$ denotes the mode-n matricized tensor.

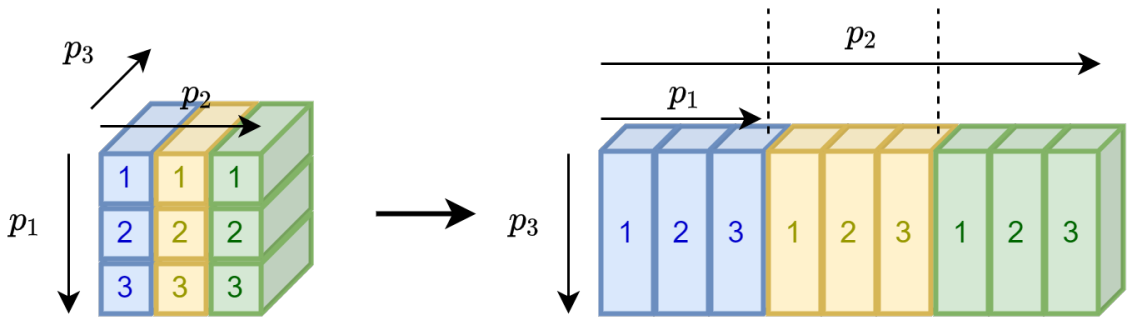


Figure 2-3: Mode-3 matricization of third-order tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$. The colors and numbers indicate how the mode fibers are concatenated next to each other. The indices $\{p_1, p_2, p_3\}$ show how navigation through the tensor is translated into its matricized form.

Mode-n product The mode-n product is the multiplication and summation of the n'th mode of a tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times \ldots P_n \ldots \times P_N}$ with the second mode of a matrix $\mathbf{M} \in \mathbb{R}^{J \times P_n}$ and is denoted by $\mathbf{X} \times_n \mathbf{M}$ [28].

Definition 2.13: Mode-N product [28]

The elementwise definition of the mode-n product between a tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times \ldots P_n \ldots \times P_N}$ and matrix $\mathbf{M} \in \mathbb{R}^{J \times P_n}$ is given in [28] as:

$$\left(\mathcal{X} \times_n \mathbf{M}\right)(p_1, \cdots, p_{n-1}, j, p_{n+1}, \cdots, p_N) = \sum_{p_n=1}^{P_n} \mathcal{X}\left(p_1, p_2 \cdots p_n \cdots p_N\right) \mathbf{M}\left(j, p_n\right). \tag{2-13}$$

The computational complexity of this operation is $O(P_1 \cdots P_N J)$.

Inner-product and tensor norm The inner-product is the summation over all dimensions of the element-wise products of of 2 tensors of identical shape. One of the recurring use-cases of the inner-product is that it can be used to compute tensor norms. Taking the square root of the inner-product of a tensor with itself results in the computation of its norm which is a measure of its magnitude.

Definition 2.14: Inner-product [28]

The **inner product** of 2 tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$ is defined as the sum of all products of their entries at identical indices:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{}^{P_1} \sum_{}^{P_2} \cdots \sum_{}^{P_N} \mathcal{X}\left(p_1, p_2, \ldots, p_N\right) \cdot \mathcal{Y}\left(p_1, p_2, \ldots, p_N\right) \tag{2-14}$$

Definition 2.15: Tensor norm [28]

The **norm** of a tensor is defined as the **square root** of the sum of all products of their entries at identical indices, which is identical to the square root of the **inner-product** of the tensor with itself:

$$\|\mathcal{X}\|_2 = \sqrt{\sum_{}^{P_1}\sum_{}^{P_2}\cdots\sum_{}^{P_N}\mathcal{X}(p_1 p_2 \cdots p_N)^2} = \sqrt{\langle \mathcal{X}, \mathcal{X}\rangle}. \tag{2-15}$$

For 2 equally sized vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^P$ the inner-product equals the dot product of the vectors with the first vector being transposed: $\langle \mathbf{u}, \mathbf{v}\rangle = \mathbf{u}^T \cdot \mathbf{v} = \mathbf{v}^T \cdot \mathbf{u}$. Depending on the coding language used, binary machines store and access their data in either row-major or column-major order. Letting operations iterate through a single row or column is faster than iteration through multiple rows or multiple columns. Thus the inner product of any 2 identically shaped tensors can alternatively be computed more efficiently by vectorizing each array following definition 2.11 and applying the previously mentioned vector inner product:

$$\langle \mathcal{X}, \mathcal{Y}\rangle = \text{vec}\,(\mathcal{X})^T \cdot \text{vec}\,(\mathcal{Y}) \tag{2-16}$$

Tensor Times Same Vector The Tensor Times Same Vector (TTSV) operation is the multiplication and summation of a tensor with the same vector along all or a selection of its modes. Below in 2.16 the definition is given over all modes and for the special case for all modes but one of a symmetric tensor.

Definition 2.16: Tensor Times Same Vector [56]

The **Tensor Times Same Vector** operation is defined as the product and summation along all modes of a cubic tensor $\mathcal{X} \in \mathbb{R}^{[N,P]}$ with a vector $\mathbf{v} \in \mathbb{R}^P$ and is denoted as follows:

$$\mathcal{X}\mathbf{v}^N = \mathcal{X} \times_1 \mathbf{v}^T \times_2 \ldots \times_N \mathbf{v}^T = \sum_{p_1=1}^{P} \cdots \sum_{p_N=1}^{P} \left(\mathcal{X}(p_1 p_2 \cdots p_N) \prod_{k=1}^{N} \mathbf{v}(p_k) \right) \tag{2-17}$$

which results in a scalar at a computational cost of $O(P^N)$.

The TTSV in all modes but one for a **symmetric** tensor $\mathcal{X}_{sym} \in \mathbb{R}^{[N,P]}$ with a vector $\mathbf{v} \in \mathbb{R}^P$ is denoted as:

$$\left(\mathcal{X}_{sym}\mathbf{v}^{N-1}\right)_{p_1} = \sum_{p_2=1}^{P} \cdots \sum_{p_N=1}^{P} \left(\mathcal{X}(p_1 p_2 \cdots p_N) \prod_{k=2}^{N} \mathbf{v}(p_k) \right) \quad \text{for all} \quad p_1 \in \{1, \ldots, P\}, \tag{2-18}$$

which results in a vector of size P. The choice of the left-out mode does not matter due to the symmetry of $\mathcal{X}_{sym}$.

Chapter 3

Independent Component Analysis

One possible method of solving the Blind Source Separation (BSS) problem is Independent Component Analysis (ICA). Describing data in a statistical framework [46] can be used to process or transform the data in such a way that hidden patterns and information can be revealed. This statistical framework is the foundation on which the workings of ICA are build. While having similarities with Principal Component Analysis (PCA), ICA can be used to find components that are statistically independent which is a much stronger notion than uncorrelatedness, although harder to satisfy. However, under specific conditions and with the right assumptions ICA can be a powerful tool with many diverse strategies [21][17] when it comes to solving the BSS problem.

Chapter organization First the core concept of statistical independence is introduced along with the central limit theorem. This forms the foundation of ICA which can be categorized into optimization and algebraic methods. For the algebraic methods the general working principle of approximate diagonalization is explained through the example of the method known as COM2. FastICA is briefly introduced together with some of its noteworthy properties as the state-of-the-art optimization method. Afterwards, the definitions of algebraic and optimization methods are compared with each other and their general respective pro's and con's are listed. Formulation of the scaling issue of algebraic methods serves as the stepping stone into the subsequent chapter.

3-1 Statistical Independence

The working principle of ICA is the assumption that the original source signals are statistically independent of each other. From a practical point of view this means that when statistically analysing one source component nothing can be inferred about the other components. This assumption however, only tells us something about the source components. For that reason, the Central Limit Theorem (CLT) [38] is needed which allows an important observation to be made about how the mixing mode can be estimated. Before that, the reader is given a brief reminder of the Gaussian distribution as it is used as a measure of independence.

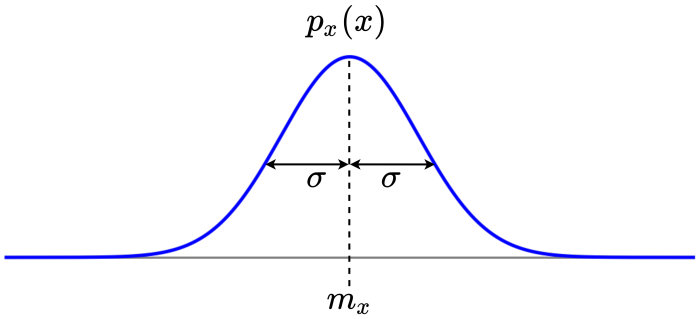


Figure 3-1: A Gaussian probability density function $p_x(x)$ of a random variable x with mean m_x and standard deviation σ.

3-1-1 The Gaussian distribution

The Gaussian distribution, also known as the normal distribution or informally as a bell curve, is a unique distribution which serves for many situations as the go-to distribution for modelling additive noise.

Definition 3.1: Gaussian (normal) distribution

A **Gaussian** distribution is a probability distribution characterized by its the **probability density function** given as:

$$p_x(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-m_x}{\sigma}\right)^2} \tag{3-1}$$

where σ denotes the standard deviation and m_x the mean.

The **standard Gaussian** distribution is when the distribution has zero-mean $m_x = 0$ and unit variance $\sigma^2 = 1$.

$$p_x(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{3-2}$$

Figure 3-1 shows an example of a Gaussian distribution together with a mean of m_x and standard deviation of σ.

Central Limit Theorem The central limit theorem establishes that under the condition that random variables are Independently and Identically Distributed (IID), a normalized sum of independent random variables will inherently tend towards a Gaussian (normal) distribution from definition 3.1. For this particular reason the Gaussian distribution is a useful approximation of noise as noise usually consists of a combination of unknown elements. The original variables themselves do not have to be normally distributed in order for this theorem to hold true. The theorem together with the definition of IID variables is given below in definition 3.2 and theorem 3.1. A more practical interpretation of theorem 3.1 is that given the non-Gaussian and IID conditions, a sum of random variables will be more Gaussian than any of its respective components.

Definition 3.2: Independent and Identically Distributed

n **random variables** $\{X_1, \ldots, X_n\}$ which take values in $I \subseteq \mathbb{R}$, are **independent** and **identically distributed** if and only if their individual density functions $f_{X_i}(x)$ and joint density function $f_{X_1,\ldots,X_n}(x_1, \ldots, x_n)$ meet the following requirements:

$$\begin{aligned} p_{X_i}(x) &= p_{X_j}(x) \quad \forall i, j \in \{1, \ldots, n\} \wedge \forall x \in I \\ p_{X_1,\ldots,X_n}(x_1, \ldots, x_n) &= p_{X_1}(x_1) \cdots p_{X_n}(x_n). \end{aligned} \tag{3-3}$$

Theorem 3.1: Central Limit Theorem [57]

Let $\{X_1, \ldots, X_n, \ldots\}$ be a sequence of Independent and Identically Distributed (I.I.D.) **random variables**, drawn from a distribution with expected value $\mathbb{E}[X_i] = \mu$ and **finite** variance $\mathrm{Var}[X_i] = \sigma^2 < \infty$. As n approaches **infinity**, the random variables $\sqrt{n}\left(\frac{\bar{X}_n - \mu}{\sigma}\right)$ where $\bar{X}_n$ denotes the **sample average**, converge in distribution to a **normal** (Gaussian) distribution $\mathcal{N}(0, 1)$:

$$\lim_{n\to\infty} \sqrt{n}\left(\frac{\bar{X}_n - \mu}{\sigma}\right) \xrightarrow{d} \mathcal{N}(0, 1)\,. \tag{3-4}$$

Why the CLT is of importance for ICA is easily shown using the linear mixing model from definition (1-1). Assume that the 3 source components $\{s_1, s_2, s_3\}$ from Figure 1-1 in chapter 1 are mixed linearly into 3 observed signals $\{x_1, x_2, x_3\}$:

$$\begin{aligned} x_1 &= a_{11}s_1 + a_{12}s_2 + a_{13}s_3 \\ x_2 &= a_{21}s_1 + a_{22}s_2 + a_{23}s_3 \\ x_3 &= a_{31}s_1 + a_{32}s_2 + a_{33}s_3 \end{aligned} \tag{3-5}$$

where the mixing coefficient a_{ij} represent the element of the i'th row and j'th column of the mixing matrix $\mathbf{A}$.

Given that these independent signals are non-Gaussian and identically distributed, the central limit theorem states that the distribution of any weighted sum of these 3 variables will be more Gaussian than the distributions of the individual components themselves. Given the assumption about statistical independence, it is implied that to estimate one of the original components $\mathbf{s} = (s_1, s_2, s_3)^T$, one has to find a linear combination $\mathbf{q}$ of the observed variables $\mathbf{x} = (x_1, x_2, x_3)^T$ that decreases the Gaussianity. Let us denote the estimate of a source component with $\hat{s}_i$. The estimate can be expressed as a linear combination $\mathbf{q}^T\mathbf{A}$ of the source components $\mathbf{s}$:

$$\hat{s}_i = \mathbf{q}^T\mathbf{x} = \mathbf{q}\mathbf{A}\mathbf{s}. \tag{3-6}$$

In the ideal case that $\mathbf{q}^T$ represents the i'th row of the inverse mixing matrix $\mathbf{A}^{-1}$, the estimate $\hat{s}_i$ will be identical to a component s_i. This means that $\hat{s}_i$ will be the least Gaussian as it equals one of the source components s_i. In practice, as nothing is known about the assumed linear mixing model $\mathbf{A}$, $\mathbf{q}^T$ cannot be determined exactly [21]. However, as $\hat{s}_i$ must

be as non-Gaussian as possible in order to equal a source component, $\mathbf{q}^T$ can be found by minimizing some measure of Gaussianity. Or in other words, $\mathbf{q}^T$ has to maximize the non-Gaussianity of $\hat{s}_i = \mathbf{q}^T\mathbf{x}$ in order to retrieve the Independent Component (IC) s_i. For a more extensive explanation on this topic, the reader is advised to read chapter 8 of [21].

In the general case one is looking for all Independent Components (IC's) simultaneously as it is impossible to determine beforehand which independent component will result from the vector $\mathbf{q}^T$. For this reason (3-6) can be rewritten in matrix form such that it captures all IC's:

$$\hat{\mathbf{s}} = \mathbf{Q}\mathbf{x} = \mathbf{Q}\mathbf{A}\mathbf{s}, \tag{3-7}$$

where $\mathbf{Q}$ represents a matrix whose rows are the unmixing vectors $\mathbf{q}^T$. So transformation $\mathbf{Q}$ leads to the independent components when non-Gaussianity is maximized.

In order for the result of $\mathbf{Q}\mathbf{x}$ to be a proper solution to the BSS problem, there are two more conditions that need to be satisfied. Namely the existence and uniqueness of the solution. If the problem has no unique solution but many solutions, then there is no way of telling which solution is a correct estimated representation of the IC's. Comon showed [11] that for the linear mixing case the solution to the ICA problem exists and is unique up to some trivial indeterminacies [11].

3-1-2 Indeterminacies

Ideally, the sought for transformation $\mathbf{Q}$ is found as the inverse $\mathbf{A}^{-1}$ of the mixing matrix, assuming that it is square. However, both in theory and in practice it is impossible to know when this exact transformation is found due to 2 indeterminacies that exist within ICA[11][21].

Order of components First of all, the order of the source components cannot be determined. This can be clearly explained using the linear mixture model:

$$\mathbf{x} = \mathbf{A}\mathbf{s}. \tag{3-8}$$

Any permutation matrix $\mathbf{P}$ that is added to the model together with its inverse as $\mathbf{A}\mathbf{P}^{-1}\mathbf{P}\mathbf{s}$, results in just a new mixing matrix $\mathbf{A}\mathbf{P}^{-1}$ that has to be determined. As both $\mathbf{A}$ and $\mathbf{s}$ are unknown, the order of components can be chosen freely so the permutation matrix $\mathbf{P}$ is cancelled. Note that multiplication with any square and invertible matrix other than a permutation matrix gets cancelled too. However, this can influence the problem by changing the properties of the mixing matrix while multiplication with a permutation matrix P does not. This is shown later on.

Variances Secondly, it is impossible to determine the variances of the IC's. Based on the same reasoning as above, the fact that both $\mathbf{A}$ and s are unknown leads to the cancellation of any added scalar terms. Any scalar a_i that is multiplied with one of the sources can be cancelled by division through the same scalar a_i of the corresponding column $\mathbf{a}^{(i)}$ of the mixing matrix:

$$\mathbf{x} = \sum_i \left(\frac{1}{a_i} \mathbf{a}^{(i)} \right) a_i s_i. \tag{3-9}$$

Furthermore, not only is the variance of each IC undetermined, the sign of each IC is undetermined too as each scalar a_i might as well have value -1. However, in most cases this poses no problem.

3-1-3 pre-whitening

An important step which simplifies the BSS problem is pre-whitening of the data. Whitening normalizes the variances of the P observed variables and makes the variables uncorrelated. This results in a diagonal covariance matrix of zero-mean data. As such, a transformation which diagonalizes and normalizes the covariance matrix, can be used to whiten data. When the data is zero mean, correlation and covariance are identical. Hence whitening results in a unit correlation matrix too.

Definition 3.3: Whitening

A matrix $\mathbf{W}$ which transforms the **covariance matrix** $\mathbf{C_x}$ of a random vector $\mathbf{x} \in \mathbb{R}^P$ with I observations into a diagonal matrix with unit variance:

$$\mathbf{W}\mathbf{C_x}\mathbf{W}^T = \mathbf{W}\mathbb{E}\left[(\mathbf{x} - \mathbf{m_x}) (\mathbf{x} - \mathbf{m_x})^T \right] \mathbf{W}^T = \mathbf{I}_P, \tag{3-10}$$

can be used to whiten said random vector:

$$\mathbf{z} = \mathbf{W}\mathbf{x}, \tag{3-11}$$

where $\mathbf{z}$ denotes the whitened random vector and $\mathbf{I}_P$ the identity covariance of size $P \times P$.

The main reason why pre-whitening is beneficial for solving the BSS problem is that it narrows the search space of the unmixing transformation $\mathbf{Q}$ down to orthogonal matrices. This is explained below. Let $\mathbf{C}_x$ be the covariance matrix of the mixture $\mathbf{x} = \mathbf{A}\mathbf{s}$. It is equal to:

$$\mathbf{C_x} = \mathbb{E}\left[(\mathbf{x} - \mathbf{m_x}) (\mathbf{x} - \mathbf{m_x})^T \right] = \mathbb{E}\left[(\mathbf{As} - \mathbf{Am_s}) (\mathbf{As} - \mathbf{Am_s})^T \right] = \mathbf{A}\mathbf{C}_s\mathbf{A}^T. \tag{3-12}$$

Assuming that the source components have unit variance (without loss of generality as the rows of the mixing matrix can be appropriately rescaled), the covariance simply becomes $\mathbf{C_x} = \mathbf{A}\mathbf{A}^T$. Substitution of the Singular-Value-Decomposition (SVD) of the mixing matrix $\mathbf{A} = \mathbf{U}\Sigma\mathbf{Q}^T$ results in the following expression:

$$\mathbf{C_x} = \mathbf{U}\Sigma\mathbf{Q}^T\mathbf{Q}\Sigma\mathbf{U} = \mathbf{U}\Sigma^2\mathbf{U}^T. \tag{3-13}$$

The right hand side of (3-13) is equivalent to the Eigen-Value-Decomposition (EVD) of $\mathbf{C_x}$ where $\mathbf{U}$ contains the eigenvectors and Σ the square root of the corresponding eigenvalues. In other words the source component subspace $\mathbf{U}\Sigma$ can be estimated through second-order

statistics only and the whitening transformation can be taken as $\mathbf{W} = \sqrt{\mathbf{C_x}}^{-1} = \mathbf{U}\Sigma^{-1}\mathbf{U}^T$ or due to orthogonality of $\mathbf{U}$ even simpler as $\mathbf{W} = \Sigma^{-1}\mathbf{U}^T$. Alternatively, when the data matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}(1) & \dots & \mathbf{x}(I) \end{bmatrix}$ has zero mean $\mathbf{U}\Sigma$ can be estimated through its SVD: $\mathbf{X} = \mathbf{U_X}\Sigma_\mathbf{X}\mathbf{Q_X^T}$ as $\mathbf{U}\Sigma = \frac{1}{\sqrt{I}} \cdot \mathbf{U_X}\Sigma_\mathbf{X}$ with $\mathbf{Q_X^T} = \mathbf{Q}^T\mathbf{s}$.

This shows that $\mathbf{W} = \sqrt{I} \cdot \Sigma_\mathbf{X}^{-1}\mathbf{U_X^T}$ is a whitening transformation too based on SVD. Both methods amount to PCA and in both cases the remaining unknown part of the mixing matrix is an orthogonal factor $\mathbf{Q}^T$.

So by pre-whitening the data with a whitening transformation such as $\mathbf{W} = \Sigma^{-1}\mathbf{U}^T$, the problem is simplified to the estimation of a new orthogonal mixing matrix $\mathbf{Q}^T$:

$$\mathbf{z} = \mathbf{Wx} = \mathbf{WAs} = \Sigma^{-1}\mathbf{U}^T\mathbf{U}\Sigma\mathbf{Q}^T\mathbf{s} = \mathbf{Q}^T\mathbf{s}, \tag{3-14}$$

of which the inverse is unmixing transformation $\mathbf{Q}$.

This decreases the degrees of freedom of the mixing matrix that needs to be estimated from P^2 to $P(P-1)/2$ which simplifies the overall problem. A second and more practical benefit from the orthogonality constraint is that computing the inverse of the found mixing matrix $\mathbf{A}$ when unmixing the data is as simple as transposing it. Matrix inversion can be a costly process especially for high P so circumventing it is better altogether. Lastly, the number of source components can be deduced from the rank of the covariance matrix $\mathbf{C_x}$ given that the Signal to Noise Ratio (SNR) is high enough.

It is strongly pointed out that pre-whitening does not solve the BSS problem, it simplifies it. An in-depth explanation of why this is the case can be found in [21]. As was mentioned before, statistical independence is a stronger notion than uncorrelatedness [21] and therefore conventional decorrelation methods such as pre-whitening are not an alternative to ICA but rather a helpful pre-processing step.

3-1-4 Gaussian variables in ICA

In short, Gaussian variables are forbidden in ICA [21] due to the CLT which states that the individual source components will be independent if as non-Gaussian as possible. However, if such a component is inherently Gaussian no transformation can be found which maximizes the non-Gaussianity. Thus no proper unmixing transformation can be found for that specific component. For a more detailed explanation the reader is referred to chapter 7.5 from [21].

3-2 Measure of non-Gaussianity with Higher Order Statistics (HOS)

HOS refers to functions and quantities which consist of the expectation of a sample to the power 3 or higher. HOS are particularly useful for characterizing a distributions shape with for example 'skewness' and 'tailedness'. On top of that, certain HOS quantities can be used as a measure of a distributions depart of a Gaussian distribution. In other words, HOS can be used to measure non-Gaussianity. Other measures exist such as negentropy [21]. However, in this literature review only the HOS quantity called Kurtosis is used as in its tensor form it can be diagonalized to obtain the BSS solution.

3-2-1 Kurtosis

The normalized moment of degree 4 of a random variable is known as kurtosis. It describes the 'tailedness' of a probability distribution. Distributions with a high kurtosis will in general show a high peak with thick wide tails whereas low kurtosis usually results in a flattened top with little to no tails. Kurtosis together with excess kurtosis is defined below in 3.4. Moments, centralized moments and standardized moments are all explained in appendix A-2

Definition 3.4: Kurtosis

The **fourth** standardized central moment of a random variable x represents the measure of 'tailedness' of a distribution's shape and is referred to as **kurtosis**:

$$\overline{a}_4 = \frac{a_4}{\sigma^4} = \frac{\mathbb{E}\left[(x - m_x)^4\right]}{\left(\mathbb{E}\left[(x - m_x)^2\right]\right)^{\frac{4}{2}}}. \tag{3-15}$$

Excess kurtosis is defined as the kurtosis difference a distribution has compared to a Gaussian distribution. Gaussian distributions have a kurtosis of 3 so the excess kurtosis is denoted as the kurtosis difference:

$$\tilde{a}_4 = \text{kurt}(x)\frac{a_4}{\sigma^4} - 3 = \frac{\mathbb{E}\left[(x - m_x)^4\right]}{\left(\mathbb{E}\left[(x - m_x)^2\right]\right)^{\frac{4}{2}}} - 3. \tag{3-16}$$

Distributions which have an excess kurtosis of $\tilde{a}_4 = 0$ are called **meso-kurtic**, $\tilde{a}_4 > 0$ are called **lepto-kurtic** and $\tilde{a}_4 < 0$ are called **platy-kurtic**.

Excess kurtosis Excess kurtosis represents the difference in kurtosis of a distribution when compared to a Gaussian distribution of similar scale. A standard Gaussian distribution has a kurtosis of value 3, for this reason the excess kurtosis shown in (3-16) for a standardized distribution is defined as the kurtosis from (3-15) minus 3.

Figure 3-2 shows 3 distributions. The blue distribution denotes a Gaussian distribution which has an excess kurtosis of 0, so it is meso-kurtic. Green denotes a platy-kurtic distribution which has a negative excess kurtosis and orange denotes a distribution where $\tilde{a}_4 > 0$ which is classified as lepto-kurtic. This demonstrates how excess kurtosis can be used as a measure of non-Gaussianity.

3-2-2 Fourth-order Cumulant

An alternative way of describing a probability distribution's shape is through the use of cumulants. Cumulants, are in many aspects similar to moments but especially in the higher-order case they enjoy certain properties such as linearity and additivity which moments do not. Hence the name 'cumulants'. For this report the fourth-order cumulant is of interest as it can be used as a measure of non-Gaussianity which is shown in definition 3.5. For a broader explanation of cumulants and their derivation the reader is referred to appendix A-1.

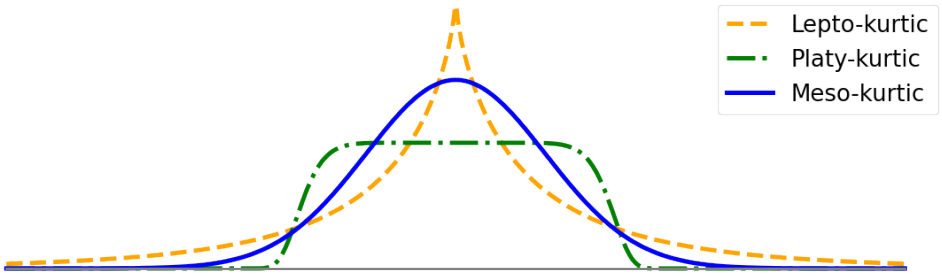


Figure 3-2: A meso-kurtic distribution (blue), a lepto-kurtic distribution (orange dashed) and a platy-kurtic distribution (green dashdotted).

Definition 3.5: Fourth order cumulant

The **fourth** order cumulant of a random variable x is denoted as:

$$\text{cum}(x,x,x,x) = \kappa_4 = \mathbb{E}[(x-m_x)^4] - 3\left(\mathbb{E}[(x-m_x)^2]\right)^2. \tag{3-17}$$

It can easily be shown that when normalized the definition for the 4th order cumulant shown above is identical to that of the earlier defined excess kurtosis from (3-16):

$$\overline{\kappa}_4 = \frac{\kappa_4}{\sigma^4} = \frac{\mathbb{E}[(x-m_x)^4]}{\left(\mathbb{E}[(x-m_x)^2]\right)^2} - 3 = \tilde{a}_4. \tag{3-18}$$

In other words, the fourth order cumulant can be used to describe the departure of a random variable's distribution from that of a Gaussian distribution.

3-2-3 Fourth-order Cumulant tensor

Analogous to how the covariance matrix is formed by computing the covariances of the random variables in the vector $\mathbf{x}$, the 4-dimensional fourth-order cumulant tensor which consists of the fourth-order cross-cumulants can be formed. The definition of this fourth-order cumulant tensor is shown below in definition 3-2-3.

Definition 3.6: Fourth order cumulant tensor [21][44]

The **fourth-order cumulant tensor** $\mathcal{C}_{\mathbf{x}}^{(4)} \in \mathbb{R}^{P\times P\times P\times P}$ is a 4-dimensional **symmetric** tensor that is comprised of the fourth-order cross cumulants of a **zero mean** random vector $\mathbf{x} \in \mathbb{R}^P$. Elementwise the tensor is denoted as:

$$\begin{aligned}\mathcal{C}_{\mathbf{x}}^{(4)}(i_1,i_2,i_3,i_4) =& \mathbb{E}\left[x_{i_1}x_{i_2}x_{i_3}x_{i_4}\right] - \mathbb{E}\left[x_{i_1}x_{i_2}\right]\mathbb{E}\left[x_{i_3}x_{i_4}\right] \\ &- \mathbb{E}\left[x_{i_1}x_{i_3}\right]\mathbb{E}\left[x_{i_2}x_{i_4}\right] - \mathbb{E}\left[x_{i_1}x_{i_4}\right]\mathbb{E}\left[x_{i_2}x_{i_3}\right]\end{aligned} \tag{3-19}$$

where i_1, i_2, i_3, i_4 are the indices of the 4-dimensional tensor ranging from 1 to P. For a non zero mean vector every x_i in (3-19) must be replaced by $(x_i - m_{x_i})\,\forall i \in \{1,2,3,4\}$.

The elements of the cumulant tensor of a **zero-mean** random vector $\mathbf{x}$ with I samples are computed with the following estimations of the joint moments:

$$\begin{aligned}\mathbb{E}\left[x_{i_1}x_{i_2}x_{i_3}x_{i_4}\right] &= \frac{1}{I}\cdot\left(\sum_{\ell=1}^{I}\prod_{k=1}^{4}x_{i_k}(\ell)\right) \\ \mathbb{E}\left[x_{i_m}x_{i_n}\right] &= \frac{1}{I}\cdot\left(\sum_{\ell=1}^{I}x_{i_m}(\ell)x_{i_n}(\ell)\right) \quad \forall m,n \in \{1,2,3,4\}\end{aligned} \tag{3-20}$$

where $x_{i_k}(\ell)$ denotes the ℓ'th sample of the random variable x_{i_k}.
The **computational cost** of forming the tensor is approximately $O(IP^4)$.

The first important property is the cumulant tensor's symmetry shown in property 3.1. This symmetry [55] is often alternatively referred to as super-symmetry. In this report however, the former is used as the term super-symmetric is challenged by Comon et all. [55]. This symmetry can potentially be exploited to compute the tensor more efficiently, as is shown later on. Analogous to how a diagonal covariance matrix represents uncorrelated data, a diagonal cumulant tensor represents statistically independent variables. This property of the cumulant tensor is exploited by the algebraic ICA methods presented later on and is shown in 3.2. Moving on, properties 3.4 and 3.3 state together that the cumulant tensor is blind to Gaussian components. This is related to the previously explained issue of why Gaussian variables do not work in ICA. However, it can be a blessing too as the property implies that Gaussian noise components do not affect the cumulant tensor whatsoever. Seen that noise can be approximated in general as Gaussian noise for practical reasons this is beneficial to solving the BSS problem. Finally, the multi-linearity property from 3.5 is what defines how the diagonalization transformation can be used to unmix the data. Below the properties of interest of the cumulant tensor are given.

Property 3.1: Symmetric [17][44][55][28]

The **symmetry** of the cumulant tensor means that the tensor's elements remain constant under a permutation σ of the indices:

$$\mathcal{C}_{\mathbf{x}}^{(4)}(p_1, p_2, p_3, p_4) = \mathcal{C}_{\mathbf{x}}^{(4)}\sigma(p_1, p_2, p_3, p_4). \tag{3-21}$$

Property 3.2: Diagonal for independent variables [44][55][28]

The **fourth-order cumulant tensor** of a vector $\mathbf{s}$ consisting of statistically independent variables is **diagonal** which means all elements equal 0 except for the elements on the superdiagonal:

$$\mathcal{C}_{\mathbf{x}}^{(4)}(p_1, p_2, p_3, p_4) \neq 0 \quad \text{only if:} \quad p_1 = p_2 = p_3 = p_4. \tag{3-22}$$

Property 3.3: Additivity when independent [17][44]

When two variables or vectors with variables $\mathbf{x}$ and $\mathbf{y}$ are independent, then the cumulant tensor of their sum equals the sum of their individual cumulant tensors:

$$\mathcal{C}_{\mathbf{x}+\mathbf{y}}^{(4)} = \mathcal{C}_{\mathbf{x}}^{(4)} + \mathcal{C}_{\mathbf{y}}^{(4)} \quad \text{only if:} \quad p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}). \tag{3-23}$$

Property 3.4: Blind to Gaussian components [17][44]

A random variable or vector $\mathbf{x}$ which is disturbed by an additive Gaussian vector $\mathbf{d}$:

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{d} \tag{3-24}$$

will have a cumulant tensor which is solely dependent on the cumulant tensor of $\mathbf{x}$:

$$\mathcal{C}_{\hat{\mathbf{x}}}^{(4)} = \mathcal{C}_{\mathbf{x}}^{(4)} + \mathcal{C}_{\mathbf{d}}^{(4)} = \mathcal{C}_{\mathbf{x}}^{(4)} \tag{3-25}$$

due to the value of excess kurtosis for a Gaussian variable being 0 which is identical for the fourth-order cumulant.

Property 3.5: Multi-linearity [17][44][55][28]

If a random vector $\mathbf{x}$ is transformed by a matrix $\mathbf{M}$:

$$\tilde{\mathbf{x}} = \mathbf{M}\mathbf{x}, \tag{3-26}$$

then the cumulant tensor is transformed through the mode-product with $\mathbf{M}$ along all of its 4 modes:

$$\mathcal{C}_{\tilde{\mathbf{x}}}^{(4)} = \mathcal{C}_{\mathbf{x}}^{(4)} \times_1 \mathbf{M} \times_2 \mathbf{M} \times_3 \mathbf{M} \times_4 \mathbf{M}. \tag{3-27}$$

3-3 ICA methods for solving BSS

The general definitions of algebraic and optimization ICA methods are briefly explained and for each definition an algorithm is presented as an example. By comparing the two categories through their respective examples, a clear distinction is made of what separates the two from each other. On top of that the comparison highlights the strengths and weaknesses of each category.

3-3-1 Algebraic methods: COM2

The solution to the BSS problem can be found algebraically [21][17] by diagonalizing a set of HOS matrices [37] or some HOS tensor [58][59]. This thesis considers only the fourth-order cumulant tensor from definition 3-2-3 as in its diagonal state it represents components which are statistically independent. The third order cumulant tensor [58] is a measure of skewness. For any symmetrical distribution it will have a value of 0. Cumulants of order ≥ 5 are more complex to compute and show more sensitivity to outliers in the data [60] than the fourth-order cumulant.

Approximate diagonalization of the fourth-order cumulant tensor

Due to model and sampling errors [21] the fourth-order cumulant tensor cannot be made exactly diagonal. On top of that, no exact tensor diagonalization method exists so diagonalization is performed approximately by optimizing over some cost function, also called a contrast function.

Lemma 3.1 shows the contrast functions for approximate diagonalization of the cumulant tensor itself which are used by algorithms such as COM2 [10]. Other algorithms such as JADE [23] or STOTD [48] jointly diagonalize a set of eigenmatrices or subtensors of the cumulant tensor for which the contrast functions are the sum of the contrasts of the individual matrices or tensors. However, the least squares principle remains the same.

Lemma 3.1: Approximate diagonalization

Let $\mathcal{X} \in \mathbb{R}^{P \times P \times \cdots \times P}$ denote an N-dimensional symmetric cubic tensor, $\mathcal{X}'$ its transformation $\mathcal{X}' = \mathcal{X} \times_1 \mathbf{Q} \times_2 \cdots \times_N \mathbf{Q}$ with $\mathbf{Q}$ an orthogonal transformation matrix and $f(\mathbf{Q})$ the following contrast function:

$$f(\mathbf{Q}) = \| \operatorname{offdiag}(\mathcal{X}') \|_2^2, \tag{3-28}$$

For any matrix $\mathbf{Q}$ that minimizes $f(\mathbf{Q})$, $\mathcal{X}'$ will be an approximate diagonalization of $\mathcal{X}$.

As $\mathbf{Q}$ is orthogonal, this is equivalent to maximizing the values on the superdiagonal of $\mathcal{X}'$:

$$g(\mathbf{Q}) = \| \operatorname{diag}(\mathcal{X}') \|_2^2. \tag{3-29}$$

What these algorithm have in common is that the (sub-)optimal $\mathbf{Q}$ can be found through a sequence of Jacobi rotations. The definition of a Jacobi rotation is given below in 3.7. For a more in-depth explanation on the Jacobi rotation the reader is referred to chapter 5 of [17]. In short, it is designed to eliminate off-diagonal entries through plane rotations. Each k'th rotation is stored by multiplying it with the previous $k-1$ iterations such that the final transformation consists of all rotations combined:

$$\mathbf{Q} = \mathbf{J}_K \cdot \mathbf{J}_{K-1} \cdots \mathbf{J}_2 \cdot \mathbf{J}_1 \tag{3-30}$$

where K denotes the total amount of iterations. The computation of the rotation is performed by finding the roots of a fourth-order polynomial. A detailed explanation the matter can be found in [10].

Definition 3.7: Jacobi rotation [17]

A **Jacobi rotation** matrix $\mathbf{J}(p,q,c,s)$ is an orthogonal similarity transformation of the following structure:

$$\mathbf{J}(p,q,c,s) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & j_{p,p} & \cdots & j_{p,q} & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & j_{q,p} & \cdots & j_{q,q} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \tag{3-31}$$

where $p < q$, $j_{p,p} = j_{q,q} = c$, $j_{p,q} = -s$, $j_{q,p} = \bar{s}$ and $(c,s) \in \mathbb{R} \times \mathbb{C}$ such that $c^2 + |s|^2 = 1$. For a Jacobi rotation matrix of size $P \times P$ there exists:

$$\sum_{n=1}^{P-1} n = \frac{P(P-1)}{2} = \begin{pmatrix} P \\ 2 \end{pmatrix} \tag{3-32}$$

possible rotation pairs where the right-hand notation denotes the binomial coefficients of the polynomial expansion. An explanation of binomial coefficients can be found in appendix B-5.

For the tensor case it cannot be said *a priori* which optimal (p,q) pair maximizes the cost reduction. Therefore the rotation pairs (p,q) are swept through in a cyclic way. This makes it difficult to prove convergence to the global optimum and to study convergence speed of the algorithm. In practice algorithms seem to converge reasonably fast to the solution and local optima and saddle-points do not seem to pose a problem. After convergence, the independent components can be found obtained using theorem 3.2. Note that when $\mathbf{Q} \in \mathbb{R}^{N \times P}$ is non-square and semi-orthogonal with $N < P$ and $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}_N$, application of the theorem results in the estimation of N components.

Theorem 3.2: Component separation through approximate Tensor diagonalization

Let $\mathbf{x} \in \mathbb{R}^P$ be a **zero-mean** pre-whitened random vector with I observations and let $\mathcal{C}_{\mathbf{x}}^{(4)} \in \mathbb{R}^{P \times P \times P \times P}$ be its corresponding fourth-order cumulant tensor. Based on the cumulant tensor properties 3.2, 3.1, 3.5 and lemma 3.1, any **orthogonal** transformation $\mathbf{Q} \in \mathbb{R}^{P \times P}$ that approximately diagonalizes the cumulant tensor through multiplication along all of its modes:

$$\mathcal{C}^{(4)\prime} \approx \mathcal{C}^{(4)} \times_1 \mathbf{Q} \times_2 \mathbf{Q} \times_3 \mathbf{Q} \times_4 \mathbf{Q}, \tag{3-33}$$

will result in P approximately statistically independent variables when multiplied with $\mathbf{x}$:

$$\hat{\mathbf{s}} = \mathbf{Q}\mathbf{x} \tag{3-34}$$

where $\hat{\mathbf{s}} \in \mathbb{R}^P$ denotes an approximation of the original source components $\mathbf{s}$.

Measure of diagonality Because algebraic methods tend to make the cumulant tensor as diagonal as possible it is logical to define a solutions quality through the measure of a tensor's approximate diagonality. A straightforward way of doing so is looking at the fraction of the the norm of the superdiagonal over the norm of the entire tensor. A perfectly diagonal tensor will have a fraction equal to 1. Any tensor less than perfectly diagonal will have a fraction in the range of $[0, 1)$. As such, the following metric is devised to measure a tensor's diagonality.

Definition 3.8: Measure of diagonality

A tensor's **measure of diagonality** is given by the ratio of the norm of its diagonal entries to the norm of all of its entries:

$$\tau_D = \frac{\| \operatorname{diag}(\mathcal{X}) \|_2}{\|\mathcal{X}\|_2} \tag{3-35}$$

where the tensor norm is defined as in definition 2.15. For any given tensor $\tau_D \in [0, 1]$.

3-3-2 Optimization methods: FastICA

Opposed to algebraic methods for which the solution is clearly defined as a diagonal tensor, optimization methods do not have a clear definition of what the solution should look like. What can only be said about the solution is that a variable's absolute kurtosis value should be as high as possible. This can be performed by optimizing over this value.

Devised by Hyvarinen [33], FastICA is a fixed-point method optimization method which optimizes over a contrast function defined usually by an approximation of neg-entropy or excess kurtosis from (3-16). The base optimization problem of which kurtosis based FastICA is derived from is presented below in definition 3.9. The bottom part of the definition shows the fixed-point iteration which FastICA performs.

Definition 3.9: FastICA [33]

The **cost function** of **FastICA** for a **zero-mean** random vector $\mathbf{z} \in \mathbb{R}^P$ with I samples consists of the sum of absolute self-kurtosis values:

$$\begin{aligned} \max_{\mathbf{U}} f(\mathbf{U}) &= \sum_{i=1}^{N} |\operatorname{kurt}(\mathbf{u}_i^T \mathbf{Z})| \quad \text{with} \quad \mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_N \end{bmatrix} \\ \operatorname{kurt}(\mathbf{u}_i^T \mathbf{Z}) &= \frac{1}{I} \mathbf{u}_i^T \mathbf{Z} \left[\mathbf{Z}^T \mathbf{u}_i\right]^3 - 3 \left(\frac{1}{I} \mathbf{u}_i^T \mathbf{Z} \mathbf{Z}^T \mathbf{u}_i\right)^2. \end{aligned} \tag{3-36}$$

The corresponding gradient for each unmixing vector $\mathbf{u}_i$ is defined as:

$$\frac{\partial f}{\partial \mathbf{u}_i} = 4 \cdot \operatorname{sign}(\operatorname{kurt}(\mathbf{u}_i^T \mathbf{Z})) \cdot \left[\frac{1}{I} \mathbf{Z} \left[\mathbf{Z}^T \mathbf{u}_i\right]^3 - 3 \langle \mathbf{u}_i, \mathbf{u}_i \rangle \mathbf{u}_i\right]. \tag{3-37}$$

For **whitened** data the equations are simplified due to the covariance being diagonal $\frac{1}{I} \mathbf{Z} \mathbf{Z}^T = \mathbf{I}$ and the orthogonality constraint on the unmixing matrix $\|\mathbf{u}_i\|^2 = 1$.

In the **fixed-point** iteration that FastICA performs each new estimate of every unmixing vector $\mathbf{u}_i$ is set to be proportionally equal to its gradient:

$$\mathbf{u}_i \propto \frac{1}{I} \mathbf{Z} \left[\mathbf{Z}^T \mathbf{u}_i\right]^3 - 3 \langle \mathbf{u}_i, \mathbf{u}_i \rangle \mathbf{u}_i \tag{3-38}$$

where $\propto$ means proportionally equal to.

Although widely used thanks to its speed and low memory cost, the attractiveness of the algorithm does overshadow its shortcomings and in distinct cases it is proven not to be the best ICA algorithm [1]. For example, in [47] FastICA was found to fail for weak sources or when the sources are highly spatially correlated. It is suggested that functional Magnetic Resonance Imaging (fMRI) data can show high spatial correlation depending on which brain region the data is from [40] and that some ICA methods perform worse when spatial independence is assumed [61]. This implies that FastICA would have trouble with fMRI data. Research on group-subject methods [39] shows that this is not necessarily the case when the proper nonlinearity is chosen with which kurtosis is approximated. However it was found on multiple occasions that FastICA's performance suffers when a low SNR is present [1][47]. Another drawback of FastICA mentioned reccuringly is how its performance depends on a "good"

intialization as saddle-points in the optimization landscape can have detrimental effects on its performance [3][21][1]. Alterations to the algorithm exist which address its shortcomings but are at times paired with some trade-off [21][2][34].

One mentioned trade-off is the rise of additional tuning parameters. For example, kurtosis computed by cumulants can be replaced by some nonlinear approximation which speeds up the algorithm and mitigates the sensitivity to outliers. The trade-off made is that a non-linearity has to be chosen *a priori* for which performances can vary widely [21]. Criteria for the optimal choice of non-linearity have been derived [62] but are dependent on additional statistical assumptions such as the source component's probability density function which further complicates the problem. On top of that, not all nonlinear approximations are blind to Gaussian noise as cumulants are. This means that Gaussian noise can degrade the performance of the algorithm.

In general FastICA and its alterations do provide good results. However, there are some distinct situations such as when a low SNR or many saddle-points in the optimization landscape are present that the performance deteriorates more than other ICA methods. In most cases the best solution is to run the algorithm multiple times with varying settings and initializations. However, this makes it inherently somewhat of a "shotgun-hail" method. Below the points made are summarized and below that the complexities and convergence property of the FastICA algorithm are listed.

FastICA tuning parameters and trade-offs [21]

Measure of non-Gaussianity Different measures of non-Gaussianity can be used for fastICA such as **kurtosis** and **negentropy**. While the former is sensitive to outliers the latter can be very hard to compute exactly. Alternatively, nonlinear functions can be used to approximate either with the trade-off of non-linearity selection. Depending on the choice of non-linearity, the method can loose its property of being blind to Gaussian noise. This proves to be detrimental in situations when a low SNR is present.

Mixing matrix initialization An initial estimate of the mixing matrix has to be given. Due to the non-convexity of the optimization problem as explained in [21] and [17], the performance of FastICA can be heavily dependent on the initial estimate as for example saddle-points have been shown to be detrimental to its performance [3]. This results in FastICA not being robust so "shotgun-hail" tactics have to be used to obtain good results.

FastICA properties

Storage complexity Taken from [2], the storage complexity for memory efficient FastICA is approximately $O(RP)$ where R are the amount of components to be estimated and P are the amount of mixtures used.

Computational complexity In [1] the computational cost per iteration for FastICA is approximated to be $O(IRP)$ where R equals the amount of components to be estimated, P are the amount of mixtures used and I the number of used observations.

Convergence It has been proven that the convergence of FastICA is cubic [21].

3-4 Comparison of algebraic methods with optimization methods

The cost function of FastICA in equation (3-36) can alternatively be defined from a cumulant tensor perspective with either the Tensor Times Same Vector (TTSV) operation from definition 2.16 or a mode-n product sequence:

$$\max_{\mathbf{U}} f(\mathbf{U}) = \sum_{i=1}^{N} |\mathcal{C}_{\mathbf{z}}^{(4)} \mathbf{u}_i^4| = \| \operatorname{diag}\left(\mathcal{C}^{(4)} \times_1 \mathbf{U} \times_2 \mathbf{U} \times_3 \mathbf{U} \times_4 \mathbf{U}\right) \|_1. \tag{3-39}$$

In this regard FastICA can be considered as a cumulant tensor method which only focuses on the entries on the superdiagonal. First of all, this can raise some confusion about the exact difference between what are considered optimization methods and what are considered algebraic methods. While somewhat ambiguous, an actual difference lies in the computation of the unmixing matrix. The Jacobi-type rotations from the COM2 method from definition 3.7 are designed to maximize entries on the superdiagonal while simultaneously eliminating offdiagonal entries. In contrast to this, FastICA only focuses on maximizing the diagonal entries. As a consequence the off-diagonal entries are decreased. However, information present on the off-diagonal entries is never explicitly used when computing the unmixing matrix as is the case with the COM2 algorithm. As a result, algebraic methods such as COM2 can show a lot of consistency in their performances and results [63][26]. This does however depend somewhat on the nature of the BSS problem. Furthermore, algebraic methods have been shown to outperform non-kurtosis based FastICA [50][26] when for example, a low SNR is present for Gaussian noise [63], illustrating the effectiveness of property 3.4 of the cumulant tensor.

Secondly, (3-39) and the gradient in definition 3.9 show that FastICA uses parts of the cumulant tensor in implicit fashion. This suggests that this implicit scheme can also be used to computed parts of the entire tensor for algebraic methods.

3-4-1 Issue of algebraic methods: curse of dimensionality

The downside of algebraic methods becomes apparent when comparing the storage and computational cost with that of FastICA. Table 3-1 illustrates the storage, computational and cumulant tensor forming cost of the algebraic method COM2 against FastICA.

	Costs		
	Initial	Per iteration	Storage
COM2	IP^4	P^5	P^4
FastICA	-	IRP	RP

Table 3-1: The dominant initiation, storage and computational complexities of the COM2 and parallel kurtosis-based FastICA algorithms for R components to be estimated with I observations and P mixtures. For COM2 the initial cost consists of computing the cumulant tensor. For COM2 the storage needed is $O(P^4)$ for the cumulant tensor. For FastICA the storage needed is dominated by the orthogonal transformation and the gradient. For ease of analysis the computational complexities are solely determined through the amount of multiplications. The complexities for parallel FastICA are taken from [1][2][3].

The first column shows the initial onetime cost each algorithm has. For COM2 this consists only of forming the cumulant tensor. For the algebraic method the initial cost scales up quartically whereas FastICA has none. The second column shows that the cost COM2 is dominated by the size of the cumulant tensor which increases quartically with the amount of components P. The middle column of Table 3-1 shows the cost per iteration of each algorithm. The cost for FastICA scales woth I, $R$and P whereas the algebraic methods only scale with P. However, it is impossible to know beforehand how many iterations K are needed for convergence so nothing can be directly stated about the total computational cost. HComparing the per iteration cost does clearly show that the algebraic methods suffer from higher-order powers of P. Taking into consideration how the amount of possible rotation pairs grows according definition 3.7, the amount of iterations needed for the algebraic method grows too in polynomial fashion which adds up even further to the already growing per iteration computational cost.

3-5 Chapter summary and contributions

This chapter serves as an introduction to the concept of ICA and its sub-classes: optimization based methods and algebraic methods. As was shown there is some ambiguity present concerning this classification as most algebraic methods are based on an optimization process too. The cost functions of FastICA and COM2 can both be written as the maximization of an L-norm of the diagonal entries, however the main difference was found to be whether a method uses off-diagonal cumulant tensor entries or not. Literature suggests the superiority of algebraic methods when considering robustness due to use of off-diagonal entries. However, the use of off-diagonal entries results in general in many more iterations needed by the algebraic methods, which in combination with the high cumulant tensor forming and handling costs results in poor scaling with the amount of mixtures P used.

All of the above is listed below as a set of concrete contributions.

3.1: Contributions chapter 3

- All necessary **basic knowledge** concerning ICA is written in compact and logically structured fashion such that newcomers to the topic may understand its basic principles.
- Through the use of examples the difference between **optimization based** and **algebraic** ICA methods is explained.
- The benefits and drawbacks of both optimization based and algebraic ICA methods in general are listed and explained through examples.
- It is shown how through the **curse of dimensionality** algebraic methods scale generally speaking in computation cost with $O(IP^4)$ and in storage cost with $O(P^4)$.

Chapter 4

Implicit tensor decomposition for Independent Component Analysis (ICA)

The manipulating and storing of the fourth-order cumulant tensor suffers from the curse of dimensionality due to it being quartic in size. For this reason it is natural to consider tensor decomposition as a solution of lifting this curse. Different tensor decomposition formats have varying useful properties which for example allow for data compression, more efficient tensor manipulation and tensor diagonalization. Several formats have already found their use in ICA. However, for the cumulant tensor these potential benefits come at the cost of having to first explicitly form and store the tensor. By itself this can be reason enough to avoid using the tensor altogether. However, by exploiting the cumulant tensor's properties varying tensor decomposition formats can be computed implicitly. Meaning that the tensor formation and storing step is bypassed. The result of doing so is the creation of a set of implicit tensor decomposition algorithms which can be used to find a solution for ICA.

Chapter outline In this section we present how three decomposition formats of the cumulant tensor can be computed implicitly at lower costs than doing so explicitly. Their working principles together with their potential use for ICA are discussed. Before presenting each implicit decomposition, we give a general description the format together with its history and use concerning ICA. Alongside this we make important observations between cumulant tensor diagonality and tensor ranks which are used as the rational for Higher-Order Singular Value Decomposition (HOSVD) as a preprocessing step.

4-1 Tucker Decomposition

First the already existing Tucker decomposition is shown. The formal definition of a Tucker decomposition is presented below in definition 4.1. Tucker decomposition refers to the general format where a tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times \dots P_N}$ is decomposed into a core tensor $\mathcal{G}$ and factor matrices $\{\mathbf{U}_1, \dots, \mathbf{U}_N\}$. Figure 4-1 shows a graphical representation of how a 3-way tensor is decomposed into a core tensor $\mathcal{G}$ and its 3 factor matrices $\mathbf{U}_i \quad \forall i = 1, 2, 3$.

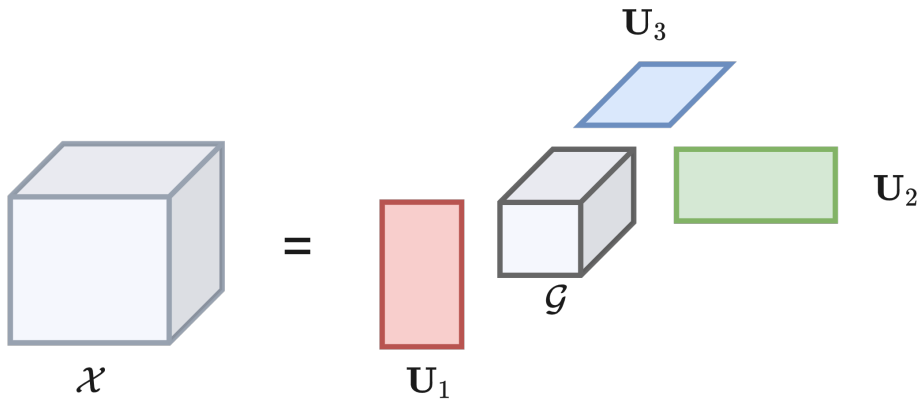


Figure 4-1: Tucker decomposition of a 3-way tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$ into a tensor core $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ together with 3 factor matrices $\mathbf{U}_i \in \mathbb{R}^{P_i \times R_i} \quad \forall i = 1, 2, 3$.

Definition 4.1: Tucker decomposition [28]

The **Tucker decomposition** of a tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times \dots \times P_N}$ consists of a tensor core $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \dots \times R_N}$ which is multiplied along each of its modes $\{1, \dots, N\}$ with a corresponding factor matrix $\mathbf{U}_i = \begin{bmatrix} \mathbf{u}_{i,1} & \dots & \mathbf{u}_{i,R_i} \end{bmatrix} \in \mathbb{R}^{P_i \times R_i} \quad \forall i = 1, \dots, N$:

$$\begin{aligned} \mathcal{X} &= [\![\mathcal{G}; \mathbf{U}_1, \cdots, \mathbf{U}_N]\!] = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \cdots \times_N \mathbf{U}_N \\ &= \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} \mathcal{G}(r_1, \cdots, r_N) \cdot \mathbf{u}_{1,r_1} \circ \cdots \circ \mathbf{u}_{N,r_N}, \end{aligned} \tag{4-1}$$

where the $[\![\mathcal{G}; \mathbf{U}_1, \cdots, \mathbf{U}_N]\!]$ notation denotes a tensor core $\mathcal{G}$ that is multiplied along its modes with the factor matrices $\{\mathbf{U}_1, \dots, \mathbf{U}_N\}$.
The **Tucker rank** is the N-tuple consisting of the n-ranks R_n.

Tucker decomposition forms the baseline of the decomposition formats used throughout this thesis. Each of these formats is a special case of Tucker decomposition with specific constraints.

4-2 Higher-Order-Singular-Value-Decomposition HOSVD

Higher-Order-Singular-Value-Decomposition (HOSVD) is a special case of Tucker decomposition where the factor matrices and the core are constrained to be orthogonal. It is implied that it actually is a form of higher-order Principal Component Analysis (PCA) [64] due to the Singular-Value-Decomposition (SVD) along each tensor mode. The HOSVD of a symmetric tensor has the properties that the resulting core is symmetric too and that all factor matrices are identical. The latter means that the SVD of only a single mode has to be computed which lowers the computational complexity of the decomposition. The definition of a symmetric HOSVD is given in definition 4.2. The self derived storage and computational complexities of the symmetric HOSVD are shown in 4.1.

Definition 4.2: HOSVD of a symmetric tensor [64]

The **HOSVD** of a **symmetric** tensor $\mathcal{X} \in \mathbb{R}^{[N,P]}$ consists of a symmetric tensor core $\mathcal{G} \in \mathbb{R}^{[N,R]}$ which is multiplied along each of its modes $\{1, \ldots, N\}$ with a single factor matrix $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \ldots & \mathbf{u}_R \end{bmatrix} \in \mathbb{R}^{P \times R}$:

$$\begin{aligned} \mathcal{X} &= [\![\mathcal{G}; \mathbf{U}, \cdots, \mathbf{U}]\!] = \mathcal{G} \times_1 \mathbf{U} \times_2 \cdots \times_N \mathbf{U} \\ &= \sum_{r_1=1}^{R} \cdots \sum_{r_N=1}^{R} \mathcal{G}(r_1, \cdots, r_N) \cdot \mathbf{u}_{r_1} \circ \cdots \circ \mathbf{u}_{r_N}, \end{aligned} \tag{4-2}$$

The semi-orthogonal factor matrix $\mathbf{U}$ with $R \leq P$ has the property of being left-invertible $\mathbf{U}^T\mathbf{U} = \mathbf{I}_R$. As all factor matrices are identical the core $\mathcal{G}$ of a symmetric HOSVD is symmetric too.

Symmetric HOSVD properties

Storage complexity The storage complexity of the HOSVD of a symmetric tensor $\mathcal{X} \in \mathbb{R}^{[N,P]}$ is $O\left(R^N + PR\right)$ where R is the Tucker-rank.

Computational complexity The computational complexity of symmetric HOSVD of a symmetric tensor $\mathcal{X} \in \mathbb{R}^{[N,P]}$ is dominated by the computation a single SVD $O(P^{N+1})$.

Computing the symmetric HOSVD of a tensor amounts to performing SVD on a single mode-n matricization. This is illustrated in figure 4-2 for a third-order tensor $\mathcal{X}$. After the SVD the diagonal matrix $\Sigma \in \mathbb{R}^{R \times R}$ is multiplied with the right-orthogonal matrix $\mathbf{V}^T \in \mathbb{R}^{R \times R^2}$ which is then reshaped into the HOSVD core $\mathcal{G} \in \mathbb{R}^{R \times R \times R}$.

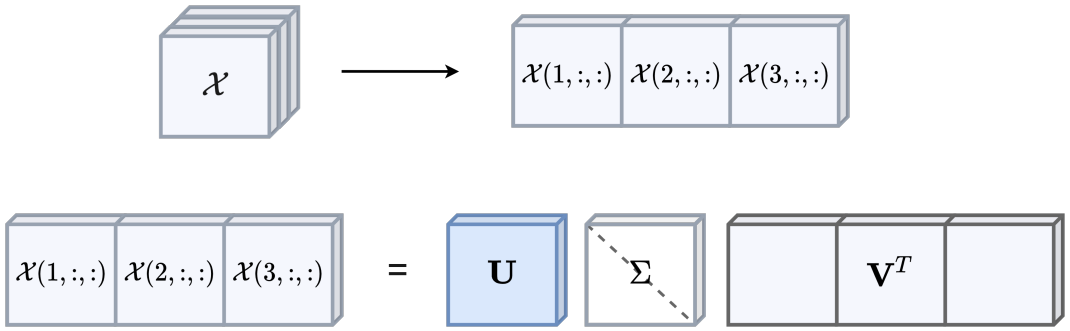


Figure 4-2: The computation of the factor matrix $\mathbf{U}$ of a symmetric third-order tensor $\mathcal{X}$.

HOSVD of the cumulant tensor Decomposing the fourth-order cumulant tensor with HOSVD results in the following symmetric decomposition from definition 4.2:

$$\mathcal{C}_{\mathbf{x}}^{(4)} = \mathcal{G}_{\mathcal{C}_{\mathbf{x}}^{(4)}} \times_1 \mathbf{U} \times_2 \mathbf{U} \times_3 \mathbf{U} \times_4 \mathbf{U}. \tag{4-3}$$

Comparing this expression with the multilinearity property 3.5 of the cumulant tensor shows that the inverse of the orthogonal factor matrix $\mathbf{U}$, equal to its transpose, can be applied as a un-mixing transformation $\mathbf{Q} = \mathbf{U}^T$ on the observed mixtures:

$$\hat{\mathbf{s}} = \mathbf{Q}\mathbf{x} = \mathbf{U}^T\mathbf{x}. \tag{4-4}$$

The HOSVD core $\mathcal{G}_{\mathcal{C}_{\mathbf{x}}^{(4)}}$ is in general not diagonal which means that this is not an approximate diagonal transformation of the cumulant tensor. By theorem 3.2 the estimate $\hat{\mathbf{s}}$ of the true sources $\mathbf{s}$ will not have statistical independent components and by definition fails to be an effective ICA method.

Experimental results show that although HOSVD is in general not able to properly separate mixed components [52][65], it does provide a useful starting estimate of the unmixing matrix. On top of that, it is often used too as the starting estimate for a Tucker decomposition in which the resulting core is constrained to be diagonal. We later show how computing a diagonally constrained Tucker decomposition of the cumulant tensor can benefit from preprocessing it with HOSVD.

One key issue concerning the computation of the HOSVD of the cumulant tensor is the fact that a matricized tensor of size P^4 has to be processed and stored. On top of that, there is no guarantee that the Tucker-rank of the resulting symmetric core is any smaller than P.

In the following section we present an algorithm which exploits the structure of the cumulant tensor such that its HOSVD can be computed iteratively without surpassing a storage requirement of P^2 at any given iteration.

4-2-1 Computing the HOSVD implicitly

In this section we present an algorithm which allows for implicit computation of the whitened fourth-order cumulant tensor HOSVD. The main benefit of doing so related to ICA is the computation of the factor matrix $\mathbf{U}$ which can be used as an initial estimate for ICA and Canonical Polyadic Decomposition (CPD) algorithms. Due to not having to explicitly compute and store the cumulant tensor $\mathcal{C}^{(4)} \in \mathbb{R}^{[4,P]}$ and the HOSVD core $\mathcal{G} \in \mathbb{R}^{[4,R]}$, the storage cost of the algorithm never exceeds $O(P^2)$. The algorithm for implicit HOSVD of the cumulant tensor is presented below and its complexities we derived are presented in 4.2.

Algorithm 1 Implicit HOSVD

Require: whitened data: $\mathbf{Z} \in \mathbb{R}^{P \times I}$

procedure IMPLICIT HOSVD($\mathbf{Z}$)
  for $i = 1, \ldots, P$ **do**
    for $j = i, \ldots, P$ **do**
      $\mathbf{W} = \text{zeros}(\text{size} = [P, P]), \quad \mathbf{W}(i, j) = 1$
      $\mathbf{M}_{eye} = (\mathbf{I}_P(:,:) \cdot \mathbf{I}_P(i,j)) + \mathbf{W} + \mathbf{W}^T$
      $\mathbf{M}_{ij} = \frac{1}{I}\mathbf{Z}(\mathbf{Z}(i,:) * \mathbf{Z}(j,:) * \mathbf{Z})^T - \mathbf{M}_{eye}$
      if $i = 1$ and $j = 1$ **then**
        $\mathbf{U}, \Sigma, \mathbf{V}^T \leftarrow \text{SVD}(\mathbf{M})$
      else
        if $i = j$ **then**
          $\mathbf{U}, \Sigma \leftarrow \text{SVD-update}(\mathbf{M})$
        else
          $\mathbf{U}, \Sigma \leftarrow \text{SVD-update}(\begin{bmatrix} \mathbf{M} \end{bmatrix})$ twice
        end if
      end if
    end for
  end for
  return $\mathbf{U}, \Sigma$
end procedure

PROPERTY 4.2: IMPLICIT HOSVD PROPERTIES

Computational complexity For the cumulant tensor $\mathcal{C}^{(4)} \in \mathbb{R}^{[4,P]}$ the dominant computational costs of the implicit HOSVD algorithm are the computation of all slices $O(\frac{IP^3(P+1)}{2})$ and for $P >> 0$, the cost of performing all svd-updates $O((P^3 - 1) \cdot P^2(\log(P)^2) \approx O(P^5(\log(P)^2)$.

Storage complexity The storage complexity of the algorithm is $O(P^2)$.

At the heart of implicit HOSVD lies a stable algorithm presented in [66] for updating an already existing SVD. By iteratively computing the cumulant tensor's slices the SVD of its matricization is continuously updated. Furthermore, the algorithm benefits from the tensor's symmetry by computing fewer slices. Both are explained in the following sections together with an explanation on how slices of the cumulant tensor can be computed separately.

Iterative SVD The core SVD-update method utilized by our Iterative-implicit-HOSVD algorithm is presented in [66]. To update an already existing SVD of a square matrix $\mathbf{M}_1 = \mathbf{U}\Sigma\mathbf{V}^T$ by appending a vector to it as $\mathbf{M}_2 = \begin{bmatrix} \mathbf{M} & \mathbf{z} \end{bmatrix}$, the algorithm from [66] updates the singular values by finding the eigenvalues of

$$\mathbf{M}'\mathbf{M}'^T \begin{bmatrix} \Sigma & \mathbf{z} \end{bmatrix} \begin{bmatrix} \Sigma \\ \mathbf{z}^T \end{bmatrix}. \tag{4-5}$$

The eigenvalues are found using the Fast Multipole method (FMM) proposed in [67][68]. For more in-depth information on the FMM the reader is directed to said references. The used implementation of this method taken from [69] contains several alterations and additional techniques presented in [70] and [71] which allow for a faster computation of sequential updates at a computational cost of $O(P^2(\log(P)^2)$ per update. The complete problem the SVD-update method solves can be found in appendix C-3. Figure 4-3 illustrates how the SVD-update method works for a matricized third-order cubic tensor $\mathcal{X} \in \mathbb{R}^{[3,P]}$.

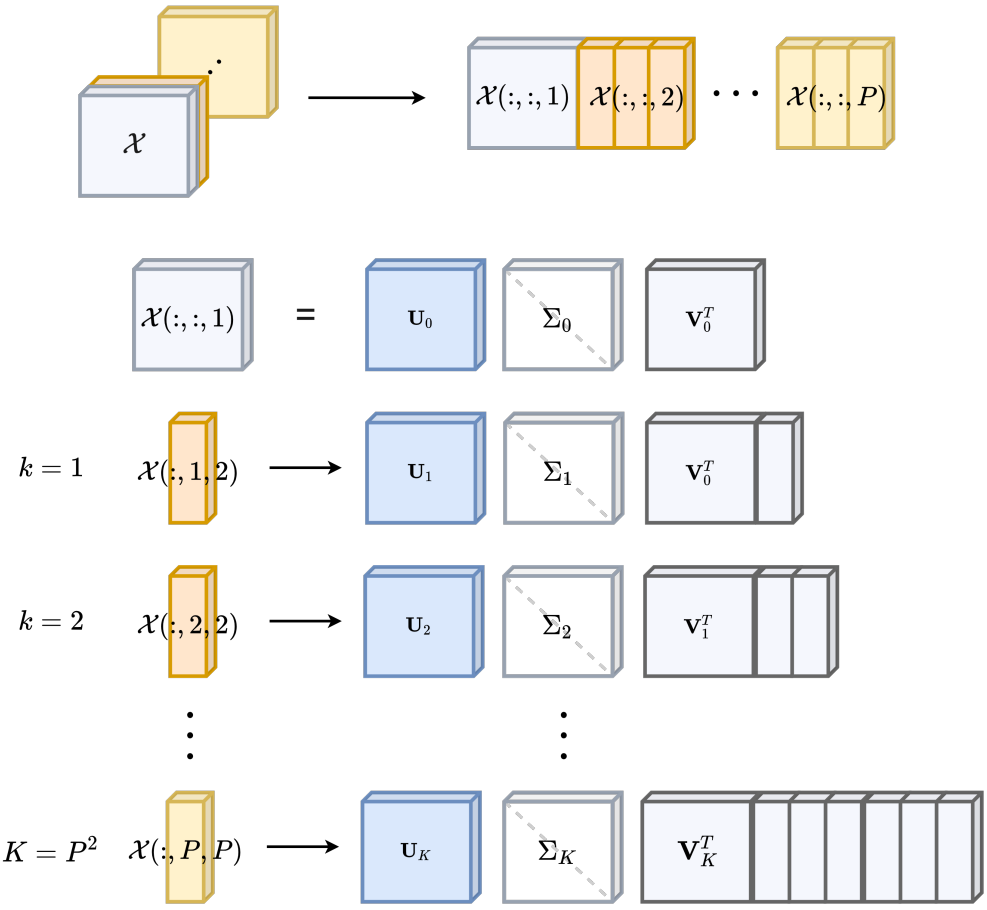


Figure 4-3: Iterative SVD of a matricized 3-way tensor $\mathcal{X} \in \mathbb{R}^{[3,P]}$. The tensor $\mathcal{X} \in \mathbb{R}^{[3,P]}$ is first matricized along mode 1 (top row) after which the HOSVD is initiated with an SVD of the first frontal slice $\mathcal{X}(:,:,1)$ (second row). The SVD is updated $P(P-1)$ times with the set of fibers $F = \{x_{:ij} \quad \forall i,j \in \{1,\ldots,P\}\}$.

The algorithm is initialized with the SVD of the first frontal slice $\mathcal{X}(:,:,1)$ after which the SVD is updated $P(P-1)$ times with the $P(P-1)$ remaining columns. The matrix $\mathbf{V}^T$ containing the right singular vectors increases in size according to the size of the appended decomposed matrix. However, the SVD-update algorithm from [66] only needs to update the singular vectors which results in only the singular values and the left orthogonal matrix $\mathbf{U} \in \mathbb{R}^{P \times P}$ needed to be stored at a cost of $O(P^2)$. After the SVD has been updated with all of the cumulant tensor slices the orthogonal matrix $\mathbf{U}$ can be then used as an initial estimate for ICA or CPD.

In order to benefit from the lower storage cost of the iterative SVD, the cumulant tensor slices have to be computed and processed in the SVD-update iteratively. We present a partial cumulant tensor computation scheme with which parts of the cumulant tensor of arbitrary size can be computed. This partial computation scheme is the cornerstone of the implicit methods we propose in this thesis.

Cumulant tensor slice computation Here we present how any part of the cumulant tensor can be computed separately. The same method can be used to compute single vectors or values of the tensor. For the sake of simplicity and for the added benefit of an orthogonality constraint on the solution, only whitened data $\mathbf{Z} \in \mathbb{R}^{P \times I}$ is considered here when computing the cumulant tensor which is defined as follows.

Definition 4.3: Fourth-order cumulant tensor for whitened data

The **fourth-order** cumulant tensor for a **whitened** random vector $\mathbf{z} \in \mathbb{R}^P$ with I samples in its full tensor form is expressed as:

$$\begin{aligned} \mathcal{C}_{\mathbf{z}}^{(4)} &= \mathcal{M}_{\mathbf{z}}^{(4)} - \mathcal{T}_{\mathbf{C}_{\mathbf{z}}^{(2)}} \\ \text{with} \quad \mathcal{T}_{\mathbf{C}^{(2)}\mathbf{z}} &= \mathbf{C}_{\mathbf{z}}^{(2)} \circ \mathbf{C}_{\mathbf{z}}^{(2)} + \left(\mathbf{C}_{\mathbf{z}}^{(2)} \circ \mathbf{C}_{\mathbf{z}}^{(2)}\right)^{T\sigma_1} + \left(\mathbf{C}_{\mathbf{z}}^{(2)} \circ \mathbf{C}_{\mathbf{z}}^{(2)}\right)^{T\sigma_2} \\ \sigma_1 &= [1,3,2,4], \quad \sigma_2 = [1,4,3,2] \end{aligned} \tag{4-6}$$

where the fourth order moment tensor $\mathcal{M}_{\mathbf{x}}^{(4)}$ and the covariance $\mathbf{C}_{\mathbf{z}}^{(2)}$ are computed using the elementwise operation defined in 3-20.

The tensor $\mathcal{T}_{\mathbf{C}_{\mathbf{z}}^{(2)}}$ can be equivalently written by replacing $\mathbf{C}_{\mathbf{z}}^{(2)}$ with an identity matrix $\mathbf{I}_P$ due to whiteness:

$$\begin{aligned} \mathcal{T}_{\mathbf{C}_{\mathbf{z}}^{(2)}} = \mathcal{T}_{\mathbf{I}} &= \mathbf{I}_P \circ \mathbf{I}_P + (\mathbf{I}_P \circ \mathbf{I}_P)^{T\sigma_1} + (\mathbf{I}_P \circ \mathbf{I}_P)^{T\sigma_2} \\ \sigma_1 &= [1,3,2,4], \quad \sigma_2 = [1,4,3,2]. \end{aligned} \tag{4-7}$$

Due to whiteness the second-order part $\mathcal{T}_{\mathbf{I}}$ of (4-7) is a sparse tensor in which the nonzero entries are symmetrically placed. The structure of this tensor is created through the summation of index permutated outer-products of identity matrices.

To better understand how to navigate through the slices of a 4-way tensor $\mathcal{T} \in \mathbb{R}^{P_1 \times P_2 \times P_3 \times P_4}$, the tensor can be considered as a set of P_4 3-way tensors which is represented in figure Figure 4-4. Arrows indicate how the indices $p_i \quad \forall i \in \{1,2,3,4\}$ move along this set of 3-way tensors.

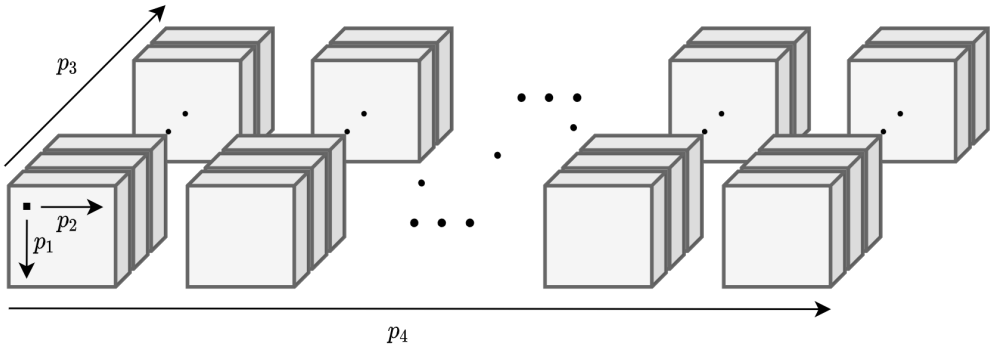


Figure 4-4: Visual representation of a 4-way tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times P_3 \times P_4}$ as P_4 3-way tensors $\mathcal{X}_{sub} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$. The arrows p_1, p_2, p_3 and p_4 indicate how to traverse along the original tensors indices through the representation.

Alternatively, Figure 4-4 can be considered as a set of $P_3 \times P_4$ slices where each slice is defined by $p_i \in \{1, \dots, P_i\} \quad \forall i \in \{1, 2\}$. When matricizing the cumulant tensor these slices are stacked columnwise next to each other. Due to the matricization each element in the 4-dimensional space $\mathbb{R}^{P_1 \times P_2 \times P_3 \times P_4}$ at the indices (p_1, p_2, p_3, p_4) is mapped to a 2-dimensional space $\mathbb{R}^{P_1 \times P_2 P_3 P_4}$ where its indices are defined as $(p_1, p_4 P_3 P_2 + p_3 P_2 + p_2)$.

For the cumulant tensor $\mathcal{C}^{(4)} \in \mathbb{R}^{[4,P]}$ any of its mode-n matricizations is defined as follows:

$$\mathbf{C}^{(4)}_{(n)} = \mathbf{M}^{(4)}_{\mathbf{Z}(n)} - \mathbf{T}_{\mathbf{I}(n)} \quad \text{with:} \quad \mathbf{M}^{(4)}_{\mathbf{Z}(n)} = \frac{1}{I}\mathbf{Z}\left(\mathbf{Z} \odot \mathbf{Z} \odot \mathbf{Z}\right)^T. \tag{4-8}$$

Lemma 4.1 shows how to navigate through the matricized cumulant tensor according to its 4-way indices based on the definition of mode-n matricization 2.12.

Lemma 4.1: Indices of matricized cumulant tensor

For any entry of the **fourth-order cumulant tensor** $\mathcal{C}^{(4)} \in \mathbb{R}^{[4,P]}$ at the **indices** (p_1, p_2, p_3, p_4), the following equivalence holds true for any mode-n **matricization** $\mathbf{C}^{(4)}_{(n)}$ of the tensor:

$$\mathcal{C}^{(4)}(p_1, p_2, p_3, p_4) = \mathbf{C}^{(4)}_{(n)}(p_1, p_4 P^2 + p_3 P + p_2). \tag{4-9}$$

Using (4-8) and lemma 4.1 navigation through the fourth-moment tensor $\mathcal{M}^{(4)}_{\mathbf{z}}$ is defined as follows. For $\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T & \mathbf{z}_2^T & \dots & \mathbf{z}_P^T \end{bmatrix}^T, \quad \mathbf{z}_i \in \mathbb{R}^{1 \times I} \quad \forall i \in \{1, \dots, P\}$ the value of the original tensor at the indices (p_1, p_2, p_3, p_4) corresponds to the following:

$$\mathcal{M}^{(4)}_{\mathbf{Z}}(p_1, p_2, p_3, p_4) = \mathbf{M}^{(4)}_{\mathbf{Z}(n)}(p_1, p_4 P^2 + p_3 P + p_2) = \frac{1}{I}\mathbf{z}_{p_1}\left(\mathbf{z}_{p_4} \odot \mathbf{z}_{p_3} \odot \mathbf{z}_{p_2}\right)^T. \tag{4-10}$$

With identical reasoning a similar notation can be found for the sparse tensor $\mathbf{T}_{\mathbf{I}(1)}$. This means that any of the $P_3 \times P_4$ slices of the tensor can be computed using definition 4.4.

Definition 4.4: Partial cumulant tensor computation

For $\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T & \mathbf{z}_2^T & \dots & \mathbf{z}_P^T \end{bmatrix}^T$, $\mathbf{z}_i \in \mathbb{R}^{1\times I} \quad \forall i \in \{1,\dots,P\}$, the cumulant tensor **entry** at (p_1,p_2,p_3,p_4) can be computed as follows:

$$\mathcal{C}^{(4)}_{\mathbf{Z}}(p_1,p_2,p_3,p_4) = \frac{1}{I}\mathbf{z}_{p_1}\left(\mathbf{z}_{p_4} * \mathbf{z}_{p_3} * \mathbf{z}_{p_2}\right)^T - \mathcal{T}_{\mathbf{I}}(p_1,p_2,p_3,p_4) \quad \text{with}$$
$$\mathcal{T}_{\mathbf{I}}(p_1,p_2,p_3,p_4) = \mathbf{I}(p_1,p_2)\circ\mathbf{I}(p_3,p_4) + \mathbf{I}(p_1,p_3)\circ\mathbf{I}(p_2,p_4) + \mathbf{I}(p_1,p_4)\circ\mathbf{I}(p_3,p_2). \tag{4-11}$$

A **vector** at the indices $(:,p_2,p_3,p_4)$ can be computed by varying $p_1 = 1,\dots,P$, and a **slice** at the indices $(:,:,p_3,p_4)$ can be computed by varying $p_1,p_2 = 1,\dots,P$.

Now that we now how to compute the cumulant tensor slices separately we show how to lower the computational cost of the implicit HOSVD by exploiting the cumulant tensor's symmetry.

Exploiting symmetry Due to the cumulant tensor's symmetry not all slices from definition 4.4 are unique. The equivalence between slices is visually represented using colors in Figure 4-5. the figure shows that out of all P^2 slices, only $\frac{P(P+1)}{2}$ are unique.

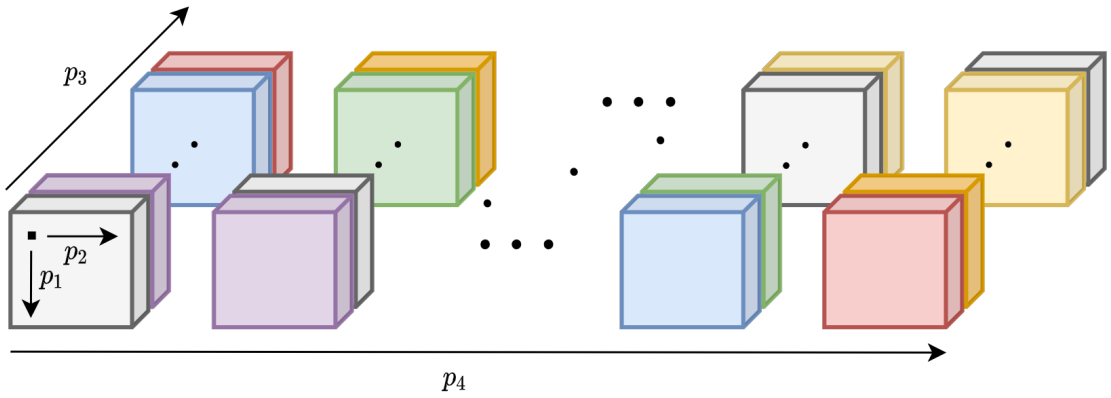


Figure 4-5: Visualization of identical slices of the cumulant tensor due to symmetry through representation with colors.

The implicit HOSVD algorithm from 1 exploits this symmetry by only computing each unique slice once. This is visually represented in figure 4-6. In order to still compute the proper HOSVD the SVD is updated twice with the columns of each unique slice with multiplicity 2, denoted in the figure with the color blue. This reduces the cost of computing the tensor slices from $O(IP^4)$ to $O(\frac{IP^3(P+1)}{2})$.

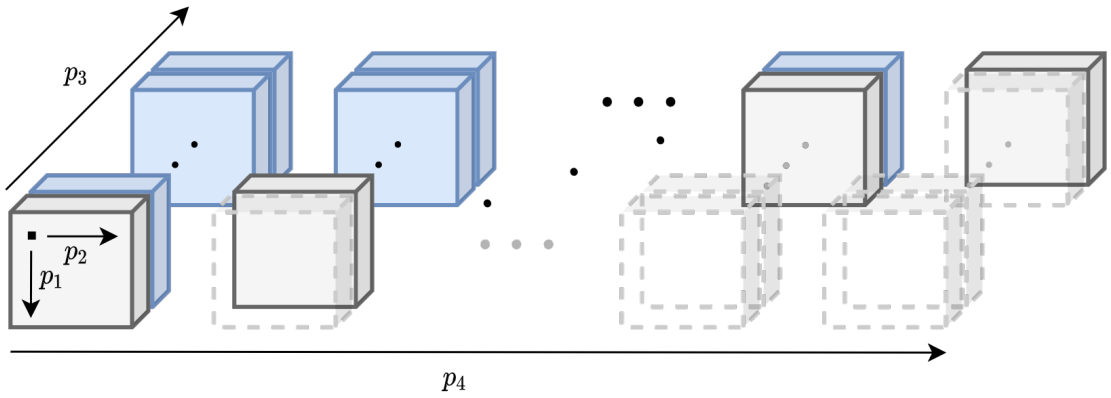


Figure 4-6: Visualization of the unique slices of the cumulant tensor. Computation of all slices with dashed outlines are skipped by updating the SVD twice with the columns of the slices in blue.

U-Implicit HOSVD and C-Implicit HOSVD

The implicit HOSVD still requires the computation of the entire cumulant tensor. On top of that, the total computational cost of all svd-updates is more expensive than performing a single svd on the matricized tensor. For these reasons the following 2 versions of the implicit algorithm are proposed which use different sub-selections of the cumulant tensor slices. It is important to note that these versions of the algorithm do not produce a correct HOSVD as not all information is processed. The rational behind these versions is that perhaps using only part of the cumulant tensor can be enough to find a good initial estimate for ICA algorithms while lowering overall computational complexity.

U-Implicit HOSVD The U-Implicit HOSVD version updates the svd only once with each unique slice from Figure 4-6. This means that the total computational cost of the svd-updates is brought down to $O((\frac{P(P+1)}{2} - 1) \cdot P \cdot P^2(\log(P)^2) \approx O(\frac{P^5}{2}(\log(P)^2)$

C-Implicit HOSVD The C-Implicit HOSVD version only uses the "core" slices of the cumulant tensor. Our definition for core slices is given as follows.

Definition 4.5: Core slices of a symmetric tensor

The **core** slices of a symmetric N-way tensor $\mathcal{X} \in \mathbb{R}^{[N,P]}$ are defined as all slices for which the following holds true:

$$\mathcal{X}(:,:,p_3,\ldots,p_N) \quad \text{with} \quad p_3 = \ldots = p_N \quad \forall p_3 = 1,\ldots,N. \tag{4-12}$$

Note that due to symmetry any permutation of the indices will result in the same set of slices.

For the cumulant tensor these are the slices for which $p_3 = p_4 \quad \forall p_4 = 1, 2, \ldots, P$. The core slices are shown below in Figure 4-7 in blue. the total svd-update cost for this version is $O((P-1) \cdot P \cdot P^2(\log(P)^2) \approx O(P^4(\log(P)^2)$.

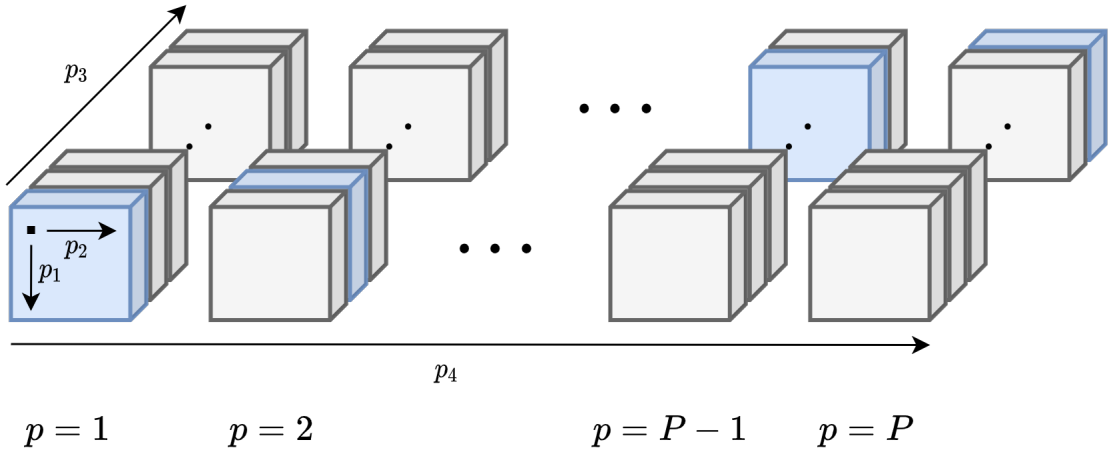


Figure 4-7: Slice selection of the 4-way cumulant tensor using previously introduced visual representation. Each slice in blue is computed at the value of p indicated below.

4-3 Tucker decomposition through a QR Tensor method

In this section we present a novel algorithm together with its implicit version for the cumulant tensor which allows for the computation of a Tucker decomposition with specific structural properties. Inspired by the QR algorithm for matrices, the presented algorithm computes and applies successive QR-decompositions on a tensor in order to transform it. The algorithm is presented below in pseudo-code. The algorithm has a lot of similarity with an already existing method for computing the real eigenpairs of a symmetric tensor [72]. However, instead of iterating through entire slices as is done in [72], the slices used by algorithm 2 for QR-decomposition decrease in size.

Algorithm 2 QRT

Require: N-dimensional tensor: $\mathcal{T} \in \mathbb{R}^{[P,N]}$, max iterations: K_{max}

procedure QR-T($\mathcal{T}$)
  $\mathbf{Q}_{store} = \mathbf{I}_P$
  for $p = 1, \ldots, P-1$ **do**
    for $k = 1, \ldots, K_{max}$ **do**
      $\mathbf{Q}' = \text{QR-decomposition}\left(\mathcal{T}\left(p:P, p:P, p, \ldots, p\right)\right)$
      $\mathbf{Q} = \begin{bmatrix} I_{p-1} & 0 \\ 0 & \mathbf{Q}' \end{bmatrix} \in \mathbb{R}^{P \times P}$
      $\mathbf{Q}_{store} \leftarrow \mathbf{Q}\mathbf{Q}_{store}$
      $\mathcal{T} \leftarrow \mathcal{T} \times_1 \mathbf{Q} \times_2 \ldots \times_N \mathbf{Q}$
    end for
  end for
  return $\mathcal{T}, \mathbf{Q}_{store}$
end procedure

We remind the reader of the definition of "core" slices shown in definition 4.5. The core slices the QRT algorithm uses for a 4-way tensor are shown below in Figure 4-8.

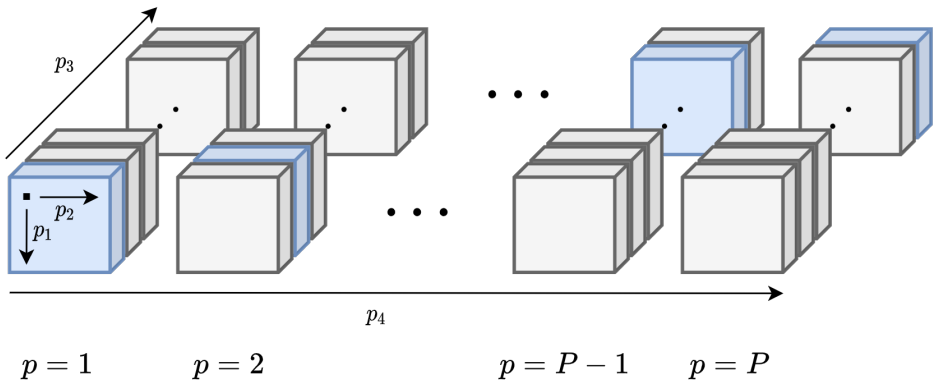


Figure 4-8: Slice selection of the 4-way cumulant tensor using previously introduced visual representation. Each slice in blue is computed at the value of p indicated below.

4-3-1 The QR-algorithm for tensors

Our inspiration for the QRT alorithm comes from the QR algorithm. The QR algorithm or QR iteration, developed simultaneously but independently by John G. F. Francis and by Vera N. Kublanovskaya [73][74][75], is a procedure to calculate the eigendecomposition of a matrix. The QR algorithm is shown below in definition 4.6.

Definition 4.6: QR algorithm [73][74][75]

The **eigenvalue** decomposition of a **symmetric** matrix $\mathbf{M} \in \mathbb{R}^{P \times P}$ can be computed using a sequence of **QR decompositions**. After computing the **QR decomposition** of $\mathbf{M}$:

$$\mathbf{Q}_k, \mathbf{R}_k \leftarrow \text{QR-decomposition}(\mathbf{M}_k), \tag{4-13}$$

the upper triangular $\mathbf{R}$ matrix is multiplied from the right with the orthogonal matrix $\mathbf{Q}$ to obtain the next iteration of $\mathbf{M}$:

$$\mathbf{M}_{k+1} = \mathbf{R}_k \mathbf{Q}_k = \mathbf{Q}_k^T \mathbf{M}_k \mathbf{Q}_k. \tag{4-14}$$

As $k \to \infty$ the QR iteration will converge to:

$$\lim_{k \to \infty} \mathbf{Q}_k^T \cdots \mathbf{Q}_0^T \mathbf{M}_0 \mathbf{Q}_0 \cdots \mathbf{M}_k = \mathbf{E}^T \mathbf{\Lambda} \mathbf{E} \tag{4-15}$$

where $\mathbf{E} \in \mathbb{R}^{P \times P}$ is a matrix containing all of the eigenvectors of $\mathbf{M}$ and $\mathbf{\Lambda} = \text{diag}_2(\lambda_1, \ldots, \lambda_P)$ is a diagonal matrix with the corresponding eigenvalues on the diagonal.

Several procedures exist for computing a QR-decomposition of which the most widely used ones are through householder-reflection matrices or through Givens-rotations. Both will not be discussed in detail but the former can be found in appendix C-4 and the latter follows the definition of a Jacobi-rotation from definition 3.7. Givens-rotations and Jacobi-rotations are essentially the same. The difference comes from how they are used. When using Givens-rotations to diagonalize a matrix, as is done in the QR-algorithm from definition 4.6, they are called Jacobi-rotations. By themselves, Givens-rotations are designed to introduce zeros into an array which do not change anymore afterwards. Using them in Jacobi-rotation fashion decreases off-diagonal elements successively.

From a multi-linear perspective, the QR algorithm can be considered as a diagonalizaton method for tensors of order 2. The question then arises whether such an algorithm can exist for higher-order tensors to diagonalize them. As a matter of fact, we have established that such an algorithm already exists for symmetric tensors which can be considered as the higher-order equivalent of definition 4.6. This is the earlier presented COM2 algorithm from section 3-3-1. The computed rotation for the Jacobi-rotation follows the same root finding procedure with a quadri-linear based polynomial as it does for the bi-linear case [10][17]. However, as we mentioned before the COM2 algorithm scales poorly due to the cyclic way the rotation pairs are swept through.

QR-decomposition on a symmetric tensor slice Instead of using a direct higher-order analogue of the QR-algorithm as COM2, our proposed method uses bi-linear QR-decomposition on slices of a symmetric tensor and applies them iteratively on all modes using the mode-n product. The idea here is that this will impose some form of diagonality on the structure of the tensor.

Writing out the sequence can give more insight in the process. Let $\mathcal{X} \in \mathbb{R}^{[N,P]}$ be a symmetric tensor and let $\mathbf{X}_{(n)} = \left[\begin{array}{c|c} \mathbf{M}_1 & \mathbf{M}_2 \end{array}\right] \in \mathbb{R}^{P\times(N-1)P}$ be any of its its mode-n matricizations where $\mathbf{M}_1 \in \mathbb{R}^{P\times P}$ consists of the first P columns and $\mathbf{M}_2 \in \mathbb{R}^{P\times(N-2)P}$ of all the other $(N-2)P$ columns. First the QR-decomposition of $\mathbf{M}_1$ at iteration k is computed:

$$\mathbf{Q}_k, \mathbf{R}_k \leftarrow \text{QR-decomposition}(\mathbf{M}_{1,k}). \tag{4-16}$$

Afterwards the tensor is updated by multiplying all of its modes with the orthogonal transformation $\mathbf{Q}$:

$$\mathcal{X}_{k+1} = \mathcal{X}_k \times_1 \mathbf{Q}_k^T \times_2 \ldots \times_N \mathbf{Q}_k^T. \tag{4-17}$$

In matricized form this is equivalent to:

$$\mathbf{X}_{(n),k+1} = \mathbf{Q}_k^T \mathbf{X}_{(n),k} \underbrace{(\mathbf{Q}_k \otimes \ldots \otimes \mathbf{Q}_k)}_{N-2 \quad \text{Kronecker products}} \tag{4-18}$$

Expressing $\mathbf{X}_{(n),k+1}$ as a sequence of $\mathbf{X}_{(n),0}$ results in:

$$\mathbf{X}_{(n),k+1} = \mathbf{Q}_k^T \cdots \mathbf{Q}_0^T \mathbf{X}_{(n),0} \, (\mathbf{Q}_0 \cdots \mathbf{Q}_k \otimes \ldots \otimes \mathbf{Q}_0 \cdots \mathbf{Q}_k) \tag{4-19}$$

which can than be used to express $\mathbf{M}_{1,k}$ with $\mathbf{X}_{(n),0}$:

$$\begin{aligned} \mathbf{M}_{1,k+1} &= \mathbf{Q}_k^T \cdots \mathbf{Q}_0^T \mathbf{X}_{(n),0} \left(\underbrace{\mathbf{Q}_0 \cdots \mathbf{Q}_{k-1}\mathbf{q}_{1,k} \otimes \ldots \otimes \mathbf{Q}_0 \cdots \mathbf{Q}_{k-1}\mathbf{q}_{1,k}}_{N-3 \quad \text{Kronecker products}} \otimes \mathbf{Q}_0 \cdots \mathbf{Q}_k \right) \\ &= \mathbf{U}_{\mathbf{Q},k}^T \mathbf{X}_{(n),0} \mathbf{V}_{\mathbf{Q},k} \\ &\hookrightarrow \mathbf{X}_{(n),0} = \mathbf{U}_{\mathbf{Q},k} \mathbf{M}_{1,k+1} \mathbf{X}_{(n),0} \mathbf{V}_{\mathbf{Q},k}^T \end{aligned} \tag{4-20}$$

where $\mathbf{Q}_k = \left[\begin{array}{ccc} \mathbf{q}_{1,k} & \cdots & \mathbf{q}_{P,k} \end{array}\right]$. The matrix $\mathbf{U}_{\mathbf{Q},k} \in \mathbb{R}^{P\times P}$ is orthogonal and the matrix $\mathbf{V}_{\mathbf{Q},k} \in \mathbb{R}^{P^{N-1}\times P}$ is semi-orthogonal. If $\mathbf{M}_{1,K+1}$ is diagonal, equation (4-20) has the resemblance of an SVD of $\mathbf{X}_{(n),0}$ where the right singular vectors are constrained with a multi-linear structure as $\mathbf{V}_{\mathbf{Q},k}^T = \mathbf{u}_{\mathbf{Q},k,1} \otimes \ldots \otimes \mathbf{u}_{\mathbf{Q},k,1} \otimes \mathbf{U}_{\mathbf{Q},k}$ and where $\mathbf{u}_{\mathbf{Q},k,1}$ is the first column of $\mathbf{U}_{\mathbf{Q},k}$. Equation (4-18) shows that the matrix $\mathbf{V}_{\mathbf{Q},k} \in \mathbb{R}^{P^{N-1}\times P}$ consists only of the first P columns of $(\mathbf{Q}_k \otimes \ldots \otimes \mathbf{Q}_k)$. This means computing the SVD of $\mathbf{X}_{(n),0}$ will will not work as it is not equivalent to equation (4-20) due to the multi-linear constraint $(\mathbf{Q}_k \otimes \ldots \otimes \mathbf{Q}_k)$. This would result in only the next iteration of $\mathbf{M}_{1,k}$ and not the entire tensor. The larger second part $\mathbf{M}_{2,k}$ will be discarded if applying SVD. However, we believe that this comparison with SVD illustrates to some extent the workings of the QRT algorithm as our findings show that for $K >> 0$ the slice $\mathbf{M}_{1,K+1}$ does become diagonal.

Increasing slice diagonal entries We attempt to illustrate how applying $\mathbf{Q}$ in all modes can increase the diagonal entries of $\mathbf{M}_{1,k}$.

For convergence of the QR-algorithm [73][74][75], we observe the following. For any iteration of the QR-algorithm for a symmetric matrix the following inequality related to the first value on the diagonal holds true as it converges to $|\lambda_1|$:

$$|\mathbf{A}_{k+1}(1,1)| = |\mathbf{a}^T\mathbf{A}_{k+1}\mathbf{a}| = |\mathbf{q}_{k,1}^T\mathbf{A}_k\mathbf{q}_{k,1}| \geq |\mathbf{a}^T\mathbf{A}_k\mathbf{a}| = |\mathbf{A}_k(1,1)| \tag{4-21}$$

where $\mathbf{a} = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^P$. The absolute value must be taken here because $\mathbf{A}$ can be indefinite.

Now for a tensor this can be used as follows. Using previously defined tensor $\mathcal{X}$, its slice $\mathbf{M}_1$ is equivalent to the Tensor Times Same Vector (TTSV) operation $\mathbf{M}_1 = \mathcal{X}\mathbf{a}^{N-2}$. Computing the QR-decomposition of this slice and applying it on both sides results in the following equation for the top left entry:

$$\mathbf{q}_{1,k}^T\mathbf{M}_{1,k}\mathbf{q}_{1,k} = \mathbf{q}_{1,k}^T(\mathcal{X}_k\mathbf{a}^{N-2})\mathbf{q}_{1,k}. \tag{4-22}$$

This is the entry at the indices $(1,\dots,1)$ but note that this is not equivalent to equation (4-17). Due to the tensor's symmetry it does not matter in what modes the multiplication with $\mathbf{q}_{1,k}$ takes place. For example:

$$\mathbf{q}_{1,k}^T(\mathcal{X}_k\mathbf{a}^{N-2})\mathbf{q}_{1,k} = \mathbf{a}^T((\mathcal{X}_k\mathbf{q}_{1,k}^2)\mathbf{a}^{N-4})\mathbf{a}. \tag{4-23}$$

Writing out the absolute value similarly to (4-21) results in the following inequality:

$$|\mathbf{q}_{1,k}^T(\mathcal{X}_k\mathbf{a}^{N-2})\mathbf{q}_{1,k}| \geq |\mathbf{a}^T(\mathcal{X}_k\mathbf{a}^{N-2})\mathbf{a}| = |\mathcal{X}_k\mathbf{a}^N| = |\mathcal{X}_k(1,\dots,1)|. \tag{4-24}$$

This equality makes sense as it is nothing else but the QR-algorithm applied on the slice $\mathbf{M}_1$ of the higher-order tensor $\mathcal{X}$:

$$\mathcal{X}_{k+1} = \mathcal{X}_k \times_1 \mathbf{Q}^T \times_2 \mathbf{Q}^T \times_3 \mathbf{I} \times \dots \times_N \mathbf{I}. \tag{4-25}$$

Our question related to our QRT algorithm is would the increase of the first superdiagonal entry hold true too when multiplying 3 or more modes with $\mathbf{Q}^T$ instead of just 2? However, due to there not yet existing a higher-order equivalent of equation (4-21) we cannot make any further direct statements. Ideally, for the vector $\mathbf{q}_1$ it must be proven that:

$$\begin{gathered} |\mathcal{X}_{k+1}(1,\dots,1)| \geq |\mathcal{X}_k\mathbf{a}\mathbf{q}_{1,k}^N| \geq |\mathcal{X}_k\mathbf{a}^N| = |\mathcal{X}_k(1,\dots,1)|, \\ \text{and} \\ \lim_{k\to\infty}\mathbf{Q}_k = \mathbf{I}, \end{gathered} \tag{4-26}$$

which means that the first superdiagonal entry keeps on increasing until $\mathbf{Q}$ has converged. Up to this day, our results have not presented any counterexample of this hypothesis.

We believe that such a higher-order analogue of the QR-algorithm convergence on a tensor slice exists. More precisely, we hypothesize that multiplying all modes of a symmetric tensor $\mathcal{X}$ with $\mathbf{Q}^T$ of the QR-decomposition of its frontal slice $\mathcal{X}(:,:,1,\ldots,1)$, leads to its diagonalization of that slice. This hypothesis is presented below.

Hypothesis 4.1: QR-algorithm on tensor slice

Let $\mathcal{X} \in \mathbb{R}^{[N,P]}$ be a **symmetric** tensor of order N and let $\mathbf{X}_{(n)} = \left[\begin{array}{c|c} \mathbf{M}_1 & \mathbf{M}_2 \end{array}\right]$ be any of its its mode-n matricizations where $\mathbf{M}_1 \in \mathbb{R}^{P\times P}$ consists of the first P columns and $\mathbf{M}_2 \in \mathbb{R}^{P\times(N-2)P}$ of all the other $(N-2)P$ columns.
By computing the QR-decomposition of $\mathbf{X}_1$ at iteration k:

$$\mathbf{Q}_k, \mathbf{R}_k \leftarrow \text{QR-decomposition}(\mathbf{X}_{1,k}) \tag{4-27}$$

and applying the orthogonal transformation $\mathbf{Q}_k$ in all tensor modes to obtain a new iteration of $\mathcal{X}$:

$$\mathcal{X}_{k+1} = \mathcal{X}_k \times_1 \mathbf{Q}_k^T \times_2 \ldots \times_N \mathbf{Q}_k^T, \tag{4-28}$$

we hypothesize that:

$$\begin{gathered} \lim_{k\to\infty} \mathcal{X}_0 \times_1 \mathbf{Q}_0^T \cdots \mathbf{Q}_k^T \times_2 \ldots \times_N \mathbf{Q}_0^T \cdots \mathbf{Q}_k^T \to \mathbf{M}_{1,k} = \mathbf{\Lambda}' \\ \text{and} \\ \lim_{k\to\infty} \mathbf{Q}_k = \mathbf{I} \end{gathered} \tag{4-29}$$

where $\mathbf{\Lambda}'$ represents a diagonal matrix.

The hypothesis above implies that due to tensor symmetry, all slices equivalent to $\mathcal{X}(:,:,1,\ldots,1)$ will be simultaneously diagonalized. These are slices which arise from any permutation of the index set $S_N = \{s_1, s_2, 3, \ldots, N\}$ where we vary $s_1 = s_2$ over $[1,\ldots,N] \in \mathbb{Z}$. These are all of the slices that together form part of the outer-layer of the tensor, or *outer-shell*. The slices of the outer-shell that are diagonalized are considered as the 'top' part of the *outer-shell*. This is visually represented for a 3-way tensor in Figure 4-9.

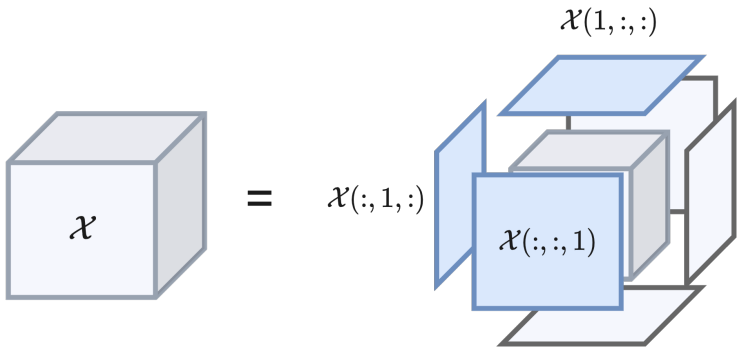


Figure 4-9: Exploded view of the outer shell of a 3-way tensor $\mathcal{X}$. The blue slices are together considered as the 'top' part of the outer-shell.

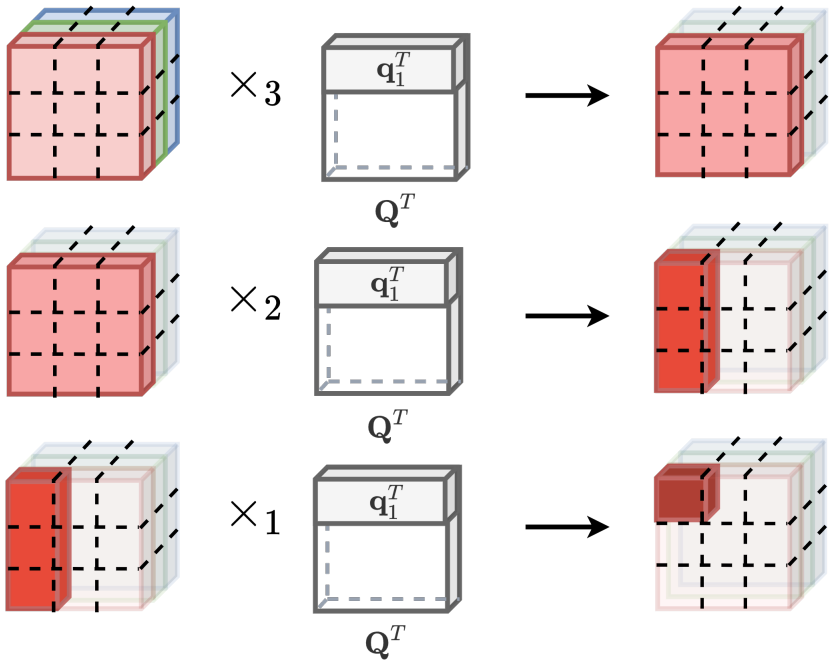


Figure 4-10: Visual representation of how the norm of elements is increased due to the successive mode-n multiplications with $\mathbf{q}_1$. A darker red color indicates that the norm of that tensor slice, fiber or entry has increased in comparison to before the previous mode-n multiplication.

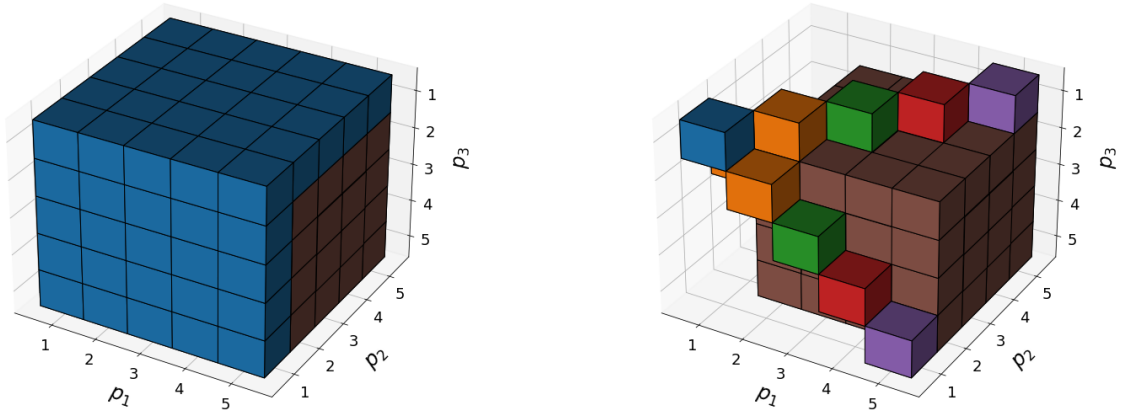
What is happening during the QRT procedure can be intuitively explained using Figure 4-10. In the figure above a 3-way tensor of size $3 \times 3 \times 3$ is shown of which the frontal slices are colored in red, green and blue consecutively. Each row of the figure represents the multiplication of this tensor's mode n with the $\mathbf{Q}^T = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 \end{bmatrix}$ matrix of the red slice's QR-decomposition. The figure focuses on how the norm of parts of the tensor is partially condensed into the top left corner at indices $(1, 1, 1)$ due to $\mathbf{q}_1^T$. The darkening of the red color is to illustrate how the norm of that particular sub part increases in comparison to before the last mode-n product. In this small example we assume for simplicity that $\mathbf{q}_2^T$ and $\mathbf{q}_3^T$ are null vectors. The faded parts of the tensor are those that are not affected by $\mathbf{q}_1$.

Top outer-shell diagonalization: numerical example We present a small numerical example on the workings of the QRT algorithm on a symmetric 3-way tensor $\mathcal{X}^{[3,3]}$. The tensor is created by multiplying all modes of a diagonal core tensor $\mathcal{C}$ with 1's on its superdiagonal with 5 stacked vectors randomly generated from a uniform distribution:

$$\mathcal{X} = \mathcal{C} \times_1 \mathbf{U} \times_2 \mathbf{U} \times_3 \mathbf{U}. \tag{4-30}$$

The result of performing the QRT procedure from algorithm 2 for $K = 25$ iterations is shown below in figure Figure 4-11. Both figures show the 3-way tensor as a voxelplot where visible voxels at the indices (p_1, p_2, p_3) represents a value of $\mathcal{X}(p_1, p_2, p_3) > 0$ and non-visible voxels are entries in the tensor of value 0. The left Figure 4-11a shows the tensor before applying the

QRT procedure and the right Figure 4-11b after. The values of the unique diagonal entries of the top outer-shell $\mathbf{d} = \{d_1, \ldots, d_5\}$ are plotted in Figure 4-12.



(a) Tensor $\mathcal{X} \in \mathbb{R}^{[3,5]}$. The top outer-shell is indicated in blue.

(b) Tensor $\mathcal{X} \in \mathbb{R}^{[3,5]}$ after performing the QRT iteration for $K = 25$ iterations.

Figure 4-11: Tensor $\mathcal{X} \in \mathbb{R}^{[3,5]}$ with indices (p_1, p_2, p_3) before and after the QRT procedure. All non-visible voxels are entries in the tensor of value 0. The top outer-shell originally indicated in blue has been changed into a collection of diagonal slices of which the unique entries $\mathbf{d} = \{d_1, \ldots, d_5\}$ are denoted in colors which match Figure 4-12.

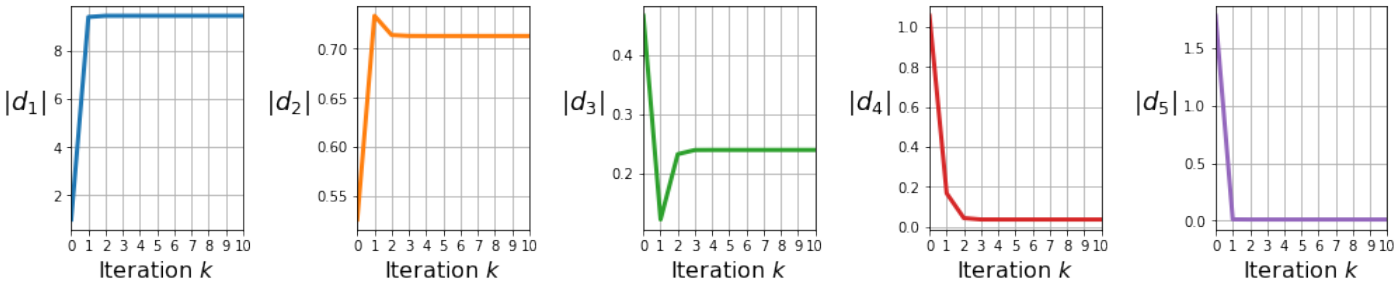
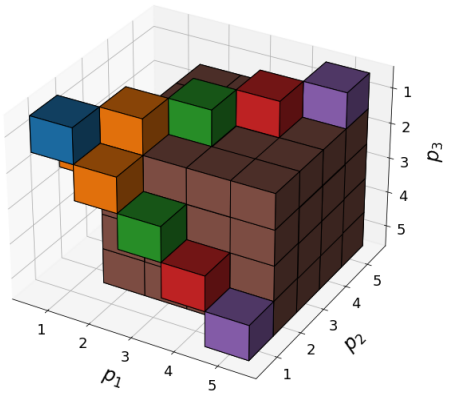


Figure 4-12: Absolute values of $\mathbf{d}$: $|d_i| \quad \forall i = 1, 2, 3, 4, 5$ during the QRT procedure on the top outer-shell. Only the first 10 iterations are shown.

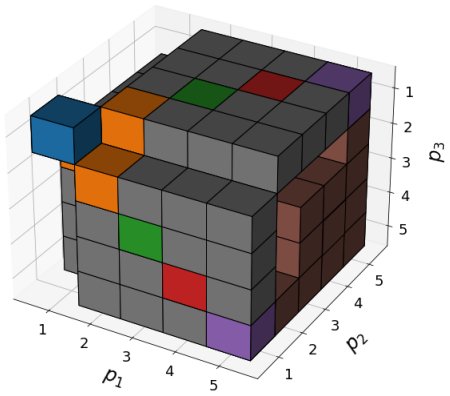
The tensor value $|d_1| = |\mathcal{X}(1, 1, 1)|$ shown in blue keeps increasing up till a certain value. Conversely the last 2 values $|d_5| = |\mathcal{X}(5, 5, 5)|$ and $|d_4| = |\mathcal{X}(4, 4, 1)|$ decrease in a similar fashion. The values $|d_2| = |\mathcal{X}(2, 2, 1)|$ and $|d_3| = |\mathcal{X}(3, 3, 1)|$ show a more interesting behaviour. Value $|d_2|$ reaches a certain maximum at $k = 1$, decreases afterwards and finally settles. For $|d_3|$ the same happens but then in a decreasing way. We conclude that this odd behaviour of $|d_2|$ and $|d_3|$ is necessary for the increase or decrease of $|d_1|$, $|d_4|$ and $|d_5|$ as they have not reached their maximum or minimum yet at $k = 1$. The final value of $|d_1|$ is between $\approx 10\times$ and $\approx 1000\times$ higher than that $|d_2|$, $|d_3|$, $|d_4|$ and $|d_5|$. The ratio between the value of $|d_1|$ and $\sqrt{3|d_2|^2 + 3|d_3|^2 + 3|d_4|^2 + 3|d_5|^2}$ equals $\tau_D = 0.991$. This can be considered as the measure of diagonality τ_D from definition 2.6 of only the top outer-shell. The values $d_i \quad \forall i = 2, 3, 4, 5$ are multiplied with 3 as they have a multiplicity of 3 in the top outer-shell. We observe the

same behavior of the diagonal entries of a symmetric matrix during the QR-algorithm. An example is shown in appendix C-5.

As the top outer-shell has been diagonalized the procedure is repeated on the inner part of the tensor indicated in brown in Figure 4-11b by computing the QR-decomposition of the truncated slice $\mathcal{X}(2:,2:,2)$. This procedure is said to work on the first top *inner-shell* of the tensor $\mathcal{X}$. The converged resulting tensor is displayed in Figure 4-13b and the values of $\mathbf{d}$ are shown below that in Figure 4-14. The figures show that the top outer-shell has lost its diagonal form and that the values $|d_i| \quad \forall i = 2,3,4$ have changed. Again a settling behaviour can be seen in the changed values albeit this time with a lot more fluctuation beforehand. However, the previous measure of diagonality has remained the same $\tau_D = 0.991$.



(a) Tensor $\mathcal{X} \in \mathbb{R}^{[3,5]}$ after performing the QRT iteration for $K = 25$ only once on the top outer-shell.



(b) Tensor $\mathcal{X} \in \mathbb{R}^{[3,5]}$ after performing the QRT iteration for $K = 25$ on the top outer-shell and the first inner shell.

Figure 4-13: Tensor $\mathcal{X} \in \mathbb{R}^{[3,5]}$ with indices (p_1, p_2, p_3) before and after the QRT procedure on the first top inner-shell. All non-visible voxels are entries in the tensor of value 0. The previously diagonalized top outer-shell has now lost its diagonality. This is indicated by the tensor entries in grey.

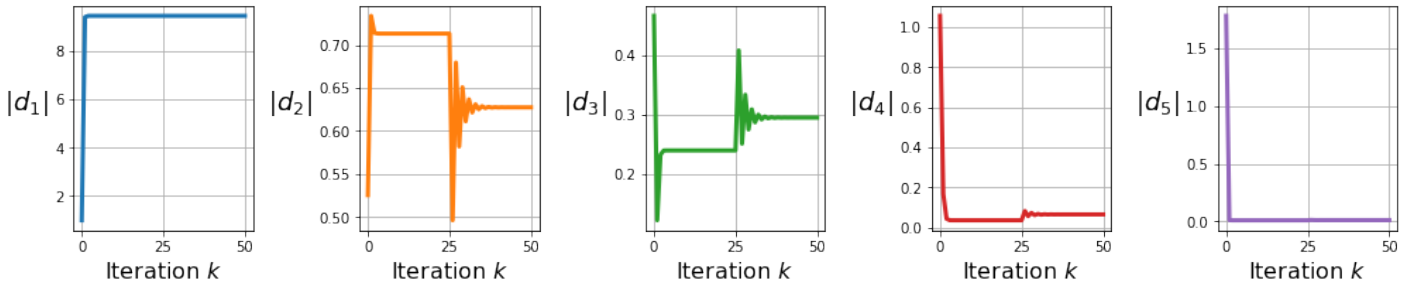


Figure 4-14: Absolute values of $\mathbf{d}$: $|d_i| \quad \forall i = 1,2,3,4,5$ during the QRT procedure on the top outer-shell and first top inner-shell.

The fibers $\mathcal{X}(:,1,1)$, $\mathcal{X}(1,:,1)$ and $\mathcal{X}(1,1,:)$ have remained unchanged too. The reason why is visually shown in Figure 4-15. Due to the new but smaller $\mathbf{Q}'$ matrix being extended with an identity matrix, the tensor entry $\mathcal{X}(1,1,1)$ remains untouched. On top of that, paired with

the zeros already present on the fibers $\mathcal{X}(:,1,1)$, $\mathcal{X}(1,:,1)$ and $\mathcal{X}(1,1,:)$, the fibers effectively do not change either. The top row of $\mathbf{Q}^T$ does not change anymore after the QRT procedure on the top outer-shell. This has an interesting implication for ICA. Namely that the 4'th order analogue of Figure 4-13a and Figure 4-13b result in the same solution for the first estimated source component.

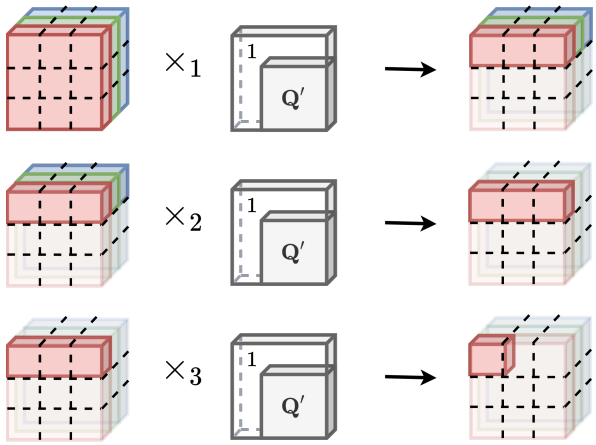


Figure 4-15: Unaffected part of the tensor $\mathcal{X}$ when multiplied in all modes with the $\mathbf{Q}'$ matrix computed from the QR-decomposition of $\mathcal{X}(2:,2:,2)$. The after each multiplication the effected part is faded-out and the unaffected part is kept in full color. The only truly unaffected element is the top of the superdiagonal $\mathcal{X}(1,1,1)$.

Lastly, 'peeling' of the top outer-shell reveals that the previously diagonal form is now present on the first top inner-shell. This means that this diagonal structure is present on the top shell layer that the QRT-procedure is working on. Applying the QRT-procedure iteratively through the top outer-shell and all of the top inner-shells will result in a tensor of which all fibers $\mathcal{X}(i:,i,i)$, $\mathcal{X}(i,i:,i)$ and $\mathcal{X}(i,i,i:)$ $\forall i = 1,2,3,4$ will be null-vectors, or null-fibers, with a single nonzero element at the top which corresponds to the elements on the tensors superdiagonal $\mathcal{X}(i,i,i)$ $\forall i = 1,2,3,4,5$.

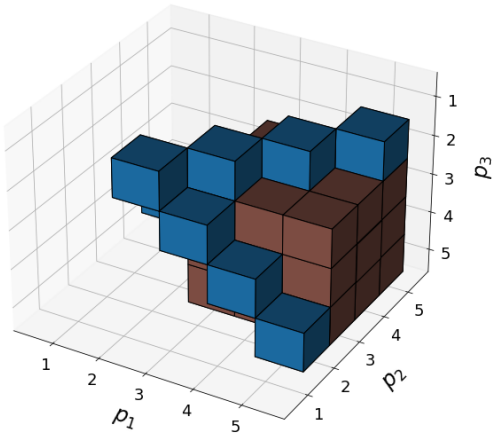


Figure 4-16: Tensor $\mathcal{X} \in \mathbb{R}^{[3,5]}$ after performing the QRT iteration for $K = 25$ on the top outer-shell and the first inner shell of which the top outer-shell is 'peeled' off.

4-3-2 Implicit QRT for the cumulant tensor

In this section we present how our partial cumulant tensor entry, fiber and slice computation method from section 4-2-1 can be combined with the QRT algorithm for implicit use. In the previous section we showed that for ICA the solution of the i'th component did not depend on the full diagonalization of the i'th top inner-shell. Only the part affected by the multiplication in all modes with the column vector $\mathbf{q}_i$ of the orthogonal transformation $\mathbf{Q}$ was shown to be relevant.

This means that for the cumulant tensor only the (truncated) fibers $\mathcal{C}_{\mathbf{z}}^{(4)}(i\ :, i, i, i) \quad \forall i = 1, 2, \ldots, P$ of the core slices shown in Figure 4-17 are needed. If all core slices from Figure 4-8 are stacked next to each other, then the relevant (truncated) vectors are shown in Figure 4-18 below. In other words, based on the workings of any QR-decomposition procedure only the QR-decomposition of these fibers is needed.

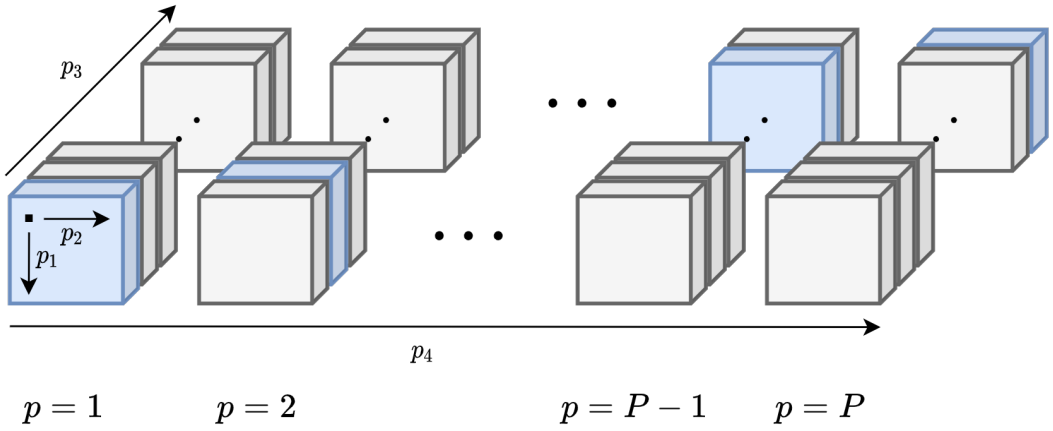


Figure 4-17: The core slices of the cumulant tensor.

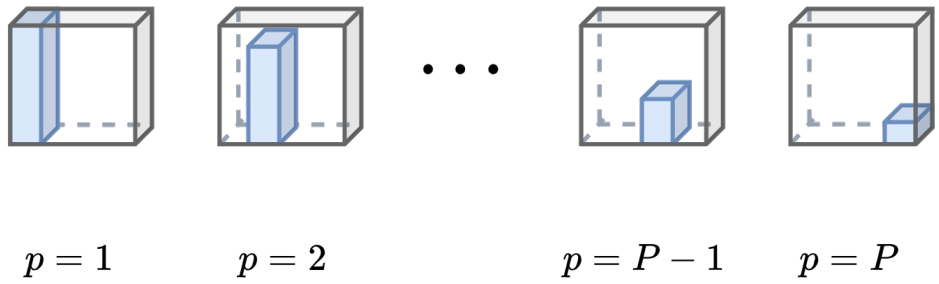


Figure 4-18: The part of the columns of the core slices indicated in blue in Figure 4-17 relevant for the implicit computation of the QRT procedure for the cumulant tensor.

Cumulant tensor slice update The second-order sparse tensor $\mathcal{T}_{\mathbf{I}}$ from definition 4.3 has a property which is exploited by the implicit QRT method. Using lemma 4.2, it can be stated that the outer-product of identity matrices remains unchanged when multiplied with the same orthogonal transformation $\mathbf{Q}$ in all modes:

$$(\mathbf{I}_P \circ \mathbf{I}_P) \times_1 \mathbf{Q} \times_2 \mathbf{Q} \times_3 \mathbf{Q} \times_4 \mathbf{Q} = \left(\mathbf{Q}\mathbf{I}_P\mathbf{Q}^T\right) \circ \left(\mathbf{Q}\mathbf{I}_P\mathbf{Q}^T\right) = (\mathbf{I}_P \circ \mathbf{I}_P)\,. \tag{4-31}$$

Lemma 4.2: Identity outer-product transformation

The mode-n product of the **outer-product** of 2 identity matrices with the matrix $\mathbf{M}$ is equivalent to multiplying the n'th dimension in the outer-product sequence with $\mathbf{M}$:

$$\begin{aligned}
(\mathbf{I}_P \circ \mathbf{I}_P) \times_1 \mathbf{M} &= (\mathbf{M}\mathbf{I}_P) \circ (\mathbf{I}_P)\,, \quad (\mathbf{I}_P \circ \mathbf{I}_P) \times_2 \mathbf{M} \quad = \left(\mathbf{I}_P\mathbf{M}^T\right) \circ (\mathbf{I}_P) \\
(\mathbf{I}_P \circ \mathbf{I}_P) \times_3 \mathbf{M} &= (\mathbf{I}_P) \circ (\mathbf{M}\mathbf{I}_P)\,, \quad (\mathbf{I}_P \circ \mathbf{I}_P) \times_4 \mathbf{M} \quad = (\mathbf{I}_P) \circ \left(\mathbf{I}_P\mathbf{M}^T\right).
\end{aligned} \tag{4-32}$$

The proof of this lemma can be found in C-6.

This proposition remains true for index permutation. So by extension the second-order part $\mathcal{T}_{\mathbf{I}}$ of the cumulant tensor from definition (4.3) is invariant under any orthogonal transformation of the original whitened data $\mathbf{Z}$. This is shown in definition 4.7.

Definition 4.7: Invariant whiteness

The **second-order** part $\mathcal{T}_{\mathbf{I}} \in \mathbb{R}^{P\times P\times P\times P}$ of the cumulant tensor from (4.3) for whitened data is invariant under any orthogonal transformation $\mathbf{Q}\mathbb{R}^{P\times P}$ of the original whitened data $\mathbf{Z} \in \mathbb{R}^{P\times I}$:

$$\begin{aligned}
\mathbf{Q}\mathbf{Z} \quad &\rightarrow \quad \mathcal{T}_{\mathbf{I}} \times_1 \mathbf{Q} \times_2 \mathbf{Q} \times_3 \mathbf{Q} \times_4 \mathbf{Q} = \mathcal{T}_{\mathbf{I}} \\
\text{with} \quad \mathcal{T}_{\mathbf{I}} &= (\mathbf{I}_P) \circ (\mathbf{I}_P) + ((\mathbf{I}_P) \circ (\mathbf{I}_P))^{T\sigma_1} + ((\mathbf{I}_P) \circ (\mathbf{I}_P))^{T\sigma_2} \\
\sigma_1 &= [1,3,2,4], \quad \sigma_2 = [1,4,3,2]
\end{aligned} \tag{4-33}$$

From a statistical standpoint definition 4.7 makes sense too as whitened data remains whitened under any orthogonal transformation. For the implicit version of the algorithm this means that the mode-n product in all modes with $\mathbf{Q}^T$ can be replaced by $\mathbf{Q}^T\mathbf{Z}$ and that afterwards only the fourth-order moment part $\mathcal{M}^{(4)}$ has to be recomputed which results in a minimal amount of necessary computations. Definition 4.4 shows that the second order part $\mathcal{T}_{\mathbf{I}}$ consists during every iteration of only an identity matrix multiplied by 3.

The implicit QRT algorithm for the cumulant tensor. We combine all of these findings to form the Implicit QR-Tensor algorithm (QRT) for the cumulant tensor shown in algorithm 3. For completeness the computation of the covariance step $\mathbf{C}_{\mathbf{z}}^{(2)}$ is kept. However, due to whitening this step can be substituted by $\mathbf{C}_{\mathbf{z}}^{(2)} = \mathbf{I}_P$.

The algorithm contains a convergence error computation such that a certain tolerance can be specified. The computation of this error is shown later on in the chapter containing numerical comparisons. The complexities of our algorithm are presented below in property 4.3. The implicit algorithm circumvents the one-time cost of forming the entire tensor $O(IP^4)$ and the per iteration intensive mode-n multiplication cost of $O(P^5)$.

Algorithm 3 Implicit QRT for the cumulant tensor

Require: Whitened data: $\mathbf{Z} \in \mathbb{R}^{P \times I}$, max iterations: K_{max}, number of components to estimate: R, tolerance: tol

procedure QR-T($\mathbf{Z}, K_{max}, R$)
  $\mathbf{Q}_{store} = \mathbf{I}_P$
  while $\varepsilon > tol$ or $i < K_{max}$ **do**
    $\mathbf{C}_{\mathbf{z}}^{(2)} = \frac{1}{I}\mathbf{Z}\mathbf{Z}^T$
    $\mathbf{M} = \frac{1}{I}\mathbf{Z}\left[\mathbf{Z}(1:R,:)^T\right]^3 - 3 \cdot \mathbf{C}_{\mathbf{z}}^{(2)}(1:R,:)$
    $\mathbf{Q} = \text{QR-decomposition}\,(\mathbf{M})$
    $\varepsilon = \text{Convergence Error function}(\mathbf{Q})$
    $\mathbf{Q}_{store} \leftarrow \mathbf{Q}^T\mathbf{Q}_{store}$
    $\mathbf{Z} \leftarrow \mathbf{Q}^T\mathbf{Z}$
  end while
  $\hat{\mathbf{S}} = \mathbf{Z}(0:R,:)$
  return $\hat{\mathbf{S}}, \mathbf{Q}_{store}$
end procedure

Property 4.3: Complexities of implicit QRT for the cumulant tensor

Computational complexity The computational complexity of algorithm 3 is dominated by the computation of the tensor fibers $\mathbf{M}$. For a cumulant tensor of size $\mathcal{C}_{\mathbf{Z}}^{(4)} \in \mathbb{R}^{[4,P]}$ this cost is $O(IRP)$ per iteration.

Storage complexity For a cumulant tensor of size $\mathcal{C}_{\mathbf{Z}}^{(4)} \in \mathbb{R}^{[4,P]}$ the storage complexity of algorithm 3 is $O(P^2)$.

4-4 Canonical Polyadic Decomposition

As was mentioned before, a special case of the Tucker decomposition exists where the resulting core is constrained to be diagonal from definition 2.6 called Canonical Polyadic Decomposition (CPD). Before CPD and its functionality for ICA are presented, the following tensor rank concepts related to CPD are explained.

4-4-1 Tensor ranks

Rank-one tensor The concept of a *rank-one* tensor is easily understood with the definition of outer-products. Its definition is given below in 4.8 and a visual representation of a third-order rank-one tensor can be found in Figure 4-19.

Definition 4.8: Rank-one Tensor [28]

A tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$ of order N which can be written as the **outer-product** of N vectors is of **rank one**:

$$\mathcal{X} = \mathbf{x}_1 \circ \mathbf{x}_2 \circ \cdots \circ \mathbf{x}_N. \tag{4-34}$$

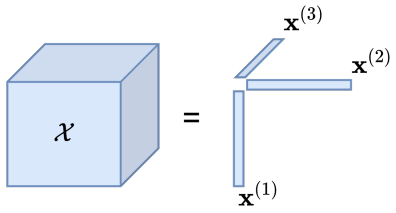


Figure 4-19: A 3-way rank-one tensor $\mathcal{X}$ which is expressed as a sequence of vector outer products of the vectors $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$ and $\mathbf{x}^{(3)}$.

Symmetric outer-product A sequence of outer-products of a vector with only itself is defined as a *symmetric outer-product*. The symmetric outer-product, or d-way outer product, of a vector is defined below in definition 4.9 and results in a symmetric rank-one tensor.

Definition 4.9: Symmetric outer-product [53]

The N**-way outer-product** of a P-dimensional vector $\mathbf{x} \in \mathbb{R}^P$ with **itself** is denoted as:

$$\mathbf{x}^{\circ N} = \underbrace{\mathbf{x} \circ \mathbf{x} \circ \cdots \circ \mathbf{x}}_{N \text{ times}} \quad \text{with} \quad \left(\mathbf{x}^{\circ N}\right)_{i_1 \ldots i_N} = \prod_{k=1}^{N} \mathbf{x}_{i_k} \text{ for } i_1, \ldots, i_N \in \{1, \ldots, P\} \tag{4-35}$$

which results in an N-dimensional **symmetric rank-one** tensor $\mathcal{X} \in \mathbb{R}^{[N,P]}$ formed at the cost of $O(I^N)$.

Canonical-Polyadic-rank The Canonical-Polyadic-rank (CP-rank) of a tensor extends the notion of the bi-linear rank of a matrix into multi-linear algebra. It plays an important role in many tensor decomposition related problems [28][5][30][15] and forms under certain assumptions the foundation of theorems [36] and procedures [30][53] presented later on. Although seemingly similar, there are some crucial differences between tensor rank and matrix rank. First of all, finding the CP-rank of a tensor is not as straightforward as for matrices as it is classified as NP-Hard [76]. Secondly, tensor ranks follow a different set of rules as for example they can differ over $\mathbb{R}$ and $\mathbb{C}$. Moreover, the CP-rank can in fact be greater than the size of the largest mode.

The CP-rank R of a tensor can be expressed as the minimal amount of rank-one tensors from definition 4.8 needed to represent the tensor. The definition of the CP-rank is given below and a graphical representation can be seen in Figure 4-20. Lower and upper bounds for the CP-rank tensor can be derived depending on the tensors size and order [28]. However, in practice the CP-rank is estimated numerically by fitting several CP models with varying ranks R [28].

Definition 4.10: Tensor Canonical-Polyadic rank R [28]

The **CP-rank** R of a tensor $\mathcal{X}$ of order N is the **minimum** number of **rank-one** tensors that when summed up form $\mathcal{X}$ [77]:

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{x}_{1,r} \circ \mathbf{x}_{2,r} \circ \ldots \circ \mathbf{x}_{N,r}, \tag{4-36}$$

where $\mathbf{x}_{n,r}$ denotes the n'th vector in the r'th outer-product sequence to form the r'th rank-one tensor from definition 4.8. Computing the CP-rank R of a tensor has been proven to be an NP-Hard problem [78][76].

Canonical Polyadic decomposition Introduced by Hitchcock [77], Canonical Polyadic decomposition factorizes a tensor into an estimate of the minimal sum of rank-one tensors from definition 4.10. Although reintroduced as CANDECOMP (canonical decomposition) by Carrol and Chang [79] and later as PARAFAC (parralel factors) by Harshman [80], CPD has gained a lot of attention over the years in many areas such as signal processing [15][81], deep neural networks [5] and data analysis [7] due to its unique structure and properties.

The general definition of a weighted CPD is given below together with an illustrated example of the CPD of a 3-way tensor in Figure 4-20.

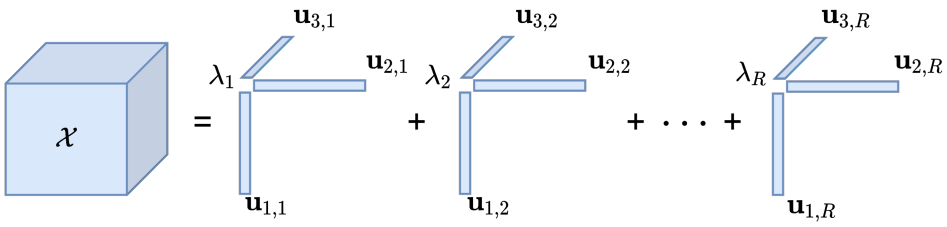


Figure 4-20: A 3-way tensor $\mathcal{X}$ expressed as a sum of R 3-way rank-one tensors where R is the CP-rank of the tensor.

Definition 4.11: Canonical Polyadic Decomposition

The **Canonical Polyadic decomposition** of a tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$ consists of R weighted **rank-one** tensors which when summed up form $\mathcal{X}$:

$$\mathcal{X} \approx [\![\text{diag}_N(\boldsymbol{\lambda}); \mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N]\!] \equiv \sum_{r=1}^{R} \lambda_r \mathbf{u}_{1,r} \circ \mathbf{u}_{2,r} \circ \ldots \circ \mathbf{u}_{N,r}, \tag{4-37}$$

where R is the CP-rank of the tensor, $\boldsymbol{\lambda} \in \mathbb{R}^R$ is a vector consisting of the concatenated weights and each factor matrix $\mathbf{U}_n = \begin{bmatrix} \mathbf{u}_{n,1} & \ldots & \mathbf{u}_{n,R} \end{bmatrix} \mathbb{R}^{P_n \times R} \quad \forall n = 1, \ldots, N.$

Definition 4.11 shows that CPD is in fact a special case of the Tucker decomposition in which the core tensor is a diagonal tensor from definition 2.6 with the weights λ on its superdiagonal. This means that CPD is in fact a diagonalization transformation of a tensor. However, in order to make use of the multi-linearity property 3.5 of the cumulant tensor such that observed data can be unmixed, an additional symmetry constraint has to be imposed on the factor matrices themselves. This introduces the symmetric counterparts of CP-rank and CPD.

Symmetric CP-rank For symmetric tensors there exists a second rank in which the rank-one tensors are constrained to be symmetric too. This is called the symmetric CP-rank [55][82]. Its definition is presented below in definition 4.12.

Definition 4.12: Symmetric Canonical-Polyadic rank S_R [83]

The **symmetric CP-rank** S_R of a symmetric tensor $\mathcal{X}$ of order N is the **minimum** number of symmetric **rank-one** tensors that when summed up form $\mathcal{X}$:

$$\mathcal{X} = \sum_{r=1}^{S_R} \mathbf{x}_r^{\circ N}. \tag{4-38}$$

Currently it remains unknown for symmetric tensors if the symmetric rank is always identical to the CP-rank while both have been proven to always exist [55]. Two of the most challenging problems concerning these tensor ranks are Strassen's Conjecture on the additivity of tensor ranks and Comon's Conjecture on the equality of the CP-rank and the symmetric CP-rank. The latter has been proven to hold true for a number of cases [55][82][84], e.g. when the CP-rank is lower than the mode size. At first hand this seems useful for Blind Source Separation (BSS) when the problem is considered to be over-determined. However it is covered later on why these tensor ranks of the cumulant tensor can be misleading and must be used with caution when consdering ICA.

Symmetric Canonical Polyadic Decomposition Estimating a decomposition of a symmetric tensor into the format of its symmetric CP-rank is called symmetric CPD. Its mathematical definition is given below in definition 4.13. The authors of [85][53] refer to it alternatively as the Kruskal format due to his work on tensors of this format [86][87]. Identical as with definition 4.11, the factors are normalized where each scalar λ_r results from the normalization of its corresponding factor symmetric N-way outer-product $\mathbf{u}_r^{\circ N}$.

Definition 4.13: Symmetric Canonical Polyadic Decomposition

The **symmetric Canonical Polyadic decomposition** of a symmetric tensor $\mathcal{X} \in \mathbb{R}^{[N,P]}$ consists of S_R weighted and symmetric **rank-one** tensors which when summed up form $\mathcal{X}$:

$$\mathcal{X} \approx [\![\text{diag}_N(\boldsymbol{\lambda}); \mathbf{U}, \mathbf{U}, \ldots, \mathbf{U}]\!] \equiv \sum_{r=1}^{S_R} \lambda_r \mathbf{u}_r^{\circ N}, \tag{4-39}$$

where S_R is the symmetric CP-rank of a symmetric tensor, $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \ldots & \mathbf{u}_R \end{bmatrix} \in \mathbb{R}^{P \times S_R}$ is the single factor matrix and $\boldsymbol{\lambda} \in \mathbb{R}^{S_R}$ are the weighting coefficients of the factors.

From this point on, whenever CPD, Parallel Factor Analysis (PARAFAC) or any other equivalent decomposition [79] of the cumulant tensor is mentioned, it is assumed that the symmetric CPD format from definition 4.13 is used. It is explicitly mentioned if else.

Computing the symmetric CPD of the fourth-order cumulant tensor is equivalent to the diagonalization of its eigenmatrices [36] which is performed by the Joint Approximate Diagonalization of Eigenmatrices (JADE) algorithm [23]. Cardoso [51][88] explains that identically to how a symmetric operator $\mathbf{A} \in \mathbb{R}^{P \times P}$ on a P-dimensional space can be decomposed into P real eigenvalues and P orthonormal eigenvectors, a symmetric operator on a P^2-dimensional space can be decomposed into P^2 real eigenvalues and P^2 orthonormal operators which are called "Eigenmatrices" [51] [21]. Later works [81][36][89] show that CPD of the cumulant tensor is equivalent to the joint diagonalization of its eigenmatrices which means it too is identical to a diagonalization transformation of the tensor itself. In a later section we show how a first-order optimization problem for computing the symmetric CPD of the cumulant tensor bears a lot of similarity to the earlier presented FastICA.

CPD uniqueness conditions An interesting property of the CPD of a tensor is that it is unique under the right conditions. First proven by Kruskal [86][87] for the 3-way tensor case and later on generalized by Sidiropoulos and Bro [90] for the N-dimensional case, a tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$ is stated to have a unique CPD if the following condition is met:

$$\sum_{n=1}^{N} k_{\mathbf{U}_n} \geq 2R + (N-1), \tag{4-40}$$

where $k_{\mathbf{U}_n}$ denotes the k-rank of the factor matrix $\mathbf{U}_n$ and R the tensor rank. The k-rank of a matrix $\mathbf{M}$ is defined as the maximum value k such that any k columns are linearly independent [86]. In [28] it is shown that the CPD of a tensor is unique up to a scaling and

a permutation indeterminacy. The exact same indeterminacies of ICA. Important to note is that a CPD with unitary factors is unique under even milder conditions [91].

4-4-2 Tensor ranks for ICA and cumulant tensor diagonality

Estimating either the CP-rank or symmetric CP-rank of the cumulant tensor can provide an indication of the amount of underlying independent components. This can be considered as the higher-order equivalent of how the covariance matrix rank equals the amount of principal components. However, in practice the computation of an exact Canonical Polyadic model of a cumulant tensor is hindered by the approximation of cumulants through k-statistics [60] and sampling errors present in the data [10]. On top of that, although the statistical independence assumption in practice delivers good results as for example in functional Magnetic Resonance Imaging (fMRI) studies [92][39][93][94][40][61], from a practical perspective it is somewhat unrealistic to assume a perfectly diagonal cumulant tensor and why approximately diagonal can be considered as good enough.

In this section we first provide insight in the behaviour of the cumulant tensor's ranks related to its approximate diagonalization for ICA. By doing so we show how ICA can benefit from applying HOSVD as a preprocessign step. Afterwards we briefly discuss the constraint of orthogonality on CPD factor matrices and what its implications are for the rest of our work.

Non-mixed case Consider the following noiseless example consisting of 2 signals $\mathbf{s}(t) = \begin{bmatrix} s_1(t) \\ s_2(t) \end{bmatrix}$: a sinusoidal signal $s_1(t) = \sin(2t)$ and a sawtooth signal $s_2(t) = 1.2 \cdot \left(t - \frac{1}{2} - \lfloor t \rfloor\right)$ for a timespan of $t = [0, 20]$ seconds which are shown in Figure 4-21. The reader is pointed out that these signals are not mixed in this example.

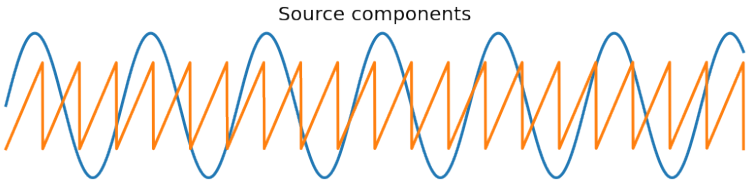


Figure 4-21: Two source components of an artificial data set consisting of a sinusoidal signal s_1 and a sawtooth signal s_2.

Computing the fourth-order cumulant tensor $\mathcal{C}_{\mathbf{s}}^{(4)}$ of these non-mixed signals for $I = 5e4$ samples and stacking its slices as follows yields:

$$\left[\begin{array}{c|c} \mathcal{C}_{\mathbf{s}}^{(4)}(:,:,1,1) & \mathcal{C}_{\mathbf{s}}^{(4)}(:,:,1,2) \\ \hline \mathcal{C}_{\mathbf{s}}^{(4)}(:,:,2,1) & \mathcal{C}_{\mathbf{s}}^{(4)}(:,:,2,2) \end{array}\right] = \left[\begin{array}{cc|cc} -3.9e-1 & -8.5e-3 & -8.5e-3 & -6.7e-4 \\ -8.5e-3 & -6.7e-4 & -6.7e-4 & -5.9e-4 \\ \hline -8.5e-3 & -6.7e-4 & -6.7e-4 & -5.9e-4 \\ -6.7e-4 & -5.9e-4 & -5.9e-4 & -1.7e-2 \end{array}\right]. \tag{4-41}$$

Inspection of the values shows that these signals are not completely statistically independent. Note that these values represent excess kurtosis from definition 3.4 which means that sign is trivial. The absolute of the entries on the cumulant tensor's superdiagonal (blue) are relatively higher than their neighbouring entries but the off-diagonal entries (black) are not zero. This

goes to show how latent source components can show some measure of statistical dependence to each other. This can often be attributed to the similar characteristics these signals have [17] due to their origin, e.g., being speech patterns or functional brain activity. However, looking at the ratio of the superdiagonal norm to the entire tensor norm from definition 2.6 for $\mathbf{S}$ after normalization and removing the mean, the value of $\tau_D = 0.999$ shows that norm-wise the tensor can be considered as diagonal. All information related to the mean and scale of the signals is lost due to whitening and the scaling indeterminacy of ICA. It must be noted that this is the smallest of possible examples with relatively simple signals. For larger problems we observed that this diagonality tends to be lower.

To some extent this can be related to how the tensor ranks behave as follows. Figure 4-22 shows the relative error $\varepsilon_{CP} = ||\hat{\mathcal{C}}_{\mathbf{s}}^{(4)} - \mathcal{C}_{\mathbf{s}}^{(4)}||_2 / ||\mathcal{C}_{\mathbf{s}}^{(4)}||_2$ of the estimated CPD and symmetric CPD of this cumulant tensor for varying ranks R and S_R. Both were computed using the PARAFAC and symmetric PARAFAC power iteration implementation from Tensorly [95] with standard settings.

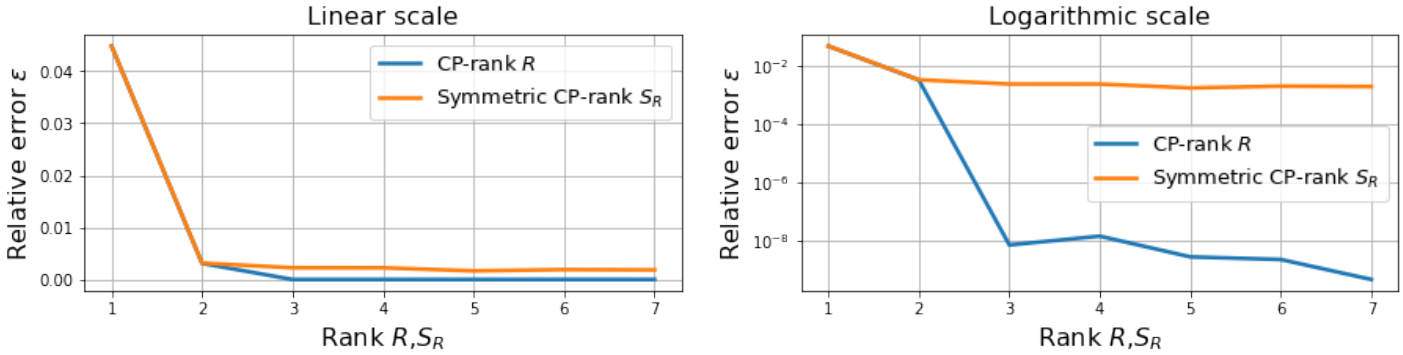


Figure 4-22: CP-rank and symmetric CP-rank estimation for the given 2 source cumulant tensor for $R = S_R = \{1, \ldots, 7\}$. The left plot shows the relative error $\epsilon = \frac{||\hat{\mathcal{C}}_{\mathbf{s}}^{(4)} - \mathcal{C}_{\mathbf{s}}^{(4)}||_2}{||\mathcal{C}_{\mathbf{s}}^{(4)}||_2}$ on a linear scale and he right plot shows the error on a logarithmic scale.

		CP	**symmetric CP**
	Error ε_{CP}	$3.08478e-3$	$3.08483e-3$
	Weights λ	$\begin{bmatrix} -0.3875 & -0.01734 \end{bmatrix}$	$\begin{bmatrix} 0.3875 & 0.01734 \end{bmatrix}$
Factors	$\mathbf{U}_1$	$\begin{bmatrix} 0.9998 & -0.03719 \\ 0.02193 & -0.9993 \end{bmatrix}$	$\begin{bmatrix} 0.9998 & 0.03719 \\ 0.02194 & 0.9993 \end{bmatrix}$
	$\mathbf{U}_2$	$\begin{bmatrix} -0.9998 & 0.03719 \\ -0.02193 & 0.9993 \end{bmatrix}$	$\mathbf{U}_1$
	$\mathbf{U}_3$	$\begin{bmatrix} -0.9998 & 0.03719 \\ -0.02193 & 0.9993 \end{bmatrix}$	$\mathbf{U}_1$
	$\mathbf{U}_4$	$\begin{bmatrix} -0.9998 & 0.03719 \\ -0.02193 & 0.9993 \end{bmatrix}$	$\mathbf{U}_1$

Table 4-1: Resulting CPD and symmetric CPD estimation of the cumulant tensor for $R = S_R = 2$. Note that the CPD is essentially identical to the symmetric CPD as all of the present minuses cancel each other out.

For this particular example, the figure initially suggests that the symmetric CP-rank and non-symmetric CP-rank are not estimated to be identical. Looking at the errors on a logarithmic scale suggests that the non-symmetric rank is ≥ 3 while the symmetric rank is suggested to be 2 due to the halt in error decrease. However, both seem to have an identical error for $R = S_R = 2$. Close inspection of the resulting decomposition in Table 4-1 shows that both methods have found the exact same decomposition.

Note that all of the minuses in the non-symmetric CPD cancel each other out which effectively makes it a symmetric decomposition. This suggests that the symmetric CP-rank and CP-rank are identical [55] and are equivalent to the number of source components. The CPD can be symmetric for different ranks R too but is not equivalent to the symmetric CPD as is the case for $R = S_R = 2$. The residual error of the symmetric CPD remains pretty much constant for $3 \geq S_R \leq 7$ which is most likely due to there not existing an exact CPD for this range of S_R. This non-perfect tensor rank matches up with the non perfect diagonality of the cumulant tensor. While the original tensor is not perfectly diagonal, the symmetric CP-rank estimation does manage to give a good indication of the amount of independent components the cumulant tensor represents. This is under the assumption that the problem is not greatly under-determined $N >> P$ as explained in appendix B-1. Figure 4-23 shows that better symmetric CPD models can be found by greatly increasing the rank. However, estimating a greater amount of sources than there are observations present $N >> P$, rarely produces reliable results [21][17], hence the assumption.

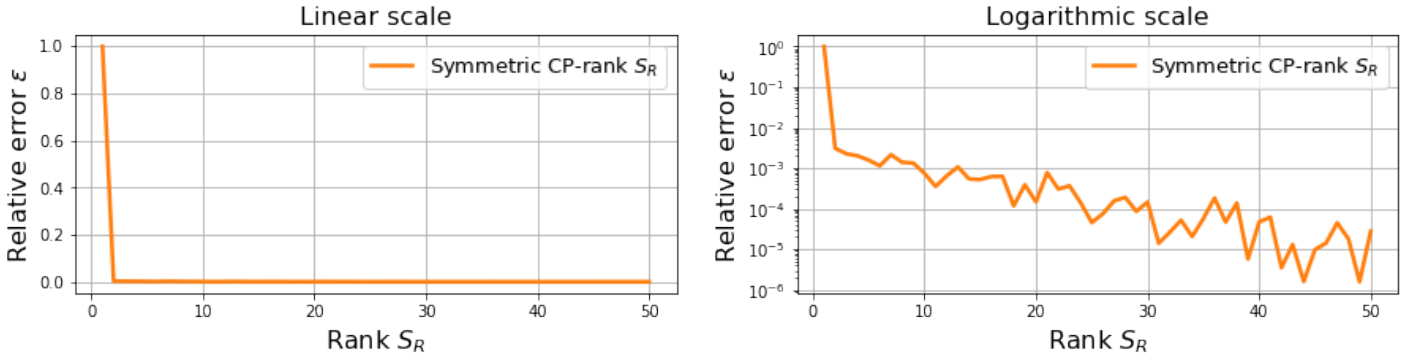


Figure 4-23: Symmetric CP-rank estimation for the given 2 source cumulant tensor for $R = S_R = \{1, \ldots, 50\}$. The left plot shows the relative error $\epsilon = \frac{||\hat{\mathcal{C}}_{\mathbf{s}}^{(4)} - \mathcal{C}_{\mathbf{s}}^{(4)}||_2}{||\mathcal{C}_{\mathbf{s}}^{(4)}||_2}$ on a linear scale and the right plot shows the error on a logarithmic scale.

Mixed case Switching from the non-mixed case to the case where the signals have been linearly mixed but now whitened, the rank estimation becomes a lot harder. The previously shown 2 signals are now mixed with a linear mixing model $\mathbf{X} = \mathbf{AS}$ where the mixing matrix is sampled from a uniform distribution over $[0, 1)$:

$$\mathbf{A} = \begin{bmatrix} 0.5508 & 0.7081 \\ 0.2909 & 0.5108 \end{bmatrix}, \tag{4-42}$$

which results in the following 2 observed mixtures shown in figure 4-24.

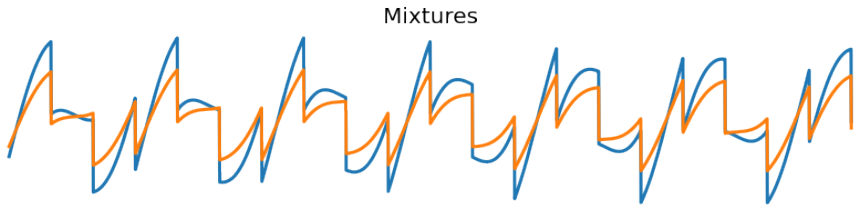


Figure 4-24: Two mixtures $\mathbf{x}_1$ and $\mathbf{x}_2$ of the original source signals $\mathbf{s}_1$ and $\mathbf{s}_1$.

The corresponding fourth-order cumulant tensor $\mathcal{C}_{\mathbf{x}}^{(4)}$ of these mixtures is given as:

$$\left[\begin{array}{c|c} \mathcal{C}_{\mathbf{x}}^{(4)}(:,:,1,1) & \mathcal{C}_{\mathbf{x}}^{(4)}(:,:,1,2) \\ \hline \mathcal{C}_{\mathbf{x}}^{(4)}(:,:,2,1) & \mathcal{C}_{\mathbf{x}}^{(4)}(:,:,2,2) \end{array}\right] = \left[\begin{array}{cc|cc} -4.5e-2 & -2.5e-2 & -2.5e-2 & -1.4e-2 \\ -2.5e-2 & -1.4e-2 & -1.4e-2 & -7.9e-3 \\ \hline -2.5e-2 & -1.4e-2 & -1.4e-2 & -7.9e-3 \\ -1.4e-2 & -7.9e-3 & -7.9e-3 & -4.5e-2 \end{array}\right]. \tag{4-43}$$

Compared to its unmixed counterpart $\mathcal{C}_{\mathbf{s}}^{(4)}$, the differences between the diagonal and offdiagonal entries of the mixture cumulant tensor are a lot smaller. The corresponding measure of diagonality has now a much lower value of $\tau_D = 0.586$ which This shows how maximizing τ_D, i.e. the diagonal entries entries, can lead to a solution akin to the original source components. The decrease in τ_D signifies the distribution of information previously present on the superdiagonal now along the offdiagonal entries. This impacts the difficulty of estimating the tensor rank.

Figure 4-25 shows the relative estimation errors for the observed mixtures. The rank cannot be estimated anymore as clearly as was the case with the source cumulant tensor. The fact that the tensor from (4-43) has higher off-diagonal entries makes it harder for the CPD and symmetric CPD to find a good fit. Looking at the symmetric CP-rank, it is now suggested that the tensor is of rank 4 instead of 2. On top of that, the CPD and symmetric CPD do not share a comon solution anymore.

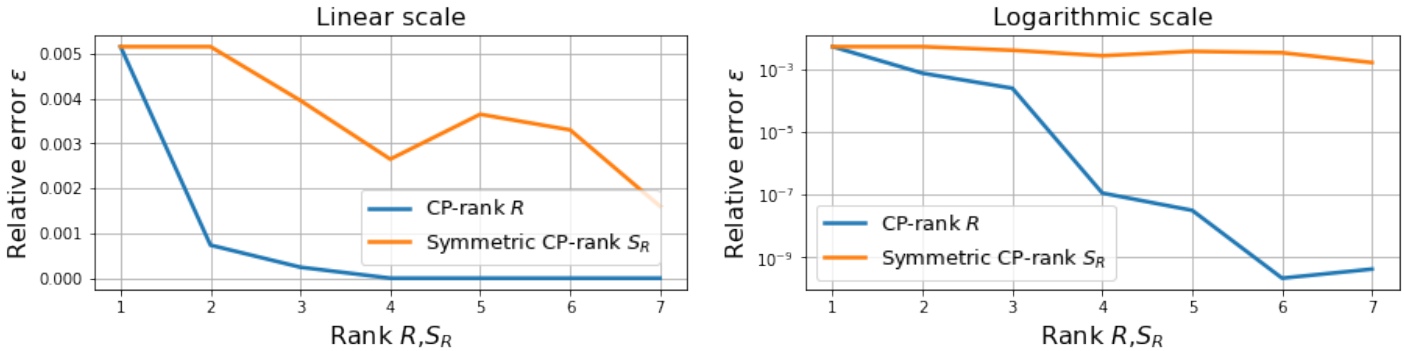


Figure 4-25: CP-rank and symmetric CP-rank estimation for the given 2 observed mixtures cumulant tensor for $R = S_R = \{1, \ldots, 7\}$. The left plot shows the relative error $\epsilon = \frac{||\hat{\mathcal{C}}_{\mathbf{s}}^{(4)} - \mathcal{C}_{\mathbf{s}}^{(4)}||_2}{||\mathcal{C}_{\mathbf{s}}^{(4)}||_2}$ on a linear scale and the right plot shows the error on a logarithmic scale.

This example is an extreme simplification of real world BSS problems as it consists of only 2 signals without any noise. On top of that, the solution is already known which introduces a bias in how results are analysed and evaluated. In real practical problems only the information related to the mixed case is present. It is for this reason why one should remain skeptical of any cumulant tensor CP-rank R or symmetric CP-rank S_R estimation as it may deviate from the true number of latent source components. However, the rank estimation problem can to some extent be simplified through the use of whitening and HOSVD.

Whitening and HOSVD The importance of pre-whitening data for ICA was already explained in section 3-1-3. Here it is shown how whitening combined with HOSVD can help even more with simplifying the BSS problem. HOSVD is often used to initialize the estimation of a CPD so it is natural to consider it for ICA too.

First, the previously shown mixtures $\mathbf{x}$ are whitened with an SVD based whitening transformation $\mathbf{W}$:

$$\mathbf{z} = \mathbf{W}\mathbf{x}. \tag{4-44}$$

and result in the following cumulant tensor:

$$\left[\begin{array}{c|c} \mathcal{C}_{\mathbf{z}}^{(4)}(:,:,1,1) & \mathcal{C}_{\mathbf{z}}^{(4)}(:,:,1,2) \\ \hline \mathcal{C}_{\mathbf{z}}^{(4)}(:,:,2,1) & \mathcal{C}_{\mathbf{z}}^{(4)}(:,:,2,2) \end{array}\right] = \left[\begin{array}{cc|cc} -9.1e-1 & -2.9e-1 & -2.9e-1 & -5.7e-1 \\ -2.9e-1 & -5.7e-1 & -5.7e-1 & 1.8e-1 \\ \hline -2.9e-1 & -5.7e-1 & -5.7e-1 & 1.8e-1 \\ -5.7e-1 & 1.8e-1 & 1.8e-1 & -6.7e-1 \end{array}\right]. \tag{4-45}$$

This tensor has a measure of diagonality of $\tau_D = 0.588$ which has little difference compared to the non whitened case. However, the benefit of whitening becomes apparent when looking at the rank estimation shown below in Figure 4-26. The same rank behaviour as with the source signals is seen here for the symmetric CP-rank which suggests a rank of $S_R = 2$.

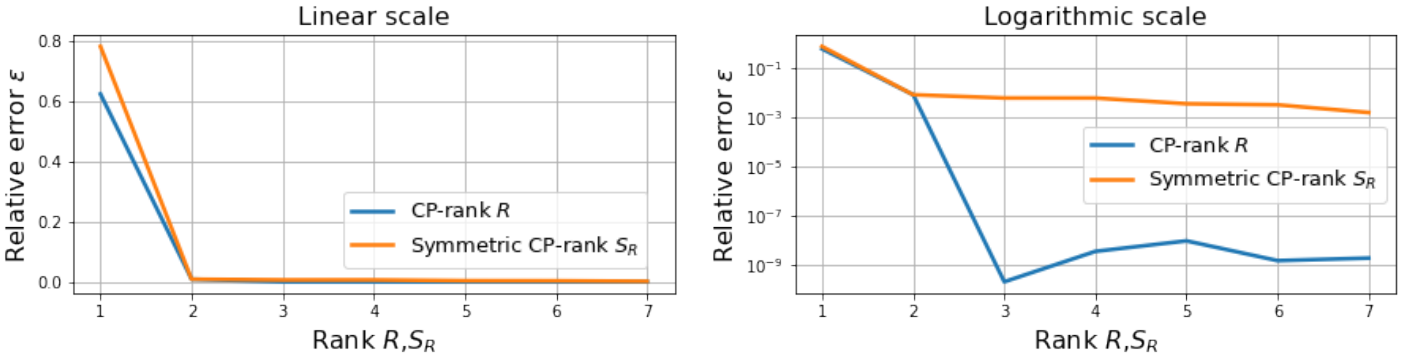


Figure 4-26: CP-rank and symmetric CP-rank estimation for the given 2 observed mixtures cumulant tensor for $R = S_R = \{1, \ldots, 7\}$ after whitening. The left plot shows the relative error $\epsilon = \frac{||\hat{\mathcal{C}}_{\mathbf{s}}^{(4)} - \mathcal{C}_{\mathbf{s}}^{(4)}||_2}{||\mathcal{C}_{\mathbf{s}}^{(4)}||_2}$ on a linear scale and the right plot shows the error on a logarithmic scale.

On top of that, for $R = S_R = 2$ the CPD and symmetric CPD show again a near identical relative error. Inspecting the numerical results presented in Table 4-2 shows that the regular CPD has again resulted in a symmetric format which is near identical to the symmetric CPD. As was the case with the source signal cumulant tensor, this can be a good indication that the correct amount of source components have been estimated.

		CP	**symmetric CP**
	Error ε_{CP}	$8.495e-3$	$8.497e-3$
	Weights λ	$\begin{bmatrix} 1.504 & 1.200 \end{bmatrix}$	$\begin{bmatrix} -1.200 & -1.504 \end{bmatrix}$
Factors	$\mathbf{U}_1$	$\begin{bmatrix} 0.8472 & 0.5730 \\ 0.5313 & -0.8196 \end{bmatrix}$	$\begin{bmatrix} -0.5731 & 0.8472 \\ 0.8194 & 0.5313 \end{bmatrix}$
	$\mathbf{U}_2$	$\begin{bmatrix} -0.8472 & -0.5730 \\ -0.5313 & 0.8196 \end{bmatrix}$	$\mathbf{U}_1$
	$\mathbf{U}_3$	$\begin{bmatrix} -0.8472 & -0.5730 \\ -0.5313 & 0.8196 \end{bmatrix}$	$\mathbf{U}_1$
	$\mathbf{U}_4$	$\begin{bmatrix} -0.8472 & -0.5730 \\ -0.5313 & 0.8196 \end{bmatrix}$	$\mathbf{U}_1$

Table 4-2: Resulting CPD and symmetric CPD estimation of the cumulant tensor for $R = S_R = 2$ for the whitened case $\mathbf{z}$. Note that the CPD is essentially identical to the symmetric CPD as all of the present minuses cancel each other out.

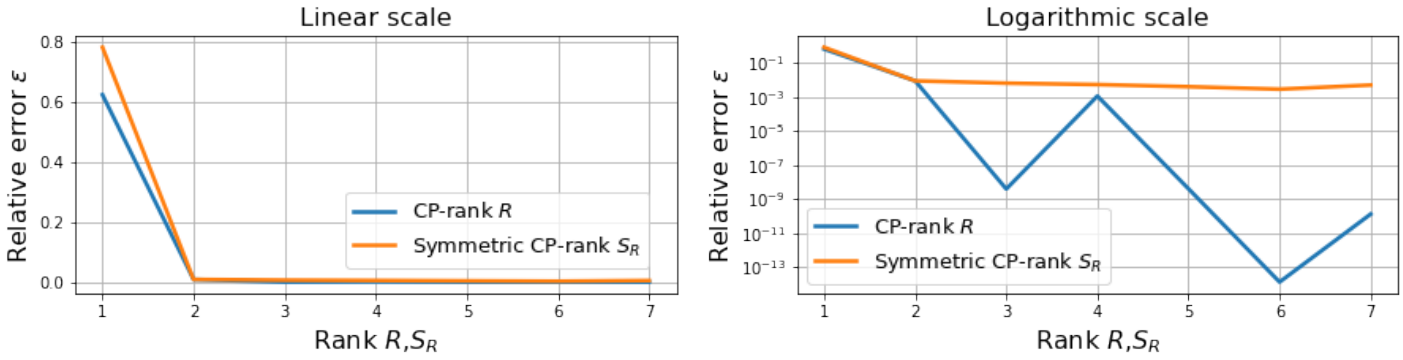


Figure 4-27: CP-rank and symmetric CP-rank estimation for the given 2 observed mixtures cumulant tensor for $R = S_R = \{1, \ldots, 7\}$ after whitening and applying HOSVD. The left plot shows the relative error $\epsilon = \frac{||\hat{\mathcal{C}}_{\mathbf{s}}^{(4)} - \mathcal{C}_{\mathbf{s}}^{(4)}||_2}{||\mathcal{C}_{\mathbf{s}}^{(4)}||_2}$ on a linear scale and the right plot shows the error on a logarithmic scale.

Next the whitened data is additionally transformed using the inverse HOSVD factor matrix $\mathbf{U}$ of the cumulant tensor and the multi-linearity property of the cumulant tensor 3.5: $\mathbf{z}' = \mathbf{U}^T\mathbf{z} = \mathbf{U}^T\mathbf{W}\mathbf{x}$. Computing the cumulant tensor $\mathcal{C}_{\mathbf{z}'}^{(4)}$ yields the following set of slices:

$$\left[\begin{array}{c|c} \mathcal{C}^{(4)}_{\mathbf{z}'}(:,:,1,1) & \mathcal{C}^{(4)}_{\mathbf{z}'}(:,:,1,2) \\ \hline \mathcal{C}^{(4)}_{\mathbf{z}'}(:,:,2,1) & \mathcal{C}^{(4)}_{\mathbf{z}'}(:,:,2,2) \end{array}\right] = \left[\begin{array}{cc|cc} -1.5 & 1.2e-1 & 1.2e-1 & -3.9e-2 \\ 1.2e-1 & -3.9e-2 & -3.9e-2 & -1.5e-1 \\ \hline 1.2e-1 & -3.9e-2 & -3.9e-2 & -1.5e-1 \\ -3.9e-2 & -1.5e-1 & -1.5e-1 & -1.2 \end{array}\right]. \tag{4-46}$$

which have a measure of diagonality of $\tau_D = 0.977$. The results show that HOSVD has made the tensor more diagonal compared to the observed mixtures cumulant tensor. The corresponding symmetric CP-rank estimation errors shown in Figure 4-27 show no difference.

The benefit of applying HOSVD on this example can be seen when looking at the transformed mixtures. Figure 4-28 show the signals after only whitening and Figure 4-29 after whitening and HOSVD. The latter shows that whitening of the data combined with the HOSVD of the cumulant tensor has managed to unmix the source signals adequately, hence the high value for τ_D.

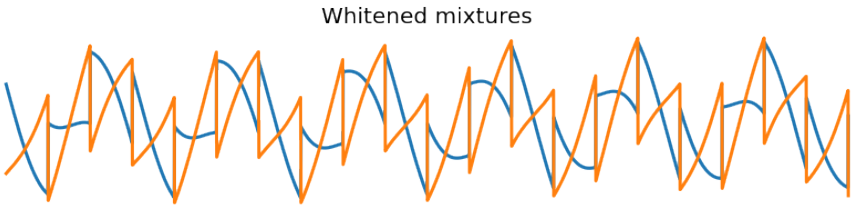


Figure 4-28: Two mixtures $\mathbf{x}_1$ and $\mathbf{x}_2$ of the original source signals $\mathbf{s}_1$ and $\mathbf{s}_1$ after whitening.

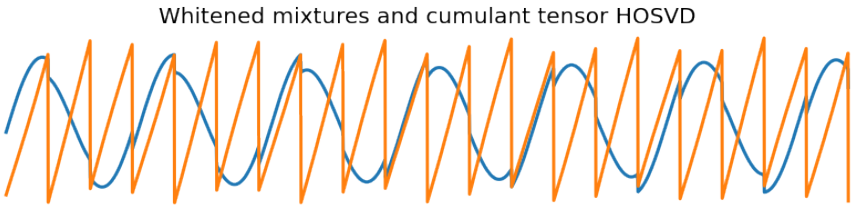


Figure 4-29: Two mixtures $\mathbf{x}_1$ and $\mathbf{x}_2$ of the original source signals $\mathbf{s}_1$ and $\mathbf{s}_1$ after whitening and applying HOSVD.

It is pointed out that the presented example is extremely simple and more often than not HOSVD will fail to unmix the data as was mentioned before. Moreover, the results from this example cannot be generalized to all cases. However, it does illustrate estimating the rank of the cumulant tensor can benefit from applying both whitening and HOSVD. This is not restriced to only CPD but is applicable in general for ICA methods. The pre-processing step of applying the HOSVD on the cumulant tensor after whitening is shown below in definition 4.14.

Definition 4.14: HOSVD pre-processing

HOSVD pre-processing consists of applying the inverse factor matrix of fourth-order cumulant tensor $\mathcal{C}_{\mathbf{z}}^{(4)}$ HOSVD:

$$\mathcal{C}_{\mathbf{z}}^{(4)} = \mathcal{C}_{\mathbf{z}'}^{(4)} \times_1 \mathbf{U} \times_2 \mathbf{U} \times_3 \mathbf{U} \times_4 \mathbf{U} \tag{4-47}$$

on a whitened random vector $\mathbf{z} \in \mathbb{R}^P$:

$$\mathbf{z}' = \mathbf{U}^T \mathbf{z} = \mathbf{U}^T \mathbf{W} \mathbf{x}. \tag{4-48}$$

where $\mathbf{W}$ is the whitening transformation. Due to $\mathbf{U}$ being orthogonal the transformed vector $\mathbf{z}'$ keeps the whitened property from whitened vector $\mathbf{z}$.

Canonical polyadic decomposition and orthogonality We have deliberately avoided the topic of the orthogonality of the factor matrices up to this point to not further complicate the introduction of CPD and symmetric CPD. Most CPD computation methods such as Alternating Least Squares (ALS) or Non-linear Least Squares (NLS) do not produce orthogonal factor matrices. The effects of having a (semi-)orthogonal constraint on the factor matrices or some of the factor matrices was extensively studied by the authors of [91]. They established that the uniqueness of a CPD with at least one (semi-)orthogonal factor can be proven under even milder conditions than was shown in the previous paragraph. On top of that, the authors propose an Orthogonally-constraint-Alternating-Least-Squares (OALS) method to compute such a decomposition. After pointing out the weaknesses of the OALS method the authors of [96] propose a different scheme based on the probabilistic inference framework.

Considering these methods for ICA solves the problem of having no orthogonality of the factor matrices but introduces several others. First of all, pointed out by the authors of [96] OALS is not a robust method. On top of that, it requires knowledge on the rank of the tensor *a priori* as the rank is used to structure the problem. The probabilistic inference method from [96] presents a key issue too. Namely that the resulting factor matrices are not guaranteed to be identical and hence there is no guarantee a symmetric CPD is computed. By no guarantee it is meant that there is a probability of computing a symmetric CPD similarly to how the CPD presented in Table 4-1 and Table 4-2 resulted in a symmetric CPD by appropriate tensor rank choice.

Nevertheless, needing to have *a priori* knowledge on the cumulant tensor's rank or possibly breaking the symmetry of the cumulant tensor and abusing the multi-linearity property 3.5 of the cumulant tensor are not considered as desirable compromises for the formulation of a CPD based ICA algorithm. We do somewhat contradict this statement as the implicit Generalized EigenValue Decomposition (GEVD) based CPD algorithm presented in the following section breaks symmetry and does not result in all orthogonal factor matrices. However, we opted to proceed with the implicit GEVD based algorithm as its explicit counterpart is often used for finding reliably good initial estimates of a CPD due to it being a deterministic method.

4-4-3 Computing the Canonical Polyadic decomposition

Typical approaches to obtaining a CPD are through ALS [79][80] or NLS [97][98]. However, the former is known to not be accurate in the case when over-estimating the tensor CP-rank R and the latter is relatively much slower due to the much higher per iteration computational cost. On top of that, the quality of the solution of both methods is sensitive to what values are used for initialization. For these reasons we present three devised algorithms for computing the CPD of the cumulant tensor in implicit fashion. First a deterministic approach which is derived from an already existing method commonly used for initialization of CPD algorithms [99]. Secondly, a relatively simple first-order gradient approach and lastly a fixed-point iteration which both surprisingly bear a lot of similarity to FastICA.

4-4-4 CPD through generalized eigenvalue decomposition

Here we present our deterministic algorithm for computing the CPD of a cumulant tensor implicitly. The algorithm is based on the trilinear decomposition method from [99] which uses generalized eigenvalue decomposition to find the factors of a 3-way tensor. Our implicit Canonical-Polyadic-Generalized-Eigenvalue-decomposition (CP-GEVD) is presented below in algorithm 4 and its workings are explained in the following paragraphs.

Algorithm 4 Implicit CP-GEVD

Require: Whitened data: $\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T & \mathbf{z}_2^T & \dots & \mathbf{z}_P^T \end{bmatrix}^T \in \mathbb{R}^{1P \times I}$, Rank: R

procedure IMPLICIT CPD-GEVD($\mathbf{Z}$)
  $\mathbf{U}_P \leftarrow$ ImplicitHOSVD($\mathbf{Z}$), From ImplicitHOSVD($\mathbf{Z}$) save: $\mathbf{M}_{ij} \forall i = j = \{1, \dots, R\}$
  $\mathbf{G}_1 = \mathbf{U}_P^T \mathbf{M}_{11}, \quad \mathbf{G}_2 = \mathbf{U}_P^T \mathbf{M}_{22}$
  GEVD of pencil: $(\mathbf{G}_1, \mathbf{G}_2) \rightarrow \boldsymbol{\lambda}, \mathbf{V}_{\text{eig}}$
  Cast imaginary part of second in complex conjugate pair as new part:
  $\mathcal{R}e(\lambda_i) \in \mathbb{R}, \mathcal{R}e(\mathbf{v}_{\text{eig},i}) \in \mathbb{R}^P \rightarrow \lambda_i, \mathbf{v}_{\text{eig},i} \quad \forall i \in S_{\text{complexpairs}(i,j)}$
  $\mathcal{I}m(\lambda_j) \in \mathbb{C}, \mathcal{I}m(\mathbf{v}_{\text{eig},j}) \in \mathbb{C}^P \rightarrow \lambda_j, \mathbf{v}_{\text{eig},j} \quad \forall j \in S_{\text{complexpairs}(i,j)}$
  Inwards projection
  $\mathbf{X} = [\mathbf{M}_{11}, \dots, \mathbf{M}_{RR}]$
  for $r = 1, \dots, R$ **do**
    $\mathbf{v}_r^T \leftarrow \text{SVD}\left(\mathbf{V}_{\text{eig}}^{-1} \mathbf{U}_P^T \mathbf{M}_{rr}\right)$ keep first right orthogonal vector
  end for
  $\mathbf{U}_{\text{out}}^i = [\mathbf{v}_1, \dots, \mathbf{v}_r] \quad \forall i \in \{2, \dots, 4\}$
  $\mathbf{K} = \left[\left(\left[\mathbf{U}_{\text{out}}^2 (1,:) \right]^2 \odot \mathbf{U}_{\text{out}}^2 \right)^T, \dots, \left(\left[\mathbf{U}_{\text{out}}^2 (R,:) \right]^2 \odot \mathbf{U}_{\text{out}}^2 \right)^T \right]^T$
  $\mathbf{U}_{\text{out}}^1 = \mathbf{X} \left(\mathbf{K}^T \right)^\dagger$
  for $r = 1, \dots, R$ **do**
    $\mathbf{U}_{\text{out}}^i(:,r) \leftarrow \frac{\mathbf{U}_{\text{out}}^i(:,r)}{||\mathbf{U}_{\text{out}}^i(:,r)||_2^2}, \quad \lambda_r = ||\mathbf{U}_{\text{out}}^1(:,r)||_2^2 \cdot \dots \cdot ||\mathbf{U}_{\text{out}}^N(:,r)||_2^2 \quad \forall i = \{1, 2, \dots, 4\}$
  end for
  return $[\![\boldsymbol{\lambda}; \mathbf{U}_{\text{out}}^1, \dots, \mathbf{U}_{\text{out}}^N]\!]$
end procedure

It is important to note that this algorithm does not produce a symmetric CPD due to $\mathbf{U}^1_{\text{out}}$ being different.

Generalized Eigenvalue Decomposition After applying implicit HOSVD on the cumulant tensor the first step in the CP-GEVD algorithm is the GEVD of the matrix pencil $(\mathbf{G}_1, \mathbf{G}_2)$. The definition of GEVD is shown below in definition 4.15.

Definition 4.15: Generalized Eigenvalue problem

The **generalized eigenvalue** problem is the problem of finding a nonzero vector $\mathbf{v}$ for which the following equality holds true for 2 equally shaped square matrices $\mathbf{A}$ and $\mathbf{B}$:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{B}\mathbf{v}. \tag{4-49}$$

Any such vector $\mathbf{v}$ for which the above holds true is called a **generalized eigenvector** The scalar λ is its corresponding **generalized eigenvalue** for which the following must hold true:

$$\det(\mathbf{A} - \lambda\mathbf{B}) = 0. \tag{4-50}$$

In [99] it is explained that by finding all generalized eigenvectors of the two frontal slices of a 3-tensor $\mathcal{X} \in \mathbb{R}^{P\times P\times 2}$ its CPD can be computed directly and algebraically. For 3-way tensors that have more than 2 frontal slices or that have more than 3 modes the method can be used to compute an approximation of the CPD factors. Several methods are suggested for the selection of used slices but the main proposed method is to use HOSVD on the tensor and take the first two front most slices as they contain the principal components with largest variances. This means that for a tensor with the indices $(p_1, p_2, p_3, \ldots, p_{N-1}, p_N)$ its slices at $(:, :, 1, \ldots, 1, 1)$ and $(:, :, 1, \ldots, 1, 2)$ are taken.

For the higher-order whitened cumulant tensor $\mathcal{C}^{(4)}_{\mathbf{z}'}(p_1, p_2, p_3, p_4)$ this amounts to selecting the slices $\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 1, 1)$ and $\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 1, 2)$. However, we propose a different selection criterion. Due to symmetry many of the cumulant tensor's slices show partial symmetry to another. This holds true too for $\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 1, 2)$ compared to $\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 1, 1)$. The point is to compute the GEVD with as much information as possible as this will maximize the probability of finding a good CPD estimation. So instead of $\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 1, 2)$ as $\mathbf{G}_2$, the slice $\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 2, 2)$ is chosen for $\mathbf{G}_2$ as this slices contains more unique information when compared to $\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 1, 1)$. The shared amount of unique values these slices have is shown in table Table 4-3.

	$\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 1, 2)$	$\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 2, 2)$
$\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 1, 1)$	$P^2 + (P-2)$	$P^2 + (P-1)$

Table 4-3: Table showing the amount of unique values the slices on the top have together with slice $\mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 1, 1)$ for a higher-order whitened cumulant tensor of size $\mathcal{C}^{(4)}_{\mathbf{z}'} \in \mathbb{R}^{[4,P]}$.

For the computation of the slices $\mathbf{G}_1 = \mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 1, 1)$ and $\mathbf{G}_2 = \mathcal{C}^{(4)}_{\mathbf{z}'}(:, :, 2, 2)$ the regular cumu-

lant tensor slices $\mathcal{C}_{\mathbf{z}}^{(4)}(:,:,1,1)$ and $\mathcal{C}_{\mathbf{z}}^{(4)}(:,:,2,2)$ computed during the previous implicit HOSVD step and are multiplied from the left with $\mathbf{U}^T$. All slices $\mathcal{C}_{\mathbf{z}}^{(4)}(:,:,r,r) \quad \forall r = 1,2,\ldots,R$ are stored as they are needed for the following inwards projection step. The slices are denoted in algorithm 4 as $\mathbf{M}_{ii} \forall i = 1,2,\ldots,R$.

Casting complex conjugates In the case that the GEVD produces eigenvalue-eigenvector pairs which are complex conjugates of each others these must be cast to being only real vectors. The reason behind this is that real valued signals produce real cumulant values. The used independenc solution of the BSS problem is defined as diagonalizing the cumulant tensor in its original space. For ICA with real valued variables this means that cumulant tensor must be diagonalized in real space $\mathbb{R}^{[4,P]}$. As complex signals are not considered in this thesis, any complex conjugate eigen-pairs from the GEVD must be dealt with. However, this cannot simply be done by discarding the imaginary part of such eigen-pairs. This will result in duplicate eigenvectors $\mathbf{v}_{\text{eig}}$ which means that not all columns of $\mathbf{V}_{\text{eig}}$ are linearly independent and hence do not represent separate components. Instead, for each complex conjugate eigen-pair (i,j) in the set of complex conjugate eigen-pairs $S_{\text{complexpairs}}$, one of the eigen-pairs is set to only its real part and the other eigen-pair to only its imaginary part casted as real:

$$\begin{aligned} &\mathcal{R}e(\lambda_i) \in \mathbb{R}, \mathcal{R}e(\mathbf{v}_{\text{eig},i}) \in \mathbb{R}^P \rightarrow \lambda_i, \mathbf{v}_{\text{eig},i} \quad \forall i \in S_{\text{complexpairs}(i,j)} \\ &\mathcal{I}m(\lambda_j) \in \mathbb{C}, \mathcal{I}m(\mathbf{v}_{\text{eig},j}) \in \mathbb{C}^P \rightarrow \lambda_j, \mathbf{v}_{\text{eig},j} \quad \forall j \in S_{\text{complexpairs}(i,j)}. \end{aligned} \tag{4-51}$$

This ensures that the columns of $\mathbf{V}_{\text{eig}}$ are at the very least different from each other. The casting of complex conjugate eigen-pairs in such a fashion is done too in other Eigen-Value-Decomposition (EVD) based ICA algorithms [100][17] for the same reasons when considering real valued signals.

Inwards projection As the name suggests, the inwards projection step consists of projecting the transformed tensor inwards which results in the final factor matrix $\mathbf{U}_{\text{out}}^1$. This step is in fact the least squares solution of the best fitting final factor matrix. In algorithm 4 this is performed in lines 7 to 13. The authors of [99] use the entire tensor in this step which for the cumulant tensor is impractical. The objective here is to bypass forming the entire tensor. In order to so, we only use the previously stored slices $\mathcal{C}_{\mathbf{z}}^{(4)}(:,:,r,r) \quad \forall r = 1,2,\ldots,R$. These are the R first core slices from definition 4.5. Each r'th core slice contains the self-cumulants $\text{cum}(z_r, z_r, z_r, z_r)$ of the r'th variable in the random vector $\mathbf{z} = \begin{bmatrix} z_1 & z_2 & \ldots & z_P \end{bmatrix}^T$ which are the values on the cumulant tensor superdiagonal. On top of that, the slice contains all of the cross-cumulants $\text{cum}(z_r, z_r, z_i, z_j) \quad \forall i,j = 1,2,\ldots,P$ closest to z_r which contain the most information about how z_r is related to the other z's. It is for these reasons why the core slices are considered as a suitable sub-selection for the inwards projection step.

Each core slice's HOSVD counterpart $\mathcal{C}_{\mathbf{z}'}^{(4)}(:,:,r,r) \quad \forall r = 1,2,\ldots,R$ from definition 4.14 is transformed using the inverse of the found generalized eigenvectors $\mathbf{V}_{\text{eig}}$. From this transformed slice its first principal component is taken as the first right orthogonal vector through its SVD:

$$\mathbf{v}_r^T \xleftarrow[Firstcomponent]{SVD} \mathbf{V}_{\text{eig}}^{-1}\mathcal{C}_{\mathbf{z}'}^{(4)}(:,:,r,r) = \mathbf{V}_{\text{eig}}^{-1}\mathbf{U}_P^T\mathcal{C}_{\mathbf{z}}^{(4)}(:,:,r,r), \tag{4-52}$$

which contains the largest part of information of that slice and hopefully, contains most information relevant to the source component s_r.

The final step is the actual inwards projection. In [99] it is shown that this can be done in the least squares sense by projecting the original tensor inwards using the inverse estimated factor matrices $\mathbf{U}_i \forall i = 2,3,4$. In matricized format this amounts to multiplying the mode-1 matricization of the original tensor $\mathcal{T}$ with the pseudo-inverse of a transposed Khatri-Rao product of the factor matrices:

$$\mathbf{U}_{\text{out}}^1 = \mathbf{T}_{(1)}\left(\mathbf{K}^T\right)^\dagger, \quad \text{with} \quad \mathbf{K} = \left[\mathbf{U}_{\text{out}}^2 \odot \mathbf{U}_{\text{out}}^3 \odot \mathbf{U}_{out}^4\right]. \tag{4-53}$$

In our algorithm 4 only the 'core' slices are used so only the corresponding columns of $\mathbf{K}$ are needed. These columns have the same index behaviour as the cumulant tensor as was presented in lemma 4.1 so at the indices $\mathbf{K}(:, rP^2+rP+1 : rP^2+rP+P) \quad \forall r = 2,\ldots,4$. Instead of selecting these columns they can be computed directly as:

$$\mathbf{K} = \begin{bmatrix} \mathbf{U}_{\text{out}}^2(1,:) * \mathbf{U}_{\text{out}}^3(1,:) \odot \mathbf{U}_{\text{out}}^4 \\ \vdots \\ \mathbf{U}_{\text{out}}^2(R,:) * \mathbf{U}_{\text{out}}^3(R,:) \odot \mathbf{U}_{\text{out}}^4 \end{bmatrix} = \begin{bmatrix} \left[\mathbf{U}_{\text{out}}^2(1,:)\right]^2 \odot \mathbf{U}_{\text{out}}^2 \\ \vdots \\ \left[\mathbf{U}_{\text{out}}^2(R,:)\right]^2 \odot \mathbf{U}_{\text{out}}^2 \end{bmatrix} \tag{4-54}$$

where the rightmost equivalence is thanks to $\mathbf{U}_{\text{out}}^2 = \mathbf{U}_{\text{out}}^3 = \mathbf{U}_{\text{out}}^4$. After the inwards projection the final step is normalizing the corresponding factors and storing the resulting scalar values into $\boldsymbol{\lambda}$.

Complexities The time complexity and storage complexity of algorithm 4 are listed down below in property 4.4. The algorithm from [99] has a dominant computational cost of $O(P^5)$. On top of that, the cost of computing the cumulant tensor $O(IP^4)$ itself has to be factored in too.

The dominant computational costs of our algorithm 4 are the implicit HOSVD step: $O(\frac{1}{2}IP^4) + O(P^5(\log(P)^2))$ and the inwards projection step: $O(RP^4)$ which amounts to a total of $O(\frac{1}{2}IP^4) + P^5(\log(P)^2) + RP^4)$.

Comparing this total cost with that of the original cost of $O(IP^4 + P^5)$ shows that our algorithm 4 scales worse with P than the algorithm from [99] due to the logarithmic term. Additionally this does comes at the loss of not using all information of the cumulant tensor as the implicit CPD method only uses a selection of the cumulant tensor matrices whereas [99] uses the entire tensor. The storage complexity however has been reduced from $O(P^4)$ to $O(RP^2)$. This effectively means that theoretically there has been a trade-off between computational speed and storage requirement.

Property 4.4: CPD-GEVD complexities

Computational complexity The computational complexity of the implicit CP-GEVD algorithm is dominated by the cost of the HOSVD-iter operation and the inwards projection: $O(\frac{1}{2}IP^4 + P^5(\log(P)^2) + RP^4)$.

Storage complexity The storage complexity of the implicit CP-GEVD algorithm is dominated by the R tensor slices that have to be stored at a cost of $O(RP^2)$.

U-Implicit CP-GEVD and C-Implicit CP-GEVD The implicit CP-GEVD allows for a U-type and C-type variant as well depending on the type of implicit HOSVD algorithm used. In both cases the dominant computational complexity changes to that of the specific implicit HOSVD version used + the inwards projection step $O(RP^4)$. Note that where the full implicit HOSVD produced an identical result as the explicit HOSVD, the implicit GEVD method does not produce the same result as its explicit counterpart. This is due to the smaller approximate inwards projection step. The original CP-GEVD algorithm from [99] can be found in appendix C-7.

In the coming section we present a different method of computing the CPD of the cumulant tensor implicitly.

4-4-5 CPD through first-order optimization

In [101][102] the CPD of the cumulant tensor is reduced to a single step least-squares optimization process by exploiting its symmetry. Alternatively, the authors of [103] propose a first-order optimization method for the CPD of any tensor which is shown to be more accurate than ALS while keeping the per iteration cost near identical to that of ALS. Inspired by this method, Kolda shows in [104] how the first-order method can be used for more efficient computation of the CPD of symmetric tensors.

First order optimization problem

The base optimization problem Kolda describes in [104] is given below in definition 4.16. The least-squares approach of computing the CPD of a symmetric tensor is solved with a gradient-based approach. The gradients of the weights λ_r and the factor vectors $\mathbf{u}_r$ can be computed as shown due to the symmetry of the tensor $\mathcal{X}$.

Definition 4.16: First-order optimization for symmetric CPD [104]

In order to obtain the rank-R CPD of a symmetric tensor $\mathcal{X} \in \mathbb{R}^{[N,P]}$ as defined in 4.13, the following non-convex optimization problem needs to be solved:

$$\min_{\boldsymbol{\lambda},\mathbf{U}} f(\boldsymbol{\lambda},\mathbf{U}) \equiv \|\mathcal{X} - \hat{\mathcal{X}}\|^2 \quad \text{s.t.} \quad \hat{\mathcal{X}} \equiv \sum_{r=1}^{R} \lambda_r \mathbf{u}_r^{\circ N}, \tag{4-55}$$

with $\mathbf{U} \in \mathbb{R}^{P\times R}$ and $\boldsymbol{\lambda} \in \mathbb{R}^R$.
From [103], the **gradients** of this problem are defined as:

$$\frac{\partial f}{\partial \lambda_r} = -2\left[\mathcal{X}\mathbf{u}_r^N - \sum_{k=1}^{R} \lambda_k \langle \mathbf{u}_r, \mathbf{u}_k\rangle^N\right] \quad \text{and} \tag{4-56}$$

$$\frac{\partial f}{\partial \mathbf{u}_r} = -2N\lambda_r\left[\mathcal{X}\mathbf{u}_r^{N-1} - \sum_{k=1}^{R} \lambda_k \langle \mathbf{u}_r, \mathbf{u}_k\rangle^{N-1}\mathbf{u}_k\right] \tag{4-57}$$

where $\mathcal{X}\mathbf{u}_r^N$ and $\mathcal{X}\mathbf{u}_r^{N-1}$ denote the TTSV operation defined in 2.16 for all modes and all modes but one. With these gradients the optimization problem can be solved using a **first-order** optimization method. The TTSV operations dominate the cost of computing the gradients at a cost of $O(RP^N)$.

A recent work by Sherman and Kolda [53] expands Kolda's first-order optimization CPD method for symmetric tensors from definition 4.16 to Higher Order Statistics (HOS) moment tensors. It presents how the first-order optimization problem can be rewritten such that higher-order moment tensors never have to be explicitly formed in order to compute their CPD. It primarily exploits the outer-product from definition 4.9 with which such tensors are created. More specifically, the algorithm exploits the fact that for a random vector $\mathbf{x} \in \mathbb{R}^P$ with I samples, the N'th empirical moment tensor $\mathcal{M}_{\mathbf{x}}^{(d)}$ can be expressed as the sum of I rank-one tensors:

$$\mathcal{M}_{\mathbf{x}}^{(N)} = \frac{1}{I} \cdot \left(\sum_{l=1}^{I} \mathbf{x}(\ell)^{\circ N} \right), \tag{4-58}$$

where $\mathbf{x}(\ell)$ denotes the ℓ'th sample of the random vector $\mathbf{x}$. However, Sherman and Kolda do not show how their method can be used for the fourth-order cumulant tensor. Due to the additional outer-product of second order terms, shown below in definition 4.17, the computation of the gradient and cost function becomes more complex.

Definition 4.17: Fourth-order cumulant tensor full expression

The **fourth-order** cumulant tensor for a **zero-mean** random vector $\mathbf{x} \in \mathbb{R}^P$ with I samples in its full tensor form is expressed as:

$$\begin{aligned} \mathcal{C}_{\mathbf{x}}^{(4)} =& \mathcal{M}_{\mathbf{x}}^{(4)} - \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} - \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1} - \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2} \\ & \text{with} \quad \sigma_1 = \{1,3,2,4\}, \quad \sigma_2 = \{1,4,3,2\} \end{aligned} \tag{4-59}$$

where the fourth order moment tensor $\mathcal{M}_{\mathbf{x}}^{(4)}$ and the covariance matrix $\mathbf{C}_{\mathbf{x}}^{(2)}$ are computed using the elementwise operations defined in 3-20.

Before we present our final algorithm for the first-order CPD optimization of the cumulant tensor, we present the derivation of the gradients. Furthermore, we present the final expression for the computation of the cost function. This is done for the zero-mean non-whitened case after which it is simplified even further for the whitened data case.

Computing the gradients implicitly We rewrite the elementwise operations for the fourth-order moment tensor and second-order moment tensors from 3-20 as summations of symmetric outer-products as follows:

$$\begin{aligned} \mathcal{M}_{\mathbf{x}}^{(4)} &= \mathbb{E}\left[\mathbf{x}^{\otimes 4}\right] = \frac{1}{I} \cdot \left(\sum_{l=1}^{I} \mathbf{x}(\ell)^{\otimes 4} \right) \\ \mathbf{C}_{\mathbf{x}}^{(2)} &= \mathbb{E}\left[\mathbf{x}^{\otimes 2}\right] = \frac{1}{I} \cdot \left(\sum_{l=1}^{I} \mathbf{x}(\ell)^{\otimes 2} \right), \end{aligned} \tag{4-60}$$

where $\mathbf{x}(\ell)$ denotes the ℓ'th sample of the zero-mean random vector $\mathbf{x}$. Using definition 4.17 the full cumulant tensor can be expressed as the following series of symmetric outer-product summations:

$$\begin{aligned} \mathcal{C}^{(4)} =& \frac{1}{I} \sum_{\ell=1}^{I} \mathbf{x}(\ell)^{\circ 4} - \frac{1}{I^2} \left(\sum_{\ell=1}^{I} \mathbf{x}(\ell)^{\circ 2} \right) \circ \left(\sum_{k=1}^{I} \mathbf{x}(k)^{\circ 2} \right) \\ & - \frac{1}{I^2} \left(\left(\sum_{\ell=1}^{I} \mathbf{x}(\ell)^{\circ 2} \right) \circ \left(\sum_{k=1}^{I} \mathbf{x}(k)^{\circ 2} \right) \right)^{T\sigma=\{1,3,2,4\}} \\ & - \frac{1}{I^2} \left(\left(\sum_{\ell=1}^{I} \mathbf{x}(\ell)^{\circ 2} \right) \circ \left(\sum_{k=1}^{I} \mathbf{x}(k)^{\circ 2} \right) \right)^{T\sigma=\{1,4,3,2\}} \end{aligned} \tag{4-61}$$

Using equation (4-58), lemma 4.3 and lemma 4.4 we rewrite the TTSV operation of the gradient (C-49) for the moment tensor part as follows:

$$\mathcal{M}_{\mathbf{x}}^{(4)}\mathbf{u}_r^3 = \frac{1}{I}\sum_{\ell=1}^{I}\left(\mathbf{x}(\ell)^{\circ 3}\right)\mathbf{u}^3 = \frac{1}{I}\sum_{\ell=1}^{I}\langle\mathbf{x}(\ell),\mathbf{u}\rangle^3\,\mathbf{x}(\ell). \tag{4-62}$$

Lemma 4.3: [53]

Let $\mathbf{a}$,$\mathbf{b} \in \mathbb{R}^P$. Then the inner-product of the N-way symmetric rank-one tensors constructed from the vectors satisfies:

$$\left\langle \mathbf{a}^{\circ N}, \mathbf{b}^{\circ N} \right\rangle = \langle \mathbf{a}, \mathbf{b}\rangle^N. \tag{4-63}$$

Lemma 4.4: [53]

Let $\mathbf{a}$,$\mathbf{b} \in \mathbb{R}^P$. Let $\mathcal{X} \in \mathbb{R}^{[N,P]}$ be the symmetric rank-one tensor defined by $\mathcal{X} = \mathbf{b}^{\circ N}$. Then the TTSV in all modes but one of tensor $\mathcal{X}$ with the vector $\mathbf{a}$ satisfies:

$$\mathcal{X}\mathbf{a}^{N-1} = \langle \mathbf{a}, \mathbf{b}\rangle^{N-1}\mathbf{b}. \tag{4-64}$$

Furthermore, the TTSV in all modes satisfies:

$$\mathcal{X}\mathbf{a}^{N} = \langle \mathbf{a}, \mathbf{b}\rangle^{N}. \tag{4-65}$$

Lemma 4.5: [53]

Using the symmetric outer-product definition 4.9 and the elementwise power operation from definition 2.9. Let $\mathcal{X} = \frac{1}{I}\sum_{\ell=1}^{I}\mathbf{x}_\ell^{\circ N} \in \mathbb{R}^{[N,P]}$ and $\mathbf{u} \in \mathbb{R}^P$, $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_I \end{bmatrix} \in \mathbb{R}^{P\times I}$. Then $\mathcal{X}\mathbf{u}^{N-1} = \frac{1}{I}\mathbf{X}\left[\mathbf{X}^T\mathbf{u}\right]^{N-1}$.

By defining a matrix $\mathbf{Y} \in \mathbb{R}^{P\times R}$ of which each column $\mathbf{y}_r = \mathbf{y}_r^{\mathcal{M}} - \mathbf{y}_r^{\mathbf{C}} \in \mathbb{R}^P$ for $r \in \{1,\dots,R\}$ represents a separate TTSV operation for $\mathbf{u}_r$, we compute each subpart $\mathbf{y}_r^{\mathcal{M}}$ containing the moment tensor with lemma 4.5 as follows:

$$\mathbf{y}_r^{\mathcal{M}} = \mathcal{M}_{\mathbf{x}}^{(4)}\mathbf{u}_r^3 = \frac{1}{I}\mathbf{X}\left[\mathbf{X}^T\mathbf{u}_r\right]^3 \quad \forall r = \{1,2,\dots,R\}. \tag{4-66}$$

Close inspection shows that the outer-product of the second-order cumulant tensor with itself from equation (4-61) is identical to:

$$\left(\sum_{\ell=1}^{I}\mathbf{x}_\ell^{\circ 2}\right)\circ\left(\sum_{k=1}^{I}\mathbf{x}_k^{\circ 2}\right) = \sum_{\ell=1}^{I}\sum_{k=1}^{I}\mathbf{x}_\ell^{\circ 2}\circ\mathbf{x}_k^{\circ 2}. \tag{4-67}$$

Lemma 4.6:

Given the following sequence of vector outer-products:

$$\mathbf{x}_1 \circ \mathbf{x}_2 \circ \mathbf{x}_3 \circ \mathbf{x}_4. \tag{4-68}$$

The index permutation from definition 2.4 of this sequence is identical to the permutation of the order of the vectors. For example with $\sigma = \{1, 3, 2, 4\}$:

$$(\mathbf{x}_1 \circ \mathbf{x}_2 \circ \mathbf{x}_3 \circ \mathbf{x}_4)^{T\sigma} = \mathbf{x}_1 \circ \mathbf{x}_3 \circ \mathbf{x}_2 \circ \mathbf{x}_4. \tag{4-69}$$

Using lemma 4.6, we rewrite the tensor transposes of (4-67) as:

$$\begin{aligned} \left(\left(\sum_{\ell=1}^{I} \mathbf{x}_\ell^{\circ 2}\right) \circ \left(\sum_{k=1}^{I} \mathbf{x}_k^{\circ 2}\right)\right)^{T\sigma=\{1,3,2,4\}} &= \sum_{\ell=1}^{I}\sum_{k=1}^{I} \mathbf{x}_\ell \circ \mathbf{x}_k \circ \mathbf{x}_\ell \circ \mathbf{x}_k \\ \left(\left(\sum_{\ell=1}^{I} \mathbf{x}(\ell)^{\circ 2}\right) \circ \left(\sum_{k=1}^{I} \mathbf{x}(k)^{\circ 2}\right)\right)^{T\sigma=\{1,4,3,2\}} &= \sum_{\ell=1}^{I}\sum_{k=1}^{I} \mathbf{x}(\ell) \circ \mathbf{x}(k) \circ \mathbf{x}(k) \circ \mathbf{x}(\ell) \end{aligned} \tag{4-70}$$

Combining equations (4-61), (4-67) and (4-70), the computation of the TTSV of all modes but one $\mathcal{C}_{\mathbf{x}}^{(4)}\mathbf{u}_r^3$ for the part $\mathbf{y}_r^{\mathbf{C}}$ is rewritten into the following split TTSV expression:

$$\begin{aligned} \mathbf{y}_r^{\mathbf{C}} = &\left[\frac{1}{I^2}\sum_{\ell=1}^{I}\sum_{k=1}^{I} \mathbf{x}_\ell^{\circ 2} \circ \mathbf{x}_k^{\circ 2}\right] \mathbf{u}_r^3 \\ &+ \left[\frac{1}{I^2}\sum_{\ell=1}^{I}\sum_{k=1}^{I} \mathbf{x}_\ell \circ \mathbf{x}_k \circ \mathbf{x}_\ell \circ \mathbf{x}_k\right] \mathbf{u}_r^3 \\ &+ \left[\frac{1}{I^2}\sum_{\ell=1}^{I}\sum_{k=1}^{I} \mathbf{x}_\ell \circ \mathbf{x}_k \circ \mathbf{x}_k \circ \mathbf{x}_\ell\right] \mathbf{u}_r^3. \end{aligned} \tag{4-71}$$

Finally, using lemmas 4.4, 4.7 and 4.8 we can simplify the TTSV further as:

$$\mathbf{y}_r^{\mathbf{C}} = 3\frac{1}{I^2}\left(\mathbf{u}_r^T\mathbf{X}\mathbf{X}^T\mathbf{u}_r\right)\mathbf{X}\mathbf{X}^T\mathbf{u}_r \tag{4-72}$$

Lemma 4.7:

Let $\mathbf{X} = \begin{bmatrix} \mathbf{x}(1) & \mathbf{x}(2) & \dots & \mathbf{x}(I) \end{bmatrix} \in \mathbb{R}^{P\times I}$, $\mathbf{u} \in \mathbb{R}^P$ and let S_4 be the set consisting of all permutations of $S = \{1, 2, 3, 4\}$. Then for any permutation $\sigma \in S$ it holds that:

$$\left[\left(\sum_{\ell=1}^{I}\sum_{k=1}^{I} \mathbf{x}_\ell \circ \mathbf{x}_k \circ \mathbf{x}_\ell \circ \mathbf{x}(k)\right)^{T\sigma}\right] \mathbf{u}^3 = \sum_{\ell=1}^{I}\sum_{k=1}^{I} \langle \mathbf{x}_\ell, \mathbf{u}\rangle^2 \langle \mathbf{x}_k, \mathbf{u}\rangle \mathbf{x}_k. \tag{4-73}$$

Lemma 4.8:

Let $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_I \end{bmatrix} \in \mathbb{R}^{P \times I}$ and $\mathbf{u} \in \mathbb{R}^P$.
Then:

$$\sum_{\ell=1}^{I} \sum_{k=1}^{I} \langle \mathbf{x}_\ell, \mathbf{u} \rangle^2 \langle \mathbf{x}_k, \mathbf{u} \rangle \mathbf{x}_k = \left(\mathbf{u}^T \mathbf{X} \mathbf{X}^T \mathbf{u} \right) \mathbf{X} \mathbf{X}^T \mathbf{u}. \tag{4-74}$$

The TTSV of the gradient of λ_r is computed by taking the inner-product of $\mathbf{y}_R$ and $\mathbf{u}_r$. For completeness up to this point the gradients are computed for the non-whitened case which contains the whitened case too. For explicitly the whitened case, we simplify the TTSV operations of the gradients even further as follows:

$$\begin{aligned} \mathbf{y}_r^{\mathbf{C}} &= \left[\mathbf{I}_P \circ \mathbf{I}_P + (\mathbf{I}_P \circ \mathbf{I}_P)^{T\sigma_1} + (\mathbf{I}_P \circ \mathbf{I}_P)^{T\sigma_2} \right] \mathbf{u}_r^3 \\ &\text{with} \quad \sigma_1 = [1,3,2,4], \quad \sigma_2 = [1,4,3,2] \end{aligned} \tag{4-75}$$

$$\mathbf{y}_r^{\mathbf{C}} = 3 \left[\mathbf{I}_P \circ \mathbf{I}_P \right] \mathbf{u}_r^3 = 3 \langle \mathbf{u}_r, \mathbf{u}_r \rangle \cdot \mathbf{u}_r \tag{4-76}$$

This results in the final gradients presented in definition 4.18 down below.

Definition 4.18: Gradients first-order CPD cumulant tensor problem

The **gradients** of the first-order optimization problem from definition 4.16 for the **cumulant tensor** $\mathcal{C}^{(4)}$ of non-whitened data $\mathbf{X}$ are defined as:

$$\frac{\partial f}{\partial \lambda_r} = -2 \left[\mathbf{y}_r^T \mathbf{u}_r - \sum_{k=1}^{R} \lambda_k \langle \mathbf{u}_r, \mathbf{u}_k \rangle^4 \right] \quad \text{and} \tag{4-77}$$

$$\frac{\partial f}{\partial \mathbf{u}_r} = -2 \cdot 4 \lambda_r \left[\mathbf{y}_r - \sum_{k=1}^{R} \lambda_k \langle \mathbf{u}_r, \mathbf{u}_k \rangle^3 \mathbf{u}_k \right], \tag{4-78}$$

with each $\mathbf{y}_r \in \mathbb{R}^P$ for $r \in \{1, \dots, R\}$ is defined as

$$\mathbf{y}_r = \mathcal{C}^{(4)} \mathbf{u}_r^3 = \frac{1}{I} \mathbf{X} \left[\mathbf{X}^T \mathbf{u}_r \right]^3 - 3 \frac{1}{I^2} \left(\mathbf{a}_r^T \mathbf{a}_r \right) \mathbf{X} \mathbf{a}_r \quad \text{with} \quad \mathbf{a}_r = \mathbf{X}^T \mathbf{u}_r. \tag{4-79}$$

By first computing $\mathbf{a}_r$ separately, the dominant computational cost of equation (4-79) remains $O(IP)$. derivation for this can be found in appendix C-9.

For whitened data $\mathbf{Z} \in \mathbb{R}^{P \times I}$ the computation of each TTSV operation $\mathbf{y}_r$ is simplified as:

$$\mathbf{y}_r = \mathcal{C}^{(4)} \mathbf{u}_r^3 = \frac{1}{I} \mathbf{Z} \left[\mathbf{Z}^T \mathbf{u}_r \right]^3 - 3 \langle \mathbf{u}_r, \mathbf{u}_r \rangle \cdot \mathbf{u}_r. \tag{4-80}$$

Comparing the result from equation 4-79 for the cumulant tensor with that of Sherman and Kolda's fourth-order moment tensor from equation4-66, shows that the computation of each vector $\mathbf{y}_r$ has gained the additional term: $-3 \frac{1}{I^2} \left(\mathbf{a}_r^T \mathbf{a}_r \right) \mathbf{X} \mathbf{a}_r$ with $\mathbf{a}_r = \mathbf{X}^T \mathbf{u}_r$. The

additional computational cost of this term is $O\left(2IP + I\right)$ per vector $\mathbf{y}_r$ which amounts to a total additional cost of $O\left(R \cdot (2IP + I)\right)$. This shows that the implicit version reduces the R amount of TTSV operations of the cumulant tensor from $O(RP^4)$ to $O(IPR)$ while simultaneously avoiding the $O(IP^4)$ cost of forming the tensor. The storage complexity of the implicit case is $O(PR)$ for both the outputs and intermediaries.

For completeness the computation for λ_r is kept while due to the scaling indeterminacy of ICA it can effectively be discarded. Only its sign is relevant. Alternatively, the computation of λ_r can be considered as a normalization step during each iteration when solving the first-order optimization problem. However, discarding λ as a variable from the optimization problem may cause overflow of underflow issues in the computation of the vectors $\mathbf{u}_r$. On top of that, keeping λ ensures that no sign functions are needed as is the case with the fastICA gradient.

Computing the cost function The written out cost function from definition 4.16 for the cumulant tensor is presented below in definition 4.19. As $\|\mathcal{C}_{\mathbf{x}}^{(4)}\|^2$ is a constant, it can be essentially left-out of the algorithm. Sherman and Kolda [53] propose to replace $\|\mathcal{C}_{\mathbf{x}}^{(4)}\|^2$ by adding an additional parameter α to their method. The reason being that for tensors of large size ($P >> 2$) one wants to avoid ever computing the tensor. Any estimate of the tensor inner product or arbitrary large enough value for α which keeps the cost positive will suffice. The decrease in cost is of interest which is of course relative. However, for completeness we keep the computation of $\|\mathcal{C}_{\mathbf{x}}^{(4)}\|^2$ here.

Definition 4.19: Function first-order CPD cumulant tensor problem

The **cost function** from 4.16 for a **zero-mean** random variable $\mathbf{x} \in \mathbb{R}^P$ with I samples equals:

$$\|\mathcal{C}_{\mathbf{x}}^{(4)} - \hat{\mathcal{C}}_{\mathbf{x}}^{(4)}\|^2 = \|\mathcal{C}_{\mathbf{x}}^{(4)}\|^2 + \|\hat{\mathcal{C}}_{\mathbf{x}}^{(4)}\|^2 - 2\langle \mathcal{C}_{\mathbf{x}}^{(4)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle. \tag{4-81}$$

By defining the vector $\mathbf{v}_I = \begin{bmatrix} \frac{1}{I} & \dots & \frac{1}{I} \end{bmatrix}^T \in \mathbb{R}^I$ equation (4-81) for the **cumulant tensor** can be completely written out into:

$$\begin{aligned} \|\mathcal{C}_{\mathbf{x}}^{(4)}\|^2 =& \mathbf{v}_I^T \left[\mathbf{X}^T\mathbf{X}\right]^4 \mathbf{v}_I - 6 \cdot \mathbf{v}_I^T \left[\mathbf{X}^T\mathbf{X}\right]^2 \operatorname{diag}_2(\mathbf{v}_I) \left[\mathbf{X}^T\mathbf{X}\right]^2 \mathbf{v}_I \\ &+ 3 \cdot \left[\mathbf{v}_I^T \left[\mathbf{X}^T\mathbf{X}\right]^2 \mathbf{v}_I\right]^2 + 6 \cdot \mathbf{v}_I^T \left[\mathbf{X}^T\mathbf{X} \operatorname{diag}_2(\mathbf{v}_I) \mathbf{X}^T\mathbf{X}\right]^2 \mathbf{v}_I \end{aligned}$$

$$\|\hat{\mathcal{C}}_{\mathbf{x}}^{(4)}\|^2 = \boldsymbol{\lambda}^T \left[\mathbf{U}^T\mathbf{U}\right]^4 \boldsymbol{\lambda} \tag{4-82}$$

$$\langle \mathcal{C}_{\mathbf{x}}^{(4)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle = \mathbf{v}_I^T \left[\mathbf{X}^T\mathbf{U}\right]^4 \boldsymbol{\lambda} - 3 \cdot \mathbf{v}_I^T \left[\left[\mathbf{X}^T\mathbf{U}\right]^2 \operatorname{diag}_2(\boldsymbol{\lambda}) \left[\mathbf{U}^T\mathbf{X}\right]\right]^2 \mathbf{v}_I.$$

The full derivation of the cost function can be found in appendix C-10.

The algorithm Our final algorithm for computing the gradients and cost of the first-order optimization problem for the cumulant tensor is presented below. It contains several addi-

tional steps which speed up the process by eliminating duplicate operations such $\mathbf{B} = \mathbf{U}^T\mathbf{U}$ and $\mathbf{C} = [\mathbf{U}]^3$. The algorithm computes the gradients and cost of the optimization problem. As such it can be used with any first-order solver in Matlab or Python. The complexities of the algorithm we derived are given in property 4.5.

Algorithm 5 Implicit computation of cumulant tensor CPD gradients

Require: I,$\mathbf{Z}$,$\mathbf{U}$,$\boldsymbol{\lambda}$,α

procedure IMPLICIT COMPUTATION(I,$\mathbf{X}$,$\boldsymbol{\lambda}$,$\mathbf{U}$,α,R)
  for $r = 1, \ldots, R$ **do**
    $\mathbf{a}_r = \mathbf{X}^T\mathbf{u}_r$
    $\mathbf{y}_r = \frac{1}{I}\mathbf{X}\left[\mathbf{X}^T\mathbf{u}_r\right]^3 - 3\frac{1}{I^2}\left(\mathbf{a}_r^T\mathbf{a}_r\right)\mathbf{X}\mathbf{a}_r$
    $w_r = \mathbf{u}_r^T\mathbf{y}_r$
  end for
  $\mathbf{B} = \mathbf{U}^T\mathbf{U}$
  $\mathbf{C} = [\mathbf{B}]^3$
  $\mathbf{v} = (\mathbf{B} * \mathbf{C})\,\boldsymbol{\lambda}$
  $f = \alpha + \boldsymbol{\lambda}^T\mathbf{v} - 2\mathbf{w}^T\boldsymbol{\lambda}$
  $\mathbf{g}_{\boldsymbol{\lambda}} = -2(\mathbf{w} - \mathbf{v})$
  $\mathbf{G}_{\mathbf{U}} = -8(\mathbf{Y} - \mathbf{U}\mathbf{D}_{\boldsymbol{\lambda}}\mathbf{C})\mathbf{D}_{\boldsymbol{\lambda}}$
  return $f, \mathbf{g}_{\boldsymbol{\lambda}}, \mathbf{G}_{\mathbf{U}}$
end procedure

PROPERTY 4.5: COMPLEXITIES OF IMPLICIT CPD GRADIENT METHOD

Computational complexity The dominant computational complexity for computing the gradients with algorithm 5 is $O(IRP)$.

Storage complexity The storage complexity of the implicit CPD algorithm is $O(RP)$.

Important to note is that in our base optimization problem the factor matrix $\mathbf{U}$ is not constrained to be orthogonal. We discuss this in the following section where the gradients are compared with that of fastICA.

Comparison of implicit CPD with FastICA

By comparing implicit CPD with FastICA for whitened data $\mathbf{Z} \in \mathbb{R}^{P \times I}$ we make several key observations.

First of all, both gradients can be used for ICA to estimate a single column of $\mathbf{U}$ which amounts to estimating a single source component. However, both optimization problems have no orthogonality constraints so estimating all columns of $\mathbf{U}$ simultaneously will result in a non-orthogonal result. As the whitened ICA solution is defined to be an orthogonal transformation, the found result will not be a proper solution. Hyvärinen clearly mentions in [33] and [21] that in order to estimate all components simultaneously with the gradient of fastICA (3-37), each iteration of the estimate $\mathbf{U}$ must be symmetrically orthogonalized. This ensures that the whiteness and the orthogonality of the solution is maintained. This implies that symmetric orthogonalization is needed if the implicit CPD gradient of equation (C-49) is to be used for ICA. Definition 4.20 presents one such symmetric orthogonalization scheme based on eigenvalue decomposition. The name symmetric is somewhat misleading as the resulting matrix is not specifically made symmetric. Symmetric refers to the fact that all columns of the matrix $\mathbf{U} \in \mathbb{R}^{P \times R}$ are treated equally. It is sufficient to find any orthogonal basis for the subspace spanned by the original columns. As a result, after symmetric orthgonalization $\mathbf{U}^T\mathbf{U} = \mathbf{I}_P \in \mathbb{R}^{P \times P}$.

Definition 4.20: Symmetric orthogonalization [21]

A matrix $\mathbf{U} \in \mathbb{R}^{P \times R}$ can be **symmetrically orthogonalized** as:

$$\mathbf{U} \leftarrow \left(\mathbf{U}\mathbf{U}^T\right)^{-1/2}\mathbf{U} \tag{4-83}$$

where the the inverse square root of $\mathbf{U}\mathbf{U}^T$ is obtained through its eigendecomposition:

$$\left(\mathbf{U}\mathbf{U}^T\right)^{-1/2} = \mathbf{E}\,\mathrm{diag}\left(d_1^{-1/2}, \ldots, d_R^{-1/2}\right)\mathbf{E}^T, \tag{4-84}$$

where $\mathbf{E} \in \mathbb{R}^{P \times P}$ contains the left eigenvectors and $d_i \quad \forall i = 1, \ldots, R$ are the corresponding eigenvalues.

Secondly, when estimating all components simultaneously with a symmetric orthogonalization method in place, both gradients are simplified even further due to orthogonality:

$$\text{FastICA:} \quad \frac{\partial f}{\partial \mathbf{u}_i} = \quad 4 \cdot \mathrm{sign}(\mathrm{kurt}(\mathbf{u}_i^T\mathbf{Z})) \cdot \left[\frac{1}{I}\mathbf{Z}\left[\mathbf{Z}^T\mathbf{u}_i\right]^3 - 3\mathbf{u}_i\right]. \tag{4-85}$$

$$\text{Implicit CPD:} \quad \frac{\partial f}{\partial \mathbf{u}_r} = -2 \cdot \lambda_r \cdot 4 \cdot \left[\frac{1}{I}\mathbf{Z}\left[\mathbf{Z}^T\mathbf{u}_r\right]^3 - 3\mathbf{u}_r - \lambda_r \cdot \mathbf{u}_r\right] \quad \text{and}$$

$$\frac{\partial f}{\partial \lambda_r} = -2\left[\frac{1}{I}\mathbf{Z}\left[\mathbf{u}_r^T\mathbf{Z}^T\mathbf{u}_r\right]^3 - 3 - \lambda_r\right] \tag{4-86}$$

Recall that excess kurtosis is defined in 3.4 as the departure from a Gaussian distribution. For the whitened case this departure is computed by subtracting the kurtosis of a Gaussian

distribution which is always of value 3. This is where the $-3\mathbf{u}_i$ term from gradient (4-85) originates from. Looking at the simplified gradients from equation (4-86) and rearranging the right-hand sides as

$$\frac{\partial f}{\partial \mathbf{u}_r} = -2 \cdot \lambda_r \cdot 4 \cdot \left[\frac{1}{I} \mathbf{Z} \left[\mathbf{Z}^T \mathbf{u}_r \right]^3 - (3 + \lambda_r) \cdot \mathbf{u}_r \right], \quad \frac{\partial f}{\partial \lambda_r} = -2 \left[\frac{1}{I} \mathbf{u}_r^T \mathbf{Z} \left[\mathbf{Z}^T \mathbf{u}_r \right]^3 - (3 + \lambda_r) \right] \tag{4-87}$$

results in a slightly different departure of $(3 + \lambda_r)$. This departure of $(3 + \lambda)$ ensures that the gradients of (4-87) point in the direction of a locally best fitting CPD model whereas fastICA's departure points in the direction of least-Gaussianity. (4-87) can be interpreted as fastICA with a CPD constraint.

Implicit CPD as a fixed-point iteration Due to the similarity the gradient of implicit CPD has with that of fastICA it is logical to ask the question how a fixed-point iteration would work. Hyvärinen clearly shows [21] that for ICA the direction of the gradient is important due to the scaling indeterminacy. As such, a fixed-point iteration will work too and will be much faster than any gradient method. Writing the gradients from (4-87) as fixed-point iterations[1] by discarding all constants results in the following proportional gradients:

$$\begin{aligned}\frac{\partial f}{\partial \mathbf{u}_r} &\propto \lambda_r \cdot \left[\frac{1}{I}\mathbf{Z}\left[\mathbf{Z}^T\mathbf{u}_r\right]^3 - (3+\lambda_r)\cdot\mathbf{u}_r\right] \\ \frac{\partial f}{\partial \lambda_r} &\propto \frac{1}{I}\mathbf{u}_r^T\mathbf{Z}\left[\mathbf{Z}^T\mathbf{u}_r\right]^3 - (3+\lambda_r).\end{aligned} \tag{4-88}$$

This results in the following fixed-point algorithm shown below in 6. Next to the function value f the algorithm contains an additional convergence error computation ϵ. The computation of this error is shown in the next chapter containing numerical comparisons. The idea behind using a fixed-point method instead of a gradient one is that it can be a lot faster. The complexities of algorithm 6 are identical to those of the previously presented gradient method. However, due to the fixed-point iteration the values of lambda will not correspond to the actual values, they merely point in the direction of the better estimate. As such, the computed gradient of $\boldsymbol{\lambda}$ is used internally and is initialized each iteration as a unit vector.

Algorithm 6 Implicit CPD fixed-point iteration

Require: I,$\mathbf{Z}$,$\mathbf{U}$,$\boldsymbol{\lambda}$,α, number of components R, function tolerance tol, max iterations K

procedure IMPLICIT COMPUTATION(I,$\mathbf{X}$,$\boldsymbol{\lambda}$,$\mathbf{U}$,α)
  while $\varepsilon > tol$ or $i < K$ **do**
    $\boldsymbol{\lambda} = [1,\ldots,1]^T \in \mathbb{R}^R$
    for $r = 1,\ldots,R$ **do**
      $\mathbf{a}_r = \mathbf{X}^T\mathbf{u}_r$
      $\mathbf{y}_r = \frac{1}{I}\mathbf{X}\left[\mathbf{X}^T\mathbf{u}_r\right]^3 - 3\frac{1}{I^2}\left(\mathbf{a}_r^T\mathbf{a}_r\right)\mathbf{X}\mathbf{a}_r$
      $w_r = \mathbf{u}_r^T\mathbf{y}_r$
    end for
    $\mathbf{B} = \mathbf{U}^T\mathbf{U}$
    $\mathbf{C} = [\mathbf{B}]^3$
    $\mathbf{v} = (\mathbf{B} * \mathbf{C})\,\boldsymbol{\lambda}$
    $f_i = \alpha + \boldsymbol{\lambda}^T\mathbf{v} - 2\mathbf{w}^T\boldsymbol{\lambda}$
    $\boldsymbol{\lambda} \leftarrow \mathbf{w} - \mathbf{v}$
    $\mathbf{U} \leftarrow (\mathbf{Y} - \mathbf{U}\mathbf{D}_{\boldsymbol{\lambda}}\mathbf{C})\mathbf{D}_{\boldsymbol{\lambda}}$
    $\mathbf{U} \leftarrow \frac{\mathbf{U}\,\mathrm{diag}_2(\boldsymbol{\lambda})}{\|\mathbf{U}\,\mathrm{diag}_2(\boldsymbol{\lambda})\|_2}$
    $\mathbf{U} \leftarrow$ Symmetric Orthogonalization($\mathbf{U}$)
    $\varepsilon =$ Convergence Error function($\mathbf{U}$)
  end while
  return $\mathbf{f}, \mathbf{U}$
end procedure

[1] In a fixed-point iteration the updated variable x_{k+1} is equated to its function value $x_{k+1} = f(x_k)$.

4-5 Chapter summary and contributions

This chapter presented a set of algorithms to compute a HOSVD, QR Tucker decomposition and CPD of the cumulant tensor in implicit fashion. Starting with the former, first an implicit algorithm to compute the full HOSVD of the cumulant tensor is proposed which uses SVD updating procedures to ensure the storage cost never exceeds $O(P^4)$. However, this comes a the price of an additional per svd-update cost. When summed up this results in a higher cost than performing a single SVD on the entire matricized cumulant tensor. Literature suggests the use of HOSVD for finding initial estimates of the mixing matrix in ICA and for the computation of a cumulant tensor CPD. As such, the U and C variant of the implicit HOSVD algorithm are proposed of which the former uses only the cumulant tensor's unique slices once and the latter only its core slices. These simplifications result in lower computational costs but at the price of computing incomplete HOSVD's.

Secondly a novel algorithm called QRT is proposed which uses successive QR-decompositions to simultaneously diagonalize the outer-slices of a symmetric tensor. By iterating through all layers of the tensor a special structure is imposed in it in which the fibers extending from the diagonal values are null-vectors with only the diagonal entry as non-zero. The algorithm is inspired by the QR-algorithm for computing the EVD of symmetric matrices and as such it is hypothesized that its convergence will follow similar reasoning. However, due to the lack of multi-linear equivalents of specific lemma's no constructive proof of convergence could be derived. A computationally more efficient implicit version of the QRT algorithm is proposed for the cumulant tensor which has a per iteration cost equivalent to that of FastICA $O(IRP)$.

Lastly, the final format considered in this thesis is the CPD due to its diagonal core property. The general notion of tensor ranks is presented together with a small demonstration of how such ranks behave in relation to the cumulant tensor's diagonality. This is done to reinforce the proposal of using HOSVD as an initial estimate for computing a CPD and the mixing matrix in ICA. On top of that, it is shown how the measure of diagonality adequately reflects the independentness of all estimated components. Three methods for computing the CPD of the cumulant tensor implicitly are proposed. The first computes an approximation in deterministic through GEVD of the first 2 core tensor slices. The second method is based on a first-order gradient based optimization approach presented in [53] for the computation of a CPD of higher-order moment tensors. By rewriting the gradient approach as a fixed-point iteration an algorithm is developed which can in fact be considered as FastICA with a CPD constraint. The gradient and fixed-point method enjoy complexities identical to that of FastICA.

All of the above is summarize below in the following list of contributions.

4.1: Contributions chapter 4

- A scheme is presented based on the cumulant tensor's matricization which allows for simple yet **efficient computation** of a single **element**, a **fiber** or entire **slices** in both whitened and non-whitened case.
- It is shown that the storage cost of HOSVD of the cumulant tensor is decreased from $O(P^4)$ to $O(P2)$ using an implicit scheme.
- A **novel** algorithm (QRT) based on the QR-algorithm for matrices is presented which simultaneously **diagonalizes** the outer slices of a symmetric tensor.
- An **implicit** version of the QRT algorithm for a whitened cumulant tensor for ICA is presented which has an iteration cost of $O(IRP)$ and a storage cost of $O(RP)$.
- Due to a small study in the behavior of tensor ranks for ICA which is related to the diagonality of the cumulant tensor it is identified that the measure of diagonality adequately reflects a solutions quality of independence.
- A **first-order** optimization problem for **implicitly** computing the CPD of the cumulant tensor is presented which has an iteration cost of $O(IRP)$ and a storage cost of $O(RP)$.
- It is shown how rewriting the first-order CPD optimization problem as a **fixed-point** iteration results in fastICA with a CPD constraint which too has an iteration cost of $O(IRP)$ and a storage cost of $O(RP)$.
- Guidelines are presented which are devised to help alter the proposed implicit computation scheme and algorithms for HOS tensors other than the use fourth-order cummulant tensor.

Chapter 5

Experimental results

In this chapter the previously introduced algorithms are tested for their performance in Blind Source Separation (BSS) on an artificial dataset. The theoretical complexities of the algorithms are tested and the performance of all algorithms are compared with each other and with the renowned fastICA on performance measures such as source estimation error, diagonality and computation time. While not all shown results are to be generalized, the point is to provide a relative comparison in a controlled environment of which the solutions and mixing model is known such that the noteworthy differences can be identified and explained.

Chapter outline First the used artificial data-set and mixing model are described. Secondly, the tested algorithms and used performance measures are described together with a self devised classification algorithm. Afterwards the algorithms are put to the test in BSS of which empirical results are presented. First the non-iterative algorithms Canonical-Polyadic-Generalized-Eigenvalue-decomposition (CP-GEVD), implicit CP-GEVD, Higher-Order Singular Value Decomposition (HOSVD) and implicit HOSVD are considered and afterwards the iterative algorithms FastICA, QRT, implicit Canonical Polyadic Decomposition (CPD) gradient based and implicit CPD as a fixed point iteration.

5-1 Test setup

In this sectionthe used test-setup is explained. Afterwards the computation of the convergence error is presented and the tested algorithms are listed together with the general settings that are used such as the tolerance for the convergence error. Lastly, the metrics that measure the algorithms performance are listed.

Source signals The testset consists of 4 signals $\mathbf{s}(t) = \begin{bmatrix} s_1(t)^T & s_2(t)^T & s_3(t) & s_4(t) \end{bmatrix}^T$: a sinusoid $s_1(t) = 0.6 \cdot \sin(2t)$, a square wave $s_2(t) = \text{sign}(\sin(3t))$, a sawtooth signal $s_2(t) = 1.2 \cdot \left(t - \frac{1}{2} - \lfloor t \rfloor\right)$ and a square root signal $s_4(t) = \sqrt{t}$ for a timespan of $t = [0, 10]$ seconds which are shown in Figure 5-1. The signals are sampled I times. Relatively simple yet easy recognisable signals are chosen as this simplifies evaluating the found the solutions which is shown later on. The order and coloring of the signals presented here is maintained throughout the rest of this report.

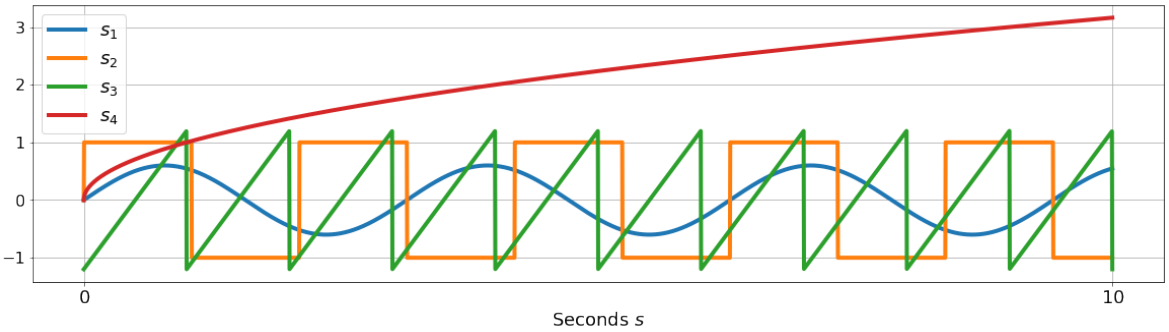


Figure 5-1: The source signals of the artificial dataset. The signals consist of a sine wave s_1, a square wave s_2, a sawtooth function s_3 and a square root function s_4 consecutively.

In order to visualize the structure and diagonality of the cumulant tensor a heatmap will be used throughout this chapter. Identically to how it was done in the Canonical-Polyadic-rank (CP-rank) section the cumulant tensor slices are tiled next to each other on a grid to form a single matrix:

$$\mathbf{M}_{\mathcal{C}^{(4)}} = \left[\begin{array}{c|c|c} \mathcal{C}_{\mathbf{s}}^{(4)}(:,:,1,1) & \dots & \mathcal{C}_{\mathbf{s}}^{(4)}(:,:,1,P) \\ \hline \vdots & \ddots & \vdots \\ \hline \mathcal{C}_{\mathbf{s}}^{(4)}(:,:,P,1) & \dots & \mathcal{C}_{\mathbf{s}}^{(4)}(:,:,P,P) \end{array}\right]. \tag{5-1}$$

However, due to the size of this matrix $\mathbf{M}_{\mathcal{C}} \in \mathbb{R}^{P^2 \times P^2}$ the actual values will not be shown. Instead, a heat-map is used as this provides an easier visual representation of the tensor's structure. Figure 5-2 shows the heat-map of the normalized source signals.

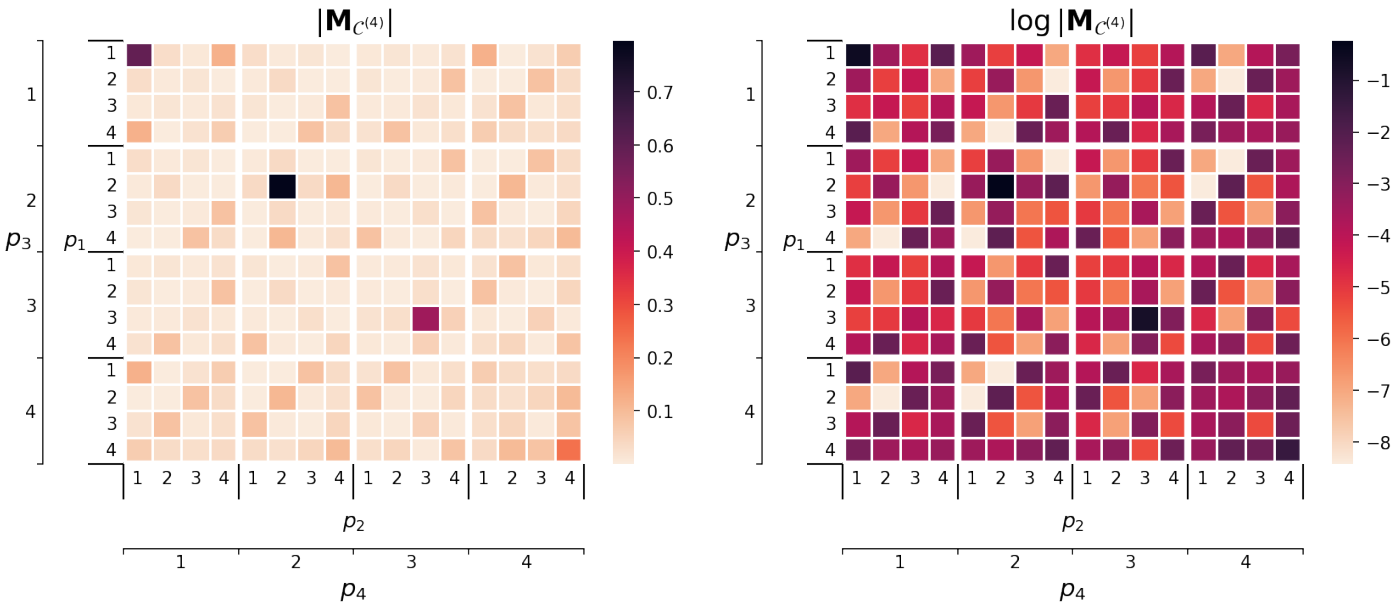


Figure 5-2: Heat-map of the cumulant tensor of the normalized source signals $\mathbf{s}$. On the left the absolute values are on a linear scale and on the right on a logarithmic scale.

The left heat-map of Figure 5-2 shows the absolute values on a linear scale and the right heatmap on a logarithmic scale. The linear scale can clearly represent the diagonality of the tensor which is to be seen by the values at the indices $p_1 = p_2 = p_3 = p_4 = 1, 2, \ldots, 4$ while the logarithmic scale reveals the difference between small values of order 10^{-3} and values of order 10^{-8} which for all intends and purposes can be considered as 0. The navigation through these heat-maps is done by following the tensor indices along both x and y axes. For example: the slice at the indices $(:, :, 1, 1)$ corresponds to the top 4 rows and left-most columns. Note that due to the tensor's symmetry the indices along the axes can be permuted in any way while the heat-map remains the same.

This particular set of heat-maps show that the source-component cumulant tensor has a clear diagonal structure. The corresponding measure of diagonality from definition 2.6 has a value of $\tau_D = 1.00$ which substantiates this observation. However, all information related to the mean and scale of the signals is lost due to whitening and the scaling indeterminacy of Independent Component Analysis (ICA). Therefore the measure of diagonality the solution should strive to have is at least that of the normalized source signals with zero-mean: $\tau_D = 0.855$.

Mixing model The mixing model used for the setup is a time-invariant linear mixing model with additive Gaussian-White-Noise (GWN) and is shown in equation (5-2). The entries of the mixing matrix $\mathbf{A} \in \mathbb{R}^{P \times N}$ are sampled from a uniform distribution over $[0, 1)$ and the noise is given a variance of $\sigma^2 = 0.15$. This results in the signal to noise ratios shown in Table 5-1.

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \epsilon, \quad \epsilon \sim \mathcal{N}\left(0, \sigma^2\right) \quad \text{with} \quad \sigma^2 = 0.15 \tag{5-2}$$

s_1	s_2	s_3	s_4
1.18	6.65	3.20	3.71

Table 5-1: Signal to noise ratios (SNR) of the source components.

An example of what a mixture of the original source signals looks like is shown in Figure 5-3. When generating the test-data multiple-times for multiple tests the mixing matrix $\mathbf{A}$ and the GWN ϵ are sampled using a different seed every time. More precisely, for N_{test} different tests the seeds are taken as the sequence: $0, 1, \ldots, N_{test} - 1$.

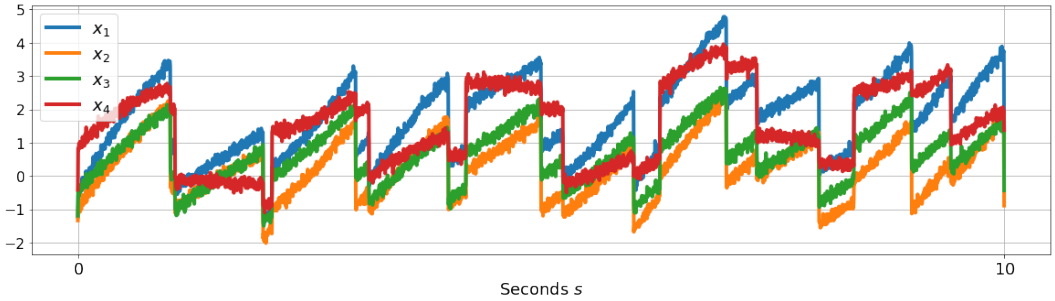


Figure 5-3: Four mixtures of the source signals of the artificial dataset.

5-1-1 Tested algorithms

The algorithms that are tested can be divided into two categories: non-iterative algorithms and iterative algorithms. While the implicit HOSVD uses an iteration internally to decompose the cumulant tensor, it does so in a fixed amount of iterations. On the other hand the main procedures of the iterative algorithms can be repeated many times in order to improve the solution.

The used implicit non-iterative algorithms are compared with their explicit counterparts. They are listed down below together with their used abbreviations and sections.

Abbreviation	Algorithm	Variants	Section
HOSVD	Symmetric HOSVD		4-2
GEVD	CP-GEVD		4-4-4
I-HOSVD	Implicit HOSVD	*full*, U and C	4-2-1
I-GEVD		*full*, U and C	4-4-4

Table 5-2: Tested non-iterative algorithms with their variants and their abbreviations.

The non-iterative algorithms will also be used to initialize the iterative algorithms as is suggested in literature. The presented new implicit algorithms from this thesis are compared with each other and with the algebraic COM2 method and the optimization based FastICA procedure.

Abbreviation	Algorithm	Section
FICA	Parallel kurtosis based FastICA	3-3-2
QRT	Implicit QRT for whitened cumulant tensor	4-3
I-CPD-G	Implicit CPD first-order gradient approach	4-4-5
I-CPD-FF	Implicit CPD fixed-point iteration approach	4-4-5

Table 5-3: Tested iterative algorithms and their abbreviations.

Convergence error The convergence error for the iterative algorithms is shown below in equation 5-3 and is taken from [17]. The error is basically a measure of the orthgonality between two iterations of the unmixing matrix $\mathbf{Q}$.

$$\varepsilon = \max\left(\| \operatorname{diag}\left(\mathbf{Q}_k \mathbf{Q}_{k-1}^T\right) | - 1|\right) \tag{5-3}$$

5-1-2 Performance measures

In order to automate the process of determining whether a solution is correct or not, algorithm 10 in appendix D-1 was developed. The ICA Solution Sorting and Evalutation (ICA-SSE) algorithm first sorts the found solutions based on the order of maximum correlation they have to the source components and determines afterwards whether the estimated components match up with source components and sorts them again by looking for the minimal relative

errors between the source components and the estimated components (lines 4 to 13). By first sorting on maximum correlation and afterwards on minimum error the probability of finding the correct matching pairs is increased. After sorting, the errors and correlations of the found components are compared with the tolerances $\mathbf{e}^{tol}$ and $\mathbf{c}^{tol}$ to determine whether these components are defined as correct.

Estimation error The relative estimation errors of the estimated components from the ICA-SSE algorithm in 10 are combined to form the total relative estimation error of the run:

$$\varepsilon_{total,k} = \frac{\|\mathbf{e}_k\|_2}{\|\mathbf{S}\|_2}, \tag{5-4}$$

where $\mathbf{e}_k \in \mathbb{N}$ denotes the source component estimation errors of the k'th run.

Measure of diagonality The measure of diagonality introduced in chapter 2 definition 2.6 is used to represent the transformed cumulant tensor's diagonality:

$$\tau_{D,k} = \frac{\|\operatorname{diag}(\mathcal{X})\|_2}{\|\mathcal{X}\|_2}. \tag{5-5}$$

For the *GEVD* and *I-GEVD* methods this means that the cumulant tensor has been transformed using the first factor matrix. The measure of diagonality was shown in section 4-4-2 to be a good indicator of the quality of the solution as it represents the statistical independence of the components. Conversely, the CP-error is not used as a performance metric in this experiment. Simply due to the fact that a good solution is defined as having a cumulant tensor which is as diagonal as possible. This relaxed condition implies that the off-diagonal values can be greater than zero as long as they are relatively much smaller than the values on the superdiagonal. The CP-error however punishes these off-diagonal values more and is hence not completely in line with the approximate diagonal philosophy of ICA. Results of the CP-error can be found in appendix D-2.

Computational time For each algorithm during each run the time needed to compute a solution is measured. This means that for all algorithms the timer starts when each algorithm starts working on a solution and stops as soon as the unmixed signals have been produced. For the explicit algorithms the time for computing the cumulant tensor is added. To make the comparison between the explicit and implicit algorithms as fair as possible the cumulant tensor is computed slice by slice using the earlier mentioned method from section 4-2-1. There are several different ways in which the cumulant tensor can be computed with a numerical machine. Depending on the coding language, code structure and computational method the computational performance can vary by several orders of magnitude. In Python, the fastest way of doing so is computing the entire tensor with several C-function calls and no loops as C-functions and their internal loops are in general many orders faster than native *Python* looping. On top of that, nested looping in *Python* can substantially degrade computational speed. As the implicit HOSVD algorithm is written with nested loops in *Python* and not in C the cumulant tensor is computed in the same way for this example.

5-2 Solving the BSS problem

In this section experimental test results of the algorithms are presented. First an example is shown which reflects the general behaviour of data-sets generated from the test set when whitened. The example set is sampled with $I = 1e4$ observations and consists of $P = 4$ observations which equals the amount of underlying source components. Afterwards, results of the non-iterative and iterative algorithms are shown which are performed for varying $P = 4, 5, \ldots, 10$ where for each value of P the test is repeated $N_{test} = 100$ times. The amount of samples I is kept constant at $I = 1e4$. The focus of the entire experiment is to see how the algorithms perform for varying P which is the reason for the creation of the implicit algorithms to begin with. Moreover, the tests are conducted with additive noise present as this is a better reflection of reality. It is important to note that many of the results are dependent on the output of the sorting algorithm 10. Therefore relative performances are considered as important which means the analysis done is qualitative.

5-2-1 Whitening example

Starting conditions The heat-maps of the mixtures from Figure 5-3 are shown in Figure 5-4 and the corresponding performance measures are shown in Table 5-4. Both show that the starting mixtures do not have a diagonal cumulant tensor and as expected do not match the source components which results in a high total estimation error ε_{total} and low diagonality τ_D.

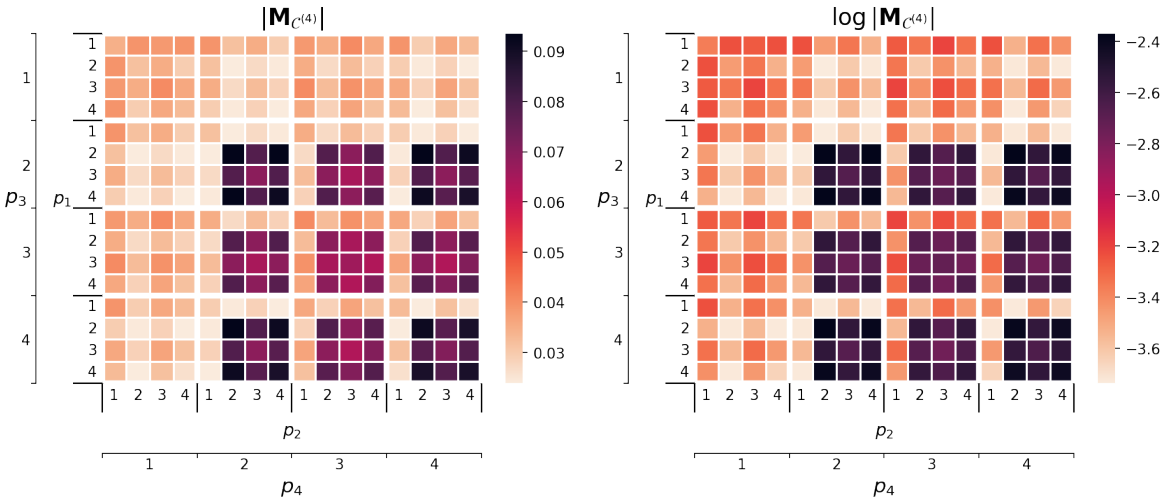


Figure 5-4: Heat-map of the cumulant tensor of the mixtures $\mathbf{x}$. On the left the absolute values are on a linear scale and on the right on a logarithmic scale.

ε_{total}	τ_D	ε_{CP}
0.896	0.192	-

Table 5-4: Performance measures of the mixtures $\mathbf{x}$.

Whitening Figure 5-5 shows the whitened mixtures from Figure 5-3. Although Whitening has not solved the problem, the result can be considered as somewhat recognizable when compared to the true source components. Out of 100 sampled test data-sets for each $P = 4, \ldots, 10$ not once did the whitening procedure produce a result which was evaluated as correct.

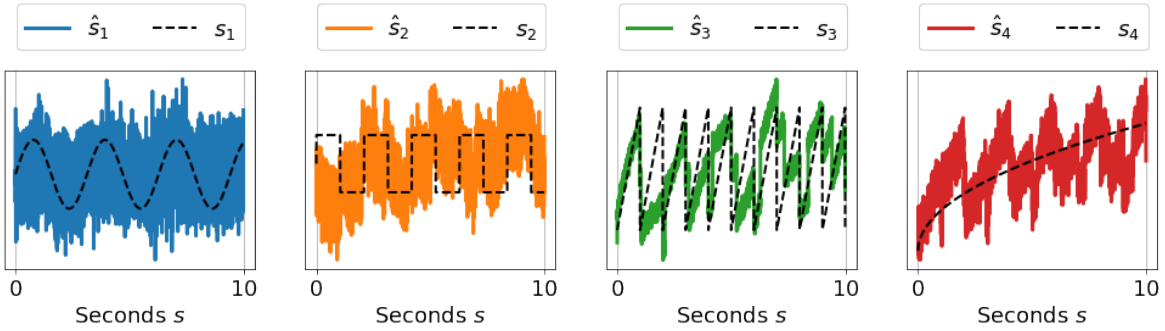


Figure 5-5: Example of a Failed set of estimated components after only pre-whitening.

The heat-maps shown in Figure 5-6 and measure of diagonality from Table 5-5 show that whitening has structured the cumulant tensor towards a more diagonal form. On top of that, the higher kurtosis values are now near at the lower tensor index values $p_i = 1, 2 \quad \forall i = 1, 2, 3, 4$. The estimation error has increased but not by any significant amount. From this point on all used data-sets are whitened when used with any algorithm.

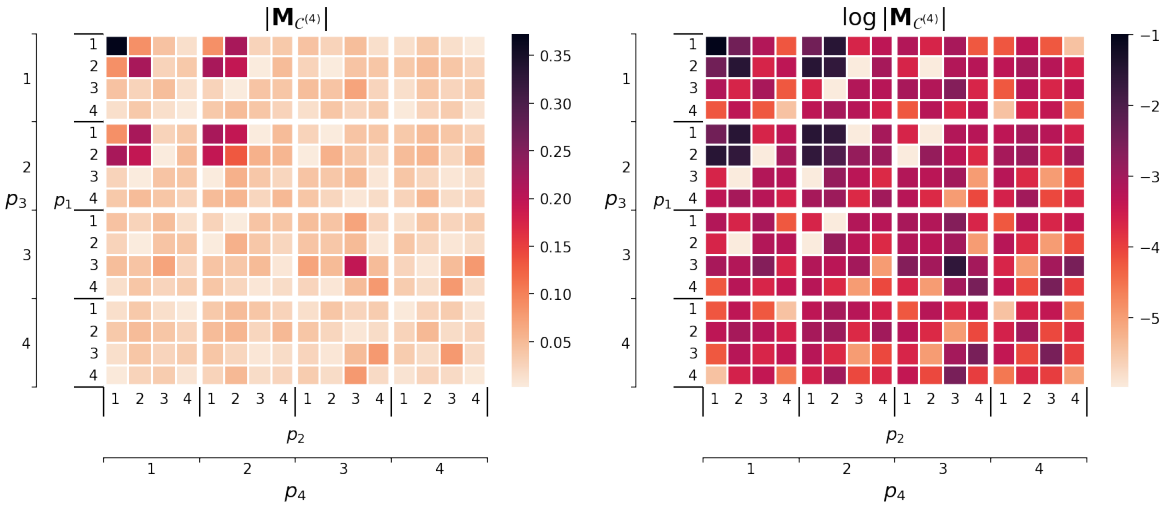


Figure 5-6: Heat-map of the cumulant tensor of the whitened mixtures $\mathbf{z}$. On the left the absolute values are on a linear scale and on the right on a logarithmic scale.

ε_{total}	τ_D	ε_{CP}
0.910	0.448	-

Table 5-5: Performance measures for only whitening of the data.

5-2-2 Non-iterative algorithms

First the performance of the non-iterative algorithms is compared. This is done with the previously presented test-setup. The implicit non-iterative algorithms have two main purposes: to be able to produce (nearly) identical or better results than their explicit counterparts and to do so at lower computational costs expressed in computation time and storage cost which was proven theoretically.

Successful estimations Figure 5-7 shows the percentage of runs per non-iterative algorithm that are classified as successful. Noteworthy is the fact that *GEVD* outperforms all other algorithms. Compared with HOSVD this can be attributed to the fact that *GEVD* performs the additional Generalized EigenValue Decomposition (GEVD) step after HOSVD. For the *I-HOSVD* algorithms it is observed that the U and C versions perform better than the full version at higher values of P with U the best uot of the two. The same cannot be said about the *I-GEVD* algorithms. Here the full algorithm performs arguable overall the best with its U version as a close second. The C however, does not manage to produce a single successful result. The vast differences that are present in the *I-GEVD* algorithms can be attributed to the fact that the amount of information on the first 2 core slices of the tensor is different when using simpler *I-HOSVD* algorithms.

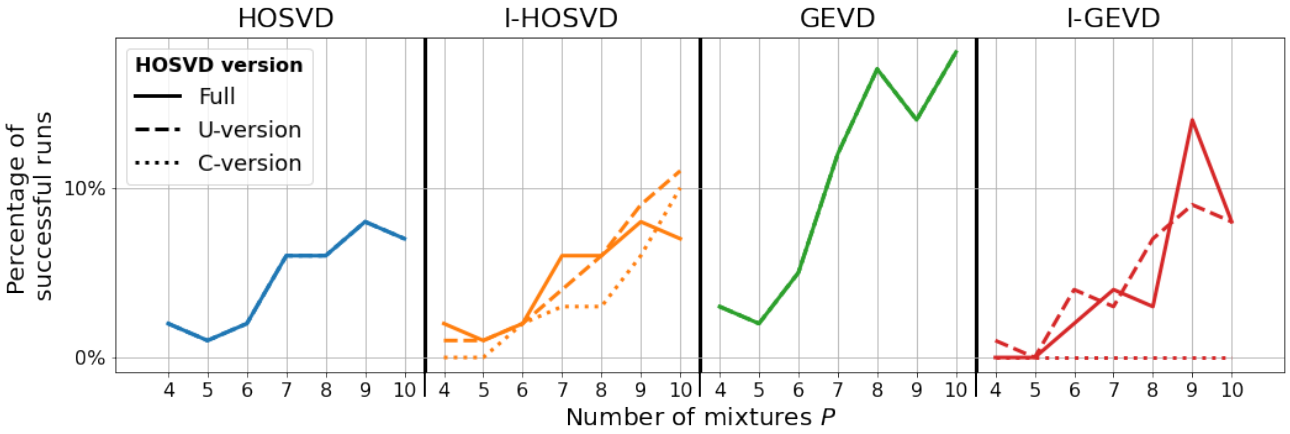


Figure 5-7: Average percentage of estimations for varying amounts of mixtures $P = 4, \ldots, 10$ which are classified as correct for a total of $N_{test} = 100$ runs.

Total Estimation error In Figure 5-8 the average relative estimation errors ε_{total} of the non-iterative algorithms are presented for all runs (solid) and only the successful runs (dashed) together with their variances (shaded areas). Clearly for all methods the average errors of the successful runs are lower than the average of all runs. The variances of the HOSVD based algorithms for all runs narrows as P increases which implies that using more mixtures results in general in better estimates. The total errors of the *C-I-HOSVD* algorithm differ the most from the others. Its average error, considering successful and failed runs together, is the lowest of all 4 HOSVD methods.

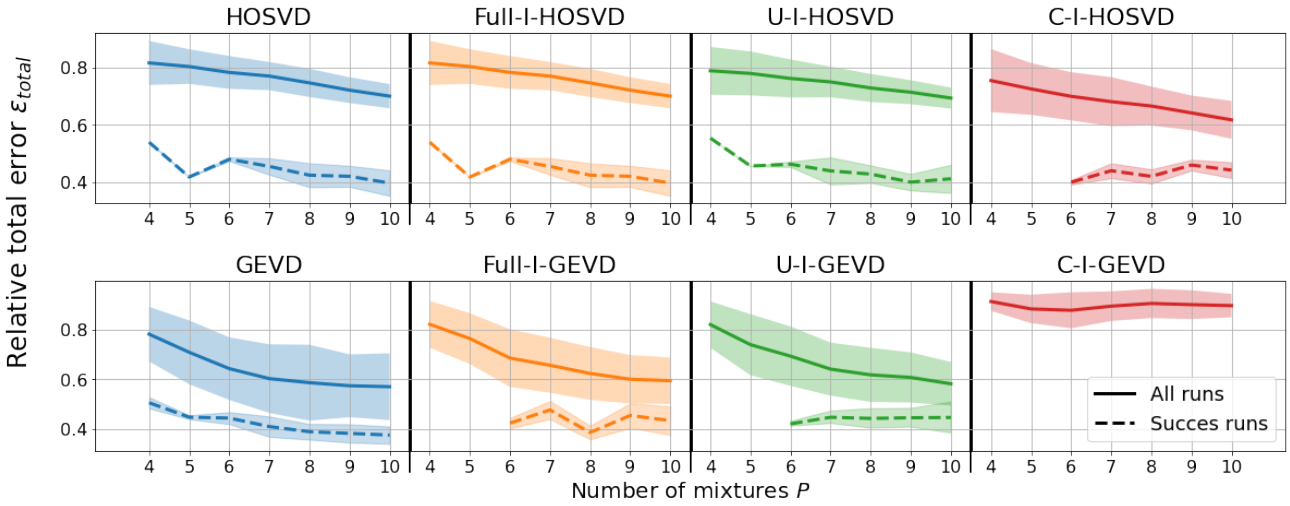


Figure 5-8: Average total estimation errors ε_{total} (solid and dashed lines) together with their standard deviations (shaded areas) for varying amounts of mixtures $P = 4, \ldots, 10$ for a total of $N_{test} = 100$ runs. The solid lines represent only the successful runs and the dashed lines represent all 100 runs. Notice how for example the successful run (dashed line) of *C-I-HOSVD* only starts at $P = 6$ showing that no successful solution was found at $P = 4$ and $P = 5$.

The average succesful estimation errors of the GEVD methods are lower than the HOSVD algorithms except for *C-I-GEVD*. All 3 versions of the *I-GEVD* algorithm failed for $P = 4$ and $P = 5$ while *GEVD* does not. This implies that its smaller and partial inwards projection step is the cause. The *C-I-GEVD* algorithm fails to produce any successful estimation, indicating that applying HOSVD on only the core slices together with the partial inwards projection step does not work.

Overall, the errors show that the GEVD methods produce on average results with lower errors but with higher variabilities compared to the HOSVD methods. However, for successful estimation the results are on average better with the HOSVD methods, if ever so slightly.

Diagonality Figure 5-9 shows the diagonality of the different algorithmss. The diagonality of the cumulant tensor of normalized and zero-mean $\mathbf{s}$ is shown too as a dotted line at $\tau_D = 0.855$. The figure shows that the HOSVD algorithms do a much better job at imposing diagonality on the entire cumulant tensor than the GEVD algorithms do. The explanation for this is simple: the GEVD algorithms do not compute a symmetric decomposition. The implicit versions of the GEVD algorithm perform worse than their explicit counterpart. Again, this can be attributed to the partial inwards projection as the *Full-C-GEVD* uses the full HOSVD too.

It is interesting to see that the *U-I-HOSVD* method performs on average very similar to the *HOSVD* and *Full-I-HOSVD* methods while the performance deteriorates a lot more when only the core slices are used with *C-I-HOSVD*. As in real life there i in many cases no definite way of knowing when a solution is a success or not the general performance of the algorithm is what you want to depend on. Seeing that the *U-I-HOSVD* performs on average similar on the aspects of estimation error and diagonality as regular HOSVD suggests it could be considered as a computationally more efficient alternative.

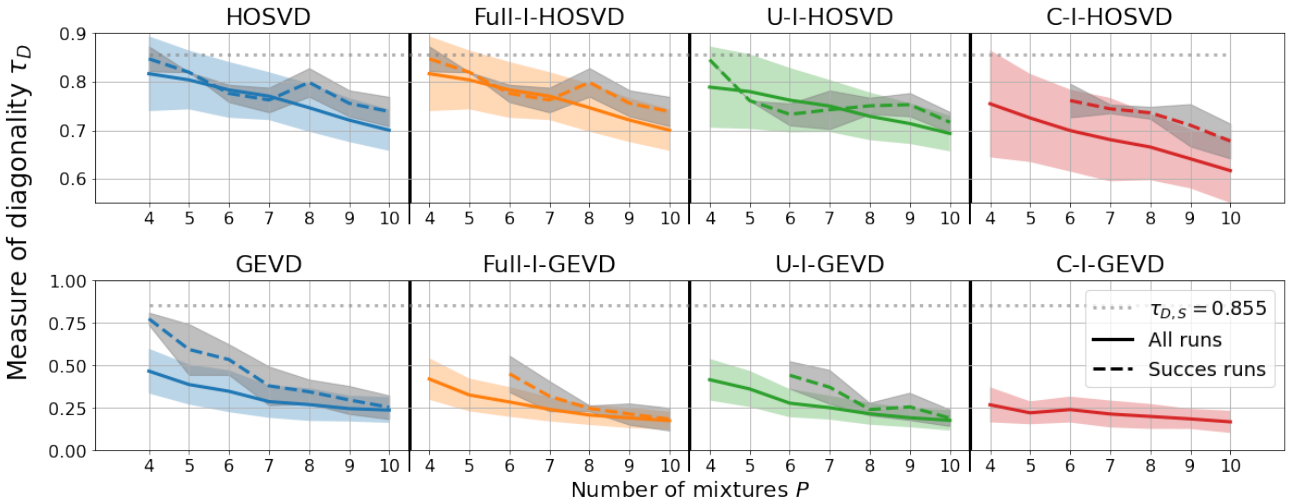


Figure 5-9: Average measure of diagonality τ_D (solid and dashed lines) together with their variances (color shaded for all runs and gray shaded for successful runs) of the entire cumulant tensor for varying amounts of mixtures $P = 4, \ldots, 10$ for a total of $N_{test} = 100$ runs. The solid lines represent only the successful runs and the dashed lines represent all 100 runs. The measure of diagonality of the normalized and zero-mean source components $\tau_{D,\mathbf{S}} = 0.855$ is shown too as a grey loosely dashed line.

The measure of diagonality shown here is for the entire transformed cumulant tensor. However, as P increases there are more noise components present which resemble each other statistically speaking. Due to finite smapling these are approximate GWN components which will show high cross-kurtosis values and low self-kurtosis values or in other words, low diagonal values and relatively high off-diaognal values. Therefore it is of interest to consider the measure of diagonality of the smaller sub-cumulant tensor created from only the estimated source components $\hat{s}_i \quad \forall i = 1, 2, 3, 4$ which is shown in figure Figure 5-10.

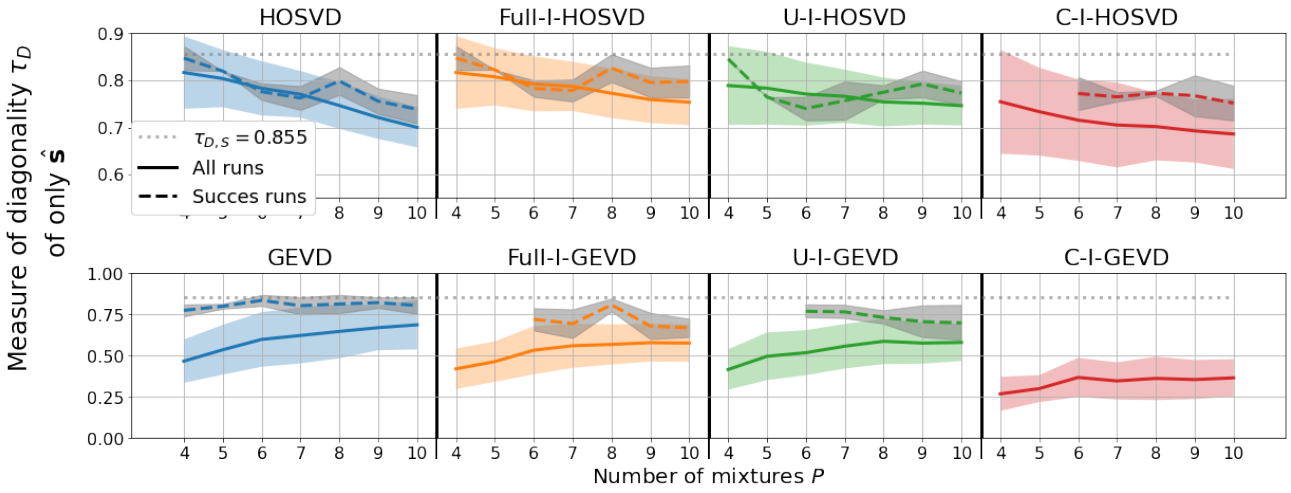


Figure 5-10: Average measure of diagonality τ_D (solid and dashed lines) together with their variances (color shaded for all runs and gray shaded for successful runs) of thesub-cumulant tensor of the estimated components $\hat{\mathbf{s}}$ for varying amounts of mixtures $P = 4, \ldots, 10$ for a total of $N_{test} = 100$ runs. The solid lines represent only the successful runs and the dashed lines represent all 100 runs. The measure of diagonality of the normalized and zero-mean source components $\tau_{D,\mathbf{S}} = 0.855$ is shown too as a grey loosely dashed line.

In comparison to Figure 5-9, the figure shows that for the HOSVD methods the diagonality of the cumulant tensor changes little when considering only parts of it suggesting an even distribution of the diagonality among components. The diagonality of the GEVD methods is this time a lot higher. Especially for the successful cases which means that they manage to diagonalize the parts of the cumulant tensor related to the source components. This makes sense as the GEVD step is used to separate the strongest components present in the chosen tensor slices which in this case are the source components. Furthermore, as was mentioned before the residual noise components after estimation will have low ratios of absolute diagonal values to absolute off-diagonal values.

What can be stated when combining all of these results together is that for the non-iterative algorithms the measure of diagonality of the estimated source components is a clear indicator of the quality of a solution. This makes sense as the ICA solution is defined to have a cumulant tensor for the estimated source signals which is as diagonal as possible.

Runtime and computational complexities The average running times are shown in Figure 5-11 for varying P. Starting with the explicit algorithms *HOSVD* and *GEVD*, it is clear that the cost of these methods is dominated by the cumulant tensor forming cost which is visible through the gray x-shaped markers. The *GEVD* algorithm has a slight increase compared to the *HOSVD* algorithm for higher values of P which is due to the additional inwards projection step. This holds true too for each version pair of the *I-HOSVD* and *I-GEVD* algorithms.

The computational time of the implicit algorithms varies over which version is used. Both the full and U versions take more time to compute than the original explicit algorithms. The C version is much faster, which is due to less slices that needed to be computed.

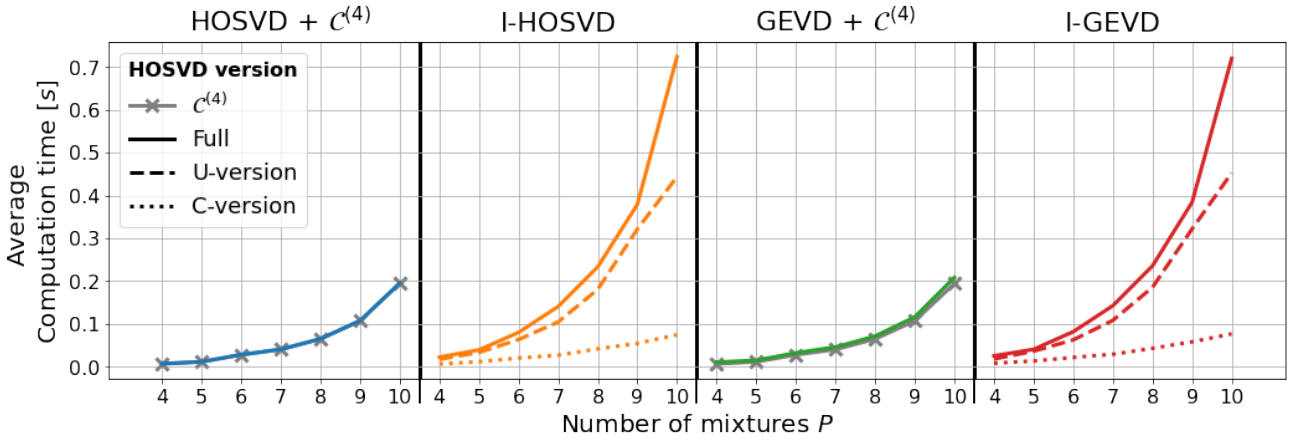


Figure 5-11: Average computation time for varying amounts of mixtures $P = 4, \ldots, 10$ for a total of $N_{test} = 100$ runs for varying values of $I = \{2, 5e3, 5e3, 10e3\}$.

The difference in scaling of the implicit algorithms can be related to their complexities listed in Table 5-6. The table in combination with the figure shows that the additional terms due to the svd-update steps (blue) and the inwards projection step for the *I-GEVD* (red) have a significant impact on the total computation cost as P goes up.

	C4	HOSVD	I-HOSVD	GEVD	I-GEVD
Full	IP^4	IP^4	$\frac{1}{2}IP^4 + P^5(\log(P)^2)$	IP^4	$\frac{1}{2}IP^4 + P^5(\log(P)^2) + RP^4$
U-method	-	-	$\frac{1}{2}IP^4 + \frac{P^5}{2}(\log(P)^2$	-	$\frac{1}{2}IP^4 + \frac{P^5}{2}(\log(P)^2 + RP^4$
C-method	-	-	$IP^3 + P^4(\log(P)^2$	-	$IP^3 + P^4(\log(P)^2 + RP^4$

Table 5-6: Dominant computational complexities of the cumulant tensor forming, the non-iterative explicit algorithms and their implicit counterparts. The terms in blue are due to the svd-update procedure and the term in red is due to the inwards projection step.

These results show that the implementing the implicit scheme for both HOSVD and GEVD comes at the cost of an increase in computational effort. Essentially a trade-off is made between storage complexity and computaitonal complexity.

5-2-3 Iterative algorithms

All iterative algorithms are set to have a convergence tolerance of 10^{-4} and a maximum amount of iterations of $K_{max} = 300$. Furthermore, the gradient CPD algorithm is solved using the Limited-memory-Broyden–Fletcher–Goldfarb–Shanno-Bound (L-BFGS-B) algorithm implemented in the Python scipy package with the custom convergence stopping criterion from (5-3). This is the same solver used by the authors of the paper from which the implicit first-order gradient problem for statistical tensors originates from [53]. For the gradient methods the total amount of iterations is taken as the number of function evaluations times the number iterations. In general, only the successful estimations are considered in the analysis of the iterative algorithms performance. It is stated if otherwise.

Algorithm initialization All iterative algorithms are tested with different initial estimates to see whether and how they impact performance. What initialization means differs per algorithm. For the *QRT* algorithm it means that before feeding the data to the algorithm it is transformed using an initial estimate of the mixing matrix $\mathbf{Q}_0$:

$$\mathbf{Z}' = \mathbf{Q}_0^T \mathbf{Z}. \tag{5-6}$$

The final estimate consists of the QRT solution multiplied with the initial estimate $\mathbf{Q}_{final} = \mathbf{Q}_0 \mathbf{Q}_{QRT}$. For the other algorithms the initial estimate $\mathbf{Q}_0$ is used in the computation of the gradient. For the *I-CPD-G* and *I-CPD-FF* algorithms the initial estimate of $\boldsymbol{\lambda}_0$ is used too.

The used initialization schemes with the following initial estimates are listed below.

Name	Mixing matrix $\mathbf{Q}_0$	Kurtosis coefficients $\boldsymbol{\lambda}$
None	$\mathbf{I}_P$	$\begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T \in \mathbb{R}^P$
HOSVD	$\mathbf{U}_1 \leftarrow$ HOSVD	$\begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T \in \mathbb{R}^P$
U-I-HOSVD	$\mathbf{U}_1 \leftarrow$ U-Implicit HOSVD	$\begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T \in \mathbb{R}^P$
C-I-HOSVD	$\mathbf{U}_1 \leftarrow$ C-Implicit HOSVD	$\begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T \in \mathbb{R}^P$
GEVD	$\mathbf{U}_1 \leftarrow$ CP-GEVD	$\boldsymbol{\lambda} \leftarrow$ CP-GEVD
I-GEVD	$\mathbf{U}_1 \leftarrow$ Full Implicit CP-GEVD	$\boldsymbol{\lambda} \leftarrow$ Full Implicit CP-GEVD
U-I-GEVD	$\mathbf{U}_1 \leftarrow$ U-Implicit CP-GEVD	$\boldsymbol{\lambda} \leftarrow$ U-Implicit CP-GEVD
C-I-GEVD	$\mathbf{U}_1 \leftarrow$ C-Implicit CP-GEVD	$\boldsymbol{\lambda} \leftarrow$ C-Implicit CP-GEVD

Table 5-7: Used initialization schemes for the numerical performance comparison of the iterative algorithms.

Successful estimations Figure 5-12 shows the percentage of successful runs of all 4 iterative algorithms with the varying initialisation methods. *FICA*, *QRT* and *I-CPD-FF* all perform very similar for no initialization method and for HOSVD. It would be no surprise if the slight differences would vanish as the total amount of experiment runs N_{total} is increased. However, due to time constraints this was not verified. With the HOSVD method *I-CPD-G* performs equally well as the others. However, with no initializing method and with GEVD it performs worse, especially with the latter. This clearly shows how the gradient method's performance greatly depends on the initial estimate used. This is in line with the general performance of the gradient version of fastICA [33][21].

Looking at *QRT* shows that GEVD initialization resulted in nearly 0 successful runs. There is a simple yet logical explanation for this. The first factor matrix produced by the *GEVD* is not orthogonal. Due to the multiplication of the whitened data with the initial non-orthogonal estimate $\mathbf{Q}_0$ the data loses its whiteness. The implicit *QRT* algorithm was designed to work only with whitened data, hence there are no successful runs with these initialization schemes. This implies that the *QRT* algorithm will perform poorly with the other GEVD methods too.

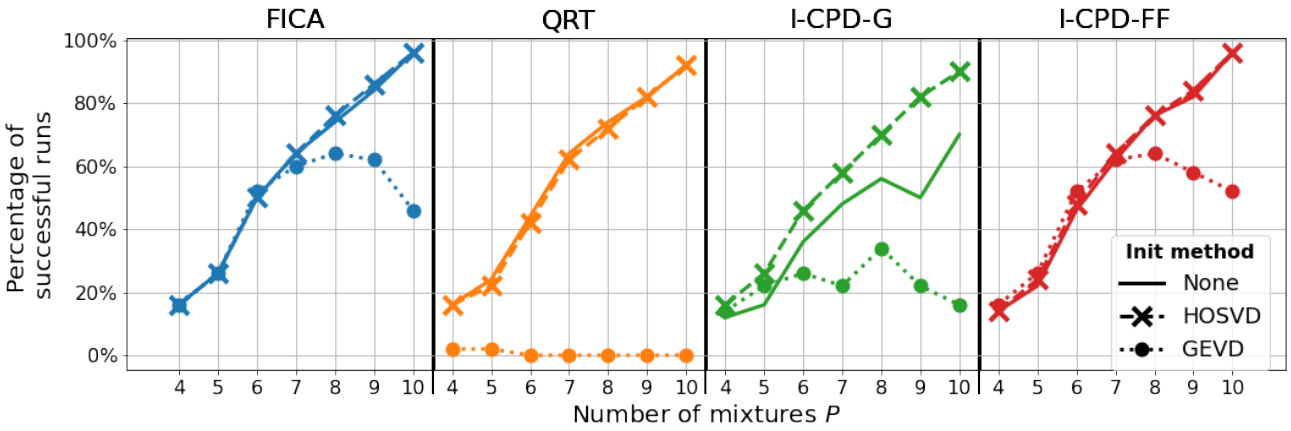


Figure 5-12: Average percentage of estimations for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs. For each algorithm the results are plotted with no initialization used and when standard HOSVD and *GEVD* are used.

The other algorithms perform worse with the GEVD initialization too. For *FICA* it starts of identical to the other methods. However, after $P = 6$ the percentage declines. The same is true for *I-CPD-FF* with the differences being that the GEVD method performs slightly better at lower values of $P \leq 6$ and that the final decline is less.

Considering the figure but then for all HOSVD based methods only, shown in Figure 5-13, several key differences can be observed. First of all, the performance of *FICA* and *QRT* does not differ between methods. Secondly, *I-CPD-G* performs better with all HOSVD methods than without any initialization. Again, stressing the importance of a good initial estimate for gradient ICA algorithms. Lastly, the performance of *I-CPD-FF* with *U-I-HOSVD* has improved over several values of P suggesting its effectiveness as an initial estimate of quality for the *I-CPD-FF* algorithm.

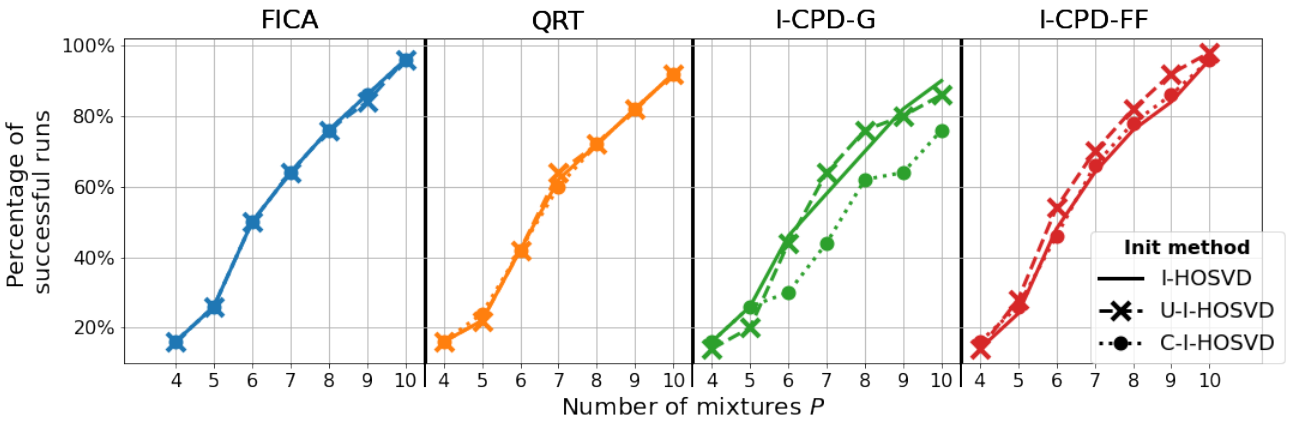


Figure 5-13: Average percentage of estimations for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs. For each algorithm the results are plotted with all of the possible HOSVD based initializations.

Moving on to the GEVD based initialization methods shown in Figure 5-14 it can be observed that as was previously suggested the *QRT* algorithm fails for every method. The performance of *FICA* and *I-CPD-FF* with each implicit version is better than with the original GEVD algorithm with again the *U* version having the highest scores over several values of P for the *I-CPD-F* algorithm. This suggests that the partial inwards projection is preferred over the entire inwards projection for these methods for this data-set at higher-values of P. The gradient approach *I-CPD-G* performs again worse with the GEVD based methods as was suggested by Figure 5-12.

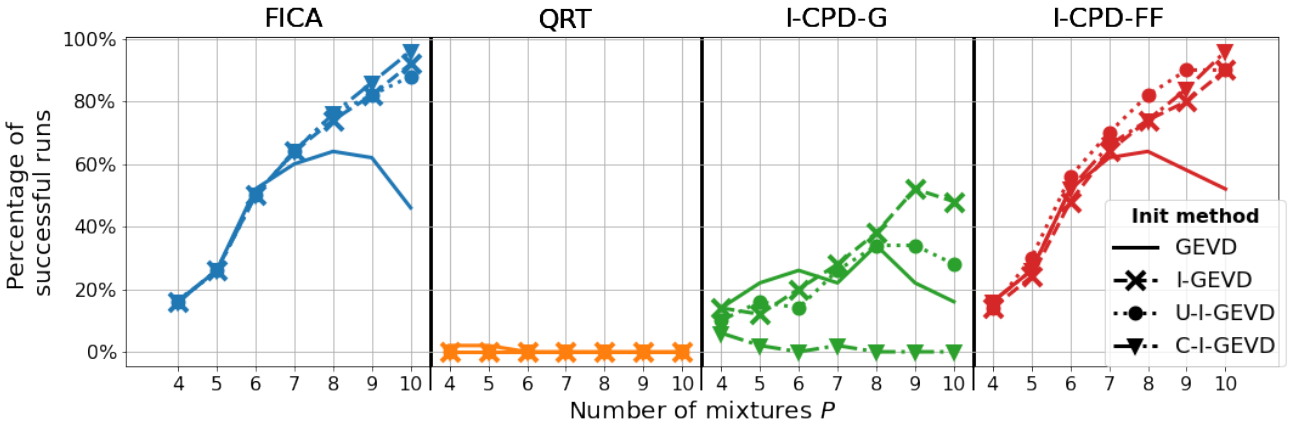


Figure 5-14: Average percentage of estimations for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs. For each algorithm the results are plotted with all of the possible HOSVD based initializations.

The general take-away of these results is that the gradient and fixed-point iteration algorithms show a lot of sensitivity to the initial estimate. For *QRT* no such statement can be made from only these results. The GEVD based initialization methods do not count for *QRT* as it was not designed to work with non-orthogonal matrices.

Total Estimation error The total estimation error of only successful runs of the full methods is shown in Figure 5-15. Taking into account the percentages of successes, the figure shows that the *I-CPD-G* algorithm manages to achieve the lowest possible estimation error. Gradient based approaches in general can achieve better results than fixed-point methods when correctly initialized. Next to the *I-CPD-G* method the *QRT* and *I-CPD-FF* seem to prefer the initialization of HOSVD too over the other methods. Compared over all methods the GEVD initialization performs the worst. *FICA* and *I-CPD-FF* have again similar performances not considering GEVD.

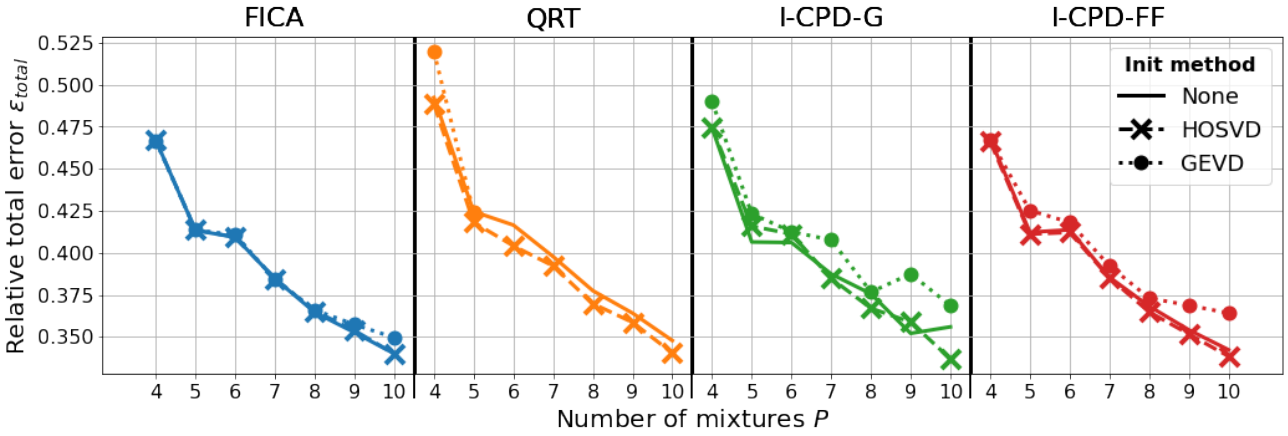


Figure 5-15: Average total error ε_{total} for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs. For each algorithm the results are plotted with no initialization used and when HOSVD and *GEVD* is used.

The estimation errors with the HOSVD type initializing algorithms is shown in Figure 5-16. Two things can be stated when observing these results. Firstly, *FICA*, *I-CPD-FF* and *I-CPD-G* perform nearly identical for all types of the implicit HOSVD algorithm. Secondly, *QRT*'s average estimation error has decreased with the U type.

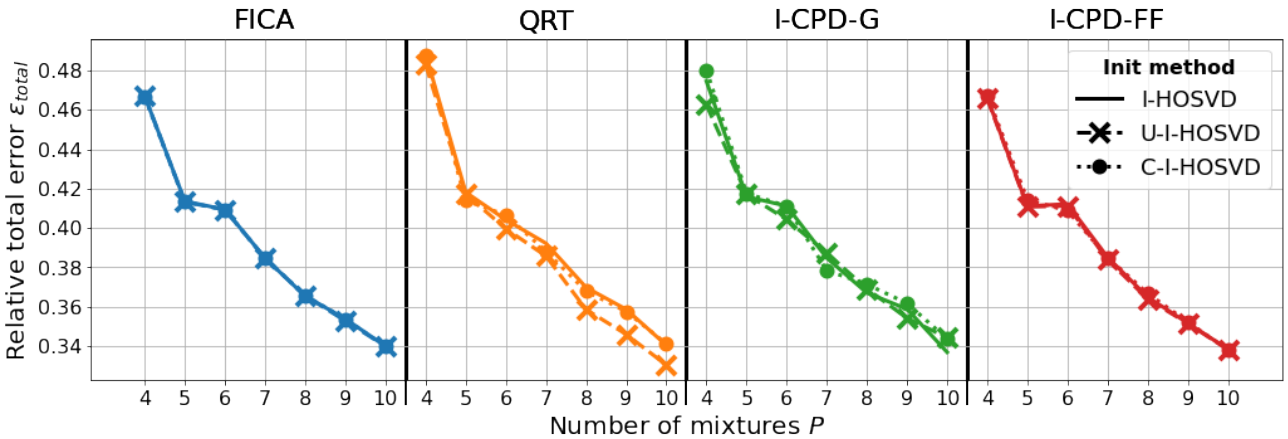


Figure 5-16: Average total error ε_{total} for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs. For each algorithm the results are plotted with all different HOSVD type initializations used.

Lastly, the results of the algorithms with GEVD initialization is shown in Figure 5-17. Compared to the HOSVD based initializations these methods produce overall slightly worse results. Hinting at the iterative-algorithms' preference for HOSVD type initializations.

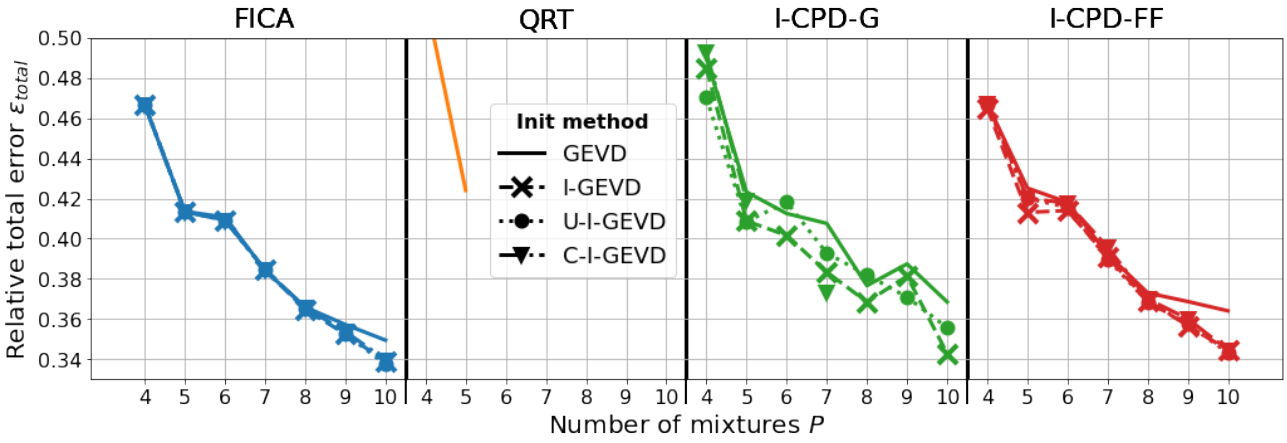


Figure 5-17: Average total error ε_{total} for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs. For each algorithm the results are plotted with all different GEVD type initializations used.

What all methods have in common is that the estimation error decreases as P increases. This is simply due to the fact that more mixtures allow for more possible seperation of signal and noise. An example of the *QRT* method with *U-I-HOSVD* initialization is shown below in Figure 5-18 for $P = 10$. The results clearly show that the signals have been properly unmixed and that noise is adequately removed from the estimated components.

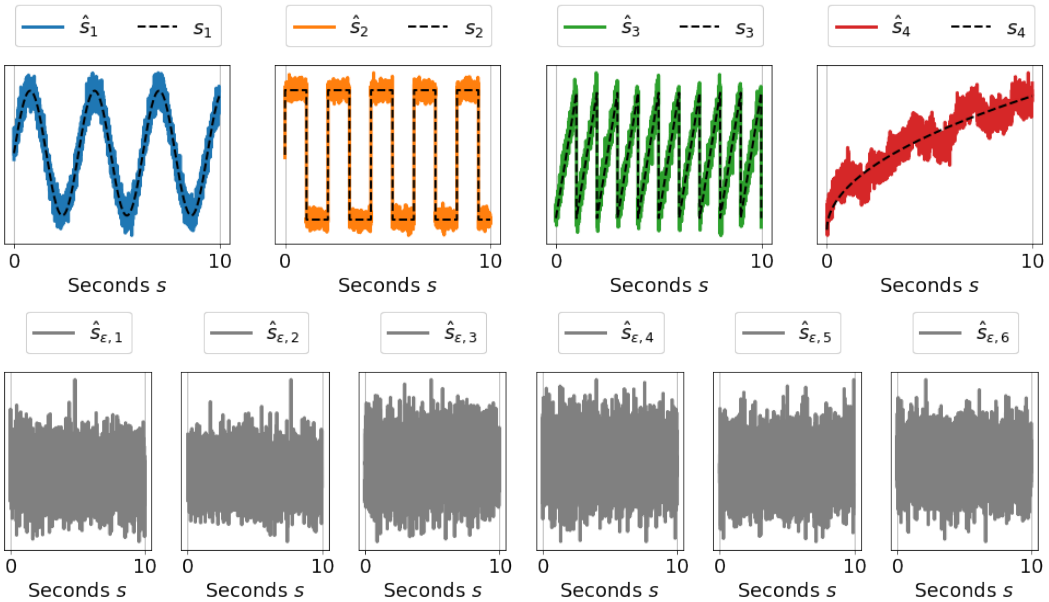


Figure 5-18: Estimated source components $\hat{s}_i \quad \forall i = 1, 2, 3, 4$ together with the residual noise components $\hat{s}_{\varepsilon,i} \quad \forall i = 1, 2, 3, 4, 5, 6$ of a successful run of the *QRT* algorithm with *U-I-HOSVD* initialization for $P = 10$ mixtures.

Diagonality Figure 5-19 shows the diagonality of the no initialization, HOSVD and the GEVD initialization scheme. The results show that all algorithms are able to make the cumulant tensor more diagonal than the supposed solution. The GEVD scheme performs overall the worst with this measure too and HOSVD could arguably be considered as the overall best. This corresponds with the previous observations concerning the estimation errors.

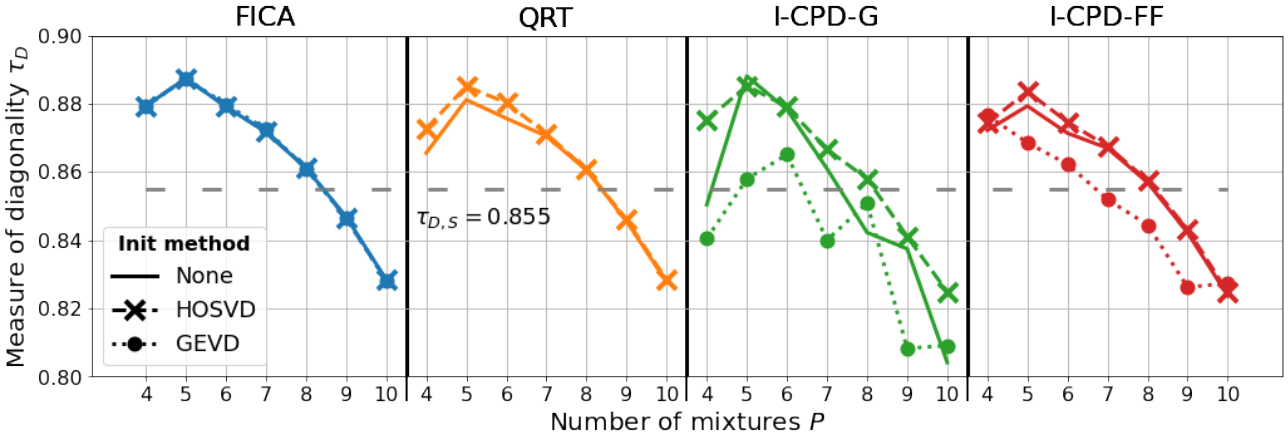


Figure 5-19: Average measure of diagonality τ_D (solid and dashed lines) of the entire transformed cumulant tensor for varying amounts of mixtures $P = 4, \ldots, 10$ for a total of $N_{test} = 100$ runs. The solid lines represent only the successful runs and the dashed lines represent all 100 runs. The measure of diagonality of the normalized and zero-mean source components $\tau_{D,\mathbf{s}} = 0.855$ is shown too as a grey loosely dashed line.

Continuing to the HOSVD methods in Figure 5-20 shows that for all algorithms the full *I-HOSVD* and U type perform equally well except for a minor dip at $P = 5$ for the gradient approach. Even the C type approach manages to keep up with the others.

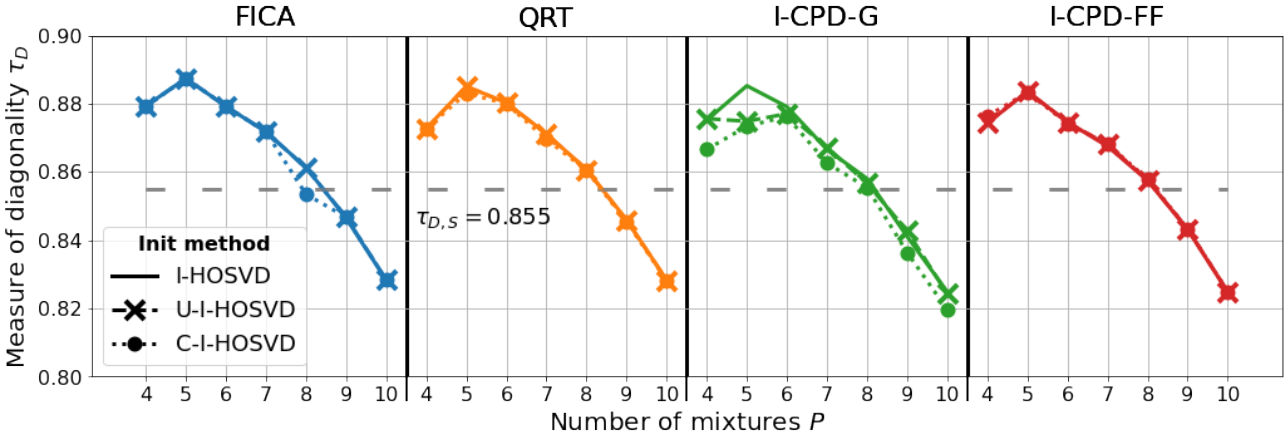


Figure 5-20: Average measure of diagonality τ_D (solid and dashed lines) of the entire transformed cumulant tensor for varying amounts of mixtures $P = 4, \ldots, 10$ for a total of $N_{test} = 100$ runs. The solid lines represent only the successful runs and the dashed lines represent all 100 runs. The measure of diagonality of the normalized and zero-mean source components $\tau_{D,\mathbf{s}} = 0.855$ is shown too as a grey loosely dashed line.

Cross-referencing this with the previous results indicates that at least for the fixed-point iterations and the *QRT* method the choice of the HOSVD type initialization scheme does not worsen the result. However, the same cannot be said for the GEVD based methods. Only *FICA*'s performance has not deteriorated in comparison to the HOSVD based methods and no initialization method.

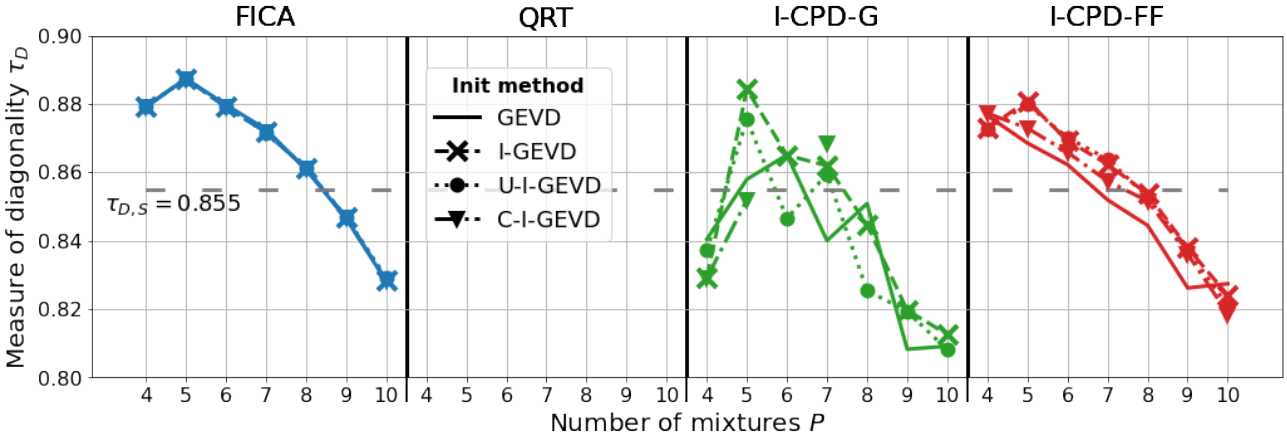


Figure 5-21: Average measure of diagonality τ_D (solid and dashed lines) of the entire transformed cumulant tensor for varying amounts of mixtures $P = 4, \ldots, 10$ for a total of $N_{test} = 100$ runs. The solid lines represent only the successful runs and the dashed lines represent all 100 runs. The measure of diagonality of the normalized and zero-mean source components $\tau_{D,\mathbf{S}} = 0.855$ is shown too as a grey loosely dashed line.

The constant decrease in the diagonality as P goes up initially seems to go against the error that gets lower as P increases. However, this can again be attributed to the additional residual noise components. It was previously mentioned that they have high cross kurtosis values and low self kurtosis values which and cannot be made less gaussian as they are GWN. So considering once again the measure of diagonality of only the estimated source components sub-cumulant tensor shown in Figure 5-22 the diagonality behaviour changes.

This time the results are only shown for no initialization, HOSVD and GEVD based initialization. For *FICA* and*QRT* the diagonality τ_D of the sub tensor remains for $P \geq 5$ at a consistently high level with very little variance amongst the values for P. This is most likely due to there being 5 latent components in the data present. Namely the 4 source signals and the GWN noise component. in the infinite sample case excess kurtosis is blind to GWN. However, this is not the case here as excess kurtosis is approximated using a finite number of samples. Hence the strong presence of the noise as a separate component.

The slightly lower diagonality score of the *I-CPD-FF* algorithm is most likely the result of the algorithms focus on computing a CPD instead of maximizing the diagonal entries such as *FICA* and *QRT* do. However, the difference here is minor and the overall performance of the *I-CPD-FF* algorithm shown up to this point is on par with that of *FICA*

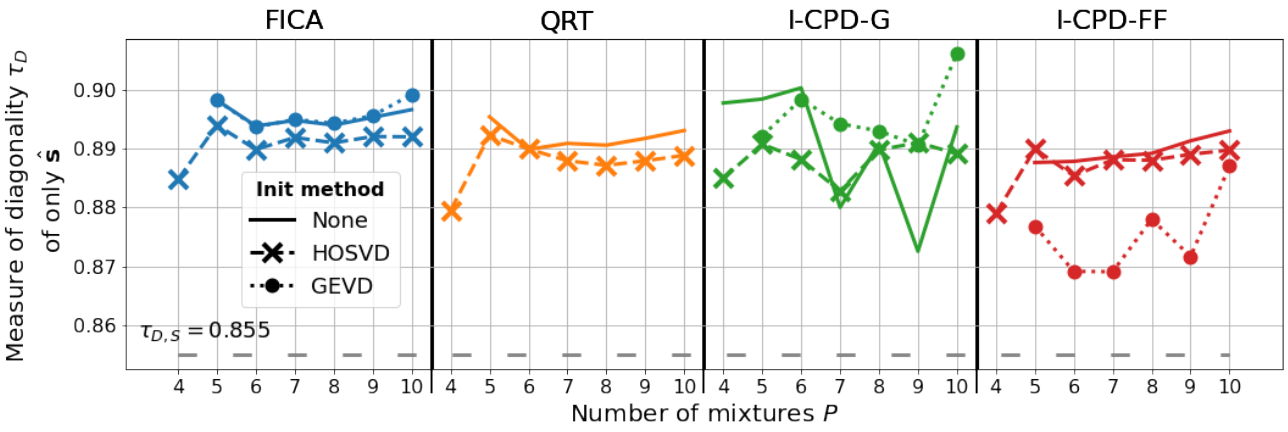


Figure 5-22: Average measure of diagonality τ_D (solid and dashed lines) of the sub cumulant tensor from only the estimated source components $\hat{\mathbf{s}}$ for varying amounts of mixtures $P = 4, \ldots, 10$ for a total of $N_{test} = 100$ runs. The solid lines represent only the successful runs and the dashed lines represent all 100 runs. The measure of diagonality of the normalized and zero-mean source components $\tau_{D,\mathbf{s}} = 0.855$ is shown too as a grey loosely dashed line.

Lastly, the heat-maps of the estimated source-components sub cumulant tensor are shown in Figure 5-18. The linear scale heat-map clearly shows the diagonal structure of the tensor. Moreover, the null-fiber structure the *QRT* algorithm applies on a tensor discussed in section 4-3-1 can be clearly observed on the locarithmicly scaled heat-map.

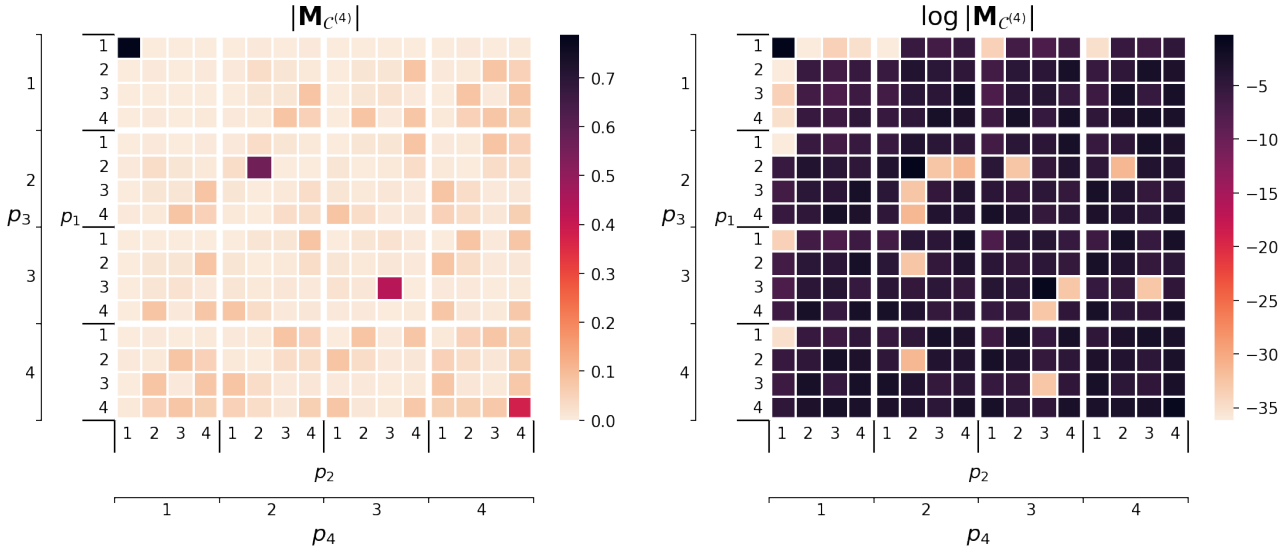


Figure 5-23: Heat-map of the cumulant tensor of the source components $\hat{s}_i \quad i = 1, 2, 3, 4$ estimated with *QRT* and HOSVD initialization for $P = 10$ mixtures. On the left the absolute values are on a linear scale and on the right on a logarithmic scale.

Runtime and computational complexities Arguably some of the most interesting results of the experiment are to be seen whenn looking at needed computation times shown on a linear scale in Figure 5-24 and for alogarithmic scale in Figure 5-25. Before anything else, note that the *I-CPD-G* results have been scaled down by a factor 2 in the time figure and by 2 orders of magnitude in the number of iterations figure Figure 5-26 which is done for clearer visualization of the overall results. Only the results of no initialization, HOSVD and GEVD are shown here as the latter are representable for the average times of all of their alternate versions. First and foremost it must be stated that both *QRT* and *I-CPD-FF* algorithm are a lot faster than *FICA*. The former by nearly an order of magnitude and the latter by a factor of 5. While *I-CPD-FF* starts of slower than *FICA* its average time increases at a much slower pace.

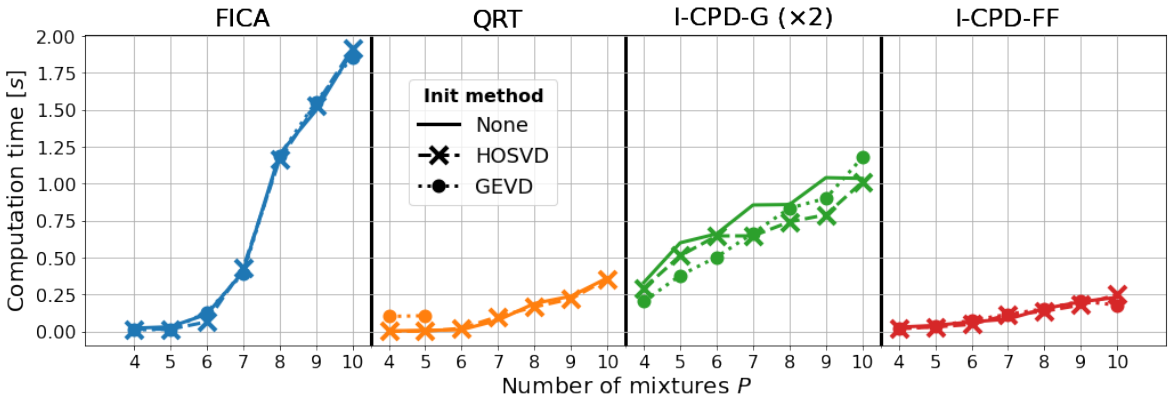


Figure 5-24: Average runtime for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs on a linear scale. For each algorithm the results are plotted with no initialization used and when HOSVD, *GEVD* or no initialization is used. Notice that the *I-CPD-G* algorithm is scaled down for clearer visual presentation. The actual times of the *I-CPD-G* algorithm are a factor 2 as high.

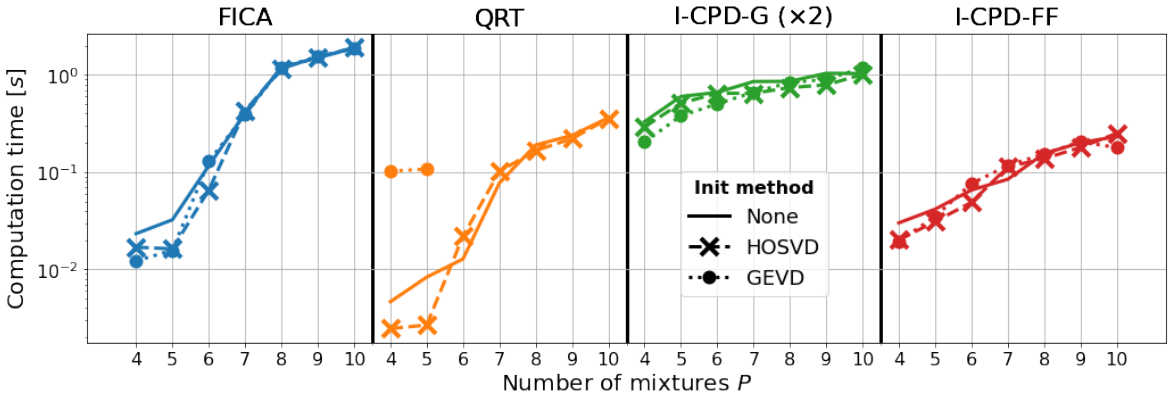


Figure 5-25: Average runtime for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs on a logarithmic scale. For each algorithm the results are plotted with no initialization used and when HOSVD, *GEVD* or no initialization is used. Notice that the *I-CPD-G* algorithm is scaled down for clearer visual presentation. The actual times of the *I-CPD-G* algorithm are a factor 2 as high.

Inspecting the amount of iterations each algorithm performs shown in Figure 5-26 on average explains these differences. The figure shows that while *QRT* and *FICA* have to perform a near equal amount of iterations to reach the same level of convergence, *I-CPD-FF* needs far less. The cost per iteration for *FICA* and*I-CPD-FF* are nearly identical implying that *I-CPD-FF* converges faster towards a solution. This is most likely due to the CPD constraint present in the gradient together with the gradient computation of $\boldsymbol{\lambda}$ which are all a result of the cost function. The cost function of *FICA* is the L_1 norm of the self kurtosis values whereas the cost function of *I-CPD-FF* is the L_2 norm squared over the difference between the entire cumulant tensor and its CPD estimate.

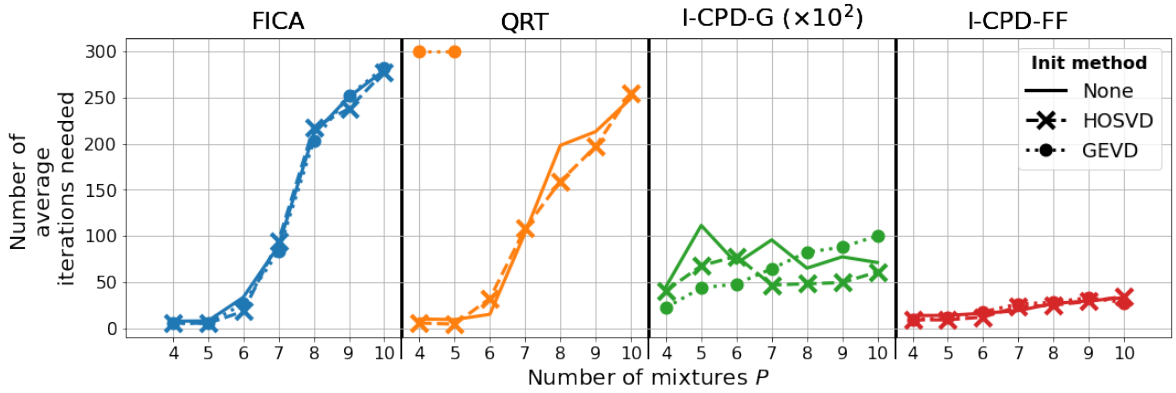


Figure 5-26: Average total amount of iterations needed per algorithm for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs. For each algorithm the results are plotted with no initialization used and when HOSVD, *GEVD* or no initialization is used. Notice that the *I-CPD-G* algorithm is scaled down for clearer visual presentation. The actual number of iterations of the *I-CPD-G* algorithm are 2 oders of magnitude higher.

The fact that *QRT* and *FICA* need approximately an equivalent amount of iterations but have different times suggests that they have different per iteration costs. This difference in cost per iteration can be traced down to the orthogonalization step of both algorithms. The dominant cost of computing the gradient in *FICA* is identical to the dominant cost of computing the tensor slice and updating the data in *QRT*: $O(IPR)$. However, the dominant cost of the QR-decomposition and Eigen-Value-Decomposition (EVD) of the symmetric orthogonalization step differ. While both scale with $O(RP^2)$ there is a difference in the amount of steps each decomposition needs. If for example the QR-algorithm from definition 4.6 is used as EVD then a QR-decomposition has to be computed during each iteration of the EVD. While it is not the fastest EVD algorithm the QR-algorithm does clearly illustrate the difference in computational effort needed and hence why the *QRT* algorithm has a lower per iteration cost than *FICA*.

A small empirical demonstration of this is shown in Figure 5-27. The figure shows how the computation time of the symmetric EVD based orthogonalization and QR-decomposition scales for varying P. This costly difference is due to the aforementioned difference between EVD and QR-decomposition. On top of that, QR-decomposition is performed in a fixed amount of steps while eigenvalue algorithms converge towards the EVD.

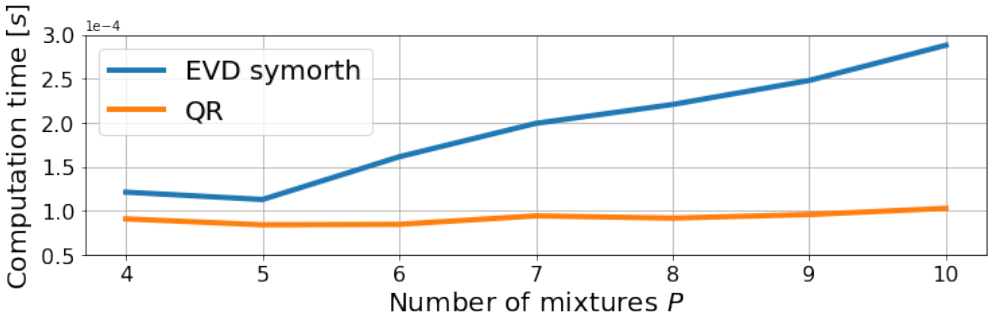


Figure 5-27: Average computation time of the symmetric orthogonalization through EVD and of the QR-decomposition of a random symmetric matrix $\mathbf{M} \in \mathbb{R}^{P \times P}$. The results are averaged over $10e3$ iterations.

Besides differences in per iteration costs the following is observed about the error convergence of *FICA*, *QRT* and *I-CPD-FF*. Figure 5-28 shows the total amount of iterations need but this time with the standard deviation for only the no initialization scheme. The figure shows that for both *FICA* and *QRT* there is a lot of variance in the amount of iterations needed for higher values of $P \geq 6$. Furthermore, both *QRT*'s and *FICA*'s amount of average iterations seem to decrease growth after $P = 8$.

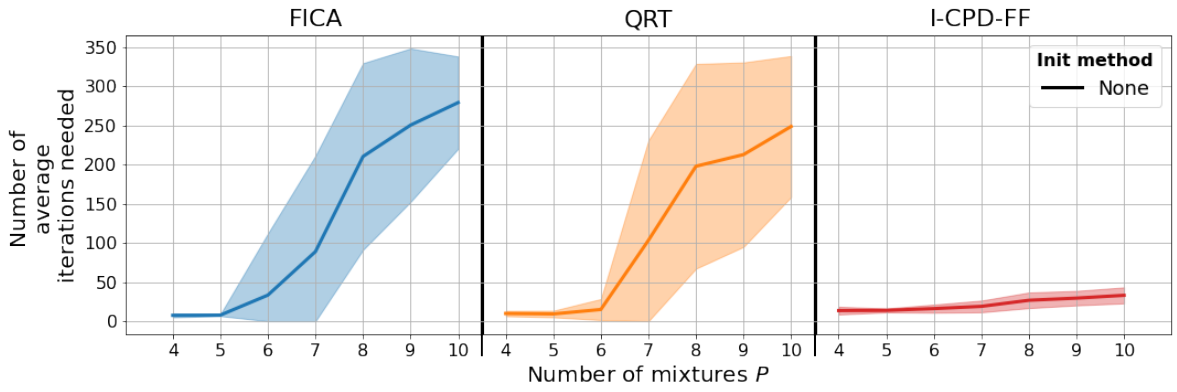


Figure 5-28: Average total amount of iterations needed per algorithm with the standard deviation for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs. For each algorithm the results are plotted only with no initialization scheme used.

To understand why this is the case the general behavior of the convergence errors is studied. The convergence errors for a single run of $P = 4, 7, 10$ is shown for these 3 methods in Figure 5-29. The figure clearly shows why both *FICA* and *QRT* need many more iterations than *I-CPD-FF*. As P increases the behavior of the convergence error of *FICA* and *QRT* becomes less predictable whereas for *I-CPD-FF* the behavior remains consistent. For this particular example for $P = 10$ *FICA* did not even manage to reach convergence within 200 iterations while these errors are from a solution which is considered as successful. For this particular example *QRT* manages to reach convergence for $P = 10$. However, there are many cases which are similar to the one of *FICA* shown here.

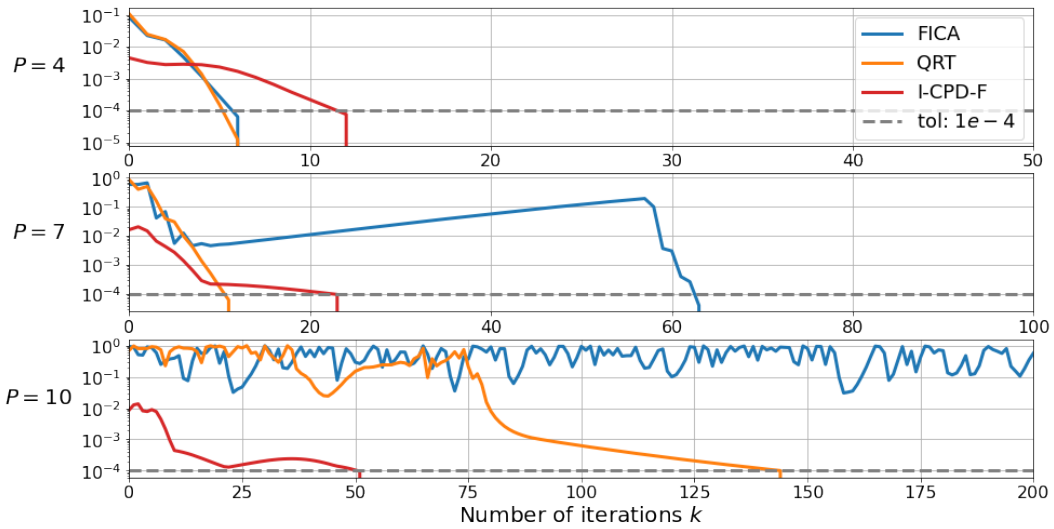


Figure 5-29: Convergence errors of the *FICA*, *QRT* and *I-CPD-FF* methods with no initialization scheme for $P = 4, 7, 10$. The figure shows that as P increase so does the number of iterations needed to reach convergence. However, whereas *I-CPD-FF* presents consistent convergence behavior the *FICA* and *QRT* methods do not. Up to the point where for high values of $P \geq 8$ *FICA* never reaches convergence. Note that from top to bottom the x-axis increases in size.

The same result but then averaged over 100 successful runs is shown below in Figure 5-30. This figure shows that on average both QRT and $FICA$ suffer from the problem of possibly never reaching convergence for this range of iterations while still producing successful estimations.

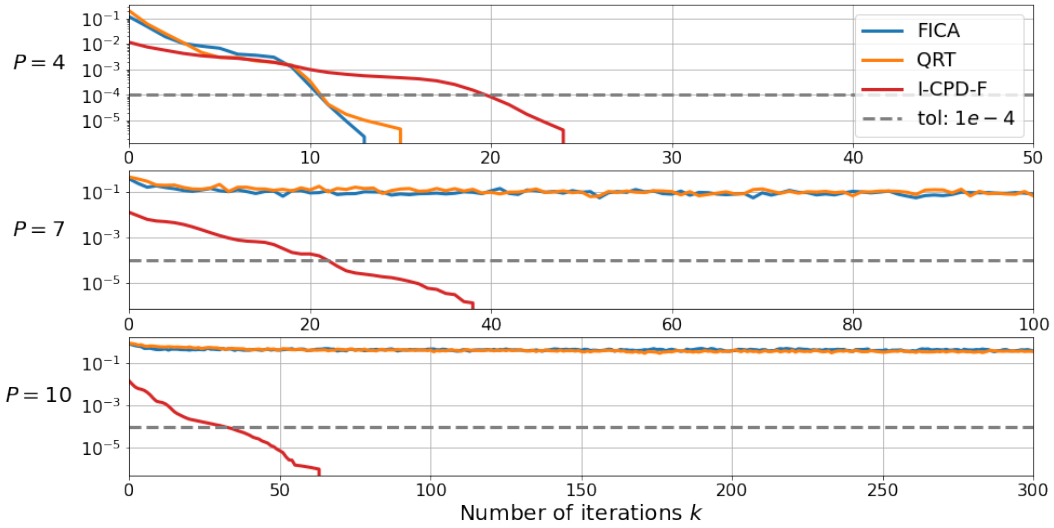


Figure 5-30: Average convergence errors of the *FICA*, *QRT* and *I-CPD-FF* methods with no initialization scheme for $P = 4, 7, 10$. The figure shows that as P increase so does the number of iterations needed to reach convergence. However, whereas *I-CPD-FF* presents consistent convergence behavior the *FICA* and *QRT* methods do not. Up to the point where for high values of $P \geq 8$ they never reach convergence. Note that from top to bottom the x-axis increases in size.

This can be deduced by the fact that there is not a general iteration after which the *FICA* and *QRT* errors drop for $P = 7$ and $P = 10$. Conversely, *I-CPD-FF* does on average manage to reach convergence after a set amount of iteration. This partially explains the decrease in growth of the algorithm iterations from Figure 5-26. Both the *FICA* and *QRT* algorithms often reach the iteration ceiling of $K_{max} = 300$ while for some runs they converge in fewer iterations which averages out as a decrease in average iteration growth. Nevertheless, the implication of it all is that both *FICA* and *QRT* take a long time to converge for high values of P and that the convergence measure from 5-3 does not in all cases properly indicate when these methods have estimated a solution as they do manage to produce successful solutions without reaching convergence.

Lastly, Figure 5-31 shows the computation time of each algorithm for only no initialization method where the tolerance has been set to $tol = 0$ such that all algorithms perform the same amount of computations for all values of P. Comparing the average computation times with the dominant computational complexities shown in Table 5-8 shows that the algorithms behave accordingly to their theoretical scaling.

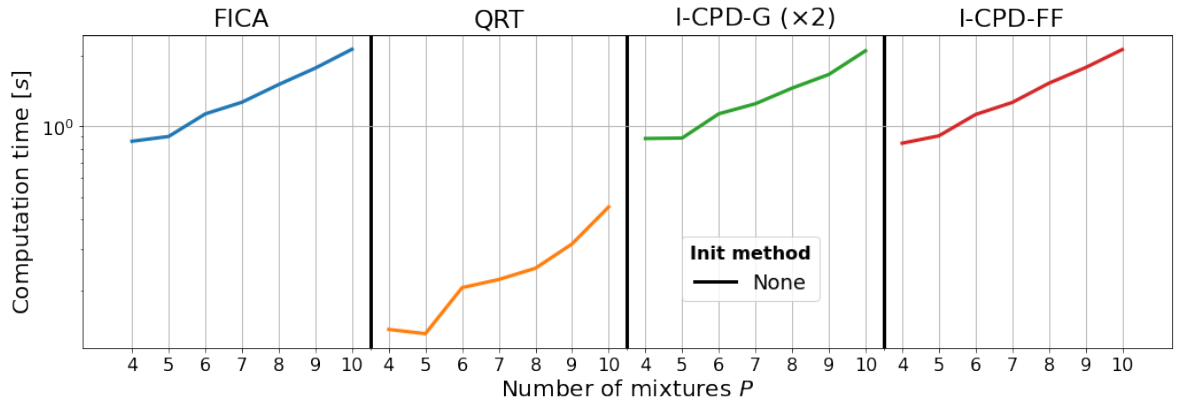


Figure 5-31: Average runtime for varying amounts of mixtures $P = 4, \ldots, 10$ of runs classified as correct for a total of $N_{test} = 100$ runs on a logarithmic scale. For each algorithm the results are plotted with no initialization used and the tolerance is set to $tol = 0$ such that all algorithms perform an identical amount of iterations for each value of P.

Dominant per iteration cost			
FICA	*QRT*	*I-CPD-G*	*I-CPD-FF*
IRP	*IRP*	*IRP*	*IRP*

Table 5-8: Dominant computational complexities of iterative algorithms. In the experimental setup the number of estimated components R was chosen continuously to be equal to the amount of mixtures P.

5-3 Chapter summary and discussion of results

This section discusses the previously found numerical results. The reader is noted that all statements made in the following are in relation to the performance on the used data-set. This does not mean that these results generalize to all cases. However, where thought possible it is suggested what the found results would imply for general performance.

Non-iterative methods Starting with the non-iterative algorithms only, it can be stated that the explicit GEVD, implicit full GEVD and implicit GEVD type-U methods in general perform better than their HOSVD counterparts for $P \geq 8$ due to the lower average errors and the slightly higher percentages of successes. However, the overall performance of all HOSVD methods remains more consistent when varying P. The U sub-type of both implicit HOSVD and implicit GEVD show similar results and performance as their full implicit counterparts. Note that the full implicit HOSVD is identical to the explicit HOSVD while the full implicit GEVD is not identical to its explicit counterpart. As all GEVD based methods perform HOSVD too, these results suggest that the additional GEVD step does help with estimating the source components as long as the 2 used slices in the GEVD contain enough information of all other tensor slices. The reasoning behind this can be understood when looking at the implicit HOSVD and implicit GEVD C-types. Both C-types performed worse than their other variants. However, the C-type implicit GEVD algorithm performed considerably the worst as no estimation was considered as successful. On top of that, the average error of all runs is much higher compared to the other GEVD methods. This can be attributed to the fact that due to an incomplete HOSVD the first 2 slices of the resulting tensor core are not a proper linear combination of all original slices. Next to this the implicit GEVD algorithms already use less of the tensor's information due to the partial inwards projection. combining these 2 facts is what most likely caused the GEVD step to fail when only using the core slices.

Considering the computation times of the non-iterative algorithms it can be stated that a significant trade-off is made when using the full implicit versions. While the implicit algorithms have a lower storage complexity compared to their explicit versions, the added SVD-update cost dominates the computation time of both algorithms and as a result only their C-types are faster than the explicit algorithm. While the theoretical complexities cannot be directly changed a faster implementation of the implicit HOSVD is most definitely possible. As was said before, the current implementation contains a nested *Python* loop which is slow. On top of that, much of the SVD-update code is written in *Python*. *Python* is a versatile language, yet by far not the fastest. Writing the implicit HOSVD code in a faster language such as C and directly implementing the SVD-update procedure into it can speed up the algorithm by removing any over-head cost caused by *Python*. Nevertheless, the computaitonal time results of the non-iterative methods suggest that they could be practical when time is not an issue but storage is.

Iterative methods When using no initialization scheme it can be stated that both the *QRT* and *I-CPD-FF* algorithms produce similar results as FastICA, albeit with slightly higher estimation errors and lower tensor diagonality scores. The gradient based *I-CPD-G* approach performed a lot worse than all other methods which corresponds with the performance of

gradient methods in general for ICA[21]. As such, its performance is not further discussed here.

The advantage both the *QRT* and *I-CPD-FF* have over FastICA is the reduced computation time they need when increasing the amount of mixtures P to reach convergence. With the former this is the result of a lower per iteration computational cost and with the latter it is the result of converging in fewer iterations. However, similar to FastICA the *QRT* algorithm exhibits convergence issues for higher values of P. As a consequence both algorithms cannot stop prematurely which results in unnecessary higher computation times considering the fact that both methods might still produce a successful estimate.

This has several practical implications. First of all this introduces an uncertainty in the quality of solutions which have not converged as they might in fact actually be considered as adequate estimations. Secondly, not having a properly functioning convergence tolerance stopping criterion might lead to unnecessary long computation times. This means one might resort to shotgun-hail tactics to find an adequate maximum number of iterations which adds up extra time.

The *I-CPD-FF* algorithm on the other hand was shown to reach convergence within a consistent amount of iterations between different runs. On top of that, for $P \geq 5$ the algorithm reaches convergence in considerably fewer iterations than both FastICA and *QRT*. It is believed that this is due to the additional terms in the *I-CPD-FF* gradient together with the internal computation of a $\boldsymbol{\lambda}$ estimate. As was mentioned before, *I-CPD-FF* could be considered as a CPD constrained version of FastICA where the gradient does not point in the direction of a locally maximum sum of absolute self kurtosis but in the direction of a locally best fitting CPD model. Additionally, the internal multiplication of the gradient of the factors with the gradient of the $\boldsymbol{\lambda}$ estimate is in fact a relative scaling correction of the gradient's rows. This can be considered as a weighting of the estimated source components during each iteration based on their self-kurtosis value. Components with higher self-kurtosis estimates lead to higher weights and thus have a more dominant effect in the direction of the solution.

The faster convergence of the fixed-point *I-CPD-FF* algorithm at higher values of $P > 4$ does not mean that it will always converge faster. The experiment contained only 4 latent source signals without including the GWN. It may very well be that as the amount of source components is increased too the convergence performance of the *I-CPD-FF* algorithm deteriorates. However, we cannot draw any such conclusion based on the results of our small experiment. What we can say is that the results suggest *I-CPD-FF* may converge faster for problems with more source signals.

In conclusion, it can be stated that the definitive differences of the algorithms compared to each other and to FastICA are to be found in the convergence behavior and computational speed. Furthermore, both the fixed-point first-order CPD method and the QRT method can find solutions of similar quality to a BSS problem in less time than FastICA.

Iterative methods with initialization schemes The 7 different initialization schemes are categorized as HOSVD based methods or GEVD based methods. Table 5-9. Shows the qualitative comparison of each algorithm/initialization scheme pair with the case when no initialization scheme is used. Overall, the HOSVD category yielded better results when used as an initial estimate for the iterative methods. The GEVD based methods resulted in most

cases in worse performance of all 4 iterative algorithms with the non-orthogonality of the initial estimate even breaking the QRT method completely. The reason why FastICA, *I-CPD-G* and *I-CPD-FF* can deal with the non-orthogonality is because of the symmetric orthogonalization step present in said methods. This implies that the factor matrix from the GEVD methods used for initialization could be symmetrically orthogonalized before passing it through to the *QRT* algorithm. However, this would not make it the least squares approximate solution of the inwards projections step anymore which essentially means it loses all meaning. Nevertheless, it can be concluded that the lack of orthogonality of the factors and the asymmetry of the CPD the GEVD methods compute do not benefit the presented ICA algorithms.

	HOSVD			GEVD			
	Explicit / Full	Type-U	Type-C	Explicit	Full	Type-U	Type-C
FICA	0	0	0	−	0	0	0
QRT	+	++	+	—	—	—	—
I-CPD-G	+	+	+	−	−	−	−
I-CPD-FF	+	++	+	−	0	0	0

Table 5-9: Qualitative analysis of the 4 iterative algorithms with all of the different initialization schemes. Each algorithm/initialization scheme pair is compared to the case when no initialization scheme is used. A 0 indicates that no clear difference was observed when using that combination, a + indicates minor improvements in overall results, ++ indicates the best overall performance increase, a − indicates a slight deterioration in performance and — indicates total failure when using said combination.

Without any additional initialization scheme, non of the newly introduced methods showed a significant advantage over FastICA when considering the successful amount of solutions and the estimation errors. However, the results of both the *I-CPD-FF* algorithm and the *QRT* algorithm did improve when initialized with HOSVD based methods even surpassing that of fastICA on estimation error or percentage of successes. This suggests the following: that these methods too are sensitive to initial estimates and that these type of initializations may result in overall better and more robust source component estimation.

Somewhat surprising, the U-type implicit HOSVD method resulted overall in the best performance increase when combined with the *QRT*, *I-CPD-G* and *I-CPD-FF* methods. It is unclear whether this method outperforms explicit HOSVD as an initialization method in general. However, it is believed that the results produced by the U-type implicit HOSVD method for this-dataset aligned favorably with possible solutions. For both the *QRT* and *I-CPD-FF* algorithms the C-type implicit HOSVD method performed adequately too as the performance was never lesser than initialization with a complete HOSVD. The above 2 findings do not prove nor do they imply the general success of these HOSVD based methods. However, it does show that using different initial estimates based on varying amounts of information of the cumulant tensor can potentially improve results for a specific problem in a consistent way.

Concerning the computational time of iterative methods combined with non-iterative methods for initialization the following can be said. Due to the scaling of the non-iterative algorithms the cost of computing of an initial estimate can quickly exceed the computation time needed by the iterative algorithm while there not being any guarantee the initial estimate will help.

This raises the question whether or not it is better to retry the iterative algorithm with several random estimates instead. There is no definitive answer to this question. However, if wanting to use one of the non-iterative algorithms for initialization it is best practice to start with the C-types and work up towards the full explicit algorithms if the need for better estimates persists.

Overall, the initial impression is that the QRT and *I-CPD-FF* method prove to be a valid alternative to FastICA in terms of estimation error and solution correctness while shown to be superior in terms of convergence and speed. All of the presented and discussed findings are summarized as the following list of contributions.

5.1: Contributions chapter 5

- It is shown that the implicit HOSVD and implicit GEVD algorithms do not scale favorably with the amount of mixtures P due to the total SVD-update cost.
- It is shown that computing the implicit HOSVD of the cumulant tensor with only its unique slices or only its core slices can result in similar performance for ICA as computing the full HOSVD.
- It is shown that for a linearly mixed BSS problem with additive GWN the source component estimation performance of QRT and *I-CPD-FF* is on par with that of kurtosis-based parallel FastICA in terms of estimation error and solution correctness.
- It is shown that the *QRT* and *I-CPD-FF* algorithms can achieve better results than FastICA when combined with the HOSVD based algorithms as initial estimates in terms of correctness and estimation error.
- It is shown that the **computation time** of the QRT algorithm to reach a certain convergence tolerance is lower than that of FastICA by nearly an order of magnitude due to its lower per iteration cost.
- It is shown that a **fixed-point** iteration of the implicit CPD first order optimization method converges in fewer iterations towards a solution than FastICA. As a result the computational time needed by the implicit CPD first order optimization method scales much more favorably with P as it does for FastICA.
- It is shown that for the case when FastICA and *QRT* struggle to reach convergence, the fixed-point implicit CPD first order optimization method manages to converge in a consistent and much smaller amount of iterations.

Chapter 6

Conclusion and Recommendations

6-1 Thesis Conclusion

In this thesis it has been researched whether implicit decomposition of the cumulant tensor can be used to solve the Blind Source Separation (BSS) problem. The objective was to derive efficient cumulant tensor diagonalization methods by decreasing the cost of its manipulation and storage. Cumulant tensor methods suffer from the curse of dimensionality due to their quartic dependence on the amount of mixtures $O(P^4)$. To address this scalability issue, the tensor's symmetry and structure are exploited such that its diagonalization through tensor decomposition is performed implicitly. The result is a class of implicit cumulant tensor decomposition algorithms which scale more favorably than their explicit counterparts in terms of either computational cost, storage cost or both.

This thesis is concluded by revisiting the research question posed in the first chapter together with its sub-questions:

Can implicit decomposition of the cumulant tensor for Independent Component Analysis (ICA) provide a competitive alternative to fastICA in terms of convergence, solution quality and cost?

- *Can the storage cost and computational cost of manipulating the cumulant tensor be reduced through the use of an implicit computation scheme?*
- *Can tensor decomposition methods which have already existing uses for ICA benefit from this implicit computation scheme?*

The implicit cumulant tensor $\mathcal{C}^{(4)} \in \mathbb{R}^{[4,P]}$ computation scheme presented in this thesis is the cornerstone of all derived implicit methods. It allows for the computation of any part of the tensor of arbitrary size. With this scheme, two low cost iterative algorithms have been derived which diagonalize the cumulant tensor through its implicit decomposition. The computational and storage cost of each method scales linearly with the amount of estimated

components $O(RP)$, opposed to the previously quartic dependence on the amount of mixtures $O(P^4)$. Both methods have a theoretical storage and per iteration computation cost identical to that of FastICA.

One of these methods iterates through the core slices of the cumulant tensor and while performing successive QR-decomposition to increase the tensor's diagonality. Akin to the QR-algorithm for matrices, the QR-Tensor algorithm simultaneously diagonalizes the set slices of any designated top-layer of a symmetric tensor. Analysis of the unique structure this decomposition introduces into a tensor showed that for ICA it can be simplified by computing only the QR-decomposition of the columns along the superdiagonal. Combined with the implicit computation strategy this results in an efficient method for finding a solution to the BSS problem at the computational cost of $O(RP)$ per iteration and with a storage cost of $O(P^2)$.

The second low-cost method is a fixed-point iteration of a gradient-based first-order Canonical Polyadic Decomposition (CPD) optimization problem for the cumulant tensor. The first-order gradient-based method uses a gradient for the factors $\mathbf{U}$ and a gradient for the scaling values $\boldsymbol{\lambda}$ to find a locally best fitting CPD model. Bearing a lot of similarity to the gradient of FastICA, it is shown that the problem is in fact FastICA with a CPD constraint. In order to produce an orthogonal result a symmetric orthogonalization scheme is introduced into the optimization problem. Moreover, due to the scaling indeterminacy of ICA only direction of the solution is importance. For this reason the first-order optimization problem is rewritten as a fixed-point iteration. Both the gradient and fixed-point approach have a per iteration computational cost of $O(RP)$ and a storage cost of $O(RP)$.

Additionally, two non-iterative implicit cumulant tensor decomposition methods are derived. The first computes the Higher-Order Singular Value Decomposition (HOSVD) of the cumulant tensor implicitly. It does so through a Singular-Value-Decomposition (SVD)-updating method and keeps the storage cost at $O(P^2)$. However, the scaling of the computational cost of performing the implicit decomposition as become less favorable with P compared to the explicit HOSVD. While not equivalent to a diagonalization transformation, it is shown through a small numerical example how ICA can benefit from initialization with HOSVD. As such, 2 variants of the implicit HOSVD are proposed which update the SVD with fewer slices to reduce overall computation cost. The U-type uses only the tensor's unique slices and the C-type uses only the core slices.

The second non-iterative implicit algorithm computes an approximate non-symmetric CPD through Generalized EigenValue Decomposition (GEVD) of 2 slices of the cumulant tensor's HOSVD. From literature it is known that the explicit GEVD procedure is often used for finding initial estimates of a tensor CPD. The connection between CPD and cumulant tensor diagonalization suggests the use of the GEVD method as initialization for ICA. The computational cost of the algorithm scales nearly identical as the implicit HOSVD method. For this reason it too has been given a U-type and C-type variant.

Finally, the derived implicit tensor decomposition algorithms have been tested on an artificial linear BSS separation problem with additive Gaussian-White-Noise (GWN) consisting of four source components. By comparison of the amount of successful estimations, the estimation errors and the cumulant tensor diagonality it is shown that both the fixed-point CPD and QRT method perform nearly identical to kurtosis based parallel FastICA. However, in terms of reaching convergence and computation time on the presented problem both methods are shown to be superior to FastICA. Due to the QRT having a lower per iteration cost and the

fixed-point CPD method needing fewer iterations to reach convergence. For the latter it is shown that for higher values of $P >> 4$ it keeps converging within a fixed-amount of iterations while both QRT and FastICA struggle to do so. The gradient-based first-order approach does not perform well relative to the other methods which is in line with results from literature.

The implicit HOSVD and GEVD methods and their variants show similar or identical performance in solving the BSS with respect to their explicit counterparts. However, when used as an initialization for the iterative algorithms the HOSVD based methods proved to be superior in terms of estimation improvement. Conversely, the GEVD methods are shown to deteriorate performance at varying levels. The asymmetry and the non-orthogonality of the solution they produce is stated to be counterproductive in aiding ICA algorithms. It was shown that computing the implicit HOSVD and GEVD based decompositions comes at the price of poorer scaling with the amount of mixtures P. Effectively this means a trade-off is made between computational cost and storage cost.

To answer the research question and sub-questions, implicit decomposition of the cumulant tensor can solve the BSS problem at storage and computational costs equivalent to that of the state-of-the-art FastICA whilst converging faster. The presented partial computation scheme allows for efficient and selective computation of the cumulant tensor entries necessary for updating the current estimate of the mixing matrix.

6-2 Recommendations for Future research

Based on the theoretical and numerical results of this thesis we suggest several topics for future research as we believe they show potential.

In depth performance analysis of the presented methods for ICA The numerical experiment in which the algorithms are tested in solving BSS served as but a proof of concept. In order to better understand the performance of the presented fixed-point CPD and QR-Tensor algorithm (QRT) algorithms for ICA and how they measure up to FastICA a more extensive experiment must be performed. As was mentioned before, we increased the amount of mixtures while our experiment only contained 4 source signals. The estimation performance and convergence behavior of the algorithms may be different for problems containing more latent source components. Our results do not exclude whether the faster convergence of the QRT and fixed-point CPD methods is only due to the relatively low amount of source components present $P = 4$. Moreover, we only considered a symmetric BSS problem were the amount of estimated components R is equal to the amount of mixtures P. Our algorithms allow for the estimation of fewer components than mixtures used. It is unclear whether performance changes when doing so. We suggest that a scalable BSS experiment should be performed such that the effects of varying amounts of source components can be studied together with the asymmetric case $P > R$.

Furthermore, the latent source components we used are simple for the reason of being able to easily identify whether an estimation can be considered as successful or not. However, in real-life the source components can be a lot more complex and are not so easily identifiable from each other. In order to truly benchmark the performance of our methods we suggest that they should be tested on a BSS problem with real practical signals of which the solutions

are known. For example, separate audio signals or vectorized images can be used as source components.

Lastly, the non-iterative HOSVD and GEVD based methods were used for the computation of initial estimates. Our results suggest that the implicit HOSVD can improve estimation results of other iterative algorithms. However, it is unclear whether this benefit remains when problem size is increased. We suggest to take this into the same scalable experiment mentioned above. We advise that before doing so, a faster implementation of the implicit HOSVD algorithms is coded which circumvents part or all of the overhead cost caused by *Python* as was described in chapter 5.

Implicit CPD algorithms All of our derived algorithms for computing the cumulant tensor CPD have been tested for their performance of performing BSS. However, the performance of their original intended use of computing a CPD has never been explored. Computing the CPD allows for a compressed format of the cumulant tensor which can be desirable if its values are often needed. Good CPD performance does not necessarily imply a good ICA performance. As such, our results do not illustrate whether the implicit Canonical-Polyadic-Generalized-Eigenvalue-decomposition (CP-GEVD) and first-order CPD methods are adequate competitors for computing the cumulant tensor CPD. For ICA the factor matrices were constrained to be orthogonal. However we believe that this orthogonality constraint is not beneficial in CPD estimation. We suggest to test the implicit GEVD and its variants, the first-order gradient CPD and first-order fixed-point CPD on their performances of computing the cumulant tensor CPD.

QRT algorithm The true purpose of the novel QRT algorithm we presented is the simultaneous diagonalization of a symmetric tensor's top outer-slices. While writing this thesis we did not find any literature which presents a similar process, nor have we found any cases in which it is shown that this outer-slice diagonal form is desired. We used the algorithm for the approximate diagonalization of the cumulant tensor as the structure it introduces into the cumulant tensor resulted in approximately independent components. It is not unimaginable that the unique structure it introduces can have other uses too. However, instead of looking for potential use-cases, we suggest that the algorithm and its workings should be further studied. We hypothesized that its working principle and convergence are closely intertwined with that of the QR-algorithm for matrices. As such, we believe it is of interest to understand what the unique structure QRT introduces into a tensor signifies mathematically.

On top of that, the algorithm was shown to converge fast for low values of P but struggled at higher values of P. The QR-algorithm for matrices can be sped up by applying what is known as a shift [73][74][75]. The QRST algorithm from [72] which has a lot of similarity to the QRT algorithm uses a shift for faster convergence too. This raises the question whether the convergence of QRT can be sped up too using a shift and whether this results in better estimation performance.

Lastly, it is desirable to prove general convergence of the QRT algorithm if it exists in the first place. Many shifted versions of the QR-algorithm for matrices have never been proven to always converge while no counter-examples have been found either. Proving convergence of the QRT algorithm will most likely prove to be difficult to say the least. Nevertheless,

further investigation of the QRT algorithm may open new doors in our understanding of how bi-linear algebraic methods such as QR-decomposition work with multi-linear objects.

Combining implicit tensor decomposition with diagonalization methods In this thesis we only studied how implicit tensor decomposition of the cumulant tensor which simultaneously diagonalizes it can be used for ICA. However, there are many other tensor decomposition formats. Many of them allow for compression of a tensor, i.e., (approximately) representing the full tensor with fewer values. It is unclear whether other specific tensor decomposition formats can aid algebraic diagonalization methods such as the COM2 algorithm. As such, we recommend the exploration of other decomposition formats in combination with algebraic ICA algorithms.

Group-method with FastICA Instead of pitting the algorithms against each other, they can be used together as a group-method. Literature shows that group-methods provide in general the most reliable results as they have a robust performance. The most commonly used group-method is one in which the FastICA algorithm is run multiple times with varying settings and initial conditions after which the results are clustered and combined to form the final estimate. When using varying ICA methods the problem group-methods have is that the whole process is bottle-necked by its slowest ICA algorithm. However, our presented algorithms will most likely not be the bottleneck. For this reason we suggest experimenting with a group-method of only the QRT or fixed-point CPD methods and a group-method of said methods combined with FastICA.

Appendix A

Statistics

In this appendix several general definitions related to moments and cumulants are presented.

A-1 Derivation of moments and cumulants

How cumulants are found follows the same method of how moments are derived using the moment generating function (mgf). This section briefly outlines this process. Any formal and extensive proof can either be found in the appendix or is left out entirely due to its complexity.

Moment generating function The probability distribution of a random variable x can be fully defined using a complex function named the Characteristic function (CF). The first CF is defined as the continuous Fourier transform of the probability density function $p_x(x)$:

$$\varphi(\omega) = \mathrm{E}[e^{i\omega x}] = \int_{-\infty}^{\infty} e^{(i\omega x)} p_x(x) dx \tag{A-1}$$

where i represents the imaginary unit $\sqrt{-1}$. By expanding the first CF into its Taylor series an expression is derived of which the coefficients of the infinite sum are the earlier presented statistical moments, hence the name moment generating function.

Definition I: Moment generating function (mgf)

The **moment generating function** is defined as the **Taylor series expansion** of the first characteristic function $\varphi(\omega)$ from (A-1):

$$\varphi(\omega) = \int_{-\infty}^{\infty} \left(\sum_{k=0}^{\infty} \frac{x^k (i\omega)^k}{k!} \right) p_x(x) dx = \sum_{k=0}^{\infty} \mathbb{E}[x^k] \frac{(i\omega)^k}{k!}, \tag{A-2}$$

where the expansion **coefficients** are given by the statistical moments $\mathbb{E}[x^k]$. In other words, evaluating the derivatives of the mgf at zero gives the statistical moments:

$$\left. \frac{d^n \varphi(\omega)}{d\omega^n} \right|_{\omega=0} = \mathbb{E}[x^n]. \tag{A-3}$$

Cumulant generating function The second characteristic function is defined as the natural logarithm of the first characteristic function from (A-1):

$$\phi(\omega) = \ln(\varphi(\omega)) = \ln(\mathbb{E}[e^{i\omega x}]). \tag{A-4}$$

As the first characteristic function can be used to fully define a probability distribution of a random variable x, its natural logarithmic transformation inherently has the same property. However, due to this transformation the Taylor series expansion of (A-4) can be used to obtain the cumulants of a probability distribution which exhibit some interesting properties which moments do not share, especially in the higher-order case. The first three cumulants are directly related to the first 3 central moments, but from the fourth cumulant on this is not directly the case anymore. Instead, the fourth cumulant is related to the excess kurtosis presented before.

Definition II: Cumulant generating function (cgf)

The **cumulant generating function** is defined as the **Taylor series expansion** of the second characteristic function $\phi(\omega)$ from (A-4):

$$\phi(\omega) = \sum_{k=0}^{n} \kappa_k \frac{(i\omega)^k}{k!}. \tag{A-5}$$

By evaluating the **derivatives** of the cumulant generating function (cgf) at 0 the **cumulants** κ of the probability distribution are obtained:

$$\kappa_k = (-i)^k \frac{d^k \phi(\omega)}{d\omega^k}\bigg|_{\omega=0}. \tag{A-6}$$

The **first** cumulant of a random variable x is equal to the mean of x:

$$\kappa_1 = \mathbb{E}[x] = m_x. \tag{A-7}$$

The **second** cumulant equals the second central moment of x:

$$\kappa_2 = \mathbb{E}[(x - m_x)^2] = \sigma^2. \tag{A-8}$$

The **third** cumulant is equal to the third central moment of x:

$$\kappa_3 = \mathbb{E}[(x - m_x)^3]. \tag{A-9}$$

A-2 Moments

Moments are measures that describe a function in a quantitative way. When such a function describes a probability distribution, these moments represent statistical quantities of a distribution such as mean, variance, skewness and kurtosis depending on whether they are centralized or standardized.

Definition III: Statistical moments

The j'th **moment** of a random variable x's probability distribution, is denoted as the **expectation** of x to the power j which is equal to the following integral:

$$\mu_j = \mathbb{E}\left[x^j\right] = \int_{-\infty}^{\infty} \xi^j p_x(\xi) d\xi, \tag{A-10}$$

where $p_x(x)$ denotes the **probability density** function of x.

The **first moment** of a random variable x is called the **mean** m_x of that variable. In vector form this can be denoted as the **mean vector** $\mathbf{m_x}$:

$$\mathbf{m_x} = \mathbb{E}\left[\mathbf{x}\right] = \int_{-\infty}^{\infty} \boldsymbol{\xi} p_{\mathbf{x}}(\boldsymbol{\xi}) d\boldsymbol{\xi}, \tag{A-11}$$

From the ordinary statistical moments the mean represents a measure of where a distribution is located. It can be used to centralize a distribution such that other characteristics of a distribution around its mean can be found which are suitably named: central moments.

A-2-1 Central moments

Central moments offer a more meaningful characterization of a distribution as they represent measurable quantities of said distribution around its mean. This means these quantities can be used to describe a distributions shape around its center.

Definition IV: Central moments

The j'th **central moment** of a random variable x's probability distribution about the random variable's **mean**, is denoted as the j'th moment of the mean m_x subtracted from x prior to taking the expectation:

$$\alpha_j = \mathbb{E}\left[(x - \mu_1)^j\right] = \int_{-\infty}^{\infty} (\xi - m_x)^j \, p_x(\xi) d\xi. \tag{A-12}$$

The **first central moment** of a random variable x equals 0 as the mean is subtracted from itself:

$$\alpha_1 = \mathbb{E}\left[x - m_x\right] = \mathbb{E}\left[x\right] - m_x = 0. \tag{A-13}$$

The **second central moment** of a random variable x is called the **variance**:

$$\alpha_2 = \mathbb{E}\left[(x - m_x)^2\right] = \sigma^2 \tag{A-14}$$

which is equal to the **standard deviation** σ squared.

A-2-2 Standardized moments

In order to be able to properly compare distributions, moments can be standardized. Standardizing moments renders them scale invariant which means that when scaled by some factor the distribution they represent does not change shape.

Definition V: Standardized moments

A **standardized** version of a **central moment** a_j is identical to said moment normalized through some expression of the **standard deviation** which renders the moment **scale invariant**. The standardized central moment of degree k is denoted as:

$$\overline{\alpha}_k = \frac{\alpha_k}{\sigma^k}. \tag{A-15}$$

By centralizing and standardizing moments many expressions can be simplified as terms consisting of the mean are eliminated and unit variance allows for direct comparison of distributions. As is shown later on in this review through whitening of data, this is beneficial for solving the Blind Source Separation (BSS) problem.

Appendix B

Independent Component Analysis (ICA)

This appendix chapter contains additional elaboration on the varying type of ICA models and the problem's properties. Moreover, the derivation of the computaitonal cost of the cumulant tensor is shown, the derivation of the computational cost of the COM2 algorithm is shown and a short explanation on binomial coefficients is shown.

B-1 Mixture models

In [17] the Blind Source Separation (BSS) problem is clearly defined as the retrieval of the original source signals $\mathbf{s} = (s_1, s_2, \ldots, s_N)^T \in \mathbb{R}^N$ from a set of observed signals $\mathbf{x} = (x_1, x_2, \ldots, x_P)^T \in \mathbb{R}^P$ which are directly related to each other through a mixture model $\mathcal{A}$ shown in definition B-1. The source signals are latent variables which means that they cannot be directly observed [105].

Definition B.1: Mixture model $\mathcal{A}$ [17]

A **mixture model** consists of a mapping $\mathcal{A}$ which maps a set of **source signals** $\mathbf{s} = (s_1, s_2, \ldots, s_N)^T \in \mathbb{R}^N$ to a set of **output signals** $\mathbf{x} = (x_1, x_2, \ldots, x_P)^T \in \mathbb{R}^P$:

$$\mathbf{x} = \mathcal{A}(\mathbf{s}). \tag{B-1}$$

Solving the BSS problem amounts to identifying the inverse mapping of the mixture process $\mathcal{Q}$ such that the source components can be found through $\mathbf{s} = \mathcal{Q}(\mathbf{x})$. Occasionally information on the mixture model can be inferred from the context of the problem at hand. For example, crosstalk which is typically encountered in multi-user communication systems such as by phone carriers, is when the signals of several mobile users on the same frequency channel interfere with each other causing co-channel interference (CCI). Due to the different spatial

origins of the signals it is natural to think of the mixture as a weighted sum of the individual signals where each weight simply reflects the relative distance of its corresponding signal.

However, such simple to interpret prior knowledge about the mixing model $\mathcal{A}$ is not always available. Therefore preliminary assumptions have to be made about the mapping's structure. These assumptions boil down to whether the mixing model is stationary, linear or nonlinear and whether the problem is over- or under-determined.

B-1-1 Stationary problem

The BSS problem is considered stationary when the mapping $\mathcal{A}$ does not change during sampling intervals.

Definition B.2: Stationary Mixture Model

A **mixture model** is **stationary** if its mapping $\mathcal{A}$ is independent of the time of sampling:

$$\mathcal{A}(t) = \mathcal{A} \quad \forall t, \tag{B-2}$$

where t represents the sampling index.

For non-stationary mixtures this mapping is dynamic and behaves differently during different sampling intervals. In many cases it can be assumed that this dynamic behaviour is a function related to time [106] [107] [108] but non-monochromatic mappings in the time-frequency domain have been identified too [12].

B-1-2 Overdetermined problem

The BSS problem is overdetermined whenever the amount of sources to be identified is smaller than or equal to the amount of observed signals $N \leq P$. This condition is in many cases necessary if the inverse transformation of the mapping $\mathcal{A}$ is used to recover the source signals $\mathbf{s}$. For linear mixture models this means that if $N < P$, in most cases the observed data $\mathbf{x}$ is first to be projected onto a subspace with N dimensions after which the mixing matrix $\mathbf{A}$ can be inverted to obtain the reconstructed source components. Alternatively if the mixing matrix $\mathbf{A} \in \mathbb{R}^{P \times N}$ is semi-orthogonal so $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ with identity $\mathbf{I} \in \mathbb{R}^{N \times N}$, then its transpose can simply be used to unmix the signals.

B-1-3 Underdetermined problem

The undetermined case is given when the amount of components to be estimated is more than the amount of observed mixtures $P > N$. This makes the problem significantly more difficult. Additional assumptions on data sparsity [109][110][22][111] or specific conditions on the sensors [24] can ease the problem to some degree.

B-2 Mixture models

As was mentioned before, to find a solution the BSS problem a mixing model must be selected for the mapping $\mathcal{A}$ from (B-1) which can be either linear or nonlinear. Both linear and nonlinear models are classified in literature as one of 2 types [17][5]: instantaneous or convolutive. The research presented in this thesis concerns itself with a linear instantaneous model.

Linear model When the mixture is assumed to be linear the observed signals $\mathbf{x}$ are considered as a linear combination of the source signals $\mathbf{s}$.

Definition B.3: Linear Mixture model A

For a **linear mixture model** the set of **output signals** $\mathbf{x} = (x_1, x_2, \ldots, x_P)^T \in \mathbb{R}^P$ consist of a weighted sum of the source signals $\mathbf{s} = (s_1, s_2, \ldots, s_N)^T \in \mathbb{R}^N$:

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \tag{B-3}$$

where the mapping $\mathbf{A}$ is a $P \times N$ matrix of which its entries $\{a_{i,j} \quad \forall i \in P, j \in N\}$ are scalar values denoting the weight coefficients of the mixture.

The linear model has been widely covered already in literature [17][21][22] and a diverse range of solutions and frameworks [21][25][15][26][27] already exist to the problem each with its own pro's and con's for particular use-cases.

Instantaneous mixtures An instantaneous mixture is one where the observed signals are a combination of the source signals at the exact same time when measured.

Definition B.4: Instantaneous Mixture

For an **instantaneous mixture model** the set of **output signals** $\mathbf{x}(t)$ is dependent only on the source signals $\mathbf{s}(t)$ at the same sampling index:

$$\mathbf{x}(t) = \mathcal{A}\left(\mathbf{s}(t)\right). \tag{B-4}$$

Nonlinear models In contrast to the linear model, the nonlinear mixture models have not been as thoroughly researched yet mainly due to the increase in problem complexity and the wide range of possible nonlinearites. However, motivated by their associated application fields, advances in linear-quadratic [112][113] and post-nonlinear mixture models [114][115] have been made the past 2 decades and even attempts at generalizing the nonlinear framework have been made [17][21][116][117].

Convolutive mixtures In contrast to instantaneous mixtures, convolutive mixtures are dependent on values at multiple sampling indices. In definition **??** the linear convolutive model is defined.

Definition B.5: Convolutive Mixture

For a linear **convolutive mixture model** the set of **output signals** $\mathbf{x}(t)$ is dependent not only on the current value at index t of the **source signals** $\mathbf{s}(t)$, but on past values up to $t - K$ too. This is denoted as:

$$\mathbf{x}(t) = \sum_{j=1}^{N} \sum_{k=0}^{K} h_{ij}(k) \cdot s_j(t-k), \tag{B-5}$$

where K denotes the amount of past samples and $h_{ij}(k)$ represent the coefficients of the Multi-Input-Multi-Output (MIMO) linear mixing filters [5].

General approaches for convolutive mixtures have already been covered in literature [118][17] although most recent works focus on application specific cases [18][19].

Combinations of the described models exist as for example the complex post-nonlinear convolutive model [119] which apply a nonlinear distortion on the convolutive model of B.5. However, the use of such complex models is highly unconventional and is predominantly seen in cases when all other less complex models fail.

B-3 Computational cost of cumulant tensor

Step	Computational cost	Remark
$\mathbb{E}\left[x_{i_1}x_{i_2}x_{i_3}x_{i_4}\right] = \frac{1}{I}\cdot\left(\sum_{l=1}^{I}\prod_{k=1}^{4}x_{i_k}(l)\right)$	$O(3IP^4) \rightarrow O(IP^4)$	-
$\mathbb{E}\left[x_{i_m}x_{i_n}\right] = \frac{1}{I}\cdot\left(\sum_{l=1}^{I}x_{i_m}(l)x_{i_n}(l)\right) \quad \forall m,n \in \{1,2,3,4\}$	$O(IP^2)$	zero mean
$\mathbb{E}\left[x_{i_m}x_{i_n}\right] \circ \mathbb{E}\left[x_{i_m}x_{i_n}\right]$	$O(P^4)$	zero mean
total asymptotic:	$O(IP^4)$	

B-4 Derivation of complexity COM2 algorithm

Step	Computational cost	Remark
Polynomial root finding	$O(1)$	
Accumulation of rotations	$O(P^3)$	
Tensor update	$O(4P^5)$	
Final unmixing	$O(IP^2)$	
Total asymptotic cost:	$O(P^5)$	

B-5 Binomial coefficients

The binomial coefficients denoted as $\begin{pmatrix} n \\ k \end{pmatrix}$, represent the coefficients of the x^k terms in the polynomial expansion of the binomial power $(x+1)^n$, and is computed as $\begin{pmatrix} n \\ k \end{pmatrix} = \frac{n!}{k!(n-k)!}$.

Appendix C

Tensor decomposition for ICA

This appendix chapter contains the derivation of computational complexities, referenced algorithms, additional numerical results, the derivation of the implicit fixed-point CPD cost function and guidelines for use of our presented methods with other statistical tensors. Furthermore, one of our earliest attempts of integrating a rank-one constraint on tensorial data into the first-order implicit Canonical Polyadic Decomposition (CPD) problem is shown. We wrote out the full problem and its gradients but abandoned it early on during the making of this thesis as it was deemed to be to far out of the scope of this thesis.

C-1 HOSVD

Algorithm 7 shows how to compute the Higher-Order Singular Value Decomposition (HOSVD) of a tensor in pseudocode. The columns of the factor matrices are the left singular vectors of the subspaces of $\mathcal{X}$.

Algorithm 7 HOSVD

Require: N-dimensional tensor:$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$

1: **procedure** HOSVD($\mathcal{X}$)
2: **for** $n = 1, \ldots, N$ **do**
3: $\mathbf{U}^{(n)} \leftarrow$`left singular vectors of` $\mathbf{X}_{(n)}$
4: **end for**
5: $\mathcal{G} \leftarrow \mathcal{X} \times_1 \mathbf{U}^{(1)T} \times_2 \mathbf{U}^{(2)T} \ldots \times_N \mathbf{U}^{(N)T}$
6: **return** $\mathcal{G}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}$
7: **end procedure**

C-2 Time complexity implicit HOSVD

Step	Computational cost	Remark
Initial SVD	$O(P^3)$	
cost per SVD-update	$O(P^2(\log(P))^2))$	
Slice computation cost	$O(I + IP + IP^2)$	
Total cost:	$O(P(P-1)(I + IP + P^3(\log(P)^2)))$ + $O(P^3 + \frac{IP^3(P+1)}{2})$	Exploiting slice symmetry

C-3 SVD-update lemma

The core Singular-Value-Decomposition (SVD)-update method utilized by the Iterative-implicit-HOSVD algorithm is presented in [66]. To update an already existing SVD of square matrix $\mathbf{M}_1 = \mathbf{U}\Sigma\mathbf{V}^T$ with vector a $\mathbf{z}$, the algorithm exploits the properties in lemma C.1 shown below.

Lemma C.1: [120]

Let $\mathbf{U}\Sigma\mathbf{V}^T$ be the SVD of $\mathbf{M} \in \mathbb{R}^{P\times P}$ and let $\mathbf{U}'\begin{bmatrix} \Sigma' & 0 \end{bmatrix}^T\mathbf{V}'^T$ be the SVD of $\mathbf{M}' = \begin{bmatrix} \Sigma & \mathbf{z}^T \end{bmatrix}^T \in \mathbb{R}^{P+1\times P}$ with:

$$\begin{aligned} \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_P), \quad \Sigma' = \text{diag}(\sigma'_1, \ldots, \sigma'_P \\ \mathbf{U}' = \begin{bmatrix} \mathbf{u}'_1 & \ldots & \mathbf{u}'_P & \mathbf{u}'_{P+1} \end{bmatrix}, \quad \mathbf{V}' = \begin{bmatrix} \mathbf{v}'_1 & \ldots & \mathbf{v}'_P \end{bmatrix} \end{aligned} \tag{C-1}$$

where $0 \leq \sigma_1 \leq \ldots \leq \sigma_P$ and $0 \leq \sigma'_1 \leq \ldots \leq \sigma'_P$.
Then

$$\mathbf{M}'^T\mathbf{M}' = \Sigma^2 + zz^T = \mathbf{V}'\Sigma'^2\mathbf{V}'^T. \tag{C-2}$$

is the eigendecomposition of $\mathbf{M}'^T\mathbf{M}'$ with the corresponding secular equation:

$$\mathcal{F}(\sigma') = 1 + \sum_{i=1}^{n} \frac{z_i^2}{\sigma_i^2 - \sigma'^2} = 0 \tag{C-3}$$

and the singular values $\{\sigma_i\}_{i=1}^P$ satisfy the interlacing property:

$$0 \leq \sigma_1 \leq \sigma'_1 \leq \ldots \leq \sigma_P \leq \sigma'_P \leq \sigma_P + ||\mathbf{z}||_2. \tag{C-4}$$

The corresponding singular vectors satisfy:

$$\begin{aligned} \mathbf{u}'_j &= \left(\frac{\sigma_1 z_1}{\sigma_1^2 - \sigma'^2_j}, \ldots, \frac{\sigma_P z_P}{\sigma_P^2 - \sigma'^2_j}, -1\right)^T \Big/ \sqrt{1 + \sum_{i=1}^{P} \frac{\sigma_i^2 z_i^2}{\left(\sigma_i^2 - \sigma'^2_j\right)^2}}, \\ \mathbf{u}'_{P+1} &= \left(\frac{z_1}{\sigma_1}, \ldots, \frac{z_P}{\sigma_P}, -1\right)^T \Big/ \sqrt{1 + \sum_{i=1}^{P} \frac{z_i^2}{\sigma_i^2}}, \\ \mathbf{v}'_j &= \left(\frac{z_1}{\sigma_1^2 - \sigma'^2_j}, \ldots, \frac{z_P}{\sigma_P^2 - \sigma'^2_j}\right)^T \Big/ \sqrt{\sum_{i=1}^{P} \frac{z_i^2}{\left(\sigma_i^2 - \sigma'^2_j\right)^2}}, \end{aligned} \tag{C-5}$$

where $j = 1, \ldots, P$.

Lemma C.1 shows that the singular values of an appended matrix can be updated by finding the roots σ' of the secular equation (C-2) [1]. Afterwards, the updated singular vectors can

[1] The secular equation, also known as the characteristic polynomial of a square matrix, is a polynomial which has the eigenvalues of the square matrix as its roots.

be easily computed using the newly found singular values. The roots of the secular function are found using the Fast Multipole method (FMM) proposed in [67][68]. For more in-depth information on the FMM the reader is directed to said references. The used implementation [2] of this method contains several alterations and additional techniques presented in [70] and [71] which allow for a faster computation of sequential updates at a computational cost of $O(P^2(\log(P)^2)$ per update.

The algorithm is initialized with the SVD of the first frontal slice $\mathcal{X}(:,:,1)$ after which the SVD is updated $P(P-1)$ times with the $P(P-1)$ remaining columns. The matrix $\mathbf{V}^T$ containing the right singular vectors increases in size according to the size of the appended decomposed matrix. However, the equations (C-5) in lemma C.1 show that only the previous Σ and updated singular values Σ' are needed to update the singular vectors which results in only the singular values needed to be stored at a cost of $O(P)$.

[2]The implementation together with associated literature can be found in [69].

C-4 QR: Householder reflection matrices

Below the House-holder reflection matrices algorithm for performing QR-decomposition is shown.

Algorithm 8 Householder reflections QR

Require: Matrix: $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_N \end{bmatrix} \in \mathbb{R}^{M \times N}$ with $M \geq N$

1: **procedure** HHR-QR($\mathbf{A}$)
2: $\mathbf{A}' = \mathbf{A}$
3: **for** $m = 1, \dots, t = \min(M-1, N)$ **do**
4: $p = 1$
5: $|\alpha| = ||\mathbf{a}'_m||$, $\text{sign}(\alpha) \leftarrow -\text{sign}(a_m^p)$, ${a'_m}^p$ is pivot point in vector $\mathbf{a}_m$
6: $\mathbf{e}_m = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}^T \in \mathbb{R}^{M+1-m}$
7: $\mathbf{u} = \mathbf{a}'_m - \alpha \mathbf{e}_m$
8: $\mathbf{v} = \frac{\mathbf{u}}{||\mathbf{u}||}$
9: $\mathbf{Q}'_m = \mathbf{I}_{M+1-m} - 2\mathbf{v}\mathbf{v}^T$
10: $\mathbf{R}' = \mathbf{Q}'_m \mathbf{A}' = \begin{bmatrix} \alpha_m & \star & \dots & \star \\ 0 & & & \\ \vdots & & \mathbf{A}' & \\ 0 & & & \end{bmatrix} \in \mathbb{R}^{M+1-m \times N+1-m}$, $\mathbf{A}' \in \mathbb{R}^{M-m \times N-m}$
11: **if** $m = 1$ **then**
12: $\mathbf{Q}_1 = \mathbf{Q}'$
13: **else if** $m \geq 2$ **then**
14: $\mathbf{Q}_m = \begin{bmatrix} I_{m-1} & 0 \\ 0 & \mathbf{Q}_m \end{bmatrix} \in \mathbb{R}^{M \times M}$
15: **end if**
16: **end for**
17: $\mathbf{Q} = \mathbf{Q}_1^T \mathbf{Q}_2^T \cdots \mathbf{Q}_t^T$
18: $\mathbf{R} = \mathbf{Q}^T \mathbf{A}$
19: **return** $\mathbf{Q}$, $\mathbf{R}$
20: **end procedure**

C-5 Comparison of QRT with QR-algorithm on a symmetric matrix

Figure C-1 shows the behavior of the entries on the diagonal of an outer slice of a symmetric tensor when applying the QRT algorithm. The absolute values are taken. These are the exact same values from the example in the main text.

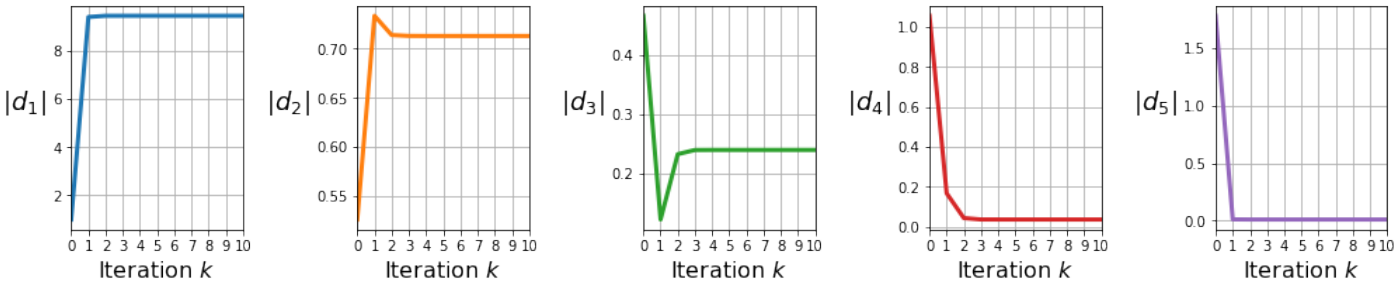


Figure C-1: Absolute values of $\mathbf{d}$: $|d_i| \quad \forall i = 1, 2, 3, 4, 5$ during the QRT procedure on the top outer-shell. Only the first 10 iterations are shown.

The behavior of these values shares a lot of similarity with the behavior of the diagonal entries of a matrix during the QR-algorithm. For example, let the matrix $\mathbf{M} \in \mathbb{R}^{5\times5}$ be sampled from a normal distribution with variance 1 and multiplied with itself to make it symmetrical:

$$\mathbf{M}^T\mathbf{M} = \begin{bmatrix} 10.717 & -1.183 & -0.2913 & 6.071 & -3.122 \\ -1.183 & 5.837 & 1.613 & 0.9582 & 1.990 \\ -0.2913 & 1.613 & 2.349 & 1.596 & 5.241 \\ 6.071 & 0.9582 & 1.596 & 5.696 & 2.245 \\ -3.122 & 1.990 & 5.241 & 2.245 & 9.735 \end{bmatrix}. \tag{C-6}$$

The absolute values on the diagonal of $\mathbf{M}^T\mathbf{M}$ during the QR-algorithm are shown in Figure C-2. We state for clarity that the final resulting values on the diagonalof the outer-slices are not equivalent to its eigenvalues.

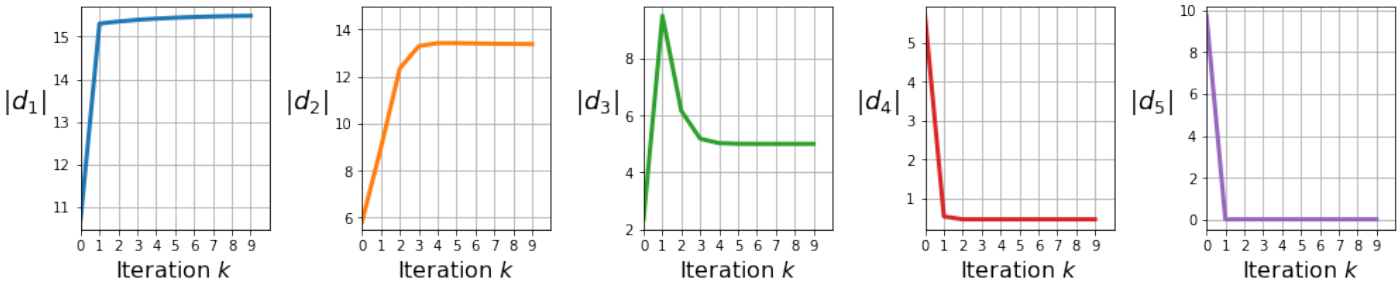


Figure C-2: Absolute values of $\mathbf{d}$: $|d_i| \quad \forall i = 1, 2, 3, 4, 5$ during the QR-algorithm on a symmetric matrix $\mathbf{M}$.

C-6 Proof of lemma identity outer product transformation

In this section we present the proof of the following lemma used in the main text.

Lemma C.2: Identity outer-product transformation

The mode-n product of the **outer-product** of 2 identity matrices with the matrix $\mathbf{M}$ is equivalent to multiplying the n'th dimension in the outer-product sequence with $\mathbf{M}$:

$$\begin{aligned}(\mathbf{I}_P \circ \mathbf{I}_P) \times_1 \mathbf{M} &= (\mathbf{M}\mathbf{I}_P) \circ (\mathbf{I}_P)\,, \quad (\mathbf{I}_P \circ \mathbf{I}_P) \times_2 \mathbf{M} \quad = \left(\mathbf{I}_P \mathbf{M}^T\right) \circ (\mathbf{I}_P) \\ (\mathbf{I}_P \circ \mathbf{I}_P) \times_3 \mathbf{M} &= (\mathbf{I}_P) \circ (\mathbf{M}\mathbf{I}_P)\,, \quad (\mathbf{I}_P \circ \mathbf{I}_P) \times_4 \mathbf{M} \quad = (\mathbf{I}_P) \circ \left(\mathbf{I}_P \mathbf{M}^T\right).\end{aligned} \tag{C-7}$$

Using the definition of tensor indices of a sequence of outer-products we define the following equality:

$$(\mathbf{I}_P \circ \mathbf{I}_P)\,(p_1, p_2, p_3, p_4) = (\mathbf{I}_P(p_1, p_2) \circ \mathbf{I}_P(p_3, p_4))\,. \tag{C-8}$$

Next we use the index notation from the mode-n matricization with $\mathbf{I}_P \in \mathbb{R}^{P \times P}$:

$$(\mathbf{I}_P \circ \mathbf{I}_P)\,(p_1, p_2, p_3, p_4) = (\mathbf{I}_P \circ \mathbf{I}_P)_{(1)}\,(p_1, (p_4 - 1) \cdot P^2 + (p_3 - 1) \cdot P + p_2). \tag{C-9}$$

Following the index notations from above the mode-1 matricization can be fully written out as:

$$(\mathbf{I}_P \circ \mathbf{I}_P)_{(1)} = \begin{bmatrix} \mathbf{I}_P \cdot \mathbf{I}_P(1,1) & \dots & \mathbf{I}_P \cdot \mathbf{I}_P(P,1) & \mathbf{I}_P \cdot \mathbf{I}_P(1,2) & \dots & \mathbf{I}_P \cdot \mathbf{I}_P(P,P) \end{bmatrix}. \tag{C-10}$$

Using the mode-n product definition:

$$\begin{aligned}&(\mathbf{I}_P \circ \mathbf{I}_P) \times_1 \mathbf{M} = \mathbf{M}\,(\mathbf{I}_P \circ \mathbf{I}_P)_{(1)} \\ &= \mathbf{M} \begin{bmatrix} \mathbf{I}_P \cdot \mathbf{I}_P(1,1) & \dots & \mathbf{I}_P \cdot \mathbf{I}_P(P,1) & \mathbf{I}_P \cdot \mathbf{I}_P(1,2) & \dots & \mathbf{I}_P \cdot \mathbf{I}_P(P,P) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{M}\mathbf{I}_P \cdot \mathbf{I}_P(1,1) & \dots & \mathbf{M}\mathbf{I}_P \cdot \mathbf{I}_P(P,1) & \mathbf{M}\mathbf{I}_P \cdot \mathbf{I}_P(1,2) & \dots & \mathbf{M}\mathbf{I}_P \cdot \mathbf{I}_P(P,P) \end{bmatrix}\end{aligned} \tag{C-11}$$

The last line of (C-12) can be reversed with the mode-1 matricization definition in (C-9) and results in:

$$\begin{aligned}&\begin{bmatrix} \mathbf{M}\mathbf{I}_P \cdot \mathbf{I}_P(1,1) & \dots & \mathbf{M}\mathbf{I}_P \cdot \mathbf{I}_P(P,1) & \mathbf{M}\mathbf{I}_P \cdot \mathbf{I}_P(1,2) & \dots & \mathbf{M}\mathbf{I}_P \cdot \mathbf{I}_P(P,P) \end{bmatrix} \\ &= (\mathbf{I}_P \circ \mathbf{I}_P) \times_1 \mathbf{M} = (\mathbf{M}\mathbf{I}_P) \circ (\mathbf{I}_P)\,,\end{aligned} \tag{C-12}$$

meaning that we have proven:

$$(\mathbf{I}_P \circ \mathbf{I}_P) \times_1 \mathbf{M} = (\mathbf{M}\mathbf{I}_P) \circ (\mathbf{I}_P)\,. \tag{C-13}$$

For all other modes the same reasoning can be applied. This is left to the reader.

C-7 CP-GEVD

This section shows the Canonical-Polyadic-Generalized-Eigenvalue-decomposition (CP-GEVD) algorithm presented in [99].

Algorithm 9 CPD-GEVD

Require: N-dimensional tensor: $\mathcal{T} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$, rank: R, machine epsilon: ϵ

procedure CPD-GEVD($\mathcal{T}$)
  HOSVD of $\mathcal{T} \rightarrow [\![\mathcal{G}; \mathbf{U}^1, \ldots, \mathbf{U}^N]\!]$, store mode-$n$ singular values as: $\mathbf{S}^n$
  for $n = 1, \ldots, N$ **do**
    $\delta_n = \max(P_n, \frac{\prod_{i=1}^{N} P_i}{P_n}) \cdot \epsilon$
    $P'_n = \text{Column-rank-truncation}\,(\mathbf{S}^n, \delta_n)$
  end for
  Sort on descending order: $(P'_1, \ldots, P'_N) \leftarrow \sigma\,(P'_1, \ldots, P'_N)$, permutation indices: σ
  permute: $\mathcal{G}(i_1, \ldots, i_N) \leftarrow \mathcal{G}\sigma(i_1, \ldots, i_N), \quad \left(\mathbf{U}^1, \ldots, \mathbf{U}^N\right) \leftarrow \sigma\left(\mathbf{U}^1, \ldots, \mathbf{U}^N\right)$
  $P'_i \leftarrow \min\,(P'_i, R) \quad \forall i = \{1, 2\}, \quad P'_2 = 2, \quad P'_j = 1 \quad \forall i = \{3, \ldots, N\}$
  Truncate: $\mathbf{U}^{1\prime} = \mathbf{U}^1(:, 1 : P'_1)$
  create slices:
    $\mathbf{G}_1 = \mathcal{G}(1 : P'_1, 1 : P'_2, 1, P'_4 \ldots, P'_N), \quad \mathbf{G}_2 = \mathcal{G}(1 : P'_1, 1 : P'_2, 2, P'_4 \ldots, P'_N)$
  GEVD of pencil: $(\mathbf{G}_1, \mathbf{G}_2) \rightarrow \boldsymbol{\lambda}, \mathbf{V}$
  Matricize: $\mathcal{T} \rightarrow \mathbf{T}_{(1)}$
  Project inwards: $\mathbf{X} = \mathbf{T}_{(1)}^T \mathbf{U}^{1\prime} \mathbf{V}$
  for $r = 1, \ldots, R$ **do**
    $\mathcal{X} \leftarrow \text{reshape}(\mathbf{X}(:, r), [P_2, \ldots, P_N])$
    HOSVD of $\mathcal{X} \rightarrow [\![\mathcal{G}_{\mathcal{X}}; \mathbf{U}^1_{\mathcal{X}}, \ldots, \mathbf{U}^{N-1}_{\mathcal{X}}]\!]$
    $\mathbf{U}^i_{out}(:, r) = \mathbf{U}^{i-1}_{\mathcal{X}}(:, 1) \quad \forall i \in \{2, \ldots, N\}$
  end for
  $\mathbf{K} = \mathbf{U}^N_{out} \odot \mathbf{U}^{N-1}_{out} \odot \ldots \odot \mathbf{U}^2_{out}$
  $\mathbf{U}^1_{out} = \mathbf{T}_{(1)} \left(\mathbf{K}^T\right)^{\dagger}$
  Normalize factors:
  for $r = 1, \ldots, R$ **do**
    $\mathbf{U}^i_{out}(:, r) \leftarrow \frac{\mathbf{U}^i_{out}(:,r)}{||\mathbf{U}^i_{out}(:,r)||_2^2} \cdot \sqrt[N]{||\mathbf{U}^1_{out}(:, r)||_2^2 \cdot \ldots \cdot ||\mathbf{U}^N_{out}(:, r)||_2^2} \quad \forall i = \{1, 2, \ldots, N\}$
  end for
  return $[\![\mathcal{G}; \mathbf{U}^1_{out}, \ldots, \mathbf{U}^N_{out}]\!]$
end procedure

C-8 Derivation of full fourth-order cumulant tensor expression

$$\mathbb{E}\left[x_{i_1}x_{i_2}x_{i_3}x_{i_4}\right] = \frac{1}{I}\cdot\left(\sum_{l=1}^{I}\prod_{k=1}^{4}x_{i_k}(l)\right) \xrightarrow{\text{Full tensor}} \mathcal{M}_{\mathbf{x}}^{(4)} = \mathbb{E}\left[\mathbf{x}^{\otimes 4}\right] = \frac{1}{I}\cdot\left(\sum_{l=1}^{I}\mathbf{x}(\ell)^{\otimes 4}\right) \tag{C-14}$$

$$\mathbb{E}\left[x_{i_m}x_{i_n}\right] = \frac{1}{I}\cdot\left(\sum_{l=1}^{I}x_{i_m}(l)x_{i_n}(l)\right) \xrightarrow{\text{Full tensor}} \mathbf{C}_{\mathbf{x}}^{(2)} = \mathbb{E}\left[\mathbf{x}^{\otimes 2}\right] = \frac{1}{I}\cdot\left(\sum_{l=1}^{I}\mathbf{x}(\ell)^{\otimes 2}\right) \tag{C-15}$$

$$\mathbb{E}\left[x_{i_1}x_{i_2}\right]\mathbb{E}\left[x_{i_3}x_{i_4}\right] \xrightarrow{\text{Full tensor}} \mathbb{E}\left[\mathbf{x}^{\otimes 2}\right]\circ\mathbb{E}\left[\mathbf{x}^{\otimes 2}\right] = \mathbf{C}_{\mathbf{x}}^{(2)}\circ\mathbf{C}_{\mathbf{x}}^{(2)} \tag{C-16}$$

$$\mathbb{E}\left[x_{i_1}x_{i_3}\right]\mathbb{E}\left[x_{i_2}x_{i_4}\right] \xrightarrow{\text{Full tensor}} \left(\mathbb{E}\left[\mathbf{x}^{\otimes 2}\right]\circ\mathbb{E}\left[\mathbf{x}^{\otimes 2}\right]\right)^{T\sigma} = \left(\mathbf{C}_{\mathbf{x}}^{(2)}\circ\mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma} \quad \text{with} \quad \sigma = \{1,3,2,4\} \tag{C-17}$$

$$\mathbb{E}\left[x_{i_1}x_{i_3}\right]\mathbb{E}\left[x_{i_2}x_{i_4}\right] \xrightarrow{\text{Full tensor}} \left(\mathbb{E}\left[\mathbf{x}^{\otimes 2}\right]\circ\mathbb{E}\left[\mathbf{x}^{\otimes 2}\right]\right)^{T\sigma} = \left(\mathbf{C}_{\mathbf{x}}^{(2)}\circ\mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma} \quad \text{with} \quad \sigma = \{1,4,3,2\} \tag{C-18}$$

C-9 Derivation of computational complexity of cumulant TTSV data form

The derivaiton of the computational complexity of:

$$\mathbf{y}_r = \mathcal{C}^{(4)}\mathbf{u}_r^4 = \mathbf{X}\mathbf{D}_I\left[\mathbf{X}^T\mathbf{u}_r\right]^3 - 3\left(\mathbf{z}^T\left[\mathbf{D}_I\right]^2\mathbf{z}\right)\mathbf{X}\mathbf{z} \quad \text{with} \quad \mathbf{z} = \mathbf{X}^T\mathbf{u}, \tag{C-19}$$

is shown in the table below. Remember, the steps are performed successively so all terms in the rows above have already been computed.

Step	Computational cost	Remark
$\mathbf{z} = \mathbf{X}^T\mathbf{u}$	$O(IP)$	
$\mathbf{X}\mathbf{z}$	$O(PI)$	
$\left[\mathbf{D}_I\right]^2$	$O(I)$	
$\mathbf{z}^T\left[\mathbf{D}_I\right]^2$	$O(I^2)$	
$\mathbf{z}^T\left[\mathbf{D}_I\right]^2\mathbf{z}$	$O(I)$	
$\left(\mathbf{z}^T\left[\mathbf{D}_I\right]^2\mathbf{z}\right)\mathbf{X}\mathbf{z}$	$O(P)$	
Total asymptotic cost:	$O(IP)$	

C-10 Derivation of cost function for implicit CPD of cumulant tensor

$$\min_{\boldsymbol{\lambda},\mathbf{U}} f(\boldsymbol{\lambda},\mathbf{U}) \equiv \|\mathcal{C}_{\mathbf{x}}^{(4)} - \hat{\mathcal{C}}_{\mathbf{x}}^{(4)}\|^2 \quad \text{s.t.} \quad \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \equiv \sum_{r=1}^{R} \lambda_r \mathbf{u}_r^{\circ 4} \tag{C-20}$$

$$\|\mathcal{C}_{\mathbf{x}}^{(4)} - \hat{\mathcal{C}}_{\mathbf{x}}^{(4)}\|^2 = \|\mathcal{C}_{\mathbf{x}}^{(4)}\|^2 + \|\hat{\mathcal{C}}_{\mathbf{x}}^{(4)}\|^2 - 2\langle \mathcal{C}_{\mathbf{x}}^{(4)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle \tag{C-21}$$

$$\begin{aligned} \|\hat{\mathcal{C}}_{\mathbf{x}}^{(4)}\|^2 = \langle \hat{\mathcal{C}}_{\mathbf{x}}^{(4)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle &= \sum_{r=1}^{R}\sum_{s=1}^{R} \lambda_r \lambda_s \left\langle \mathbf{u}_r^{\circ 4}, \mathbf{u}_s^{\circ 4} \right\rangle = \sum_{r=1}^{R}\sum_{s=1}^{R} \lambda_r \lambda_s \langle \mathbf{u}_r, \mathbf{u}_s \rangle^4 \\ &= \sum_{r=1}^{R}\sum_{s=1}^{R} \lambda_r \left[\mathbf{u}_r^T \mathbf{u}_s\right]^4 \lambda_s = \boldsymbol{\lambda}^T \left[\mathbf{U}^T \mathbf{U}\right]^4 \boldsymbol{\lambda} \end{aligned} \tag{C-22}$$

$$\begin{aligned} \|\mathcal{C}_{\mathbf{x}}^{(4)}\|^2 = \langle \mathcal{C}_{\mathbf{x}}^{(4)}, \mathcal{C}_{\mathbf{x}}^{(4)} \rangle = \\ &\langle \mathcal{M}_{\mathbf{x}}^{(4)}, \mathcal{M}_{\mathbf{x}}^{(4)} \rangle - \langle \mathcal{M}_{\mathbf{x}}^{(4)}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle \\ &- \langle \mathcal{M}_{\mathbf{x}}^{(4)}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1} \rangle - \langle \mathcal{M}_{\mathbf{x}}^{(4)}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2} \rangle \\ &- \langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \mathcal{M}_{\mathbf{x}}^{(4)} \rangle + \langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle \\ &+ \langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1} \rangle + \langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2} \rangle \\ &- \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1}, \mathcal{M}_{\mathbf{x}}^{(4)} \rangle + \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle \\ &+ \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1} \rangle + \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2} \rangle \\ &- \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2}, \mathcal{M}_{\mathbf{x}}^{(4)} \rangle + \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle \\ &+ \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1} \rangle + \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2} \rangle \end{aligned} \tag{C-23}$$

Due to symmetry:

$$\begin{aligned} &\langle \mathcal{M}_{\mathbf{x}}^{(4)}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle = \langle \mathcal{M}_{\mathbf{x}}^{(4)}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1} \rangle = \langle \mathcal{M}_{\mathbf{x}}^{(4)}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2} \rangle \\ &\langle \mathcal{M}_{\mathbf{x}}^{(4)}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle = \langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \mathcal{M}_{\mathbf{x}}^{(4)} \rangle \\ &\langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle = \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)T\sigma_1} \rangle \\ &= \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2} \rangle \\ &\langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle = \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_1}, \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2} \rangle \\ &= \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}\right)^{T\sigma_2}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle \end{aligned} \tag{C-24}$$

$$\begin{aligned}\|\mathcal{C}_{\mathbf{x}}^{(4)}\|^2 = \langle \mathcal{C}_{\mathbf{x}}^{(4)}, \mathcal{C}_{\mathbf{x}}^{(4)} \rangle = \\ \langle \mathcal{M}_{\mathbf{x}}^{(4)}, \mathcal{M}_{\mathbf{x}}^{(4)} \rangle - 6\langle \mathcal{M}_{\mathbf{x}}^{(4)}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle \\ + 3\langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle \\ + 6\langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \right)^{T\sigma_1}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle\end{aligned} \tag{C-25}$$

$$\mathbf{1} = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T \in \mathbb{R}^I \tag{C-26}$$

$$\begin{aligned}\langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \right)^{T\sigma_1}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle &= \frac{1}{I^4} \sum_{i=1}^{I} \sum_{j=1}^{I} \sum_{k=1}^{I} \sum_{l=1}^{I} \langle \mathbf{x}_i, \mathbf{x}_k \rangle \cdot \langle \mathbf{x}_j, \mathbf{x}_k \rangle \cdot \langle \mathbf{x}_i, \mathbf{x}_l \rangle \cdot \langle \mathbf{x}_j, \mathbf{x}_l \rangle \\ &= \frac{1}{I^4} \mathbf{1}^T \left[\mathbf{X}^T \mathbf{X} \mathbf{X}^T \mathbf{X} \right]^2 \mathbf{1}\end{aligned} \tag{C-27}$$

$$\begin{aligned}\langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle &= \frac{1}{I^4} \sum_{\ell_1=1}^{I} \sum_{k_1=1}^{I} \sum_{\ell_2=1}^{I} \sum_{k_2=1}^{I} \langle \mathbf{x}_{\ell_1}^{\circ 2} \circ \mathbf{x}_{k_1}^{\circ 2}, \mathbf{x}_{\ell_2}^{\circ 2} \circ \mathbf{x}_{k_2}^{\circ 2} \rangle \\ &= \frac{1}{I^4} \sum_{\ell_1=1}^{I} \sum_{k_1=1}^{I} \sum_{\ell_2=1}^{I} \sum_{k_2=1}^{I} \left[\mathbf{x}_{\ell_1}^T \mathbf{x}_{\ell_2} \right]^2 \cdot \left[\mathbf{x}_{k_1}^T \mathbf{x}_{k_2} \right]^2 \\ &= \frac{1}{I} \cdot \mathbf{1}^T \left[\mathbf{X}^T \mathbf{X} \right]^2 \mathbf{1} \cdot \frac{1}{I} \cdot \frac{1}{I} \cdot \mathbf{1}^T \left[\mathbf{X}^T \mathbf{X} \right]^2 \mathbf{1} \cdot \frac{1}{I}, \quad \mathbf{1} \in \mathbb{R}^I\end{aligned} \tag{C-28}$$

$$\begin{aligned}\langle \mathcal{M}_{\mathbf{x}}^{(4)}, \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \rangle &= \frac{1}{I^3} \sum_{i=1}^{I} \sum_{j=1}^{I} \sum_{k=1}^{I} \langle \mathbf{x}_i, \mathbf{x}_j \rangle^2 \cdot \langle \mathbf{x}_j, \mathbf{x}_j \rangle^2 \\ &= \frac{1}{I^3} \mathbf{1}^T \left[\mathbf{X}^T \mathbf{X} \right]^2 \cdot \left[\mathbf{X}^T \mathbf{X} \right]^2 \mathbf{1}\end{aligned} \tag{C-29}$$

$$\langle \mathcal{M}_{\mathbf{x}}^{(4)}, \mathcal{M}_{\mathbf{x}}^{(4)} \rangle = \frac{1}{I^2} \sum_{i=1}^{I} \sum_{j=1}^{I} \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \frac{1}{I} \cdot \mathbf{1}^T \left[\mathbf{X}^T \mathbf{X} \right]^4 \mathbf{1} \cdot \frac{1}{I}, \quad \mathbf{1} \in \mathbb{R}^I \tag{C-30}$$

$$\begin{aligned}\langle \mathcal{C}_{\mathbf{x}}^{(4)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle = \langle \mathcal{M}_{\mathbf{x}}^{(4)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle + \langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle + \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \right)^{T\sigma_1}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle \\ + \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \right)^{T\sigma_2}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle\end{aligned} \tag{C-31}$$

$$\langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle = \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \right)^{T\sigma_1}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle = \langle \left(\mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)} \right)^{T\sigma_2}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle \tag{C-32}$$

$$\langle \mathcal{C}_{\mathbf{x}}^{(4)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle = \langle \mathcal{M}_{\mathbf{x}}^{(4)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle - 3\langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle \tag{C-33}$$

$$\langle \mathcal{M}_{\mathbf{x}}^{(4)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle = \sum_{i=1}^{I} \sum_{r=1}^{R} \frac{1}{I} \left[\mathbf{x}_i^T \mathbf{u}_r \right]^4 \lambda_r = \frac{1}{I} \mathbf{1}^T \left[\mathbf{X}^T \mathbf{U} \right]^4 \boldsymbol{\lambda} \tag{C-34}$$

$$\begin{aligned}\langle \mathbf{C}_{\mathbf{x}}^{(2)} \circ \mathbf{C}_{\mathbf{x}}^{(2)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle &= \frac{1}{I^2} \sum_{i=1}^{I} \sum_{j=1}^{I} \sum_{r=1}^{R} \lambda_r \cdot \langle \mathbf{u}_r, \mathbf{x}_i \rangle^2 \cdot \langle \mathbf{u}_r, \mathbf{x}_j \rangle^2 \\ &= \frac{1}{I^2} \mathbf{1}^T \left[\mathbf{X}^T \mathbf{U} \right]^2 \mathrm{diag}_2(\boldsymbol{\lambda}) \left[\mathbf{U}^T \mathbf{X} \right]^2 \mathbf{1}\end{aligned} \tag{C-35}$$

Final cost function update is:

$$\begin{aligned}&\|\mathcal{C}_{\mathbf{x}}^{(4)} - \hat{\mathcal{C}}_{\mathbf{x}}^{(4)}\|^2 = \|\mathcal{C}_{\mathbf{x}}^{(4)}\|^2 + \|\hat{\mathcal{C}}_{\mathbf{x}}^{(4)}\|^2 - 2\langle \mathcal{C}_{\mathbf{x}}^{(4)}, \hat{\mathcal{C}}_{\mathbf{x}}^{(4)} \rangle \\ &= \frac{1}{I} \cdot \mathbf{1}^T \left[\mathbf{X}^T \mathbf{X} \right]^4 \mathbf{1} \cdot \frac{1}{I} - 6 \cdot \frac{1}{I^3} \mathbf{1}^T \left[\mathbf{X}^T \mathbf{X} \right]^2 \cdot \left[\mathbf{X}^T \mathbf{X} \right]^2 \mathbf{1} + 3 \cdot \left[\frac{1}{I} \cdot \mathbf{1}^T \left[\mathbf{X}^T \mathbf{X} \right]^2 \mathbf{1} \cdot \frac{1}{I} \right]^2 \\ &+ 6 \cdot \frac{1}{I^4} \mathbf{1}^T \left[\mathbf{X}^T \mathbf{X} \mathbf{X}^T \mathbf{X} \right] \mathbf{1} + \boldsymbol{\lambda}^T \left[\mathbf{U}^T \mathbf{U} \right]^4 \boldsymbol{\lambda} \\ &- 2 \cdot \left[\frac{1}{I} \mathbf{1}^T \left[\mathbf{X}^T \mathbf{U} \right]^4 \boldsymbol{\lambda} + 3 \cdot \frac{1}{I^2} \mathbf{1}^T \left[\mathbf{X}^T \mathbf{U} \right]^2 \mathrm{diag}_2(\boldsymbol{\lambda}) \left[\mathbf{U}^T \mathbf{X} \right]^2 \mathbf{1} \right]\end{aligned} \tag{C-36}$$

C-11 Additional low rank assumption for tensorial data

Tensor-based Blind Source Separation (BSS) methods rely on the assumption that patterns of interest are low in rank which means they can be approximated by decomposing the data tensor using any low-rank decomposition. Various works and frameworks have been presented on this topic [15][5][28][29][7].

Theorem C.1: Low-rank assumption

In this section an attempt at introducing a rank-one constraint on tensorial in combination with the first-order CPD problem for the cumulant tensor is shown.
Let $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$ denote an N-dimensional data array which consists of mixed components when vectorized are represented as $\mathbf{s}$. Under the assumption that said source components are **low in rank**, they can be approximated by decomposing the tensor into low-rank components.

Using theorem C.1, the source components of a mixture tensor which are assumed to be low in rank can be found using CPD for example. However, it is noted that when the assumption is incorrect tensor-based BSS methods fail completely. Alternatively, data can be low in rank but not in all modes. Therefore, inspection and pre-processing of the data are important steps one must consider when applying tensor-based BSS methods [5][15]. A different approach is to combine tensor-based BSS methods with the independence of assumption of Independent Component Analysis (ICA). This is presented in the following section.

C-11-1 Enforcing statistical independence on multi-dimensional data

Beckmann and Smith propose in [93] a method of analysing the components of multi-dimensional data by combining the independence and low-rank assumption. Their work is meant specifically for multi-subject or multi-session functional Magnetic Resonance Imaging (fMRI) data but can be generalized as follows.

The tensor-pICA algorithm constraints the data to be maximally non-Gaussian in selected modes. As an example we shall use a third-order tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$ which decomposed with Parallel Factor Analysis (PARAFAC) results in the following model:

$$\hat{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \quad \mathbf{a} \in \mathbb{R}^{P_1}, \mathbf{b} \in \mathbb{R}^{P_2}, \mathbf{c} \in \mathbb{R}^{P_3}. \tag{C-37}$$

Note that this is identical to a CPD with all weights $\lambda_r = 1$. The factors can be found by solving the following optimization problem:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} f(\mathbf{A},\mathbf{B},\mathbf{C}) \equiv \|\mathcal{X} - \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r\|_F^2 \tag{C-38}$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 & \ldots & \mathbf{a}_R \end{pmatrix}, \mathbf{B} = \begin{pmatrix} \mathbf{b}_1 & \ldots & \mathbf{b}_R \end{pmatrix}, \mathbf{C} = \begin{pmatrix} \mathbf{c}_1 & \ldots & \mathbf{c}_R \end{pmatrix}.$$

By matricizing $\mathbf{X}_{(n)}$ and the model from (C-37) along each of its modes the following matrix product expressions of are obtained:

$$\hat{\mathbf{X}}_{(1)}^T = (\mathbf{C} \odot \mathbf{B})\,\mathbf{A}^T, \quad \hat{\mathbf{X}}_{(2)}^T = (\mathbf{C} \odot \mathbf{A})\,\mathbf{B}^T, \quad \hat{\mathbf{X}}_{(3)}^T = (\mathbf{B} \odot \mathbf{A})\,\mathbf{C}^T. \tag{C-39}$$

By fixing any 2 of the matrices such as $\mathbf{B}$ and $\mathbf{C}$ the problem reduces to a linear least-squares for which the optimal solution is given by:

$$\mathbf{A}_{opt}^T = [(\mathbf{C} \odot \mathbf{B})]^\dagger\, \mathbf{X}_{(1)}^T \,^3 \tag{C-40}$$

where $[\mathbf{M}]^\dagger$ denotes the Moore-Penrose or better known as the pseudo-inverse of matrix $\mathbf{M}$. Alternating Least Squares (ALS) consists of cylically alternating the free matrix in the process described above. Observe that the equations from (C-44) are in fact linear mixing models with for example the mixing matrix being the Khatri-Rhao structure $(\mathbf{C} \odot \mathbf{B})$ and source components being $\mathbf{A}^T$. The tensor-pICA algorithm exploits this by first estimating $(\mathbf{C} \odot \mathbf{B})$ as a single mixing matrix with conventional ICA methods and afterwards decomposing it into $\mathbf{B}$ and $\mathbf{C}$. This can be done iteratively or by a fully converged ICA estimation which can be followed by a single decomposition. Either way, the consequent steps in the direction of independence and the direction of the Khatri-Rhao structure enforce the components to be statistically independent in the respective set of modes and can be applied to all modes.

The generalization of this method referred to as tensor Independent Component Analysis (tICA) from now is shown in definition C.1.

Definition C.1: tensor Independent Component Analysis

Tensor ICA is the process of unmixing tensorial data $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}$ into its rank-one source components and constraining the factors in the PARAFAC decomposition:

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{v}_{1,r} \circ \mathbf{v}_{2,r} \circ \ldots \circ \mathbf{v}_{N,r}, \quad \mathbf{v}_{i,r} \in \mathbb{R}^{P_i} \quad \forall i \in \{1, \ldots, N\} \tag{C-41}$$

to be **statistically independent** in mode n. This is done through decomposition of $\hat{\mathbf{M}}$ into $(\mathbf{V}_N \odot \ldots \odot \mathbf{V}_{n+1} \odot \mathbf{V}_{n-1} \odot \ldots \odot \mathbf{V}_1)$ after acquiring the linear mixing matrix estimate $\hat{\mathbf{M}}$ through ICA:

$$\mathbf{X}_{(n)}^T = \mathbf{M}\mathbf{V}_n^T \xrightarrow{\text{ICA}} \hat{\mathbf{M}}. \tag{C-42}$$

Although shown that the tICA method manages to obtain better results than just PARAFAC or ICA for cases such as with fMRI data [93], the algorithm is prone to failure since the independence step and Khatri-Rhao step have different objectives [121]. Based on [121], this can be addressed by combining the steps which is shown in the subsequent section for the previously presented symmetric first-order method.

[3] A more efficient version of this computation can be performed by rewriting the Khatri-Rhao pseudo inverse as: $\mathbf{A} = \mathbf{X}_{(1)}\,(\mathbf{C} \odot \mathbf{B})\left(\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B}\right)^\dagger$ for which the pseudo inverse of an $R \times R$ matrix has to be taken as opposed to $P_3P_2 \times R$.

Definition C.2: First-order optimization for symmetric CPD constraint to rank-one

For data tensor $\mathcal{X} \in \mathbb{R}^{P_1 \times P_2 \times \ldots P_N}$ PARAFAC model is:

$$\hat{\mathcal{X}} \approx \sum_{r=1}^{R} \mathbf{v}_{r,1} \circ \mathbf{v}_{r,2} \circ \ldots \circ \mathbf{v}_{r,N} = [\![\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_N]\!], \quad \mathbf{v}_{r,j} \in \mathbb{R}^{P_j} \quad \forall j \in \{1, \ldots, N\} \tag{C-43}$$

Reshape according to reshaping of data:

$$\hat{\mathbf{X}}_{(n)}^T = (\mathbf{V}_N \odot \ldots \odot \mathbf{V}_{n+1} \odot \mathbf{V}_{n-1} \odot \ldots \odot \mathbf{V}_1)\, \mathbf{V}_n^T = \mathbf{U}\mathbf{V}_n^T \tag{C-44}$$

with $P' = \prod_{\substack{i=1 \\ i \neq n}}^{N} P_i$, $\hat{\mathbf{X}}_{(n)} \in \mathbb{R}^{P_n \times P'}$, $\mathbf{U} \in \mathbb{R}^{P' \times R}$ and $\mathbf{V}_n \in \mathbb{R}^{P_n \times R}$.

Create cumulant tensor $\mathcal{C}_{(4)} \in \mathbb{R}^{[4,P']}$ from matricized data $\mathbf{X}_{(n)}^T$.
Following first-order optimization poblem to find independent components:

$$\min_{\boldsymbol{\lambda}, \mathbf{U}} f(\boldsymbol{\lambda}, \mathbf{U}) \equiv \|\mathcal{C}_{(4)} - \hat{\mathcal{C}}_{(4)}\|^2 \quad \text{s.t.} \quad \hat{\mathcal{C}}_{(4)} \equiv \sum_{r=1}^{R} \lambda_r \mathbf{u}_r^{\circ 4}, \tag{C-45}$$

$$\frac{\partial f}{\partial \lambda_r} = -2 \left[\mathcal{C}_{(4)} \mathbf{u}_r^4 - \sum_{k=1}^{R} \lambda_k \langle \mathbf{u}_r, \mathbf{u}_k \rangle^4 \right] \quad \text{and} \tag{C-46}$$

$$\frac{\partial f}{\partial \mathbf{u}_r} = -8\lambda_r \left[\mathcal{C}_{(4)} \mathbf{u}_r^3 - \sum_{k=1}^{R} \lambda_k \langle \mathbf{u}_r, \mathbf{u}_k \rangle^3 \mathbf{u}_k \right] \tag{C-47}$$

only now the factors are constrained to the following structure:

$$\begin{aligned} \mathbf{U} &= (\mathbf{V}_N \odot \ldots \odot \mathbf{V}_{n+1} \odot \mathbf{V}_{n-1} \odot \ldots \odot \mathbf{V}_1) \in \mathbb{R}^{P' \times R} \\ \mathbf{u}_r &= \mathbf{v}_{N,r} \otimes \ldots \otimes \mathbf{v}_{n+1,r} \otimes \mathbf{v}_{n-1,r} \otimes \ldots \otimes \mathbf{v}_{1,r} \in \mathbb{R}^{P'}. \end{aligned} \tag{C-48}$$

Now it can be written as a single optimization problem with the following adjusted gradients:

$$\frac{\partial f}{\partial \mathbf{v}_{j,r}} = \frac{\partial f}{\partial \mathbf{u}_r} \frac{\partial \mathbf{u}_r}{\partial \mathbf{v}_{j,r}} \quad \forall r \in \{1, \ldots, R\}, j \in \{1, \ldots, n-1, n+1, \ldots, N\}, \tag{C-49}$$

C-12 Higher-order statistical tensors in general

It is briefly discussed here how the presented algorithms can be used for Higher Order Statistics (HOS) tensors in general. The original first-order implicit CPD method was devised to work with any N'th-order moment tensor for a random vector $\mathbf{z} \in \mathbb{R}^P$ with I samples:

$$\mathcal{M}_{\mathbf{z}}^{(4)} = \sum_{i=1}^{I} \frac{1}{I} \mathbf{z}_i^{\circ N}. \tag{C-50}$$

The key to as why this is possible is the exploitation of the tensor structure. For the implicit HOSVD, implicit QRT and implicit CP-GEVD algorithms this can be done too. The key exploit was that frouth-order moment part of the cumulant tensors matricization has a simple Khatri-rhao structure:

$$\mathbf{M}^{(4)}_{(n),\mathbf{z}} = \frac{1}{I}\mathbf{Z}(\mathbf{Z} \odot \mathbf{Z} \odot \mathbf{Z})^T. \tag{C-51}$$

This same structure holds true for any N'th order moment tensor:

$$\mathbf{M}^{(N)}_{(n),\mathbf{z}} = \frac{1}{I}\mathbf{Z} \underbrace{(\mathbf{Z} \odot \ldots \odot \mathbf{Z})^T}_{N-2 \quad \text{Khatri-rao products}} . \tag{C-52}$$

For the fourth-order cumulant tensor the second-order part was simplified due to whitening. Cumulant tensors of order ≥ 5 have more complex structures such as for exmample the fifth-order cumulant:

$$\kappa_5(X) = \mathbb{E}\left((X - \mathbb{E}(X))^5\right) - 10\mathbb{E}\left((X - \mathbb{E}(X))^3\right)\mathbb{E}\left((X - \mathbb{E}(X))^2\right). \tag{C-53}$$

However, structurally these cumulants are always expressible in moment tensors of equivalent or lower order or in cumulant tensors of lower order. If the reader wants to apply the presented algorithms on a HOS tensor different than the cumulant tensor, the reader is advised to perform the following steps:

1. Write out full definition of the HOS quantity and its tensorized version with symmetric outer-products as in (C-50).
2. Inspect the simplification of terms due to whitening.
3. Write out the mode-n matricization of the used HOS tensor in an expression similar to (C-52) and check whether the indexing strategy of lemma 4.1 holds true.
4. Replace all instances of tensor slice computation in the desired algorithm with previously derived matricized expression.
5. Correct the algorithms such that for example the amount of mode-n products and index notation match tensor order.

There are a few big side notes. First of all, the desired HOS must have the multilinearity property 3.5 of the cumulant tensor too. Secondly, preliminary investigation of said HOS tensor's symmetry can lead to varying exploits which are in many cases tensor order dependent. Lastly, the added benefit of whitening can vary greatly amongst HOS tensors.

This goes to show that using such algorithms on HOS tensors in general requires customization and exploitation of that particular tensor's properties.

Appendix D

Experimental results

In this appendix chapter the process of determining the classification algorithm's hyper-parameters is shown. Next to this several additional results of the numerical Blind Source Separation (BSS) experiment are discussed which have not been used in the main text.

D-1 Classification algorithm and Tolerance determination

The algorithm below is used to determine whether a solution is considered as successful or failed.

In order to automate the process of determining whether a solution is correct or not, algorithm 10 in appendix D-1 was developed. The ICA Solution Sorting and Evalutation (ICA-SSE) algorithm first sorts the found solutions based on the order of maximum correlation they have to the source components and determines afterwards whether the estimated components match up with source components and sorts them again by looking for the minimal relative errors between the source components and the estimated components (lines 4 to 13). By first sorting on maximum correlation and afterwards on minimum error the probability of finding the correct matching pairs is increased. After sorting, the errors and correlations of the found components are compared with the tolerances $\mathbf{e}^{tol}$ and $\mathbf{c}^{tol}$ to determine whether these components are defined as correct.

Algorithm 10 ICA-SSE

Require: $\mathbf{S} = \begin{bmatrix} \mathbf{s}_1^T & \dots & \mathbf{s}_N^T \end{bmatrix}^T \in \mathbb{R}^{N \times I}$,$\hat{\mathbf{S}} = \begin{bmatrix} \hat{\mathbf{s}}_1^T & \dots & \hat{\mathbf{s}}_P^T \end{bmatrix}^T \in \mathbb{R}^{P \times I}$, Error tolerances: $\mathbf{e}^{tol}$, Correlation tolerances: $\mathbf{c}^{tol}$

procedure ICA-SSE($\mathbf{S}$,$\hat{\mathbf{S}}$)
    List to store indices of $\hat{\mathbf{S}}$ that match with a source component: L_{found}
    Normalize and center rows of $\mathbf{S}$ and $\hat{\mathbf{S}}$
    Sort based on correlation matrix: $L_{sorted} = \text{argmax}_{\text{col}}(\mathbf{C}_{\mathbf{S},\hat{\mathbf{S}}})$
    for $n = 1, \dots, N$ **do**
        for p iterate through L_{sorted} **do**
            if $p \in L_{found}$ **then**
                $\varepsilon_p = \infty$
            end if
            $\varepsilon_p = \min\left(\|\mathbf{s}_n - \hat{\mathbf{s}}_p\|_2, \|\mathbf{s}_n + \hat{\mathbf{s}}_p\|_2\right)$
            L_{signs} assign sign depending on: $\text{argmin}\left(\|\mathbf{s}_n - \hat{\mathbf{s}}_p\|_2, \|\mathbf{s}_n + \hat{\mathbf{s}}_p\|_2\right)$
        end for
        Add index of best matching solution to L_{found}: $\text{argmin}([\varepsilon_1, \dots \varepsilon_P])$
        Store error: $\mathbf{e}_n = \min([\varepsilon_1, \dots \varepsilon_P])$
    end for
    $\hat{\mathbf{S}}_{sorted} \leftarrow$ Sort $\hat{\mathbf{S}}$ in order of L_{found}
    Permute correlation matrix in order of $\hat{\mathbf{S}}_{sorted}$: $\mathbf{C}_{\mathbf{S},\hat{\mathbf{S}}_{sorted}} \leftarrow \mathbf{C}_{\mathbf{S},\hat{\mathbf{S}}}$
    for $n = 1, \dots, N$ **do**
        if $\mathbf{e}(n) > \mathbf{e}^{tol}(n)$ or $\mathbf{C}_{\mathbf{S},\hat{\mathbf{S}}_{sorted}}(n,n) < \mathbf{c}^{tol}(n)$ **then**
            Store index of failed component in: L_{failed}
        end if
    end for
    return $\mathbf{e}, L_{found}, L_{fail}, L_{signs}$
end procedure

The method how we determined the tolerances is explained here.

The tolerances are determined through an experiment were the first 25 solutions of each iterative algorithm (*FICA*,*QRT*,*I-CPD-G*,*I-CPD-FF*) which we deemed as correct are evaluated on their performance. With first 25 solutions we mean the first 25 solutions deemed correct as we reiterate through the test-set while every time increasing the seed with 1 starting at 0. Table D-1 shows the maximum error tolerances and minimum correlation tolerances found for which we deemed a solution to be correct. The values in red correspond to the algorithm and its solution number on the left. The values in black do not necessarily correspond to these algorithms.These values were found

		Maximum e or minimum c allowed tolerances							
Responsible Method	Seed	e_1	e_2	e_3	e_4	c_1	c_2	c_3	c_4
QRT	83	0.624	0.589	0.461	0.601	-	-	-	-
QRT	37	-	-	-	-	0.806	0.871	0.848	0.839

Table D-1: Maximum found error tolerances and minimum found correlation tolerances. In total 238 runs were needed to find 25 successful estimations for each algorithm at $P = 4$. The colors in red indicate the maximum or minimum value that was found. The algorithm in red on the left corresponds to this value. The values in black are not necessarily a result from that particular algorithm.

The final error tolerance values for all components are set to the highest value found and rounded upwards. For the correlation tolerance the lowest value found was taken and was rounded downwards. This results in the following final values:

$$\mathbf{e}^{tol} = [0.65, 0.65, 0.65, 0.65], \quad \mathbf{c}^{tol} = [0.8, 0.8, 0.8, 0.8]. \tag{D-1}$$

These tolerances can be considered as rather lenient. On top of that, setting the error tolerances higher than the found maxima and the correlation tolerances lower than the found minima seems counter intuitive as it suggests faulty solutions may be deemed correct. However, our reasoning behind this choice is 2-fold. Firstly, we do not know whether these are the true maxima and minima for which we would deem a solution to still be correct. As it is a subjective process there sill might exist a solution which we deem to be correct which has higher error values and lower correlation values. As we do not want to miss such possible solutions we set the error tolerances slightly higher and the correlation tolerances slightly lower than the found maxima and minima. Secondly, as will be clear from the following two examples, only one component has to fail to indicate whether a solution has failed. Whenever one or several estimated components are considered as "failed" they tend to have high errors > 0.8, low correlations < 0.6 or both. The other components tend to be the opposite by showing low errors < 0.5 and high correlations > 0.9. This means that the effect of slightly increasing the error tolerance and slightly decreasing the correlation tolerance will be that potential solutions with more extreme values will still be considered as successful while failed solutions will still be evaluated as failed.

An example of a correct set of estimated components for a data-set with 4 mixtures and 5000 observations can be seen in Figure D-1 and an example of a set that is considered as failed can be seen in Figure D-2. The tables in D-2 show the per estimated source component estimation error and correlations of both the successful and the failed case. Most of the components of the failed case have higher errors than the successful case. The fourth component of the failed case $\hat{s}_4$ is what caused this solution to be classified as incorrect due to its error and correlation. However, the solutions of both cases still show a lot of noise with the noise being differently distributed over the components per case. This suggests that by adding additional mixtures to the problem and estimating additional components in which the noise can be

maximized can lead to improved results. This is shown later on.

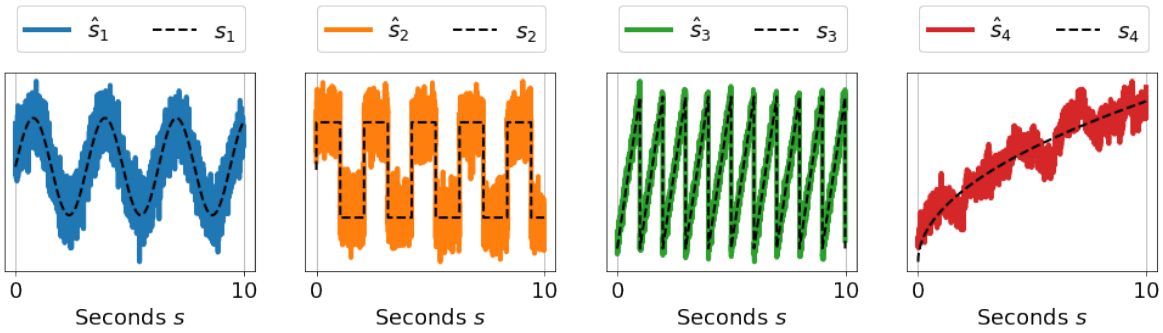


Figure D-1: Example of a correct set of estimated components from a data-set with $P = 6$ mixtures. The estimated components are corrected for sign and mean such that they align correctly with the source components shown in dashed black lines.

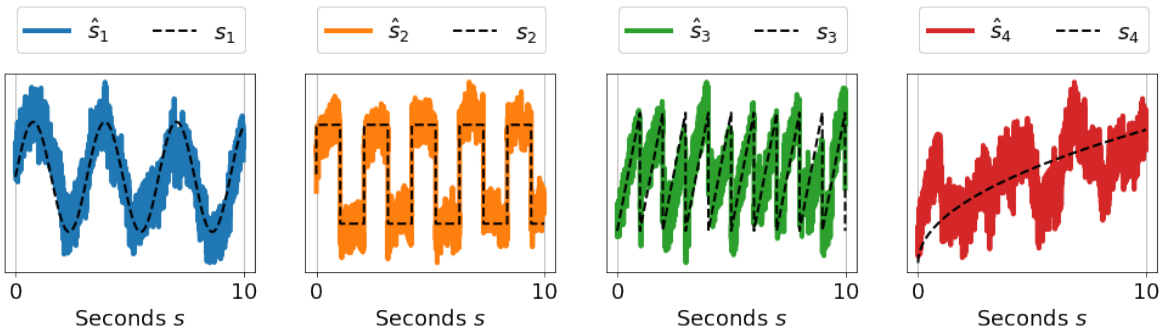


Figure D-2: Example of a failed set of estimated components from a data-set with $P = 6$ mixtures.. The estimated components are corrected for sign and mean such that they align correctly with the source components shown in dashed black lines.

	$\hat{s}_1$	$\hat{s}_2$	$\hat{s}_3$	$\hat{s}_4$
e	0.557	0.512	0.170	0.213
c	0.845	0.907	0.986	0.869

(a) The error values e and correlations c of the estimated sources from Figure D-1.

	$\hat{s}_1$	$\hat{s}_2$	$\hat{s}_3$	$\hat{s}_4$
e	0.599	0.434	0.533	0.820
c	0.773	0.871	0.695	0.624

(b) The error values e and correlations c of the estimated sources from Figure D-2.

Table D-2: Illustration of the possible range of estimation errors and correlation values for estimated sources.

D-2 Canonical Polyadic Decomposition (CPD) performance

The figures below show the CP-error of the non-iterative and iterative algorithms with no, Higher-Order Singular Value Decomposition (HOSVD) and Generalized EigenValue Decomposition (GEVD) initialization. For the GEVD methods the symmetric and non-symmetric CP-errors are shown due to the asymmetry of the CPD they compute. Clearly their asymmetric CPD performance is better than their symmetric one. The CP-error shows a lot of similarity to the measure of diagonality. This is another reason why it was opted to discard the CP-error as a performance metric.

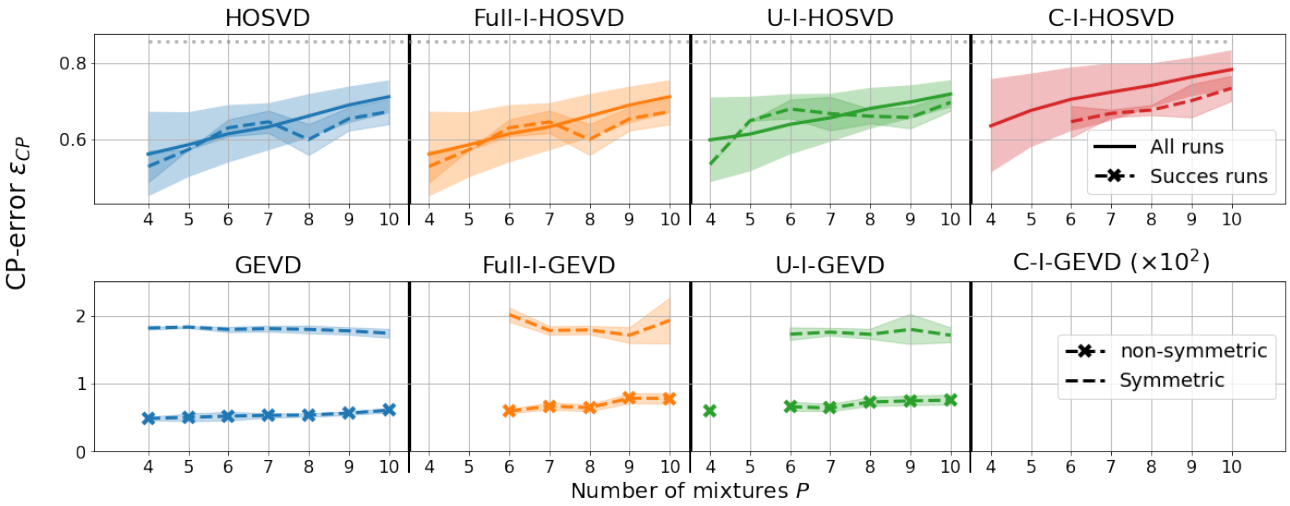


Figure D-3: Average CPD errors ε_{CP} (solid and dashed lines) together with their standard deviations (shaded areas) for varying amounts of mixtures $P = 4, \ldots, 10$ for a total of $N_{test} = 100$ runs. The solid lines represent only the successful runs and the dashed lines represent all 100 runs. The results are shown for the non-iterative algorithms.

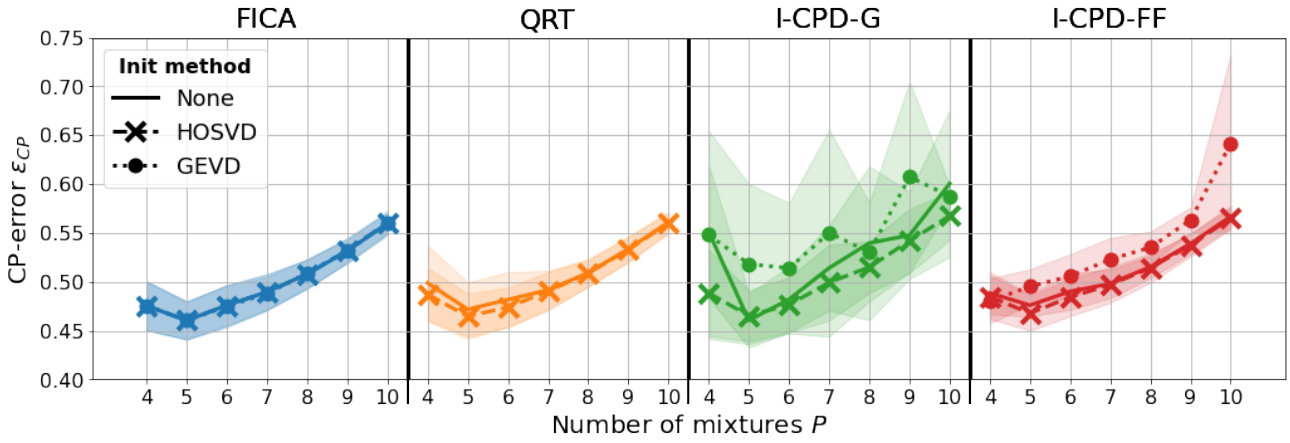


Figure D-4: Average CPD errors ε_{CP} (solid and dashed lines) together with their standard deviations (shaded areas) for varying amounts of mixtures $P = 4, \ldots, 10$ for a total of $N_{test} = 100$ runs. The solid lines represent only the successful runs and the dashed lines represent all 100 runs. The results are shown for the iterative with no, HOSVD and GEVD initialization only.

D-3 Qualitative analysis

The table below presents the complete overview of the qualitative analysis performed in chapter 5. The smaller table presented in chapter 5 is a generalization of these results.

	HOSVD			GEVD			
Success %	Explicit / Full	Type-U	Type-C	Explicit	Full	Type-U	Type-C
FICA	0	0	0	-	0	0	0
QRT	0	0	0	-	-	-	-
I-CPD-G	+	+	+	-	-	-	-
I-CPD-FF	+	++	0	-	0	+	0
Error ε_{total}							
FICA	0	0	0	-	0	0	0
QRT	+	+	+	-	-	-	-
I-CPD-G	+	+	+	-	+	0	-
I-CPD-FF	0	0	0	-	0	0	0
Diagonality τ_D							
FICA	0	0	0	0	0	0	0
QRT	+	++	+	-	-	-	-
I-CPD-G	+	+	0	-	-	-	-
I-CPD-FF	+	+	+	-	+	+	0
Time s							
FICA	+	+	+	+	0	0	0
QRT	+	+	+	-	-	-	-
I-CPD-G	+	+	+	+	+	+	-
I-CPD-FF	+	+	+	0	0	0	0

D-4 Convergence of fixed-point CPD

An example of the convergence behavior of the fixed-point CPD is shown below. The top figure shows the convergence error and the figure below that shows the absolute values of the weights $\lambda_i \quad \forall i = 1, 2, 3, 4, 5$.

The figures are on a logarithmic scale. One can clearly see the converging behavior of the algorithm. Moreover, the weights show the same type of converging behavior as the error. While not decreasing monotonically, all of the shown graphs illustrate the converging property of the fixed-point CPD method.

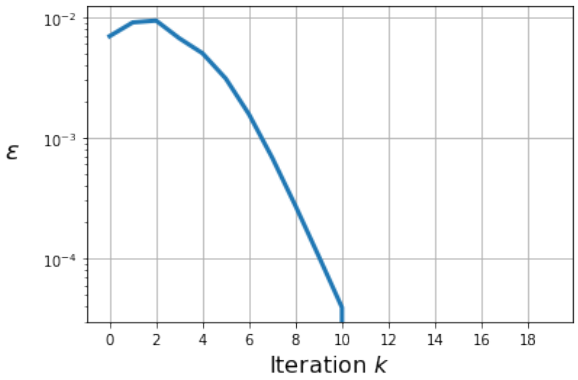


Figure D-5: Convergence error during the fixed-point CPD procedure.

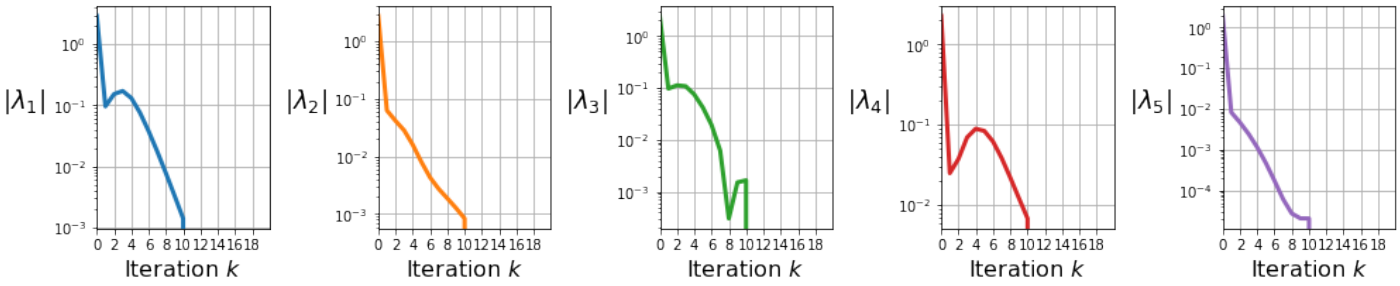


Figure D-6: Values of $\lambda_i \quad \forall i = 1, 2, 3, 4, 5$ during the fixed-point CPD procedure.

Bibliography

[1] V. Zarzoso and P. Comon, "How fast is FastICA?; How fast is FastICA?," tech. rep., 2006.

[2] S. C. Douglas and J. C. Chao, "Simple, robust, and memory-efficient fastICA algorithms using the huber M-estimator cost function," in *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 48, pp. 143–159, 8 2007.

[3] P. Tichavský, Z. Koldovský, and E. Oja, "Performance analysis of the FastICA algorithm and Cramér-Rao bounds for linear independent component analysis," *IEEE Transactions on Signal Processing*, vol. 54, pp. 1189–1203, 4 2006.

[4] A. Cuzzocrea, I.-Y. Song, and K. C. Davis, "Analytics over large-scale multidimensional data: the big data revolution!," in *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP*, pp. 101–104, 2011.

[5] Y. Liu, "Tensors for Data Processing: Theory, Methods, and Applications," tech. rep., 2022.

[6] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor Decomposition for Signal Processing and Machine Learning," *IEEE Transactions on Signal Processing*, vol. 65, pp. 3551–3582, 7 2017.

[7] N. Lee and A. Cichocki, "Fundamental tensor operations for large-scale data analysis using tensor network formats," *Multidimensional Systems and Signal Processing*, vol. 29, pp. 921–960, 7 2018.

[8] N. Lassance, V. DeMiguel, and F. Vrins, "Optimal Portfolio Diversification via Independent Component Analysis," *Operations Research*, vol. 70, pp. 55–72, 1 2022.

[9] C. R. Harvey, J. C. Liechty, M. W. Liechty, and P. Müller, "Portfolio selection with higher moments," *Quantitative Finance*, vol. 10, no. 5, pp. 469–485, 2010.

[10] P. Comon, "A useful Tool in Signal Processing," tech. rep., 1994.

[11] P. Comon, "Independent Component Analysis, a new concept? SIGNAL PROCESSING Independent component analysis, A new concept?*," 1994.

[12] Y. Guo and A. Kareem, "System identification through nonstationary data using Time-Frequency Blind Source Separation," *Journal of Sound and Vibration*, vol. 371, pp. 110–131, 6 2016.

[13] M. Kothandaraman, Z. Law, M. A. G. Ezra, and C. H. Pua, "Adaptive Independent Component Analysis–Based Cross-Correlation Techniques along with Empirical Mode Decomposition for Water Pipeline Leakage Localization Utilizing Acousto-Optic Sensors," *Journal of Pipeline Systems Engineering and Practice*, vol. 11, p. 04020027, 8 2020.

[14] N. Hassan and D. A. Ramli, "A Comparative study of Blind source separation for Bioacoustics sounds based on FastICA, PCA and NMF," in *Procedia Computer Science*, vol. 126, pp. 363–372, Elsevier B.V., 2018.

[15] B. Hunyadi, P. Dupont, W. Van Paesschen, and S. Van Huffel, "Tensor decompositions and data fusion in epileptic electroencephalography and functional magnetic resonance imaging data," 1 2017.

[16] H. Huang, J. Lu, J. Wu, Z. Ding, S. Chen, L. Duan, J. Cui, F. Chen, D. Kang, L. Qi, W. Qiu, S. W. Lee, S. J. Qiu, D. Shen, Y. F. Zang, and H. Zhang, "Tumor tissue detection using blood-oxygen-level-dependent functional MRI based on independent component analysis," *Scientific Reports*, vol. 8, 12 2018.

[17] P. Comon and C. Jutten, "Handbook of Blind Source Separation_Chapters," 2010.

[18] S. Guo, M. Shi, Y. Zhou, J. Yu, and E. Wang, "An efficient convolutional blind source separation algorithm for speech signals under chaotic masking," *Algorithms*, vol. 14, no. 6, 2021.

[19] L. Drude and R. Haeb-Umbach, "Integration of Neural Networks and Probabilistic Spatial Models for Acoustic Blind Source Separation," *IEEE Journal on Selected Topics in Signal Processing*, vol. 13, pp. 815–826, 8 2019.

[20] M. E. Fouda, E. Neftci, A. Eltawil, and F. Kurdahi, "Independent Component Analysis Using RRAMs," *IEEE Transactions on Nanotechnology*, vol. 18, pp. 611–615, 2019.

[21] A. Hyvärinen, J. Karhunen, and E. Oja, "Independent Component Analysis," tech. rep., 2001.

[22] P. Boÿll and M. Zibulevsky, "Underdetermined blind source separation using sparse representations," tech. rep., 2001.

[23] J.-F. Cardoso and A. Souloumiac, "Blind beamforming for non-gaussian signals," in *IEE proceedings F (radar and signal processing)*, vol. 140, pp. 362–370, IET, 1993.

[24] L. De Lathauwer, J. Castaing, and J. F. Cardoso, "Fourth-order cumulant-based blind identification of underdetermined mixtures," *IEEE Transactions on Signal Processing*, vol. 55, pp. 2965–2973, 6 2007.

[25] T. Blaschke and L. Wiskott, "CuBICA: Independent component analysis by simultaneous third- and fourth-order cumulant diagonalization," *IEEE Transactions on Signal Processing*, vol. 52, pp. 1250–1256, 5 2004.

[26] G. Sahonero-Alvarez and H. Calderon, "A Comparison of SOBI, FastICA, JADE and Infomax Algorithms," tech. rep.

[27] S. Zhang and C. Zhao, "Hybrid independent component analysis (H-ICA) with simultaneous analysis of high-order and second-order statistics for industrial process monitoring," *Chemometrics and Intelligent Laboratory Systems*, vol. 185, pp. 47–58, 2 2019.

[28] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," 2009.

[29] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," 3 2015.

[30] C. Chatzichristos, E. Kofidis, M. Morante, and S. Theodoridis, "Blind fMRI source unmixing via higher-order tensor decompositions," *Journal of Neuroscience Methods*, vol. 315, pp. 17–47, 3 2019.

[31] C. Chatzichristos, M. Vandecapelle, E. Kofidis, S. Theodoridis, L. De Lathauwer, and S. Van Huffel, "Tensor-based blind fmri source separation without the gaussian noise assumption—a β-divergence approach," in *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1–5, IEEE, 2019.

[32] N. Sompairac, P. V. Nazarov, U. Czerwinska, L. Cantini, A. Biton, A. Molkenov, Z. Zhumadilov, E. Barillot, F. Radvanyi, A. Gorban, U. Kairov, and A. Zinovyev, "Independent component analysis for unraveling the complexity of cancer omics datasets," 9 2019.

[33] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural computation*, vol. 9, no. 7, pp. 1483–1492, 1997.

[34] L. Chen, X. Zou, B. B. Chen, Y. Chen, J. Li, H. Zou, and J. Xiong, "An improved FastICA algorithm and its application in image feature extraction," in *Advanced Materials Research*, vol. 204-210, pp. 1485–1489, 2011.

[35] J. F. Cardoso, "Source separation using higher order moments," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 4, pp. 2109–2112, Publ by IEEE, 1989.

[36] 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, 2006.

[37] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines, "A Blind Source Separation Technique Using Second-Order Statistics," Tech. Rep. 2, 1997.

[38] J. Z. Buchwald, J. L. Berggren, J. L. Advisory, B. C. Fraser, T. Sauer, and A. Shapiro, "Sources and Studies in the History of Mathematics and Physical Sciences Managing Editor," tech. rep.

[39] N. Correa, T. Adali, and V. D. Calhoun, "Performance of Blind Source Separation Algorithms for FMRI Analysis using a Group ICA Method," tech. rep.

[40] V. D. Calhoun and T. Adali, "Unmixing fMRI with independent component analysis," *IEEE Engineering in Medicine and Biology Magazine*, vol. 25, pp. 79–90, 3 2006.

[41] V. D. Calhoun, J. Liu, and T. Adalı, "A review of group ica for fmri data and ica for joint inference of imaging, genetic, and erp data," *Neuroimage*, vol. 45, no. 1, pp. S163–S172, 2009.

[42] V. D. Calhoun, T. Adali, G. Pearlson, and J. Pekar, "Group ica of functional mri data: separability, stationarity, and inference," in *Proc. Int. Conf. on ICA and BSS San Diego, CA. p*, vol. 155, 2001.

[43] T. H. Kim and H. White, "On more robust estimation of skewness and kurtosis," *Finance Research Letters*, vol. 1, pp. 56–73, 3 2004.

[44] P. McCullagh, *Tensor methods in statistics*. Chapman and Hall/CRC, 2018.

[45] A. K. Nandi, *Blind Estimation Using Higher-Order Statistics*. Springer US, 1999.

[46] M. A. Lindquist, "The Statistical Analysis of fMRI Data," *Statistical Science*, vol. 23, pp. 439–464, 11 2008.

[47] P. Chevalier, L. Albera, P. Comon, and A. Ferréol, "Comparative performance analysis of eight blind source separation methods on radiocommunications signals," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, vol. 1, pp. 273–278, IEEE, 2004.

[48] L. De Lathauwer, B. De Moor, and J. Vandewalle, "Independent component analysis and (simultaneous) third-order tensor diagonalization," *IEEE Transactions on Signal Processing*, vol. 49, no. 10, pp. 2262–2271, 2001.

[49] J. Miettinen, S. Taskinen, K. Nordhausen, and H. Oja, "Fourth moments and independent component analysis," *Statistical Science*, vol. 30, no. 3, pp. 372–390, 2015.

[50] K. Nordhausen and H. Oja, "Independent component analysis: A statistical perspective," 9 2018.

[51] J. F. Cardoso, "Eigen-structure of the fourth-order cumulant tensor with application to the blind source separation problem," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 5, pp. 2655–2658, Publ by IEEE, 1990.

[52] B. L. R. De Moor, K. U. Leuven, D. Elektrotechniek, L. De Lathauwer, B. De Moor, and J. Vandewalle, "Blind Source Separation by Higher-Order Singular Decomposition Least Squares SVM View project Quantum entanglement distillation View project Katholieke Universiteit Leuven Blind Source Separation by Higher-Order Singular Value Decomposition 1 Blind Source Separation by Higher-Order Singular Value Decomposition 1," tech. rep., 1994.

[53] S. Sherman and T. G. Kolda, "Estimating higher-order moments using symmetric tensor decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 41, no. 3, pp. 1369–1387, 2020.

[54] R. Pan, "Tensor Transpose and Its Properties," 11 2014.

[55] P. Comon, G. Golub, L.-H. Lim, and B. Mourrain, "Symmetric tensors and symmetric tensor rank," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1254–1279, 2008.

[56] L. Qi, "Eigenvalues of a real supersymmetric tensor," *Journal of Symbolic Computation*, vol. 40, no. 6, pp. 1302–1324, 2005.

[57] H. Fischer, *A history of the central limit theorem: from classical to modern probability theory*. Springer, 2011.

[58] L. De Lathauwer, B. De Moor, and J. Vandewalle, "Independent component analysis and (simultaneous) third-order tensor diagonalization," *IEEE Transactions on Signal Processing*, vol. 49, pp. 2262–2271, 10 2001.

[59] V. Maurandi and E. Moreau, "Non-orthogonal simultaneous diagonalization of k-order complex tensors for source separation," *IEEE Signal Processing Letters*, vol. 24, no. 11, pp. 1621–1625, 2017.

[60] P. Mccullagh, "Tensor Methods in Statistics," tech. rep.

[61] X. Gao, T. Zhang, and J. Xiong, "Comparison between spatial and temporal independent component analysis for blind source separation in fMRI data," in *Proceedings - 2011 4th International Conference on Biomedical Engineering and Informatics, BMEI 2011*, vol. 2, pp. 690–692, 2011.

[62] A. Dermoune and T. Wei, "FastICA algorithm: Five criteria for the optimal choice of the nonlinearity function," *IEEE Transactions on Signal Processing*, vol. 61, no. 8, pp. 2078–2087, 2013.

[63] P. Chevalier, L. Albera, P. Comon, and A. Ferréol, "Comparative performance analysis of eight blind source separation methods on radiocommunications signals," in *IEEE International Conference on Neural Networks - Conference Proceedings*, vol. 1, pp. 273–278, 2004.

[64] 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, 2000.

[65] L. De Lathauwer and B. De Moor, "From matrix to tensor: Multilinear algebra and signal processing," in *Institute of mathematics and its applications conference series*, vol. 67, pp. 1–16, Citeseer, 1998.

[66] M. Gu and S. C. Eisenstat, "A stable and fast algorithm for updating the singular value decomposition," 1993.

[67] L. Greengard, L. Greengard, and V. Rokhlin, "A fast algorithm for particle simulations Related papers A Fast Algorithm for Particle Simulations*," tech. rep., 1997.

[68] J. Carriert, L. Greengardt, and V. Rokhlint, "A FAST ADAPTIVE MULTIPOLE ALGORITHM FOR PARTICLE SIMULATIONS*," Tech. Rep. 4, 1988.

[69] A. Grig, "Svd-update: Fast recalculation of svd for new attached column (row)." https://github.com/AlexGrig/svd_update, 2015.

[70] P. Stange, "On the Efficient Update of the Singular Value Decomposition Subject to Rank-One Modifications On the Efficient Update of the Singular Value Decomposition Subject to Rank-One Modifications *," tech. rep., 2011.

[71] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra and Its Applications*, vol. 415, pp. 20–30, 5 2006.

[72] K. Batselier and N. Wong, "A qr algorithm for symmetric tensors," *arXiv preprint arXiv:1411.1926*, 2014.

[73] J. G. Francis, "The qr transformation a unitary analogue to the lr transformation—part 1," *The Computer Journal*, vol. 4, no. 3, pp. 265–271, 1961.

[74] J. G. Francis, "The qr transformation—part 2," *The Computer Journal*, vol. 4, no. 4, pp. 332–345, 1962.

[75] V. N. Kublanovskaya, "On some algorithms for the solution of the complete eigenvalue problem," *USSR Computational Mathematics and Mathematical Physics*, vol. 1, no. 3, pp. 637–657, 1962.

[76] C. Hillar and L. Lim, "Most tensor problems are np-hard (2012)," *Preprint*.

[77] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[78] J. Håstad, "Tensor rank is np-complete," *Journal of Algorithms*, vol. 11, no. 4, pp. 644–654, 1990.

[79] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[80] R. A. Harshman *et al.*, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[81] P. Comon and B. Mourrain, "Decomposition of quantics in sums of powers of linear forms," tech. rep., 1999.

[82] X. Zhang, Z. H. Huang, and L. Qi, "Comon's conjecture, rank decomposition, and symmetric rank decomposition of symmetric tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 4, pp. 1719–1728, 2016.

[83] P. Comon, G. Golub, L. H. Lim, and B. Mourrain, "Symmetric tensors and symmetric tensor rank," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1254–1279, 2008.

[84] A. Casarotti, A. Massarenti, and M. Mella, "On comon's and strassen's conjectures," *Mathematics*, vol. 6, no. 11, p. 217, 2018.

[85] B. W. Bader and T. G. Kolda, "Efficient matlab computations with sparse and factored tensors," *SIAM Journal on Scientific Computing*, vol. 30, no. 1, pp. 205–231, 2008.

[86] 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.

[87] J. B. Kruskal, "Rank, decomposition, and uniqueness for 3-way and n-way arrays," in *Multiway data analysis*, pp. 7–18, 1989.

[88] J.-F. Cardoso, "Super-symmetric decomposition of the fourth-order cumulant tensor. blind identification of more sources than sensors.," in *ICASSP*, vol. 91, pp. 3109–3112, Citeseer, 1991.

[89] P. Comon, "Tensor Decompositions, State of the Art and Applications," 5 2009.

[90] N. D. Sidiropoulos and R. Bro, "On the uniqueness of multilinear decomposition of n-way arrays," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 14, no. 3, pp. 229–239, 2000.

[91] M. Sørensen, L. D. Lathauwer, P. Comon, S. Icart, and L. Deneire, "Canonical polyadic decomposition with a columnwise orthonormal factor matrix," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1190–1213, 2012.

[92] A. M. Dale and R. L. Buckner, "Selective averaging of rapidly presented individual trials using fmri," *Human brain mapping*, vol. 5, no. 5, pp. 329–340, 1997.

[93] C. F. Beckmann and S. M. Smith, "Tensorial extensions of independent component analysis for multisubject FMRI analysis," *NeuroImage*, vol. 25, no. 1, pp. 294–311, 2005.

[94] A. Stegeman, "Comparing Independent Component Analysis and the Parafac model for artificial multi-subject fMRI data," tech. rep., 2007.

[95] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "Tensorly: Tensor learning in python," *Journal of Machine Learning Research*, vol. 20, no. 26, pp. 1–6, 2019.

[96] L. Cheng, Y.-C. Wu, and H. V. Poor, "Probabilistic tensor canonical polyadic decomposition with orthogonal factors," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 663–676, 2016.

[97] P. Paatero, "A weighted non-negative least squares algorithm for three-way 'parafac'factor analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 38, no. 2, pp. 223–242, 1997.

[98] G. Tomasi, *Practical and Computational Aspects in Chemometric Data Analysis: Ph. D. Dissertation.* Department of Food Science, Royal Veterinary and Agricultural University, 2006.

[99] E. Sanchez and B. R. Kowalski, "Tensorial resolution: a direct trilinear decomposition," *Journal of Chemometrics*, vol. 4, no. 1, pp. 29–45, 1990.

[100] "Cardoso_Souloumiac,"

[101] C. E. R. Fernandes, G. Favier, and J. C. M. Mota, "Blind channel identification algorithms based on the parafac decomposition of cumulant tensors: the single and multiuser cases," *Signal Processing*, vol. 88, no. 6, pp. 1382–1401, 2008.

[102] I. Domanov and L. De Lathauwer, "Blind channel identification of miso systems based on the cp decomposition of cumulant tensors," in *2011 19th European Signal Processing Conference*, pp. 2215–2218, IEEE, 2011.

[103] E. Acar, D. M. Dunlavy, and T. G. Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *Journal of Chemometrics*, vol. 25, pp. 67–86, 2 2011.

[104] T. G. Kolda, "Numerical Optimization for Symmetric Tensor Decomposition," 10 2014.

[105] E. Vincent, A. Yeredor, Z. Koldovský, and P. Tichavský, eds., *Latent Variable Analysis and Signal Separation*, vol. 9237 of *Lecture Notes in Computer Science*. Cham: Springer International Publishing, 2015.

[106] M. K. Tsatsanis and C. Kweon, "Blind source separation of non-stationary sources using second-order statistics," in *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1574–1578, IEEE Comp Soc, 1998.

[107] S. J. Roberts, W. Addison, and S. Roberts, "Blind source separation with non-stationary mixing using waveletss," tech. rep., 2006.

[108] L. Parra and C. Spence, "On-line Convolutive Blind Source Separation of Non-Stationary Signals," tech. rep., 2000.

[109] P. Georgiev, F. Theis, and A. Cichocki, "Sparse component analysis and blind source separation of underdetermined mixtures," *IEEE Transactions on Neural Networks*, vol. 16, pp. 992–996, 7 2005.

[110] L. Zhen, D. Peng, Z. Yi, Y. Xiang, and P. Chen, "Underdetermined Blind Source Separation Using Sparse Coding," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 3102–3108, 12 2017.

[111] Y. Li, A. Cichocki, Y. Li, A. Cichocki, and S. Amari, "Analysis of Sparse Representation and Blind Source Separation Shun-ichi Amari," tech. rep., 2004.

[112] O. Eches and M. Guillaume, "A bilinear-bilinear nonnegative matrix factorization method for hyperspectral unmixing," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, pp. 778–782, 4 2014.

[113] L. Jarboui, S. Hosseini, Y. Deville, R. Guidara, and A. Ben Hamida, "A new unsupervised method for hyperspectral image unmixing using a linear-quadratic model," in *2014 1st International Conference on Advanced Technologies for Signal and Image Processing, ATSIP 2014*, pp. 423–428, IEEE Computer Society, 2014.

[114] S. Bermejo, C. Jutten, and J. Cabestany, "ISFET source separation: Foundations and techniques," *Sensors and Actuators, B: Chemical*, vol. 113, pp. 222–233, 1 2006.

[115] L. T. Duarte, C. Jutten, and S. Moussaoui, "A Bayesian Nonlinear Source Separation Method for Smart Ion-Selective Electrode Arrays," *IEEE Sensors Journal*, vol. 9, no. 12, pp. 1763–1771, 2009.

[116] B. Ehsandoust, "Blind Source Separation in Nonlinear Mixtures," tech. rep.

[117] E. Vincent, A. Yeredor, Z. Koldovský, and P. Tichavský, "Latent Variable Analysis and Signal Separation," tech. rep.

[118] A. Holobar and D. Zazula, "Multichannel blind source separation using convolution Kernel compensation," *IEEE Transactions on Signal Processing*, vol. 55, pp. 4487–4496, 9 2007.

[119] L. C. Khor, W. L. Woo, J. Zhang, W. L. Woo, and S. S. Dlay, "Nonlinear Convolutive Blind Source Separation of Non-Stationary Signals Single Channel Source Separation View project Sclera Recognititon View project Nonlinear Convolutive Blind Source Separation of Non-Stationary Signals," Tech. Rep. 4, 2007.

[120] E. R. Jessup and D. C. Sorensen, "A parallel algorithm for computing the singular value decomposition of a matrix," *Siam Journal on Matrix Analysis and Applications*, vol. 15, no. 2, pp. 530–548, 1994.

[121] M. De Vos, D. Nion, S. Van Huffel, and L. De Lathauwer, "A combination of parallel factor and independent component analysis," *Signal Processing*, vol. 92, pp. 2990–2999, 12 2012.

Glossary

List of Acronyms

CP-rank	CP-rank
BSS	Blind Source Separation
ICA	Independent Component Analysis
fMRI	functional Magnetic Resonance Imaging
CPD	Canonical Polyadic Decomposition
HOS	Higher Order Statistics
IID	Independently and Identically Distributed
MIMO	Multi-Input-Multi-Output
PCA	Principal Component Analysis
CLT	Central Limit Theorem
I.I.D.	Independent and Identically Distributed
IC	Independent Component
IC's	Independent Components
CF	Characteristic function
mgf	moment generating function
cgf	cumulant generating function
CPD	Canonical Polyadic Decomposition
HOSVD	Higher-Order Singular Value Decomposition
SVD	Singular-Value-Decomposition
EVD	Eigen-Value-Decomposition
ALS	Alternating Least Squares
SNR	Signal to Noise Ratio
TTSV	Tensor Times Same Vector

NLS Non-linear Least Squares

PARAFAC Parallel Factor Analysis

tICA tensor Independent Component Analysis
CPDCanonical-Polyadic-Decomposition

CP-rank Canonical-Polyadic-rank

GEVD Generalized EigenValue Decomposition

CCI co-channel interference

JADE Joint Approximate Diagonalization of Eigenmatrices

FMM Fast Multipole method

CP-GEVD Canonical-Polyadic-Generalized-Eigenvalue-decomposition

GWN Gaussian-White-Noise

ICA-SSE ICA Solution Sorting and Evalutation

L-BFGS-B Limited-memory-Broyden–Fletcher–Goldfarb–Shanno-Bound

OALS Orthogonally-constraint-Alternating-Least-Squares

QRT QR-Tensor algorithm