Some mathematical results on three-way component analysis

Jorge Tendeiro

Cover design: Ridderprint BV

Printing: Ridderprint BV

This research was supported by a grant from POPH/FSE (Programa Operacional Potencial Humano/Fundo Social Europeu).

RIJKSUNIVERSITEIT GRONINGEN

Some mathematical results on three-way component analysis

Proefschrift

ter verkrijging van het doctoraat in de
Gedrags- en Maatschappijwetenschappen
aan de Rijksuniversiteit Groningen
op gezag van de
Rector Magnificus, dr. F. Zwarts,
in het openbaar te verdedigen op
donderdag 28 oktober 2010
om 16.15 uur

door

Jorge Nunes Tendeiro
geboren op 26 december 1977
te Port Kembla, Australië

Promotores:	Prof. dr. J. M. F. Ten Berge
	Prof. dr. H. A. L. Kiers
Beoordelingscommisie:	Prof. dr. P. J. F. Groenen
	Prof. dr. P. M. Kroonenberg
	Prof. dr. R. Rocci

ISBN: 978-90-367-4613-7

Contents

Introduction

One can define the general purpose of Component Analysis as finding summarizers of the relationships among observed variables. More precisely, Component Analysis aims at finding a small set of conceptual (non-observed) variables called *components*. Components are expected to explain, to a large extent, the existing relations among the set of observed variables at hand.

The study of the relations between variables goes back to K. Pearson and G. U. Yule (late 19th century), who were interested in studying the pairwise correlations between variables (Mulaik [62], pp. 5-6). The merit of pioneering Component Analysis is usually credited to Pearson [71] and Hotelling [32] (Jolliffe [34], p. 7). It should be noted that Common Factor Analysis also arose in around the same period (with Spearman [81]). Both Component and Common Factor Analysis are techniques which seek to perform a variable reduction, but they are different in nature. This thesis is primarily focused on Component Analysis techniques.

The idea of extending the concepts of Component Analysis to higher dimensions occured naturally. Two-way component models do not fit well data that arise from processes where there are at least three sources of variation present. For instance, consider a situation where data matrices concerning a group of subjects on several variables were collected on a weekly basis. Such data has typically a three-way structure: each data entry is triply indexed according to the corresponding subject, variable and week. Although one could think of applying two-way procedures to each dataset one at a time, it should be stressed that such an analysis would endanger disregarding the joint interactions inherent in the data. A more adequate way of tackling such data is precisely by resorting to three-way, or more generally, multi-way methods.

The multi-way analysis of data had its origins with Hitchcock [29, 30]. Tucker in the 1960s was the first to formulate principal component techniques for three-way data. His three-way Principal Component Analysis is one of the most common multi-way components models used. Later, Harshman [24] and Carroll and Chang [11] independently rediscovered the model previously proposed by Hitchcock. They named their (equivalent) models as PARAFAC and CANDECOMP, respectively. These models quickly became quite popular due to its appealing algebraic properties. Since then much has been achieved in the multi-way world. New models and algorithms have been proposed, and applications to real data show how useful these methods can be. Nowadays there are several models suitable for analyzing three- and higher-way data sets. The choice model to use depends on the type of data at hand and on the objectives that one wishes to accomplish.

In spite of this, the use of multi-way models is still far from being widespread. The mathematical complexity of these models can be hard to grasp, thus inhibiting its use by less technical users. Also, multi-way techniques are still scarcely implemented (if at all) in the major statistical packages. Therefore, it seems pertinent to contribute to making multi-way tools more accessible.

This thesis is thought as a contribution in further understanding some core properties in the multi-way field, specifically in three-way analysis. Some of the most important concepts in three-way analysis are introduced and discussed in the course of this thesis. Also, some new results will be presented. Their main purpose is to further enlighten the potential and the limitations of multi-way methods.

Organization of this thesis

This thesis can be thought of as consisting of two main parts. The first main part includes Chapters 1-5, and the second main part includes Chapters 6-7 together with the appendices.

First part of the thesis: Chapters 1-5

The first part of the thesis (Chapters 1-5) was written with two main goals in mind. The first goal was to present some relevant concepts and results in the field of three-way analysis. The second goal was to prepare the readers who wish to follow the second part of the thesis (Chapters 6-7 and appendices).

We did not intend to perform a full literature coverage. We were mainly focused on our research needs, and the choice of subjects covered in these Chapters reflects this fact. Our hopes are that this part of the thesis can prepare the reader who wishes to read Chapters 6-7, which directly concern our research. The reader of this thesis should keep this idea in mind when approaching Chapters 1-5.

Almost all the material to be found in Chapters 1-5 is not new. The main exceptions are the proof of Lemma 1 (p. 43), and a reference to a proof for Theorem 3 (p. 47), for which an alternative proof was written together with Jos ten Berge (this ultimately lead to a publication, see Ten Berge and Tendeiro [112]).

In Chapter 1 some concepts and notation are introduced. The idea is to provide an easy access to definitions that will occur in all subsequent chapters of this thesis. The focus is given to special types of matrix operators (outer product of vectors, Kronecker product, Khatri-Rao product, Hadamard product, vec operator), matrix decompositions (eigenvalue decomposition, singular value decomposition, principal components analysis), higher order structures called *arrays* and some of its properties. Also, the issue of simultaneous diagonalization of square matrices is addressed; this also serves as an introduction to the contents of Chapters 5 and 6.

Chapter 2 serves to present the most important three-way models that were in the core of our research project. These models are 3PCA, CANDECOMP/PARAFAC (CP) and INDSCAL. Constrained 3PCA models are also covered; these form a family of models, which can be seen as a bridge between 3PCA and PARAFAC. Important considerations for each model are detailed. Algorithms are not introduced but references to the literature are given. It should be noted that there are more models in multi-way analysis than the ones presented here, but we do not introduce them in detail since they were not part of our research.

Chapter 3 concerns the property of uniqueness of the solutions provided by the 3PCA and CP models. It is explained that a 3PCA solution is usually non-unique: for a given array there exist an infinity of different solutions providing the same fit. Therefore, solutions apparently different might pertain to the same array. We shall discuss this aspect, as it is one of the important motivations for the issue of simplicity (Chapters 5 and 6). In contrast with the non-uniqueness of 3PCA solutions, it is explained that a CP solution is unique up to trivial transformations of components, under relatively mild conditions. This property also plays a role in Chapter 7, both in the theory and in applications.

Chapter 4 revolves around the problem of degeneracy in CP. Degeneracy is a common problem that sometimes affects the execution of the algorithm of CP. Typically, a CP sequence is said to be degenerate when the convergence of the algorithm becomes extremely slow, while some components become more and more correlated as the algorithm progresses. In this Chapter we look into the specificities of degeneracy. We discuss what degeneracy consists of, what are the reasons that can lead to degeneracy, and what can be done in order to try to avoid degeneracy. At this stage, it is important to understand that one must try to avoid degeneracy as much as possible, since degenerate solutions are not interpretable. Our motivation to discuss degeneracy in this dissertation is directly linked to Chapter 7. It will be argued that the occurence of degeneracy highly limitates the use of the tools that are developed in that Chapter.

In Chapters 5 and 6 we discuss simplicity of a 3PCA solution. Recall that in Chapter 3 it is explained how the 3PCA model is characterized by non-uniqueness. Therefore, one can try to think about ways of transforming 3PCA solutions into "more suitable" solutions. The idea would be to try to transform a 3PCA solution into another solution which is easier to interpret. Such transformation is supposed to be done without loss of fit involved. This simplicity problem in 3PCA is very similar to the rotational problem in two-way Principal Components Analysis.

Simplifying a 3PCA solution can be considered from two possible perspectives. One might want to find an equivalent solution with "simpler" components, counter-transforming the core array accordingly. Alternatively, one might want "simple" weights, counter-transforming the components accordingly. By "simple weights" we usually mean "having as many zeros as possible". Usually it is not possible to pursue both simplifications simultaneously. The approach in our research was the latter: we were interested in developing techniques that allow to find simple weights. Hence, our goal was to find methods that allow to transform (some) 3PCA solutions into equivalent 3PCA solutions in which many of the weights are transformed to zero.

The question of simplicity can be put in more general mathematical terms. It may be noted that the weights of a 3PCA solution can be organized in a three-way array. Therefore, simplifying the weights of a 3PCA solution is a case of simplifying an array. In fact, the methods of simplification of three-way arrays that will be presented can be applied in the wider framework of tensor theory. The simplification of 3PCA solutions is only one of the possible applications of these simplification procedures.

In Chapter 5 we discuss some computational approaches to simplicity transformations that are already available in the literature (SIMPLIMAX, the Orthogonal Complement Method, multiple orthonormality transformation). A couple of specific examples about simplicity are given. Also, the concept of maximal simplicity is introduced. The topics covered in this Chapter are supposed to prepare the reader to better understand the new results about simplicity that can be found in Chapter 6.

Second part of the thesis: Chapters 6-7

The second part of the thesis includes the main results that were found during our research. We divided the contents in two Chapters: Chapter 6 concerns research about simplicity for arrays with symmetric frontal slices, and Chapter 7 concerns a generalization of a study done by Bennani Dosse and Ten Berge [3].

The contents of each Chapter were recently submitted for publication. We added the published papers as Appendix I (p. 97) and Appendix II (p. 115). Because of this, we decided that the contents of both Chapters 6 and 7 should not repeat all the details already present in the papers. This means that these Chapters only give general overviews of the research that was done. The reader interested in a more detailed description of the performed research is invited to consult the appendices at the end of this dissertation.

In Chapter 6 we deal with the simplification of three-way arrays with symmetric frontal slices. The specificity of these arrays usually forbids applying any of the methods described in Chapter 5. The reason is that we are interested in maintaining the symmetry of the frontal slices after the transformation is performed. Therefore, specific methods were developed for this type of situation. These methods are presented on a case-by-case basis, due to the difficulty of finding unifying procedures that solve the problem for larger families of arrays.
Some applications illustrating the usefulness of simplification of arrays are shown. These applications revolve around the concepts of typical rank and maximal simplicity of an array.

In Chapter 7 we take a look into the differential structure of the loss function for CP and INDSCAL. The idea is to develop a tool that allows to classify stationary points of the optimization criteria, using information provided by the Jacobian and Hessian matrices. The motivation for this research came after Bennani Dosse and Ten Berge [3], who showed that there were specific INDSCAL decompositions that could only correspond to saddle points (stationary points which are not local optima) of the corresponding loss function. Bennani Dosse and Ten Berge [3] only analysed the situation with $r = 1$ component. Therefore we were interested in picturing what happens for solutions with more than one component.

Firstly, we rewrite the optimization functions for CP and INDSCAL. Afterwards, we compute the first and second order derivatives using matrix differential calculus. Both unconstrained and constrained situations are inspected. We also discuss some numerical problems that can arise during the computation of the Jacobian and Hessian matrices.

Three applications are worked out as exemplifications of how the classification of stationary points can be useful. The first application consists of revisiting Ten Berge, Knol and Kiers [106], which discussed results concerning an algorithm for INDSCAL under orthonormality constraints. The last two applications consist of simulation studies designed to inspect the nature of the stationary points that seem to occur in CP and INDSCAL, when more than one component is retained.

We end this thesis with a Summary, where the most important facts discussed in the thesis are presented.

Notation

In this Section we summarize most of the notation to be used throughout this thesis.

In general, we will denote scalar values with lower case italic font: a, x, λ. Matrices will be denoted with upper case bold-face font: $\mathbf{A}$, $\mathbf{X}$, $\mathbf{\Lambda}$. Arrays (defined in Section 1.8) will be indicated by an underlined upper case bold-face font: $\underline{\mathbf{A}}$, $\underline{\mathbf{X}}$, $\underline{\mathbf{\Lambda}}$. Because vectors are special cases of matrices, they will be denoted with lower case bold-face font: $\mathbf{a}$, $\mathbf{x}$, $\boldsymbol{\lambda}$.

Entries of matrices will often be indexed by the row and column of the matrix they belong to. For example, entry (i, j) of matrix $\mathbf{C}$ is c_{ij}. The (i, j, k) entry of array $\underline{\mathbf{X}}$ will be denoted by x_{ijk}.

The transpose of matrix $\mathbf{A}$ will be indicated by $\mathbf{A}'$. The j-th column of matrix $\mathbf{A}$ will be denoted by $\mathbf{a}_j$. The i-th row of $\mathbf{A}$ will receive no special notation, so $\mathbf{a}'_i$ shall denote the transposed i-th column of $\mathbf{A}$. Specific notation will be introduced whenever there is need to refer to a row of a matrix. The trace of $\mathbf{A}$ is denoted by $\mathrm{tr}(\mathbf{A})$. $\mathrm{vec}(\mathbf{A})$ reshapes $\mathbf{A}$ into a column vector by stacking the columns in sequence, one below the other. $\mathrm{diag}_{\mathrm{V}}(\mathbf{A})$ is the column vector holding the diagonal of $\mathbf{A}$. $\mathrm{vec}^*(\mathbf{A})$ denotes the vec of $\mathbf{A}'$. The k-rank of $\mathbf{A}$ (Section 1.2) will be denoted by $k_{\mathbf{A}}$.

For given matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \otimes \mathbf{B}$, $\mathbf{A} * \mathbf{B}$ and $\mathbf{A} \odot \mathbf{B}$ denote the Kronecker, Hadamard and Khatri-Rao products of $\mathbf{A}$ and $\mathbf{B}$, respectively. These products will be presented in Section 1.3.

A matrix that admits an inverse is said to be *invertible* or *nonsingular*.

Given a vector $\mathbf{d}$, $\mathrm{diag}_{\mathrm{M}}(\mathbf{d})$ denotes the diagonal matrix whose diagonal is equal to $\mathbf{d}$.

Given array $\underline{\mathbf{X}}$, we will denote $\mathbf{X}_a$ as matrix $[\mathbf{X}_1|\cdots|\mathbf{X}_K]$, $\mathbf{X}_{\mathrm{vec}}$ as matrix $[\mathrm{vec}(\mathbf{X}_1)|\cdots|\mathrm{vec}(\mathbf{X}_K)]$, $\mathbf{X}_{\mathrm{vec}^*}$ as matrix $[\mathrm{vec}^*(\mathbf{X}_1)|\cdots|\ \mathrm{vec}^*(\mathbf{X}_K)]$, and $\mathbf{x}_a$ will denote vector $\mathrm{vec}(\mathbf{X}_{\mathrm{vec}})$.

Special matrices

$\mathbf{0}_{mn}$ is the zero matrix of order $m \times n$. The vector of order n with only zero (unit) elements will be denoted by $\mathbf{0}_n$ ($\mathbf{1}_n$). The identity matrix of order n will be denoted by $\mathbf{I}_n$ or simply $\mathbf{I}$, when its order is clear from the context. Similarly, the unit superdiagonal array of order $n \times n \times n$ will be denoted by $\underline{\mathbf{I}}_n$ or just $\underline{\mathbf{I}}$. This array satisfies $\underline{\mathbf{I}}(i,j,k) = 1$ if $i = j = k$, and $\underline{\mathbf{I}}(i,j,k) = 0$ elsewhere.

$\mathbf{C}_{mn}$ is the $mn \times mn$ commutation matrix, i.e., $\mathbf{C}_{mn}\mathrm{vec}\,(\mathbf{A}) = \mathrm{vec}\,(\mathbf{A}')$; $\mathbf{T}_n$ is the $n^2 \times n$ matrix with unit entries in position $((i-1)n + i, i)$ for $i = 1, \ldots, n$ and zero elsewhere, and $\mathbf{E}_n = \mathbf{I}_{n^2} - \mathbf{T}_n\mathbf{T}_n'$. For example for $n = 3$

$$\mathbf{T}_3 = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{array}\right], \quad \mathbf{E}_3 = \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

Chapter 1

Matrices and arrays

1.1 Introduction

In this Chapter some fundamental concepts are introduced. These concepts include nomenclature, mathematical operations and matrix decompositions, among others. Some sections can be skipped by readers that are familiar with the concepts to be presented.

There is no general consensus regarding notation and terminology in Multiway Analysis. For examples on tutorials about good notation in the field we can refer to Kiers [41] and Harshman [26]. Here we will follow Kiers' approach whenever possible. In the following sections of this Chapter we will introduce the most important terminology to be used.

It is assumed that the reader has already some experience with basic Linear Algebra. Specifically, it is expected that the reader is familiar with the following concepts concerning matrices: addition and multiplication, rank, inverse, determinant, trace, partitioned matrices. Other fields from Linear Algebra that are expected to be familiar are: solving (systems of) linear equations, vector spaces, quadratic forms. Clarifications related to any of these concepts will be given whenever it is deemed useful.

The notion of k-rank is introduced in Section 1.2. Some useful operators for matrix computations are presented in Section 1.3. In Sections 1.4 and 1.5 we will present two matrix decompositions: the Eigenvalue Decomposition and the Singular Value Decomposition, respectively. Simultaneous diagonalization of matrices will be addressed in Section 1.6. A general overview of Principal Components Analysis (PCA) is introduced in Section 1.7. Arrays will be presented in Section 1.8.

Each of the topics discussed in this Chapter is relevant for better understanding the remaining of this dissertation. A general guideline is sketched below:

	1.2		Chapter 3
	1.3		Chapters 2-3, 5-7
	1.4		Chapters 3, 5-7
Section…	1.5	…is relevant for…	Chapters 2-3, 5-7
	1.6		Chapters 5-6
	1.7		Chapter 2
	1.8		Chapters 2-7

1.2 The k-rank of a matrix

Kruskal [49] used a special concept of matrix rank which has been named k-rank after him by Harshman and Lundy [27]. Throughout this thesis we will use the notation $k_{\mathbf{A}}$ to denote the k-rank of matrix $\mathbf{A}$.

The k-rank of a matrix is the largest value of m such that every subset of m columns of the matrix is linearly independent. It is clear from the definition that the k-rank of a matrix never exceeds its rank. Moreover, rank and k-rank coincide for matrices of full column rank.

The k-rank of a matrix is insensitive to premultiplication by a nonsingular matrix.

1.3 Vectors and matrices: special products and operators

The *outer product* of vectors $\mathbf{u}$ and $\mathbf{v}$ is the matrix given by $\mathbf{u} \circ \mathbf{v} = \mathbf{u}\mathbf{v}'$. The columns of this matrix are all proportional to $\mathbf{u}$, therefore it has rank one if $\mathbf{u}$ and $\mathbf{v}$ are nonzero vectors.

Consider now matrices $\mathbf{A}$ and $\mathbf{B}$ of order $p \times r$ and $k \times s$, respectively. The *Kronecker* product $\mathbf{A} \otimes \mathbf{B}$ is the $pk \times rs$ matrix defined by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1r}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{p1}\mathbf{B} & \cdots & a_{pr}\mathbf{B} \end{bmatrix}. \tag{1.1}$$

The *Khatri-Rao* product is defined for matrices with the same column order. Given matrices $\mathbf{A}$ and $\mathbf{B}$ of order $p \times r$ and $k \times r$, respectively, $\mathbf{A} \odot \mathbf{B}$ is the $pk \times r$

matrix defined by $[\mathbf{a}_1 \otimes \mathbf{b}_1 | \cdots | \mathbf{a}_r \otimes \mathbf{b}_r]$. So the Khatri-Rao product of $\mathbf{A}$ and $\mathbf{B}$ is the columnwise Kronecker product of $\mathbf{A}$ and $\mathbf{B}$.

The *elementwise* or *Hadamard* product, denoted by the symbol $*$, applies to matrices of the same order. Given matrices $\mathbf{A}$ and $\mathbf{B}$ of order $p \times r$,

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & \cdots & a_{1r}b_{1r} \\ \vdots & \ddots & \vdots \\ a_{p1}b_{p1} & \cdots & a_{pr}b_{pr} \end{bmatrix}. \tag{1.2}$$

A useful operator often applied to matrices is the "vec" operator. For a $p \times r$ matrix $\mathbf{A}$, $\text{vec}(\mathbf{A})$ is the $pr \times 1$ vector that consists of all columns of $\mathbf{A}$ stacked vertically, one below another. So, vec reshapes any matrix into a vector form. A useful property of the vec operator is

$$\text{vec}(\mathbf{ABC}) = (\mathbf{C}' \otimes \mathbf{A})\text{vec}(\mathbf{B}). \tag{1.3}$$

Some authors define the vec of $\mathbf{A}$ as the column vector containing all transposed rows of $\mathbf{A}$, one below another. This vector equals $\text{vec}\,(\mathbf{A}')$, and will be denoted by $\text{vec}^*(\mathbf{A})$. In this dissertation both versions of vec will be used.

Several properties associated to these operations can be found, for instance, in Magnus and Neudecker [60].

1.4 Eigenvalue Decomposition

We will be using the eigenvalue decomposition throughout this thesis, with special focus on Chapter 6. Here we recover some of the most important concepts associated to the eigenvalue decomposition.

The *eigenvalue decomposition*, or *eigendecomposition*, is defined for square matrices. Given a square matrix $\mathbf{A}$ of order n, λ is an *eigenvalue* of $\mathbf{A}$ with an associated *eigenvector* $\mathbf{k}$ if it satisfies

$$\mathbf{Ak} = \lambda\mathbf{k}, \tag{1.4}$$

where $\mathbf{k}$ is a nonzero vector or order n. The set of all eigenvalues of $\mathbf{A}$ can be found by solving the *characteristic equation* $\det(\mathbf{A} - \lambda\mathbf{I}_n) = 0$. The number of times each root occurs is called the *algebraic multiplicity* of the eigenvalue. For each eigenvalue λ, the associated eigenvectors are found by computing the right null of $(\mathbf{A} - \lambda\mathbf{I}_n)$. Eigenvectors associated to the same eigenvalue define a vector space, since any linear combination is still an eigenvector associated to the same eigenvalue.

The dimension of this space is called the *geometric multiplicity* of the eigenvalue. The geometric multiplicity does not exceed the algebraic multiplicity, but it can be smaller

A matrix of order n has n eigenvalues, real or complex. Eigenvectors associated to different eigenvalues are linearly independent. If $\mathbf{A}$ admits an n-dimensional basis of eigenvectors, then the eigendecomposition can be written in matrix notation

$$\mathbf{A} = \mathbf{K}\mathbf{\Lambda}\mathbf{K}^{-1}, \tag{1.5}$$

where $\mathbf{\Lambda}$ is a diagonal matrix holding the eigenvalues and $\mathbf{K}$ holds the corresponding eigenvectors as columns. Almost every matrix admits this decomposition. Matrices that do not have a basis of eigenvectors are called *defective*. They are characterized by having eigenvalues with geometric multiplicity smaller than the algebraic multiplicity. Defective matrices will not be of concern to us mainly for two reasons. One reason is that we mostly work with matrices randomly sampled from a continuous distribution, therefore with all eigenvalues distinct. Secondly, symmetric matrices are never defective, which will also help us to avoid this problem.

It is well-known that, for a symmetric matrix $\mathbf{S}$, $\mathbf{K}$ is orthogonal (its columns are mutually orthogonal) and it can always be made orthonormal ($\mathbf{K}'\mathbf{K} = \mathbf{I}_n = \mathbf{K}\mathbf{K}'$). So, for symmetric matrices the eigendecomposition reads

$$\mathbf{S} = \mathbf{K}\mathbf{\Lambda}\mathbf{K}', \tag{1.6}$$

where $\mathbf{K}$ is orthonormal and $\mathbf{\Lambda}$ is diagonal. Depending on the signs of the eigenvalues, $\mathbf{S}$ is said to be positive (semi)definite if the eigenvalues are all positive (non-negative), negative (semi)definite if the eigenvalues are all negative (non-positive), and indefinite otherwise.

If $\mathbf{S}$ is singular of rank r ($r < n$) then the decomposition can be written as

$$\mathbf{S} = \mathbf{K}_r\mathbf{\Lambda}_r\mathbf{K}_r', \tag{1.7}$$

where $\mathbf{\Lambda}_r$ is the $r \times r$ diagonal matrix holding the nonzero diagonal entries of $\mathbf{\Lambda}$, and $\mathbf{K}_r$ holds the corresponding columns of $\mathbf{K}$. Note that $\mathbf{\Lambda}_r$ is nonsingular and $\mathbf{K}_r$ is columnwise orthonormal ($\mathbf{K}_r'\mathbf{K}_r = \mathbf{I}_r$).

It is helpful to retain some results associated to the eigenvalue decomposition. We shall state these results without proof (see, for example, Magnus and Neudecker [60]).

Proposition 1

1. A real symmetric matrix has only real eigenvalues.

2. The sum of the eigenvalues equals the trace of the matrix.
3. The product of the eigenvalues equals the determinant of the matrix.
4. The rank of a matrix equals the number of its nonzero eigenvalues.
5. A symmetric matrix is positive definite (positive semidefinite) if and only if all its eigenvalues are positive (non-negative).

1.5 Singular Value Decomposition (SVD)

Unlike the eigenvalue decomposition, the *Singular Value Decomposition* (SVD) can be applied to any matrix.

Let $\mathbf{M}$ be a $p \times q$ matrix, $p \geqslant q$, of rank r. The SVD of $\mathbf{M}$ is the decomposition

$$\mathbf{M} = \mathbf{PDQ}', \tag{1.8}$$

with $\mathbf{P}'\mathbf{P} = \mathbf{I}_q$, $\mathbf{Q}'\mathbf{Q} = \mathbf{QQ}' = \mathbf{I}_q$, and $\mathbf{D}$ diagonal with non-negative diagonal elements arranged from high (upper left) to low (lower right). The diagonal elements of $\mathbf{D}$ are called the *singular values* of $\mathbf{M}$. The SVD decomposes $\mathbf{M}$ as a weighted sum of rank-one matrices,

$$\mathbf{M} = \sum_{i=1}^{q} d_i(\mathbf{p}_i \circ \mathbf{q}_i), \tag{1.9}$$

where $\mathbf{p}_i$ and $\mathbf{q}_i$ are the i-th columns of $\mathbf{P}$ and $\mathbf{Q}$, respectively, and d_i is the i-th diagonal entry of $\mathbf{D}$.

The rank of a matrix equals the number of its nonzero singular values. In fact,

$$\text{rank}(\mathbf{M}) = \text{rank}(\mathbf{M}'\mathbf{M}) = \text{rank}(\mathbf{QD}^2\mathbf{Q}') = \text{rank}(\mathbf{D}), \tag{1.10}$$

and the rank of a diagonal matrix is the number of nonzero entries in its diagonal.

When $\mathbf{M}$ is not of full rank (i.e. $r < q$), the SVD can be simplified to

$$\mathbf{M} = \mathbf{P}_r\mathbf{D}_r\mathbf{Q}_r', \tag{1.11}$$

where $\mathbf{P}_r$ (resp. $\mathbf{Q}_r$) is the matrix containing the first r columns of $\mathbf{P}$ (resp. $\mathbf{Q}$), and $\mathbf{D}_r$ is the upper left $r \times r$ submatrix of $\mathbf{D}$. Notice that $\mathbf{P}_r$ and $\mathbf{Q}_r$ are both columnwise orthonormal and $\mathbf{D}_r$ is nonsingular. Since $\mathbf{M}$ has rank r, no summation with less than r rank-one matrices is possible.

Another way of writing the SVD that is used, for example, by *Matlab*, is

$$\mathbf{M} = \mathbf{P}_1\mathbf{D}_1\mathbf{Q}', \tag{1.12}$$

where $\mathbf{P}_1$ is $p \times p$ orthonormal matrix (completed from $\mathbf{P}$ by adding columns), $\mathbf{D}_1 = \left[\begin{array}{c}\mathbf{D}\\ \hline \mathbf{0}\end{array}\right]$, and $\mathbf{Q}'\mathbf{Q} = \mathbf{Q}\mathbf{Q}' = \mathbf{I}_q$.

For horizontal matrices ($p < q$) we will use the SVD of its transpose.

The best rank-k approximation to $\mathbf{M}$ in the least squares sense is given by $\sum_{i=1}^{k} d_i(\mathbf{p}_i \circ \mathbf{q}_i)$, Eckart and Young [20]. This means that $\mathbf{X} = \sum_{i=1}^{k} d_i(\mathbf{p}_i \circ \mathbf{q}_i)$, the truncated SVD of $\mathbf{X}$ using the first k singular vectors/values, minimizes the function $f(\mathbf{X}) = \|\mathbf{X} - \mathbf{M}\|^2$ subject to the constraint that $\mathbf{X}$ be of rank k or less.

There is a close relation between the eigenvalue decomposition and the SVD for symmetric matrices. Let $\mathbf{S}$ be a symmetric matrix with eigendecomposition

$$\mathbf{S} = \mathbf{K}\boldsymbol{\Lambda}\mathbf{K}'. \tag{1.13}$$

If $\boldsymbol{\Lambda}$ has no negative diagonal elements then (1.13) is also a SVD. Otherwise, define a diagonal matrix $\mathbf{T}$, with the same order as $\boldsymbol{\Lambda}$, such that $t_{ii} = 1$ if $\lambda_{ii} \geqslant 0$, and $t_{ii} = -1$ if $\lambda_{ii} < 0$. Then $\mathbf{S} = (\mathbf{K}\mathbf{T})(\mathbf{T}\boldsymbol{\Lambda})\mathbf{K}'$ is a SVD for $\mathbf{S}$.

1.6 Simultaneous diagonalization of square matrices

In this Section we are interested in describing ways of simultaneously transforming a pair of matrices of the same order, say $\mathbf{A}$ and $\mathbf{B}$, into diagonal form. We will assume that at least one of the matrices, say $\mathbf{A}$, is nonsingular.

1.6.1 General situation

Suppose we have two square matrices $\mathbf{A}$ and $\mathbf{B}$ of the same order. We wish to simultaneously diagonalize $\mathbf{A}$ and $\mathbf{B}$, that is, to find invertible matrices $\mathbf{S}$ and $\mathbf{T}$ such that

$$\mathbf{S}'\mathbf{A}\mathbf{T} = \boldsymbol{\Phi}_1, \quad \mathbf{S}'\mathbf{B}\mathbf{T} = \boldsymbol{\Phi}_2, \tag{1.14}$$

where $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ are diagonal matrices. For a nonsingular matrix $\mathbf{A}$, a necessary and sufficient condition to simultaneously diagonalize $\mathbf{A}$ and $\mathbf{B}$ is that $\mathbf{A}^{-1}\mathbf{B}$ has a real eigendecomposition $\mathbf{A}^{-1}\mathbf{B} = \mathbf{K}\boldsymbol{\Phi}\mathbf{K}^{-1}$, that is, $\mathbf{K}$ and $\boldsymbol{\Phi}$ do not have complex elements. In fact, if the eigendecomposition of $\mathbf{A}^{-1}\mathbf{B}$ is real, take $\mathbf{S}' = \mathbf{K}^{-1}\mathbf{A}^{-1}$ and $\mathbf{T} = \mathbf{K}$. Conversely, suppose that $\mathbf{A}$ and $\mathbf{B}$ are simultaneously diagonalizable. This means there exist invertible matrices $\mathbf{S}$ and $\mathbf{T}$ such that $\mathbf{S}'\mathbf{A}\mathbf{T} = \boldsymbol{\Phi}_1$ and $\mathbf{S}'\mathbf{B}\mathbf{T} = \boldsymbol{\Phi}_2$, where $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ are diagonal matrices. Solving the last equalities in terms of $\mathbf{A}$ and $\mathbf{B}$ renders $\mathbf{A} = (\mathbf{S}')^{-1}\mathbf{\Phi}_1\mathbf{T}^{-1}$, $\mathbf{B} = (\mathbf{S}')^{-1}\mathbf{\Phi}_2\mathbf{T}^{-1}$, so

$$\mathbf{A}^{-1}\mathbf{B} = \mathbf{T}\mathbf{\Phi}_1^{-1}\mathbf{S}'(\mathbf{S}')^{-1}\mathbf{\Phi}_2\mathbf{T}^{-1} = \mathbf{T}\mathbf{\Phi}_1^{-1}\mathbf{\Phi}_2\mathbf{T}^{-1}, \tag{1.15}$$

which is a real eigendecomposition of $\mathbf{A}^{-1}\mathbf{B}$.

1.6.2 Special situation: symmetric matrices

Let $\mathbf{A}$ and $\mathbf{B}$ be symmetric matrices of the same order, $\mathbf{A}$ nonsingular.

We have seen in the previous section that a necessary and sufficient condition for simultaneously diagonalizing $\mathbf{A}$ and $\mathbf{B}$ is that $\mathbf{A}^{-1}\mathbf{B}$ has real eigenvalues. Moreover, the symmetry of $\mathbf{A}$ and $\mathbf{B}$ makes it possible to take $\mathbf{S}$ equal to $\mathbf{T}$ when the eigenvalues of $\mathbf{A}^{-1}\mathbf{B}$ are real: write the eigendecomposition $\mathbf{A}^{-1}\mathbf{B} = \mathbf{K}\mathbf{\Phi}\mathbf{K}^{-1}$ and make $\mathbf{S} = \mathbf{T} = \mathbf{K}$. To see that $\mathbf{K}'\mathbf{A}\mathbf{K}$ and $\mathbf{K}'\mathbf{B}\mathbf{K}$ are indeed diagonal matrices, observe that $\mathbf{A}\mathbf{K} = \mathbf{B}\mathbf{K}\mathbf{\Phi}^{-1}$ and hence the (i, j)-th element of $\mathbf{K}'\mathbf{A}\mathbf{K}$, with $i \neq j$, is given by

$$\mathbf{k}_i'(\mathbf{A}\mathbf{k}_j) = \phi_j^{-1}\mathbf{k}_i'\mathbf{B}\mathbf{k}_j \tag{1.16}$$

or, alternatively,

$$(\mathbf{k}_i'\mathbf{A})\mathbf{k}_j = \phi_i^{-1}\mathbf{k}_i'\mathbf{B}\mathbf{k}_j, \tag{1.17}$$

where $\mathbf{k}_i$ and $\mathbf{k}_j$ are columns i and j of $\mathbf{K}$, and ϕ_i, ϕ_j are the i-th and j-th diagonal entries of $\mathbf{\Phi}$. Both expressions for $\mathbf{k}_i'\mathbf{A}\mathbf{k}_j$ are equal if and only if $\mathbf{k}_i'\mathbf{B}\mathbf{k}_j = 0$, under the hypothesis that $\mathbf{A}^{-1}\mathbf{B}$ is nondefective (this happens almost surely). Therefore, $\mathbf{K}'\mathbf{B}\mathbf{K}$ is a diagonal matrix. Swapping $\mathbf{A}$ with $\mathbf{B}$ in the preceding argument proves that $\mathbf{K}'\mathbf{A}\mathbf{K}$ is also diagonal.

Another sufficient condition for simultaneously diagonalizing $\mathbf{A}$ and $\mathbf{B}$ is that one of the matrices, say $\mathbf{A}$, is positive definite (all eigenvalues are positive). This can be understood using the fact that $\mathbf{A}^{-1}\mathbf{B}$ and $\mathbf{A}^{-1/2}\mathbf{B}\mathbf{A}^{-1/2}$ have the same nonzero eigenvalues; these eigenvalues must be real due to the symmetry of the latter matrix.

1.6.3 When simultaneous diagonalization of symmetric matrices fails

Suppose that $\mathbf{A}$ and $\mathbf{B}$ are symmetric matrices of the same order such that $\mathbf{A}^{-1}\mathbf{B}$ has complex eigenvalues. It is possible to transform both matrices to a block-wise diagonal form using a procedure by Uhlig [115]. First consider $n = 2$. Compute the eigendecomposition $\mathbf{A}^{-1}\mathbf{B} = \mathbf{K}\mathbf{\Lambda}\mathbf{K}^{-1}$ and define $\mathbf{T} = [\text{real}(\mathbf{k}_1)|\text{imag}(\mathbf{k}_2)]$, where $\mathbf{k}_i$ is the i-th column of $\mathbf{K}$ $(i = 1, 2)$. Then

$$[\mathbf{T}'\mathbf{A}\mathbf{T}|\mathbf{T}'\mathbf{B}\mathbf{T}] = \left[\begin{array}{cc|cc} a & b & c & d \\ b & -a & d & -c \end{array}\right]. \tag{1.18}$$

In general, suppose that $\mathbf{A}$ and $\mathbf{B}$ are symmetric of order n such that $\mathbf{A}^{-1}\mathbf{B}$ has n_1 real eigenvalues and $n_2 = n - n_1$ complex eigenvalues. Write the eigendecomposition $\mathbf{A}^{-1}\mathbf{B} = \mathbf{K}\mathbf{\Lambda}\mathbf{K}^{-1}$ such that the first n_1 eigenvalues are real and the last n_2 eigenvalues are complex conjugate and placed contiguously. Define

$$\mathbf{T} = \Big[\mathbf{k}_1|\cdots|\mathbf{k}_{n_1}|\text{real}(\mathbf{k}_{n_1+1})|\text{imag}(\mathbf{k}_{n_1+2})|\cdots|\text{real}(\mathbf{k}_{n-1})|\text{imag}(\mathbf{k}_n)\Big]. \tag{1.19}$$

Then $[\mathbf{T}'\mathbf{A}\mathbf{T}|\mathbf{T}'\mathbf{B}\mathbf{T}]$ has the form

$$\left[\begin{array}{ccccc|ccccc} \ddots & & & & & \ddots & & & & \\ & r_i & & & & & s_i & & & \\ & & \ddots & & & & & \ddots & & \\ & & & a_j & b_j & & & & c_j & d_j \\ & & & b_j & -a_j & & & & d_j & -c_j \\ & & & & \ddots & & & & & \ddots \end{array}\right],$$

$i = 1, \ldots, n_1$, $j = 1, \ldots, n_2/2$.

1.7 Principal Components Analysis (PCA)

Principal Components Analysis (PCA), Pearson [71], is a popular statistical method. For a given matrix with scores of a set of individuals on a set of variables, PCA finds a reduced set of *components*. These components are linear combinations of the original variables that capture most of the information contained in the original data. This allows to reduce the dimension of the data set while losing the smallest possible amount of information.

Formally, let $\mathbf{X}$ be an $I \times J$ data matrix with standardized scores of I individuals on J variables. A PCA decomposition with R components reads

$$\mathbf{X} = \mathbf{A}\mathbf{B}' + \mathbf{E}, \tag{1.20}$$

where $\mathbf{A}$ is an $I \times R$ matrix, $\mathbf{B}$ is an $J \times R$ matrix and $\mathbf{E}$ is the residual matrix. $\mathbf{A}$ is called the *component score matrix*; its columns are the *principal components*. $\mathbf{B}$ is called the *loading matrix*; the loadings are the weights that allow to reconstruct the original variables as linear combinations of the principal components. A different way to present the PCA decomposition is by the expression

$$\mathbf{X} = \sum_{i=1}^{R} \mathbf{a}_i \mathbf{b}_i' + \mathbf{E}, \tag{1.21}$$

which shows that PCA tries to fit $\mathbf{X}$ as a sum of R rank-one matrices (the outer product of two vectors).

The criterion to find the component scores and the loadings is to minimize the sum of squared residuals,

$$\min\|\mathbf{E}\|^2 = \min\|\mathbf{X} - \mathbf{A}\mathbf{B}'\|^2, \tag{1.22}$$

where $\|\cdot\|^2$ denotes the Frobenius norm (i.e., the sum of squares of all matrix entries). Recalling that the truncated SVD of $\mathbf{X}$ minimizes the same criterion (Eckart and Young [20], see Section 1.5), we can exhibit a PCA decomposition of $\mathbf{X}$ with R components in terms of its truncated SVD decomposition: from $\mathbf{X} = \mathbf{PDQ}'$ define $\mathbf{A}_1 = I^{1/2}\mathbf{P}_R (= I^{1/2}\mathbf{X}\mathbf{Q}_R\mathbf{D}_R^{-1})$, $\mathbf{B}_1' = I^{-1/2}\mathbf{D}_R\mathbf{Q}_R'$. This expresses the components as uncorrelated and normalized linear combinations of the variables of $\mathbf{X}$: $I^{-1}\mathbf{A}_1'\mathbf{A}_1 = \mathbf{I}_R$. Another way of computing a PCA decomposition can be given as follows: write the eigendecomposition of the correlation matrix $\mathbf{R} = \mathbf{K}\boldsymbol{\Lambda}\mathbf{K}'$ ($\mathbf{R} = I^{-1}\mathbf{X}'\mathbf{X}$) with the eigenvalues arranged in decreasing order in $\boldsymbol{\Lambda}$, and take $\mathbf{A}_2 = \mathbf{XM}$, $\mathbf{B}_2 = \mathbf{RM}$, where $\mathbf{K}_R$ holds the first R columns of $\mathbf{K}$ and $\mathbf{M} = \mathbf{R}^{-1/2}\mathbf{K}_R$. Component matrices $\mathbf{A}_1$ (resp. $\mathbf{B}_1$) and $\mathbf{A}_2$ (resp. $\mathbf{B}_2$) can be seen to be equal up to column signs.

The minimum number of components that give perfect fit is the number of nonzero singular values of $\mathbf{X}$. This implies that the minimum number of PCA-components required to have perfect fit is equal to the *rank* of $\mathbf{X}$. An immediate consequence is that there is no need to take a number of components R larger than the number of variables J. Usually the number of components R maintained is much smaller than the number of variables. Retaining more components than needed is commonly known as *overfactoring*, and it should be avoided.

A measure of the amount of information that a PCA solution is able to recover from the initial data is given by $\sum_{i=1}^{R} \lambda_i/\mathrm{tr}(\boldsymbol{\Lambda})$. The remaining, $\sum_{i=R+1}^{J} \lambda_i/\mathrm{tr}(\boldsymbol{\Lambda})$, is the part of information in the residuals, that is, the part not accounted for by the decomposition.

There is more than one solution possible. Notice that $\mathbf{AB}'$ is equal to $(\mathbf{AT})(\mathbf{BT}'^{-1})'$, for any $R \times R$ nonsingular $\mathbf{T}$. Thus, without loss of fit, we can take as components any base of the column space of $\mathbf{A}$, as long as we perform a compensation on the loading matrix. This consists of a transformation of the original components (columns of $\mathbf{A}$) into new ones (columns of $\mathbf{AT}$), thus a change of base is performed. Matrix $\mathbf{T}$ is also referred to as *rotation* matrix when its columns are unit length. The new components are referred to as *rotated principal components*. The rotation is *orthogonal* or *oblique* depending on $\mathbf{T}$ being orthogonal or not. The entries of the loading matrix can be regarded as the signed lengths of projections of the variables on the vector space generated by the components, when the columns of $\mathbf{A}$ have unit length.

The property of *non-uniqueness* of PCA decompositions usually implies that the researcher needs a post-processing step, in order to find a suitable rotation that facilitates the interpretation of the solution. There are several options of rotations available, depending on the desired goal.

1.8 Three-way arrays

The contents of this thesis revolve around higher order structures that we refer to as *arrays* (also known as *tensors*). Although the concept of array is applicable in the n-way sense, our focus will remain on three-way arrays. As a general rule in this thesis, the word "array" should be interpreted as "three-way array".

Three-way arrays are straightforward generalizations of matrices to the three-dimensional world. They are structures of three *ways* or *modes* (we will avoid the use of the word "dimension" in this context). Usually we settle which are the *first*, *second* and *third* modes, or modes A, B and C, respectively. This choice does not intend to categorize the modes in order of importance; the sole intention is to clarify notation and simplify computations. For instance, for a statistical data array of scores of individuals on several variables in different occasions, it is standard to take the first mode as the set of individuals, the second mode as the set of variables and the third mode as the set of occasions. The number of elements in each of the three ways defines the *order* of the array.

We will denote arrays with underlined upper case bold-face font: $\underline{\mathbf{A}}$, $\underline{\mathbf{X}}$, $\underline{\boldsymbol{\Lambda}}$.

An array $\underline{\mathbf{X}}$ of order $I \times J \times K$ has I entities or levels for mode A, J entities for mode B and K entities for mode C. We will only consider real entries, so $\underline{\mathbf{X}}$ will always be an element in $\mathbb{R}^{I\times J\times K}$. Arrays are also known by the more general term *tensor*.

Arrays are important *per se* as mathematical objects. They have been subject of research due to their inherent theoretical properties. But practical applications

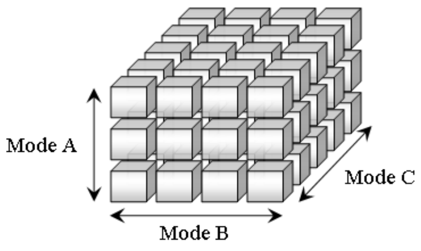

Figure 1.1: Representation of a three-way array ($I = 3$, $J = 4$, $K = 5$).

of arrays are also relevant. The most obvious examples are the numerous statistical collections of data that typically comprise three different sources of variation. For example (Kiers and Van Mechelen [43]), measurements on various anxiety scales of a number of individuals in various situations; data on the strength of various symptoms observed in various patients by a number of clinicians; data on the importance of various job requirements for various jobs, according to different job analysts; and positron-emission tomography scan data representing different areas of the brain, measured for various individuals performing a number of different mental tasks.

A generic element of $\underline{\mathbf{X}}$ will be denoted by x_{ijk}, for $i = 1, \ldots, I$, $j = 1, \ldots, J$, $k = 1, \ldots, K$. These elements are organized in terms of sets of matrices, usually referred to as *slices* or *slabs*: the *horizontal*, *lateral* and *frontal* slices (see Figure 1.2). Slices are characterized for having one mode fixed while the other two modes run over their entities. The choice of modes A, B and C is usually made so that one looks preferably at the frontal slices of the array, either for computational or for depiction purposes. The frontal slices are characterized for having their third mode fixed. This way, if we refer to "slice k of $\underline{\mathbf{X}}$" we mean the k-th frontal slice of order $I \times J$ and we shall denote it by $\mathbf{X}_k$.

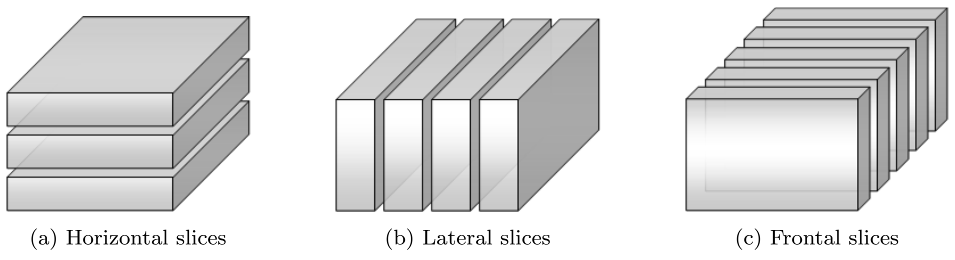

Figure 1.2: Three-way array, cut into slices.

Alternatively, $\underline{\mathbf{X}}$ can be viewed as a set of vectors or *fibers* or *tubes* (see Figure 1.3).

These fibers can be taken in the vertical, horizontal and depth direction. Fibers are characterized for having two modes fixed while the third mode runs over its entities.

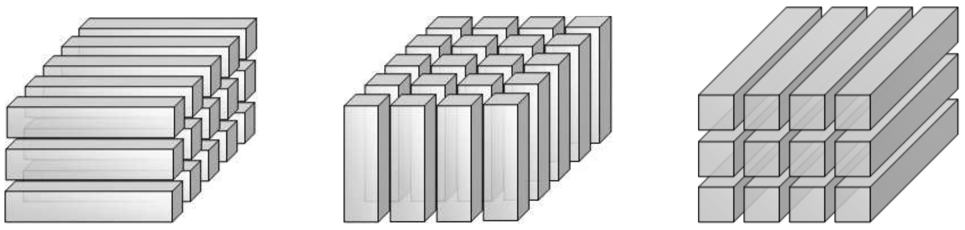
(a) Horizontal fibers (b) Vertical fibers (c) Depth fibers

Figure 1.3: Three-way array, cut into fibers.

It is common practice to rearrange the entries of an array into matrix-shape. One way to do this is, known as *matricizing* or *unfolding*, concatenates the slices of different levels in one mode side by side. Typically an array is unfolded in the third mode, i.e., by concatenating the frontal slices side by side (figure 1.4). This allows to write the array as an $I \times JK$ supermatrix $\mathbf{X}_a = [\mathbf{X}_1|\cdots|\mathbf{X}_K]$. Similarly, the horizontal slices can be stacked side by side in a $J \times KI$ supermatrix $\mathbf{X}_b$, and the lateral slices can be stacked side by side in a $K \times IJ$ supermatrix $\mathbf{X}_c$.

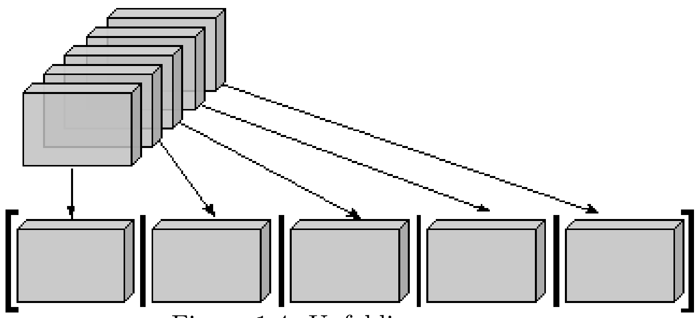
Figure 1.4: Unfolding an array.

A different way to rearrange the entries of an array uses the vec operator applied to the frontal slices of $\underline{\mathbf{X}}$, so $\underline{\mathbf{X}}$ can be rearranged in an $IJ \times K$ matrix $\mathbf{X}_{\text{vec}} = [\text{vec}(\mathbf{X}_1)|\cdots|\text{vec}(\mathbf{X}_K)]$. Still another way of rearranging the entries of $\underline{\mathbf{X}}$ is by stacking all vertical fibers one below the other: $\mathbf{x}_a = \text{vec}(\mathbf{X}_{\text{vec}})$. This way the array can be regarded in vector shape. Forms analogous to $\mathbf{X}_a$, $\mathbf{X}_{\text{vec}}$ and $\mathbf{x}_a$ can also be computed for the other two possible directions.

There is a special kind of transformation for arrays which is referred to as *slice-mixing* throughout this thesis. Slice mixing an array in the depth direction, for example, consists of replacing the frontal slices by an invertible set of linear combinations of them. This kind of operation is specially useful because it allows to transform an array (into "simpler" forms, for example) while preserving some of its properties. This will be further discussed in Section 3.3.2 (p. 43).

1.8.1 The rank of an array

It is well-known that the rank of a matrix is defined as the smallest number of rank-one matrices that generate the matrix as their sum. A rank-one matrix is the outer product of two vectors, therefore it has all rows proportional and all columns proportional. Moreover, pre- or postmultiplying a matrix by an invertible matrix is rank-preserving. This means that we can modify a matrix by taking any invertible linear combination of rows and/or columns and still preserve the rank.

A similar situation occurs with three-way arrays. The rank of a three-way array $\underline{\mathbf{X}}$ is defined as the smallest number of rank-one arrays that generate $\underline{\mathbf{X}}$ as their sum, Hitchcock [29, 30]. A rank-one array is the outer product of three vectors, so the slices are proportional in each of the three possible directions. Also, any invertible linear combination of the slices of the array, taken in any of the three possible ways, does not affect the rank of the array (an argument supporting this fact can be found on p. 43).

However, the correspondence between the matrix and the array situations ends here. In fact, the rank of three-way arrays has some very particular features, Kruskal [51]. For example, consider the concepts of *typical rank* and *maximum rank*. The typical rank of a matrix/array order is the rank that occurs with positive probability, when the elements of the matrix/array are sampled from a continuous distribution. The maximum rank is the highest rank possible that can occur amongst all matrices/arrays of a given order. Both the typical and the maximum rank of a $p \times q$ matrix ($p > q$) are q. For three-way arrays, however, the typical rank and maximum rank need not be equal. Moreover, typical rank may be twofold. For example, a $2 \times 2 \times 2$ array has maximum rank 3 but typical rank 2 or 3 (both ranks occur with positive probability).

It is not easy to determine the rank of an array. A practical approach to this problem will be addressed in Section 2.3; see also Choulakian [14] for a recent development. Although general formulas for maximal or typical rank of any array format are still absent, some results are available for some formats. Some references in this field are Kruskal [49, 51], Strassen [92], Ten Berge and Kiers [103], Catalisano, Geramita and Gimigliano [12], Lathauwer, Moor and Vandewalle [18], Ten Berge [94, 96, 97, 98], Ten Berge and Stegeman [110], Friedland [23], Comon, Ten Berge, De Lathauwer and Castaing [15], Stegeman and Comon [87] and references therein.

Chapter 2

Three-Way Component Analysis

2.1 Introduction

Attempts to develop generalizations of Factor Analysis (FA) and Principal Component Analysis (PCA) to higher dimensions go back to Hitchcock [29, 30]. Tucker [114], Carroll and Chang [11] and Harshman [24] rediscovered Hitchcock's idea of tensor decomposition, as they presented some of the most important models used nowadays.

Tucker's model is the more general model that we present in this thesis, in the sense that CP and INDSCAL, the other models covered in this thesis, can be regarded as constrained versions of this model. There are several names associated to this method that are spread over the literature: three-way PCA, Tucker-3 model, three-mode PCA, three-mode Factor Analysis (3MFA), Tucker3 PCA, 3PCA. In this thesis we will use the acronym "3PCA". This model will be introduced in the second section of this Chapter.

Carroll and Chang's CANDECOMP and Harshman's PARAFAC are the same model, proposed independently. Therefore, we will refer to it as the CANDECOMP/ PARAFAC or simply the CP model. In the third section we will present the CP model, as well as its relation to 3PCA.

A model closely related to the CP model and suited for arrays with symmetric slices in one direction is INDSCAL. This model will be presented in Section 2.4.

In Section 2.5 we discuss how models can be derived by constraining the 3PCA model.

There are more models available in the literature (example: Tucker2, Tucker1, PARAFAC2, CANDELINC). These models were not object of research during this project, so they were not included in this dissertation.

2.2 3PCA

The 3PCA model has been proposed by Tucker [114]. Given an array $\underline{\mathbf{X}}$ of order $I \times J \times K$, 3PCA determines *component matrices* for each of the three modes (individuals, variables and occasions) and a three-way array called the *core array*, which holds the weights for the joint impact of any triple of components (one from each mode). The model can be described by

$$\underline{\mathbf{X}} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr}(\mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r) + \underline{\mathbf{E}}, \tag{2.1}$$

where $\mathbf{a}_p$, $\mathbf{b}_q$, $\mathbf{c}_r$ are columns of the component matrices for individuals $\mathbf{A}$ $(I \times P)$, for variables $\mathbf{B}$ $(J \times Q)$ and for occasions $\mathbf{C}$ $(K \times R)$, respectively, g_{pqr} is an element of the $P \times Q \times R$ core array $\underline{\mathbf{G}}$, and $\underline{\mathbf{E}}$ is the array of residuals.

We can write (2.1) elementwise:

$$x_{ijk} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr}a_{ip}b_{jq}c_{kr} + e_{ijk}. \tag{2.2}$$

Alternatively, an equivalent way to write the 3PCA model is by using the Kronecker product. Matricizing $\underline{\mathbf{X}}$, $\underline{\mathbf{E}}$ and $\underline{\mathbf{G}}$ as $\mathbf{X}_a = [\mathbf{X}_1|\cdots|\mathbf{X}_K]$, $\mathbf{E}_a = [\mathbf{E}_1|\cdots|\mathbf{E}_K]$ and $\mathbf{G}_a = [\mathbf{G}_1|\cdots|\mathbf{G}_R]$, respectively, we have

$$\mathbf{X}_a = \mathbf{A}\mathbf{G}_a(\mathbf{C}' \otimes \mathbf{B}') + \mathbf{E}_a. \tag{2.3}$$

Applying the vec operator to both sides of (2.3) and using property (1.3), we can derive a vectorized version of the 3PCA model,

$$\mathbf{x}_a = (\mathbf{C} \otimes \mathbf{B} \otimes \mathbf{A})\mathbf{g}_a + \mathbf{e}_a, \tag{2.4}$$

where $\mathbf{x}_a = \text{vec}(\mathbf{X}_{\text{vec}})$, $\mathbf{g}_a = \text{vec}(\mathbf{G}_{\text{vec}})$ and $\mathbf{e}_a = \text{vec}(\mathbf{E}_{\text{vec}})$.

Yet another way of presenting the decomposition is to display one slice at a time:

$$\mathbf{X}_k = \mathbf{A}\left(\sum_{r=1}^{R} c_{kr}\mathbf{G}_r\right)\mathbf{B}' + \mathbf{E}_k, \tag{2.5}$$

for $k = 1, \ldots, K$.

Notice that the 3PCA decomposition is fully symmetric in $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. This means that the component matrices can switch places, as long as we switch the modes and the array of residuals accordingly.

The parameters of 3PCA are usually estimated by minimizing the sum of squared residuals for fixed number of components in each mode,

$$\|\underline{\mathbf{E}}\|^2 = \left\| \underline{\mathbf{X}} - \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} (\mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r) \right\|^2 , \tag{2.6}$$

see TUCKALS-3 procedure in Kroonenberg and De Leeuw [48]. Some enhancements are available in Weesie and Van Houwelingen [116], Ten Berge [101], Andersson and Bro [1], Paatero and Andersson [70], Pravdova, Estienne, Walczak and Massart [72]. Several computational implementations are available for use, see for example the N-way Toolbox for MATLAB (Andersson and Bro [2]) and 3WayPack by Pieter Kroonenberg (http://three-mode.leidenuniv.nl/).

The 3PCA model can be seen as a generalization of PCA to three-way arrays. In PCA a matrix is decomposed as a sum of outer product of two vectors, see (1.21). Similarly, 3PCA decomposes an array as a sum of outer products of three vectors, one per mode (see (2.1)). However, unlike PCA, all impacts of components are taken into account in 3PCA. In this sense we can say that 3PCA has a much richer structure than PCA, since 3PCA allows more interactions between components.

It is possible to extend the 3PCA model to higher dimensions, although this was not the subject of this thesis. The interested reader may refer to Lastovicka [55], Kapteyn, Neudecker and Wansbeek [35], Sidiropoulos and Bro [79].

2.3 CANDECOMP/PARAFAC (CP)

The CP model is a special case of 3PCA. It was proposed independently by Carroll and Chang [11] and Harshman [24]. In the CP model the number of components is the same for each mode, say R. The core array is completely constrained to be a superdiagonal $R \times R \times R$ array

$$\left[\begin{array}{cccc|cccc|c|cccc} g_{111} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & g_{222} & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & & 0 & 0 & \cdots & g_{RRR} \end{array} \right] . \tag{2.7}$$

Therefore, the CP model can be written as follows (compare with (2.1) and (2.2)):

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} g_{rrr}(\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r) + \underline{\mathbf{E}} \tag{2.8}$$

$$x_{ijk} = \sum_{r=1}^{R} g_{rrr} a_{ir} b_{jr} c_{kr} + e_{ijk}. \tag{2.9}$$

Another usual way to present the model is by rewriting the core array as the constant superdiagonal $R \times R \times R$ array

$$\mathbf{I}_a = \left[\begin{array}{cccc|cccc|c|cccc} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & & 0 & 0 & \cdots & 1 \end{array}\right]. \tag{2.10}$$

The parameters g_{rrr} are absorbed by the component matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$. This decreases the number of parameters to be estimated. Using the specific form (2.10), definitions (2.1)-(2.5) of 3PCA may be written for the CP model as follows:

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} (\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r) + \underline{\mathbf{E}} \tag{2.11}$$

$$x_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} + e_{ijk}. \tag{2.12}$$

$$\mathbf{X}_a = \mathbf{A}\mathbf{I}_a(\mathbf{C}' \otimes \mathbf{B}') + \mathbf{E}_a = \mathbf{A}(\mathbf{C} \odot \mathbf{B})' + \mathbf{E}_a \tag{2.13}$$

$$\mathbf{x}_a = (\mathbf{C} \odot \mathbf{B} \odot \mathbf{A})\mathbf{1}_R + \mathbf{e}_a \tag{2.14}$$

$$\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}' + \mathbf{E}_k, \tag{2.15}$$

where $\mathbf{C}_k$ is a diagonal matrix containing the elements of row k of $\mathbf{C}$ in the diagonal. Another useful formulation for $\mathbf{X}_{\text{vec}} = [\text{vec}(\mathbf{X}_1)|\cdots|\text{vec}(\mathbf{X}_k)]$ is

$$\mathbf{X}_{\text{vec}} = (\mathbf{B} \odot \mathbf{A})\mathbf{C}' + \mathbf{E}_{\text{vec}}. \tag{2.16}$$

There are several algorithms available for the CP model. However, it should be made clear that not all the methods are equivalent. Before applying any of the methods available, the researcher must certify which is the model that better suits his needs. For more details please refer to Hopke, Paatero, Jia, Ross and Harshman [31], Faber, Bro and Hopke [21], Tomasi and Bro [113].

The common optimization criterion is the least squares loss function, in which the estimation of the component matrices **A**, **B**, **C** attempts to minimize

$$\|\underline{\mathbf{E}}\|^2 = \left\| \underline{\mathbf{X}} - \sum_{r=1}^{R} g_{rrr}(\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r) \right\|^2 , \tag{2.17}$$

where $\|\cdot\|^2$ denotes the Frobenius norm. Some of the available algorithms to fit the CP model do not follow this rule. In this dissertation we will settle for the least squares criterion (2.17) as the optimization function to work with.

The optimization problem of the CP model can be re-stated as follows: fitting the CP model consists of finding the best rank-R approximation of $\underline{\mathbf{X}}$, i.e.

$$\begin{aligned} &\text{minimize } \|\underline{\mathbf{X}} - \underline{\mathbf{Y}}\| \\ &\text{subject to } \underline{\mathbf{Y}} \in \mathcal{D}_R, \end{aligned} \tag{2.18}$$

where $\mathcal{D}_R$ is the set of $I \times J \times K$ arrays of rank R or less

$$\mathcal{D}_R = \{\underline{\mathbf{Y}} : I \times J \times K \text{ with rank } \leqslant R\}, \tag{2.19}$$

Stegeman [83]. This type of formulation of the CP problem is very useful when addressing the issue of degeneracy, see Stegeman [82, 83] and Chapter 4 for more details.

A property intrinsic to the CP model is that the smallest number of components that allows a perfect fit ($\underline{\mathbf{E}} \equiv \underline{\mathbf{0}}$) equals the rank of the array. In fact, (2.11) gives a decomposition of $\underline{\mathbf{X}}$ as a sum of rank-one arrays when $\underline{\mathbf{E}}$, the array of residuals, is zero. This feature of the CP model can be used in practice to assess the rank of an array: run CP with the smallest number of components such that the sum of squared residuals is zero; the rank of the array will equal the number of components used in this CP decomposition. There is however some uncertainty associated to this approach: In practice the fit is never exactly zero, so one might question how "perfect" the fit is. A different approach that avoids such subjective considerations can be found in Choulakian [14].

The rank of an array can also be defined as the smallest number R such that every frontal slice can be written as a linear combination of a single collection of R matrices, Kruskal [51]. In fact, when $\underline{\mathbf{E}} = 0$ we see from (2.15) that

$$\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}' = c_{k1}(\mathbf{a}_1\mathbf{b}_1') + \cdots + c_{kR}(\mathbf{a}_R\mathbf{b}_R'), \tag{2.20}$$

which shows that any frontal slice $\mathbf{X}_k$ can be written as a linear combination of R matrices $\mathbf{a}_1\mathbf{b}_1'$, …, $\mathbf{a}_R\mathbf{b}_R'$. The number R is, moreover, invariant to the direction of the array.

The CP model shares some features with PCA. Indeed, the CP model decomposes an array as a sum of outer products of three vectors, one per mode. Each component in each mode is used only once in the decomposition. This is exactly what happens with PCA, compare (1.21) for PCA and (2.11) for the CP model. Also, in a similar fashion as for matrices in PCA, the minimum number of rank-one arrays for which CP performs a full decomposition equals the rank of the array. In spite of these similarities, there are some fundamental differences between both models. For example, an iterative algorithm is needed to solve the CP model, whereas a closed form solution for PCA exists (for example using the SVD, Eckart and Young [20]). Also, usually it is not possible to rotate a CP decomposition without losing fit, see Chapter 3. Furthermore, scaling and centering in the multi-way case is not as simple as for two-way data, see for example Kroonenberg [46], Harshman and Lundy [28], Bro [5], Kiers [41].

2.4 INDSCAL

INDSCAL (Carroll and Chang [11]) is a model related to the CP model that has been specifically designed for arrays with symmetric slices in one direction. Consider an $I \times I \times K$ array $\underline{\mathbf{X}}$ with K symmetric frontal slices $\mathbf{X}_k$ of order $I \times I$, $k = 1, \ldots, K$. INDSCAL decomposes the slices as

$$\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{A}' + \mathbf{E}_k, \quad k = 1, \ldots, K, \tag{2.21}$$

with $\mathbf{C}_k$ diagonal and nonnegative. Therefore, INDSCAL is the CP model with constraints $\mathbf{A} = \mathbf{B}$ and entries of $\mathbf{C}$ nonnegative. In this dissertation we will ignore the constraint on $\mathbf{C}$. All results to be discussed under the INDSCAL model (Chapter 7) were worked out under this principle.

2.5 Constrained 3PCA

Depending on the data that need to be analysed and on the needs of the researcher, it might be useful to impose some constraints on a 3PCA solution. Many models can be derived from 3PCA by imposing restrictions on the component matrices and/or the core array. The CP model is the most well-known example, but others do exist. There is a myriad of ways of constraining a 3PCA solution, but not all of them concern well defined models.

When imposing constraints one should be careful enough to introduce adequate and active restrictions. By adequate we mean that the constraints should reflect conjectures or properties inherent to the situation under study. By active we mean that the constraints should really define a truthful model and not just an arithmetic artefact. For example, to constrain the component matrices of a 3PCA decomposition to be columnwise orthogonal can be regarded as an inactive constraint, because usually any 3PCA solution can be transformed into one where this condition is met with no loss of fit involved. In this case we would say that the model is *trivial.* Trivial models are just arithmetic artefacts and do not add anything new as a tool of analysis, and should therefore be avoided. We will discuss this matter further in Chapters 5 and 6.

We refer to Ten Berge and Smilde [109] and Ten Berge [98] as examples of studies of constrained 3PCA models. In both papers it is proven that two models that arose in Chemometrics are, indeed, non-trivial. In both cases arguments involving the concept of (typical) rank were used.

2.6 Discussion

In this Chapter we presented the most important three-way models that are used throughout this thesis. Tucker's 3PCA model, as well as constrained 3PCA models in general, will be referred to in Chapters 5 and 6. There we will argue how techniques that allow assessing simplicity of arrays can be applied under such models. CP and INDSCAL, on the other hand, are in the core of Chapter 7. In that Chapter we will analyse the differential structure of the loss functions associated to each model.

3PCA and PARAFAC are the most common models used for decomposing a three-way array. INDSCAL is used in the context of arrays with symmetric slices.

Some interesting properties regarding these models concern the uniqueness of the decompositions and the possibility of finding such decompositions in “simple form”. The following chapters are devoted to study these features for the models presented in this Chapter.

Chapter 3

Uniqueness in 3PCA and CP

3.1 Introduction

A well-known property of PCA is the indeterminacy of its decomposition. We can replace the principal components by nonsingular linear combinations of them, as long as we compensate this transformation in the loading matrix. No information is lost after rotating the principal components.

A similar phenomenon occurs in 3PCA. There is freedom to *rotate* each of the component matrices, as long as a counter-operation is applied to the core array $\underline{\mathbf{G}}$. This will be explained in the second section of this Chapter.

In the CP model the situation is not the same. The specificity of the superdiagonal core array $\underline{\mathbf{I}}$ withdraws most of the indeterminacy that existed in the 3PCA model. Under mild conditions it can be shown that a CP solution is, in fact, *essentially unique*. What is meant by "essentially unique" and which are these "mild conditions" is to be discussed in the third part of the Chapter. Operations that can be applied to a CP decomposition while preserving the uniqueness property will also be discussed.

The issue of uniqueness is specially important in this dissertation when we reach Chapter 7. As will be discussed in that Chapter, the analysis of the eigenstructure of the second-order derivatives of the loss functions for the CP and INDSCAL models can be affected by the presence of (non-)unique decompositions. Therefore, at this point it is important to understand what uniqueness is and how it can be dealt with.

3.2 Non-uniqueness in 3PCA

Since 3PCA was presented it has been clear that a solution can always be transformed without loss of fit, Tucker [114]. To see this, we recover the matricized formulation of the model,

$$\mathbf{X}_a = \mathbf{A}\mathbf{G}_a(\mathbf{C}' \otimes \mathbf{B}') + \mathbf{E}_a. \tag{3.1}$$

If we consider nonsingular matrices $\mathbf{S}$ $(P \times P)$, $\mathbf{T}$ $(Q \times Q)$ and $\mathbf{U}$ $(R \times R)$, we can rewrite the fitted part in the last expression as

$$\begin{aligned} \mathbf{A}\mathbf{G}_a(\mathbf{C}' \otimes \mathbf{B}') &= \mathbf{A}(\mathbf{S}')^{-1}\mathbf{S}'\mathbf{G}_a(\mathbf{U} \otimes \mathbf{T})(\mathbf{U}^{-1} \otimes \mathbf{T}^{-1})(\mathbf{C}' \otimes \mathbf{B}') \\ &= \underbrace{\mathbf{A}(\mathbf{S}')^{-1}}\mathbf{S}'\mathbf{G}_a(\mathbf{U} \otimes \mathbf{T})(\underbrace{\mathbf{U}^{-1}\mathbf{C}'} \otimes \underbrace{\mathbf{T}^{-1}\mathbf{B}'}). \end{aligned} \tag{3.2}$$

This shows how we can change the component matrices from $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ to $\mathbf{A}(\mathbf{S}')^{-1}$, $\mathbf{B}(\mathbf{T}')^{-1}$, $\mathbf{C}(\mathbf{U}')^{-1}$, respectively, and maintain the fit. We just need to change the core array accordingly, from $\mathbf{G}_a = [\mathbf{G}_1|\cdots|\mathbf{G}_R]$ to $\mathbf{S}'\mathbf{G}_a(\mathbf{U} \otimes \mathbf{T})$. Other useful ways to present the transformed core array are in vec- and vec*-form: $(\mathbf{T}' \otimes \mathbf{S}')\mathbf{G}_{\text{vec}}\mathbf{U}$ and $(\mathbf{S}' \otimes \mathbf{T}')\mathbf{G}_{\text{vec}^*}\mathbf{U}$, respectively. These transformations of 3PCA solutions are known as *Tucker transformations*, Tucker [114].

As mentioned before, postmultiplying the component matrices of a 3PCA decomposition by invertible transformation matrices can always be counter-balanced with an appropriate compensation in the core array. This means that the component matrices can be transformed without changing the slices of the original array. Such a transformation can therefore be regarded as array-preserving. The process can also be inversely considered: It is possible to transform the core array in any of the three directions possible, as long as a compensating operation is applied to the component matrices in the corresponding directions.

In practical applications we assume $P \ll I$, $Q \ll J$, $R \ll K$. More specifically, Tucker [114] explains how Tucker transformations can be used to transform all component matrices into full column-rank matrices, if needed. For instance, suppose that $\mathbf{A}$ has linearly dependent columns. Then it is possible to find an invertible matrix $\mathbf{S}$ such that $\mathbf{A}(\mathbf{S}')^{-1}$ has one or more columns containing all zero entries. These zero columns of the transformed $\mathbf{A}$ can be discarded along with the corresponding rows of the transformed core array, thus reducing the column order of $\mathbf{A}$ and row order of $\mathbf{G}_a$ (Tucker [114], p. 288). Since we can proceed this way for all component matrices, we conclude that it can be always assumed that $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ have full column ranks.

The previous reasoning has a direct consequence: there is no point in considering 3PCA solutions with $P \times Q \times R$ core arrays where $P > QR$ $(Q > PR,\ R > PQ)$, which is known as *overfactoring.* To see this, suppose $P > QR$. Then $\mathbf{G}_a$ has linearly dependent rows, so it is possible to find an invertible $\mathbf{S}$ such that $\mathbf{S}'\mathbf{G}_a$ has one or more zero rows. These rows may be discarded, as well as the corresponding columns of $\mathbf{A}(\mathbf{S}')^{-1}$. So we conclude that we need only consider 3PCA solutions with core arrays such that $P \leqslant QR$.

We can still impose an extra condition on the component matrices to be found by a Tucker transformation, namely, that they are to be columnwise orthonormal. In fact, it suffices to find the appropriate $\mathbf{S}$, $\mathbf{T}$, $\mathbf{U}$ such that $\mathbf{A}(\mathbf{S}')^{-1}$, $\mathbf{B}(\mathbf{T}')^{-1}$, $\mathbf{C}(\mathbf{U}')^{-1}$ are orthonormal. We illustrate how to do this for component matrix $\mathbf{A}$. Define $\mathbf{S} = \mathbf{K}\mathbf{\Lambda}^{1/2}$, where $\mathbf{A}'\mathbf{A} = \mathbf{K}\mathbf{\Lambda}\mathbf{K}'$ is an eigendecomposition. Then $\mathbf{A}(\mathbf{S}')^{-1}$ is columnwise orthonormal, because

$$(\mathbf{A}(\mathbf{S}')^{-1})'(\mathbf{A}(\mathbf{S}')^{-1}) = \mathbf{\Lambda}^{-1/2}\mathbf{K}^{-1}\mathbf{A}'\mathbf{A}(\mathbf{K}')^{-1}\mathbf{\Lambda}^{-1/2} = \mathbf{I}_P. \tag{3.3}$$

The same applies to $\mathbf{B}$ and $\mathbf{C}$.

Alternatively, one can use the freedom of rotation in 3PCA to find specific form representations of the core array, while the component matrices are counter-transformed. For instance, the core array can be made orthogonal in the three directions. More precisely, if $\mathbf{G}_a$, $\mathbf{G}_b$ and $\mathbf{G}_c$ are full row rank then they can be jointly transformed to orthogonality, in the sense that $\mathbf{G}_a\mathbf{G}_a'$, $\mathbf{G}_b\mathbf{G}_b'$ and $\mathbf{G}_c\mathbf{G}_c'$ are proportional to an identity matrix, Ten Berge *et al.* [105]. To see this (following Weesie and Houwelingen [116]), suppose we perform an orthogonal rotation in one mode, say $\widetilde{\mathbf{G}}_a = \mathbf{T}\mathbf{G}_a$, for an orthonormal matrix $\mathbf{T}$. This is equivalent to transforming $\mathbf{G}_b$ and $\mathbf{G}_c$ into $\widetilde{\mathbf{G}}_b = \mathbf{G}_b(\mathbf{T}' \otimes \mathbf{I}_R)$ and $\widetilde{\mathbf{G}}_c = \mathbf{G}_c(\mathbf{I}_P \otimes \mathbf{T}')$, respectively. However, $\widetilde{\mathbf{G}}_b\widetilde{\mathbf{G}}'_b = \mathbf{G}_b\mathbf{G}_b'$ and $\widetilde{\mathbf{G}}_c\widetilde{\mathbf{G}}'_c = \mathbf{G}_c\mathbf{G}_c'$, which shows that an orthogonal transformation in one mode does not affect the inner products for the other modes. Therefore, all we need is to rotate $\mathbf{G}_a$, $\mathbf{G}_b$ and $\mathbf{G}_c$ to orthogonality, one at a time. This can be achieved by premultiplying $\mathbf{G}_a$ by $\mathbf{K}_a'$ (from the eigendecomposition $\mathbf{G}_a\mathbf{G}_a' = \mathbf{K}_a\mathbf{\Lambda}_a\mathbf{K}_a'$), then premultiplying $\widetilde{\mathbf{G}}_b$ by $\mathbf{K}_b'$ (from the eigendecomposition $\widetilde{\mathbf{G}}_b\widetilde{\mathbf{G}}_b' = \mathbf{K}_b\mathbf{\Lambda}_b\mathbf{K}_b'$), and finally premultiplying $\overset{\approx}{\mathbf{G}}_c$ by $\mathbf{K}_c'$ (from the eigendecomposition $\overset{\approx}{\mathbf{G}}_c\overset{\approx}{\mathbf{G}}'_c = \mathbf{K}_c\mathbf{\Lambda}_c\mathbf{K}_c'$).

Weesie and Houwelingen [116] noted that, when TUCKALS-3 is run with component matrices constrained by orthonormality, then the resulting core array is already orthogonal in the three directions. Hence, the transformation described in the previous paragraph is not needed in this situation.

3.2.1 (Non-)triviality of three-way models

As we have just discussed (see also Section 2.5), the freedom to transform a 3PCA decomposition must be taken into account. This is specially clear when defining new models, in which specific constraints are chosen. The researcher must make clear that the constraints are active, thus bringing new and meaningful properties to the model. A constrained core array that can be achieved almost surely by means of Tucker transformations does not represent a real model. It is just a mathematical artifact with little statistical meaning. Therefore, researchers interested in constructing constrained 3PCA models should aim at *non-trivial* core array models, that is, models with active constraints.

This (non)triviality of 3PCA models is an important issue to address. In Chapters 5 and 6 we will discuss some possible approaches to help avoiding problems related to triviality of models.

3.3 Uniqueness in CP

Under conditions called "mild" (e.g. Ten Berge [98], p. 17), CP has the property of *essential uniqueness*. This means that the only transformations we are allowed to perform on **A**, **B** and **C** are joint permutations and rescaling of their columns. An advantage of uniqueness is the unambiguity inherent to the decomposition. That is, the researcher does not need to decide which is the best solution possible, since there is only one possibility (assuming that the model underlies the structure of the data).

This property of CP is, in fact, in the very genesis of the method. When Harshman [24] proposed CP for the first time, he explained that one of the reasons that guided him was the "Principle of Parallel Proportional Profiles" or "Simultaneous Simple Structure" discussed by Cattell [13]. According to this principle, the factors underlying two similar sets of variables on similar population samples should be the same, being that the loadings of each factor may change proportionally. Although this principle concerns the rotation problem of two-way factor analysis, Harshman had the insight to apply it to three-way arrays. He realized that it was possible to decompose an array using "parallel proportional profiles", which would lead to unique decompositions. In other words, if there exists a true factor structure underlying the data, then more than being able to represent points in a reduced dimensional space, the orientation of the fitting axes of that space is uniquely defined. This is useful in the sense that it eliminates the need to rotate the fitting solution after its computation. The location of the axes is, therefore, an intrinsic feature of the factor decomposition itself. Many applications of the CP model in practice are based on this property, see Smilde *et al.* [80, Chapter 10] for some examples in chemometrics.

In the first part of this Section we will formally define uniqueness in the CP model. We will prove that joint permutations and rescaling of columns of the component matrices are always possible in a CP decomposition. This kind of operations will be considered to be trivial for that reason. In the second part of this Section we will present operations that can be applied to a CP decomposition while preserving the uniqueness property. These operations are specially important for us since they provide a link between the uniqueness and simplicity properties (Chapters 5 and 6). In the third and fourth parts of this Section we will discuss sufficient and necessary conditions for uniqueness to hold.

3.3.1 Definition of uniqueness

Given the array $\underline{\mathbf{X}}$ of order $I \times J \times K$, consider a CP decomposition with R components:

$$\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}' + \mathbf{E}_k, \tag{3.4}$$

for $k = 1, \dots, K$. This decomposition is said to be *(essentially) unique* if, for any other alternative CP solution with R components and equal residuals $\mathbf{E}_k$

$$\mathbf{X}_k = \mathbf{G}\mathbf{D}_k\mathbf{H}' + \mathbf{E}_k \tag{3.5}$$

we have

$$\mathbf{G} = \mathbf{A}\mathbf{\Pi}\mathbf{\Lambda}_1, \quad \mathbf{H} = \mathbf{B}\mathbf{\Pi}\mathbf{\Lambda}_2, \quad \mathbf{D} = \mathbf{C}\mathbf{\Pi}\mathbf{\Lambda}_3, \tag{3.6}$$

for some permutation matrix $\mathbf{\Pi}$ and diagonal matrices $\mathbf{\Lambda}_1$, $\mathbf{\Lambda}_2$ and $\mathbf{\Lambda}_3$ with $\mathbf{\Lambda}_1\mathbf{\Lambda}_2\mathbf{\Lambda}_3 = \mathbf{I}_R$.

Expression (3.6) means that the corresponding component matrices of both CP solutions are essentially the same, i.e., component matrices may only differ in the order of the components (due to the column permutation matrix $\mathbf{\Pi}$) and the scaling of the components (due to matrices $\mathbf{\Lambda}_i$). The permutation of the components is the same for the three pairs of component matrices, and the scaling factors for all correspondent components multiply to unity so that the overall value of the product of the components is not affected. Notice that $\mathbf{\Pi}\mathbf{\Lambda}_i$ has only one nonzero element per row and per column.

Two CP solutions that are equal up to column joint permutation and scaling are considered the same solution. This indeterminacy can not be avoided: from the rank-one decomposition $\sum_{r=1}^{R}(\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r)$ we notice that joint column permutation is just a rearrangement of the order of the summation, whereas the scaling reflects the equality

$$\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = (\boldsymbol{\Lambda}_1(r,r)\mathbf{a}_r) \circ (\boldsymbol{\Lambda}_2(r,r)\mathbf{b}_r) \circ (\boldsymbol{\Lambda}_3(r,r)\mathbf{c}_r) \tag{3.7}$$

with $\boldsymbol{\Lambda}_1(r,r)\boldsymbol{\Lambda}_2(r,r)\boldsymbol{\Lambda}_3(r,r) = 1$. We can reconfirm this fact via (2.16):

$$\begin{aligned}(\mathbf{H} \odot \mathbf{G})\mathbf{D}' &= (\mathbf{B}\boldsymbol{\Pi}\boldsymbol{\Lambda}_2 \odot \mathbf{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}_1)(\mathbf{C}\boldsymbol{\Pi}\boldsymbol{\Lambda}_3)' \\ &= (\mathbf{B}\boldsymbol{\Pi} \odot \mathbf{A}\boldsymbol{\Pi})\boldsymbol{\Lambda}_1\boldsymbol{\Lambda}_2\boldsymbol{\Lambda}_3\boldsymbol{\Pi}'\mathbf{C}' \\ &= (\mathbf{B} \odot \mathbf{A})\boldsymbol{\Pi}\boldsymbol{\Pi}'\mathbf{C}' \\ &= (\mathbf{B} \odot \mathbf{A})\mathbf{C}'.\end{aligned} \tag{3.8}$$

3.3.2 Operations that preserve uniqueness

Consider an $I \times J \times K$ array $\underline{\mathbf{X}}$ with a CP solution $\mathbf{X}_k = \tilde{\mathbf{X}}_k + \mathbf{E}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}' + \mathbf{E}_k$, for $k = 1, \ldots, K$. We will write $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ to denote this specific CP solution. As observed before, in case of uniqueness, there is a one-to-one correspondence between the component matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and the fitted part of $\underline{\mathbf{X}}$ (up to scaling and permutation of the columns of the component matrices). However, in non-uniqueness situations there is more than one triple of component matrices corresponding to $\tilde{\mathbf{X}}_k$.

We are interested in identifying algebraic operations that can be performed on $\tilde{\underline{\mathbf{X}}}$, or to its CP decomposition accordingly, such that the uniqueness property is preserved. This means that, after the operations are applied, the transformed array shares the (non-)uniqueness property with $\tilde{\underline{\mathbf{X}}}$. A useful application of these operations is to transform $\tilde{\underline{\mathbf{X}}}$ into a "simpler" array (possibly even smaller), in the sense that the inspection of the uniqueness property is easier. Notice that the operations we seek will change the slices of $\tilde{\underline{\mathbf{X}}}$, that is, these operations are not array-preserving. Although such transformations will destroy the original array entries, they may allow finding simpler arrays which are equivalent to $\tilde{\underline{\mathbf{X}}}$ in terms of the uniqueness property. This contrasts with Tucker transformations of 3PCA solutions, which transform the component matrices while preserving the array under decomposition.

We will start by considering slice mixing operations. Recall that slice mixing an array in the depth direction, for example, consists of replacing the frontal slices by an invertible set of linear combinations of them. This can be done by premultiplying the component matrix $\mathbf{C}$ by an invertible matrix. The same idea is valid for mixing horizontal and lateral slices, now applied to component matrices $\mathbf{A}$ and $\mathbf{B}$, respectively.

We have observed in Section 1.8.1 that slice mixing is an array rank preserving operation. Indeed, it is clear that transforming the component matrices does not add components (the number of columns of the component matrices remain equal). Since we only deal with invertible transformations, it is ensured that no component is lost with the transformation, because the process can be reversed.

Next we present our proof for the fact that slice mixing, besides preserving the rank, also preserve uniqueness (see also Ten Berge and Sidiropoulos [107], p. 401):

Lemma 1
Let $\underline{\mathbf{X}}$ be an array with CP solution $\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}'$, $k = 1, \ldots, K$. Let $\mathbf{S}$, $\mathbf{T}$, and $\mathbf{U}$ be nonsingular transformations yielding an array $\mathbf{Y}$ with solution $(\mathbf{SA}, \mathbf{TB}, \mathbf{UC})$. Then $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is a unique solution for $\underline{\mathbf{X}}$ if and only if $(\mathbf{SA}, \mathbf{TB}, \mathbf{UC})$ is a unique solution for $\underline{\mathbf{Y}}$.

Proof Let $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ be a unique solution for $\underline{\mathbf{X}}$. Then every other solution $(\mathbf{G}, \mathbf{H}, \mathbf{D})$ that satisfies $\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}' = \mathbf{G}\mathbf{D}_k\mathbf{H}'$ for $k = 1, \ldots, K$ is related to $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ by $\mathbf{G} = \mathbf{A}\boldsymbol{\Pi}\boldsymbol{\Lambda}_1$, $\mathbf{H} = \mathbf{B}\boldsymbol{\Pi}\boldsymbol{\Lambda}_2$ and $\mathbf{D} = \mathbf{C}\boldsymbol{\Pi}\boldsymbol{\Lambda}_3$. Now consider a transformation, first by $\mathbf{S}$ only, yielding transformed slices $\mathbf{S}\mathbf{X}_k = \mathbf{SA}\mathbf{C}_k\mathbf{B}'$, $k = 1, \ldots, K$. Suppose $(\mathbf{SA}, \mathbf{B}, \mathbf{C})$ is not a unique solution for this transformed array. Then there exists another solution $(\mathbf{M}, \mathbf{L}, \mathbf{E})$ such that at least one of these conditions is violated:

1. $\mathbf{M}$ is essentially equal to $\mathbf{SA}$,

2. $\mathbf{L}$ is essentially equal to $\mathbf{B}$,

3. $\mathbf{E}$ is essentially equal to $\mathbf{C}$.

However, from $\mathbf{S}\mathbf{X}_k = \mathbf{SA}\mathbf{C}_k\mathbf{B}' = \mathbf{M}\mathbf{E}_k\mathbf{L}'$ we have $\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}' = \mathbf{S}^{-1}\mathbf{M}\mathbf{E}_k\mathbf{L}'$, $k = 1, \ldots, K$. Because $\mathbf{M}$ is essentially equal to $\mathbf{SA}$ if and only if $\mathbf{S}^{-1}\mathbf{M}$ is essentially equal to $\mathbf{A}$, it can be seen that $(\mathbf{S}^{-1}\mathbf{M}, \mathbf{L}, \mathbf{E})$ is an alternative solution for $\underline{\mathbf{X}}$, with violation of at least one of the three essential equality conditions. So we have arrived at a contradiction. Therefore, uniqueness of $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ for $\underline{\mathbf{X}}$ is equivalent to uniqueness of $(\mathbf{SA}, \mathbf{B}, \mathbf{C})$ for the transformed array that has slices $\mathbf{SA}\mathbf{C}_k\mathbf{B}'$, $k = 1, \ldots, K$.

Using the symmetry of the CP decomposition to apply the same principle to $\mathbf{B}$ and $\mathbf{C}$, we can then say that when ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$) is an essentially unique CP solution then ($\mathbf{SA}$, $\mathbf{TB}$, $\mathbf{UC}$) is an essentially unique CP solution, for arbitrary nonsingular matrices $\mathbf{S}$, $\mathbf{T}$ and $\mathbf{U}$. The converse statement follows immediately by supposing that ($\mathbf{SA}$, $\mathbf{TB}$, $\mathbf{UC}$) is unique and then mixing the slices of the array using the inverses of $\mathbf{S}$, $\mathbf{T}$ and $\mathbf{U}$. ■

In addition, both the rank and k-rank of the component matrices are also preserved (because premultiplying a matrix by an invertible matrix preserves its rank and the linear (in)dependence of columns). So, we may use this freedom of manipulation to change the component matrices in useful ways, being assured that the rank and the uniqueness property will be preserved in the newly computed array.

We can use this property to define a new useful operation (the idea for this proof can be found in Ten Berge and Sidiropoulos [107], p. 401):

Lemma 2
Let $\underline{\mathbf{X}}$ be an array with CP solution $\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}'$, $k = 1, \ldots, K$. Transforming $\mathbf{A}$ and/or $\mathbf{B}$ and/or $\mathbf{C}$ to reduced versions of full row rank leads to a (smaller) new array whose transformed CP solution preserves the uniqueness property.

Proof We will do the proof for $\mathbf{A}$ (the argument also applies to $\mathbf{B}$ and $\mathbf{C}$ by symmetry).
Write the SVD $\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{Q}'$, with $\mathbf{P}'\mathbf{P} = \mathbf{P}\mathbf{P}' = \mathbf{I}_I$, $\mathbf{Q}'\mathbf{Q} = \mathbf{I}_{r_\mathbf{A}}$ ($r_\mathbf{A}$ denotes the rank of $\mathbf{A}$) and $\mathbf{D}$ (general) diagonal $I \times r_\mathbf{A}$ with the singular values of $\mathbf{A}$ in the diagonal (in decreasing order). Making $\mathbf{S} = \mathbf{P}'$ we have that $\mathbf{S}\mathbf{A} = \mathbf{D}\mathbf{Q}'$ has all rows zero except the first $r_\mathbf{A}$. Because of Lemma 1 we know that $(\mathbf{S}\mathbf{A}, \mathbf{B}, \mathbf{C})$ preserves uniqueness. Finally, eliminating the $I - r_\mathbf{A}$ last rows of $\mathbf{A}$ does not affect uniqueness.
$\blacksquare$

It is important to underline the power of what has been proved so far. Using Lemma 2, one can now simplify the component matrices by reducing the number of rows of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ from I, J, K to $r_\mathbf{A}, r_\mathbf{B}, r_\mathbf{C}$, and still maintain rank and uniqueness properties concerning the related CP solutions. This, in fact, means that one can simplify a given $I \times J \times K$ array to a rank- and uniqueness-equivalent $r_\mathbf{A} \times r_\mathbf{B} \times r_\mathbf{C}$ array. After this transformation, each component matrix has full row rank.

Now, using the fact that component matrices have full row rank ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ of order $r_\mathbf{A} \times R$, $r_\mathbf{B} \times R$, $r_\mathbf{C} \times R$ respectively), we can still simplify the component matrices in special cases. For instance consider, without loss of generality, that the first $r_\mathbf{A}$ columns of $\mathbf{A}$ are linearly independent. Taking the nonsingular $r_\mathbf{A} \times r_\mathbf{A}$ submatrix of $\mathbf{A}$ using its first $r_\mathbf{A}$ columns ($\mathbf{A} = [\mathbf{A}_1|\mathbf{A}_2]$), and premultiplying $\mathbf{A}$ by the inverse of that submatrix gives

$$\mathbf{A}_1^{-1}\mathbf{A} = \left[\mathbf{I}_{r_\mathbf{A}}|\mathbf{A}_1^{-1}\mathbf{A}_2\right] = \left[\begin{array}{cccc|c} 1 & 0 & \cdots & 0 & \\ 0 & 1 & \cdots & 0 & \mathbf{A}_1^{-1}\mathbf{A}_2 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & \end{array}\right]. \tag{3.9}$$

In general,

Lemma 3
Every full row rank matrix $\mathbf{A}$ can be transformed into the form $[I_{r_\mathbf{A}}|\cdots]$ while preserving uniqueness (and rank). The transformation may require a permutation of columns (if the first $r_\mathbf{A} \times r_\mathbf{A}$ submatrix is not invertible).

The same applies to $\mathbf{B}$ and $\mathbf{C}$.

3.3.3 Sufficient conditions for uniqueness

The question of whether a CP solution is unique or not was raised alongside with CP itself (Harshman [24]). Since then, several efforts have been made to develop criteria to assess this property. In this Section we will make a summary of the main results available in the literature.

Harshman [24] presents a result attributed to R. I. Jennrich as being the pioneer in this topic.

Theorem 1 (Jennrich 1970)
If matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ have full column rank, then the CP solution ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$) is essentially unique.

Note: we will refer to the condition that "$\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ have full column rank" as condition (J).

What this theorem states is that we may look for uniqueness in a solution with R factors if each mode of the array has at least order R (due to the full column-rank conditions of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$). Harshman ([24], p. 61) argued that this condition appeared to be a stronger requirement for uniqueness than was found necessary in the empirical experiments.

Harshman [25] succeeded in relaxing Jennrich's assumption on matrix $\mathbf{C}$. We now introduce this result, as well as a proof due to Ten Berge and Tendeiro [112], which is more compact than the original proof by Harshman.

Theorem 2 (Harshman 1972)
Suppose $\mathbf{A}$ and $\mathbf{B}$ have full column rank, and suppose that there exist $\mathbf{C}_1, \mathbf{C}_2$ (invertible diagonal matrices with rows of $\mathbf{C}$ in the diagonal) such that $\mathbf{C}_1\mathbf{C}_2^{-1}$ has distinct diagonal elements. Then the CP solution ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$) is essentially unique.

Proof Suppose we have two possible solutions

$$\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}' = \mathbf{G}\mathbf{D}_k\mathbf{H}', \quad k = 1, \ldots, K. \tag{3.10}$$

Notice that $\mathbf{A}$ and $\mathbf{G}$ span the same column space, as well as $\mathbf{B}$ and $\mathbf{H}$. Since both have full column rank, there exists a nonsingular $R \times R$ matrix $\mathbf{S}$ such that $\mathbf{G} = \mathbf{AS}$. Similarly, there exists a nonsingular matrix $\mathbf{T}$ such that $\mathbf{H} = \mathbf{BT}$. Then from (3.10) we can write $\mathbf{A}\mathbf{C}_k\mathbf{B}' = \mathbf{A}\mathbf{S}\mathbf{D}_k\mathbf{T}'\mathbf{B}'$. We can eliminate $\mathbf{A}$ and $\mathbf{B}'$ in this equation (they admit left inverses since they have full column rank), so we get $\mathbf{C}_k = \mathbf{S}\mathbf{D}_k\mathbf{T}'$, $k = 1, \ldots, K$.

This last equation implies that $\mathbf{C}_1\mathbf{C}_2^{-1} = \mathbf{S}\mathbf{D}_1\mathbf{D}_2^{-1}\mathbf{S}^{-1}$, so $(\mathbf{C}_1\mathbf{C}_2^{-1})\mathbf{S} = \mathbf{S}(\mathbf{D}_1\mathbf{D}_2^{-1})$, where both $\mathbf{C}_1\mathbf{C}_2^{-1}$ and $\mathbf{D}_1\mathbf{D}_2^{-1}$ are diagonal. This shows that $\mathbf{S}$ holds the eigenvectors of $\mathbf{C}_1\mathbf{C}_2^{-1}$, which holds distinct diagonal entries by hypothesis. This implies that we can write $\mathbf{S} = \mathbf{\Pi}_A\mathbf{\Lambda}_A$, where $\mathbf{\Pi}_A$ is a permutation matrix and $\mathbf{\Lambda}_A$ is a diagonal matrix ($\mathbf{S}$ has only one nonzero element per column and per row). A similar conclusion applies to $\mathbf{T}$ (since $\mathbf{C}_i$ is symmetric we have $\mathbf{C}_k = \mathbf{T}\mathbf{D}_k\mathbf{S}'$, so $\mathbf{C}_1\mathbf{C}_2^{-1}\mathbf{T} = \mathbf{T}\mathbf{D}_1\mathbf{D}_2^{-1}$), so $\mathbf{T} = \mathbf{\Pi}_B\mathbf{\Lambda}_B$ ($\mathbf{\Pi}_B$ permutation matrix, $\mathbf{\Lambda}_B$ diagonal matrix).

Replacing $\mathbf{S}$ and $\mathbf{T}$ in $\mathbf{C}_k = \mathbf{S}\mathbf{D}_k\mathbf{T}'$ gives $\mathbf{C}_k = \mathbf{\Pi}_A(\mathbf{\Lambda}_A\mathbf{D}_k\mathbf{\Lambda}_B)\mathbf{\Pi}_B'$. The symmetry of $\mathbf{C}_k$ implies that $\mathbf{\Pi}_A = \mathbf{\Pi}_B$, so $\mathbf{C}_k = \mathbf{\Pi}(\mathbf{\Lambda}_A\mathbf{D}_k\mathbf{\Lambda}_B)\mathbf{\Pi}'$. This equation leads to $\mathbf{\Pi}'\mathbf{C}_k\mathbf{\Pi} = \mathbf{D}_k\mathbf{\Lambda}_A\mathbf{\Lambda}_B$, which is equivalent to $\mathbf{C\Pi} = \mathbf{D}\mathbf{\Lambda}_A\mathbf{\Lambda}_B$, so $\mathbf{D} = \mathbf{C\Pi}\mathbf{\Lambda}_A^{-1}\mathbf{\Lambda}_B^{-1}$. Writing $\mathbf{\Lambda}_C = \mathbf{\Lambda}_A^{-1}\mathbf{\Lambda}_B^{-1}$ completes the proof. ■

Matrix $\mathbf{C}$ can be premultiplied by any nonsingular matrix without affecting essential uniqueness (see Lemma 1, p. 43). So, instead of supposing that $\mathbf{C}$ has two rows such that $\mathbf{C}_1, \mathbf{C}_2$ are invertible and $\mathbf{C}_1\mathbf{C}_2^{-1}$ has distinct diagonal elements, we can impose the condition "the row space of $\mathbf{C}$ contains a pair of vectors such that $\mathbf{C}_i\mathbf{C}_j^{-1}$ has distinct diagonal elements, for some $1 \leqslant i < j \leqslant K$". This condition is, in fact, equivalent to supposing that $k_{\mathbf{C}} \geqslant 2$, where $k_{\mathbf{C}}$ denotes the k-rank of $\mathbf{C}$, (Section 1.2):

Lemma 4
The row space of $\mathbf{C}$ holds two vectors such that $\mathbf{C}_i\mathbf{C}_j^{-1}$ has distinct diagonal elements, for some $1 \leqslant i < j \leqslant K$, if and only if $k_{\mathbf{C}} \geqslant 2$.

Therefore, Harshman's result can be enhanced by relaxing the condition on **C**, as follows:

Theorem 3
Suppose **A** and **B** have full column rank and $k_{\mathbf{C}} \geqslant 2$. Then the CP solution (**A**, **B**, **C**) is essentially unique.

Proof (Lemma 4 and Theorem 3) See Leurgans, Ross and Abel [56] for a proof of Theorem 3. Ten Berge and Tendeiro [112] present an alternative proof for the same result. ■
Note: we will refer to this relaxed sufficient condition as condition (Hr).

It is easy to see that condition (J) implies condition (Hr), but we may have situations where Harshman's condition is met but Jennrich's is not (example: $R = 3$, **A** and **B** full column rank, $\mathbf{C} = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 2 & 4 \\ 1 & 2 & 3 \end{bmatrix}$). So condition (Hr) can be considered more general than (J).

In short, if **A** and **B** have full column rank, then we should look at **C**; if $k_{\mathbf{C}} \geqslant 2$, then essential uniqueness is fulfilled. If any of the previous conditions fail then (Hr) does not apply.

The next major development for assessing uniqueness was given by Kruskal [49], who proposed a sufficient condition for uniqueness which relies on the concept of k-rank. Kruskal's (sufficient) condition is the following:

Theorem 4 (Kruskal 1977)
A CP solution (**A**, **B**, **C**) is essentially unique if

$$k_{\mathbf{A}} + k_{\mathbf{B}} + k_{\mathbf{C}} \geqslant 2R + 2. \tag{3.11}$$

Note: we will refer to Kruskal's sufficient condition as condition (K).

The proof given by Kruskal is quite inaccessible. Stegeman and Sidiropoulos [89] and Rhodes [75] devised simpler and more intuitive proofs. Here we decide to omit any proof; the reader may refer to any of these references.

We can relate Harshman's and Kruskal's conditions. When $R \geqslant 2$ it is straightforward that (Hr) implies (K). The reverse implication is true for $R = 2, 3$ (trivial) but false otherwise (for example, when $R = 4$ we may have $k_{\mathbf{A}} = 4$, $k_{\mathbf{B}} = 3$, $k_{\mathbf{C}} = 3$, thus (K) is met but (Hr) is not). This implies that (3.11) is a stronger sufficient condition for $R > 3$. When $R = 1$, (K) does not apply but (Hr) does.

The overall conclusion is that Kruskal's sufficient condition is the preferred when $R \geqslant 2$.

There is a result due to Sidiropoulos and Bro [79] that generalizes Kruskal's condition to N-way arrays, $N \geqslant 3$. It states that there is essential uniqueness if the sum of the k-ranks of all the component matrices is equal to or exceeds $2R + (N-1)$.

More recently, Jiang and Sidiropoulos [33] proved two equivalent sufficient (and necessary) conditions for what they called "restricted CP models", that is, CP solutions with a full column rank component matrix, say $\mathbf{C}$. Notice that the assumption that $\mathbf{C}$ is full column rank is typically not restrictive in applications, so both conditions can be useful in practical analysis.

De Lathauwer [16, p. 652] proved a sufficient uniqueness condition which is similar to one of the conditions presented by Jiang and Sidiropoulos [33]. De Lathauwer [16] also proposed a sufficient condition for uniqueness of restricted CP models. Stegeman *et al.* [91] made a link between Jiang and Sidiropoulos [33] and De Lathauwer [16].

3.3.4 Necessary conditions for uniqueness

So far we have only dealt with *sufficient* conditions. Now our concern will turn to properties that essentially unique solutions must have.

Leurgans *et al.* [56] discussed the next result.

Result 1
It is necessary for uniqueness that neither $\mathbf{A}$, nor $\mathbf{B}$, nor $\mathbf{C}$ has a pair of proportional columns. In other words, it is necessary to have $k_{\mathbf{A}} \geqslant 2$, $k_{\mathbf{B}} \geqslant 2$, $k_{\mathbf{C}} \geqslant 2$.

Proof Assume, without loss of generality, that the first two columns in $\mathbf{C}$ are proportional. We can write $\mathbf{A} = [\mathbf{A}^{(1)}|\mathbf{A}^{(2)}]$, $\mathbf{B} = [\mathbf{B}^{(1)}|\mathbf{B}^{(2)}]$, $\mathbf{C} = [\mathbf{C}^{(1)}|\mathbf{C}^{(2)}]$, where in each case the first block comprises the first two columns and the second block the remaining $R-2$ columns. So CP formula may be put as

$$\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}' = \mathbf{A}^{(1)}\mathbf{C}_k^{(1)}(\mathbf{B}^{(1)})' + \mathbf{A}^{(2)}\mathbf{C}_k^{(2)}(\mathbf{B}^{(2)})'. \tag{3.12}$$

Since the first two columns of $\mathbf{C}$ are proportional, there is a K-vector $\mathbf{c}$ and a 2-vector

d such that $\mathbf{C}^{(1)} = \mathbf{c}\mathbf{d}'$. Then we derive that $\mathbf{C}_k^{(1)} = c_k \mathrm{Diag}(\mathbf{1}\mathbf{d}')$. Now,

$$\begin{aligned} \mathbf{A}^{(1)}\mathbf{C}_k^{(1)}(\mathbf{B}^{(1)})' &= \mathbf{A}^{(1)}\left(c_k\mathrm{Diag}(\mathbf{1}\mathbf{d}')\right)(\mathbf{B}^{(1)})' \\ &= \underbrace{\mathbf{A}^{(1)}\mathbf{T}}\underbrace{c_k\mathbf{I}_2}\underbrace{\mathbf{T}^{-1}\mathrm{Diag}(\mathbf{1}\mathbf{d}')(\mathbf{B}^{(1)})'} \\ &= \mathbf{G}^{(1)}\mathbf{D}_k^{(1)}(\mathbf{H}^{(1)})', \end{aligned} \tag{3.13}$$

for any nonsingular 2×2 matrix **T**. This shows that the first two columns in **A** and **B** are determined up to a nonsingular transformation, hence the first two components are not unique.

The reasoning used is symmetric in **A**, **B** and **C**. ■

About the previous proof, Krijnen [44, p. 29] observes that the non-uniqueness only occurs in the first two columns of **A** and **B**. This implies that, when $R > 2$, proportional columns in **C** lead to rotational freedom only for the corresponding columns in **A** and **B**. Therefore, we can find CP solutions that are not essentially unique, but some of its components are. This is a situation of *partial uniqueness.*

Another necessary condition is due to Liu and Sidiropoulos [58]:

Result 2 (Liu and Sidiropoulos)
It is necessary for uniqueness that $\mathbf{B} \odot \mathbf{A}$ (and $\mathbf{C} \odot \mathbf{A}$ and $\mathbf{C} \odot \mathbf{B}$) has full column rank.

Proof (as given by Ten Berge and Sidiropoulos [107, p. 400])
Suppose $\mathbf{B} \odot \mathbf{A}$ does not have full column rank. Then there exists a nonzero vector **n** orthogonal to the rows of $\mathbf{B} \odot \mathbf{A}$. Adding **n** to any column of $\mathbf{C}'$ preserves the product $(\mathbf{B} \odot \mathbf{A})\mathbf{C}'$, but changes **C** by more than column permutation and/or rescaling. So there is no uniqueness. ■

Another candidate for a necessary condition for uniqueness was, for many years, Kruskal's condition (3.11) when $R > 1$. For $R = 2$ it is, in fact, necessary (it follows readily from Result 1). Kruskal had the belief that (3.11) was necessary for $R > 2$, but there was neither a proof nor a counterexample until Ten Berge and Sidiropoulos [107] settled the issue. They proved that the condition is, indeed, necessary for $R = 2, 3$ but not necessary for $R > 3$. It is interesting to note that the proof given by Ten Berge and Sidiropoulos starts by using operations that transform an array while preserving the uniqueness property, which we previously discussed in Section 3.3.2.

Ten Berge and Sidiropoulos [107] proved that Kruskal's condition is necessary when the ranks of the component matrices equal their k-ranks when $R = 4$. They conjectured that Kruskal's condition would be necessary for all cases of rank 4 and higher where ranks and k-ranks coincide. Stegeman and Ten Berge [90] refuted this conjecture.

3.4 Discussion

Uniqueness is a property that limits the possibilities of transforming the component matrices of the decompositon of an array (and the core array in the case of 3PCA), while the reconstruction of the array remains unaffected.

We have seen that 3PCA is characterized by freedom to rotate columnwise any of the component matrices. The compensation is given by transforming the core array in accordance. The CP model, on the other hand, essentially has much less freedom of rotation. Under mild conditions, only joint column permutation and rescaling are allowed.

It was also seen how some algebraic manipulations to arrays and CP decompositions do not affect their uniqueness feature. Specifically, slice mixing and reduction of the component matrices to full row rank preserve uniqueness. This way, dealing with uniqueness issues may be greatly simplified. For example, Ten Berge and Sidiropoulos [107] used these transformations to prove that Kruskal's condition is necessary for $R = 2, 3$ but not necessary for $R > 3$.

The freedom available in 3PCA can be used to our advantage. Since we are given the chance to transform the component matrices and/or the core, one might think of special useful ways to do so. In particular, transformations that "simplify" the components or the core array are relevant. Chapters 5 and 6 will deal, precisely, with this topic.

Chapter 4

Degeneracy

4.1 Introduction

In some situations the execution of a CP algorithm shows some abnormal behaviours. Harshman and Lundy [28] referred to such solutions as being "degenerate" ([28, p. 271]).

In this Chapter we intend to focus our attention on this feature of CP which is commonly known as "degeneracy". The main motivation for treating degeneracy in this thesis is that degeneracy plays an important role in Chapter 7. Therefore, it is relevant at this point to understand the main ideas concerning degeneracy. Also, one should remember that degenerate solutions are not desirable. The reason is that it is hard to defend the contribution of degenerate components to the fit in a CP decomposition (we will explain why in this Chapter). Thus, degenerate solutions should be discarded.

A characterization of degeneracy will be given. Some hypotheses concerning the reasons for degeneracy to occur will be addressed. The possibility to avoid degenerate situations will also be discussed.

4.2 Characterization of degeneracy

Sometimes running a CP program leads to solutions which are not well defined. In these situations the convergence of the program is extremely slow, and some components become more and more correlated as the program progresses.

In order to better present the degeneracy pattern we shall present the CP model as follows (relate to (2.8), p. 32): given an $I \times J \times K$ array $\underline{\mathbf{X}}$, find component matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and weights ω_r such that

$$\underline{\mathbf{X}} = \sum_{r=1}^{R} \omega_r (\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r) + \underline{\mathbf{E}}, \tag{4.1}$$

where the vectors $\mathbf{a}_r$, $\mathbf{b}_r$, $\mathbf{c}_r$ have unit length, ω_r is the weight of the r-th outer product array, and $\underline{\mathbf{E}}$ is the residual array minimized in the least squares sense.

Kruskal, Harshman and Lundy [54] describe the so-called *two-factor degeneracy* where exactly two components, say components s and t, display the following pattern:

- the columns s and t in $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ become nearly equal up to sign, the product of these signs equal to -1;
- the weights ω_s and ω_t become very large.

Such behaviour cannot be overcome by letting the CP algorithm run further: the collinearity of the columns just becomes stronger and stronger, while the weights become larger and larger. Furthermore, the degenerate rank-1 component arrays give almost opposite contributions to the fit: they nearly cancel themselves out, but together they do help to improve the fit of the model to the data. Components with this behaviour are also referred to as being *divergent*, Stegeman [84].

Three-or-more factor degeneracies can be described in a similar fashion, Harshman and Lundy [28], Stegeman [82, 84], Paatero [69], De Silva and Lim [19]. The contributions of some of the degenerate factors nearly cancels the contribution of the remaining factors, while all the factors together contribute to improve the fit of the CP model.

Degeneracy of this type– such that the improvements in loss become smaller, the collinearity of some of the components increases, and some of the weights become arbitrarily large when the algorithm runs longer– is known as strong degeneracy, unbounded degeneracy or diverging CP components, Harshman and Lundy [28], Stegeman [86]. Another type of degeneracy is when an optimal CP solution does exist, but the algorithm gets caught in *swamps*, Mitchell and Burdick [61], Paatero [69]. A swamp is typically a part of the CP sequence where the decrease in the loss fit is very small, and collinear components can occur. The algorithm may spend a lot of iterations in a swamp, thus the computation time can increase significantly. However, the weights do not get arbitrarily large, and eventually the algorithm might be able to get out of the swamp and finally converge to the desired solution. This kind of degeneracy is known as weak or bounded, Stegeman [86]. However, it is possible that swamps last for a long time. Eventually the algorithm might never be able to recover from a swamp, which can force the CP algorithm to halt due to a false convergence alarm.

Paatero [69] showed that, for rank-2 approximations to $2 \times 2 \times 2$ arrays of rank 3, temporary swamps may occur when $\underline{\mathbf{X}}$ is close to the boundary of $\mathcal{D}_2$ with $\mathcal{D}_3$ (see (2.19)). In some situations the CP sequence never recovers from its degeneracy problem.

4.3 What causes degeneracy?

Harshman and Lundy [28] discussed possible causes for the occurence of degeneracy. For example, retaining more components than the ones actually present in the data may lead to collinear components. Also, not preprocessing the data appropriately may lead to degeneracy problems. These types of degeneracies may be overcome by proper dimensioning of the model and/or processing the data. Harshman and Lundy [28] referred to these types of degeneracies as being "soft". Still, Harshman and Lundy [28] state that degeneracy can occur even though the previous scenarios do not apply. They refer to this as "hard-core" degeneracy, as there seems to exist deeper reasons for such ill situations to occur. This "hard-core" degeneracy is the hardest to understand and explain.

Kruskal *et al.* [54] present a claim (with no proof) that sheds some light on what can cause two-factor degeneracy in general: degeneracy occurs when the loss function does not admit a minimum (only an infimum), i.e., when the array has no best rank-R approximation. In particular, it is always possible to improve the fit of any solution, although these improvements become smaller and smaller. Kruskal *et al.* [54] argue that, in this case, collinear components with very large weights are due to occur. There are two problems concerning this claim: under which conditions do best rank-R decompositions fail to exist, and when do such cases lead to degeneracy situations. Krijnen, Dijkstra and Stegeman [45] answered the second part of the problem: all CP sequences that lack an optimal limiting solution will exhibit degeneracy features. However, it is still not clear *when* does an optimal CP solution exist or not.

Ten Berge *et al.* [104] present a specific $2 \times 2 \times 2$ array of rank 3 such that CP sequences of rank-2 approach an infimum without ever reaching it. Also, De Silva and Lim [19] proved that there are sequences of rank-2 arrays that do converge to a rank-3 array. Specifically, it was shown that

$$A_n = n\left(\mathbf{x}_1 + \frac{1}{n}\mathbf{y}_1\right) \circ \left(\mathbf{x}_2 + \frac{1}{n}\mathbf{y}_2\right) \circ \left(\mathbf{x}_3 + \frac{1}{n}\mathbf{y}_3\right) - n\mathbf{x}_1 \circ \mathbf{x}_2 \circ \mathbf{x}_3 \qquad (4.2)$$

is a sequence of rank-2 arrays that converges to

$$A = \mathbf{x}_1 \circ \mathbf{x}_2 \circ \mathbf{y}_3 + \mathbf{x}_1 \circ \mathbf{y}_2 \circ \mathbf{x}_3 + \mathbf{y}_1 \circ \mathbf{x}_2 \circ \mathbf{x}_3. \tag{4.3}$$

De Silva and Lim [19] also showed that CP sequences of lower rank that approach an array of higher rank always have the property that some of the weights ω_i must become arbitrarily large, which is a usual feature in degenerate CP sequences.

Stegeman [82, 83] confirmed the claim of Kruskal *et al.* [54] for generic $p \times p \times 2$ arrays of rank $p+1$ with $R = p$, generic $3 \times 3 \times p$ arrays with symmetric slices of rank $p+1$ with $R = p$, $p = 4, 5$, generic $3 \times 3 \times 5$ arrays of rank 6 with $R = 5$, and for generic $8 \times 4 \times 3$ arrays of rank 9 with $R = 8$.

De Silva and Lim [19] further clarified the issue by proving the following facts:

- The problem of determining a best rank-R approximation for a tensor in $\mathbb{R}^{d_1 \times \cdots \times d_k}$ has no solution in general for $R = 2, \ldots, \min(d_1, \ldots, d_k)$ and $k \geqslant 3$.
- The set of tensors that fail to have a best low-rank approximation has positive volume.

The first fact can be put in topological terms as follows (applying this result to three-way arrays): the set

$$\mathcal{D}_R = \{\underline{\mathbf{Y}} : I \times J \times K \text{ with rank } \leqslant R\} \tag{4.4}$$

is not closed for any $R \in \{2, \ldots, \min(I, J, K)\}$. This is equivalent to say that there exist sequences in $\mathcal{D}_R$ that converge to a limit that does not belong to $\mathcal{D}_R$. Therefore, it is *possible* that CP sequences end up converging to points that do not belong to $\mathcal{D}_R$, which by Krijnen *et al.* [45] leads to degeneracy.

Stegeman [82, 83] proved that, indeed, all sequences that converge to points outside $\mathcal{D}_R$ are degenerate, for arrays with two slices. This was done by fully caracterizing the sets $\mathcal{D}_R$, in order to better understand under which conditions it is possible to have diverging CP sequences.

4.4 How to avoid degeneracy

In some situations it is possible to avoid degeneracy. For instance, following the suggestions given by Harshman and Lundy [28], one can try to adjust the number of components of the CP decomposition to be computed. Also, preprocessing the data might help eliminating the problem. Giving different random starts to CP is also worth a try, in case of bounded degeneracies. One can also consider refining the stopping criterion in order to prevent false convergence halts.

Another possible approach to avoid degeneracy is to impose constraints on the CP decomposition such that degeneracy is prevented from occuring. One possibility is to constrain the component matrices to orthogonality, Harshman and Lundy [28]. Lim [57] shows that if $\underline{\mathbf{X}}$ is nonnegative and the component matrices are constrained to be nonnegative, then degeneracy will also not occur. In both cases it is guaranteed that degeneracy will not occur. The price to pay is some loss in the fit of the CP decomposition.

There are some improved algorithms that try to compute optimal CP solutions while avoiding degeneracy, e.g. Rayens and Mitchell [74], Kiers [39], Cao, Chen, Mo, Wu and Yu [10], Rajih, Comon and Harshman [73], Zhao [117], Navasca, De Lathauwer and Kindermann [66], Stegeman [86]. These methods try to speed up the CP algorithm once it enters a swamp. Such methods are useful in weak or bounded degeneracies situations. However, they do not solve cases like the ones studied by Stegeman [82, 83], since in these cases no optimal CP solution exists and all sequences converging to boundary points of $\mathcal{D}_R$ that do not belong to $\mathcal{D}_R$ are necessarily degenerate.

A possible workaround for this problem is given in Stegeman and De Lathauwer [88]. It is based on the generalized Schur decomposition (GSD), see De Lathauwer *et al.* [18]. A method to compute the GSD of $I \times J \times 2$ arrays is proposed. One good feature of this method is that an optimal GSD always exists. Moreover, in [88] it is shown that, for $I \times J \times 2$ arrays, the set of feasible GSD solutions equals the closure of the set of feasible CP solutions. This allows to conclude that optimal GSD solutions are the limit points for CP sequences, either convergent or divergent. In divergent cases, the limit array can either be decomposed into a nondiverging CP part and a sparse 3PCA part, or a nondiverging CP part and a smaller GSD part. If a nondiverging CP solution exists, then only the nondiverging CP part is present in these decompositions. Therefore, even in degeneracy cases it is possible to find the limiting point of CP sequences. Although mathematically appealing, the interpretation of a GSD solution differs from interpretation of a CP solution in practical terms. This makes the application of the GSD method harder in practical terms.

Chapter 5

Simplicity

5.1 Introduction

One of the practical concerns in PCA is the simplicity of the decomposition. After computing the component scores and loading matrices, it is good practice to rotate the loading matrix to a simple form. This usually means to have either high or low loadings (in absolute value), preferably with only one high loading per row of the loading matrix. This helps to understand what kind of information is stored in each component. Even so, interpretation of the (rotated) principal components may be difficult.

In 3PCA we deal with a similar problem. As was seen in Chapter 3, there is ample freedom of rotation in 3PCA. We might want to make use of this possibility to find a solution with equal fit but that is easier to interpret. A relevant question to answer is: in what sense can a solution be "simpler" than other? One can observe that the interpretation of a 3PCA solution can be rather difficult because we have to evaluate the impact of each triple of components to explain the data. In fact, (2.1) shows that a 3PCA decomposition has as many rank-one terms as there are nonzero entries in $\underline{\mathbf{G}}$. Transforming the core array into an array with a large majority of zero elements may simplify interpretation, because there will be less interactions of triples of components to explain (see, for example, Rocci and Ten Berge [78]). This may indeed ease the statistical interpretation of the model, provided that the rotated components can easily be interpreted.

Whenever we mention that our goal is to "simplify a 3PCA decomposition", we mean that we like to rewrite it with as many zeros possible in the core array. Aiming at simple forms for the component matrices will not be our goal in this research.

A word of caution is in order here. Murakami [63] gave an example where the components tend to collinearity when the core array is transformed to extreme simplicity (Rocci and Ten Berge [78]). These degenerate solutions are not desirable and should be avoided whenever possible.

More often than not, we will consider the simplicity issue from a more general point of view. Given an array $\mathbf{X}_a = [\mathbf{X}_1 | \cdots | \mathbf{X}_K]$ of order $I \times J \times K$, we wish to find nonsingular matrices $\mathbf{S}$, $\mathbf{T}$ and $\mathbf{U}$ such that the transformed array,

$$\mathbf{S}'\mathbf{X}_a(\mathbf{U} \otimes \mathbf{T}) = \left[\mathbf{S}' \left(\sum_{r=1}^{R} u_{r1}\mathbf{X}_r \right) \mathbf{T} \middle| \cdots \middle| \mathbf{S}' \left(\sum_{r=1}^{R} u_{rR}\mathbf{X}_r \right) \mathbf{T} \right], \tag{5.1}$$

has as many zeros as possible. Matrices $\mathbf{S}$, $\mathbf{T}$, $\mathbf{U}$ can be seen as slice mixers from the three possible directions (horizontal, lateral and frontal direction, respectively). Therefore, our search for simplicity can be posed in terms of simplifying arrays in general, instead of only simplifying 3PCA decompositions. The application of simplicity results for arrays in general to 3PCA decompositions becomes direct once we apply such results on the core arrays of the decompositions.

Rocci and Ten Berge [78] (pp. 352-353) mention two more important implications that simplicity can lead to. One is the fact that maximal simplicity can help the study of the core's rank (and typical rank in general). This is because the number of nonzero elements in a three-way array is a universal upper bound to the rank, and performing Tucker transformations is rank-preserving. The other implication is that simplicity can be a good tool to assess the (non-)triviality of a model. Triviality can be avoided by imposing additional or different constraints to the core array, thus achieving a real model.

It can be concluded that simplicity has several important applications within three-way analysis in particular and tensor theory in general.

Three computational methods that bring simplicity in arrays are presented in Section 5.2: SIMPLIMAX (Kiers [37, 40]), the Orthogonal Complement Algorithm (Rocci and Ten Berge [78]), and the multiple orthonormality approach (Ten Berge *et al.* [105]). Both SIMPLIMAX and the multiple orthonormality approach are iterative methods, whereas the Orthogonal Complement Algorithm provides a closed-form solution. Other methods are available but were not covered in this thesis, like rotating cubic core arrays such that the frontal slices become diagonal (for example MacCallum [59], Kroonenberg [47], Brouwer and Kroonenberg [7]), or such that the core array becomes superdiagonal (Kiers [36]).

Section 5.3 presents closed-form simplicity results for two families of arrays. Finally, in Section 5.4 the issue of maximal simplicity will be discussed.

5.2 Computational approaches

5.2.1 SIMPLIMAX

SIMPLIMAX was originally devised for two-way PCA (Kiers [37]). The goal is to look for an oblique rotation of the loading matrix so that, after rotation, the m smallest elements have a minimal sum of squares (m specified in advance).

Kiers [40] extended this procedure to three-way arrays. Three-way SIMPLIMAX finds oblique rotations (in the three directions) of the core array that minimize the sum of squares of the m smallest elements of the rotated core array, with m specified in advance. It is not known *a priori* which entries will be the smallest ones, so the algorithm will internally solve this issue. Rotating the core array to a non-predefined target was not a new idea, but also not very common. Prior to SIMPLIMAX we can mention Murakami [64] who attempted to rotate the transposed supermatrix of frontal core planes to simple structure by means of varimax rotation, Kruskal [50] who worked on a procedure for maximizing a combination of normalized quartimax functions, and Kiers [42] who worked on an orthomax rotation of the core to simple structure.

The fact that the rotating target is not fixed *a priori* has the side effect of SIMPLIMAX finding locally optimal solutions. This problem can be circumvented by using a large number of randomly started runs of the algorithm.

An adapted version of SIMPLIMAX allows to fix, in advance, the position of the m entries whose sum of squares we intend to minimize. This way it is possible to rotate arrays having a specific target array in mind. The quality of the fit of the rotated array to the specified target can be controled by analysing the sum of squares of the m elements in the fixed positions.

Kiers implemented SIMPLIMAX in *Matlab*. His program was extensively used to find hypothesis about which simplicity transformations might be possible in specific situations, see Chapter 6.

5.2.2 The Orthogonal Complement Method (OCM)

The Orthogonal Complement Method (OCM) is a method that permits transforming an array to simple form by using the previously known simplicity of a "complementary" array. This will allow to get simplicity for some arrays of orders not contemplated so far. A treatment of this topic can be found with detail in Rocci and Ten Berge [78].

Consider an $I \times J \times K$ array $\underline{\mathbf{X}}$, and its matricized vec*-form: $\mathbf{X}_{\text{vec}^*} = [\text{vec}^*(\mathbf{X}_1)| \cdots |\text{vec}^*(\mathbf{X}_K)]$, see p. 13 for notation. We will assume that $\underline{\mathbf{X}}$ has linearly independent slices in all directions, and that $IJ > K$ (avoiding overfactoring, see p. 38). Under these assumptions $\mathbf{X}_{\text{vec}^*}$ is a vertical matrix with full column rank. It is always possible to append an $IJ \times (IJ - K)$ block to $\mathbf{X}_{\text{vec}^*}$, say $\mathbf{X}^{(\text{c})}_{\text{vec}^*}$, so that $[\mathbf{X}_{\text{vec}^*}|\mathbf{X}^{(\text{c})}_{\text{vec}^*}]$ is an invertible matrix of order IJ. Notice that $\mathbf{X}^{(\text{c})}_{\text{vec}^*}$ can be seen as the vec*-version of an array of order $I \times J \times (IJ - K)$, called the *complementary array* of $\underline{\mathbf{X}}$ (and denoted by $\underline{\mathbf{X}}^{(\text{c})}$). As an example, consider $\mathbf{X}_a = \left[\begin{smallmatrix} 1 & -1 \\ 0 & 2 \end{smallmatrix}\middle|\begin{smallmatrix} 0 & 1 \\ 2 & 1 \end{smallmatrix}\right]$. Then $\mathbf{X}_{\text{vec}^*} = \left[\begin{smallmatrix} 1 & -1 & 0 & 2 \\ 0 & 1 & 2 & 1 \end{smallmatrix}\right]'$. A possible complementary block is $\mathbf{X}^{(\text{c})}_{\text{vec}^*} = \left[\begin{smallmatrix} 1 & 2 & 1 & 2 \\ -1 & 2 & 1 & 1 \end{smallmatrix}\right]'$ ($[\mathbf{X}_{\text{vec}^*}|\mathbf{X}^{(\text{c})}_{\text{vec}^*}]$ is invertible), which is the vec*-version of $\mathbf{X}^{(\text{c})}_a = \left[\begin{smallmatrix} 1 & 2 \\ 1 & 2 \end{smallmatrix}\middle|\begin{smallmatrix} -1 & 2 \\ 1 & 1 \end{smallmatrix}\right]$.

So, arrays of size $I \times J \times K$ are complementary to arrays of size $I \times J \times (IJ - K)$. However, we are interested in finding specific complementary matrices for $\mathbf{X}_{\text{vec}^*}$. Specifically, we are interested in choosing $\mathbf{X}^{(\text{c})}_{\text{vec}^*}$ so that its columns span the space orthogonal to the column space of $\mathbf{X}_{\text{vec}^*}$, that is, $\mathbf{X}'_{\text{vec}^*}\mathbf{X}^{(\text{c})}_{\text{vec}^*} = \mathbf{0}$ (the reason for this will be clear in the next paragraph). Array $\underline{\mathbf{X}}^{(\text{c})}$ is said to be an *orthogonal complement* to $\underline{\mathbf{X}}$. In the example of the previous paragraph consider $\mathbf{X}^{(\text{c})}_{\text{vec}^*} = \left[\begin{smallmatrix} -2 & -2 & 1 & 0 \\ -3 & -1 & 0 & 1 \end{smallmatrix}\right]'$ (note that $\mathbf{X}'_{\text{vec}^*}\mathbf{X}^{(\text{c})}_{\text{vec}^*} = \mathbf{0}_2$), whose associated complementary array is $\mathbf{X}^{(\text{c})}_a = \left[\begin{smallmatrix} -2 & -2 \\ 1 & 0 \end{smallmatrix}\middle|\begin{smallmatrix} -3 & -1 \\ 0 & 1 \end{smallmatrix}\right]$.

The main advantage associated to this procedure is that arrays in simple form have orthogonal complementary arrays that are also in simple form (Rocci and Ten Berge [78, p. 357]). So, as a general rule, simple arrays induce simple complementary arrays, which broadens the types of arrays we are able to simplify.

For the OCM to work, it is essential to know how to compute orthogonal complements and how to simplify the complementary type of array. Rocci and Ten Berge [78] present the algorithm as follows:

1. given the array $\mathbf{X}_{\text{vec}^*}$, compute an orthogonal complement $\mathbf{X}^{(\text{c})}_{\text{vec}^*}$;

2. compute $\mathbf{H}^{(\text{c})}_{\text{vec}^*} = (\mathbf{S}^{-1} \otimes \mathbf{T}^{-1})\mathbf{X}^{(\text{c})}_{\text{vec}^*}\mathbf{V}$ in such a way that $\mathbf{H}^{(\text{c})}_{\text{vec}^*}$ is in simple form;

3. find the orthogonal complement of $\mathbf{H}^{(\text{c})}_{\text{vec}^*}$ in simple form, say $\mathbf{H}_{\text{vec}^*}$;

4. find matrix $\mathbf{U}$ such that $\mathbf{H}_{\text{vec}^*} = (\mathbf{S}' \otimes \mathbf{T}')\mathbf{X}_{\text{vec}^*}\mathbf{U}$, or equivalently, $\mathbf{H}_a = \mathbf{S}'\mathbf{X}_a(\mathbf{U} \otimes \mathbf{T})$ (in accordance with (5.1)).

It should be noticed that the procedure above is corrected for a typing error in Rocci and Ten Berge [78] (namely in step 4).

With this method we are able to simplify infinitely many arrays, just by knowing at the start how to simplify one array. For instance, by knowing how to simplify the $3 \times 3 \times 2$ array we can simplify the array $3 \times 3 \times 7$ as a complement, which in turn has the $3 \times 18 \times 7$ as a complement, which in turn has the $3 \times 18 \times 47$ as a complement, and so on (Rocci and Ten Berge [78, p. 359]).

This algorithm was often used during our research for simplicity of symmetric slice arrays. We will defer the presentation of more practical examples until Chapter 6.

5.2.3 Multiple orthonormality

It is straightforward to transform full row (full column) rank matrices into rowwise (columnwise) orthonormal matrices. An $m \times n$ matrix $\mathbf{M}$ is said to be *columnwise orthonormal* when $\mathbf{M}'\mathbf{M} = \mathbf{I}_n$, and *rowwise orthonormal* when $\mathbf{M}\mathbf{M}' = \mathbf{I}_m$. A more relaxed concept of orthonormality defines $\mathbf{M}$ as columnwise orthonormal when $\mathbf{M}'\mathbf{M}$ is proportional to $\mathbf{I}_n$, and as rowwise orthonormal when $\mathbf{M}\mathbf{M}'$ is proportional to $\mathbf{I}_m$. This relaxed concept of orthonormality allows that the columns/rows of a matrix have a constant fixed length other than unity, but otherwise it is the same as the usual concept.

Ten Berge *et al.* [105] address the question of achieving multiple orthonormality for three-way arrays, recall Section 3.2. They focus on the three possible matricized versions of an array (unfolding the frontal, lateral or horizontal slices), and use the relaxed definition of orthonormality. Unfortunately, the question of achieving triple ortho*normality* is much harder to answer. Ten Berge *et al.* [105] present an iterative algorithm to alternately orthonormalize two modes. It is shown how this process always converges, but not necessarily to the desired double orthonormality situation. Some conditions were proven that clarify which are the cases where convergence to double orthonormality is to be expected.

As for triple orthonormality, the problem is still open. Ten Berge *et al.* [105] indicate that transformation to double orthonormality often tends to give triple orthonormality as a bonus, but it is not clear under which conditions this works.

Multiple orthonormality can be used as a tool to search for array simplicity. In fact, transforming an array to multiple orthonormality may give simplicity as an extra. For example, Ten Berge *et al.* [105] present the case of arrays of order $4 \times 3 \times 2$, that often have 18 of 24 elements zero after multiple orthonormality transformation.

5.3 Example of simplicity for two families of arrays

At this point we present two examples of families of arrays for which simple forms exist already. These examples are supposed to serve as an illustration of what is to be found in Chapter 6, where new results about simplicity will be presented.

5.3.1 $P \times Q \times R$ arrays, when $P = QR$ and $P = QR - 1$

We can always consider the dimensions of the core array $\underline{\mathbf{G}}$ to satisfy $P \leqslant QR$, in order to avoid overfactoring situations.

In the situation $P = QR$ we have $\mathbf{G}_a$ square, so taking $\mathbf{S} = \mathbf{G}_a^{-1}$, $\mathbf{T} = \mathbf{I}_Q$, $\mathbf{U} = \mathbf{I}_R$ shows that $\mathbf{S}\mathbf{G}_a(\mathbf{U} \otimes \mathbf{T}) = \mathbf{I}_P$. Therefore, when $P = QR$ the core can be simplified to have $(P^2 - P)$ zero elements (Murakami, Ten Berge and Kiers [65, p. 256]). It should be noted that, for all practical purposes, core arrays that result from the application of the 3PCA model seem to behave as if they were randomly generated from a continuous distribution function, Rocci and Ten Berge [78, p. 362]. Therefore, the existence of $\mathbf{G}_a^{-1}$ when $P = QR$ is almost surely satisfied.

Murakami *et al.* [65] provide a closed form solution for nonsingular transformations $\mathbf{S}$, $\mathbf{T}$, $\mathbf{U}$ such that only a few nonzero elements in $\mathbf{S}\mathbf{G}_a(\mathbf{U} \otimes \mathbf{T})$ remain, for any $\underline{\mathbf{G}}$ of order $P \times Q \times R$ with $P = QR - 1$ and such that $\mathbf{G}_a$ has full row rank. This case is, therefore, the closest that can be taken to the $P = QR$ situation. Their simple form is a columnwise permuted version of the matrix

$$\left[\begin{array}{c|c} \mathbf{I}_{QR-R} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{M}_{(R-1)R} \end{array}\right], \tag{5.2}$$

with $\mathbf{M}$ usually with nonzero entries. For example, the simple form for the $5 \times 3 \times 2$ array is

$$\left[\begin{array}{ccc|ccc} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \star & 0 & 0 & 0 & \star & 0 \end{array}\right], \tag{5.3}$$

where $\star$ represents a nonzero entry.

5.3.2 $P \times Q \times 2$ arrays

Ten Berge and Kiers [103] gave an explicit solution to transform arrays with dimensions $P \times Q \times 2$ when $Q < P < 2Q$. Specifically, array $\mathbf{G}_a = [\mathbf{G}_1|\mathbf{G}_2]$ is transformed into $\mathbf{Y}_a = [\mathbf{Y}_1|\mathbf{Y}_2]$ where $\mathbf{Y}_1 = \left[\begin{array}{c}\mathbf{I}_Q\\ \hline \mathbf{0}\end{array}\right]$, $\mathbf{Y}_2 = \left[\begin{array}{c}\mathbf{0}\\ \hline \mathbf{I}_Q\end{array}\right]$. This transformation is proven to work almost surely. It is curious to notice that their solution does not require slab-mixing ($\mathbf{U} = \mathbf{I}_R$).

The only case of this family not covered so far is array $\mathbf{X}_a = [\mathbf{X}_1|\mathbf{X}_2]$ with two square frontal slices of order P. The solution can be found in Rocci and Ten Berge [78]; it applies to the situations when at least one of the slices is invertible and $\mathbf{X}_1^{-1}\mathbf{X}_2$ is non-defective (recall the concept of defective matrix on p. 18). It depends on the eigenvalues of $\mathbf{X}_1^{-1}\mathbf{X}_2$, called the *generalized eigenvalues* of $\underline{\mathbf{X}}$. For example, a $3\times3\times2$ array with two complex generalized eigenvalues can be simplified to the form

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & \mu\\ 0 & 0 & 1 & 0 & -\mu & 0 \end{array}\right]. \tag{5.4}$$

5.4 Maximal simplicity

When an array is simplified into a simple form with many zeros, one can always ask if it is possible to simplify it even more. *Maximal simplicity* concerns the Tucker transformation that introduces the maximum number possible of zero entries.

Rocci and Ten Berge [78] proved that both simplicity transformations presented in Section 5.3 are optimal in this sense. They also clarified the question of maximal simplicity for arrays of order $I \times J \times K$ when $K = IJ - 2$ and $I > J$.

Maximal simplicity is an issue that we will address several times in Chapter 6. Some transformations to simple form will be presented there, and it will be proven that the newly found simple forms have maximal simplicity.

5.5 Discussion

In this Chapter we addressed the issue of simplicity in 3PCA. We argued that there are good reasons that justify transforming arrays into simple form, both from a statistical as well from a mathematical point of view. We presented three computational approaches (SIMPLIMAX, OCM, Multiple orthonormality) that allow, in some cases, to reach simple forms. Finally, we presented two practical examples of families of arrays for which simple forms have already been found in the literature.

A special situation occurs when the array to be simplified has symmetric frontal slices. The results that apply to the general situation are not suited in this framework, because usually the transformations to be made will ultimately destroy the symmetry of the frontal slices.

Some attention was given to this special situation during our research. In the next Chapter we will present and discuss the results that were found.

Chapter 6

Simplicity for symmetric-slice arrays

6.1 Introduction

Recall from last Chapter (Section 5.1) that there are several reasons that justify the importance of studying simplicity techniques. Namely, transforming arrays into simple forms can help: interpreting 3PCA decompositions, distinguishing between tautologies and non-trivial models, and studying some mathematical properties of arrays (such as typical rank, for example). This Chapter revolves around the issue of simplicity transformations applied to symmetric-slice arrays, that is, arrays with symmetric slices in one direction. The contents to be found in this part of the thesis are the result of research that we carried out during the first part of our project.

Research concerning simplicity of arrays in general is available, e.g., Kiers [36, 40], Murakami, Ten Berge and Kiers [65], Ten Berge and Kiers [103], Ten Berge *et al.* [105], and Rocci and Ten Berge [78]. However, whatever closed-form simplicity results have been obtained, they apply to arrays the elements of which are randomly sampled from a continuous distribution. In practice, data sets frequently contain symmetric slices (for example, sets of correlation matrices collected over time). For such cases the above results do not hold. Research on simplicity for arrays that have symmetric slices in one direction is still absent. In this Chapter we discuss some results for the latter type of array.

From now on we assume that $\underline{\mathbf{X}}$ is an $I \times I \times K$ array with K symmetric frontal slices $\mathbf{X}_k$ of order $I \times I$, $k = 1, \ldots, K$. We assume that these slices are linearly independent. If this is not the case, we start by transforming the superfluous slices to zero via a suitable Tucker transformation, reducing the dimensionality of the problem. This transformation is relevant from a mathematical point of view but not from a practical one, because 3PCA core arrays with $K > K_{\max}$ are usually associated with overfactoring (see p. 38). Therefore, a good choice of number of components in 3PCA should avoid this problem altogether.

The space of real symmetric matrices of order $I \times I$ has dimension $I(I+1)/2$. Therefore, the number K of symmetric slices to consider should not exceed $K_{\max} = I(I+1)/2$.

We are looking for symmetry-preserving Tucker transformations of $\underline{\mathbf{X}}$ which yield an array $\underline{\mathbf{H}}$ with a large number of zero elements. So, our goal is to determine nonsingular matrices $\mathbf{S}$, $\mathbf{U}$ such that

$$\mathbf{H}_l = \mathbf{S}' \left(\sum_{k=1}^{K} u_{kl} \mathbf{X}_k \right) \mathbf{S}, \quad l = 1, 2, \ldots, K, \tag{6.1}$$

where u_{kl} is an element of $\mathbf{U}$, has as many zero entries as possible. The number of nonzero entries in $\underline{\mathbf{H}}$ will be referred to as the *weight* of $\underline{\mathbf{H}}$.

It may be noted that we have tacitly assumed in (6.1) that $\mathbf{S}$ and $\mathbf{T}$ of (3.2) can be constrained to be equal. In fact, this is a simplification, because symmetry preserving transformations with $\mathbf{S}$ and $\mathbf{T}$ different do exist. However, this is possible only for two-slice arrays. This will be proven in Section 6.2. It should be observed that setting $\mathbf{S}$ and $\mathbf{T}$ equal for arrays with two frontal slices has not been detrimental at all in our search for transformations that minimize the weight of arrays.

In Appendix I (p. 97 ff.) we proved a number of simplicity results for arrays with symmetric slices of order 2×2, 3×3 and 4×4. At this point we will only present the main conclusions.

We observe that the main reason for treating arrays (with symmetric slices) of order 2×2, 3×3 and 4×4 is methodological. The proofs that we devised are specific to each case, because it is difficult to develop a unifying approach to larger families of arrays.

In Section 6.3 we present an adaptation of the OCM (Orthogonal Complement Method, section 5.2.2) to the special situation of symmetric-slice arrays. Section 6.4 presents the simple forms that were found for arrays with symmetric slices of order 2×2, 3×3 and 4×4. The case when the array has the maximum possible number of symmetric frontal slices is also treated. Finally, considerations about typical rank and maximal simplicity are to be found in Sections 6.5 and 6.6, respectively.

6.2 Setting S=T

Consider an $I \times I \times K$ array $\underline{\mathbf{X}}$ with symmetric frontal slices. We are interested in Tucker transformations of $\underline{\mathbf{X}}$ that preserve the symmetry of the frontal slices. This means that we are only interested in finding nonsingular matrices $\mathbf{S}$, $\mathbf{T}$ and $\mathbf{U}$ such that

$$\mathbf{H}_l = \mathbf{S}' \left(\sum_{k=1}^{K} u_{kl} \mathbf{X}_k \right) \mathbf{T} \tag{6.2}$$

is symmetric, $l = 1, 2, \ldots, K$. For $\mathbf{H}_l$ to be symmetric, it suffices to search for $\mathbf{S}$ and $\mathbf{T}$ such that $\mathbf{S}'\mathbf{X}_k\mathbf{T}$ is symmetric, $k = 1, \ldots, K$.

It is straightforward to observe that having $\mathbf{S}$ equal to $\mathbf{T}$ is a sufficient condition for $\mathbf{S}'\mathbf{X}_k\mathbf{T}$ to be symmetric, for all $k = 1, 2, \ldots, K$. We will next prove that this condition is necessary for $K \geqslant 3$ but not when $K = 2$, under the assumption that $\underline{\mathbf{X}}$ is randomly drawn from a continuous distribution. It will be shown how suitable Tucker transformations with $\mathbf{S}$ and $\mathbf{T}$ different can be computed when $K = 2$.

First consider the case $K = 2$, therefore $\underline{\mathbf{X}}$ has two symmetric frontal slices, $\mathbf{X}_1$ and $\mathbf{X}_2$. Randomly generate an $\mathbf{S}$, and define $\mathbf{T} = \mathbf{X}_1^{-1}\mathbf{X}_2\mathbf{S}$. Then $\mathbf{S}'\mathbf{X}_1\mathbf{T} = \mathbf{S}'\mathbf{X}_1\mathbf{X}_1^{-1}\mathbf{X}_2\mathbf{S} = \mathbf{S}'\mathbf{X}_2\mathbf{S}$ is symmetric, and $\mathbf{S}'\mathbf{X}_2\mathbf{T} = \mathbf{S}'\mathbf{X}_2\mathbf{X}_1^{-1}\mathbf{X}_2\mathbf{S}$ is symmetric. This shows how different $\mathbf{S}$ and $\mathbf{T}$ can be computed such that the symmetry of both frontal slices of $\underline{\mathbf{X}}$ is preserved after transformation.

Now consider $K \geqslant 3$. We will prove that it is necessary to have $\mathbf{S} = \mathbf{T}$ for $K = 3$, which immediately implies the same result for $K > 3$. Suppose that there exist matrices $\mathbf{S}$ and $\mathbf{T}$ such that ($i = 1, 2, 3$)

$$\mathbf{S}'\mathbf{X}_i\mathbf{T} = \mathbf{T}'\mathbf{X}_i\mathbf{S}. \tag{6.3}$$

Observe that, for $i \neq j$,

$$(\mathbf{S}'\mathbf{X}_i\mathbf{T})^{-1}(\mathbf{S}'\mathbf{X}_j\mathbf{T}) = (\mathbf{T}'\mathbf{X}_i\mathbf{S})^{-1}(\mathbf{T}'\mathbf{X}_j\mathbf{S}), \tag{6.4}$$

or, equivalently,

$$\mathbf{T}^{-1}\mathbf{X}_i^{-1}\mathbf{X}_j\mathbf{T} = \mathbf{S}^{-1}\mathbf{X}_i^{-1}\mathbf{X}_j\mathbf{S}. \tag{6.5}$$

Using the eigendecompositions $\mathbf{X}_1^{-1}\mathbf{X}_2 = \mathbf{K}_1\mathbf{\Lambda}_1\mathbf{K}_1^{-1}$ and $\mathbf{X}_1^{-1}\mathbf{X}_3 = \mathbf{K}_2\mathbf{\Lambda}_2\mathbf{K}_2^{-1}$, we derive that

$$(\mathbf{T}^{-1}\mathbf{K}_1)\mathbf{\Lambda}_1(\mathbf{T}^{-1}\mathbf{K}_1)^{-1} = (\mathbf{S}^{-1}\mathbf{K}_1)\mathbf{\Lambda}_1(\mathbf{S}^{-1}\mathbf{K}_1)^{-1} \tag{6.6}$$

$$(\mathbf{T}^{-1}\mathbf{K}_2)\mathbf{\Lambda}_2(\mathbf{T}^{-1}\mathbf{K}_2)^{-1} = (\mathbf{S}^{-1}\mathbf{K}_2)\mathbf{\Lambda}_2(\mathbf{S}^{-1}\mathbf{K}_2)^{-1}. \tag{6.7}$$

Each of the previous two equations are equalities between eigendecompositions. Since the diagonal entries of $\mathbf{\Lambda}_i$ $(i = 1, 2)$ are almost surely distinct, this means that $\mathbf{T}^{-1}\mathbf{K}_i$ and $\mathbf{S}^{-1}\mathbf{K}_i$ are columnwise proportional. So there exists a diagonal matrix $\mathbf{D}_i$ such that $\mathbf{S}^{-1}\mathbf{K}_i = \mathbf{T}^{-1}\mathbf{K}_i\mathbf{D}_i$, or equivalently,

$$\mathbf{T}\mathbf{S}^{-1} = \mathbf{K}_i\mathbf{D}_i\mathbf{K}_i^{-1}, \tag{6.8}$$

for $i = 1, 2$.
Suppose α is an eigenvalue of $\mathbf{T}\mathbf{S}^{-1}$ with algebraic multiplicity $m < I$ (see Section 1.4 for the definitions of algebraic and geometric multiplicities of an eigenvalue). Equations (6.8) give $2m$ eigenvectors associated to eigenvalue α. Because the $\mathbf{X}_i$'s were randomly generated, it is almost sure that the dimension of the eigenspace associated to α (i.e., the geometric multiplicity) will be at least $\min\{2m, I\}$. This would imply a geometric multiplicity greater than an algebraic multiplicity, which is impossible. Therefore, α must have algebraic multiplicity equal to I, that is, $\mathbf{T}\mathbf{S}^{-1}$ has equal eigenvalues. So $\mathbf{D}_1$ and $\mathbf{D}_2$ are equal and proportional to identity, hence $\mathbf{T}\mathbf{S}^{-1} = \alpha\mathbf{I}$ (for some real α) and $\mathbf{T} = \alpha\mathbf{S}$. This shows that $\mathbf{S}$ and $\mathbf{T}$ must be equal up to a scalar.

6.3 A symmetric version of the OCM

The Orthogonal Complement Method (OCM; see Section 5.2.2 for an overview) has played a key role in obtaining some important simplicity results. However, to be useful in the symmetric context, it needed to be adjusted in two respects. First, the dimensionality of the space of $I \times I$ matrices is I^2 in general, but it is only $I(I+1)/2$ in case of symmetry. So $\mathbf{X}_{\text{vec}^*} = [\text{vec}^*(\mathbf{X}_1)|\ldots|\text{vec}^*(\mathbf{X}_K)]$ can be regarded as an $I(I+1)/2 \times K$ matrix, after eliminating all rows that appear repeated due to symmetry. Second, we wish to constrain the transformations such that $\mathbf{S}$ and $\mathbf{T}$ are equal, as this ensures that symmetry of frontal slices is preserved after transformation. This amounts to the following general setup of OCM for symmetric-slice arrays:

1. given the array $\mathbf{X}_{\text{vec}^*}$, compute an orthogonal complement $\mathbf{X}^{(\text{c})}_{\text{vec}^*}$;
2. compute $\mathbf{H}^{(\text{c})}_{\text{vec}^*} = (\mathbf{S}^{-1} \otimes \mathbf{S}^{-1})\mathbf{X}^{(\text{c})}_{\text{vec}^*}\mathbf{V}$ in such a way that $\mathbf{H}^{(\text{c})}_{\text{vec}^*}$ is in simple form;
3. find the orthogonal complement of $\mathbf{H}^{(\text{c})}_{\text{vec}^*}$ in simple form, say $\mathbf{H}_{\text{vec}^*}$;
4. find matrix $\mathbf{U}$ such that $\mathbf{H}_{\text{vec}^*} = (\mathbf{S}' \otimes \mathbf{S}')\mathbf{X}_{\text{vec}^*}\mathbf{U}$.

6.4 Simple forms for some families of symmetric-slice arrays

In this Section we present the main simplicity results that were found during our research. For more details and proofs see Appendix I (p. 97).

6.4.1 Simplifying symmetric-slice $I \times I \times K_{\mathbf{max}}$ arrays

When the number of slices of $\underline{\mathbf{X}}$ is $K_{\max} = I(I+1)/2$ it is always possible to transform the array into a simple form with weight I^2 (Rocci and Ten Berge [77]). For example, symmetric-slice array of order $3 \times 3 \times 6$ can be simplified into

$$\left[\begin{array}{ccc|ccc|ccc|ccc|ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right]. \tag{6.9}$$

This simple form for the $I \times I \times K_{\max}$ symmetric-slice array is also useful when we are dealing with tall $I \times I \times K$ arrays, i.e., with $K > K_{\max}$. In this case the frontal slices are linearly dependent. Starting by performing a suitable slice mix in order to set the slices in excess to zero allows to reduce the number of frontal slices. When $K_{\max}$ linearly independent slices remain, the above simplification is possible.

6.4.2 Simplifying symmetric-slice $2 \times 2 \times K$ arrays

There are three cases to consider when $\underline{\mathbf{X}}$ is a $2 \times 2 \times K$ array: $K = 1$, 2, or 3.

The situation $K_{\max} = 3$ was already solved in the previous Section; the simple form associated to $\underline{\mathbf{X}}$ is $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\middle|\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\middle|\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$.

When $K = 1$ the array is merely a symmetric matrix $\mathbf{X}$. It can be diagonalized using the eigendecomposition. Therefore, a weight 2 simple form is almost surely possible.

Two possibilities arise when $K = 2$:

- if the eigenvalues of $\mathbf{X}_1^{-1}\mathbf{X}_2$ (the generalized eigenvalues of $\underline{\mathbf{X}}$) are real and $\mathbf{X}_1^{-1}\mathbf{X}_2$ is nondefective, then $\underline{\mathbf{X}}$ can be transformed into the weight 2 simple form $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\middle|\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right]$;

- otherwise, the weight 4 simple form $\left[\begin{smallmatrix} \alpha & 0 \\ 0 & -1 \end{smallmatrix}\middle|\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$, where α is a real number, is almost surely possible.

6.4.3 Simplifying symmetric-slice $3 \times 3 \times K$ arrays

For symmetric-slice $3 \times 3 \times K$ arrays we have $K_{\max} = 6$. The simple form for symmetric-slice $3 \times 3 \times 6$ arrays was already presented, see (6.9). Simple forms for K =1, 2, 4 and 5 will be discussed next. The case $K = 3$ remains an open issue.

When $K = 1$ the array is a matrix. The diagonalization via the eigendecomposition leads to a simple form with weight 3.

Simple forms for arrays $\underline{\mathbf{X}}$ when $K = 5$ arise as the orthogonal complement of $3 \times 3 \times 1$ arrays. A weight 10 simple form which is almost surely possible is

$$\left[\begin{array}{ccc|ccc|ccc|ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right], \tag{6.10}$$

where α, β are real entries. When there are eigenvalues of $\underline{\mathbf{X}}^{(c)}$ of both signs, the following weight 9 target is possible:

$$\left[\begin{array}{ccc|ccc|ccc|ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right]. \tag{6.11}$$

Symmetric-slice $3 \times 3 \times 2$ arrays can be simplified according to the type of generalized eigenvalues involved:

- if the generalized eigenvalues are real, $\underline{\mathbf{X}}$ has slices that can be diagonalized simultaneously; this leads to the weight 4 simple form

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & \beta & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right], \tag{6.12}$$

  where α, β are real entries. An alternative simple form (with higher weight), that will prove useful in later applications of the OCM, is:

$$\left[\begin{array}{ccc|ccc} -\alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right]; \tag{6.13}$$

- if the generalized eigenvalues of $\underline{\mathbf{X}}$ are complex, a weight 5 simple form is

$$\left[\begin{array}{ccc|ccc} -\alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 0 \end{array}\right], \tag{6.14}$$

  where α is real, is almost surely possible.

A simple form for $3\times3\times4$ can be found applying the OCM. Both (6.13) and (6.14) lead to the following simple form for the complement,

$$\left[\begin{array}{ccc|ccc|ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \delta\alpha & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right], \tag{6.15}$$

where α has the same meaning as in (6.13) and (6.14), and $\delta = 1$ in the first case and -1 in the second case. So, almost surely, a $3 \times 3 \times 4$ symmetric-slice array can be simplified to a weight 8 simple form.

The only situation left to analyze is the $3 \times 3 \times 3$ symmetric-slice array. In this case the OCM is not useful, because the orthogonal complement of a $3 \times 3 \times 3$ array with symmetric slices is also a $3 \times 3 \times 3$ array. So far, a closed-form transformation to simple form for this array format has not been found. However, some numerical simulations were carried out in order to inspect for possible simple forms. The most interesting simple form that was found in this way is a model studied by Kiers, Ten Berge and Rocci [42]:

$$\left[\begin{array}{ccc|ccc|ccc} * & 0 & 0 & 0 & 0 & * & 0 & * & 0 \\ 0 & 0 & * & 0 & * & 0 & * & 0 & 0 \\ 0 & * & 0 & * & 0 & 0 & 0 & 0 & * \end{array}\right]. \tag{6.16}$$

SIMPLIMAX succeeded to transform 179 out of 200 randomly generated $3 \times 3 \times 3$ symmetric-slice arrays into simple form (6.16). It is still not clear what mathematical property (if any) is shared by the 21 arrays that failed to be transformed into simple form (6.16). In Section 6.7 we shall come back to this particular problem; we will provide an argument that shows a way for finding a possible closed-form transformation of $3 \times 3 \times 3$ symmetric-slice arrays into simple form (6.16).

6.4.4 Simplifying symmetric-slice $4 \times 4 \times K$ arrays

In the $4 \times 4 \times K$ case we have $K_{\max} = 10$. So the $4 \times 4 \times 10$ case has been solved. We present simple forms for $K = 1, 2$, as well as for their complementary counterparts $K = 9, 8$, respectively. The remaining cases are still open.

Arrays (=matrices) $\mathbf{X}$ of order $4 \times 4 \times 1$ can be diagonalized by means of the eigenvalue decomposition, so a weight 4 diagonal matrix $\mathbf{D}$ is possible almost surely. Alternative forms, specially useful to compute the $4 \times 4 \times 9$ complement, are possible when $\mathbf{X}$ has eigenvalues of both signs. Specifically, denoting the eigenvalues of $\mathbf{X}$ by

d_1, d_2, d_3, d_4, we may attain the forms

$$\begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & 0 & 2d_3 \\ 0 & 0 & 2d_3 & 0 \end{bmatrix}, \tag{6.17}$$

when $d_1, d_2, d_3 > 0$ and $d_4 < 0$, and

$$\begin{bmatrix} 0 & 2d_1 & 0 & 0 \\ 2d_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2d_3 \\ 0 & 0 & 2d_3 & 0 \end{bmatrix}, \tag{6.18}$$

when $d_1, d_3 > 0$ and $d_2, d_4 < 0$.

Applying the OCM to the simple forms found for symmetric-slice $4 \times 4 \times 1$ arrays ($\mathbf{D}$, (6.17) and (6.18)), we can find various simple forms for $\underline{\mathbf{X}}$ of order $4 \times 4 \times 9$. The form of simplicity of the complement of $\underline{\mathbf{X}}$ is important, since it may lead to different weights via the OCM. When the complement is $\mathbf{D}$, we have

$$\left[\begin{array}{cccc|cccc|cccc|cccc|} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma & 0 & 0 & 0 & 0 \end{array}\right. \tag{6.19}$$

$$\left.\begin{array}{cccc|cccc|cccc|cccc|cccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

of weight 18 (α, β, γ are real entries). When the complement is (6.17) we have

$$\left[\begin{array}{cccc|cccc|cccc|cccc|} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right. \tag{6.20}$$

$$\left.\begin{array}{cccc|cccc|cccc|cccc|cccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

of weight 17 (α, β are real entries). When the complement is (6.18), we have

$$\left[\begin{array}{cccc|cccc|cccc|cccc|} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right. \tag{6.21}$$

$$\left.\begin{array}{cccc|cccc|cccc|cccc|cccc} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \alpha & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right]$$

of weight 16 (α is a real entry). The conclusion is that, in general, a simplification of a $4 \times 4 \times 9$ symmetric-slice array into a weight 18 simple array (6.19) is always possible. This result can be improved, when the signs of the eigenvalues of the orthogonal complement of $\underline{\mathbf{X}}$ permit, to a weight 17 or weight 16 array (6.20) or (6.21), respectively.

Symmetric-slice $4 \times 4 \times 2$ arrays can be simplified according to the type of generalized eigenvalues involved:

- if the generalized eigenvalues are real, $\underline{\mathbf{X}}$ has slices that can be diagonalized simultaneously; this leads to the weight 6 simple form

$$\left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & \gamma & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \beta & 0 & 0 & 0 & \delta & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \tag{6.22}$$

- if there is exactly one pair of complex generalized eigenvalues, the following weight 7 simple form is possible almost surely

$$\left[\begin{array}{cccc|cccc} \alpha & 0 & 0 & 0 & \gamma & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \end{array}\right] \tag{6.23}$$

- if there are two pairs of complex generalized eigenvalues, a weight 8 simple form is possible almost surely,

$$\left[\begin{array}{cccc|cccc} \gamma_1 & 0 & 0 & 0 & 0 & \delta_1\gamma_3 & 0 & 0 \\ 0 & -\gamma_1 & 0 & 0 & \delta_1\gamma_3 & 0 & 0 & 0 \\ 0 & 0 & \gamma_2 & 0 & 0, & 0 & 0 & \delta_2\gamma_4 \\ 0 & 0 & 0 & -\gamma_2 & 0 & 0 & \delta_2\gamma_4 & 0 \end{array}\right], \tag{6.24}$$

with $\delta_1 = \pm 1$, $\delta_2 = \pm 1$.

Finally, applying the OCM to (6.22), (6.23) and (6.24) leads to simple forms for arrays of order $4 \times 4 \times 8$. The first two cases lead to a complement of weight 18,

$$\left[\begin{array}{cccc|cccc|cccc|cccc|} -2\beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2\alpha & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \beta\gamma & 0 & 0 & -\beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \beta\gamma & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta\delta & 0 & 0 & 0 & 0 \end{array}\right. \tag{6.25}$$

$$\left.\begin{array}{cccc|cccc|cccc|cccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right],$$

where α, β and γ have the same meaning as in (6.22) and (6.23), and δ =1 in the first case and -1 in the second case. The third case leads to a complement of the form

$$\left[\begin{array}{cccc|cccc|cccc|cccc|} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\gamma_1\gamma_2^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_1\gamma_2^{-1} & 0 & 0 & \alpha & 0 \end{array}\right. \tag{6.26}$$

$$\left.\begin{array}{cccc|cccc|cccc|cccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right],$$

with $\alpha = -\delta_1\delta_2^{-1}\gamma_3\gamma_4^{-1}$, again of weight 18. The overall conclusion is that a symmetric $4 \times 4 \times 8$ array can almost surely be simplified into one out of two weight 18 arrays.

6.5 Applications to typical rank

How simplicity may help to analyze the (typical) rank of an array was discussed in Section 5.1. Appendix I shows how the simple forms of symmetric-slice arrays of order $3 \times 3 \times 4$ and $3 \times 3 \times 5$ can be used to determine the typical rank of the associated arrays. The procedure used follows closely the one used by Ten Berge *et al.* [108], but the simple forms used greatly simplify the procedure.

For example, Ten Berge *et al.* [108] proved that $3 \times 3 \times 4$ symmetric-slice arrays have typical rank $\{4, 5\}$. Their criterion to decide whether the rank is 4 or 5 relies on the evaluation of the roots of a certain fourth degree polynomial to see if they are real and distinct. With the approach proposed in Appendix I, one can determine the rank of the array by simply inspecting if two specific entries of the array are positive or not. Moreover, exhibiting closed-form CP decompositions in these cases also becomes an easy task.

Also, Ten Berge *et al.* [111] show that $4\times4\times8$ and $4\times4\times9$ arrays with symmetric slices have typical ranks $\{8,9\}$ and $\{9,10\}$, respectively. Their proofs rely on the simple forms (6.25)–(6.26) and (6.19)–(6.21), respectively, that were introduced in the previous Section.

6.6 Maximal simplicity

The maximal simplicity of the arrays considered in Section 6.4 was treated in Appendix I. The overall conclusion is that all the presented simple forms for symmetric-slice arrays of order $I \times I \times K_{\max}$, $2 \times 2 \times K$ and $3 \times 3 \times K$ have maximal simplicity. This means that, in these cases, it is almost surely not possible to decrease the weight of the simple forms.

As for the symmetric-slice arrays of order $4 \times 4 \times K$ treated in Section 6.4.4, no closed form proofs were found that ensure the maximal simplicity of the simple forms previously presented. However, simulations were conducted trying to address this issue. The results given by SIMPLIMAX showed that, for the arrays with 4×4 symmetric slices we considered, weight 18 does indeed seem to be the maximal simplicity to expect for $K = 8$ slices. As for $4 \times 4 \times 9$ symmetric-slice arrays, simulations seem to indicate that weight less than 16 indeed does not happen. Moreover, the situations for which weight 18 and 17 simple forms (6.20) and (6.21) were developed do not seem to admit simpler forms.

6.7 Revisiting the $3 \times 3 \times 3$ case

In Section 6.4.3 we mentioned how a simple form for $3 \times 3 \times 3$ symmetric-slice arrays has eluded us. Simulations showed that simple form

$$\left[\begin{array}{ccc|ccc|ccc} * & 0 & 0 & 0 & 0 & * & 0 & * & 0 \\ 0 & 0 & * & 0 & * & 0 & * & 0 & 0 \\ 0 & * & 0 & * & 0 & 0 & 0 & 0 & * \end{array}\right] \tag{6.27}$$

seems to be attainable around 90% of the times. Therefore, we tried to transform a generic $3 \times 3 \times 3$ symmetric-slice array into simple form (6.27). Although a full proof is still not available, we will present an argument that might provide the key for a solution in the near future.

Following an idea by Ten Berge and Sidiropoulos [107, p. 406], we use the fact that it is almost surely possible to transform any $3 \times 3 \times 3$ symmetric-slice array into an array of the form

$$\underline{\mathbf{H}} = \left[\begin{array}{ccc|ccc|ccc} x^3+1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & y^3+1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & z^3+1 \end{array}\right], \tag{6.28}$$

for real numbers x, y, z. In fact, $\underline{\mathbf{H}}$ is possible because the component matrices of a perfect CP solution of the initial array can be transformed by premultiplication into form

$$\mathbf{A} = \begin{bmatrix} x & 0 & 0 & 1 \\ 0 & y & 0 & 1 \\ 0 & 0 & z & 1 \end{bmatrix}, \tag{6.29}$$

and $\mathbf{H}_i = \mathbf{A}\mathbf{A}_i\mathbf{A}'$ $(i = 1, 2, 3)$, where $\mathbf{A}_i$ is the 4×4 diagonal matrix holding the i-th row of $\mathbf{A}$ in the diagonal. This implies that all frontal, lateral and horizontal slices of $\underline{\mathbf{H}}$ are symmetric matrices. Moreover, the corresponding slices in every direction are equal, ie, the i-th frontal, horizontal and lateral slices coincide for $i = 1, 2, 3$.

We want to investigate if it is possible to further transform $\underline{\mathbf{H}}$ into the simple form

$$\underline{\mathbf{H}}_{\text{simp}} = \left[\begin{array}{ccc|ccc|ccc} k_1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & k_2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & k_3 \end{array}\right], \tag{6.30}$$

for unknown real numbers k_1, k_2, k_3. Similarly to $\underline{\mathbf{H}}$, array $\underline{\mathbf{H}}_{\text{simp}}$ also holds symmetric slices in all directions. Hence, it would be helpful to find a matrix $\mathbf{B}$ such that $\mathbf{B}$ is the CP component matrix (the same in the three directions) for $\underline{\mathbf{H}}_{\text{simp}}$. Moreover, we should look for a matrix $\mathbf{B}$ that spans the same rowspace of $\mathbf{A}$: $\mathbf{B} = \mathbf{S}\mathbf{A}$, for some nonsingular 3×3 matrix $\mathbf{S}$. If we succeed in finding such a matrix $\mathbf{B}$, then transforming $\underline{\mathbf{H}}$ into $\underline{\mathbf{H}}_{\text{simp}}$ boils down to using the transformation matrix $\mathbf{S}$ in each direction for $\underline{\mathbf{H}}$. That is, $\underline{\mathbf{H}}_{\text{simp}} = \mathbf{S}'\mathbf{H}_a(\mathbf{S} \otimes \mathbf{S})$, recall (5.1).

In short, we will have succeeded transforming $\underline{\mathbf{H}}$ into $\underline{\mathbf{H}}_{\text{simp}}$ if we manage to find a 3×4 matrix $\mathbf{B}$ which: (1) is in the row space of $\mathbf{A}$, and (2) is the CP component matrix for $\underline{\mathbf{H}}_{\text{simp}}$ (the same in the three directions). Notice that k_1, k_2, k_3 need also to be estimated in the process.

Start by rescaling the rows of $\mathbf{B}$ to have the first element 1. Also, note that the rows of $\mathbf{B}$ must be orthogonal to the null of the rowspace of $\mathbf{A}$, since $\mathbf{A}$ and $\mathbf{B}$

must span the same rowspace. The null of the rowspace of $\mathbf{A}$ is $[1/x,\ 1/y,\ 1/z,\ -1]'$. Therefore, we can write $\mathbf{B}$ as

$$\mathbf{B} = \begin{bmatrix} 1 & a & b & 1/x + a/y + b/z \\ 1 & c & d & 1/x + c/y + d/z \\ 1 & e & f & 1/x + e/y + f/z \end{bmatrix}. \tag{6.31}$$

$\mathbf{B}$ is completely defined if we are able to compute a, b, c, d, e, f.

The fact that $\mathbf{B}$ is the CP component matrix in all directions for $\underline{\mathbf{H}}_{\text{simp}}$ leads to the following six equations (necessary conditions):

$$1 + a^2c + b^2d + (1/x + a/y + b/z)^2(1/x + c/y + d/z) = 0 \tag{6.32}$$

$$1 + ac^2 + bd^2 + (1/x + a/y + b/z)(1/x + c/y + d/z)^2 = 0 \tag{6.33}$$

$$1 + a^2e + b^2f + (1/x + a/y + b/z)^2(1/x + e/y + f/z) = 0 \tag{6.34}$$

$$1 + ae^2 + bf^2 + (1/x + a/y + b/z)(1/x + e/y + f/z)^2 = 0 \tag{6.35}$$

$$1 + c^2e + d^2f + (1/x + c/y + d/z)^2(1/x + e/y + f/z) = 0 \tag{6.36}$$

$$1 + ce^2 + df^2 + (1/x + c/y + d/z)(1/x + e/y + f/z)^2 = 0. \tag{6.37}$$

These equations originate from decomposing each of the zero entries of $\underline{\mathbf{H}}_{\text{simp}}$ in terms of the entries of component matrix $\mathbf{B}$.

Thus, solving the 6 equations (6.32)–(6.37) with respect to a, b, c, d, e, f would solve the problem. However, solving this system of equations has revealed to be harder than expected. Resorting to computational software (*Mathematica 7.0*, *Maple 12*) did not help in finding a solution. Attempts were made to apply techniques relying on Groebner basis (Buchberger [9]). Unfortunately adequate solutions were not found so far. The only solutions that did come out lead to matrices $\mathbf{B}$ of rank 1. These solutions are useless because they would imply that the slices of $\underline{\mathbf{H}}_{\text{simp}}$ are proportional, which is clearly not the case, see (6.30).

6.8 Discussion

We have worked under the assumption that the arrays are randomly sampled from a continuous distribution, with the constraint of symmetry in the frontal slices. This means that we have ignored cases that arise with probability zero. However, one may question the "random" nature of a core array arising from a 3PCA procedure, as it is a product of an iterative algorithm. As Rocci and Ten Berge [78, p. 362] argue, "...we cannot infer that simplicity transformations which work almost surely for random arrays will also work for Tucker-3 core arrays. Fortunately, all Tucker-3 core arrays encountered so far do seem to behave as if randomly sampled from a continuous distribution, and do allow transformations to simplicity...". Notice that the core array of a 3PCA solution, although not randomly generated (it is even three-way orthogonal), *behaves* as such in what concerns the techniques of typical rank and maximal simplicity that we have explored in this Chapter. Still, a formal proof for this is lacking.

The results of this Chapter have direct implications for the possibility of simplifying core arrays in 3PCA. However, the realm of possible applications is more general. Matrix theory on the simultaneous reduction of pairs of matrices to sparse forms is abundant, but results for more than two matrices seem absent. The present Chapter explores the possibilities of filling this gap. For instance, it has been shown that $3 \times 3 \times 4$ arrays of symmetric slices can almost surely be reduced to a form where each of the four slices has weight 2. This is an extension of matrix theory that will be of interest beyond the realm of 3PCA.

Several simplicity situations for symmetric-slice arrays were addressed in this Chapter. We covered the following situations: symmetric-slice $2 \times 2 \times K$ arrays (all possible K's); symmetric-slice $3 \times 3 \times K$ arrays (all cases except $K = 3$); symmetric-slice $4 \times 4 \times K$ arrays ($K = 1, 2, 8, 9, 10$ or more). We showed how simplicity results can be used to address questions related to typical rank and maximal simplicity.

The $3 \times 3 \times 3$ symmetric-slice case is left open. It was argued how a weight 9 simple pattern seems to apply in the vast majority of the cases. It is still not clear whether all $3 \times 3 \times 3$ symmetric-slice arrays can be transformed into a simple form of weight 9, or which conditions must be satisfied in order for a simple form of weight 9 to be attainable.

Chapter 7

First and second order derivatives for CP and INDSCAL

In Sections 2.3 and 2.4 we introduced the CP and INDSCAL models, respectively. Given a $p \times q \times m$ array $\underline{\mathbf{X}}$ with frontal $p \times q$ slices $\mathbf{X}_i$ ($i = 1, \ldots, m$), CP aims at finding the component matrices $\mathbf{A}$ ($p \times r$), $\mathbf{B}$ ($q \times r$) and $\mathbf{C}$ ($m \times r$) that minimize the function

$$f(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \sum_{i=1}^{m} \|\mathbf{X}_i - \mathbf{A}\mathbf{C}_i\mathbf{B}'\|^2, \tag{7.1}$$

where $\mathbf{C}_i$ is the diagonal matrix holding row i of $\mathbf{C}$ in the diagonal. For the special case when the array has symmetric frontal slices, say $\mathbf{S}_1, \cdots, \mathbf{S}_m$ of order $p \times p$, one can use INDSCAL, which minimizes the function

$$g(\mathbf{A}, \mathbf{C}) = \sum_{i=1}^{m} \|\mathbf{S}_i - \mathbf{A}\mathbf{C}_i\mathbf{A}'\|^2. \tag{7.2}$$

Minimizing f can be done in various ways. Carroll and Chang [11] and Harshman [24] proposed an alternating least-squares method that has become known as the CP-algorithm. However, other approaches have also been proposed, see for instance the references given in Section 2.3.

Minimizing g directly seems difficult, however. Carroll and Chang [11] suggested minimizing f instead. They conjectured that, after convergence, $\mathbf{A}$ and $\mathbf{B}$ will be equal or, at least, columnwise proportional (ie, the columns of $\mathbf{B}$ can be rescaled to match the columns of $\mathbf{A}$, while the columns of $\mathbf{C}$ absorb the inverse scaling). Such matrices will be referred to as being *equivalent*.

Carroll and Chang's conjecture seems to be valid in practical applications. However, counter-examples have already been constructed. Ten Berge and Kiers [102] proved that equivalence may be violated at global minima of f if the slices $\mathbf{S}_i$ are indefinite. When the slices are nonnegative definite and $r = 1$ then equivalence can be violated only at stationary points that do not correspond to global minima. Ten Berge and Kiers [102] conjectured that such stationary points would be local minima. However, Bennani Dosse and Ten Berge [3] proved that such stationary points must be saddle points. This was achieved by analysing the first and second order derivatives of a specific optimization function derived from the loss function of CP. Notice that the result by Bennani Dosse and Ten Berge [3] concerns the case where $r = 1$ component is used. The conjecture of Carroll and Chang seems to be an open issue when $r > 1$ components are used. In our research (see Appendix II (p. 115) for the paper where all details can be found) we aimed at finding a second-order sufficient condition that classifies CP decompositions with $r \geq 1$ components as (local) optima or saddle points. With this tool at hand we conducted a simulation study which sheds some light on the equivalence problem. A similar second-order sufficient condition, this time applied to INDSCAL, was also derived.

In Appendix II (p. 115) we present all the necessary derivations to compute the second-order conditions for CP and INDSCAL. Also, applications of the conditions that were derived are presented in the Appendix.

Summary

In Chapter 1 we presented several definitions and concepts whose comprehension was crucial to fully understand all the subsequent chapters of this thesis. These concepts consist of matrix operators, matrix decompositions, and higher order structures called three-way arrays.

In Chapter 2 we introduced the main three-way models that were in the core of the remainder of the thesis. 3PCA, CP, INDSCAL and the idea of constrained 3PCA models were introduced. Several properties concerning each model were discussed.

Chapter 3 was devoted to the uniqueness of 3PCA and CP solutions. We have shown that 3PCA solutions are typically non-unique. This means that there is some freedom to rotate the component matrices and/or the core array to "simpler" forms. CP solutions, on the other hand, are essentially unique under mild conditions. Several types of necessary and sufficient conditions for uniqueness in CP were discussed. Also, it was argued that some algebraic operations for arrays preserve the uniqueness property. These operations can be useful if one desires to analyse whether a specific CP solution is unique or not.

Degeneracy was the subject of Chapter 4. We explained what characterizes a degenerate CP solution, what can cause degeneracy, and what can be done to try to avoid degeneracy.

In Chapter 5 the issue of simplicity of a 3PCA solution was addressed. The subject was put in more general terms as simplicity of a three-way array in general. Some computational approaches to instill simplicity in three-way arrays were discussed. We presented examples of simplicity transformations for particular families of three-way arrays. The notion of maximal simplicity was also explored.

Chapter 6 was an extension of simplicity concepts introduced in Chapter 5 to arrays with symmetric slices. Some results were presented for arrays with 2×2, 3×3 and 4×4 symmetric slices. Applications of the proven simplicity results to questions around typical rank and maximal simplicity were carried out. We have also explained that a closed-form transformation to simple form for $3 \times 3 \times 3$ symmetric-slice arrays has eluded us. However, an argument possibly leading to a solution in the future was presented.

In Chapter 7 we studied the first and second order derivatives of optimization functions directly related to CP and INDSCAL. The goal was to present a criterion that allows to characterize a stationary point of the optimization functions. Conditions were devised in both unconstrained and constrained scenarios. Applications showed how the differential tools that were worked out can be used.

References

[1] Andersson, C. A., & Bro, R. (1998). Improving the speed of multi-way algorithms: Part I. Tucker3. *Chemometrics and Intelligent Laboratory Systems*, *42*, 93-103.

[2] Andersson, C. A., & Bro, R. (2000). The N-way Toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems*, *52*, 1-4.

[3] Bennani Dosse, M., & Ten Berge, J. M. F. (2008). The assumption of proportional components when Candecomp is applied to symmetric matrices in the context of Indscal. *Psychometrika*, *73*, 303-307.

[4] Bennani Dosse, M., Ten Berge, J. M. F., & Tendeiro, J. (2009). Some new results on orthogonally constrained Candecomp. Submitted.

[5] Bro, R. (1997). PARAFAC, Tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, *38*, 149-171.

[6] Bro, R. (1998). *Multiway analysis in the food industry. Models, algorithms and applications.* Doctoral dissertation, University of Amsterdam.

[7] Brouwer, P., & Kroonenberg, P. (1991). Some Notes on the Diagonalization of the Extended Three-Mode Core Matrix. *Journal of Classification*, *8*, 93-98.

[8] Browne, M. W. (1972). Oblique rotation to a partially specified target. *British Journal of Mathematical and Statistical Psychology*, *25*, 207-212.

[9] Buchberger, B. (1965). *An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal.* Doctoral dissertation, University of Innsbruck. English translation by Michael Abramson in *Journal of Symbolic Computation* (2006), *41*, 475-511.

[10] Cao,Y. Z., Chen, Z. P., Mo, C. Y., Wu, H. L., & Yu, R. Q. (2000). A PARAFAC algorithm using penalty diagonalization error (PDE) for three-way data array resolution. *The Analyst*, *125*, 2303-2310.

[11] Carroll, J. D., & Chang, J. J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition. *Psychometrika*, *35*, 283-319.

[12] Catalisano, M. V., Geramita, A. V., & Gimigliano, A. (2002). Ranks of tensors, secant varieties of Segre varieties and fat points. *Linear Algebra and its Applications*, *335*, 263-285. Erratum, *Linear Algebra and its Applications*, *367*, 347-348.

[13] Cattell, R. B. (1944). "Parallel Proportional Profiles" and other principles for determining the choice of factors by rotation. *Psychometrika*, *9*, 267-283.

[14] Choulakian, V. (2009). Some numerical results on the rank of generic three-way arrays over $\mathbb{R}$. arXiv:0906.0198v1 [stat.CO], 1 June 2009.

[15] Comon, P., Ten Berge, J. M. F., De Lathauwer, L., & Castaing, J. (2009). Generic and typical ranks of multi-way arrays. *Linear Algebra and its Applications*, *430*, 2997-3007.

[16] De Lathauwer, L. (2006). A link between the Canonical Decomposition in Multilinear Algebra and Simultaneous Matrix Diagonalization. *SIAM J. Matrix Anal. Appl.*, *28*, 642-666.

[17] De Lathauwer, L., De Moor, B., & Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, *21*, 1253-1278.

[18] De Lathauwer, L., De Moor, B., & Vandewalle, J. (2004). Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition. *SIAM J. Matrix Anal. Appl.*, *26*, 295-327.

[19] De Silva, V, & Lim, L.-H. (2008). Tensor rank and the ill-posedness of the best low-rank approximation problem. arXiv:math/0607647v2 [math.NA], 1 Apr 2008.

[20] Eckart, G, & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, *1*, 211-218.

[21] Faber, N. M., Bro, R., & Hopke, P. K. (2003). Recent developments in CANDECOMP/PARAFAC algorithms: a critical review. *Chemometrics and Intelligent Laboratory Systems*, *65*, 119-137.

[22] Fackler, P. L. (2009), *Notes on Matrix Calculus.* Retrieved on 3 February 2009, from http://www4.ncsu.edu/~pfackler/MatCalc.pdf.

[23] Friedland, S. (2009). On the generic rank of 3-tensors. ArXiv:0805.3777v3[math.AG], 29 July 2009.

[24] Harshman, R. A. (1970). Foundations of the Parafac procedure: models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, *16*, 1-84.

[25] Harshman, R. A. (1972). Determination and proof of minimum uniqueness conditions for PARAFAC1. *UCLA Working Papers in Phonetics*, *22*, 111-117.

[26] Harshman, R. A. (2001). An index formalism that generalizes the capability of matrix notation and algebra to n-way arrays. *Journal of Chemometrics*, *15*, 689-714.

[27] Harshman, R. A., & Lundy, M. E. (1984). The PARAFAC model for three-way factor analysis and multidimentional scaling. In: H. G. Law, C. W. Snyder, J. A. Hattie, R. P. McDonald (Eds.), *Research Methods for Multimode Data Analysis* (pp. 122-215). New York: Praeger.

[28] Harshman, R. A., & Lundy, M. E. (1984). Data preprocessing and the extended PARAFAC model. In: H. G. Law, C. W. Snyder, J. A. Hattie, R. P. McDonald (Eds.), *Research Methods for Multimode Data Analysis* (pp. 216-284). New York: Praeger.

[29] Hitchcock, F. L. (1927). Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, *7*, 39-70.

[30] Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, *6*, 164-189.

[31] Hopke, P. K., Paatero, P., Jia, H., Ross, R. T., & Harshman, R. A. (1998). Three-way (Parafac) factor analysis: examination and comparison of alternative computational methods as applied to ill-conditioned data. *Chemometrics and Intelligent Laboratory Systems*, *43*, 25-42.

[32] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *24*, 417-441, 498-520.

[33] Jiang, T., & Sidiropoulos, N. D. (2004). Kruskal's permutation lemma and the identification of Candecomp/Parafac and bilinear models with constant modulus constraint. *IEEE Transactions on Signal Processing*, *52*, 2625-2636.

[34] Jolliffe, I. T. (2002). Principal Component Analysis. Springer-Verlag: New York, 2nd Edition.

[35] Kapteyn, A., Neudecker, H., & Wansbeek, T. (1986). An approach to n-mode component analysis. *Psychometrika*, *51*, 269-275.

[36] Kiers, H. A. L. (1992). TUCKALS core rotations and constrained TUCKALS modeling. *Statistica Applicata*, *4*, 659-667.

[37] Kiers, H. A. L. (1994). SIMPLIMAX: Oblique rotation to an optimal target with simple structure. *Psychometrika*, *59*, 567-579.

[38] Kiers, H. A. L. (1997). Three-mode orthomax rotation. *Psychometrika*, *62*, 579-598.

[39] Kiers, H. A. L. (1998). A three-step algorithm for CANDECOMP/PARAFAC analysis of large data sets with multicollinearity. *Journal of Chemometrics*, *12*, 155-171.

[40] Kiers, H. A. L. (1998). Three-way SIMPLIMAX for oblique rotation of the three-mode factor analysis core to simple structure. *Computational Statistics & Data Analysis*, *28*, 307-324.

[41] Kiers, H. A. L. (2000). Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, *14*, 105-122.

[42] Kiers, H. A. L., Ten Berge, J. M. F., & Rocci, R. (1997). Uniqueness of three-mode factor models with sparse cores: the $3 \times 3 \times 3$ case. *Psychometrika*, *62*, 349-374.

[43] Kiers, H. A. L., & Van Mechelen, I. (2001). Three-way component analysis: Principles and illustrative application. *Psychological Methods*, *6*, 84-110.

[44] Krijnen, W. P. (1983). *The analysis of three-way arrays by constrained PARAFAC methods.* Leiden: DSWO Press.

[45] Krijnen, W.P., Dijkstra, T.K., & Stegeman A. (2008). On the non-existence of optimal solutions and the occurrence of "degeneracy" in the CANDECOMP/PARAFAC model. *Psychometrika*, *73*, 431-439.

[46] Kroonenberg, P. M. (2008). *Applied Multiway Data Analysis*. John Wiley & Sons, Inc.

[47] Kroonenberg, P. M. (1983). *Three-mode Principal Component Analysis. Theory and Applications*. Leiden: DSWO Press.

[48] Kroonenberg, P. M., & De Leeuw, J. (1980). Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, *45*, 69-97.

[49] Kruskal, J. B. (1977). Three-Way Arrays: Rank and Uniqueness of Trilinear Decompositions, with Applications to Arithmetic Complexity and Statistics. *Linear Algebra and its Applications*, *18*, 95-138.

[50] Kruskal, J. B. (1988). Simple structure for three-way data: A new method intermediate between 3-mode factor analysis and PARAFAC-CANDECOMP. Paper presented at the *53rd Annual Meeting of the Psychometric Society*, Los Angeles.

[51] Kruskal, J. B. (1989). Rank, decomposition, and uniqueness for 3-way and N-way arrays. In: R. Coppi and S. Bolasco (Eds.), *Multiway Data Analysis* (pp. 7-18). Amsterdam, Netherlands: North-Holland.

[52] Kruskal, J. B., Harshman, R. A., & Lundy, M. E., Some relationships between Tucker's three-mode factor analysis and PARAFAC/CANDECOMP. Paper presented at the annual meeting of the Psychometric Society, Los Angeles, 1983.

[53] Kruskal, J. B., Harshman, R. A., & Lundy, M. E., Several mathematical relationships between PARAFAC-CANDECOMP and three-mode factor analysis. Paper presented at the annual meeting of the Classification Society, St. John's, Newfoundland, 1985.

[54] Kruskal, J. B., Harshman, R. A., & Lundy, M. E. (1989). How 3-MFA data can cause degenerate Parafac solutions, among other relationships. In: R. Coppi and S. Bolasco (Eds.), *Multiway Data Analysis* (pp. 115-122). Amsterdam, Netherlands: North-Holland.

[55] Lastovicka, J. L. (1981). The extension of component analysis to four-mode matrices. *Psychometrika*, *46*, 47-57.

[56] Leurgans, S. E., Ross, R. T., & Abel, R. B. (1993). A decomposition for three-way arrays. *SIAM J. Matrix Anal. Appl.*, *14*, 1064-1083.

[57] Lim, L.-H. (2005). Optimal solutions to non-negative Parafac/Multilinear NMF always exist. Talk at the *Workshop on Tensor Decompositions and Applications*, 29 August-2 September, CIRM, Luminy, Marseille, France.

[58] Liu, X., & Sidiropoulos, N. D. (2001). Cramer-Rao lower bounds for low-rank decomposition of multidimensional arrays. *IEEE Transactions on Signal Processing*, *49*, 2074-2086.

[59] MacCallum, R. C. (1976). Transformation of a three-mode multidimensional scaling solution to INDSCAL form. *Psychometrika*, *41*, 385-400.

[60] Magnus, J. R., & Neudecker, H. (2007). *Matrix Differential Calculus with Applications in Statistics and Econometrics.* John Wiley & Sons: Chichester/New York, 3rd Edition.

[61] Mitchell, B. C., & Burdick, D. S. (1994). Slowly converging Parafac sequences: swamps and two-factor degeneracies. *Journal of Chemometrics*, *8*, 155-168.

[62] Mulaik, S. A. (1972). The foundations of factor analysis. McGraw-Hill.

[63] Murakami, T. (1999). A simple core does not necessarily facilitate interpretations in three-mode principal components analysis. Research Memorandum. Nagoya, Japan: Nagoya University.

[64] Murakami, T. (1983). Quasi three-mode principal component analysis-A method for assessing factor change. *Behaviormetrika*, *14*, 27-48.

[65] Murakami, T., Ten Berge, J. M. F., & Kiers, H. A. L. (1998). A case of extreme simplicity of the core matrix in three-mode principal components analysis. *Psychometrika*, *63*, 255-261.

[66] Navasca, C., De Lathauwer, L., & Kindermann, S. (2008). Swamp reducing technique for tensor decomposition. Proceedings of the *16th European Signal Processing Conference* (EUSIPCO 2008), Lausanne, Switzerland, 25-29 August 2008.

[67] Paatero, P. (1997). A weighted non-negative least squares algorithm for three-way 'PARAFAC' factor analysis. *Chemometrics and Intelligent Laboratory systems*, *38*, 223-242.

[68] Paatero, P. (1999). The Multilinear Engine– a Table-Driven, Least Squares Program for Solving Multlinear Problems, Including the n-way Parallel Factor Analysis Model. *Journal of Computational and Graphical Statistics*, *8*, 854-888.

[69] Paatero, P. (2000). Construction and analysis of degenerate PARAFAC models. *Journal of Chemometrics*, *14*, 285-299.

[70] Paatero, P., & Andersson, C. A. (1999). Further improvements of the speed of the Tucker3 three-way algorithm. *Chemometrics and Intelligent Laboratory Systems*, *47*, 17-20.

[71] Pearson, K. (1901). On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, *2*, 559-572.

[72] Pravdova, V., Estienne, F., Walczak, B., & Massart, D. L. (2001). A robust version of the Tucker3 model. *Chemometrics and Intelligent Laboratory Systems*, *59*, 75-88.

[73] Rajih, M., Comon, P., & Harshman, R. A. (2008). Enhanced line search: a novel method to accelerate Parafac. *SIAM J. Matrix Anal. Appl.*, *30*, 1148-1171.

[74] Rayens, W. S., & Mitchell, B. C. (1997). Two-factor degeneracies and a stabilization of PARAFAC. *Chemometrics and Intelligent Laboratory Systems*, *38*, 173-181.

[75] Rhodes, J. (2010). A concise proof of Kruskal's theorem on tensor decomposition. *Linear Algebra and its Applications*, *432*, 1818-1824.

[76] Rocci, R. (2001). Core matrix rotation to natural zeros in three-mode factor analysis. In S. Borra, R. Rocci, M. Vichi, M. Schader (Eds.). *Advances in classification and data analysis* (pp. 161-168). Berlin, Germany: Springer-Verlag.

[77] Rocci,R., & Ten Berge, J. M. F. (1994). A simplification of a result by Zellini on the maximal rank of symmetric three-way arrays. *Psychometrika*, *59*, 377-380.

[78] Rocci, R., & Ten Berge, J. M. F. (2002). Transforming three-way arrays to maximal simplicity. *Psychometrika*, *67*, 351-365.

[79] Sidiropoulos, N. D., & Bro, R. (2000). On the uniqueness of multilinear decomposition of N-way arrays. *Journal of Chemometrics*, *14*, 229-239.

[80] Smilde, A., Bro, R. & Geladi, P. (2004). *Multi-way analysis. Applications in the Chemical Sciences*. John Wiley & Sons, Ltd.

[81] Spearman, C. (1904). General intelligence, objectively determined and measured. *American Journal of Psychology*, *15*, 201–293.

[82] Stegeman, A. (2006). Degeneracy in Candecomp/Parafac explained for $p \times p \times 2$ arrays of rank $p + 1$ or higher. *Psychometrika*, *71*, 483-501.

[83] Stegeman, A. (2007). Degeneracy in Candecomp/Parafac and Indscal explained for several three-sliced arrays with a two-valued typical rank. *Psychometrika*, *72*, 601-619.

[84] Stegeman, A. (2008). Low-rank approximation of generic $p \times q \times 2$ arrays with diverging components in the Candecomp/Parafac model. *SIAM J. Matrix Anal. Appl.*, *30*, 988-1007.

[85] Stegeman, A. (2009). On uniqueness conditions for Candecomp/Parafac and Indscal with full column rank in one mode. *Linear Algebra and its Applications*, *431*, 211-227.

[86] Stegeman, A. (2009). Using the simultaneous generalized Schur decomposition as a Candecomp/Parafac algorithm for ill-conditioned data. *Journal of Chemometrics*, *23*, 385-392.

[87] Stegeman, A., & Comon, P. (2009). Subtracting a best rank-1 approximation may increase tensor rank, arXiv:0906.0483v1 [math.AG], 2 June 2009.

[88] Stegeman, A. & De Lathauwer, L. (2009). A method to avoid diverging components in the Candecomp/Parafac model for generic $I \times J \times 2$ arrays. *SIAM J. Matrix Anal. Appl.*, *30*, 1614-1638.

[89] Stegeman, A., & Sidiropoulos, N. D. (2007). On Kruskal's uniqueness condition for the Candecomp/Parafac decomposition. *Linear Algebra and its Applications*, *420*, 540-552.

[90] Stegeman, A., & Ten Berge, J. M. F. (2006). Kruskal's condition for uniqueness in Candecomp/Parafac when ranks and k-ranks coincide. *Computational Statistics & Data Analysis*, *50*, 210-220.

[91] Stegeman, A., Ten Berge, J. M. F., & De Lathauwer, L. (2006). Sufficient conditions for uniqueness in Candecomp/Parafac and Indscal with random component matrices. *Psychometrika*, *71*, 219-229.

[92] Strassen, V. (1983). Rank and optimal computation of generic tensors. *Linear Algebra and its Applications*, *52*, 645-685.

[93] Ten Berge, J. M. F. (1984). A joint treatment of Varimax rotation and the problem of diagonalizing symmetric matrices simultaneously in the least-squares sense. *Psychometrika*, *49*, 347-358.

[94] Ten Berge, J. M. F. (1991). Kruskal's polynomial for $2 \times 2 \times 2$ arrays and a generalization to $2 \times n \times n$ arrays. *Psychometrika*, *56*, 631-636.

[95] Ten Berge, J. M. F. (1993). Least squares optimization in Multivariate Analysis. DSWO Press, Leiden. Retrieved on 3 October 2006, from http://www.rug.nl/psy/onderzoek/onderzoeksprogrammas/pdf/leastsquaresbook.pdf.

[96] Ten Berge, J. M. F. (2000). The typical rank of tall three-way arrays. *Psychometrika*, *65*, 525-532.

[97] Ten Berge, J. M. F. (2004). Partial uniqueness in CANDECOMP/PARAFAC. *Journal of Chemometrics*, *18*, 12-16.

[98] Ten Berge, J. M. F. (2004). Simplicity and typical rank of three-way arrays, with applications to Tucker-3 analysis with simple cores. *Journal of Chemometrics*, *18*, 17-21.

[99] Ten Berge, J. M. F. (2007). Simp442.m. Why does it work?. Unpublished Note, Heymans Institute of Psychological Research, University of Groningen, The Netherlands.

[100] Ten Berge, J. M. F. (2007). The intersecting subspace idea and the 3-3-3 array (symmetric). Unpublished Note, Heymans Institute of Psychological Research, University of Groningen, The Netherlands.

[101] Ten Berge, J. M. F., De Leeuw, J., & Kroonenberg, P. M. (1987). Some additional results on principal components analysis of three-mode data by means of alternating least squares. *Psychometrika*, *52*, 183-191.

[102] Ten Berge, J. M. F., & Kiers, H. A. L. (1991). Some clarifications of the Candecomp algorithm applied to Indscal. *Psychometrika*, *56*, 317-326.

[103] Ten Berge, J. M. F., & Kiers, H. A. L. (1999). Simplicity of core arrays in three-way principal component analysis and the typical rank of $P \times Q \times 2$ arrays. *Linear Algebra and its Applications*, *294*, 169-179.

[104] Ten Berge, J. M. F., Kiers, H. A. L., & De Leeuw, J. (1988). Explicit CANDECOMP/PARAFAC solutions for a contrived $2 \times 2 \times 2$ array of rank three. *Psychometrika*, *53*, 579-583.

[105] Ten Berge, J. M. F., Kiers, H. A. L., Murakami, T. & Van der Heijden, R. (2000). Transforming three-way arrays to multiple orthonormality. *Journal of Chemometrics*, *14*, 275-284.

[106] Ten Berge, J. M. F., Knol, D. L., & Kiers, H. A. L. (1988). A treatment of the Orthomax rotation family in terms of diagonalization, and a re-examination of a singular value approach to Varimax rotation. *Computational Statistics Quarterly*, *3*, 207-217.

[107] Ten Berge, J. M. F., & Sidiropoulos, N. D. (2002). On uniqueness in Candecomp/Parafac. *Psychometrika*, *67*, 399-409.

[108] Ten Berge, J. M. F., Sidiropoulos, N. D., Rocci, R. (2004). Typical rank and indscal dimensionality for symmetric three-way arrays of order $I \times 2 \times 2$ or $I \times 3 \times 3$. *Linear Algebra and its Applications*, *388*, 363-377.

[109] Ten Berge, J. M. F., & Smilde, A. K. (2002). Non-triviality and identification of a constrained Tucker3 analysis. *Journal of Chemometrics*, *16*, 609-612.

[110] Ten Berge, J. M. F., & Stegeman, A. (2006). Symmetry transformations for square sliced three-way arrays, with applications to their typical rank. *Linear Algebra and its Applications*, *418*, 215-224.

[111] Ten Berge, J. M. F., Stegeman, A., & Bennani Dosse, M. (2009). The Carroll and Chang conjecture of equal Indscal components when Candecomp/Parafac gives perfect fit. *Linear Algebra and its Applications*, *430*, 818-829.

[112] Ten Berge, J. M. F., & Tendeiro, J. N. (2009). The link between sufficient conditions by Harshman and by Kruskal for uniqueness in Candecomp/Parafac. *Journal of Chemometrics*, *23*, 321-323.

[113] Tomasi, G., & Bro, R. (2006). A comparison of algorithms for fitting the PARAFAC model. *Computational Statistics & Data Analysis*, *50*, 1700-1734.

[114] Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*, 279-311.

[115] Uhlig, F. (1973). Simultaneous block diagonalization of two real symmetric matrices. *Linear Algebra and its Applications*, *7*, 281-289.

[116] Weesie, H. M., & Van Houwelingen, J. C. (1983). GEPCAM User's Manual. Unpublished manuscript, University of Utrecht, Institute for Mathematical Statistics, The Netherlands.

[117] Zhao, H.-G. (2008). A heuristic method for computing the best rank-r approximation to higher-order tensors. *International Journal of Comtemporary Mathematical Sciences*, *3*, 471-476.

Appendices

Appendix I

Tendeiro, J. N., Ten Berge, J. M. F., & Kiers, H. A. L. (2009).
Simplicity transformations for three-way arrays with symmetric slices, and applications to Tucker-3 models with sparse core arrays.
Linear Algebra and its Applications, *430*, 924-940.

Contents lists available at ScienceDirect

Linear Algebra and its Applications

journal homepage: www.elsevier.com/locate/laa

Simplicity transformations for three-way arrays with symmetric slices, and applications to Tucker-3 models with sparse core arrays

Jorge N. Tendeiro*, Jos M.F. Ten Berge, Henk A.L. Kiers

Department of Psychology, University of Groningen, 9700 AV Groningen, The Netherlands

ARTICLE INFO

Article history:
Received 21 February 2008
Accepted 2 September 2008
Available online 22 November 2008

Submitted by R.A. Brualdi

AMS classification:
15A69

Keywords:
Three-mode component analysis
Candecomp
Parafac
Typical tensorial rank
Tucker transformations
Maximal simplicity
Sparse arrays

ABSTRACT

Tucker three-way PCA and Candecomp/Parafac are two well-known methods of generalizing principal component analysis to three way data. Candecomp/Parafac yields component matrices **A** (e.g., for subjects or objects), **B** (e.g., for variables) and **C** (e.g., for occasions) that are typically unique up to jointly permuting and rescaling columns. Tucker-3 analysis, on the other hand, has full transformational freedom. That is, the fit does not change when **A**, **B**, and **C** are postmultiplied by nonsingular transformation matrices, provided that the inverse transformations are applied to the so-called core array $\underline{\mathbf{G}}$. This freedom of transformation can be used to create a simple structure in **A**, **B**, **C**, and/or in $\underline{\mathbf{G}}$. This paper deals with the latter possibility exclusively. It revolves around the question of how a core array, or, in fact, any three-way array can be transformed to have a maximum number of zero elements. Direct applications are in Tucker-3 analysis, where simplicity of the core may facilitate the interpretation of a Tucker-3 solution, and in constrained Tucker-3 analysis, where hypotheses involving sparse cores are taken into account. In the latter cases, it is important to know what degree of sparseness can be attained as a tautology, by using the transformational freedom. In addition, simplicity transformations have proven useful as a mathematical tool to examine rank and generic or typical rank of three-way arrays. So far, a number of simplicity results have been attained, pertaining to arrays sampled randomly from continuous distributions. These results do not apply to three-way arrays with symmetric slices in one direction. The present paper offers a number of simplicity results for arrays with symmetric slices of order $2 \times 2, 3 \times 3$ and 4×4. Some generalizations to higher orders

* Corresponding author.
E-mail addresses: j.n.tendeiro@rug.nl (J.N. Tendeiro), j.m.f.ten.berge@rug.nl (J.M.F. Ten Berge), h.a.l.kiers@rug.nl (H.A.L. Kiers).

0024-3795/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.laa.2008.09.020

are also discussed. As a mathematical application, the problem of determining the typical rank of $4 \times 3 \times 3$ and $5 \times 3 \times 3$ arrays with symmetric slices will be revisited, using a sparse form with only eight out of 36 elements nonzero for the former case and 10 out of 45 elements nonzero for the latter one, that can be attained almost surely for such arrays. The issue of maximal simplicity of the targets to be presented will be addressed, either by formal proofs or by relying on simulation results.

1. Introduction

Two of the most popular methods of component analysis for three-way arrays are Candecomp/Parafac, henceforth CP [1,3], and Tucker three-way PCA [17], henceforth 3PCA. For a three-way data array $\underline{\mathbf{X}}$ of format $I \times J \times K$, CP yields component matrices $\mathbf{A}(I \times R)$, $\mathbf{B}(J \times R)$, and $\mathbf{C}(K \times R)$, such that $\sum_{k=1}^{K} tr(\mathbf{E}_k'\mathbf{E}_k)$ is minimized in the decomposition

$$\underline{\mathbf{X}} = \sum_{r=1}^{R}(\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r) + \underline{\mathbf{E}}, \tag{1}$$

where $\mathbf{a}_r$, $\mathbf{b}_r$, $\mathbf{c}_r$ are columns r of $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$, respectively, and $\mathbf{E}_k$ denotes the kth slice of order $I \times J$ of the residual array $\underline{\mathbf{E}}$. It can be seen that an R-component CP solution approximates the data as the sum of R outer products of the form $(\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r)$, $r = 1, \ldots, R$. Equivalently, each frontal slice $\mathbf{X}_k$ of $\underline{\mathbf{X}}$ is decomposed as

$$\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}' + \mathbf{E}_k, \tag{2}$$

where $\mathbf{C}_k$ is the diagonal matrix holding the elements from row k of $\mathbf{C}$.

Like CP, 3PCA approximates the data array as a sum of outer products of columns of $\mathbf{A}(I \times P)$, $\mathbf{B}(J \times Q)$, and $\mathbf{C}(K \times R)$, but now every outer product of one of the P columns of $\mathbf{A}$, one of the Q columns of $\mathbf{B}$, and one of the R columns of $\mathbf{C}$ is involved, with P, Q, and R possibly different. In addition, each of these PQR outer products is weighted when it enters the sum. The weights are collected in the so-called core array $\underline{\mathbf{G}}$ of format $P \times Q \times R$. Specifically, 3PCA minimizes the function $\sum_{k=1}^{K} tr(\mathbf{E}_k'\mathbf{E}_k)$ in the decomposition

$$\underline{\mathbf{X}} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr}(\mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r) + \underline{\mathbf{E}}, \tag{3}$$

where $\mathbf{a}_p, \mathbf{b}_q, \mathbf{c}_r$ are columns p, q, r of $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$, respectively, and g_{pqr} are entries from $\underline{\mathbf{G}}$. Equivalently, 3PCA can be expressed as

$$\mathbf{X}_k \approx \mathbf{A}\left(\sum_{r=1}^{R} c_{kr}\mathbf{G}_r\right)\mathbf{B}', \quad k = 1, 2, \ldots, K, \tag{4}$$

where $\mathbf{X}_k$ and $\mathbf{G}_r$ denote frontal slices of $\underline{\mathbf{X}}$ and $\underline{\mathbf{G}}$, respectively. The parameters are estimated by minimizing the sum of squared residuals for fixed numbers of components in each mode [7].

Under mild conditions, a solution for CP is essentially unique [4,8,9]. That is, only joint permutations and rescaling of columns of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ will leave the fitted part of the solution unaltered. In 3PCA, on the other hand, there is no such uniqueness. In fact, the core array can be transformed in the three directions, as long as the inverse transformations are applied to the component matrices. Specifically, the slabs $\mathbf{G}_1, \ldots, \mathbf{G}_R$ can be transformed to $\mathbf{G}_1^*, \ldots, \mathbf{G}_R^*$ by means of the *Tucker transformation*

$$\mathbf{G}_l^* = \mathbf{S}'\left(\sum_{r=1}^{R} u_{rl}\mathbf{G}_r\right)\mathbf{T}, \quad l = 1, 2, \ldots, R, \tag{5}$$

where $\mathbf{S}$, $\mathbf{T}$, and $\mathbf{U}$, holding elements u_{rl}, are nonsingular, provided that the component matrices are counter-transformed into $\mathbf{A}^* = \mathbf{A}(\mathbf{S}')^{-1}$, $\mathbf{B}^* = \mathbf{B}(\mathbf{T}')^{-1}$ and $\mathbf{C}^* = \mathbf{C}(\mathbf{U}')^{-1}$ [12,17]. Expression (5) can also be written as

$$\mathbf{G}^*_{\mathrm{Vec}} = (\mathbf{T}' \otimes \mathbf{S}')\mathbf{G}_{\mathrm{Vec}}\mathbf{U}, \tag{6}$$

where $\mathbf{G}_{\mathrm{Vec}} = [\mathrm{Vec}(\mathbf{G}_1)|\cdots|\mathrm{Vec}(\mathbf{G}_R)]$ is a vectorised version of $\underline{\mathbf{G}}$, and $\mathbf{G}^*_{\mathrm{Vec}}$ is defined analogously.

One of the problems that are often associated to 3PCA is the difficulty to interpret a solution. The main reason is that the solution involves so many triplets of components. The number of these triplets is the number of relevant nonzero entries in the core array. This situation can be improved if it is possible to transform the core array so that it holds as many zero entries as possible. Such a "simple" core will decrease the number of joint impacts of triples of components from **A**, **B** and **C**, which may facilitate the interpretation. This is a first reason for considering simplicity transformations for core arrays.

Results on simplicity transformations may also be useful to distinguish between tautologies and non-trivial models. Often, researchers impose constraints on the core, based on theory. Once it is proven that a certain simple form is achievable almost surely (via Tucker transformations) for arrays of a given format, then that simple form adds nothing new as a model. The theory is then beyond falsification, and the researcher needs to impose more or different constraints in order to construct a meaningful model.

Apart from being useful for applied 3PCA with or without constraints on the core, transformations to simplicity also may serve as a tool itself for the mathematical study of three-way arrays. For instance, Ten Berge and Kiers [13] and Ten Berge [14] have given results on typical rank (over the real field) of array formats that became obvious after certain simplicity transformations. It is important to note that, although an array is altered when Tucker transformations are used, the rank remains unaffected. This means that one has the freedom to transform an array by Tucker transformations without affecting its rank.

2. Symmetry preserving Tucker transformations

Simplicity transformations for three-way arrays have received considerable attention, e.g. Kiers [5,6], Murakami et al. [10], Ten Berge and Kiers [13], Ten Berge et al. [15], and Rocci and Ten Berge [12]. However, whatever closed-form simplicity results have been obtained, they apply to arrays the elements of which are randomly sampled from a continuous distribution. In practice, data sets frequently contain symmetric slices. For such cases the above results do not hold. Research on simplicity for arrays that have *symmetric* slices in one direction is still absent. The present paper offers simplicity results for the latter type of array.

From now on we assume that $\underline{\mathbf{X}}$ is an $I \times I \times K$ array with K symmetric frontal slices $\mathbf{X}_k$ of order $I \times I, k = 1, \ldots, K$. We assume that these slices are linearly independent, otherwise, we transform the superfluous slices to zero via a suitable transformation (5), reducing the dimensionality of the problem. Because the space of real symmetric matrices of order $I \times I$ has dimension $I(I+1)/2$, the number K of symmetric slices to consider will not exceed $K_{\max} = I(I+1)/2$.

We will usually work under the assumption that $\underline{\mathbf{X}}$ is randomly sampled from a continuous distribution, with the constraint that slices are symmetric in one direction. Phenomena that arise "almost surely" are those that arise with probability one under this circumstance.

We are looking for symmetry-preserving transformations of $\underline{\mathbf{X}}$ which yield an array $\underline{\mathbf{H}}$ with a large number of zero elements. So our goal is to determine nonsingular matrices **S**, **U** such that

$$\mathbf{H}_l = \mathbf{S}'\left(\sum_{k=1}^{K} u_{kl}\mathbf{X}_k\right)\mathbf{S}, \quad l = 1, 2, \ldots, K, \tag{7}$$

where u_{kl} is an element of **U**, has as many zero entries as possible. The number of nonzero entries in $\underline{\mathbf{H}}$ will be referred to as the *weight* of $\underline{\mathbf{H}}$.

It may be noted that we have tacitly assumed in (7) that **S** and **T** of (5) can be constrained to be equal. In fact, this is a simplification, because symmetry preserving transformations with **S** and **T** different do exist. However, this is possible only for two-slice arrays. Moreover, setting **S** and **T** equal has not been detrimental at all in our search for transformations that minimize the weight of arrays.

3. A symmetric version of the orthogonal complement method

Rocci and Ten Berge [12] have developed the so-called orthogonal complement method (henceforth OCM), which permits transforming an array to simple form by using the previously known simplicity transformation of a so-called complementary array. For instance, the seven frontal slices of a $3 \times 3 \times 7$ array $\underline{\mathbf{X}}$ form a 7-dimensional subspace in 9-space, and a complementary array $\underline{\mathbf{X}}^c$ would be any $3 \times 3 \times 2$ array, the slices of which are linearly independent, and trace-orthogonal to the seven slices of $\underline{\mathbf{X}}$. OCM is based on the observation that transformations that simplify $\underline{\mathbf{X}}^c$ may also be used to simplify $\underline{\mathbf{X}}$ itself. For instance, suppose $\underline{\mathbf{X}}^c$ can be transformed to contain two diagonal 3×3 matrices. The subspace spanned by the seven slices of $\underline{\mathbf{X}}$ contains one diagonal matrix and six matrices having just one nonzero element. Finding the linear combinations that produce those seven matrices in a known subspace is easy, which means that the OCM will transform $\underline{\mathbf{X}}$ into an array that has 54 zero elements out of 63.

The OCM has played a key role in obtaining some important results of this paper. However, to be useful in the present context, it needed to be adjusted in two respects. First, the dimensionality of the space of $I \times I$ matrices is I^2 in general, but it is only $I(I+1)/2$ in case of symmetry. So $\underline{\mathbf{X}}$ can be written in a slice-wise vectorised version $\mathbf{X}_{\text{Vec}} = [\text{Vec}(\mathbf{X}_1)|\cdots|\text{Vec}(\mathbf{X}_K)]$ of order $(1/2I(I+1)) \times K$. Second, we need to constrain the transformations to have $\mathbf{S}$ and $\mathbf{T}$ equal. This amounts to the following general setup of OCM for symmetric slice arrays:

Step 1. Compute an orthogonal complement of $\mathbf{X}_{\text{Vec}}$, say $\mathbf{X}^c_{\text{Vec}}$.

Step 2. Compute $\mathbf{H}^c_{\text{Vec}} = (\mathbf{S}^{-1} \otimes \mathbf{S}^{-1})\mathbf{X}^c_{\text{Vec}}\mathbf{V}$ in such a way that $\mathbf{H}^c_{\text{Vec}}$ is in simple form.

Step 3. Compute a simple orthogonal complement of $\mathbf{H}^c_{\text{Vec}}$, say $\mathbf{H}_{\text{Vec}}$.

Step 4. Find the matrix $\mathbf{U}$ such that $\mathbf{H}_{\text{Vec}} = (\mathbf{S}' \otimes \mathbf{S}')\mathbf{X}_{\text{Vec}}\mathbf{U}$. Array $\underline{\mathbf{H}}$, reconstructed from $\mathbf{H}_{\text{Vec}}$, is the simple form found for $\underline{\mathbf{X}}$.

In the next sections, simplicity results for various array formats will be presented. We will denote the array to be simplified by $\underline{\mathbf{X}}$, with symmetric frontal slices $\mathbf{X}_1, \ldots, \mathbf{X}_K$. The simple form to be obtained from $\underline{\mathbf{X}}$ will be denoted by $\underline{\mathbf{H}}$, with symmetric frontal slices $\mathbf{H}_1, \ldots, \mathbf{H}_K$.

We start with the case where the set of symmetric slices in an array is space-filling.

4. Simplifying symmetric slice $I \times I \times K_{\max}$ arrays

When we have the maximum number of linearly independent symmetric frontal slices, the set of the frontal slices forms a basis for the space of symmetric $I \times I$ matrices. Denote the ith column of $\mathbf{I}_I$ by $\mathbf{e}_i$. A simple basis for the same space is formed by matrices $\mathbf{e}_i\mathbf{e}_i'$ for $i = 1, 2, \ldots, I$, and $(\mathbf{e}_i\mathbf{e}_j' + \mathbf{e}_j\mathbf{e}_i')$ for $1 \leqslant i < j \leqslant I$ [11]. For example, when $I = 3$ the basis is

$$\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix}, \begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix}, \begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix}, \begin{bmatrix}0&1&0\\1&0&0\\0&0&0\end{bmatrix},$$
$$\begin{bmatrix}0&0&1\\0&0&0\\1&0&0\end{bmatrix}, \begin{bmatrix}0&0&0\\0&0&1\\0&1&0\end{bmatrix}. \tag{8}$$

Therefore, there is a nonsingular $K_{\max} \times K_{\max}$ matrix $\mathbf{U}$ such that $\mathbf{H}_l = \sum_{k=1}^{K_{\max}} u_{kl}\mathbf{X}_k, l = 1, 2, \ldots, K_{\max}$, where the matrices $\mathbf{H}_l$ represent the elements of the simple basis. That is, $\underline{\mathbf{H}}$ holds the elements of the simple basis as frontal symmetric slices. This simple form has weight (number of nonzero elements) I^2, which means that the proportion of nonzero elements is $1/K_{\max} = 2/(I(I+1))$. Clearly, the relative weight gets smaller as the size of the array increases.

This simple form for the $I \times I \times K_{\max}$ symmetric slice array is also useful when we are dealing with tall $I \times I \times K$ arrays, i.e., with $K > K_{\max}$. In this case the frontal slices are linearly dependent. We start by performing a suitable slice mix in order to set the slices in excess to zero, thus we may reduce the number of frontal slices. When $K_{\max}$ linearly independent slices remain, then the above simplification is possible.

5. Simplifying symmetric slice 2 × 2 × K arrays

When $\underline{\mathbf{X}}$ is a $2 \times 2 \times K$ array, there are three cases to consider, that is, we have $K = 1, 2$, or 3. The $2 \times 2 \times 3$ case was already dealt with before ($K_{\max} = 3$). In the $2 \times 2 \times 1$ case there is merely a symmetric matrix $\mathbf{X}$ of order 2. Computing the eigenvalue decomposition $\mathbf{X} = \mathbf{KDK}'$ we get $\mathbf{K}'\mathbf{XK} = \mathbf{D}$. Rescaling $\mathbf{D}$ to make one of its entries unity, we conclude that the diagonal matrix

$$\begin{bmatrix} 1 & 0 \\ 0 & \alpha \end{bmatrix} \tag{9}$$

is a weight 2 simple form for $\mathbf{X}$. An alternative simple form for $\mathbf{D}$ with the same weight is possible if its diagonal entries, say d_1 and d_2, have opposite signs, namely,

$$\mathbf{S}'\mathbf{DS} = \begin{bmatrix} 0 & 2d_1 \\ 2d_1 & 0 \end{bmatrix} \tag{10}$$

with $\mathbf{S} = \begin{bmatrix} 1 & 1 \\ k & -k \end{bmatrix}$ and $k^2 = -(d_1/d_2)$. So $\mathbf{X}$ can be transformed into $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ (after rescaling (10)) when the eigenvalues of $\mathbf{X}$ have opposite signs.

For a $2 \times 2 \times 2$ array $\underline{\mathbf{X}} = [\mathbf{X}_1 | \mathbf{X}_2]$, the simple form can be computed using the OCM, since $2 \times 2 \times 2$ is complementary to $2 \times 2 \times 1$. It is straightforward that a simple form which is complementary to (9) is given by

$$\left[\begin{array}{cc|cc} \alpha & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \end{array}\right]. \tag{11}$$

If the orthogonal complement of $\underline{\mathbf{X}}$ has eigenvalues with different signs (that is, if α in (9) is negative), then $\underline{\mathbf{X}}^{\mathrm{c}}$ can be simplified as in (10), entailing the complementary array

$$\left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]. \tag{12}$$

Any array derived from (12) via a Tucker transformation such that one of the slices is invertible, say $\mathbf{X}_1$, will have an $\mathbf{X}_1^{-1}\mathbf{X}_2$ with real eigenvalues only. Conversely, a $2 \times 2 \times 2$ symmetric array $\underline{\mathbf{X}} = [\mathbf{X}_1 | \mathbf{X}_2]$ such that $\mathbf{X}_1^{-1}\mathbf{X}_2$ has only real eigenvalues can always be simultaneously diagonalized as in (12), thus implying an orthogonal complement matrix with one eigenvalue positive and one negative.

To sum up, a weight 4 simple form for a $2 \times 2 \times 2$ symmetric array is almost surely possible. If $\underline{\mathbf{X}}^{\mathrm{c}}$ has both a positive and a negative eigenvalue, or equivalently, if $\mathbf{X}_1^{-1}\mathbf{X}_2$ has real eigenvalues, then a weight 2 simple target is possible.

6. Simplifying symmetric slice 3 × 3 × K arrays

For $3 \times 3 \times K$ arrays, $K_{\max} = 6$. We will search simplicity for cases $K = 1, 2, 4, 5$. Note that the case $K_{\max} = 6$ has already been solved. The case $K = 3$ remains an open issue and is not covered in the present paper.

The $3 \times 3 \times 1$ array (=matrix) can be diagonalized using its eigenvalue decomposition, say $\mathbf{D} = \mathrm{diag}(d_1, d_2, d_3)$. An alternative simple form with the same weight is possible if there is a pair of d_{i}'s with opposite signs, in the same manner as was done to deduce (10). So, if $d_2 d_3 < 0$ then

$$\mathbf{S}'\mathbf{DS} = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & 0 & 2d_2 \\ 0 & 2d_2 & 0 \end{bmatrix}, \tag{13}$$

with $\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & k & -k \end{bmatrix}$ and $k^2 = -(d_2/d_3)$.

The array format $3 \times 3 \times 5$ is complementary to $3 \times 3 \times 1$. We can therefore simplify a $3 \times 3 \times 5$ symmetric array $\underline{\mathbf{X}}$ using the OCM and the known simple form for the $3 \times 3 \times 1$ complementary array

$\underline{\mathbf{X}}^c$. Matrix $\underline{\mathbf{X}}^c$ can always be diagonalized, which in terms of the orthogonal complement means that the following weight 10 simple form is always possible:

$$\left[\begin{array}{ccc|ccc|ccc|ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right]. \tag{14}$$

When $\underline{\mathbf{X}}^c$ has a pair of eigenvalues with opposite signs, then it can be simplified into form (13), which leads to a weight 9 simple form,

$$\left[\begin{array}{ccc|ccc|ccc|ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right]. \tag{15}$$

To sum up, a $3 \times 3 \times 5$ symmetric array can almost surely be simplified into the weight 10 simple array (14). In the special case that the orthogonal complement of $\underline{\mathbf{X}}$ has eigenvalues of both signs, then the weight 9 target (15) is possible. This is the reason why (13) is to be preferred to the diagonal form for the $3 \times 3 \times 1$ case, whenever possible.

Next, we deal with the $3 \times 3 \times 2$ case $\underline{\mathbf{X}} = [\mathbf{X}_1|\mathbf{X}_2]$. Define $\mathbf{M} = \mathbf{X}_1^{-1}\mathbf{X}_2$. We divide our analysis in two cases, depending on the number of complex eigenvalues in $\mathbf{M}$.

• **M has all eigenvalues real**

In this case, array $\underline{\mathbf{X}}$ has rank 3 [16], and there is a CP-solution $\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{A}'$ $(k = 1, 2)$, where $\mathbf{A}$ is nonsingular and $\mathbf{C}_k$ is diagonal. So we can write $\mathbf{C}_k = \mathbf{A}^{-1}\mathbf{X}_k(\mathbf{A}^{-1})'$, which is a way of diagonalizing both slices of $\underline{\mathbf{X}}$. Note that $\mathbf{M} = \mathbf{X}_1^{-1}\mathbf{X}_2 = (\mathbf{A}')^{-1}(\mathbf{C}_1^{-1}\mathbf{C}_2)\mathbf{A}'$, which is an eigenvalue decomposition of $\mathbf{M}$. Therefore, an easy way to perform the double diagonalization is to compute the eigendecomposition $\mathbf{M} = \mathbf{K}\mathbf{L}\mathbf{K}^{-1}$, so $\mathbf{A} = (\mathbf{K}^{-1})'$, and compute $[\mathbf{K}'\mathbf{X}_1\mathbf{K}|\mathbf{K}'\mathbf{X}_2\mathbf{K}] = [\mathbf{C}_1|\mathbf{C}_2]$.

Almost surely, $\mathbf{C}_1^{-1}\mathbf{C}_2$ has a pair of distinct diagonal elements, say c_1 and c_2. Array $[\mathbf{C}_1|\mathbf{C}_2]$ may be transformed to $[\mathbf{C}_2 - c_1\mathbf{C}_1| - \mathbf{C}_2 + c_2\mathbf{C}_1]$, which has both slices diagonal of rank two. Notice that the slice mix that was performed consists of a nonsingular transformation in the third direction. So we conclude that a possible weight 4 simple form for the $3 \times 3 \times 2$ symmetric array, in the case when $\mathbf{M}$ has all three eigenvalues real, is

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & \beta & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right]. \tag{16}$$

When all eigenvalues are real, this simplification can be generalized to any array $\underline{\mathbf{X}} = [\mathbf{X}_1|\mathbf{X}_2]$ holding two symmetric slices of order $I \times I$. The obtained simple form will have weight $(2I - 2)$.

We shall now show how to get a simple form with weight 5. Although for the $3 \times 3 \times 2$ case this means a loss of simplicity, it will be useful for the complementary $3 \times 3 \times 4$ array. We start by considering array $[\mathbf{C}_1|\mathbf{C}_2 - c\mathbf{C}_1]$, where c is an eigenvalue of $\mathbf{C}_1^{-1}\mathbf{C}_2$. If all eigenvalues of $\mathbf{C}_1^{-1}\mathbf{C}_2$ are distinct (this happens almost surely), then for at least one of the eigenvalues, say c, we have that $\mathbf{C}_2 - c\mathbf{C}_1$ will be diagonal, of rank 2, holding two entries with opposite signs in the diagonal. Rescaling both frontal slices of $[\mathbf{C}_1|\mathbf{C}_2 - c\mathbf{C}_1]$ allows us to get the form

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & a & 0 & 0 & 1 & 0 \\ 0 & 0 & b & 0 & 0 & c \end{array}\right] \tag{17}$$

with $c < 0$. Entry c may be set to -1 by pre- and postmultiplying both slices by diag $\left(1, 1, \frac{1}{\sqrt{|c|}}\right)$. Subtract $(a - b)/2$ times the second slice from the first slice to get

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & d & 0 & 0 & 1 & 0 \\ 0 & 0 & d & 0 & 0 & -1 \end{array}\right] \tag{18}$$

with $d = (a + b)/2$. Next, pre- and postmultiplying both slices by $\begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{0.5} & \sqrt{0.5} \\ 0 & \sqrt{0.5} & -\sqrt{0.5} \end{bmatrix}$ leads to

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & d & 0 & 0 & 0 & 1 \\ 0 & 0 & d & 0 & 1 & 0 \end{array}\right]. \tag{19}$$

A final rescaling of the first slice allows us to conclude that the following simple form is always possible:

$$\left[\begin{array}{ccc|ccc} -\alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right] \tag{20}$$

with α nonzero. This form will be of use when deriving a simple form for the $3 \times 3 \times 4$ case.

It can be concluded that, for the $3 \times 3 \times 2$ array when $\mathbf{M}$ has all eigenvalues real, both the form (16) and the form (20) are almost surely possible. Form (20) will appear useful later, to derive simplicity for the complementary $3 \times 3 \times 4$ format.

• **M has one pair of complex eigenvalues**

Assume that in the eigenvalue decomposition $\mathbf{M} = \mathbf{KLK}^{-1}$ the real eigenvalue is $\mathbf{L}(1, 1)$. Consider $\mathbf{S}_1 = [\mathbf{k}_1 |\, \text{real}(\mathbf{k}_2)|\, \text{imag}(\mathbf{k}_2)]$, where $\mathbf{k}_i$ is the ith column of $\mathbf{K}$. Then [18]

$$[\mathbf{S}_1'\mathbf{X}_1\mathbf{S}_1|\mathbf{S}_1'\mathbf{X}_2\mathbf{S}_1] = \left[\begin{array}{ccc|ccc} a & 0 & 0 & d & 0 & 0 \\ 0 & b & c & 0 & e & f \\ 0 & c & -b & 0 & f & -e \end{array}\right]. \tag{21}$$

Subtracting (d/a) times the first slice from the second, and then rescaling the second slice by the inverse of entry (2,2) yields the form

$$[\mathbf{Y}_1|\mathbf{Y}_2] = \left[\begin{array}{ccc|ccc} a & 0 & 0 & 0 & 0 & 0 \\ 0 & b & c & 0 & 1 & f_1 \\ 0 & c & -b & 0 & f_1 & -1 \end{array}\right] \tag{22}$$

Define $\mathbf{S}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -s \\ 0 & s & 1 \end{bmatrix}$ for $s = f_1 + \sqrt{f_1^2 + 1}$, then

$$[\mathbf{S}_2'\mathbf{Y}_1\mathbf{S}_2|\mathbf{S}_2'\mathbf{Y}_2\mathbf{S}_2] = \left[\begin{array}{ccc|ccc} a_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & b_1 & c_1 & 0 & 0 & f_2 \\ 0 & c_1 & -b_1 & 0 & f_2 & 0 \end{array}\right]. \tag{23}$$

A final linear combination of both slices allows us to have $c_1 = 0$. Therefore, a simple form obtained after a final rescaling is

$$\left[\begin{array}{ccc|ccc} -\alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 0 \end{array}\right]. \tag{24}$$

The overall conclusion for a general $3 \times 3 \times 2$ symmetric slice array is that a weight 5 simple form is always possible, see (20) and (24). In the case when $\mathbf{X}_1^{-1}\mathbf{X}_2$ has all eigenvalues real, the weight 4 form (16) has the smallest possible weight.

We will now apply the OCM to simplify the $3 \times 3 {\times} 4$ array by using the known simple form of the $3 \times 3 \times 2$ array. It is easy to verify that both (20) and (24) admit as an orthogonal complement an array with form

$$\left[\begin{array}{ccc|ccc|ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \delta\alpha & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right], \tag{25}$$

where α has the same meaning as in (20) and (24), and $\delta = 1$ in the first case and -1 in the second case. So almost surely a $3 \times 3 \times 4$ symmetric slice array can be simplified to a weight 8 simple form. Interestingly, if we had used (16) as the simple array form for the orthogonal complement (in the real eigenvalue situation), we would have obtained a weight 9 array as simple form for $\underline{\mathbf{X}}$, less simple than (25).

There is still one case left, the $3 \times 3 \times 3$ symmetric slice array. Although this array format seems to allow a simple pattern of weight 9 more often than not, a formal proof of this has evaded us.

7. Simplifying symmetric slice 4 × 4 × K arrays

In the $4 \times 4 \times K$ case we have $K_{\max} = 10$. So the $4 \times 4 \times 10$ case has been solved. We present simple forms for formats $4 \times 4 \times 1$ and $4 \times 4 \times 2$, as well as their complementary formats $4 \times 4 \times 9$ and $4 \times 4 \times 8$, respectively. The remaining cases are still open.

Arrays (=matrices) of order $4 \times 4 \times 1$ can be diagonalized by means of the eigenvalue decomposition. This means that $\mathbf{D} = \mathrm{diag}(d_1, d_2, d_3, d_4)$, the diagonal matrix holding eigenvalues, is always a possible simple form for the array. Alternative forms, specially useful for the $4 \times 4 \times 9$ complement, may be possible using the same process that was used to obtain (10) for the $2 \times 2 \times 1$ case. Specifically, we may attain the forms

$$\begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & 0 & 2d_3 \\ 0 & 0 & 2d_3 & 0 \end{bmatrix}, \tag{26}$$

when $d_1, d_2, d_3 > 0$ and $d_4 < 0$, and

$$\begin{bmatrix} 0 & 2d_1 & 0 & 0 \\ 2d_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2d_3 \\ 0 & 0 & 2d_3 & 0 \end{bmatrix}, \tag{27}$$

when $d_1, d_3 > 0$ and $d_2, d_4 < 0$. Using the OCM for the $4 \times 4 \times 9$ array and taking advantage of the simple forms found for the $4 \times 4 \times 1$ orthogonal complement $\mathbf{D}$, (26) and (27), respectively, we have various simple forms for $\underline{\mathbf{X}}$. When the complement is $\mathbf{D}$, we have

$$\begin{aligned} &\left[\begin{array}{cccc|cccc|cccc|cccc|cccc} 1&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0 \\ 0&\alpha&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&\beta&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&\gamma&0&0&0&0&0&0&0&0 \end{array}\right. \\ &\qquad \left.\begin{array}{cccc|cccc|cccc|cccc} 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0 \end{array}\right] \end{aligned} \tag{28}$$

of weight 18. When the complement is (26), we have

$$\begin{aligned} &\left[\begin{array}{cccc|cccc|cccc|cccc|cccc} 1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0 \\ 0&\alpha&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \end{array}\right. \\ &\qquad \left.\begin{array}{cccc|cccc|cccc|cccc} 0&0&0&1&0&0&0&0&0&0&0&0&\beta&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0 \end{array}\right] \end{aligned} \tag{29}$$

of weight 17. When the complement is (27), we have

$$\begin{aligned} &\left[\begin{array}{cccc|cccc|cccc|cccc|cccc} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&\alpha \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&\alpha&0 \end{array}\right. \\ &\qquad \left.\begin{array}{cccc|cccc|cccc|cccc} 0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0 \end{array}\right] \end{aligned} \tag{30}$$

of weight 16. The conclusion is that, in general, a simplification of a $4 \times 4 \times 9$ symmetric slice array into a weight 18 simple array (28) is always possible. This result can be improved, when the signs of the eigenvalues of the orthogonal complement of $\underline{\mathbf{X}}$ permit, to a weight 17 or weight 16 array (29) or (30), respectively.

Next, we treat the $4 \times 4 \times 2$ case $\underline{\mathbf{X}} = [\mathbf{X}_1|\mathbf{X}_2]$. Defining $\mathbf{M} = \mathbf{X}_1^{-1}\mathbf{X}_2$, three possibilities arise: either all eigenvalues of $\mathbf{M}$ are real, or $\mathbf{M}$ has one pair of complex eigenvalues, or $\mathbf{M}$ has two pairs of complex eigenvalues.

• **M has all eigenvalues real**

When all eigenvalues are real, $\underline{\mathbf{X}}$ has rank 4 [16]. As argued in the case of $3 \times 3 \times 2$ arrays when all eigenvalues are real, it is possible to perform a double diagonalization, followed by reducing the rank of each slice to be 3, so we can have the weight 6 array

$$\left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & \gamma & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \beta & 0 & 0 & 0 & \delta & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]. \tag{31}$$

Another simple form that can be achieved using a procedure similar to the one that led to (20) is given by

$$\left[\begin{array}{cccc|cccc} \alpha & 0 & 0 & 0 & \gamma & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}\right]. \tag{32}$$

Although (32) has weight 7, it will be useful in terms of the $4 \times 4 \times 8$ complement.

• **M has one pair of complex eigenvalues**

Let, in the eigenvalue decomposition $\mathbf{M} = \mathbf{KLK}^{-1}$, the real eigenvalues be $\mathbf{L}(1, 1)$ and $\mathbf{L}(2, 2)$. Consider matrix $\mathbf{S} = [\mathbf{k}_1|\mathbf{k}_2|\ \text{real}(\mathbf{k}_3)|\text{imag}(\mathbf{k}_3)]$, where $\mathbf{k}_i$ is the ith column of $\mathbf{K}$. Then [18] we have

$$[\mathbf{S}'\mathbf{X}_1\mathbf{S}|\mathbf{S}'\mathbf{X}_2\mathbf{S}] = \left[\begin{array}{cccc|cccc} * & 0 & 0 & 0 & * & 0 & 0 & 0 \\ 0 & a & 0 & 0 & 0 & d & 0 & 0 \\ 0 & 0 & b & c & 0 & 0 & e & f \\ 0 & 0 & c & -b & 0 & 0 & f & -e \end{array}\right], \tag{33}$$

where $*$ denotes a nonzero entry. The pair of lower-right 3×3 submatrices have an already known form, see (21). Therefore, using here the same procedure used to simplify (21) applied to these blocks, we obtain a weight 7 symmetric simple form

$$\left[\begin{array}{cccc|cccc} \alpha & 0 & 0 & 0 & \gamma & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \end{array}\right], \tag{34}$$

similar to (32). In fact, both arrays lead to $4 \times 4 \times 8$ complements with the same pattern of (non)zeros.

• **M has two pairs of complex eigenvalues**

Rocci and Ten Berge [12] explain how to perform a slice mix of $\mathbf{X}_1$ and $\mathbf{X}_2$ such that for the new slices $\widetilde{\mathbf{X}}_1, \widetilde{\mathbf{X}}_2$ we have that $\widetilde{\mathbf{X}}_1^{-1}\widetilde{\mathbf{X}}_2$ has pure imaginary eigenvalues, so we will assume that $\underline{\mathbf{X}}$ is already in such form. Write the eigenvalue decomposition of $\mathbf{M} = \mathbf{KLK}^{-1}$ with $\mathbf{L}$ holding conjugate eigenvalues placed next to each other, and consider $\mathbf{S} = [\text{real}(\mathbf{k}_1)|\text{imag}(\mathbf{k}_1)|\text{real}(\mathbf{k}_3)|\text{imag}(\mathbf{k}_3)]$, where $\mathbf{k}_i$ is the ith column of $\mathbf{K}$. Then [18]

$$[\mathbf{S}'\mathbf{X}_1\mathbf{S}|\mathbf{S}'\mathbf{X}_2\mathbf{S}] = \left[\begin{array}{cccc|cccc} a & b & 0 & 0 & e & f & 0 & 0 \\ b & -a & 0 & 0 & f & -e & 0 & 0 \\ 0 & 0 & c & d & 0 & 0 & g & h \\ 0 & 0 & d & -c & 0 & 0 & h & -g \end{array}\right]. \tag{35}$$

Denote these slices by $\mathbf{Z}_1$ and $\mathbf{Z}_2$. It is clear that $\mathbf{Z}_1^{-1}\mathbf{Z}_2$ has the same eigenvalues as $\mathbf{M}$.

The next step consists of obtaining the eigenvalue decomposition $\mathbf{Z}_1 = \mathbf{K}_1\mathbf{L}_1\mathbf{K}_1'$. Notice that $\mathbf{Z}_1$ has two pairs of real eigenvalues differing only in signs. We shall prove next that $[\mathbf{K}_1'\mathbf{Z}_1\mathbf{K}_1|\mathbf{K}_1'\mathbf{Z}_2\mathbf{K}_1]$ has the symmetric form

$$\left[\begin{array}{cccc|cccc} * & 0 & 0 & 0 & 0 & * & 0 & 0 \\ 0 & * & 0 & 0 & * & 0 & 0 & 0 \\ 0 & 0 & * & 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & * & 0 & 0 & * & 0 \end{array}\right] \tag{36}$$

if $\mathbf{L}_1$ holds the opposite eigenvalues placed next to each other, e.g., $\lambda_1, -\lambda_1, \lambda_2, -\lambda_2$. This form can be seen as related to (32) and (34), from the previous cases. We notice that the form of the first slice is obvious (it is $\mathbf{L}_1$), but the form of the second slice deserves further inspection.

We start by observing that the 2×2 diagonal blocks in $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are proportional to orthonormal matrices. Therefore, we can write

$$\mathbf{Z}_1 = \begin{bmatrix} \gamma_1\mathbf{T}_1 & \mathbf{0} \\ \mathbf{0} & \gamma_2\mathbf{T}_2 \end{bmatrix}, \quad \mathbf{Z}_2 = \begin{bmatrix} \gamma_3\mathbf{T}_3 & \mathbf{0} \\ \mathbf{0} & \gamma_4\mathbf{T}_4 \end{bmatrix} \tag{37}$$

with $\mathbf{T}_1, \ldots, \mathbf{T}_4$ symmetric and orthonormal, with eigenvalues 1 and -1. Then

$$\mathbf{Z}_1^{-1}\mathbf{Z}_2 = \begin{bmatrix} \gamma_1^{-1}\gamma_3\mathbf{T}_1\mathbf{T}_3 & \mathbf{0} \\ \mathbf{0} & \gamma_2^{-1}\gamma_4\mathbf{T}_2\mathbf{T}_4 \end{bmatrix}. \tag{38}$$

First, consider the upper-left 2×2 block. Define $\mathbf{G} = \mathbf{T}_1\mathbf{T}_3$ (so $\mathbf{T}_3 = \mathbf{T}_1\mathbf{G}$); $\mathbf{G}$ is orthonormal with pure imaginary eigenvalues. Consider its eigenvalue decomposition $\mathbf{G} = \mathbf{K}_G\mathbf{L}_G\mathbf{K}_G^{-1}$ with $\mathbf{L}_G = \operatorname{diag}(-iu, iu)$, then $\mathbf{G}^2 = -u^2\mathbf{I}_2$. This implies that $\det(\mathbf{G}^2) = u^4$, but we also have that $\det(\mathbf{G}^2) = \det(\mathbf{G}'\mathbf{G}) = \det(\mathbf{I}_2) = 1$, and so $u = \pm 1$. Hence $\mathbf{G}^2 = -\mathbf{I}_2$. From this equality and the orthonormality of $\mathbf{G}$ we get that $\mathbf{G} = -\mathbf{G}'$, so $\mathbf{G}$ is a skew matrix. Combining this fact with the orthonormality of $\mathbf{G}$ allows to conclude that $\mathbf{G} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$, up to sign. Next, since $\mathbf{T}_1$ has eigenvalues 1 and -1, we can write the eigenvalue decomposition $\mathbf{T}_1 = \mathbf{K}_{\mathbf{T}_1}\mathbf{L}_{\mathbf{T}_1}\mathbf{K'}_{\mathbf{T}_1}$ with $\mathbf{L}_{\mathbf{T}_1} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ and $\mathbf{K}_{\mathbf{T}_1}$ orthonormal. From this we have $\mathbf{K}_{\mathbf{T}_1}'\mathbf{T}_1\mathbf{K}_{\mathbf{T}_1} = \mathbf{L}_{\mathbf{T}_1}$ (diagonal) and

$$\begin{aligned} \mathbf{K}_{\mathbf{T}_1}'\mathbf{T}_3\mathbf{K}_{\mathbf{T}_1} &= \mathbf{K}_{\mathbf{T}_1}'\mathbf{T}_1\mathbf{G}\mathbf{K}_{\mathbf{T}_1} = \mathbf{K}_{\mathbf{T}_1}'\mathbf{K}_{\mathbf{T}_1}\mathbf{L}_{\mathbf{T}_1}\mathbf{K}_{\mathbf{T}_1}'\mathbf{G}\mathbf{K}_{\mathbf{T}_1} \\ &= \mathbf{L}_{\mathbf{T}_1}\mathbf{K'}_{\mathbf{T}_1}\mathbf{G}\mathbf{K}_{\mathbf{T}_1} = \pm\mathbf{L}_{\mathbf{T}_1}\mathbf{G} = \pm\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \end{aligned}$$

(the second last equality is because $\mathbf{K}_{\mathbf{T}_1}'\mathbf{G}\mathbf{K}_{\mathbf{T}_1}$ is orthonormal and skew, and therefore it equals $\mathbf{G}$ up to sign). The same process can be applied to the lower-right 2×2 blocks of $\mathbf{Z}_1$ and $\mathbf{Z}_2$. We can conclude that if we pre and postmultiply the slices of the array $[\mathbf{Z}_1|\mathbf{Z}_2]$ by $\mathbf{K}_1'$ and $\mathbf{K}_1$, respectively, where $\mathbf{K}_1 = \begin{bmatrix} \mathbf{K}_{\mathbf{T}_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{\mathbf{T}_2} \end{bmatrix}$ is the matrix of eigenvectors of $\mathbf{Z}_1$, then

$$[\mathbf{K}_1'\mathbf{Z}_1\mathbf{K}_1|\mathbf{K}_1'\mathbf{Z}_2\mathbf{K}_1] = \left[\begin{array}{cccc|cccc} \gamma_1 & 0 & 0 & 0 & 0 & \delta_1\gamma_3 & 0 & 0 \\ 0 & -\gamma_1 & 0 & 0 & \delta_1\gamma_3 & 0 & 0 & 0 \\ 0 & 0 & \gamma_2 & 0 & 0 & 0 & 0 & \delta_2\gamma_4 \\ 0 & 0 & 0 & -\gamma_2 & 0 & 0 & \delta_2\gamma_4 & 0 \end{array}\right] \tag{39}$$

with $\delta_1 = \pm 1, \delta_2 = \pm 1$.

We can use the simple forms that were deduced for symmetric $4 \times 4 \times 2$ arrays to compute simple forms for the $4 \times 4 \times 8$ case using the OCM. So, depending on which situation we have for $\underline{\mathbf{X}}^c$, we may have $\underline{\mathbf{H}}^c$ as in (32), (34) or (39). The first two cases lead to a complement of weight 18 with the following slices:

$$\left[\begin{array}{cccc|cccc|cccc|cccc|} -2\beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2\alpha & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \beta\gamma & 0 & 0 & -\beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \beta\gamma & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta\delta & 0 & 0 & 0 & 0 \end{array}\right.$$

$$\left.\begin{array}{cccc|cccc|cccc|cccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right], \tag{40}$$

where α, β, and γ have the same meaning as in (32) and (34), and $\delta = 1$ in the first case and -1 in the second case. The third case leads to a complement of the form

$$\left[\begin{array}{cccc|cccc|cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\gamma_1\gamma_2^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_1\gamma_2^{-1} & 0 & 0 & \alpha & 0 \end{array}\right.$$

$$\left.\begin{array}{cccc|cccc|cccc|cccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right], \tag{41}$$

with $\alpha = -\delta_1\delta_2^{-1}\gamma_3\gamma_4^{-1}$, again of weight 18. The overall conclusion is that a symmetric $4 \times 4 \times 8$ array can almost surely be simplified into one out of two weight 18 arrays.

8. Applications to typical rank

Results concerning typical rank for several symmetric slice arrays are presented in Ten Berge et al. [16]. Using the simplicity results presented above, we can now further clarify some of the results deduced in [16]. We shall do this by revisiting the typical rank issue (over the field of real numbers) for $3 \times 3 \times 4$ and the $3 \times 3 \times 5$ array formats, when slices are symmetric.

• **Symmetric slice** $3 \times 3 \times 4$ **arrays**

We proved that a possible simple form for $3 \times 3 \times 4$ symmetric slice arrays that works almost always is given by (25). Ten Berge et al. [16] have proven that $3 \times 3 \times 4$ symmetric slice arrays have typical rank $\{4, 5\}$. The proof consists of constructing a rank four solution for a randomly sampled symmetric slice array, and determining under which conditions we need a rank five solution. Here we shall do the same, this time using (25). Our purpose is to put in evidence how helpful simple forms can be to study the rank of an array.

We start by unfolding the array and vectorizing its frontal slices, so we get $\mathbf{X}_{\text{Vec}} = [\text{Vec}(\mathbf{X}_1)|\cdots|\text{Vec}(\mathbf{X}_4)]$. Noting that a CP decomposition can be written in the form $\mathbf{X}_{\text{Vec}} = (\mathbf{A} \bullet \mathbf{B})\mathbf{C}'$, where $\bullet$ stands for the Khatri-Rao product, it can be concluded that a rank 4 solution exists if and only if there exists a Khatri–Rao basis $\mathbf{A} \bullet \mathbf{B}$ which generates $\mathbf{X}_{\text{Vec}}$. Equivalently, we may solve $\mathbf{X}_{\text{Vec}}\mathbf{W} = \mathbf{A} \bullet \mathbf{B}$, with $\mathbf{W} = (\mathbf{C}')^{-1}$. The problem sums up to finding four linearly independent vectors $\mathbf{w}$ (columns of $\mathbf{W}$) such that $\mathbf{X}_{\text{Vec}}\mathbf{w}$ is the Kronecker product of two vectors (columns of $\mathbf{A}$ and $\mathbf{B}$), which may be rescaled to be $\mathbf{a} = [1\ a_1\ a_2]'$ and $\mathbf{b} = [1\ b_1\ b_2]'$, respectively, for scalars a_1, a_2, b_1 and b_2. We have

$$\mathbf{X}_{\text{Vec}} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & \delta\alpha & 0 & 0 \end{bmatrix} \quad \textit{and} \quad \mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} 1 \\ b_1 \\ b_2 \\ a_1 \\ a_1b_1 \\ a_1b_2 \\ a_2 \\ a_2b_1 \\ a_2b_2 \end{bmatrix}. \tag{42}$$

Solving $\mathbf{X}_{\text{Vec}}\mathbf{w} = \mathbf{a} \otimes \mathbf{b}$ for $\mathbf{w}$ yields $\mathbf{a} = \mathbf{b}$ and $\mathbf{w} = [\alpha^{-1}\ b_1^2\ \delta\alpha^{-1}\ b_2^2\ b_1\ b_2]'$. There are two equations left, $b_1b_2 = 0$ and $b_1^2 + \delta b_2^2 = \alpha$. The last two equations have four solutions if and only if $\delta = 1$ and α is positive, being the solutions $b_1 = 0, b_2 = \pm\sqrt{\alpha}$ and $b_1 = \pm\sqrt{\alpha}, b_2 = 0$. With this we already have

$$\mathbf{A}=\mathbf{B}=\begin{bmatrix}1 & 1 & 1 & 1\\ 0 & 0 & \sqrt{\alpha} & -\sqrt{\alpha}\\ \sqrt{\alpha} & -\sqrt{\alpha} & 0 & 0\end{bmatrix}. \tag{43}$$

We find $\mathbf{C}$ as $\mathbf{C}=(\mathbf{W}^{-1})'$, which is given by

$$\mathbf{C}=\begin{bmatrix}0 & 0 & 0.5 & 0.5\\ 0.5 & 0.5 & 0 & 0\\ 0 & 0 & 0.5\sqrt{\alpha^{-1}} & -0.5\sqrt{\alpha^{-1}}\\ 0.5\sqrt{\alpha^{-1}} & -0.5\sqrt{\alpha^{-1}} & 0 & 0\end{bmatrix}. \tag{44}$$

We conclude that, when $\delta=1$ and $\alpha>0$, a unique rank four decomposition of $\underline{\mathbf{X}}$ is given by $\mathbf{X}_k=\mathbf{AC}_k\mathbf{A}', k=1,\ldots,4$, with $\mathbf{A}$ and $\mathbf{C}$ given by (43) and (44).

When $\delta=-1$ or $\alpha<0$, the rank of $\underline{\mathbf{X}}$ is larger than 4. A rank 5 solution can be constructed as follows. Consider array $\underline{\mathbf{X}}$ temporarily augmented with a fifth slice $\mathbf{X}_5=\text{diag}(-\delta\alpha^2,\delta\alpha,\alpha)$, and define $\mathbf{X}_{\text{aug}}=[\text{Vec}(\mathbf{X}_1)|\ldots|\text{Vec}(\mathbf{X}_5)]$. Proceeding as before, it is possible to find $\mathbf{A}(3\times 5)$ and $\mathbf{C}_{\text{aug}}(5\times 5)$ such that $\mathbf{X}_{\text{aug}}=(\mathbf{A}\bullet\mathbf{A})\mathbf{C}'_{\text{aug}}$. We find $\mathbf{C}$ by eliminating the fifth row of $\mathbf{C}_{\text{aug}}$. There are many solutions possible. If we settle for

$$\mathbf{A}=\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 0 & 0 & 1 & -1 & 0\\ 1 & -1 & 0 & 0 & 0\end{bmatrix}, \tag{45}$$

then $\mathbf{C}$ will be

$$\mathbf{C}=\begin{bmatrix}0 & 0 & 0.5\alpha & 0.5\alpha & 1-\alpha\\ 0.5\delta\alpha & 0.5\delta\alpha & 0 & 0 & 1-\delta\alpha\\ 0 & 0 & 0.5 & -0.5 & 0\\ 0.5 & -0.5 & 0 & 0 & 0\end{bmatrix}. \tag{46}$$

This is a closed form solution for $3\times 3\times 4$ symmetric slice arrays of rank 5.

The derivation above, based on the simplicity pattern (25), has enabled us to greatly simplify the rank analysis of $3\times 3\times 4$ symmetric slice arrays, compared to [16]. All it takes to determine the rank of such an array is seeing whether or not α and δ of (25) are positive or not. This is easier than evaluating the roots of a certain fourth degree polynomial to see if they are real and distinct, [16]. In fact, that fourth degree polynomial has now been reduced to $(\lambda^2-\alpha)(\lambda^2-\delta\alpha)$. In addition, finding a rank 5 solution when a rank four solution fails is now also trivially easy, as is clear from (45) and (46).

• **Symmetric slice** $3\times 3\times 5$ **arrays**

In [16] it is proven that $3\times 3\times 5$ symmetric slice arrays have typical rank {5,6}. We proved that a possible simple form is given by (14). We can proceed as done before for the $3\times 3\times 4$ situation. Construct $\mathbf{X}_{\text{Vec}}=[\text{Vec}(\mathbf{X}_1)|\cdots|\text{Vec}(\mathbf{X}_5)]$. For a rank 5 solution to exist we need to find a Khatri–Rao basis which generates $\mathbf{X}_{\text{Vec}}=(\mathbf{A}\bullet\mathbf{B})\mathbf{C}'$. This is the same as solving $\mathbf{X}_{\text{Vec}}\mathbf{W}=\mathbf{A}\bullet\mathbf{B}$, with $\mathbf{W}=(\mathbf{C}')^{-1}$. Denoting rescaled columns of $\mathbf{A}$ and $\mathbf{B}$ by $\mathbf{a}=[1\ a_1\ a_2]'$ and $\mathbf{b}=[1\ b_1\ b_2]'$ respectively, for scalars a_1,a_2,b_1 and b_2, we get

$$\mathbf{X}_{\text{Vec}}=\begin{bmatrix}1 & 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 1 & 0 & 0\\ \alpha & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 1\\ 0 & \beta & 0 & 0 & 0\end{bmatrix} \quad\text{and}\quad \mathbf{a}\otimes\mathbf{b}=\begin{bmatrix}1\\ b_1\\ b_2\\ a_1\\ a_1b_1\\ a_1b_2\\ a_2\\ a_2b_1\\ a_2b_2\end{bmatrix}. \tag{47}$$

Solving $\mathbf{X}_{\text{Vec}}\mathbf{w}=\mathbf{a}\otimes\mathbf{b}$ for $\mathbf{w}$ implies that $\mathbf{a}=\mathbf{b}$ and $\mathbf{w}=[\alpha^{-1}b_1^2\ \beta^{-1}b_2^2\ b_1\ b_2\ b_1b_2]'$. There is one condition that remains to be solved, which is equation $\alpha^{-1}b_1^2+\beta^{-1}b_2^2=1$. This equation implies that there is a solution (and therefore a rank 5 decomposition) if and only if $\alpha>0$ and/or $\beta>0$. When the latter condition is satisfied, we can deduce closed form solutions. There is an infinite number of solutions, of which the ones presented next are only a possibility:

- if $\alpha > 0$ and $\beta > 0$:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & \sqrt{\alpha} & -\sqrt{\alpha} & 0.5\sqrt{\alpha} \\ \sqrt{\beta} & -\sqrt{\beta} & 0 & 0 & \sqrt{0.75\beta} \end{bmatrix}, \tag{48}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0.5\sqrt{\alpha^{-1}} & -0.5\sqrt{\alpha^{-1}} & 0 \\ 0.5\sqrt{\beta^{-1}} & -0.5\sqrt{\beta^{-1}} & 0 & 0 & 0 \\ -2(\sqrt{0.75}+0.75)k & 2(\sqrt{0.75}-0.75)k & -1.5k & 0.5k & 4k \end{bmatrix}, \tag{49}$$

  with $k = \sqrt{3^{-1}\alpha^{-1}\beta^{-1}}$.
- if $\alpha > 0$ and $\beta < 0$:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ \sqrt{\alpha} & -\sqrt{\alpha} & \sqrt{2\alpha} & \sqrt{2\alpha} & -\sqrt{2\alpha} \\ 0 & 0 & \sqrt{-\beta} & -\sqrt{-\beta} & \sqrt{-\beta} \end{bmatrix} \tag{50}$$

  and

$$\mathbf{C} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 \\ 1+\sqrt{0.5} & 1-\sqrt{0.5} & -0.5 & -0.5 & 0 \\ 0.5\sqrt{\alpha^{-1}} & -0.5\sqrt{\alpha^{-1}} & 0 & 0 & 0 \\ \sqrt{-0.5\beta^{-1}} & -\sqrt{-0.5\beta^{-1}} & 0 & -0.5\sqrt{-\beta^{-1}} & 0.5\sqrt{-\beta^{-1}} \\ -0.5\sqrt{-\alpha^{-1}\beta^{-1}} & 0.5\sqrt{-\alpha^{-1}\beta^{-1}} & 0.5\sqrt{-0.5\alpha^{-1}\beta^{-1}} & 0 & -0.5\sqrt{-0.5\alpha^{-1}\beta^{-1}} \end{bmatrix}.$$

When the rank is 6, that is, when $\alpha < 0$ and $\beta < 0$, a simple decomposition based on [11] is the following:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & \alpha & 0 & 0 & 0 & 0 \\ 1 & 0 & \beta & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & -1 & 1 & 0 & 0 \end{bmatrix}. \tag{51}$$

9. Maximal simplicity

We have presented simple forms for some symmetric slice arrays. A natural question that arises concerns the optimality of the simple targets, described in terms of minimal weight. That is, have the simple forms we presented the maximal simplicity possible?

For some cases it is possible to prove at once that the simple forms presented have minimal weight. For instance, the weight 9 form (8) for the $3 \times 3 \times 6$ symmetric slice array has maximal simplicity. In fact, if a simple form $\underline{\mathbf{H}}$ with weight less than 9 were possible, then it would have at least four slices of weight 1. The symmetry of these four slices implies that the weights must be placed in the diagonal of each slice, which leads to the conclusion that $\underline{\mathbf{H}}$ has linearly dependent slices. So weight 9 is associated to the maximal simplicity possible for $3 \times 3 \times 6$ symmetric slice arrays with linearly independent slices, or more generally, weight I^2 is the optimal weight for $I \times I \times K_{\max}$ symmetric slice arrays with linearly independent slices.

The weight of an array is an upper bound to the rank of an array. Usually this bound is larger than the rank, but in cases of low order it might give some insight regarding maximal simplicity. For example, it was shown that the $2 \times 2 \times 2$ symmetric slice array $\underline{\mathbf{X}}$ can be transformed into the simple form with weight 4 given by (11), and when $\alpha < 0$ the weight 2 simple target given by (12) is also possible. It is well known that $\underline{\mathbf{X}}$ has rank 2 when $\alpha < 0$ and rank 3 when $\alpha > 0$. This immediately implies that (12) has maximal simplicity when $\alpha < 0$, since in this situation the weight equals the rank. On the other hand, it is easy to show that there is no slice mix of (11) with rank 1 when $\alpha > 0$, which implies that no Tucker transformations applied to (11) can lead to slices of rank 1. This means that weight 4 is the minimal possible in this case.

The OCM can also be used in this context. Consider, for instance, the $3 \times 3 \times 5$ symmetric slice array. We showed that a randomly generated array of this order can always be simplified into a weight 10 simple array (14), and in some situations a weight 9 array is also possible, see (15). The question to answer at the moment is: is it possible to have a simpler (smaller weight) target? Assuming that it is possible, we shall consider all simpler targets available. Noting that weight 6 or less would imply linearly dependent slices, we are left with the following two types of arrays (without loss of generality):

$$\underline{\mathbf{H}}_1 = \left[\begin{array}{ccc|ccc|ccc|ccc|ccc} 0&0&0&0&0&0&0&1&0&0&0&1&0&0&0\\ 0&1&0&0&0&0&1&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0&0&0&1&0&0&0&1&0 \end{array}\right] \tag{52}$$

and

$$\underline{\mathbf{H}}_2 = \left[\begin{array}{ccc|ccc|ccc|ccc|ccc} 1&0&0&0&0&0&0&0&0&0&1&0&0&0&1\\ 0&0&0&0&1&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0&1&0&0 \end{array}\right]. \tag{53}$$

Arrays (52) and (53) admit the following orthogonal complements, respectively,

$$\underline{\mathbf{H}}_1^{\mathbf{c}} = \begin{bmatrix} 1&0&0\\ 0&0&0\\ 0&0&0 \end{bmatrix}, \quad \underline{\mathbf{H}}_2^{\mathbf{c}} = \begin{bmatrix} 0&0&0\\ 0&0&1\\ 0&1&0 \end{bmatrix}. \tag{54}$$

When an orthogonal complement set contains a single symmetric matrix that is rescaled to unit sums of squares, and rescaled to have a certain nonzero element positive (viz. the element (1,1) in $\underline{\mathbf{H}}_1^{\mathbf{c}}$, and the element (2,3) in $\sqrt{.5}\underline{\mathbf{H}}_2^{\mathbf{c}}$), that complement matrix depends continuously on the given array. Moreover, the determinant of a square matrix of order n is a real valued analytic function. Since the determinant of orthogonal complement matrices of $3 \times 3 \times 5$ symmetric slice arrays is not identically zero, we can conclude that such matrices have determinant nonzero almost surely [2]. This implies that both complements in (54) arise with probability zero, and so the simple forms (52) and (53) may be discarded. It can be concluded that $3 \times 3 \times 5$ arrays admit simple forms with weight less than 9 with probability zero.

It was proved in this paper that for $3 \times 3 \times 2$ symmetric slice arrays $\underline{\mathbf{X}} = [\mathbf{X}_1|\mathbf{X}_2]$ a weight 5 simple form is always possible (arrays (20) or (24)), and in some situations weight 4 is possible (array (16)). A simple form with weight 3 is not possible, because it can be seen that no slice mix from (20) or (24) will ever lead to a rank 1 matrix. Therefore, the maximal simplicity for $3 \times 3 \times 2$ symmetric slice arrays is weight 4. Furthermore, when $\mathbf{X}_1^{-1}\mathbf{X}_2$ has complex eigenvalues it is not possible to improve the weight 5 simple form (24), since it can be seen that any weight 4 symmetric slice array has real generalized eigenvalues.

Finally, consider the $3 \times 3 \times 4$ symmetric slice arrays. We were able to simplify these arrays into (25), which has weight 8. For this result to have maximal simplicity, we need to ensure that weight 7 or less can not occur with positive probability. It can be seen that any simple form of weight 6 or less with at least three slices of rank 1 has a $3 \times 3 \times 2$ orthogonal complement spanned by two slices that do not admit a linear combination of rank 3, which is an event of probability 0. Also, simple forms with weight 6 and only two rank 1 slices happen with probability zero, since their orthogonal complement spaces are spanned by pairs of slices with joint weight 3. So we need to only focus on weight 7 targets. There are two possible types of targets:

- We can have exactly two slices of rank 1, such as in array $\underline{\mathbf{H}}_1$:

$$\underline{\mathbf{H}}_1 = \left[\begin{array}{ccc|ccc|ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 1 & 0 & 0 \end{array}\right]. \tag{55}$$

There are 18 different possibilities in total, but the reasoning to be presented to only this example applies to all. Notice that $\underline{\mathbf{H}}_1$ is impossible almost surely, because its orthogonal complement $\underline{\mathbf{H}}_1^{\mathbf{c}}$,

$$\underline{\mathbf{H}}_1^{\mathbf{c}} = \left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & -\alpha & 0 \\ 0 & 0 & 1 & -\alpha & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 2 \end{array}\right] \tag{56}$$

is a $3 \times 3 \times 2$ symmetric slice array such that any slice mix will lead to three repeated real generalized eigenvalues, an event of probability zero. In some of the cases the orthogonal complement cannot even lead to full rank matrices by linear combination of the slices, which is also an event of probability zero.
- We can have exactly one slice of rank 1. One possibility, given by

$$\underline{\mathbf{H}}_2 = \left[\begin{array}{ccc|ccc|ccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \tag{57}$$

can be ruled out since its complement $\underline{\mathbf{H}}_2^{\mathbf{c}}$,

$$\underline{\mathbf{H}}_2^{\mathbf{c}} = \left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{58}$$

is a $3 \times 3 \times 2$ symmetric slice array with only weight 2. There are nine possibilities left. Six of them, such as $\underline{\mathbf{H}}_3$,

$$\underline{\mathbf{H}}_3 = \left[\begin{array}{ccc|ccc|ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & a & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \tag{59}$$

lead to an orthogonal complement such that any slice mix with at least one invertible slice will lead to a pair of repeated real generalized eigenvalues, which is an event of probability zero. The remaining three possibilities have orthogonal complements that again do not admit full rank matrices as linear combinations of their slices, which is an event of probability zero.

The conclusion is that weight 8 is indeed the maximal simplicity almost surely for $3 \times 3 \times 4$ symmetric slice arrays.

10. Results from simulations: the SIMPLIMAX procedure

For many situations it is not easy to find simple forms, or assess maximal simplicity. When formal proofs might be difficult, one can still use simulation as an informal way of making, reinforcing or refuting one's hypotheses. Kiers [6] has developed a procedure called SIMPLIMAX, which finds oblique rotations that give the minimum sum of squares for a previously specified number m of entries for the rotated array. It is not known *a priori* which entries will be the smallest ones, so the algorithm will internally solve this issue. This has the side effect of SIMPLIMAX finding locally optimal solutions. This problem can be circumvented by using a large number of randomly started runs of the algorithm. An adapted version of SIMPLIMAX allows to fix, in advance, the position of the m entries whose sum of squares we intend to minimize. We shall refer to this as the fixed version of SIMPLIMAX, in contrast with the not fixed version of the algorithm.

SIMPLIMAX had a crucial role in the research that led to the present paper. The *modus operandi* was usually the following: for an array order for which we were interested in finding simple forms, we randomly generated a family of 30 such arrays, fixed some values for m, and ran the not fixed algorithm using MATLAB. A rotation was considered "successful" when the sum of squares of the m smallest elements was below a fixed threshold (we settled for 10^{-15}). For each array, we repeated the algorithm with a random different starting configuration (to avoid locally optimal solutions) to a maximum of 150 tries, unless a successful solution was found meanwhile. When we had found an interesting simple form to look at, we used fixed SIMPLIMAX to test it directly (again 30 randomly sampled arrays, 150 tries for each array). This gave us empirical probabilities of success for the targets at hand. This could be done for several targets with equal weight, as a way of comparing performances.

Most of the rotations to simplicity that we proved in this paper for arrays with 3×3 or 4×4 symmetric slices were first suggested to us by SIMPLIMAX. Moreover, maximal simplicity was also inspected. For each array order we examined, we ran not fixed SIMPLIMAX for 100 randomly generated arrays, aiming for targets with smaller weight than the simple forms we present. The results concerning orders $3 \times 3 \times K$ for $K = 2, 4, 5$ were consistent with the maximal simplicity we proved in this paper. For the arrays with 4×4 symmetric slices we considered, simulations indicate that weight 18 seems to be the maximal simplicity to expect for $K = 8$ slices. As for $4 \times 4 \times 9$ symmetric slice arrays, simulations seem to indicate that weight less than 16 does not happen. Moreover, the situations for which weight 18 and 17 simple forms (28) and (29) were developed do not seem to admit simpler forms.

11. Discussion

We have worked under the assumption that the arrays are randomly sampled from a continuous distribution, with the constraint of symmetry in the frontal slices. This means that we have ignored cases that arise with probability zero. However, one may question the "random" nature of a core array arising from a 3PCA procedure, as it is a product of an iterative algorithm. As Rocci and Ten Berge [12, p. 362] argue, "…we cannot infer that simplicity transformations which work almost surely for random arrays will also work for Tucker-3 core arrays. Fortunately, all Tucker-3 core arrays encountered so far do seem to behave as if randomly sampled from a continuous distribution, and do allow transformations to simplicity …". Still, a formal proof for this is lacking.

The results of this paper have direct implications for the possibility of simplifying core arrays in Tucker 3-way PCA. However, the realm of possible applications is more general. Matrix theory on the simultaneous reduction of pairs of matrices to sparse forms is abundant, but results for more than two matrices seem absent. The present paper explores the possibilities of filling this gap. For instance, it has been shown that $3 \times 3 \times 4$ arrays of symmetric slices can almost surely be reduced to a form where each of the four slices has weight 2. This is an extension of matrix theory that will be of interest beyond the realm of Tucker-3 PCA.

References

[1] J.D. Carroll, J.J. Chang, Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart–Young decomposition, Psychometrika 35 (1970) 283–319.
[2] F.M. Fisher, The Identification problem in Econometrics, McGraw-Hill, New York, 1966.
[3] R.A. Harshman, Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multi-model factor analysis, University of California at Los Angeles. UCLA Working Papers in Phonetics, vol. 16, 1970, pp. 1–84.
[4] R.A. Harshman, Determination and proof of minimum uniqueness conditions for PARAFAC1, University of California at Los Angeles. UCLA Working Papers in Phonetics, vol. 22, 1972, pp. 111–117.
[5] H.A.L. Kiers, TUCKALS core rotations and constrained TUCKALS modeling, Stat. Appl. 4 (1992) 659–667.
[6] H.A.L. Kiers, Three-way SIMPLIMAX for oblique rotation of the three-mode factor analysis core to simple structure, Comput. Stat. Data Anal. 28 (1998) 307–324.
[7] P.M. Kroonenberg, J. de Leeuw, Principal component analysis of three-mode data by means of alternating least squares, Psychometrika 45 (1980) 69–97.
[8] J.B. Kruskal, Three-way arrays: Rank and uniqueness of trilinear decompositions, with applications to arithmetic complexity and statistics, Linear Algebra Appl. 18 (1977) 95–138.

[9] J.B. Kruskal, Rank, decomposition, and uniqueness for 3-way and *N*-way arrays, in: R. Coppi, S. Bolasco (Eds.), Multiway Data Analysis, North-Holland, Amsterdam, 1989, pp. 7–18.
[10] T. Murakami, J.M.F. Ten Berge, H.A.L. Kiers, A case of extreme simplicity of the core matrix in three-mode principal components analysis, Psychometrika 63 (1998) 255–261.
[11] R. Rocci, J.M.F. Ten Berge, A simplification of a result by Zellini on the maximal rank of symmetric three-way arrays, Psychometrika 59 (1994) 377–380.
[12] R. Rocci, J.M.F. Ten Berge, Transforming three-way arrays to maximal simplicity, Psychometrika 67 (2002) 351–365.
[13] J.M.F. Ten Berge, H.A.L. Kiers, Simplicity of core arrays in three-way principal component analysis and the typical rank of $p \times q \times 2$ arrays, Linear Algebra Appl. 294 (1999) 169–179.
[14] J.M.F. Ten Berge, The typical rank of tall three-way arrays, Psychometrika 65 (2000) 525–532.
[15] J.M.F. Ten Berge, H.A.L. Kiers, T. Murakami, R. van der Heijden, Transforming three-way arrays to multiple orthonormality, J. Chem. 14 (2000) 275–284.
[16] J.M.F. Ten Berge, N.D. Sidiropoulos, R. Rocci, Typical rank and Indscal dimensionality for symmetric three-way arrays of order $I \times 2 \times 2$ or $I \times 3 \times 3$, Linear Algebra Appl. 388 (2004) 363–377.
[17] L.R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (1966) 279–311.
[18] F. Uhlig, A canonical form for a pair of real symmetric matrices that generate a nonsingular pencil, Linear Algebra Appl. 14 (1976) 189–209.

Appendix II

Tendeiro, J. N., Bennani Dosse, M., & Ten Berge, J. M. F. (accepted).
First and second order derivatives for CP and INDSCAL.
Chemometrics and Intelligent Laboratory Systems.

Chemometrics and Intelligent Laboratory Systems xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Chemometrics and Intelligent Laboratory Systems

journal homepage: www.elsevier.com/locate/chemolab

First and second-order derivatives for CP and INDSCAL[☆]

Jorge Tendeiro [a,*], Mohammed Bennani Dosse [b], Jos M.F. ten Berge [a]

[a] University of Groningen, The Netherlands
[b] University of Rennes 2, France

ARTICLE INFO

Article history:
Received 21 January 2010
Received in revised form 6 May 2010
Accepted 22 May 2010
Available online xxxx

Keywords:
CANDECOMP
PARAFAC
INDSCAL
Jacobian matrix
Hessian matrix

ABSTRACT

In this paper we provide the means to analyse the second-order differential structure of optimization functions concerning CANDECOMP/PARAFAC and INDSCAL. Closed-form formulas are given under two types of constraint: unit-length columns or orthonormality of two of the three component matrices. Some numerical problems that might occur during the computation of the Jacobian and Hessian matrices are addressed. The use of these matrices is illustrated in three applications.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Carroll and Chang [3] and Harshman [5] independently presented two identical methods to analyse three-way arrays. The former is CANDECOMP and the latter is PARAFAC; the method is now well known as CANDECOMP/PARAFAC or simply CP. Given a $p\times q\times m$ array $\underline{\mathbf{M}}$ with frontal $p\times q$ slices $\mathbf{M}_i$ $(i=1,\ldots, m)$, CP aims at finding the component matrices $\mathbf{X}$ $(p\times r)$, $\mathbf{Y}$ $(q\times r)$ and $\mathbf{D}$ $(m\times r)$ that minimize the function

$$f(\mathbf{X},\mathbf{Y},\mathbf{D}) = \sum_{i=1}^{m} ||\mathbf{M}_i-\mathbf{X}\mathbf{D}_i\mathbf{Y}'||^2, \tag{1.1}$$

where $\mathbf{D}_i$ is the diagonal matrix holding row i of $\mathbf{D}$ in the diagonal. Minimizing f can be done in various ways. Carroll and Chang [3] and Harshman [5] proposed an alternating least-squares method that has become known as the CP decomposition. However, other approaches have also been proposed. For instance, Paatero [12] has offered a conjugate gradient algorithm.

The CP decomposition starts by initializing $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{D}$, and alternately updates each component matrix while the others remain constant. Iterations are terminated when the relative improvement in f is smaller than a predefined threshold. It is not guaranteed that CP converges; if it does converge, it is not guaranteed that the global minimum is reached. To increase the chances of finding the seeked minimum it is desirable to start CP with several initialization values.

For the special case when the array has symmetric frontal slices, say $\mathbf{S}_1,\ldots,\mathbf{S}_m$ of order $p\times p$, Carroll and Chang [3] proposed INDSCAL, which minimizes the function

$$g(\mathbf{X},\mathbf{D}) = \sum_{i=1}^{m} ||\mathbf{S}_i-\mathbf{X}\mathbf{D}_i\mathbf{X}'||^2. \tag{1.2}$$

Since minimizing g directly seems difficult, Carroll and Chang [3] suggested minimizing f instead. They conjectured that, after convergence, $\mathbf{X}$ and $\mathbf{Y}$ will be equal or, at least, columnwise proportional (i.e., the columns of $\mathbf{Y}$ can be rescaled to match the columns of $\mathbf{X}$, while the columns of $\mathbf{D}$ absorb the inverse scaling). Such matrices will be referred to as being *equivalent*.

Carroll and Chang's conjecture seems to be valid in practical applications. However, counter-examples have already been constructed. Ten Berge and Kiers [16] proved that equivalence may be violated at global minima of f if the slices $\mathbf{S}_i$ are indefinite. When the slices are non-negative definite and $r=1$ then equivalence can be violated only at stationary points that do not correspond to global minima. Ten Berge and Kiers [16] conjectured that such stationary points would be local minima. However, Bennani Dosse and Ten Berge [1] proved that such stationary points must be saddle points. This was achieved by analysing the first and second-order derivatives of a specific optimization function derived from the loss function of CP. Notice that the result by Bennani Dosse and Ten Berge [1] concerns the case where $r=1$ component is used. The conjecture of Carroll and Chang seems to be an open issue when $r>1$ components are used. In this paper, we aimed at finding a second-order sufficient condition that classifies CP

☆ This research has been supported by POPH/FSE (Programa Operacional Potencial Humano/Fundo Social Europeu).

* Corresponding author. Heijmans Institute, University of Groningen, Grote Kruisstraat 2/1, 9712 TS Groningen, The Netherlands. Tel.: +31 503636953.
E-mail address: j.n.tendeiro@rug.nl (J. Tendeiro).

0169-7439/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.chemolab.2010.05.013

decompositions with $r \geq 1$ components as (local) optima or saddle points, see Section 3. With this tool at hand we conducted a simulation study which sheds some light on the equivalence problem, see Section 7. A similar second-order sufficient condition, this time applied to INDSCAL, was also derived, see Section 4.

We extended the research of Bennani Dosse and Ten Berge [1] to the case where $r > 1$ components are extracted. First and second-order derivatives for optimization functions which follow directly from the loss functions of CP and INDSCAL were derived. The reason why the loss functions (1.1), (1.2) were not used is that it is possible to express $\mathbf{D}$ as a function of $\mathbf{X}$ and $\mathbf{Y}$ (for f) or as a function of $\mathbf{X}$ (for g) at stationary points, see Sections 3 and 4. This allows simplifying the optimization problem: the task of minimizing f and g will be replaced by maximizing simpler (= with less variables) optimization functions. Moreover, this is a necessary step if one is to use differential second-order conditions. The main reason is that the Hessian matrix is singular if no elimination of variables is performed, thus drawing inferences about minima and maxima is unwarranted.

Another source of freedom that needs to be controlled is directly related to the fact that the CP model is overparametrized. Namely, given diagonal matrices $\mathbf{\Lambda}_1$, $\mathbf{\Lambda}_2$, $\mathbf{\Lambda}_3$ such that $\mathbf{\Lambda}_1\mathbf{\Lambda}_2\mathbf{\Lambda}_3 = \mathbf{I}_r$, both $(\mathbf{X}, \mathbf{Y}, \mathbf{D})$ and $(\mathbf{X\Lambda}_1, \mathbf{Y\Lambda}_2, \mathbf{D\Lambda}_3)$ represent the same solution. This scaling indeterminacy is considered to be trivial in CP. Nevertheless it does pose a problem when optimizing f using differential tools since one has $f(\mathbf{X}, \mathbf{Y}, \mathbf{D}) = f(\mathbf{X\Lambda}_1, \mathbf{Y\Lambda}_2, \mathbf{D\Lambda}_3)$, ie, for each $(\mathbf{X}, \mathbf{Y}, \mathbf{D})$ in the domain of f there is an infinity of points which are mapped onto $f(\mathbf{X}, \mathbf{Y}, \mathbf{D})$. This has the effect of making the second-order sufficient conditions useless, since in these conditions the Hessian matrix will invariably fail to be non-singular. Therefore, determining the nature of stationary points of f via its second-order differential structure becomes unfeasible under the current setting. Notice that a similar problem applies also to INDSCAL and its associate function g, since an INDSCAL solution is also characterized by scaling indeterminacy. Also, the new optimization functions that will be derived from f and g suffer from the same problem. Since the analysis of second-order structures is one of the goals in this paper, something had to be done to overcome this issue. Constraining the domain of the optimization functions is a possible solution to the problem discussed in the previous paragraph. We settled for two types of constraint: $\mathbf{X}$ and $\mathbf{Y}$ constrained to hold unit length columns (Case I), and $\mathbf{X}$ and $\mathbf{Y}$ constrained by orthonormality (Case II). The first constraint is a so-called identification constraint; it involves no loss of fit. The second constraint is active, thus a loss of fit is due to happen when compared to the unconstrained situation. Both constraints proved to eliminate the problem of singularity of the Hessian matrix in the vast majority of the cases. Some exceptions were found, as will be discussed in latter sections.

The utility of the second-order conditions that we present in this paper extends beyond the study of the equivalence problem. In fact, minimizing f is not a straightforward optimization problem. First of all, there is usually no closed-form solution. Moreover, a solution might not even *exist*. For example, the $2 \times 2 \times 2$ symmetric slice array analysed by Ten Berge, Kiers and De Leeuw [17] showed that the loss function (1.1) has an infimum which is not a minimum. More recently, Stegeman [13] showed that (1.1) does not have a minimum when $p \times p \times 2$ arrays of rank $p+1$ or higher are decomposed into p rank-1 arrays and a residual array (see also Stegeman [14] for a follow-up). Other problems that might affect the search and quality of an optimal solution for f are: preprocessing the data, the number of components to retain, the choice of the initialization values for the algorithm, slow convergence of the algorithm, existence, uniqueness or “illness” of the solution. The fact that CP does not always converge, or that it might converge to non-optimal points, raises questions concerning the nature of the limiting points of a CP sequence. Similar questions apply to INDSCAL solutions and target function (1.2). These observations reinforce the benefit of having available a tool like the one we propose in this paper. Since our tool allows to better characterize a CP or INDSCAL solution, we have a better insight into the nature of such solution. Specifically, if a solution proves to be a saddle point, then one is sure that it cannot correspond to the seeked global minimum. Therefore a new run of the algorithm is required, possibly with different (random) starting values.

There has been some research in the past concerning the study of the differential structure of the optimization function for CP. Paatero [11,12] has developed formulas for the Jacobian and Hessian matrices for loss function (1.1). However, our approach differs from Paatero's in two ways. Firstly, Paatero does not perform variable elimination. Secondly, Paatero derives a numerical approximation to the Hessian matrix, whereas we propose in the present paper Hessian matrices in closed form.

In this paper we will use matrix differential calculus; definitions and useful differential formulas are to be found in Section 2 and Appendix A. All formulas and necessary derivations for the Jacobian and Hessian matrices are the core of Sections 3 and 4.

The benefit of analysing second-order differential structures is first illustrated in Section 5, where we revisit the data analysed by Ten Berge, Kiers and De Leeuw [17]. It is shown that, for this data, saddle points occur very often when the ALS algorithm is initialized by randomly generating orthonormal component matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$.

In Sections 6–8 we describe three simulation studies that were carried out. In the first study we used an algorithm for INDSCAL with orthogonality constraints (Ten Berge, Knol and Kiers [18]) to fit $3 \times 3 \times 3$ arrays with positive definite slices and also $3 \times 3 \times 3$ arrays with indefinite slices. We wished to detect non-optimal solutions, and to see whether they corresponded to saddle points or not. The goal was to further clarify the characterization of the solutions found in the simulation study of Ten Berge et al. [18]. In the second study we tried to see how a result of Bennani Dosse and Ten Berge ([1], pg. 306) extends to situations with $r > 1$ components. In order to do this, we generated random arrays with positive definite slices and also with indefinite slices and then computed CP solutions with more than one component. We analysed the second-order information for each solution. The first goal was to check whether non-equivalence could occur at all. In case it would occur, we were interested in verifying whether such solutions correspond to saddle points (as it is proven to happen when $r = 1$) or to local optima. The contrast between arrays with positive definite slices and indefinite slices was considered. In both situations we analysed features such as degeneracy and occurrence of different fit values. In the third study we expanded this type of analysis to CP solutions of non-symmetric slice arrays, for 29 different scenarios.

We finish the paper with a Discussion section, where some considerations about numerical stability of our second-order conditions are to be found. It is argued that differentiation might not be possible for degenerate solutions of CP or INDSCAL, since degenerate solutions correspond to points where the optimal functions are nearly non-differentiable. Some caution is therefore needed when analysing cases of this kind.

2. Derivatives of matrix functions with respect to matrix variables

2.1. Notation

Scalars will be denoted by lower case italic font (a, x, λ), vectors by lower case bold-face font ($\mathbf{a}$, $\mathbf{x}$, $\boldsymbol{\lambda}$), matrices by upper case bold-face font ($\mathbf{A}$, $\mathbf{X}$, $\mathbf{\Lambda}$), and arrays by underlined upper case bold-face font ($\underline{\mathbf{A}}$, $\underline{\mathbf{X}}$, $\underline{\mathbf{\Lambda}}$). Given matrix $\mathbf{X}$, $\mathbf{x}_i$ denotes the i-th column of $\mathbf{X}$. The only exceptions to this rule appear in definitions (3.2) and (4.2).

For given matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A}'$ is the transpose of $\mathbf{A}$; tr($\mathbf{A}$) is the trace of $\mathbf{A}$; vec($\mathbf{A}$) reshapes $\mathbf{A}$ into a column vector by stacking the columns in sequence, one below the other; $\mathbf{A} \otimes \mathbf{B}$, $\mathbf{A} * \mathbf{B}$ and $\mathbf{A} \odot \mathbf{B}$ denote the Kronecker, Hadamard and Khatri-Rao products of $\mathbf{A}$ and $\mathbf{B}$, respectively; and $\text{diag}_\text{V}(\mathbf{A})$ is the column vector holding the diagonal of $\mathbf{A}$. Given a vector $\mathbf{d}$, $\text{diag}_\text{M}(\mathbf{d})$ denotes the diagonal matrix whose diagonal is equal to $\mathbf{d}$. $\mathbf{I}_m$ is the identity matrix of order m; $\mathbf{0}_{mn}$ is the zero matrix of order $m \times n$; $\mathbf{C}_{mn}$ is the $mn \times mn$ commutation matrix, i.e., $\mathbf{C}_{mn}\text{vec}(\mathbf{A}) = \text{vec}(\mathbf{A}')$; $\mathbf{T}_n$ is the $n^2 \times n$ matrix with unit entries in position $((i-1)n+i,\ i)$ for $i=1,\ \ldots,\ n$ and zeroes elsewhere, and $\mathbf{E}_n=\mathbf{I}_{n^2}-\mathbf{T}_n\mathbf{T}_n'$. For example, for $n=3$

$$\mathbf{T}_3=\left[\begin{array}{ccc}1&0&0\\0&0&0\\0&0&0\\\hline 0&0&0\\0&1&0\\0&0&0\\\hline 0&0&0\\0&0&0\\0&0&1\end{array}\right],\quad \mathbf{E}_3=\left[\begin{array}{ccc|ccc|ccc}0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0\\\hline 0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0\\\hline 0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0\end{array}\right]. \tag{2.1}$$

In Appendices B and C there can be found several matrix functions ($\mathbf{F}_1$ to $\mathbf{F}_8$, and $\mathbf{G}_1$ to $\mathbf{G}_{16}$), as well as some associated derivatives. These functions will be used throughout the derivations in Sections 3 and 4.

2.2. Differentiation of functions with respect to matrix variables

The Jacobian matrix of f: $\mathbb{R}^n\rightarrow\mathbb{R}^m$ is the $m\times n$ matrix of partial derivatives whose entry $(i,\ j)$ is $\dfrac{\partial f_i(\mathbf{x})}{\partial x_j}$, for $\mathbf{x}\in\mathbb{R}^n$. This notion of Jacobian matrix can be extended to matrix functions with matrix variables: the Jacobian matrix of function $\mathbf{A}$: $\mathbb{R}^{p\times r}\rightarrow\mathbb{R}^{m\times n}$ is the $mn\times pr$ matrix given by $\frac{\partial\mathbf{A}}{\partial\mathbf{X}}=\frac{\partial\text{vec}(\mathbf{A})}{\partial\text{vec}(\mathbf{X})}$.

Given a scalar function f: $\mathbb{R}^n\rightarrow\mathbb{R}$, the associated Hessian matrix $\dfrac{\partial^2 f}{\partial\mathbf{X}^2}$ is the $n\times n$ matrix whose entry $(i,\ j)$ is $\dfrac{\partial^2 f(\mathbf{x})}{\partial x_j\partial x_i}$, for $\mathbf{x}\in\mathbb{R}^n$. The concept of Hessian matrix can be extended to scalar functions with matrix variables as follows: the Hessian matrix of function f: $\mathbb{R}^{p\times r}\rightarrow\mathbb{R}$ is the $pr\times pr$ matrix given by $\frac{\partial^2 f}{\partial\mathbf{X}^2}=\frac{\partial^2 f}{\partial\text{vec}(\mathbf{X})^2}$.

This is how the partial derivatives will be arranged in the sequel. For example, the Jacobian matrix of a scalar function will be a row vector. Also, all differential formulae that will be introduced are adapted to this definition. Notice that there exist authors who choose to display the partial derivatives of the Jacobian and Hessian matrices in a different way than the one done in the present paper. See, for example, Magnus and Neudecker ([10], Chapter 9) for a discussion on this subject. Therefore, some caution is needed before going into the derivations of the next sections.

2.3. Matrix differentiation formulas

In Table A1 (see Appendix A) we summarize the most important formulas of matrix differentiation that are of use in this paper. In Tables B1 and C1 (see Appendices B and C) we define functions $\mathbf{F}_1-\mathbf{F}_8$ and $\mathbf{G}_1-\mathbf{G}_{16}$, for which we present the relevant partial derivatives. These functions appear useful during the differentiation process, as they simplify the presentation of our results. In some functions we add the superscript $^{(i)}$ to denote the dependency of the function on the value of $i=1,\ \ldots,\ m$.

3. Optimization of CP

The loss function of CP (1.1) can be written as

$$f(\mathbf{X},\mathbf{Y},\mathbf{D})=\sum_{i=1}^{m}\left(||\mathbf{M}_i||^2+\text{tr}(\mathbf{Y}\mathbf{D}_i\mathbf{X}'\mathbf{X}\mathbf{D}_i\mathbf{Y}')-2\text{tr}(\mathbf{Y}'\mathbf{M}'_i\mathbf{X}\mathbf{D}_i)\right). \tag{3.1}$$

At stationary points we have

$$\mathbf{d}_i=\text{transposed row } i \text{ of } \mathbf{D}=(\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}\text{diag}_\text{V}(\mathbf{X}'\mathbf{M}_i\mathbf{Y}) \tag{3.2}$$

and

$$\text{tr}(\mathbf{Y}\mathbf{D}_i\mathbf{X}'\mathbf{X}\mathbf{D}_i\mathbf{Y}')=\text{tr}(\mathbf{Y}'\mathbf{M}'_i\mathbf{X}\mathbf{D}_i). \tag{3.3}$$

Formula (3.2) can be directly derived from the equation $\frac{\partial f}{\partial\mathbf{D}_i}=0$; it allows to express $\mathbf{D}$ in terms of $\mathbf{X}$ and $\mathbf{Y}$. Equality (3.3) can be seen as follows: define $\mathbf{e}_i=\text{diag}_\text{V}(\mathbf{Y}'\mathbf{M}'_i\mathbf{X})$, and verify that $\text{tr}(\mathbf{Y}'\mathbf{M}'_i\mathbf{X}\mathbf{D}_i)=\mathbf{e}_i'\mathbf{d}_i$, $\text{tr}(\mathbf{X}'\mathbf{X}\mathbf{D}_i\mathbf{Y}'\mathbf{Y}\mathbf{D}_i)=\mathbf{d}_i'(\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})\mathbf{d}_i=\mathbf{d}_i'\mathbf{e}_i$. Thus, to optimize the loss function of CP we can work with function

$$f(\mathbf{X},\mathbf{Y})=\sum_{i=1}^{m}\left(||\mathbf{M}_i||^2-\text{tr}(\mathbf{Y}'\mathbf{M}'_i\mathbf{X}\mathbf{D}_i)\right), \tag{3.4}$$

for $\mathbf{D}$ defined as in Eq. (3.2). Minimizing Eq. (3.4) is equivalent to maximizing

$$L^{\text{CP}}(\mathbf{X},\mathbf{Y})=\sum_{i=1}^{m}\text{tr}(\mathbf{Y}'\mathbf{M}'_i\mathbf{X}\mathbf{D}_i), \tag{3.5}$$

for $\mathbf{D}$ defined by Eq. (3.2).

We wish to describe a sufficient condition for a stationary point of $L^{\text{CP}}(\mathbf{X},\mathbf{Y})$ to be a (local) maximum. In order to do this, we will derive the Jacobian and Hessian matrices for $L^{\text{CP}}(\mathbf{X},\mathbf{Y})$ in two different scenarios: $\mathbf{X},\mathbf{Y}$ constrained to hold columns of unit length (Case I), and $\mathbf{X},\mathbf{Y}$ constrained by orthonormality (Case II). The constrained situations will be dealt with by introducing Lagrange multipliers:

$$L_c^{\text{CP}}(\mathbf{X},\mathbf{Y})=\sum_{i=1}^{m}\text{tr}(\mathbf{Y}'\mathbf{M}'_i\mathbf{X}\mathbf{D}_i)-\text{tr}(\mathbf{\Lambda}(\mathbf{X}'\mathbf{X}-\mathbf{I}_r))-\text{tr}(\mathbf{\Delta}(\mathbf{Y}'\mathbf{Y}-\mathbf{I}_r)). \tag{3.6}$$

In the case that $\mathbf{X}$ and $\mathbf{Y}$ are constrained to have unit length columns we have that $\mathbf{\Lambda}=\text{Diag}([\lambda_i])$ and $\mathbf{\Delta}=\text{Diag}([\delta_i])$ are diagonal $r\times r$ matrices holding Lagrange multipliers, and $\mathbf{D}$ is given by Eq. (3.2) with the diagonal of $\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y}$ filled with 1's. If $\mathbf{X}$ and $\mathbf{Y}$ are constrained by orthonormality then $\mathbf{\Lambda}=[\lambda_{ij}]$ and $\mathbf{\Delta}=[\delta_{ij}]$ are symmetric $r\times r$ matrices holding Lagrange multipliers and $\mathbf{D}$ is given by Eq. (3.2) with $\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y}=\mathbf{I}_r$.

3.1. Derivation of the Jacobian of L_c^{CP}

Define $\gamma_i=\text{tr}(\mathbf{Y}'\mathbf{M}'_i\mathbf{X}\mathbf{D}_i)$; we have

$$\frac{\partial\gamma_i}{\partial\mathbf{X}}=\text{vec}(\mathbf{I}_r)'(\mathbf{I}_r\otimes\mathbf{Y}'\mathbf{M}'_i)\left(\left(\mathbf{D}_i\otimes\mathbf{I}_p\right)+(\mathbf{I}_r\otimes\mathbf{X})\mathbf{T}_r\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}\right). \tag{3.7}$$

In Case I the partial derivative of $\mathbf{d}_i$ with respect to $\mathbf{X}$ is

$$\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}=(\text{diag}_\text{V}(\mathbf{X}'\mathbf{M}_i\mathbf{Y})'\otimes\mathbf{I}_r)\frac{\partial(\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}}{\partial\mathbf{X}}+(\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}\mathbf{T}'_r(\mathbf{I}_r\otimes\mathbf{Y}'\mathbf{M}'_i); \tag{3.8}$$

see Appendix B for the derivation of $\frac{\partial(\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}}{\partial\mathbf{X}}$. In Case II we have that

$$\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}=\mathbf{T}'_r(\mathbf{I}_r\otimes\mathbf{Y}'\mathbf{M}'_i). \tag{3.9}$$

Analogously,

$$\frac{\partial\gamma_i}{\partial\mathbf{Y}}=\text{vec}(\mathbf{I}_r)'(\mathbf{I}_r\otimes\mathbf{X}'\mathbf{M}_i)\left(\left(\mathbf{D}_i\otimes\mathbf{I}_q\right)+(\mathbf{I}_r\otimes\mathbf{Y})\mathbf{T}_r\frac{\partial\mathbf{d}_i}{\partial\mathbf{Y}}\right). \tag{3.10}$$

In Case I the partial derivative of $\mathbf{d}_i$ with respect to $\mathbf{Y}$ is

$$\frac{\partial \mathbf{d}_i}{\partial \mathbf{Y}} = (\mathrm{diag}_V(\mathbf{X}'\mathbf{M}_i\mathbf{Y})'\otimes\mathbf{I}_r)\frac{\partial(\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}}{\partial\mathbf{Y}} + (\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}\mathbf{T}'_r(\mathbf{I}_r\otimes\mathbf{X}'\mathbf{M}_i); \tag{3.11}$$

see Appendix B for the derivation of $\frac{\partial(\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}}{\partial\mathbf{Y}}$. In Case II we have that

$$\frac{\partial \mathbf{d}_i}{\partial \mathbf{Y}} = \mathbf{T}'_r(\mathbf{I}_r\otimes\mathbf{X}'\mathbf{M}_i). \tag{3.12}$$

The Jacobian of L_c^{CP} is the $1\times(p+q)r$ row vector

$$\mathrm{Jac}\left(L_c^{\mathrm{CP}}\right) = \left[\sum_{i=1}^{m}\frac{\partial\gamma_i}{\partial\mathbf{X}}\quad\sum_{i=1}^{m}\frac{\partial\gamma_i}{\partial\mathbf{Y}}\right] - 2\left[\mathrm{vec}(\mathbf{X})'(\boldsymbol{\Lambda}\otimes\mathbf{I}_p)\quad\mathrm{vec}(\mathbf{Y})'(\boldsymbol{\Delta}\otimes\mathbf{I}_q)\right]. \tag{3.13}$$

3.2. Derivation of the Lagrange multipliers

To find expressions for the Lagrange multipliers as functions of $\mathbf{X}$ and $\mathbf{Y}$ we need to solve $\frac{\partial L_c^{\mathrm{CP}}}{\partial\mathbf{X}}=0, \frac{\partial L_c^{\mathrm{CP}}}{\partial\mathbf{Y}}=0$. We shall solve the first equation; the process is the same for the second one. Equation $\frac{\partial L_c^{\mathrm{CP}}}{\partial\mathbf{X}}=0$ is equivalent to $\sum_{i=1}^{m}\frac{\partial\gamma_i}{\partial\mathbf{X}}=2\mathrm{vec}(\mathbf{X})'(\boldsymbol{\Lambda}\otimes\mathbf{I}_p)$. This implies that

$$\sum_{i=1}^{m}\frac{\partial\gamma_i}{\partial\mathbf{x}_k} = 2\sum_{j=1}^{r}\lambda_{jk}\mathbf{x}'_j, \tag{3.14}$$

for $k=1,\ldots,r$. In case I Eq. (3.14) becomes $\sum_{i=1}^{m}\frac{\partial\gamma_i}{\partial\mathbf{x}_k}=2\lambda_{kk}\mathbf{x}'_k$, which implies that

$$\lambda_{kk} = \frac{1}{2}\sum_{i=1}^{m}\frac{\partial\gamma_i}{\partial\mathbf{x}_k}\mathbf{x}_k. \tag{3.15}$$

In case II we have

$$\lambda_{jk} = \frac{1}{2}\sum_{i=1}^{m}\frac{\partial\gamma_i}{\partial\mathbf{x}_k}\mathbf{x}_j, \tag{3.16}$$

for $j=1,\ldots,r$.

3.3. Derivation of the Hessian of L_c^{CP}

Next we derive the second-order derivatives. Define the following constant matrices with respect to $\mathbf{X}$: $\mathbf{J}_1=\mathrm{vec}(\mathbf{I}_r)'(\mathbf{I}_r\otimes\mathbf{Y}'\mathbf{M}'_i)$; $\mathbf{J}_2=-\mathrm{diag}_M(\mathrm{vec}(\mathbf{Y}'\mathbf{Y}))\mathbf{E}_r(\mathbf{I}_{r^2}+\mathbf{C}_{rr})$; $\mathbf{J}_3=\mathbf{T}'_r(\mathbf{I}_r\otimes\mathbf{Y}'\mathbf{M}'_i)$. It can be seen that

$$\begin{aligned}\frac{\partial^2\gamma_i}{\partial\mathbf{X}^2} = \left(\mathbf{I}_{pr}\otimes\mathbf{J}_1\right)\Bigg[&\left(\mathbf{I}_r\otimes\mathbf{C}_{pr}\otimes\mathbf{I}_p\right)\left(\mathbf{I}_{r^2}\otimes\mathrm{vec}\left(\mathbf{I}_p\right)\right)\mathbf{T}_r\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}\\ &+\left(\left(\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}\right)'\otimes\mathbf{I}_{pr}\right)\left(\mathbf{T}'_r\otimes\mathbf{I}_{pr}\right)\left(\mathbf{I}_r\otimes\mathbf{C}_{rr}\otimes\mathbf{I}_p\right)\left(\mathrm{vec}(\mathbf{I}_r)\otimes\mathbf{I}_{pr}\right)\\ &+\left(\mathbf{I}_{pr}\otimes(\mathbf{I}_r\otimes\mathbf{X})\mathbf{T}_r\right)\frac{\partial^2\mathbf{d}_i}{\partial\mathbf{X}^2}\Bigg],\end{aligned} \tag{3.17}$$

with $\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}$ given by Eq. (3.8) (in Case I) or Eq. (3.9) (in Case II). The term $\frac{\partial^2\mathbf{d}_i}{\partial\mathbf{X}^2}$ is $\mathbf{0}_{pr^2,pr}$ in Case II; to derive $\frac{\partial^2\mathbf{d}_i}{\partial\mathbf{X}^2}$ in Case I we start by rewriting Eq. (3.8):

$$\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}} = \mathbf{F}_4^{(i)}\mathbf{J}_2(\mathbf{I}_r\otimes\mathbf{X}') + (\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}\mathbf{J}_3, \tag{3.18}$$

where $\mathbf{F}_4^{(i)}=(\mathrm{diag}_V(\mathbf{X}'\mathbf{M}_i\mathbf{Y})'\otimes\mathbf{I}_r)((\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}\otimes(\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1})$, see Appendix B. We can now write

$$\begin{aligned}\frac{\partial^2\mathbf{d}_i}{\partial\mathbf{X}^2} = {}&((\mathbf{I}_r\otimes\mathbf{X})\mathbf{J}'_2\otimes\mathbf{I}_r)\frac{\partial\mathbf{F}_4^{(i)}}{\partial\mathbf{X}} + \left(\mathbf{I}_{pr}\otimes\mathbf{F}_4^{(i)}\right)\left(\mathbf{I}_{pr}\otimes\mathbf{J}_2\right)\\ &\times\left(\mathbf{I}_r\otimes\mathbf{C}_{pr}\otimes\mathbf{I}_r\right)\left(\mathrm{vec}(\mathbf{I}_r)\otimes\mathbf{I}_{pr}\right)\mathbf{C}_{pr} + (\mathbf{J}'_3\otimes\mathbf{I}_r)\frac{\partial(\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}}{\partial\mathbf{X}}.\end{aligned} \tag{3.19}$$

We proceed in a similar way to derive the second-order derivatives with respect to $\mathbf{Y}$. Define the following constant matrices with respect to $\mathbf{Y}$: $\mathbf{K}_1=\mathrm{vec}(\mathbf{I}_r)'(\mathbf{I}_r\otimes\mathbf{X}'\mathbf{M}_i)$; $\mathbf{K}_2=-\mathrm{diag}_M(\mathrm{vec}(\mathbf{X}'\mathbf{X}))\mathbf{E}_r(\mathbf{I}_{r^2}+\mathbf{C}_{rr})$; $\mathbf{K}_3=\mathbf{T}'_r(\mathbf{I}_r\otimes\mathbf{X}'\mathbf{M}_i)$. It can be seen that

$$\begin{aligned}\frac{\partial^2\gamma_i}{\partial\mathbf{Y}^2} = \left(\mathbf{I}_{qr}\otimes\mathbf{K}_1\right)\Bigg[&\left(\mathbf{I}_r\otimes\mathbf{C}_{qr}\otimes\mathbf{I}_q\right)\left(\mathbf{I}_{r^2}\otimes\mathrm{vec}(\mathbf{I}_q)\right)\mathbf{T}_r\frac{\partial\mathbf{d}_i}{\partial\mathbf{Y}}\\ &+\left(\left(\frac{\partial\mathbf{d}_i}{\partial\mathbf{Y}}\right)'\otimes\mathbf{I}_{qr}\right)\left(\mathbf{T}'_r\otimes\mathbf{I}_{qr}\right)\left(\mathbf{I}_r\otimes\mathbf{C}_{rr}\otimes\mathbf{I}_q\right)\left(\mathrm{vec}(\mathbf{I}_r)\otimes\mathbf{I}_{qr}\right)\\ &+\left(\mathbf{I}_{qr}\otimes(\mathbf{I}_r\otimes\mathbf{Y})\mathbf{T}_r\right)\frac{\partial^2\mathbf{d}_i}{\partial\mathbf{Y}^2}\Bigg],\end{aligned} \tag{3.20}$$

with $\frac{\partial\mathbf{d}_i}{\partial\mathbf{Y}}$ given by Eq. (3.11) (in Case I) or Eq. (3.12) (in Case II). The term $\frac{\partial^2\mathbf{d}_i}{\partial\mathbf{Y}^2}$ is $\mathbf{0}_{qr^2,\mathrm{qr}}$ in Case II; to derive $\frac{\partial^2\mathbf{d}_i}{\partial\mathbf{Y}^2}$ in Case I we start by rewriting (3.11):

$$\frac{\partial\mathbf{d}_i}{\partial\mathbf{Y}} = \mathbf{F}_4^{(i)}\mathbf{K}_2(\mathbf{I}_r\otimes\mathbf{Y}') + (\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}\mathbf{K}_3. \tag{3.21}$$

We can now write:

$$\begin{aligned}\frac{\partial^2\mathbf{d}_i}{\partial\mathbf{Y}^2} = {}&((\mathbf{I}_r\otimes\mathbf{Y})\mathbf{K}'_2\otimes\mathbf{I}_r)\frac{\partial\mathbf{F}_4^{(i)}}{\partial\mathbf{Y}}\\ &+\left(\mathbf{I}_{qr}\otimes\mathbf{F}_4^{(i)}\right)\left(\mathbf{I}_{qr}\otimes\mathbf{K}_2\right)\left(\mathbf{I}_r\otimes\mathbf{C}_{qr}\otimes\mathbf{I}_r\right)\left(\mathrm{vec}(\mathbf{I}_r)\otimes\mathbf{I}_{qr}\right)\mathbf{C}_{qr}\\ &+(\mathbf{K}'_3\otimes\mathbf{I}_r)\frac{\partial(\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}}{\partial\mathbf{Y}}.\end{aligned} \tag{3.22}$$

In order to derive the crossed derivative define the following constants with respect to $\mathbf{Y}$: $\mathbf{L}_1=\mathrm{vec}(\mathbf{I}_r)'$; $\mathbf{L}_2=(\mathbf{I}_r\otimes\mathbf{X})\mathbf{T}_r$; $\mathbf{L}_3=-\mathbf{E}_r(\mathbf{I}_{r^2}+\mathbf{C}_{rr})(\mathbf{I}_r\otimes\mathbf{X}')$. We can rewrite $\frac{\partial\gamma_i}{\partial\mathbf{X}}$:

$$\frac{\partial\gamma_i}{\partial\mathbf{X}} = \mathbf{L}_1(\mathbf{I}_r\otimes\mathbf{Y}'\mathbf{M}'_i)\left(\left(\mathbf{D}_i\otimes\mathbf{I}_p\right)+\mathbf{L}_2\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}\right). \tag{3.23}$$

Differentiating $\frac{\partial\gamma_i}{\partial\mathbf{X}}$ with respect to $\mathbf{Y}$ gives us

$$\begin{aligned}\frac{\partial}{\partial\mathbf{Y}}\left(\frac{\partial\gamma_i}{\partial\mathbf{X}}\right) = {}&\left(\left(\mathbf{D}_i\otimes\mathbf{I}_p\right)+\left(\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}\right)'\mathbf{L}'_2\right)\left(\mathbf{I}_{pr}\otimes\mathbf{L}_1\right)\frac{\partial(\mathbf{I}_r\otimes\mathbf{Y}'\mathbf{M}'_i)}{\partial\mathbf{Y}}\\ &+\left(\mathbf{I}_{pr}\otimes\mathbf{L}_1(\mathbf{I}_r\otimes\mathbf{Y}'\mathbf{M}'_i)\right)\left(\frac{\partial\left(\mathbf{D}_i\otimes\mathbf{I}_p\right)}{\partial\mathbf{Y}}+(\mathbf{I}_{pr}\otimes\mathbf{L}_2)\frac{\partial}{\partial\mathbf{Y}}\left(\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}\right)\right),\end{aligned} \tag{3.24}$$

see Appendix B for the derivations of $\frac{\partial\left(\mathbf{I}_r\otimes\mathbf{Y}'\mathbf{M}'_i\right)}{\partial\mathbf{Y}}$ and $\frac{\partial(\mathbf{D}_i\otimes\mathbf{I}_p)}{\partial\mathbf{Y}}$. We have $\frac{\partial}{\partial\mathbf{Y}}\left(\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}\right)$ left to derive. Start by rewriting $\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}}$:

$$\frac{\partial\mathbf{d}_i}{\partial\mathbf{X}} = \mathbf{F}_6^{(i)}\mathbf{L}_3 + (\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y})^{-1}\mathbf{T}'_r(\mathbf{I}_r\otimes\mathbf{Y}'\mathbf{M}'_i)\text{ for Case I} \tag{3.25}$$

$$\frac{\partial \mathbf{d}_i}{\partial \mathbf{X}} = \mathbf{T}'_r(\mathbf{I}_r \otimes \mathbf{Y}'\mathbf{M}'_i) \text{ for Case II,} \tag{3.26}$$

where $\mathbf{F}_6^{(i)} = (\mathrm{diag}_{\mathrm{V}}(\mathbf{X}'\mathbf{M}_i\mathbf{Y})' \otimes \mathbf{I}_r)\left((\mathbf{X}'\mathbf{X} * \mathbf{Y}'\mathbf{Y})^{-1} \otimes (\mathbf{X}'\mathbf{X} * \mathbf{Y}'\mathbf{Y})^{-1}\right)\mathrm{diag}_{\mathrm{M}}(\mathrm{vec}(\mathbf{Y}'\mathbf{Y}))$.

We have that

$$\frac{\partial}{\partial \mathbf{Y}}\left(\frac{\partial \mathbf{d}_i}{\partial \mathbf{X}}\right) = (\mathbf{L}'_3 \otimes \mathbf{I}_r)\frac{\partial \mathbf{F}_6^{(i)}}{\partial \mathbf{Y}} + ((\mathbf{I}_r \otimes \mathbf{M}_i\mathbf{Y})\mathbf{T}_r \otimes \mathbf{I}_r)\frac{\partial (\mathbf{X}'\mathbf{X} * \mathbf{Y}'\mathbf{Y})^{-1}}{\partial \mathbf{Y}} + \left(\mathbf{I}_{pr} \otimes (\mathbf{X}'\mathbf{X} * \mathbf{Y}'\mathbf{Y})^{-1}\right)\left(\mathbf{I}_{pr} \otimes \mathbf{T}'_r\right)\frac{\partial (\mathbf{I}_r \otimes \mathbf{Y}'\mathbf{M}'_i)}{\partial \mathbf{Y}} \tag{3.27}$$

for Case I, and

$$\frac{\partial}{\partial \mathbf{Y}}\left(\frac{\partial \mathbf{d}_i}{\partial \mathbf{X}}\right) = (\mathbf{I}_{pr} \otimes \mathbf{T}'_r)\frac{\partial (\mathbf{I}_r \otimes \mathbf{Y}'\mathbf{M}'_i)}{\partial \mathbf{Y}} \tag{3.28}$$

for Case II.

The Hessian of L_c^{CP} is the $(p+q)r \times (p+q)r$ symmetric matrix

$$\mathrm{Hess}\left(L_c^{\mathrm{CP}}\right) = \begin{bmatrix} \dfrac{\partial^2 L^{\mathrm{CP}}}{\partial \mathbf{X}^2} & \dfrac{\partial}{\partial \mathbf{Y}}\left(\dfrac{\partial L^{\mathrm{CP}}}{\partial \mathbf{X}}\right) \\ \dfrac{\partial}{\partial \mathbf{X}}\left(\dfrac{\partial L^{\mathrm{CP}}}{\partial \mathbf{Y}}\right) & \dfrac{\partial^2 L^{\mathrm{CP}}}{\partial \mathbf{Y}^2} \end{bmatrix} - 2\begin{bmatrix} \mathbf{\Lambda} \otimes \mathbf{I}_p & \mathbf{0} \\ \mathbf{0} & \mathbf{\Delta} \otimes \mathbf{I}_q \end{bmatrix}, \tag{3.29}$$

where $\frac{\partial^2 L^{\mathrm{CP}}}{\partial \mathbf{X}^2} = \sum_{i=1}^{m} \frac{\partial^2 \gamma_i}{\partial \mathbf{X}^2}$, $\frac{\partial^2 L^{\mathrm{CP}}}{\partial \mathbf{Y}^2} = \sum_{i=1}^{m} \frac{\partial^2 \gamma_i}{\partial \mathbf{Y}^2}$, $\frac{\partial}{\partial \mathbf{Y}}\left(\frac{\partial L^{\mathrm{CP}}}{\partial \mathbf{X}}\right) = \sum_{i=1}^{m} \frac{\partial}{\partial \mathbf{Y}}\left(\frac{\partial \gamma_i}{\partial \mathbf{X}}\right)$ and $\frac{\partial}{\partial \mathbf{X}}\left(\frac{\partial L^{\mathrm{CP}}}{\partial \mathbf{Y}}\right) = \left(\frac{\partial}{\partial \mathbf{Y}}\left(\frac{\partial L^{\mathrm{CP}}}{\partial \mathbf{X}}\right)\right)'$.

3.4. Sufficient second-order conditions

A sufficient condition for a stationary point of L_c^{CP} to be a maximum depends on the type of constraint:

- in Case I it is sufficient for a maximum that $\mathbf{W}'\frac{\partial^2 L_c^{\mathrm{CP}}}{\partial \mathbf{X}\partial \mathbf{Y}}\mathbf{W}$ is negative definite, where $\mathbf{W}$ is the $(p+q)r \times (p+q-2)r$ matrix whose columns span the subspace orthogonal to $\begin{bmatrix} \mathbf{I}_r \odot \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_r \odot \mathbf{Y} \end{bmatrix}$;
- in Case II it is sufficient for a maximum that $\mathbf{W}'\frac{\partial^2 L_c^{\mathrm{CP}}}{\partial \mathbf{X}\partial \mathbf{Y}}\mathbf{W}$ is negative definite, where $\mathbf{W}$ is the $(p+q)r \times (p+q-r-1)r$ matrix whose columns span the subspace orthogonal to matrix

$$\left[\begin{array}{cc|cc} \mathbf{I}_r \odot \mathbf{X} & \mathbf{0} & \mathbf{H}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_r \odot \mathbf{X} & \mathbf{0} & \mathbf{H}_2 \end{array}\right], \tag{3.30}$$

where

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{x}_2 & \cdots & \mathbf{x}_r & & & & \cdots & \\ \mathbf{x}_1 & & & \mathbf{x}_3 & \cdots & \mathbf{x}_r & \cdots & \\ & & & \mathbf{x}_2 & & & \cdots & \\ & \ddots & & & \ddots & & \cdots & \\ & & & & & & \cdots & \mathbf{x}_r \\ & & \mathbf{x}_1 & & & \mathbf{x}_2 & \cdots & \mathbf{x}_{r-1} \end{bmatrix} \tag{3.31}$$

and $\mathbf{H}_2$ is similar to $\mathbf{H}_1$ with all occurrences of $\mathbf{x}$'s replaced by $\mathbf{y}$'s, Magnus and Neudecker ([10], Chapter 7).

4. Optimization of INDSCAL

In a similar fashion as was done for CP, we can reformulate the problem of minimizing the loss function (1.2) of INDSCAL as equivalent to the problem of maximizing

$$L^{\mathrm{IND}}(\mathbf{X}) = \sum_{i=1}^{m} \mathrm{tr}\left(\mathbf{X}'\mathbf{S}_i\mathbf{X}\tilde{\mathbf{D}}_i\right), \tag{4.1}$$

where $\tilde{\mathbf{D}}_\mathbf{i}$ is the diagonal matrix holding

$$\tilde{\mathbf{d}}_i = \text{transposed row } i \text{ of } \tilde{\mathbf{D}} = (\mathbf{X}'\mathbf{X} * \mathbf{X}'\mathbf{X})^{-1}\mathrm{diag}_{\mathrm{V}}(\mathbf{X}'\mathbf{S}_i\mathbf{X}) \tag{4.2}$$

in the diagonal. The Lagrangean is defined by

$$L_c^{\mathrm{IND}}(\mathbf{X}) = \sum_{i=1}^{m} \mathrm{tr}\left(\mathbf{X}'\mathbf{S}_i\mathbf{X}\tilde{\mathbf{D}}_i\right) - \mathrm{tr}(\mathbf{\Lambda}(\mathbf{X}'\mathbf{X} - \mathbf{I}_r)), \tag{4.3}$$

where $\mathbf{\Lambda} = \mathrm{Diag}([\lambda_i])$ is a diagonal $r \times r$ matrix holding Lagrange multipliers and $\tilde{\mathbf{D}}$ is given by Eq. (4.2) with the diagonal of $\mathbf{X}'\mathbf{X} * \mathbf{X}'\mathbf{X}$ filled with 1's in Case I, or $\mathbf{\Lambda} = [\lambda_{ij}]$ is a symmetric $r \times r$ matrix holding Lagrange multipliers and $\tilde{\mathbf{D}}$ is given by Eq. (4.2) with $\mathbf{X}'\mathbf{X} * \mathbf{X}'\mathbf{X} = \mathbf{I}_r$ in Case II.

4.1. Derivation of the Jacobian of L_c^{IND}

Define $\sigma_i = \mathrm{tr}(\mathbf{X}'\mathbf{S}_i\mathbf{X}\tilde{\mathbf{D}}_i)$. We have

$$\frac{\partial \sigma_i}{\partial \mathbf{X}} = \mathrm{vec}(\mathbf{I}_r)'\left(\left(\tilde{\mathbf{D}}_i\mathbf{X}' \otimes \mathbf{I}_r\right)(\mathbf{S}_i \otimes \mathbf{I}_r)\mathbf{C}_{pr} + (\mathbf{I}_r \otimes \mathbf{X}'\mathbf{S}_i) \times \left(\left(\tilde{\mathbf{D}}_i \otimes \mathbf{I}_p\right) + (\mathbf{I}_r \otimes \mathbf{X})\mathbf{T}_r\frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}}\right)\right). \tag{4.4}$$

In Case I the partial derivative of $\tilde{\mathbf{d}}_i$ with respect to $\mathbf{X}$ is

$$\frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}} = \left(\mathrm{diag}_{\mathrm{V}}(\mathbf{X}'\mathbf{S}_i\mathbf{X})' \otimes \mathbf{I}_r\right)\frac{\partial (\mathbf{X}'\mathbf{X} * \mathbf{X}'\mathbf{X})^{-1}}{\partial \mathbf{X}} + (\mathbf{X}'\mathbf{X} * \mathbf{X}'\mathbf{X})^{-1}\mathbf{T}'_r\left((\mathbf{X}'\mathbf{S}_i \otimes \mathbf{I}_r)\mathbf{C}_{pr} + (\mathbf{I}_r \otimes \mathbf{X}')(\mathbf{I}_r \otimes \mathbf{S}_i)\right), \tag{4.5}$$

see Appendix C for the derivation of $\frac{\partial (\mathbf{X}'\mathbf{X} * \mathbf{X}'\mathbf{X})^{-1}}{\partial \mathbf{X}}$. In Case II we have that

$$\frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}} = \mathbf{T}'_r\left((\mathbf{X}'\mathbf{S}_i \otimes \mathbf{I}_r)\mathbf{C}_{pr} + (\mathbf{I}_r \otimes \mathbf{X}')(\mathbf{I}_r \otimes \mathbf{S}_i)\right). \tag{4.6}$$

The Jacobian of L_c^{IND} is the $1 \times pr$ row vector

$$\mathrm{Jac}\left(L_c^{\mathrm{IND}}\right) = \sum_{i=1}^{m} \frac{\partial \sigma_i}{\partial \mathbf{X}} - 2\mathrm{vec}(\mathbf{X})'\left(\mathbf{\Lambda} \otimes \mathbf{I}_p\right). \tag{4.7}$$

4.2. Derivation of the Lagrange multipliers

Proceeding in a similar fashion as done in Section 3, it is straightforward to verify that

$$\lambda_{kk} = \frac{1}{2}\sum_{i=1}^{m} \frac{\partial \sigma_i}{\partial \mathbf{x}_k}\mathbf{x}_k \tag{4.8}$$

in Case I, and

$$\lambda_{jk} = \frac{1}{2}\sum_{i=1}^{m} \frac{\partial \sigma_i}{\partial \mathbf{x}_k}\mathbf{x}_j \tag{4.9}$$

in Case II ($j, k = 1, \ldots, r$).

4.3. Derivation of the Hessian of L_c^{IND}

Now define the following matrices which are constant with respect to $\mathbf{X}$: $\mathbf{N}_1 = \mathrm{vec}(\mathbf{I}_r)'$; $\mathbf{N}_2 = (\mathbf{S}_i \otimes \mathbf{I}_r)\mathbf{C}_{pr}$; $\mathbf{N}_3 = \mathbf{E}_r(\mathbf{I}_{r^2} + \mathbf{C}_{rr})$. We can rewrite

$$\frac{\partial \sigma_i}{\partial \mathbf{X}} = \mathbf{N}_1\left(\mathbf{G}_2^{(i)}\mathbf{N}_2 + \mathbf{G}_6^{(i)} + \mathbf{G}_7^{(i)}\mathbf{T}_r \frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}}\right). \tag{4.10}$$

It can be seen that

$$\frac{\partial^2 \sigma_i}{\partial \mathbf{X}^2} = \left(\mathbf{I}_{pr} \otimes \mathbf{N}_1\right)\left((\mathbf{N}'_2 \otimes \mathbf{I}_{r^2})\frac{\partial \mathbf{G}_2^{(i)}}{\partial \mathbf{X}} + \frac{\partial \mathbf{G}_6^{(i)}}{\partial \mathbf{X}} + \left(\left(\frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}}\right)' \otimes \mathbf{I}_{r^2}\right) \times (\mathbf{T}'_r \otimes \mathbf{I}_{r^2})\frac{\partial \mathbf{G}_7^{(i)}}{\partial \mathbf{X}} + \left(\mathbf{I}_{pr} \otimes \mathbf{G}_7^{(i)}\mathbf{T}_r\right)\frac{\partial^2 \tilde{\mathbf{d}}_i}{\partial \mathbf{X}^2}\right), \tag{4.11}$$

with $\frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}}$ given by Eq. (4.5) (in Case I) or Eq. (4.6) (in Case II). To derive $\frac{\partial^2 \tilde{\mathbf{d}}_i}{\partial \mathbf{X}^2}$ in Case I we start by rewriting Eq. (4.5):

$$\frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}} = 2\mathbf{G}_{14}^{(i)}\mathbf{N}_3\mathbf{G}'_5 + \mathbf{G}_8\mathbf{T}'_r\mathbf{G}_{16}^{(i)}. \tag{4.12}$$

It can be seen that

$$\frac{\partial^2 \tilde{\mathbf{d}}_i}{\partial \mathbf{X}^2} = 2(\mathbf{G}_5\mathbf{N}'_3 \otimes \mathbf{I}_r)\frac{\partial \mathbf{G}_{14}^{(i)}}{\partial \mathbf{X}} + 2\left(\mathbf{I}_{pr} \otimes \mathbf{G}_{14}^{(i)}\right)\left(\mathbf{I}_{pr} \otimes \mathbf{N}_3\right)\mathbf{C}_{pr,r^2}\frac{\partial \mathbf{G}_5}{\partial \mathbf{X}} + \tag{4.13}$$

$$+ \left(\left(\mathbf{G}_{16}^{(i)}\right)'\mathbf{T}_r \otimes \mathbf{I}_r\right)\frac{\partial \mathbf{G}_8}{\partial \mathbf{X}} + \left(\mathbf{I}_{pr} \otimes \mathbf{G}_8\right)\left(\mathbf{I}_{pr} \otimes \mathbf{T}'_r\right)\frac{\partial \mathbf{G}_{16}^{(i)}}{\partial \mathbf{X}}. \tag{4.14}$$

In Case II we have that

$$\frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}} = \mathbf{T}'_r\mathbf{G}_{16}^{(i)}, \quad \frac{\partial^2 \tilde{\mathbf{d}}_i}{\partial \mathbf{X}^2} = \left(\mathbf{I}_{pr} \otimes \mathbf{T}'_r\right)\frac{\partial \mathbf{G}_{16}^{(i)}}{\partial \mathbf{X}}. \tag{4.15}$$

The Hessian of L_c^{IND} is the $pr \times pr$ symmetric matrix

$$\mathrm{Hess}\left(L_c^{\mathrm{IND}}\right) = \sum_{i=1}^{m} \frac{\partial^2 \sigma_i}{\partial \mathbf{X}^2} - 2\left(\mathbf{\Lambda} \otimes \mathbf{I}_p\right). \tag{4.16}$$

4.4. Sufficient second-order conditions

A sufficient condition for a stationary point of L_c^{IND} to be a maximum depends on the type of constraint:

- in Case I it is sufficient that $\mathbf{W}' \frac{\partial^2 L_c^{\mathrm{IND}}}{\partial \mathbf{X} \partial \mathbf{Y}} \mathbf{W}$ is negative definite, where $\mathbf{W}$ is the $pr \times (pr - r)$ matrix whose columns span the subspace orthogonal to $\mathbf{I}_r \odot \mathbf{X}$;
- in Case II it is sufficient that $\mathbf{W}' \frac{\partial^2 L_c^{\mathrm{IND}}}{\partial \mathbf{X} \partial \mathbf{Y}} \mathbf{W}$ is negative definite, where $\mathbf{W}$ is the $pr \times \left(pr - \frac{r(r+1)}{2}\right)$ matrix whose columns span the subspace orthogonal to matrix

  $$[\mathbf{I}_r \odot \mathbf{X} | \mathbf{H}], \tag{4.17}$$

  where $\mathbf{H}$ is the same as in Eq. (3.31).

5. Illustration: the KHL data

Ten Berge, Kiers and De Leeuw [17] analysed a contrived array which they christened “KHL data”, due to previous work by Kruskal, Harshman and Lundy [8,9]. The KHL data is the $2 \times 2 \times 2$ array

$$\underline{\mathbf{X}} = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \end{array}\right]. \tag{5.1}$$

We ran the ALS algorithm 200 times for $\underline{\mathbf{X}}$ with $r = 2$ components. The component matrices were randomly initialized by orthonormal matrices. In all runs the algorithm halted on solutions with loss $f = 2$. We wanted to test the nature of these solutions, i.e., whether these solutions correspond to minima and/or saddle points.

We computed the Jacobian and Hessian for each of the 200 solutions under unit length constraint. In general, each of the 200 solutions displays a similar behaviour: $\mathbf{A}$ has rank 1, $\mathbf{B}$ and $\mathbf{C}$ are orthonormal, Jac is approximately $\mathbf{0}_{1 \times 8}$ (its entries are usually in the order of 10^{-14}), and the Hess is of the form

$$\mathrm{Hess}_{\mathrm{CP}} = \begin{bmatrix} 0 & 0 & a & b \\ 0 & 0 & -a & -b \\ a & -a & 0 & 0 \\ b & -b & 0 & 0 \end{bmatrix}, \tag{5.2}$$

for real numbers a, b. The eigenvalues of $\mathrm{Hess}_{\mathrm{CP}}$ are typically $\{0, 0, -\lambda, \lambda\}$, for real λ. Therefore, it can be concluded that each of the 200 solutions are, indeed, saddle points.

This example shows two things. On one hand, there exist cases for which the occurrence of saddle points is a severe problem, like the KHL data. On the other hand, it is relevant to have a tool available that diagnoses whether a solution is a saddle point. Once spotted, such solutions should be discarded at once.

Ten Berge, Kiers and De Leeuw [17] showed that the CP loss function (1.1) has infimum 1 when 2 components are extracted. This reinforces the fact that none of the 200 solutions that were found could correspond to the global minimum. However, in the absence of this information, the researcher would profit from knowing that all solutions were saddle points and therefore useless. This is possible by analysing the second-order differential structure as we have done here.

The KHL data is a contrived example. The question of whether similar behaviour is to be expected for real data is still unanswered. The applications discussed in Sections 6–8 are intended to better understand what happens in general.

6. Application I: INDSCAL under orthonormality constraint

Ten Berge et al. [18] discussed an algorithm for INDSCAL with orthogonality constraints referred to as the SVD-approach. This algorithm was originally deviced as a Varimax procedure based on an SVD, but Ten Berge [15] observed that the problem could be reformulated in terms of diagonalizing a set of symmetric matrices simultaneously. The SVD-approach provides a direct procedure to fit the INDSCAL model under orthogonality constraints.

The SVD-approach attempts to find a columnwise orthonormal $\mathbf{X}$ such that $L_c^{\mathrm{IND}}(\mathbf{X})$ is maximized; it proceeds as follows:

```
Step 1  Initialize $\mathbf{X}$ ($p \times r$ orthonormal).
Step 2  Compute $\mathbf{D}_i = \mathrm{diag}(\mathbf{X}'\mathbf{S}_i\mathbf{X})$, $i = 1, \ldots, m$.
Step 3  Compute the SVD $\sum_{i=1}^{m} \mathbf{S}_i\mathbf{X}\mathbf{D}_i = \mathbf{PLQ}'$, and update $\mathbf{X}$ by $\mathbf{X} = \mathbf{PQ}'$.
Step 4  Repeat Steps 2 and 3 until the relative increase in $L_c^{\mathrm{IND}}(\mathbf{X})$ is
        smaller than a predefined convergence criterion.
```

The SVD-approach to INDSCAL has been proved to converge monotonically when the frontal slices of array $\underline{\mathbf{S}}$ are positive or negative semidefinite, Ten Berge et al. [18]. Thus, we will work with arrays holding semidefinite frontal slices in the remaining of this section.

Ten Berge et al. [18] ran some experiments where they argue that the SVD-approach to INDSCAL seems to be hampered by the occurrence of local maxima of L_c^{IND}. However, the possibility of the occurrence of saddle points was not considered. Notice that there exist contrived examples for which saddle points do occur. For example, consider the $2 \times 3 \times 3$ array with positive semidefinite slices (Ten Berge and Kiers [16])

$$\mathbf{S}_1 = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{S}_2 = \begin{bmatrix} 3 & -1 & 0 \\ -1 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.1}$$

The (orthonormally constrained) INDSCAL optimal solution with $r = 2$ components is

$$\mathbf{X} = \begin{bmatrix} \sqrt{.5} & -\sqrt{.5} \\ \sqrt{.5} & \sqrt{.5} \\ 0 & 0 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}; \tag{6.2}$$

it corresponds to the global minimum 1 of (1.2). There are, however, non-optimal orthonormally constrained INDSCAL solutions corresponding to saddle points. The following four solutions are stationary points of (1.2) that correspond to non-optimal values of (1.2) (5, 20, 22, 22, respectively). They are all saddle points.

$$\mathbf{X}^{(1)} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{D}^{(1)} = \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix} \tag{6.3}$$

$$\mathbf{X}^{(2)} = \begin{bmatrix} \sqrt{.5} & 0 \\ \sqrt{.5} & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{D}^{(2)} = \begin{bmatrix} 4 & 0 \\ 2 & 1 \end{bmatrix} \tag{6.4}$$

$$\mathbf{X}^{(3)} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{D}^{(3)} = \begin{bmatrix} 0 & 3 \\ 1 & 3 \end{bmatrix} \tag{6.5}$$

$$\mathbf{X}^{(4)} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{D}^{(4)} = \begin{bmatrix} 3 & 0 \\ 3 & 1 \end{bmatrix}. \tag{6.6}$$

The fact that such non-optimal solutions exist does not imply that the SVD-approach algorithm will converge to them. This is precisely the point that we wanted to investigate in this application: is it possible that the SVD-approach algorithm converges to saddle points? The answer to this question can clarify the type of solutions that the SVD-approach usually finds, therefore the interpretation of the solution is further enriched.

A simulation study was carried out to test whether saddle points occur (software: *Matlab* R2008a). We randomly generated 150 $3 \times 3 \times 3$ symmetric slice arrays with positive definite slices. Each slice was generated as $\mathbf{M}'\mathbf{M}$, where $\mathbf{M}$ is a 3×3 matrix whose entries were uniformly generated from the interval $[-1, 1]$. For each array we ran the SVD-approach to INDSCAL with $r = 2$ components using 10 different random initializations for $\mathbf{X}$; each $\mathbf{X}$ was a 3×2 matrix whose entries were uniformly generated from the interval $[-1, 1]$; afterwards, $\mathbf{X}$ was orthonormalized via the Gram–Schmidt procedure. The convergence criterion was fixed at $1e-06$. After convergence, the Jacobian and Hessian for each INDSCAL solution $(\mathbf{X}, \mathbf{D})$ were computed, and we inspected whether $\mathbf{W}' \frac{\partial^2 L_c^{\text{IND}}}{\partial \mathbf{X} \partial \mathbf{Y}} \mathbf{W}$ was negative definite or indefinite (second-order sufficient condition).

The same procedure was repeated, this time for arrays with positive semidefinite slices. Each slice was generated as $\mathbf{M}'\mathbf{M}$, where $\mathbf{M}$ is a 2×3 matrix whose entries were uniformly generated from the interval $[-1, 1]$.

All results were numerically stable, as expected. We verified that the SVD-approach for INDSCAL never halted on saddle points. Also, it was verified that local maxima occurred for 12 arrays (8% of the cases) for arrays with positive definite slices, whereas for arrays with positive semidefinite slices local maxima occurred for 17 arrays (11% of the cases). Although these results do not formally prove that convergence to saddle points is impossible, it can be concluded that there are no indications to that effect.

7. Application II: INDSCAL equivalence problem in CP formulation

Carroll and Chang [3] suggested running CP in order to fit INDSCAL because they conjectured that $\mathbf{X}$ and $\mathbf{Y}$ would end up equal or at least columnwise proportional if CP converged. If Carroll and Chang's conjecture is correct, CP can be used as an algorithm to compute INDSCAL solutions for symmetric slice arrays. This conjecture seems to be valid in practical applications. However, counter-examples have already been constructed. Ten Berge and Kiers [16] proved that equivalence may be violated at global minima of f if the slices $\mathbf{S}_i$ are not positive definite. They considered the array (Ten Berge and Kiers [16])

$$\mathbf{S}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{S}_2 = \begin{bmatrix} 0 & 0 & 2 \\ 0 & -2 & 0 \\ 2 & 0 & 0 \end{bmatrix}, \tag{7.1}$$

for which a global minimum of (1.1) with $r = 2$ components and $\mathbf{X}$ not equivalent to $\mathbf{Y}$ was presented. We ran CP 500 times with $r = 2$ and $r = 3$ components for the previous array. In both cases all runs converged to a global minimum of (1.1) with $\mathbf{X}$ and $\mathbf{Y}$ non-equivalent. When $r = 1$ the algorithm sometimes did converge to a solution with $\mathbf{X}$ and $\mathbf{Y}$ equivalent.

When the slices are nonnegative definite and $r = 1$ then equivalence can be violated only at stationary points that do not correspond to global minima. In this case, Ten Berge and Kiers [16] conjectured that such stationary points would correspond to local minima. However, Bennani Dosse and Ten Berge [1] proved that such stationary points can only be saddle points.

Bennani Dosse, Ten Berge and Tendeiro [2] showed that equivalence occurs when the components are constrained by orthonormality, the slices are positive semidefinite and the saliences are non-negative. It is still not clear whether non-equivalence occurs or not under circumstances different from these, or whether CP converges to saddle points or not. We conducted some simulations to try to understand what happens in cases where components are not orthonormal, slices are not necessarily positive semidefinite, and saliences are unconstrained. Eleven situations were considered, revolving around arrays with 3×3 symmetric slices: $2 \times 3 \times 3$ $(r = 2)$, $3 \times 3 \times 3$ $(r = 2, 3)$, $4 \times 3 \times 3 (r = 2, 3)$, $5 \times 3 \times 3$ $(r = 2, 3, 4)$, $6 \times 3 \times 3$ $(r = 2, 3, 4)$. Both positive definite and indefinite slice arrays were considered.

One hundred arrays were generated for each case. The positive definite slices were generated as $\mathbf{M}'\mathbf{M}$, where $\mathbf{M}$ is a 3×3 matrix whose entries were uniformly generated from the interval $[-1, 1]$. The entries of the diagonal and the upper-triangular parts of the indefinite slices were uniformly generated from the interval $[-1, 1]$; the lower-triangular part of each slice was filled in such that symmetry would occur. Each array was given 100 different random startups; the convergence criterium was set at $1e-08$. No constraint was imposed on the saliences in $\mathbf{D}$. A solution was declared degenerate when at least one of the non-diagonal entries of the so-called triple cosine matrix was below -0.95. Our main goal was to check whether non-equivalence occurred or not, and to what kind of stationary point it corresponded (local optimum or saddle point).

The Jacobian matrices associated to non-degenerate solutions were analysed. Its entries were relatively small (usually with modulus smaller than $1e-05$), thus analysing the second-order differential structure seems legitimate. We worked under unit length constraints. Occurrences of degeneracy and of different values for the loss function were registered. The results found are summarized in Tables 1 and 2. The variables read: NonEquiv = number of arrays for which at least one startup ended up with non-equivalent CP solution; Deg = number of arrays with degenerate solutions, within the 100 startups ($x + y$: $x =$ all 100 startups are degenerate; $y = <100$ startups are degenerate); SadPt = number of arrays for which at least one startup ended in a saddle point, non-degenerate (x/y: $x =$ for CP's Hessian; $y =$ for INDSCAL's Hessian); DiffFit = number of arrays with at least two different values for CP's loss function within the 100 startups, with at least one non-degenerate solution. Some special situations are marked with asterisks, as follows: (*) = the associated CP

Table 1
Arrays with positive definite slices.

Dim. array	# comps	NonEquiv	Deg	SadPt	DifFit
2×3×3	$r=2$	–	–	–	4
3×3×3	$r=2$	–	–	–	12
	$r=3$	–	2+2	–	5
4×3×3	$r=2$	–	–	–	17
	$r=3$	–	3+0	–	8
5×3×3	$r=2$	–	–	–	10
	$r=3$	–	0+1	–	7
	$r=4$	–	40+7	–	20
6×3×3	$r=2$	–	–	–	10
	$r=3$	–	–	–	7
	$r=4$	–	45+7	2/2 (**)	7

solution is degenerate (all components are almost proportional); (**) = one or more of the eigenvalues of the Hessian are relatively small in magnitude (smaller than $1e-01$), indicating that the Hessian is nearly singular; (***) = for one startup of one array the Hessian for CP was nearly singular, but the Hessian for INDSCAL was negative definite.

The first observation to be made is that non-equivalence was never observed for non-degenerate solutions. Since no non-equivalent solution was found for arrays with positive definite slices, it was not possible to test whether the result of Bennani Dosse and Ten Berge [1] for arrays with positive definite slices does apply to cases with $r>1$ components. Also, saddle points were rarely observed. In addition, we can observe that arrays with indefinite slices are more prone to suffer from degeneracy, occurrence of saddle points, and multiple fit values.

The cases reported with (**) are situations where it is not clear whether we are facing a saddle point or not, since the Hessian matrix seems to be almost singular. These cases should be treated with care, since the second-order sufficient condition applies to non-singular Hessian matrices. It is not clear why such points occur. An anonymous reviewer suggested that the problem might be originated in rank-deficient component matrices. We verified that this was true for five of the situations reported by (**). It should be noted that these component matrices were estimated rather than randomly generated, and that these decompositions are not degenerate.

8. Application III: CP in general

A simulation study was conducted to inspect the occurrence of saddle points for CP solutions of generic arrays. Twenty nine situations were considered, for which uniqueness is proved to hold due to Kruskal's sufficient condition for uniqueness (Kruskal [7]). One hundred arrays were randomly generated for each situation, the entries being uniformly generated from the interval $[-1, 1]$. Each array was given 100 different random startups; the convergence criterium was set at $1e-08$. As before, we also registered the occurrences of degeneracies and multiple fit values. Both $r=1$ and $r>1$ situations were considered. We computed the Hessian under unit length constraint. The results found are summarized in Table 3.

Table 2
Arrays with indefinite slices.

Dim. array	# comps	NonEquiv	Deg	SadPt	DifFit
2×3×3	$r=2$	–	38+19	2/1 (**)(***)	32
3×3×3	$r=2$	–	21+39	3/3	62
	$r=3$	3 (*)	63+11	1/1 (**)	19
4×3×3	$r=2$	–	24+39	–	64
	$r=3$	3 (*)	55+15	—	26
5×3×3	$r=2$	–	23+39	–	72
	$r=3$	2 (*)	59+15	1/1 (**)	25
	$r=4$	1 (*)	81+5	1/1 (**)	10
6×3×3	$r=2$	–	20+48	–	76
	$r=3$	–	63+16	1/1	27
	$r=4$	–	86+5	–	7

It can be seen that saddle points occur scarcely; almost all these occurrences relate to a nearly singular Hessian matrix. It is not clear why such solutions occur. In addition, we point out that retaining more components seems to have the effect of increasing the number of degenerate solutions.

9. Discussion

In this paper we dealt with first and second-order differential structures of optimization functions related to CP and INDSCAL. Our goal was to provide a tool to further characterize three-way solutions. Closed form formulas for the Jacobian and Hessian matrices were derived, under two different types of constraints.

Simulations that highlight the usefulness of Hessian structure were performed. The results of the simulations seem to tell that saddle points do not occur frequently, although they do occur with positive probability. In some cases the Hessian matrix showed to be ill-conditioned. The reasons for this phenomenon are still not clear and need further investigation.

Some numerical problems occur when we consider degenerate CP/INDSCAL solutions (Harshman and Lundy [6]). Typically, a degenerate solution is one where some components become more and more proportional, while some entries of these components become larger and larger, as the algorithm progresses. In a degenerate solution, the contributions of some of the degenerate components nearly cancel the contributions of other degenerate components, while the components together contribute to improve the fit.

The computation of the Jacobian and the Hessian matrices are free of numerical problems for CP/INDSCAL solutions which are not degenerate. However, degenerate solutions do lead to problems. These problems are more or less severe depending on how many degenerate components exist and how strong the degeneracy is. The core of this problem resides in matrix $\mathbf{\Gamma}=\mathbf{X}'\mathbf{X}*\mathbf{Y}'\mathbf{Y}$ (for CP) and $\mathbf{\Gamma}=\mathbf{X}'\mathbf{X}*\mathbf{X}'\mathbf{X}$ (for INDSCAL), recall Eqs. (3.2) and (4.2). When a CP/INDSCAL solution is degenerate, $\mathbf{\Gamma}$ becomes almost rank deficient, which creates numerical problems when computing $\mathbf{\Gamma}^{-1}$. Equivalently, the problem is that the optimization function is (almost) non-differentiable at the

Table 3
Arrays with generic slices.

Dim. array	# comps	Deg	SadPt	DifFit
2×2×2	$r=1$	–	–	13
3×2×2	$r=1$	–	–	25
	$r=2$	18+2	1 (**)	1
3×3×2	$r=1$	–	–	39
	$r=2$	25+6	–	14
3×3×3	$r=1$	–	–	43
	$r=2$	19+15	1	30
	$r=3$	50+12	1 (**)	17
4×2×2	$r=1$	–	1	32
	$r=2$	22+2	1 (**)	2
4×3×2	$r=1$	–	1	43
	$r=2$	17+12	–	22
	$r=3$	54+4	1,2 (**)	4
4×4×2	$r=1$	–	–	41
	$r=2$	28+12	1 (**)	26
	$r=3$	53+8	3 (**)	3
4×3×3	$r=1$	–	1 (**)	57
	$r=2$	15+32	1	54
	$r=3$	52+15	–	17
	$r=4$	68+15	1,1 (**)	17
5×2×2	$r=1$	–	–	36
	$r=2$	22+0	–	–
5×3×2	$r=1$	–	–	44
	$r=2$	23+11	1 (**)	29
	$r=3$	59+0	–	—
5×4×2	$r=1$	–	–	57
	$r=2$	14+14	1 (**)	27
	$r=3$	47+11	1 (**)	15
	$r=4$	78+6	1 (**)	7

point corresponding to a degenerate solution. The problem might vary from mild to severe, depending on how close or far is $\boldsymbol{\Gamma}$ from singularity. In some severe situations the computations might need to be completely disregarded. For instance, we observed solutions for which the severity of the deficiency of $\boldsymbol{\Gamma}$ leads to loss of symmetry of the Hessian, which is naturally a serious problem.

A CP solution is never degenerate under an orthonormality constraint on $\mathbf{X}$ and $\mathbf{Y}$, Harshman and Lundy [6]. Likewise, an INDSCAL solution is not degenerate if $\mathbf{X}$ is constrained by orthonormality. Therefore, orthonormality constraints typically avoid any numerical problems. In any other case, we advise to first check whether the solution at hand is degenerate or not. If the solution is not degenerate then the use of the formulas to compute the Jacobian and the Hessian is warranted. In case of degeneracy, one should do some prior analysis on the rank deficiency of $\boldsymbol{\Gamma}$. If the problem is not very severe, it is possible that $\boldsymbol{\Gamma}^{-1}$ is relatively well defined, and therefore all the computations will follow safely. A posterior test to the numerical stability of the process is, for example, to compute the Hessian matrix Hess and afterwards compute $\rho = \mathrm{tr}(\mathrm{Hess} - \mathrm{Hess}')$; large values of ρ (say, $\rho > 1e-20$) indicate that Hess is further from symmetry than it should. Therefore, Hess should be discarded in such cases.

Appendix A. Matrix differentiation formulas

Consider the following matrices: $\mathbf{A}$ $(m \times n)$; $\mathbf{B}$ $(p \times q)$; $\mathbf{d}$ $(n \times 1)$; $\mathbf{D} = \mathrm{diag}_{\mathrm{M}}(\mathbf{d})$.

Table A1 presents the most important formulas of matrix differentiation that are of use in this paper.

Table A1
Matrix differentiation formulas.

$\frac{\partial \mathbf{AB}}{\partial \mathbf{X}} = (\mathbf{B}' \otimes \mathbf{I}_m)\frac{\partial \mathbf{A}}{\partial \mathbf{X}} + (\mathbf{I}_q \otimes \mathbf{A})\frac{\partial \mathbf{B}}{\partial \mathbf{X}}$, if $n = p$	$\frac{\partial \mathrm{tr}(\mathbf{A})}{\partial \mathbf{X}} = \mathrm{vec}(\mathbf{I}_n)'\frac{\partial \mathbf{A}}{\partial \mathbf{X}}$, if $m = n$
$\frac{\partial \mathbf{A}(\mathbf{B}(\mathbf{X}))}{\partial \mathbf{X}} = \frac{\partial \mathbf{A}(\mathbf{Y})}{\partial \mathbf{Y}} \cdot \frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$, where $\mathbf{Y} = \mathbf{B}(\mathbf{X})$	$\frac{\partial \mathbf{A}}{\partial \mathbf{A}} = \mathbf{I}_{mn}, \quad \frac{\partial \mathbf{A}'}{\partial \mathbf{A}} = \mathbf{C}_{mn}$
$\frac{\partial \mathbf{A}'\mathbf{A}}{\partial \mathbf{A}} = (\mathbf{I}_{n^2} + \mathbf{C}_{nn})(\mathbf{I}_n \otimes \mathbf{A}')$	$\frac{\partial \mathbf{A} * \mathbf{B}}{\partial \mathbf{X}} = \mathrm{diag}_{\mathrm{M}}(\mathrm{vec}(\mathbf{B}))\frac{\partial \mathbf{A}}{\partial \mathbf{X}} + \mathrm{diag}_{\mathrm{M}}(\mathrm{vec}(\mathbf{A}))\frac{\partial \mathbf{B}}{\partial \mathbf{X}}$
$\frac{\partial \mathbf{A}^{-1}}{\partial \mathbf{A}} = -\left(\left(\mathbf{A}^{-1}\right)' \otimes \mathbf{A}^{-1}\right)$, if $m = n$	$\frac{\partial \mathbf{D}}{\partial \mathbf{d}} = \mathbf{T}_n$
$\frac{\partial \mathbf{A} \otimes \mathbf{B}}{\partial \mathbf{A}} = \left(\mathbf{I}_n \otimes \mathbf{C}_{qm} \otimes \mathbf{I}_p\right)(\mathbf{I}_{mn} \otimes \mathrm{vec}(\mathbf{B}))$	$\frac{\partial \mathbf{D}}{\partial \mathbf{X}} = \mathbf{T}_n \frac{\partial \mathbf{d}}{\partial \mathbf{X}}$
$\frac{\partial \mathbf{A} \otimes \mathbf{B}}{\partial \mathbf{B}} = \left(\mathbf{I}_n \otimes \mathbf{C}_{qm} \otimes \mathbf{I}_p\right)\left(\mathrm{vec}(\mathbf{A}) \otimes \mathbf{I}_{pq}\right)$	$\frac{\partial \mathrm{vec}(\mathbf{A})}{\partial \mathbf{A}} = \mathbf{I}_{mn}$

The formulas in the first column can be found in Fackler [4]. The formulas in the second column can be obtained as follows:

- $$\frac{\partial \mathrm{tr}(\mathbf{A})}{\partial \mathbf{X}} = \frac{\partial \mathrm{tr}(\mathbf{A})}{\partial \mathbf{A}}\frac{\partial \mathbf{A}}{\partial \mathbf{X}} = \mathrm{vec}(\mathbf{I}_n)'\frac{\partial \mathbf{A}}{\partial \mathbf{X}}$$
- $$\frac{\partial \mathbf{A}}{\partial \mathbf{A}} = \frac{\partial \mathrm{vec}(\mathbf{A})}{\partial \mathrm{vec}(\mathbf{A})} = \mathbf{I}_{mn}$$
  $$\frac{\partial \mathbf{A}'}{\partial \mathbf{A}} = \frac{\partial \mathrm{vec}(\mathbf{A}')}{\partial \mathrm{vec}(\mathbf{A})} = \frac{\partial \mathbf{C}_{mn}\mathrm{vec}(\mathbf{A})}{\partial \mathrm{vec}(\mathbf{A})} = (\mathbf{I}_1 \otimes \mathbf{C}_{mn})\mathbf{I}_{mn} = \mathbf{C}_{mn}$$
- entry (i,j) of $\mathbf{A} * \mathbf{B}$ is equal to $a_{ij}b_{ij}$, where both a_{ij} and b_{ij} are functions of $\mathbf{X}$. Therefore we can apply the rule to differentiate a product to each entry of $\mathbf{A} * \mathbf{B}$. The derivative of $\mathbf{A} * \mathbf{B}$ with respect to $\mathbf{X}$ when $\mathbf{B}$ is constant is equal to $\mathrm{diag}_{\mathrm{M}}(\mathrm{vec}(\mathbf{B}))\frac{\partial \mathbf{A}}{\partial \mathbf{X}}$, and the derivative of $\mathbf{A} * \mathbf{B}$ with respect to $\mathbf{X}$ when $\mathbf{A}$ is constant is equal to $\mathrm{diag}_{\mathrm{M}}(\mathrm{vec}(\mathbf{A}))\frac{\partial \mathbf{B}}{\partial \mathbf{X}}$.
- $\frac{\partial \mathbf{D}}{\partial \mathbf{d}} = \frac{\partial \mathrm{vec}(\mathbf{D})}{\partial \mathbf{d}}$ where $\mathrm{vec}(\mathbf{D})$ is the $n^2 \times 1$ vector $[d_1 0 \cdots 0 | \cdots | 0 \cdots 0 d_n]'$. The derivative of $\mathrm{vec}(\mathbf{D})$ with respect to d_i is the zero vector except for entry $(i-1)n+i$ where it is 1. Collecting all these derivatives side by side in a $n^2 \times n$ matrix gives $\mathbf{T}_n$.
- $\frac{\partial \mathbf{D}}{\partial \mathbf{X}} = \frac{\partial \mathbf{D}}{\partial \mathbf{d}}\frac{\partial \mathbf{d}}{\partial \mathbf{X}} = \mathbf{T}_n \frac{\partial \mathbf{d}}{\partial \mathbf{X}}$
- $\frac{\partial \mathrm{vec}(\mathbf{A})}{\partial \mathbf{A}} = \frac{\partial \mathrm{vec}(\mathbf{A})}{\partial \mathrm{vec}(\mathbf{A})} = \mathbf{I}_{mn}$

Appendix B

Notation: $\underline{\mathbf{M}}$ is a $p \times q \times m$ array with $p \times q$ frontal slices $\mathbf{M}_i$ $(i = 1, \ldots, m)$, $\mathbf{X}$ is a $p \times r$ matrix, $\mathbf{Y}$ is a $q \times r$ matrix. $\mathbf{D}_i$ is the diagonal matrix defined by Eq. (3.2).

Table B1 summarizes the expressions of functions $\mathbf{F}_1 - \mathbf{F}_8$. Also, the partial derivatives that are relevant for the paper are presented.

Table B1
Functions $\mathbf{F}_1$ to $\mathbf{F}_8$ and some of their partial derivatives.

Function	Derivative
$\mathbf{F}_1 = (\mathbf{X}'\mathbf{X} * \mathbf{Y}'\mathbf{Y})^{-1}$	In Case I ($\mathbf{X,Y}$ constrained to hold unit length columns): $\frac{\partial \mathbf{F}_1}{\partial \mathbf{X}} = -(\mathbf{F}_1 \otimes \mathbf{F}_1)\mathrm{diag}_{\mathrm{M}}(\mathrm{vec}(\mathbf{Y}'\mathbf{Y}))\mathbf{E}_r(\mathbf{I}_{r^2} + \mathbf{C}_{rr})(\mathbf{I}_r \otimes \mathbf{X}')$ $\frac{\partial \mathbf{F}_1}{\partial \mathbf{Y}} = -(\mathbf{F}_1 \otimes \mathbf{F}_1)\mathrm{diag}_{\mathrm{M}}(\mathrm{vec}(\mathbf{X}'\mathbf{X}))\mathbf{E}_r(\mathbf{I}_{r^2} + \mathbf{C}_{rr})(\mathbf{I}_r \otimes \mathbf{Y}')$ In Case II ($\mathbf{X,Y}$ constrained by orthonormality): $\frac{\partial \mathbf{F}_1}{\partial \mathbf{X}} = \mathbf{0}_{r^2,pr}; \frac{\partial \mathbf{F}_1}{\partial \mathbf{Y}} = \mathbf{0}_{r^2,qr}$
$\mathbf{F}_2^{(i)} = (\mathrm{diag}_{\mathrm{V}}(\mathbf{X}'\mathbf{M}_i\mathbf{Y})' \otimes \mathbf{I}_r)$	$\frac{\partial \mathbf{F}_2^{(i)}}{\partial \mathbf{X}} = (\mathbf{I}_r \otimes \mathrm{vec}(\mathbf{I}_r))\mathbf{T}'_r(\mathbf{I}_r \otimes \mathbf{Y}'\mathbf{M}_i')$ $\frac{\partial \mathbf{F}_2^{(i)}}{\partial \mathbf{Y}} = (\mathbf{I}_r \otimes \mathrm{vec}(\mathbf{I}_r))\mathbf{T}'_r(\mathbf{I}_r \otimes \mathbf{X}'\mathbf{M}_i)$
$\mathbf{F}_3 = \mathbf{F}_1 \otimes \mathbf{F}_1$	$\frac{\partial \mathbf{F}_3}{\partial \mathbf{X}} = (\mathbf{I}_r \otimes \mathbf{C}_{rr} \otimes \mathbf{I}_r)(\mathbf{I}_{r^4} + \mathbf{C}_{r^2r^2})(\mathbf{I}_{r^2} \otimes \mathrm{vec}(\mathbf{F}_1))\frac{\partial \mathbf{F}_1}{\partial \mathbf{X}}$ $\frac{\partial \mathbf{F}_3}{\partial \mathbf{Y}} = (\mathbf{I}_r \otimes \mathbf{C}_{rr} \otimes \mathbf{I}_r)(\mathbf{I}_{r^4} + \mathbf{C}_{r^2r^2})(\mathbf{I}_{r^2} \otimes \mathrm{vec}(\mathbf{F}_1))\frac{\partial \mathbf{F}_1}{\partial \mathbf{Y}}$
$\mathbf{F}_4^{(i)} = \mathbf{F}_2^{(i)}\mathbf{F}_3$	$\frac{\partial \mathbf{F}_4^{(i)}}{\partial \mathbf{X}} = (\mathbf{F}'_3 \otimes \mathbf{I}_r)\frac{\partial \mathbf{F}_2^{(i)}}{\partial \mathbf{X}} + \left(\mathbf{I}_{r^2} \otimes \mathbf{F}_2^{(i)}\right)\frac{\partial \mathbf{F}_3}{\partial \mathbf{X}}$ $\frac{\partial \mathbf{F}_4^{(i)}}{\partial \mathbf{Y}} = (\mathbf{F}'_3 \otimes \mathbf{I}_r)\frac{\partial \mathbf{F}_2^{(i)}}{\partial \mathbf{Y}} + \left(\mathbf{I}_{r^2} \otimes \mathbf{F}_2^{(i)}\right)\frac{\partial \mathbf{F}_3}{\partial \mathbf{Y}}$
$\mathbf{F}_5 = \mathrm{diag}_{\mathrm{M}}(\mathrm{vec}(\mathbf{Y}'\mathbf{Y}))$	In Case I ($\mathbf{X,Y}$ constrained to hold unit length columns): $\frac{\partial \mathbf{F}_5}{\partial \mathbf{Y}} = \mathbf{T}_{r^2}\mathbf{E}_r(\mathbf{I}_{r^2} + \mathbf{C}_{rr})(\mathbf{I}_r \otimes \mathbf{Y}')$ In Case II ($\mathbf{X,Y}$ constrained by orthonormality): $\frac{\partial \mathbf{F}_5}{\partial \mathbf{Y}} = \mathbf{0}_{r^4,qr}$
$\mathbf{F}_6^{(i)} = \mathbf{F}_4^{(i)}\mathbf{F}_5$	$\frac{\partial \mathbf{F}_6^{(i)}}{\partial \mathbf{Y}} = (\mathbf{F}'_5 \otimes \mathbf{I}_r)\frac{\partial \mathbf{F}_4^{(i)}}{\partial \mathbf{Y}} + \left(\mathbf{I}_{r^2} \otimes \mathbf{F}_4^{(i)}\right)\frac{\partial \mathbf{F}_5}{\partial \mathbf{Y}}$
$\mathbf{F}_7^{(i)} = \mathbf{I}_r \otimes \mathbf{Y}'\mathbf{M}'_i$	$\frac{\partial \mathbf{F}_7^{(i)}}{\partial \mathbf{Y}} = \left(\mathbf{I}_r \otimes \mathbf{C}_{pr} \otimes \mathbf{I}_r\right)\left(\mathrm{vec}(\mathbf{I}_r) \otimes \mathbf{I}_{pr}\right)(\mathbf{M}_i \otimes \mathbf{I}_r)\mathbf{C}_{qr}$
$\mathbf{F}_8^{(i)} = \mathbf{D}_i \otimes \mathbf{I}_p$	$\frac{\partial \mathbf{F}_8^{(i)}}{\partial \mathbf{Y}} = \left(\mathbf{I}_r \otimes \mathbf{C}_{pr} \otimes \mathbf{I}_p\right)\left(\mathbf{I}_{r^2} \otimes \mathrm{vec}(\mathbf{I}_p)\right)\mathbf{T}_r \frac{\partial \mathbf{d}_i}{\partial \mathbf{Y}}$

Appendix C

Notation: $\underline{\mathbf{S}}$ is a $p \times p \times m$ array of symmetric frontal slices $\mathbf{S}_i$ $(i = 1, \ldots, m)$, $\mathbf{X}$ and $\mathbf{Y}$ are $p \times r$ matrices. $\tilde{\mathbf{D}}_i$ is the diagonal matrix defined by Eq. (4.2).

Table C1
Functions $\mathbf{G}_1$ to $\mathbf{G}_{16}$ and some of their partial derivatives.

Function	Derivative
$\mathbf{G}_1^{(i)} = \mathbf{X}'\mathbf{S}_i\mathbf{X}\tilde{\mathbf{D}}_i$	$\frac{\partial \mathbf{G}_1^{(i)}}{\partial \mathbf{X}} = \left(\tilde{\mathbf{D}}_i\mathbf{X}' \otimes \mathbf{I}_r\right)(\mathbf{S}_i \otimes \mathbf{I}_r)\mathbf{C}_{pr} + (\mathbf{I}_r \otimes \mathbf{X}'\mathbf{S}_i)\left(\left(\tilde{\mathbf{D}}_i \otimes \mathbf{I}_p\right) + (\mathbf{I}_r \otimes \mathbf{X})\mathbf{T}_r \frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}}\right)$
$\mathbf{G}_2^{(i)} = \tilde{\mathbf{D}}_i\mathbf{X}' \otimes \mathbf{I}_r$	$\frac{\partial \mathbf{G}_2^{(i)}}{\partial \mathbf{X}} = \left(\mathbf{I}_p \otimes \mathbf{C}_{rr} \otimes \mathbf{I}_r\right)\left(\mathbf{I}_{pr} \otimes \mathrm{vec}(\mathbf{I}_r)\right) \cdot \left((\mathbf{X} \otimes \mathbf{I}_r)\mathbf{T}_r \frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}} + \left(\mathbf{I}_p \otimes \tilde{\mathbf{D}}_i\right)\mathbf{C}_{pr}\right)$
$\mathbf{G}_3^{(i)} = \mathbf{I}_r \otimes \mathbf{X}'\mathbf{S}_i$	$\frac{\partial \mathbf{G}_3^{(i)}}{\partial \mathbf{X}} = \left(\mathbf{I}_r \otimes \mathbf{C}_{pr} \otimes \mathbf{I}_r\right)\left(\mathrm{vec}(\mathbf{I}_r) \otimes \mathbf{I}_{pr}\right)(\mathbf{S}_i \otimes \mathbf{I}_r)\mathbf{C}_{pr}$
$\mathbf{G}_4^{(i)} = \tilde{\mathbf{D}}_i \otimes \mathbf{I}_p$	$\frac{\partial \mathbf{G}_4^{(i)}}{\partial \mathbf{X}} = \left(\mathbf{I}_r \otimes \mathbf{C}_{pr} \otimes \mathbf{I}_p\right)\left(\mathbf{I}_{r^2} \otimes \mathrm{vec}(\mathbf{I}_p)\right)\mathbf{T}_r \frac{\partial \tilde{\mathbf{d}}_i}{\partial \mathbf{X}}$
$\mathbf{G}_5 = \mathbf{I}_r \otimes \mathbf{X}$	$\frac{\partial \mathbf{G}_5}{\partial \mathbf{X}} = \left(\mathbf{I}_r \otimes \mathbf{C}_{rr} \otimes \mathbf{I}_p\right)\left(\mathrm{vec}(\mathbf{I}_r) \otimes \mathbf{I}_{pr}\right)$
$\mathbf{G}_6^{(i)} = \mathbf{G}_3^{(i)}\mathbf{G}_4^{(i)}$	$\frac{\partial \mathbf{G}_6^{(i)}}{\partial \mathbf{X}} = \left(\left(\mathbf{G}_4^{(i)}\right)' \otimes \mathbf{I}_{r^2}\right)\frac{\partial \mathbf{G}_3^{(i)}}{\partial \mathbf{X}} + \left(\mathbf{I}_{pr} \otimes \mathbf{G}_3^{(i)}\right)\frac{\partial \mathbf{G}_4^{(i)}}{\partial \mathbf{X}}$
$\mathbf{G}_7^{(i)} = \mathbf{G}_3^{(i)}\mathbf{G}_5$	$\frac{\partial \mathbf{G}_7^{(i)}}{\partial \mathbf{X}} = \left(\mathbf{G}'_5 \otimes \mathbf{I}_{r^2}\right)\frac{\partial \mathbf{G}_3^{(i)}}{\partial \mathbf{X}} + \left(\mathbf{I}_{r^2} \otimes \mathbf{G}_3^{(i)}\right)\frac{\partial \mathbf{G}_5}{\partial \mathbf{X}}$
$\mathbf{G}_8 = (\mathbf{X}'\mathbf{X} * \mathbf{X}'\mathbf{X})^{-1}$	In Case I ($\mathbf{X,Y}$ constrained to hold unit length columns): $\frac{\partial \mathbf{G}_8}{\partial \mathbf{X}} = -2(\mathbf{G}_8 \otimes \mathbf{G}_8)\mathrm{diag}_{\mathrm{M}}(\mathrm{vec}(\mathbf{X}'\mathbf{X}))\mathbf{E}_r(\mathbf{I}_{r^2} + \mathbf{C}_{rr})\left(\mathbf{I}_r \otimes \mathbf{X}'\right)$ In Case II ($\mathbf{X,Y}$ constrained by orthonormality): $\frac{\partial \mathbf{G}_8}{\partial \mathbf{X}} = \mathbf{0}_{r^2,pr}$
$\mathbf{G}_9^{(i)} = \mathrm{diag}_{\mathrm{V}}(\mathbf{X}'\mathbf{S}_i\mathbf{X})$	$\frac{\partial \mathbf{G}_9^{(i)}}{\partial \mathbf{X}} = \mathbf{T}'_r\left(\left(\mathbf{X}'\mathbf{S}_i \otimes \mathbf{I}_r\right)\mathbf{C}_{pr} + \left(\mathbf{I}_r \otimes \mathbf{X}'\right)(\mathbf{I}_r \otimes \mathbf{S}_i)\right)$

(continued on next page)

Table C1 (*continued*)

Function	Derivative
$\mathbf{G}_{10}^{(i)} = -((\mathbf{G}_9^{(i)})' \otimes \mathbf{I}_r)$	$\frac{\partial \mathbf{G}_{10}^{(i)}}{\partial \mathbf{X}} = -(\mathbf{I}_r \otimes \text{vec}(\mathbf{I}_r))\mathbf{T}'_r\left(\left(\mathbf{X}'\mathbf{S}_i \otimes \mathbf{I}_r\right)\mathbf{C}_{pr} + \left(\mathbf{I}_r \otimes \mathbf{X}'\right)(\mathbf{I}_r \otimes \mathbf{S}_i)\right)$
$\mathbf{G}_{11} = \mathbf{G}_8 \otimes \mathbf{G}_8$	$\frac{\partial \mathbf{G}_{11}}{\partial \mathbf{X}} = (\mathbf{I}_r \otimes \mathbf{C}_{rr} \otimes \mathbf{I}_r)(\mathbf{I}_{r^4} + \mathbf{C}_{r^2 r^2})(\mathbf{I}_{r^2} \otimes \text{vec}(\mathbf{G}_8))\frac{\partial \mathbf{G}_8}{\partial \mathbf{X}}$
$\mathbf{G}_{12}^{(i)} = \mathbf{G}_{10}^{(i)}\mathbf{G}_{11}$	$\frac{\partial \mathbf{G}_{12}^{(i)}}{\partial \mathbf{X}} = \left(\mathbf{G}'_{11} \otimes \mathbf{I}_r\right)\frac{\partial \mathbf{G}_{10}^{(i)}}{\partial \mathbf{X}} + \left(\mathbf{I}_{r^2} \otimes \mathbf{G}_{10}^{(i)}\right)\frac{\partial \mathbf{G}_{11}}{\partial \mathbf{X}}$
$\mathbf{G}_{13} = \text{diag}_{\text{M}}(\text{vec}(\mathbf{X}'\mathbf{X}))$	In Case I (**X**,**Y** constrained to hold unit length columns): $\frac{\partial \mathbf{G}_{13}}{\partial \mathbf{X}} = \mathbf{T}_{r^2}\mathbf{E}_r(\mathbf{I}_{r^2} + \mathbf{C}_{rr})(\mathbf{I}_r \otimes \mathbf{X}')$ In Case II (**X**,**Y** constrained by orthonormality): $\frac{\partial \mathbf{G}_{13}}{\partial \mathbf{X}} = \mathbf{0}_{r^4,pr}$
$\mathbf{G}_{14}^{(i)} = \mathbf{G}_{12}^{(i)}\mathbf{G}_{13}$	$\frac{\partial \mathbf{G}_{14}^{(i)}}{\partial \mathbf{X}} = (\mathbf{G}'_{13} \otimes \mathbf{I}_r)\frac{\partial \mathbf{G}_{12}^{(i)}}{\partial \mathbf{X}} + \left(\mathbf{I}_{r^2} \otimes \mathbf{G}_{12}^{(i)}\right)\frac{\partial \mathbf{G}_{13}}{\partial \mathbf{X}}$
$\mathbf{G}_{15}^{(i)} = \mathbf{X}'\mathbf{S}_i \otimes \mathbf{I}_r$	$\frac{\partial \mathbf{G}_{15}^{(i)}}{\partial \mathbf{X}} = \left(\mathbf{I}_p \otimes \mathbf{C}_{rr} \otimes \mathbf{I}_r\right)\left(\mathbf{I}_{pr} \otimes \text{vec}(\mathbf{I}_r)\right)(\mathbf{S}_i \otimes \mathbf{I}_r)\mathbf{C}_{pr}$
$\mathbf{G}_{16}^{(i)} = \mathbf{G}_{15}^{(i)}\mathbf{C}_{pr} + \mathbf{G}'_5 (\mathbf{I}_r \otimes \mathbf{S}_i)$	$\frac{\partial \mathbf{G}_{16}^{(i)}}{\partial \mathbf{X}} = \left(\mathbf{C}'_{pr} \otimes \mathbf{I}_{r^2}\right)\frac{\partial \mathbf{G}_{15}^{(i)}}{\partial \mathbf{X}} + (\mathbf{I}_r \otimes \mathbf{S}_i \otimes \mathbf{I}_{r^2})\mathbf{C}_{pr,r^2}\frac{\partial \mathbf{G}_5}{\partial \mathbf{X}}$

Table C1 summarizes the expressions of functions $\mathbf{G}_1 - \mathbf{G}_{16}$, with the relevant partial derivatives.

References

[1] M. Bennani Dosse, J.M.F. Ten Berge, The assumption of proportional components when Candecomp is applied to symmetric matrices in the context of Indscal, Psychometrika 73 (2008) 303–307.

[2] M. Bennani Dosse, J.M.F. Ten Berge, J. Tendeiro, Some new results on orthogonally constrained Candecomp, submitted for publication.

[3] J.D. Carroll, J.J. Chang, Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart–Young decomposition, Psychometrika 35 (1970) 283–319.

[4] P.L. Fackler, Notes on Matrix Calculus, , 2009 Retrieved on 3 February 2009 from http://www4.ncsu.edu/~pfackler/MatCalc.pdf.

[5] R.A. Harshman, Foundations of the Parafac procedure: models and conditions for an "explanatory" multimodal factor analysis, UCLA Working Papers in Phonetics, 16, 1970, pp. 1–84.

[6] R.A. Harshman, M.E. Lundy, Data preprocessing and the extended PARAFAC model, in: H.G. Law, C.W. Snyder, J.A. Hattie, R.P. McDonald (Eds.), Research Methods for Multimode Data Analysis, Praeger, New York, 1984, pp. 216–284.

[7] J.B. Kruskal, Three-way arrays: rank and uniqueness of trilinear decompositions, with applications to arithmetic complexity and statistics, Linear Algebra and its Applications 18 (1977) 95–138.

[8] J.B. Kruskal, R.A. Harshman, M.E. Lundy, Some relationships between Tucker's three-mode factor analysis and PARAFAC/CANDECOMP, Paper Presented at the Annual Meeting of the Psychometric Society, Los Angeles, 1983.

[9] J.B. Kruskal, R.A. Harshman, M.E. Lundy, Several mathematical relationships between PARAFAC-CANDECOMP and three-mode factor analysis, Paper Presented at the Annual Meeting of the Classification Society, St. John's, Newfoundland, 1985.

[10] J.R. Magnus, H. Neudecker, Matrix Differential Calculus with Applications in Statistics and Econometrics, 3rd EditionJohn Wiley & Sons, Chichester/New York, 2007.

[11] P. Paatero, A weighted non-negative least squares algorithm for three-way 'PARAFAC' factor analysis, Chemometrics and Intelligent Laboratory systems 38 (1997) 223–242.

[12] P. Paatero, The multilinear engine — a table-driven, least squares program for solving multlinear problems, including the n-way parallel factor analysis model, Journal of Computational and Graphical Statistics 8 (1999) 854–888.

[13] A. Stegeman, Degeneracy in Candecomp/Parafac explained for $p \times p \times 2$ arrays of rank $p+1$ or higher, Psychometrika 71 (2006) 483–501.

[14] A. Stegeman, Degeneracy in Candecomp/Parafac and Indscal explained for several three-sliced arrays with a two-valued typical rank, Psychometrika 72 (2007) 601–619.

[15] J.M.F. Ten Berge, A joint treatment of Varimax rotation and the problem of diagonalizing symmetric matrices simultaneously in the least-squares sense, Psychometrika 49 (1984) 347–358.

[16] J.M.F. Ten Berge, H.A.L. Kiers, Some clarifications of the Candecomp algorithm applied to Indscal, Psychometrika 56 (1991) 317–326.

[17] J.M.F. Ten Berge, H.A.L. Kiers, J. De Leeuw, Explicit CANDECOMP/PARAFAC solutions for a contrived $2 \times 2 \times 2$ array of rank three, Psychometrika 53 (1988) 579–583.

[18] J.M.F. Ten Berge, D.L. Knol, H.A.L. Kiers, A treatment of the Orthomax rotation family in terms of diagonalization, and a re-examination of a singular value approach to Varimax rotation, Computational Statistics Quarterly 3 (1988) 207–217.

Samenvatting (Summary in Dutch)

Onderzoekers hebben vaak te maken met gegevens die boven de gebruikelijke "gegevens × variabelenstructuur" uitgaan. Soms bestaan de gegevens bijvoorbeeld uit matrices die op verschillende tijdstippen zijn verzameld. Gegevens met een dergelijke structuur worden *meerweggegevens* genoemd. Ze roepen de vraag op naar meerwegtechnieken.

Meerwegtechnieken vormen een natuurlijke uitbreiding van twee-wegtechnieken. De eerste bijdrage stamt uit 1927 (Hitchcock [29, 30]). In 1966 kwam Tucker [114] met een andere driewegcomponentenanalyse. In 1970 hebben Carroll and Chang [11] en Harshman [24] de methode van Hitchcock herontdekt (Hoofdstuk 2). Sindsdien is er veel nieuws bijgekomen: veel wiskundige eigenschappen van meerwegmodellen zijn aan het licht gebracht en tal van nieuwe modellen zijn geïntroduceerd.

In dit proefschrift ligt de nadruk op driewegmodellen. Een van onze doelstellingen was een beknopt overzicht te geven van enkele lineair-algebraïsche eigenschappen van driewegarrays. Ook hebben we eigenschappen bestudeerd van diverse statistische modellen voor de analyse van drieweggegevens. De eerste vijf hoofdstukken van dit proefschrift bestaan voornamelijk uit het bij elkaar brengen van resultaten die verspreid in de literatuur voorkomen. In Hoofdstuk 6 en 7 worden nieuwe resultaten naar voren gebracht.

In Hoofdstuk 1 worden grondbegrippen inzake matrices en driewegarrays geïntroduceerd. Doel is de lezers voor te bereiden op de latere hoofdstukken. Het accent ligt op ontbindingstechnieken voor matrices, zoals de eigenontbinding, de singuliere waarden-ontbinding en Principale Componentenanalyse, en op het begrip driewegarray.

Hoofdstuk 2 bevat een bespreking van driewegmodellen die nader onderzocht worden. Het gaat om 3PCA (Tucker [114]), het CP-model (Carroll and Chang [11] en Harshman [24]) en het INDSCAL-model (Carroll and Chang [11]). Voor elk model wordt een aantal wiskundige formuleringen gegeven.

Hoofdstuk 3 gaat over uniciteit. Uitgelegd wordt waarom het 3PCA-model niet geïdentificeerd is, terwijl het CP-model dat normaliter wel is. Diverse uniciteitvoorwaarden uit de literatuur worden behandeld. Daaraan worden nieuwe verkorte bewijzen voor Stelling 2 (p. 46) en Stelling 3 (p.47) toegevoegd.

In Hoofdstuk 4 worden gedegenereerde oplossingen behandeld. Uitgelegd wordt waarom gedegenereerdheid problematisch is in de context van het CP-model, en hoe het mogelijk is dat het CP-algoritme soms tot gedegenereerde oplossingen komt. Verder worden manieren besproken om gedegenereerde oplossingen te vermijden.

Het begrip “simplicity” (eenvoud) staat centraal in Hoofdstuk 5. De rotatievrijheid in 3PCA maakt het mogelijk simplicity in de kern en/of in de componentmatrices te bereiken. Transformatie van 3PCA-oplossingen naar eenvoudige vorm kan de interpretatie van de oplossing vereenvoudigen. Maar dergelijke procedures zijn ook wiskundig interessant. Met name de “typical rank” (waarschijnlijke rang) van array-formaten is veel gemakkelijker te bepalen wanneer gebruik wordt gemaakt van “simplicity transformations”.

In Hoofdstuk 6 worden nieuwe resultaten gepresenteerd inzake simplicity van driewegarrays, met name inzake arrays die uit symmetrische matrices bestaan. Voor dergelijke arrays zijn niet eerder simplicity-resultaten gevonden. Er wordt een uitgebreide analyse gegeven voor arrays waarin de symmetrische matrices van orde 2×2, 3×3 en 4×4 zijn. Er worden voorbeelden gegeven waarin simplicity-transformaties de bepaling van waarschijnlijke rang vereenvoudigen. Ook wordt aandacht besteed aan het bepalen van “maximal simplicity”.

In Hoofdstuk 7 worden eerste en tweede orde afgeleiden van optimalisatiefuncties voor CP en Indscal gegeven. Doel is een gereedschap te ontwikkelen waarmee we zadelpunten kunnen herkennen. Twee typen randvoorwaarden zijn daarbij in acht genomen: kolommen van lengte 1 of orthonormaliteit, in twee van de drie com-

ponentenmatrices. Enkele numerieke problemen die daarbij optreden (verwant aan gedegenereerdheid) worden besproken. Ter illustratie worden drie toepassingen behandeld.

Acknowledgements

This section has been, by far, the most enjoyable to write. While organizing my thoughts before writing these words, I realized that there are several people that had a direct or indirect influence on this thesis. I will try not to forget anyone in the following paragraphs.

I suppose that first things come first. And this means that the first person that I want to thank is Jos ten Berge. It is not easy for me to accurately describe how much Jos influenced the development of this thesis in particular and the whole project in general. I have tried to give a more proper description of my thoughts by dividing Jos into several kinds of Jos's that I met during the last 4 years.

There is the "scientific" Jos: he is the ultimate researcher, in the sense that he is always ready to dig into anything for which there is no available answer. I met no exception to this rule during the 4 years of cooperation between us. His knowledge and expertise still amaze me now, and it makes me realize that I still have a very long way to go myself.

There is also the "supervisor" Jos: in the vast majority of times he helped me choosing the best direction to proceed with my work. In the few cases that he did not do so, either I blame myself or maybe it was because walking in the dark is just what research is all about, even if you are being supervised.

There is also the "man" Jos, who gave me enough time to gather myself together in times of suffering, and who understood that I am not a Dutch person (with all that being a non-Dutch person implies). There were several "meetings" in his office in which we did everything but work. Those moments I do not forget, as they were only possible because Jos is how he is. Jos is also the guy who took a Sunday morning to help me carry a lot of bags and boxes when I moved into a better room. He is the one who borrowed me his seasonal card of FC Groningen several times, so I got to see several Eredivisie matches for free. These are only some of the things that no one can read in the theorems and proofs of this thesis, but that decisively (and indirectly) contributed in the writing of them.

Finally, there is also the “child” Jos. The one who gets over-enthusiastic about his beloved FC Groningen. The one who taught me how a smoker can still go out and have a nice *cappuccino* downtown. The one who can find the most amusing joke in the most normal aspects of life. At moments Jos made me feel that he was 6.5 years old instead of 65. This part of Jos taught me more than he can ever imagine, and I will treasure it forever.

Another person that I wish to thank for is Henk Kiers. Henk was the first person that I ever met in Groningen, at the train station back in February 2006. He went to big efforts to make me feel well in the Netherlands. These efforts included two trips with his wife to the flower fields sometime in April 2007 and 2008, which I really enjoyed. All my experience during the PhD was better due to these acts from a man who already had so much to do but still found gaps in his agenda to stroll around with some portuguese dude.

Henk always read my material, gave me loads of suggestions (and corrected a lot of mistakes with which I adore to populate my manuscripts), and made me feel extra safe with it. This thesis has a somehow different flavour since it was also developed under the expertise of Henk. To him I also wish to thank very much.

I want to leave a general word of appreciation to my department. During these four years I attended many research meetings, which directly or indirectly affected the way this thesis turned out. I had the opportunity to present some of my results during those meetings, which helped me a lot. The writing of this thesis was positively influenced by those meetings. I thank everyone for it.

From the organizational point of view there are several persons or entities that I wish to thank. These include Hanny and Greetje, who always put effort to help me solving any problem that came about, while being extremely nice at the same time. In a more general way I wish to thank all those who eventually helped me somehow: these include P&O, informatic technicians, portiers, and workers at the canteen (my wallet still hates you, though).

As for friends (Groningers!) who were closer to me, there are a few names that I want to mention. These include Rink (my best man!, that says it all) and Arnout (my first Dutch friends ever, up to this very day) and their partners Jana and Jantine (and baby Sarah!), Radek (Groningen is not the same since he left…), Zhenya (and little Sasha) and Aysa, Anne Fetsje (who, sadly for me, moved to an office far far away, across the garden), Fabia, Inês, Berfu, Tejas, Sayaka, and the great Jan Pieter.

From Portugal there are also some names that I need to mention. They are part of me, and therefore of this book. "Vocês sabem de quem eu estou a falar!": João & Titinha (and Luisinha!), João Pereira (Boda!!) & Xana, Sílvia & Jorge, Alberto & Carla (e as suas duas princesas Rita e Joana), and Sofia (por falar em princesas).

I wish to send a gigantic hug to both my parents António e Maria da Conceição. I wouldn't be here without them. A part of this is dedicated to them. I also want to send a big kiss to my sister, who I haven't seen since before the PhD started.

I send a special kiss to my new Japanese family: Kotaro-san (幸太郎), Tomoko-san (智子), Nobuo-san (信男) and Shun-san (峻).
ありがとう！

Finally, I wish to send a gigantic kiss to my now wife Rei Monden (Tendeiro). When I met her this project was half way. Since that moment she has always been by my side, in ways that I cannot fully describe with words. What the hell, I did marry her, right?, I don't need to say anything more. God bless you, sweety.