UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

Fast Matrix Multiplication

Dissertation presented by
Guillaume Berger

for obtaining the Master's degree in
Mathematical Engineering

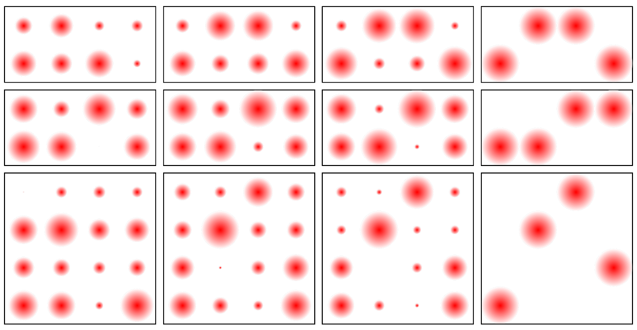
Supervisors
Pierre-Antoine Absil, Lieven De Lathauwer

Readers
Laurent Jacques, Marc Van Barel

Academic year 2016–2017

Fast Matrix Multiplication

Guillaume Berger



Dissertation committee:

Pierre-Antoine Absil (UCL, advisor)

Lieven De Lathauwer (KU Leuven, advisor)

Laurent Jacques (UCL, external member of jury)

Marc Van Barel (KU Leuven, advisor)

June 9, 2017

Acknowledgements

This thesis is the synthesis of six months of research under the supervision of Prof. Pierre-Antoine Absil, Prof. Lieven De Lathauwer and Prof. Marc Van Barel. I would like to sincerely thank them for their insightful comments, fruitful ideas and enthusiastic support all along that period.

I am also grateful to Prof. Yurii Nesterov for his help concerning non-smooth optimization methods.

Table of contents

1 Introduction and outline **9**

2 State of the art **13**

3 The problem of fast matrix multiplication **15**

3.1 Basic definitions and notations . . . 15

3.2 Presentation of the problem . . . 17

3.2.1 Complexity of matrix multiplication . . . 17

3.2.2 The matrix multiplication tensor . . . 18

3.2.3 Relation between rank and complexity . . . 18

3.3 Sparse discrete decompositions . . . 21

3.3.1 Sparse solutions $\rightarrow$ stability + complexity . . . 21

3.3.2 Discrete solutions $\rightarrow$ exactness + complexity . . . 22

3.4 Invariant transformations . . . 22

3.4.1 Elementary transformations . . . 23

3.4.2 "Trace-like" transformations . . . 23

4 Computation of sparse discrete decompositions **26**

4.1 A procedure to compute SD-sol's . . . 26

4.1.1 A non-smooth optimization problem . . . 27

4.1.2 Gradient method for composite functions . . . 28

4.1.3 Practical use of the gradient method . . . 33

4.2 Examples of computation of SD-sol's . . . 34

4.2.1 Sparse discrete decompositions of small tensors . . . 34

4.2.2 Sparse discrete decompositions of larger tensors . . . 37

4.2.3 Comparison with two state-of-the-art methods . . . 38

5 Discretizing decompositions through inv-transformations **39**

5.1 Properties of discretizable decompositions . . . 39

5.1.1 Motivation $\leftarrow$ the two-steps approach . . . 39

5.1.2 Characteristic polynomials of a F-PD . . . 40

5.2 Distribution of discretizable decompositions . . . 41

5.2.1 Statistics on the characteristic polynomials . . . 42

5.2.2 Implementation $\rightarrow$ getting rid of T-perm 44
5.2.3 Case-by-case analysis and conclusions 45

6 Analysis of the inv-equivalence classes of decompositions 49
6.1 Computing the inv-transformations joining two F-PD's 49
6.1.1 Preliminaries $\rightarrow$ the $\oplus$-rank of a matrix 49
6.1.2 Computation (part 1) $\rightarrow$ T-scale + T-trace 53
6.1.3 Computation (part 2) $\rightarrow$ getting rid of T-perm 58
6.2 Distribution of the $\oplus$-ranks and inv-equivalence classes 60
6.2.1 Distribution of the $\oplus$-rank of the factor matrices 60
6.2.2 Distribution of the inv-equivalence classes of F-PD's 61

7 Conclusions 64

A Lipschitz-continuous gradient and quadratic upper bound 68

B Gradient of the F-terms polyadic decomposition error 70

C Tables and figures for the computation of SD-sol's 71

D Description of the Tichavský et al.'s LM-method 81

E Overview of the Matlab codes attached to the report 83

Chapter 1

Introduction and outline

Because matrix multiplication is such a central operation in many numerical computations, it is worth investing work in making matrix multiplication algorithms efficient. The "straightforward" way to multiply two (n, n) matrices costs $\mathcal{O}(n^3)$ arithmetic operations $\{+, -, \times, \div\}$. In particular, multiplying two $(2, 2)$ matrices with the "straightforward" method requires 8 scalar multiplications. However, the arithmetic operations can be grouped cleverly to reduce the work to 7 multiplications only (the number of required multiplications decreases but we need more additions; nevertheless, as we will see later, we do not have to worry about additions when we address the problem of asymptotic complexity of matrix multiplication). Doing this recursively, we can reduce the cost for the multiplication of two (n, n) matrices to $\mathcal{O}(n^{2.81})$ operations (in fact, the recursion is precisely the reason why we want to reduce the number of multiplications and do not worry about the number of additions). More substantial computational savings may be obtained by starting from the reduction that can be achieved for the multiplication of $(3, 3)$ or $(4, 4)$ matrices for instance. The reduction of the complexity may actually become so significant that a new architecture for large matrix multiplication emerges. Essential is first that we find inexpensive schemes for the multiplication of relatively small matrices.

The multiplication of (m, p) matrices by (p, n) matrices can be represented by a third-order tensor. Finding inexpensive schemes for the multiplication of such matrices amounts to finding decompositions of the associated tensor. The decomposition consists of a sum of simpler tensors called "rank-1 tensors". The minimal number of rank-1 terms necessary to decompose a tensor is its rank. In the case of matrix multiplication, the rank of the associated tensor is equal to the smallest number of active multiplications needed to compute the matrix product (by "active multiplication", we mean multiplication between factors that "both depend on the entries of the matrices"; multiplications of those entries with scalar coefficients are not considered as active multiplications). We will see that the number of active multiplications will give us directly an upper bound for the asymptotic complexity of matrix multiplication. In other words, determining the rank of the associated tensor allows us to find an exponent ω such that the asymptotic complexity for the multiplication of (n, n) matrices is at most in $\mathcal{O}(n^{\omega})$ arithmetic operations.

Moreover, two decompositions of a given tensor might not have the same usefulness even if they consist of the same number of rank-1 terms. For the problem of matrix multiplication, decompositions with a lot of zero values and a small number of different values in the rank-1 terms are more conclusive and practically useful. Such decompositions are called “sparse” and “discrete” solutions. For instance, we may want that all the rank-1 term entries belong to $\{-1, 0, +1\}$ with the “0” most represented. However, the classical iterative processes for computing tensor decompositions do not lead in general to solutions of this kind.

In this thesis, we present a new algorithm/procedure to compute sparse discrete decompositions. This involves successive “solve’s” of a non-smooth constrained optimization problem. The idea is to induce the solution to be sparse using a regularization term involving the L^1-norm of the rank-1 terms and then recursively fix the most extremal values in the rank-1 terms to integer values. With this algorithm, we easily computed sparse decompositions with rank-1 terms having their entries in $\{-1, 0, +1\}$ for the multiplication of matrices with sizes up to $(3, 3)$.

We are also interested in the relations that exist between the different decompositions of a given tensor. Let us start with a small example. Suppose we have decomposed a tensor into the sum of rank-1 tensors. Then summing the rank-1 terms in a different order leaves the result unchanged. Therefore, permutation of the terms is called an “invariant transformation” of the decomposition and we say that the two decompositions are “equivalent up to permutation of the terms”. For matrix multiplication tensors, more subtle invariant transformations are possible. This raises questions like “is there a unique algorithm for computing the product of (m, p) matrices by (p, n) matrices with F active multiplications only?” and, otherwise, “given a matrix multiplication tensor, how many different decompositions with F terms exist such that two decompositions are not equivalent by invariant transformations?”.

We present an algorithm to decide whether two decompositions are equivalent by invariant transformations and, if they are equivalent, compute the invariant transformations that link them. The algorithm involves linear algebra and a novel strategy to get rid of the permutation equivalence without trying every possible permutation of the rank-1 terms, which would be prohibitive (for example, the decompositions we considered in this report for the multiplication of $(3, 3)$ matrices involve 23 rank-1 terms).

We are also interested in the “characteristic polynomials of the decompositions” (to be defined later) of matrix multiplication tensors. The genesis of this work was to answer the question “for a given matrix multiplication tensor, can we join any decomposition to a discrete decomposition with invariant transformations?”. Indeed, such decompositions (joinable to a discrete decomposition), which will be denoted as “discretizable” decompositions, would be interesting in what we call the “two-steps approach”, i.e. first compute a decomposition of the matrix multiplication tensor and then transform it into a sparse discrete decomposition with invariant transformations. In this thesis, we give a necessary condition for a decomposition to be discretizable. This condition involves the “characteris-

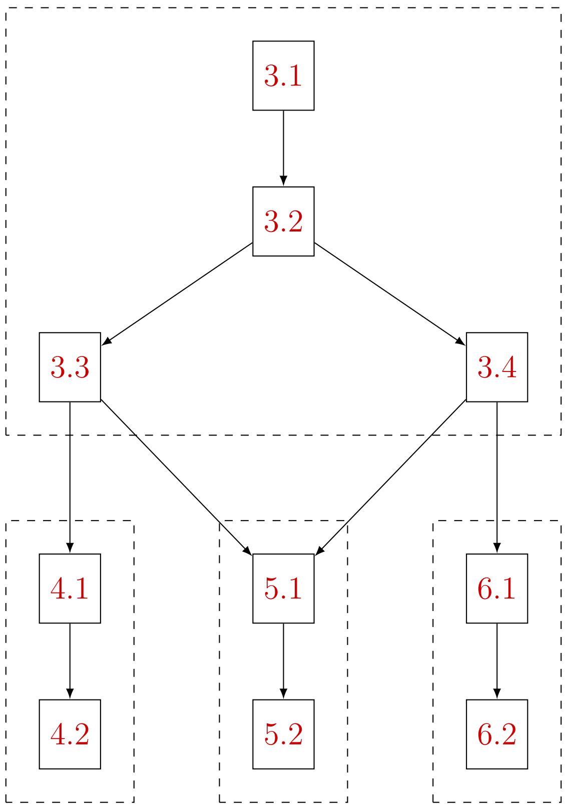


Figure 1.1 – Dependencies between sections.

tic polynomials of the decomposition". We will also see that, in practice, the "characteristic polynomials of the decompositions" of matrix multiplication tensors are not so spread as we might have expected.

The report is divided into seven chapters including this introduction and the final conclusions. We start with Chapter 2 giving a small history of the work that has been done in the field of matrix multiplication complexity and also present the main results obtained in recent years. Chapter 3 introduces the problem of "fast matrix multiplication" and matrix multiplication complexity. More precisely, in Sec. 3.1, we present the basic concepts and notations that will be necessary to understand the report. In Sec. 3.2, we provide a rigorous definition for the asymptotic complexity of matrix multiplication. We also show precisely how the problem is related to the problem of determining the rank of the associated tensor. In Sec. 3.3, we introduce the notion of sparse discrete solutions and explain briefly why they are more useful in practice. Invariant transformations are discussed in Sec. 3.4.

Chapter 4 deals with the computation of sparse discrete decompositions. In Sec. 4.1, we present the algorithm we have implemented to compute them. Decompositions obtained with the algorithm are presented in Sec. 4.2. Rather than simply presenting the results, we explain in detail how the algorithm can be used to compute sparse discrete solutions of matrix multiplication tensors up to the $(3, 3)$ case.

The purpose of Chapter 5 is to give keys for answering the question "for a given matrix multiplication tensor, can we join any decomposition to a discrete decomposition with invariant transformations?". Let us recall that a decomposition joinable to a discrete decomposition is called "discretizable". We explain, in Sec. 5.1, how we can characterize discretizable decompositions. In Sec. 5.2, we analyze the distribution of discretizable decompositions for tensors up to the $(3, 3)$ case: is every decomposition discretizable? If a decomposition is "chosen randomly" (in a sense to be defined later), is there any chance for it to be discretizable?

Finally, in Chapter 6, we are interested in the equivalence classes of decompositions through invariant transformations. We present, in Sec. 6.1, an algorithm for deciding whether two decompositions of a given matrix multiplication tensor are equivalent through invariant transformations. We will use this algorithm in Sec 6.2 to analyze the distribution of the equivalence classes among the different decompositions of a given matrix multiplication tensor.

The different sections appearing in Chapter 3 to Chapter 6 are linked together in multiple ways. To understand some section, it is sometimes necessary to have read before the sections on which this section depends. Hence, to help the reader, we have represented a schematic of the dependencies between those sections in Fig. 1.1.

We have implemented the algorithms presented in this report in Matlab. A zip file containing the codes can be downloaded from `https://drive.google.com/file/d/0B_BURxd4zWUIZjN4U2ZmTVQ1TjQ/view?usp=sharing`. In Appendix E, we also give an overview of the different functions and scripts we have written.

Chapter 2

State of the art

Although the problem of matrix multiplication complexity is quite old, only partial results are known so far. The work was initiated in 1969 when Strassen proposed [1] an algorithm to compute the product of $(2,2)$ matrices with 7 active multiplications only. Since then, the multiplication of $(2,2)$ matrices has been completely understood. The rank of the associated tensor is 7 so that Strassen's algorithm is optimal. It was proved by de Groote [2] that Strassen's algorithm is unique in the sense that every other algorithm for the multiplication of $(2,2)$ matrices involving 7 active multiplications can be shown to be equivalent to it through invariant transformations (c.f. Sec. 3.4).

For the $(3,3)$ case, an algorithm computing the product with 23 active multiplications was proposed in 1976 by Laderman [3]. This means that the rank of the associated tensor is at most 23. On the other hand, Bläser proved [4] in 2003 that the rank for the multiplication of $(3,3)$ matrices should be at least 19. The gap 19–23 has not been reduced since then. For multiplying two $(4,4)$ matrices, one can use Strassen's algorithm twice, and therefore the rank is at most 49. An algorithm involving 100 active multiplications was proposed in 1987 by Makarov [5] for the product of $(5,5)$ matrices. As far as we know, there is still no proof whether those algorithms are optimal or not.

Recent developments in numerical multi-linear algebra open the door for further improvements. The emergence of numerical algorithms for the decomposition of tensors induced a paradigm shift in the estimation of matrix multiplication complexity. Authors like Smirnov [6] and Tichavský et al. [7] proposed innovative approaches to the numerical computation of decompositions for matrix multiplication tensors. Nevertheless, the problem of decomposing a given tensor in rank-1 terms remains difficult. In particular, the numerous invariant transformations that can be applied on the rank-1 terms without changing the result are responsible for the existence of a multitude of "flat regions" (when we address the problem of rank determination as an optimization problem). Those will have a negative impact on the global and local convergence of the iterative processes that can be used for solving the problem.

The paper [8, p. 1388] of Ballard et al. gives a good overview of the practical algorithms that are available in 2016 for the multiplication of matrices with sizes going from $(2,2)$ by $(2,2)$ to sizes like $(3,6)$ by $(6,3)$. We give, in Table 2.1, the number of rank-1 terms

Multiplication	Number of multiplications with "straightforward" method	Number of rank-1 terms
$(1,2)$ by $(2,1)$	2	$F = 2$
$(2,1)$ by $(1,2)$	4	$F = 4$
$(2,2)$ by $(2,2)$	8	$F = 7$
$(2,3)$ by $(3,2)$	12	$F = 11$
$(3,2)$ by $(2,3)$	18	$F = 15$
$(3,3)$ by $(3,3)$	27	$F = 23$

Table 2.1 – Main cases treated in this report.

involved in such algorithms for the different cases we will treat in this report (for example, when we will want to apply our results and algorithms on practical cases). Note that the two first cases are not coming from [8] but are straightforward since it amounts to the multiplication of 2-elements column-vectors $\mathbf{x}$ and $\mathbf{y}$, namely $\mathbf{x}^\top\mathbf{y}$ (requires 2 active multiplications) and $\mathbf{x}\,\mathbf{y}^\top$ (requires 4 active multiplications). A proof that these two cases cannot be computed with less active multiplications can be found in [9, p. 216].

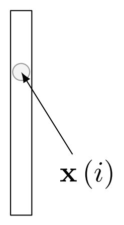
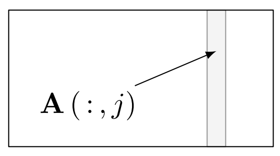
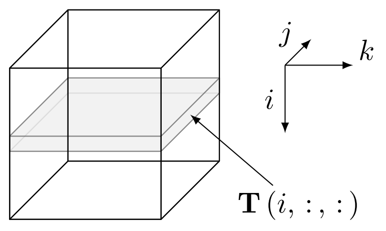
Chapter 3

The problem of fast matrix multiplication

3.1 Basic definitions and notations

There are, in the literature, numerous equivalent definitions of tensors. We limit ourselves to describe tensors as multi-way arrays. More formally, a *tensor* is an array with indices $(i_1, \ldots, i_N)$. In particular, vectors are tensors with one index and matrices have two indices. Tensors with three or more indices are called "higher-order" tensors. For the purpose of this work, we restrict ourselves to real-valued tensors up to third order. In the following, we refer to third-order tensors simply by calling them "tensors" contrasting with "vectors" and "matrices".

All along this report, we will use Matlab notation for the indexing of vectors, matrices and tensors. For instance, if $\mathbf{x}$ is a vector, then $\mathbf{x}(i)$ denotes its i-th element. In the same way, $\mathbf{A}(a\!:\!b\,,\,:)$ denotes the rows a to b of the matrix $\mathbf{A}$ while $\mathbf{T}(\,:\,, j, \,:\,)$ is the j-th "slice" along the second mode of the tensor $\mathbf{T}$. More precisely, $\mathbf{T}(\,:\,, j, \,:\,)$ is the matrix with row index i and column index k and such that $\mathbf{T}(\,:\,, j, \,:\,)(i,k) \coloneqq \mathbf{T}(i,j,k)$. Further examples of tensors and indexing and their graphical equivalent are represented below:

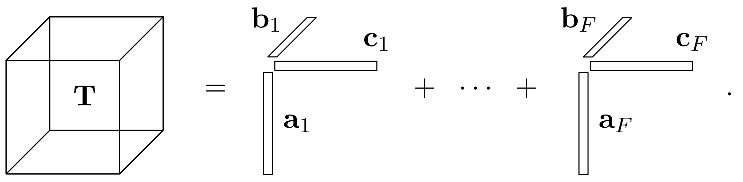
Vector	Matrix	Tensor
		

.

The concept of rank of a tensor will be central in the analysis of the asymptotic complexity of matrix multiplication. We define a *rank-at-most-*1 tensor to be a tensor that can be expressed as $\mathbf{T}(i,j,k) = \mathbf{a}(i) \cdot \mathbf{b}(j) \cdot \mathbf{c}(k)$ for some vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$. Now let $\mathbf{T}$ be any tensor, a F-terms *polyadic decomposition* of $\mathbf{T}$ is a decomposition of $\mathbf{T}$ as the sum of F rank-at-most-1 tensors:

$$\mathbf{T}(i,j,k) = \sum_{r=1}^{F} \mathbf{a}_r(i) \cdot \mathbf{b}_r(j) \cdot \mathbf{c}_r(k) \ .$$

Graphically, this is often represented as follows:



The *rank* of a non-zero tensor $\mathbf{T}$ is defined as the smallest F such that $\mathbf{T}$ admits a F-terms polyadic decomposition. By convention, the rank of a zero tensor is zero. The rank of a tensor is thus a generalization of the concept of rank for matrices. Let $\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r$ be the vectors involved in a F-terms polyadic decomposition of some tensor $\mathbf{T}$. Define the matrices $\mathbf{A} \coloneqq [\mathbf{a}_1, \ldots, \mathbf{a}_F]$, $\mathbf{B} \coloneqq [\mathbf{b}_1, \ldots, \mathbf{b}_F]$ and $\mathbf{C} \coloneqq [\mathbf{c}_1, \ldots, \mathbf{c}_F]$ and define the "polyadic decomposition operator" $[\![\cdot , \cdot , \cdot]\!]$ as follows:

$$[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] (i, j, k) \coloneqq \sum_{r=1}^{F} \mathbf{A}(i, r) \; \mathbf{B}(j, r) \; \mathbf{C}(k, r) \; .$$

From the definition of $\mathbf{A}, \mathbf{B}, \mathbf{C}$, it is clear that $\mathbf{T} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$. Reversely, if some matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ have F columns each and satisfy $\mathbf{T} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, then we say that we have a F-terms polyadic decomposition of $\mathbf{T}$ with *factor matrices* $\mathbf{A}, \mathbf{B}, \mathbf{C}$. With abuse of notation, we will often say more simply that "$[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is a F-PD of $\mathbf{T}$".

Let us remind the definition of two particular matrix products that will be useful in the following. First the *Kronecker product* of two matrices $\mathbf{A}$ and $\mathbf{B}$ with sizes (I_1, J_1) and (I_2, J_2) respectively is the $(I_1 I_2, J_1 J_2)$ matrix defined by

$$\mathbf{A} \otimes \mathbf{B} \coloneqq \begin{bmatrix} \mathbf{A}(1,1) \cdot \mathbf{B} & \mathbf{A}(1,2) \cdot \mathbf{B} & \cdots & \mathbf{A}(1,J_1) \cdot \mathbf{B} \\ \mathbf{A}(2,1) \cdot \mathbf{B} & \mathbf{A}(2,2) \cdot \mathbf{B} & \cdots & \mathbf{A}(2,J_1) \cdot \mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}(I_1,1) \cdot \mathbf{B} & \mathbf{A}(I_1,2) \cdot \mathbf{B} & \cdots & \mathbf{A}(I_1,J_1) \cdot \mathbf{B} \end{bmatrix}$$

or equivalently

$$(\mathbf{A} \otimes \mathbf{B}) ([i_1 - 1] I_2 + i_2, [j_1 - 1] J_2 + j_2) \coloneqq \mathbf{A}(i_1, j_1) \cdot \mathbf{B}(i_2, j_2) \; .$$

Now suppose that $\mathbf{A}$ and $\mathbf{B}$ have F columns each. Let $\{\mathbf{a}_1, \ldots, \mathbf{a}_F\}$ and $\{\mathbf{b}_1, \ldots, \mathbf{b}_F\}$ be the columns of $\mathbf{A}$ and $\mathbf{B}$ respectively. Then we define the *Khatri-Rao product* or *column-wise Kronecker product* of $\mathbf{A}$ and $\mathbf{B}$ as follows:

$$\mathbf{A} \odot \mathbf{B} \coloneqq [\, \mathbf{a}_1 \otimes \mathbf{b}_1 \, , \; \ldots \; , \, \mathbf{a}_F \otimes \mathbf{b}_F \,] \; .$$

An equivalent definition of the Khatri-Rao product is

$$(\mathbf{A} \odot \mathbf{B}) ([i_1 - 1] I_2 + i_2, r) \coloneqq \mathbf{A}(i_1, r) \cdot \mathbf{B}(i_2, r) \; .$$

It is sometimes useful to "unfold" a tensor into a matrix. Let $\mathbf{T}$ be a tensor with size (I, J, K). We explain how to build $\text{unfold}_{[n]}(\mathbf{T})$, the "unfolding of $\mathbf{T}$ along the n-th mode": first cycle-shift the modes of $\mathbf{T}$ so that n is the new first mode of $\mathbf{T}'$. Then stack the "slices" $\mathbf{T}'(:,:,k')$ side-by-side to build $\text{unfold}_{[n]}(\mathbf{T})$. Equivalently, we could have given a definition based on the indices. For instance, for $n = 2$, we have

$$\text{unfold}_{[2]}(\mathbf{T})(j, [i-1]K + k) \coloneqq \mathbf{T}(i, j, k) \ .$$

Finally, when doing numerical computations, we will have to measure the accuracy of some approximations. Sometimes, we also want to prevent some variables to grow too fast or excessively. This leads us to define norms on vectors, matrices and tensors. Let $\mathbf{T}$ be any array with indices $(i_1, \ldots, i_N)$ and let $0 < p < \infty$. We define the L^p-norm of $\mathbf{T}$ as

$$\| \mathbf{T} \|_{\mathrm{L}^p} \coloneqq \left[\sum_{i_1,\ldots,i_N} | \mathbf{T}(i_1, \ldots, i_N) |^p \right]^{1/p} \ .$$

We also define the L^∞-norm of $\mathbf{T}$ to be

$$\| \mathbf{T} \|_{\mathrm{L}^\infty} \coloneqq \max_{i_1,\ldots,i_N} | \mathbf{T}(i_1, \ldots, i_N) | \ .$$

3.2 Presentation of the problem

The section is divided into three subsections. First we give a formal definition for the asymptotic complexity of matrix multiplication. Then we show how matrix multiplication can be represented with a third-order tensor. Finally, we explain how the problem of matrix multiplication asymptotic complexity is linked to the problem of determining the rank of the associated tensor.

3.2.1 Complexity of matrix multiplication

In the following, we will use the symbol $\langle m, p, n \rangle$ for referring to the "multiplication of a (m, p) matrix with a (p, n) matrix". Let $M(h)$ be the minimal number of scalar additions and multiplications necessary to compute $\langle h, h, h \rangle$. An exact determination of $M(h)$ seems to be outside the range of methods available at the present time (c.f. the first and second paragraphs of Chapter 2). We limit ourselves to analyze the asymptotic growth of $M(h)$ as h tends to infinity. Therefore, we define the *exponent of matrix multiplication* as

$$\omega \coloneqq \inf \left\{ \tau \in \mathbb{R} \,\middle|\, \text{there exists a } C \in \mathbb{R} \text{ such that } M(h) \le C h^\tau \text{ for all } h \ge 0 \right\} \ .$$

If we can determine the value of ω, then we would be able to say that "the complexity of $\langle h, h, h \rangle$ is in $\mathcal{O}(h^\omega)$". From the "straightforward" or "naive" computation of $\langle h, h, h \rangle$, we already know that $M(h) \le h^2 (2h - 1)$. On the other hand, the product

$$\begin{bmatrix} x_1 & 0 & \cdots & 0 \\ x_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x_h & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_h \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

requires h^2 independent multiplications. Hence, we have a first estimate $\omega \in [2, 3]$.

3.2.2 The matrix multiplication tensor

Let $\mathbf{W}$ be some (m,n) matrix and observe that $\mathbf{W}$ is completely determined if we know the value of $\operatorname{trace}(\mathbf{WZ})$ for every (n,m) matrix $\mathbf{Z}$. Indeed the value of $\mathbf{W}(i,j)$ is directly given by $\operatorname{trace}(\mathbf{WZ}_{ij})$ where $\mathbf{Z}_{ij}$ is the matrix defined by $\mathbf{Z}_{ij}(k,\ell) = 1$ if $(k,\ell) = (j,i)$ and $\mathbf{Z}_{ij}(k,\ell) = 0$ otherwise. Now consider the bilinear map

$$\mathbb{R}^{m\times p} \times \mathbb{R}^{p\times n} \to \mathbb{R}^{m\times n} : (\mathbf{X},\mathbf{Y}) \mapsto \mathbf{XY}\,.$$

From the observations above, this map is completely determined by the map

$$\mathbb{R}^{m\times p} \times \mathbb{R}^{p\times n} \times \mathbb{R}^{n\times m} : (\mathbf{X},\mathbf{Y},\mathbf{Z}) \mapsto \operatorname{trace}(\mathbf{XYZ})\ .$$

This map is a trilinear map in the variables $\mathbf{X},\mathbf{Y},\mathbf{Z}$ and, from it, we will be able to define the "structural tensor for $\langle m,p,n\rangle$". This tensor is denoted $\mathbf{T}_{mpn}$ and is defined by

$$\boxed{\operatorname{trace}(\mathbf{XYZ}) = \sum_{i=1}^{mp}\sum_{j=1}^{pn}\sum_{k=1}^{nm} \mathbf{T}_{mpn}(i,j,k)\cdot \operatorname{vec}(\mathbf{X})(i)\cdot \operatorname{vec}(\mathbf{Y})(j)\cdot \operatorname{vec}(\mathbf{Z})(k)}\ .$$

For example, the "structural tensor for $\langle 2,2,2\rangle$" is given by

$$\left[\underbrace{\begin{array}{cccc} 1&0&0&0\\ 0&0&0&0\\ 0&1&0&0\\ 0&0&0&0 \end{array}}_{\mathbf{T}_{222}(:,:,1)}\middle|\underbrace{\begin{array}{cccc} 0&0&1&0\\ 0&0&0&0\\ 0&0&0&1\\ 0&0&0&0 \end{array}}_{\mathbf{T}_{222}(:,:,2)}\middle|\underbrace{\begin{array}{cccc} 0&0&0&0\\ 1&0&0&0\\ 0&0&0&0\\ 0&1&0&0 \end{array}}_{\mathbf{T}_{222}(:,:,3)}\middle|\underbrace{\begin{array}{cccc} 0&0&0&0\\ 0&0&1&0\\ 0&0&0&0\\ 0&0&0&1 \end{array}}_{\mathbf{T}_{222}(:,:,4)}\right].$$

3.2.3 Relation between rank and complexity

We already mentioned that determining the rank of the associated "structural tensor" can give us upper bounds for the asymptotic complexity of matrix multiplication. Let's see how the two concepts are related to each other. Let m,p,n be fixed positive integers and suppose we have a F-PD of $\mathbf{T}_{mpn}$. From this F-PD, we can build an algorithm for computing $\langle m,p,n\rangle$ with F active multiplications only. By "active multiplications", we mean multiplications between factors that "both depend on the entries of the matrices"; multiplications of those entries with scalar coefficients are not considered as active multiplications. For example, if $\mathbf{X}$ and $\mathbf{Y}$ are matrices, then an expression like

$$[\alpha_1\mathbf{X}(1,2) + \alpha_2\mathbf{X}(2,4)]\cdot[\beta_1\mathbf{Y}(2,1) + \beta_2\mathbf{Y}(3,2)]$$

involves four multiplications of the entries of the matrices with scalar coefficients and, since both expressions $[\alpha_1\mathbf{X}(1,2) + \alpha_2\mathbf{X}(2,4)]$ and $[\beta_1\mathbf{Y}(2,1) + \beta_2\mathbf{Y}(3,2)]$ "depend on the entries of the matrices", there is also one active multiplication. The reasons for this distinction is that we will apply this algorithm recursively on matrices with size (m^α, p^α)

and (p^α, n^α). Doing this, we obtain an algorithm to compute $\langle m^\alpha, p^\alpha, n^\alpha \rangle$ and we can show that this algorithm has a complexity in $\mathcal{O}\left(F^\alpha\right)$. This leads us to the following theorem:

Theorem 3.1. *Let m, p, n be positive integers and suppose we have a F-PD of $\mathbf{T}_{mpn}$, then we have an upper bound on the exponent ω of matrix multiplication given by the inequality $(mpn)^{\omega/3} \leq F$.*

For the interested reader, a proof of this theorem can be found in [10, p. 378]. Instead of giving the complete proof, we prefer to give an example that will exhibit how to build an algorithm for $\langle m, p, n \rangle$ starting from a F-PD of $\mathbf{T}_{mpn}$ and also how this algorithm can be used to find the inequality $(mpn)^{\omega/3} \leq F$ appearing in the theorem. Consider the $\langle 2, 2, 2 \rangle$ case and observe that the following matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ provide a 7-PD of $\mathbf{T}_{222}$:

$$\begin{array}{|c|c|c|}
\hline
\mathbf{A}^\top & \mathbf{B}^\top & \mathbf{C}^\top \\
\hline
\begin{bmatrix} +1 & 0 & 0 & +1 \\ +1 & 0 & 0 & 0 \\ 0 & +1 & 0 & -1 \\ -1 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \\ +1 & +1 & 0 & 0 \\ 0 & 0 & +1 & +1 \end{bmatrix} &
\begin{bmatrix} +1 & 0 & 0 & +1 \\ 0 & 0 & +1 & +1 \\ +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 \\ +1 & +1 & 0 & 0 \\ -1 & 0 & +1 & 0 \\ 0 & +1 & 0 & -1 \end{bmatrix} &
\begin{bmatrix} +1 & 0 & 0 & +1 \\ 0 & +1 & 0 & -1 \\ 0 & 0 & +1 & +1 \\ +1 & +1 & 0 & 0 \\ -1 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \\ +1 & 0 & 0 & 0 \end{bmatrix} \\
\hline
\end{array} . \tag{3.1}$$

In general, if $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is a F-PD of $\mathbf{T}_{mpn}$, then, from the definition of the matrix multiplication tensor $\mathbf{T}_{mpn}$ and the definition of a F-PD, we have that

$$\operatorname{trace}(\mathbf{XYZ}) = \sum_{r=1}^{F} \left[\operatorname{vec}(\mathbf{X})^\top \mathbf{A}(:, r)\right] \cdot \left[\operatorname{vec}(\mathbf{Y})^\top \mathbf{B}(:, r)\right] \cdot \left[\operatorname{vec}(\mathbf{Z})^\top \mathbf{C}(:, r)\right] . \tag{3.2}$$

From the decomposition (3.1) and using the identity (3.2), we are able to build an algorithm for the computation of $\langle 2, 2, 2 \rangle$ with 7 active multiplications. Let $\mathbf{X}$ and $\mathbf{Y}$ be two $(2, 2)$ matrices. The algorithm to compute their product $\mathbf{Z} := \mathbf{XY}$ consists of first computing the 7 following active products:

Product	$\left[\text{vec}(\mathbf{X})^\top \mathbf{A}(:,r)\right] \cdot \left[\text{vec}(\mathbf{Y})^\top \mathbf{B}(:,r)\right]$
m_1	$[\mathbf{X}(1,1)+\mathbf{X}(2,2)] \cdot [\mathbf{Y}(1,1)+\mathbf{Y}(2,2)]$
m_2	$\mathbf{X}(1,1) \cdot [\mathbf{Y}(1,2)+\mathbf{Y}(2,2)]$
m_3	$[\mathbf{X}(2,1)-\mathbf{X}(2,2)] \cdot \mathbf{Y}(1,1)$
m_4	$[\mathbf{X}(1,2)-\mathbf{X}(1,1)] \cdot \mathbf{Y}(2,2)$
m_5	$\mathbf{X}(2,2) \cdot [\mathbf{Y}(1,1)+\mathbf{Y}(2,1)]$
m_6	$[\mathbf{X}(1,1)+\mathbf{X}(2,1)] \cdot [\mathbf{Y}(1,2)-\mathbf{Y}(1,1)]$
m_7	$[\mathbf{X}(1,2)+\mathbf{X}(2,2)] \cdot [\mathbf{Y}(2,1)-\mathbf{Y}(2,2)]$

(3.3)

and then assemble them as follows to obtain the elements of **Z**:

Entry	Definition	Value
$\mathbf{Z}(1,1)$	$\sum_r m_r \cdot \mathbf{C}(1,r)$	$m_1+m_4-m_5+m_7$
$\mathbf{Z}(1,2)$	$\sum_r m_r \cdot \mathbf{C}(2,r)$	m_2+m_4
$\mathbf{Z}(2,1)$	$\sum_r m_r \cdot \mathbf{C}(3,r)$	m_3+m_5
$\mathbf{Z}(2,2)$	$\sum_r m_r \cdot \mathbf{C}(4,r)$	$m_1-m_2+m_3+m_5$

. (3.4)

We now show how the algorithm can be used to get a better bound on the exponent of matrix multiplication. The algorithm involves 7 active multiplications. Additions and subtractions between the entries of $\mathbf{X}$ and $\mathbf{Y}$ and between the products m_r and multiplications of the entries of $\mathbf{X}$ and $\mathbf{Y}$ and the products m_r with scalar coefficients are called "basic operations". The algorithm above needs 18 of such "basic operations". Now suppose that the entries $\mathbf{X}(i,j)$ and $\mathbf{Y}(i,j)$ are matrices with size $(2^\alpha, 2^\alpha)$ instead of just scalars. Hence, we have a block decomposition of $\mathbf{X}$ and $\mathbf{Y}$ which are now $(2^{\alpha+1}, 2^{\alpha+1})$ matrices. The computation of the product $\mathbf{XY}$ will now involve 7 active multiplications of the $(2^\alpha, 2^\alpha)$ matrices and 18 "basic operations" on $(2^\alpha, 2^\alpha)$ matrices. Each of those "basic operations" requires exactly 4^α scalar additions/subtractions/multiplications. Now let $M(2^\alpha)$ be the number of scalar additions/subtractions/multiplications necessary to compute $\langle 2^\alpha, 2^\alpha, 2^\alpha \rangle$ with the recursive algorithm. Then we have the recurrence

$$M(2^{\alpha+1}) = 7 \cdot M(2^\alpha) + 18 \cdot 4^\alpha \quad \text{and} \quad M(2^0) = 1 \ .$$

The solution of the recurrence is given by

$$M(2^\alpha) = 7 \cdot 7^\alpha - 6 \cdot 4^\alpha \ .$$

Hence, computing $\langle 2^\alpha, 2^\alpha, 2^\alpha \rangle$ requires at most $\mathcal{O}(7^\alpha)$ operations. Letting $h := 2^\alpha$, we have that $M(h)$ is asymptotically lower than $Ch^{\log_2(7)}$ for some $C \in \mathbb{R}$ and thus

$$\omega \leq \log_2(7) < 2.807 \,.$$

This is the bound originally found [1] by Strassen. In 1990, Coppersmith and Winograd proposed [11] the so-called "Coppersmith–Winograd algorithm" together with the upper bound $\omega < 2.376$. It is also important to point out that "this algorithm is unpractical due to the astronomically large constant in the complexity estimate" [6].

3.3 Sparse discrete decompositions

In the case of matrix multiplication tensors, some F-PD's are more interesting than others. We are especially interested in sparse and discrete decompositions (also called "sparse discrete solutions" and abbreviated as "SD-sol's"). In this section, we define what we mean by "sparse discrete solutions" and explain why they are more interesting for us.

3.3.1 Sparse solutions $\to$ stability + complexity

Let $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ be a F-PD of some matrix multiplication tensor $\mathbf{T}_{mpn}$. We say that $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is a *sparse* decomposition or solution if $\mathbf{A}, \mathbf{B}, \mathbf{C}$ contain only but a few zero elements. The decomposition (3.1) is an example of a sparse decomposition since each matrix $\mathbf{A}, \mathbf{B}, \mathbf{C}$ contains 12 non-zero entries compared on a total number of 28 entries.

A first advantage of sparse solutions concerns the "computational cost" of the resulting matrix multiplication algorithm. Indeed, consider, for instance, the computations (3.3) and (3.4) involved in the algorithm for $\langle 2, 2, 2 \rangle$. It is clear that the computation of

$$\operatorname{vec}(\mathbf{X})^\top \mathbf{A}(:, r) \qquad \text{and} \qquad \operatorname{vec}(\mathbf{Y})^\top \mathbf{B}(:, r) \tag{3.5}$$

involved in the active products

$$m_r = \left[\operatorname{vec}(\mathbf{X})^\top \mathbf{A}(:, r)\right] \cdot \left[\operatorname{vec}(\mathbf{Y})^\top \mathbf{B}(:, r)\right] \,,$$

and the computation of

$$\sum_{r=1}^{F} m_r \cdot \mathbf{C}(k, r) \tag{3.6}$$

will be faster if $\mathbf{A}(:, r)$, $\mathbf{B}(:, r)$ and $\mathbf{C}(k, :)$ contain mainly zero elements.

However, the rapidity of the algorithm is not the only motivation for minimizing the number of operations we have to compute in (3.5) and (3.6). Indeed, computing (3.5) and (3.6) with fewer additions will also improve the numerical stability of the algorithm. To see this, note that all the computations are done in floating-point arithmetic with machine precision `eps` and thus, when we evaluate an operation like $a + b$, the computed value, denoted by $\operatorname{float}(a + b)$, has the form

$$\operatorname{float}(a + b) = (a + b) \cdot (1 + \theta) \qquad \text{for some } |\theta| \leq \texttt{eps}.$$

For summations with more than two terms, we have

$$\text{float}\left(a_1 + \ldots + a_N\right) = \left(a_1 + \ldots + a_N\right) \cdot (1+\theta)^{N-1} \qquad \text{for some } |\theta| \leq \texttt{eps}$$

if the terms a_μ are accumulated sequentially and

$$\text{float}\left(a_1 + \ldots + a_N\right) = \left(a_1 + \ldots + a_N\right) \cdot (1+\theta)^{1+\text{ceil}[\log_2(N)]} \qquad \text{for some } |\theta| \leq \texttt{eps}$$

if we use a sequential "divide-and-conquer" algorithm. In each case, the precision of the computed result float $(a_1 + \ldots + a_N)$ decreases when the number N of terms to be summed is large. Hence, it is preferable to have only a few non-zero elements to compute in (3.5) and (3.6). For a detailed stability analysis of algorithms resulting from a F-PD $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ of the associated tensor $\mathbf{T}_{mpn}$, we refer the reader to [8].

3.3.2 Discrete solutions → exactness + complexity

Considering again the 7-PD (3.1) of $\mathbf{T}_{222}$, we observe that all the non-zero elements of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are either -1 or $+1$. In general, we will try to find decompositions such that the entries of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are rational numbers taking no more than five different values. For instance, we will search for solutions with entries in $\{-1, 0, +1\}$ or in $\{-1, -1/2, 0, +1/2, +1\}$ but other values are also possible. Such decompositions will be called *discrete* solutions.

There are two reasons we favor discrete solutions. The first one deals with the exactness of the decomposition. If a F-PD $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ of $\mathbf{T}_{mpn}$ is computed with a numerical algorithm, the decomposition will be precise up to machine precision:

$$\mathbf{T}_{mpn} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \pm \theta \mathbf{S} \qquad \text{where } \|\mathbf{S}\|_{\mathrm{L}^2} = 1 \text{ and } \theta \in \mathcal{O}\left(\texttt{eps}\right).$$

If we know that the entries of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ might take at most five different rational values, then we can easily recover them and obtain an exact decomposition of $\mathbf{T}_{mpn}$.

The second reason is that rational values will lower the computational cost of the resulting matrix multiplication algorithm. Indeed, when computing vec $(\mathbf{X})\,(i) \cdot \mathbf{A}\,(i, r)$ and vec $(\mathbf{Y})\,(j) \cdot \mathbf{B}\,(j, r)$ in the vector inner products (3.5) and when computing $m_r \cdot \mathbf{C}\,(k, r)$ in (3.6), it is less expensive to make computations like $\pm 1 \cdot \delta$ (does not cost anything) or $\pm 1/2 \cdot \delta$ (consists of shifting one bit of a) instead of computing $+0.845593745698 \cdot \delta$ or $-0.284574528463 \cdot \delta$ for example. It is important to point out that rational values might only lower the cost of the "basic operations" and thus have an positive impact on the exact cost $M(h)$. However, this cannot reduce the asymptotic complexity $\mathcal{O}\left(h^\omega\right)$.

3.4 Invariant transformations

We present the different transformations that can act on the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ involved in the F-PD of a matrix multiplication tensor $\mathbf{T}_{mpn}$. Invariant transformations modify the factor matrices but leave the recomposition $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ unchanged. We distinguish two types of transformation: the elementary transformations which are also valid for general tensors, and the "trace-like" transformations valid only if $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is a F-PD of a matrix multiplication tensor.

3.4.1 Elementary transformations

Let $[\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$ be a F-PD of some tensor $\mathbf{T}$ and let $\{\mathbf{a}_1,\dots,\mathbf{a}_F\}$, $\{\mathbf{b}_1,\dots,\mathbf{b}_F\}$ and $\{\mathbf{c}_1,\dots,\mathbf{c}_F\}$ be the columns of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ respectively. Let σ be a permutation of $\{1,\dots,F\}$ and observe that the result is unchanged if we sum the rank-at-most-1 terms in any order:

$$\sum_{r=1}^{F} \mathbf{a}_r(i)\cdot\mathbf{b}_r(j)\cdot\mathbf{c}_r(k) = \sum_{r=1}^{F} \mathbf{a}_{\sigma(r)}(i)\cdot\mathbf{b}_{\sigma(r)}(j)\cdot\mathbf{c}_{\sigma(r)}(k) \ .$$

Hence, we have a first class of elementary transformations:

permutation of the columns	$\mathbf{a}_r \leftarrow \mathbf{a}_{\sigma(r)}$ $\mathbf{b}_r \leftarrow \mathbf{b}_{\sigma(r)}$ $\mathbf{c}_r \leftarrow \mathbf{c}_{\sigma(r)}$	for all $r \in \{1,\dots,F\}$.

Now let $\{\lambda_1,\dots,\lambda_F\}$, $\{\mu_1,\dots,\mu_F\}$ and $\{\nu_1,\dots,\nu_F\}$ be real numbers satisfying

$$\lambda_r\mu_r\nu_r = 1 \qquad \text{for all } r\in\{1,\dots,F\}.$$

Then it is clear that

$$\sum_{r=1}^{F} \mathbf{a}_r(i)\cdot\mathbf{b}_r(j)\cdot\mathbf{c}_r(k) = \sum_{r=1}^{F} [\mathbf{a}_r(i)\cdot\lambda_r]\cdot[\mathbf{b}_r(j)\cdot\mu_r]\cdot[\mathbf{c}_r(k)\cdot\nu_r]$$

so that we can define a second class of elementary transformations:

scaling and counter-scaling of the columns	$\mathbf{a}_r \leftarrow \mathbf{a}_r\cdot\lambda_r$ $\mathbf{b}_r \leftarrow \mathbf{b}_r\cdot\mu_r$ $\mathbf{c}_r \leftarrow \mathbf{c}_r\cdot\nu_r$	for all $r \in \{1,\dots,F\}$.

3.4.2 "Trace-like" transformations

We define a class of invariant transformations specific for the F-PD's of matrix multiplication tensors $\mathbf{T}_{mpn}$. Like in the previous section, let $[\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$ be a F-PD of $\mathbf{T}_{mpn}$ and let $\{\mathbf{a}_1,\dots,\mathbf{a}_F\}$, $\{\mathbf{b}_1,\dots,\mathbf{b}_F\}$ and $\{\mathbf{c}_1,\dots,\mathbf{c}_F\}$ be the columns of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ respectively. First let us introduce the "reshape operator" which maps a IJ-elements vector $\mathbf{x}$ to the (I,J) matrix defined as follows:

$$\operatorname{rshp}_{[I,J]}(\mathbf{x})(i,j) := \mathbf{x}([j-1]I+i) \ .$$

We also introduce the map $\langle\,\cdot\,,\cdot\,\rangle : \mathbb{R}^{I\times J}\times\mathbb{R}^{I\times J}\to\mathbb{R}$ defined by

$$\langle\mathbf{U},\mathbf{V}\rangle := \operatorname{trace}\left(\mathbf{U}^\top\mathbf{V}\right) = \sum_{i=1}^{I}\sum_{j=1}^{J}\mathbf{U}(i,j)\,\mathbf{V}(i,j) \ . \tag{3.7}$$

If $\mathbf{X}$ is a (I,J) matrix and $\mathbf{y}$ is a IJ-elements vector, then observe that

$$\operatorname{vec}(\mathbf{X})^\top\mathbf{y} = \left\langle\mathbf{X},\operatorname{rshp}_{[I,J]}(\mathbf{y})\right\rangle \ .$$

Hence, from the identity (3.2), we have that

$$\operatorname{trace}(\mathbf{XYZ}) = \sum_{r=1}^{F} \left\langle \mathbf{X}, \operatorname{rshp}_{[m,p]}(\mathbf{a}_r) \right\rangle \cdot \left\langle \mathbf{Y}, \operatorname{rshp}_{[p,n]}(\mathbf{b}_r) \right\rangle \cdot \left\langle \mathbf{Z}, \operatorname{rshp}_{[n,m]}(\mathbf{c}_r) \right\rangle .$$

Now let $\mathbf{P} \in \operatorname{GL}(m)$, $\mathbf{Q} \in \operatorname{GL}(p)$ and $\mathbf{R} \in \operatorname{GL}(n)$ (where "$\operatorname{GL}(h)$" stands for the general linear group of degree h, i.e. the set of invertible (h, h) matrices) and consider the identity

$$\operatorname{trace}\left(\left[\mathbf{P}^{-1}\mathbf{XQ}\right] \cdot \left[\mathbf{Q}^{-1}\mathbf{YR}\right] \cdot \left[\mathbf{R}^{-1}\mathbf{ZP}\right]\right) = \operatorname{trace}(\mathbf{XYZ}) .$$

This allows us to define a new class of invariant transformations:

Transformation of reshaped columns through $\mathbf{P}, \mathbf{Q}, \mathbf{R}$	$\operatorname{rshp}_{[m,p]}(\mathbf{a}_r) \leftarrow \mathbf{P}^{-1} \cdot \operatorname{rshp}_{[m,p]}(\mathbf{a}_r) \cdot \mathbf{Q}$ $\operatorname{rshp}_{[p,n]}(\mathbf{b}_r) \leftarrow \mathbf{Q}^{-1} \cdot \operatorname{rshp}_{[p,n]}(\mathbf{b}_r) \cdot \mathbf{R}$ $\operatorname{rshp}_{[n,m]}(\mathbf{c}_r) \leftarrow \mathbf{R}^{-1} \cdot \operatorname{rshp}_{[n,m]}(\mathbf{c}_r) \cdot \mathbf{P}$	for all $r \in \{1, \ldots, F\}$.

Equivalently, this provides transformations of the columns of $\mathbf{A}, \mathbf{B}, \mathbf{C}$:

Transformation of columns through $\mathbf{P}, \mathbf{Q}, \mathbf{R}$	$\mathbf{a}_r \leftarrow \left(\mathbf{Q}^\top \otimes \mathbf{P}^{-1}\right) \cdot \mathbf{a}_r$ $\mathbf{b}_r \leftarrow \left(\mathbf{R}^\top \otimes \mathbf{Q}^{-1}\right) \cdot \mathbf{b}_r$ $\mathbf{c}_r \leftarrow \left(\mathbf{P}^\top \otimes \mathbf{R}^{-1}\right) \cdot \mathbf{c}_r$	for all $r \in \{1, \ldots, F\}$.

In the following, transformations via permutation of the columns are abbreviated as "T-perm", scaling/counter-scaling is abbreviated as "T-scale" while transformations through invertible matrices $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ are denoted by "T-trace". Let $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ be two F-PD's of a matrix multiplication tensor. Suppose there exists a transformation T-perm such that we can transform $\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1$ into $\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2$. Then we say that $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ are "(T-perm)-equivalent". Similarly, we can also have "(T-scale+T-trace)-equivalent" or "(T-perm+T-scale+T-trace)-equivalent" F-Pd's. For simplicity, "(T-perm+T-scale+T-trace)-equivalence" will often be abbreviated as "inv-equivalence". Being inv-equivalent is a equivalence relation. If $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is a F-PD of some matrix multiplication tensor $\mathbf{T}_{mpn}$, its inv-equivalence class is denoted

$$\mathcal{S}(\mathbf{A}, \mathbf{B}, \mathbf{C}) := \left\{ \mathbf{A}', \mathbf{B}', \mathbf{C}' \;\middle|\; [\![\mathbf{A}', \mathbf{B}', \mathbf{C}']\!] \text{ is inv-equivalent to } [\![\mathbf{A}', \mathbf{B}', \mathbf{C}']\!] \right\} .$$

The main questions we will address in the two penultimate chapters are summarized in Table 3.1. As we will see in Sec. 5.2 and Sec. 6.1, one of the main challenges in the implementation of algorithms for answering Question 1 and Question 2 will come from the (T-perm)-equivalence. Indeed, for such transformations a naive algorithm (trying all possible permutations) would be prohibitive in running time since the number of FLOP's would grow as $\mathcal{O}(F!)$. Hence, we will have to find tricks to reduce the cost of getting rid of the (T-perm)-equivalence.

Question 1: (Chapter 5)	Given a F-PD $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ of $\mathbf{T}_{mpn}$, does a discrete (T-scale+T-trace)-equivalent F-PD $\mathbf{A}^*, \mathbf{B}^*, \mathbf{C}^*$ exist (transformations T-perm are not relevant here since permutating the columns will not change the discrete character of the decomposition)? If this is the case, such a F-PD will be called a "discretizable" decomposition.
Question 2: (Chapter 6)	What is the distribution of the $\mathcal{S}(\mathbf{A}, \mathbf{B}, \mathbf{C})$'s if the F-PD's are "chosen randomly" (in a sense that we will defined later)? For example, is there a unique equivalent class $\mathcal{S}(\mathbf{A}, \mathbf{B}, \mathbf{C})$? If we "choose randomly" two F-PD's of the same tensor, is there a non-zero probability for them to be inv-equivalent?

Table 3.1 – Questions related to invariant transformations.

Remark. At first sight, it might look like the transformations T-scale act as a particular case of the transformations T-trace with

$$\mathbf{P} := \left(\frac{\nu}{\lambda}\right)^{1/3} \mathbf{I}_m \quad , \quad \mathbf{Q} := \left(\frac{\lambda}{\mu}\right)^{1/3} \mathbf{I}_p \quad , \quad \mathbf{R} := \left(\frac{\mu}{\nu}\right)^{1/3} \mathbf{I}_n$$

for example. In fact, this is not the case since those $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ will rescale all the columns of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ with the same coefficient λ, μ, ν (provided $\lambda\mu\nu = 1$). Transformations T-scale are more general since we admit different $\{\lambda_1, \ldots, \lambda_F\}$, $\{\mu_1, \ldots, \mu_F\}$ and $\{\nu_1, \ldots, \nu_F\}$.

Chapter 4

Computation of sparse discrete decompositions

4.1 A procedure to compute SD-sol's

We present a procedure for computing sparse discrete decompositions (if some exists) of a given tensor. The procedure involves sequential "solve's" of a non-smooth optimization problem. The first subsection deals with this optimization problem and how we use it in the procedure to find SD-sol's. In the second subsection, we present a method for solving problems belonging to a general sub-class (defined in the same subsection) of non-smooth optimization problems. In the last subsection, we explain precisely how the method applies to our specific optimization problem.

The procedure, as presented, is dedicated to compute only sparse decompositions with factor matrices having their entries in the set $\{-1, 0, +1\}$ (and not in $\{-2, -1, 0, +1, +2\}$ or $\{-1, -1/2, 0, +1/2, +1\}$ for example while such decompositions might still be interesting, as we pointed out in Sec. 3.3). The reason is that we constrain the entries of the factor matrices between -1 and $+1$ (hence, it is impossible to get a "± 2") and we fix recursively the values that are extremal (in absolute value), i.e. we fix the values close to ± 1 (and thus, it is difficult to converge to a "$\pm 1/2$" in the factor matrices). Modifications of the procedure to compute more general sparse discrete solutions are discussed briefly in the remark at the end of Sec. 4.2.

4.1.1 A non-smooth optimization problem

We would like to solve the following optimization problem:

$$
\begin{array}{|c|}
\hline
\text{minimize} \\
\hdashline
\|\mathbf{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_{\mathrm{L}^2}^2 + \rho\{\|\mathbf{A}\|_{\mathrm{L}^1} + \|\mathbf{B}\|_{\mathrm{L}^1} + \|\mathbf{C}\|_{\mathrm{L}^1}\} \\
\hdashline
\text{subject to} \\
\hdashline
\begin{array}{lll}
 & \mathbf{A}(i,r) \in [-1,+1] & \text{for all } i,r \\
(\#1) & \mathbf{B}(j,r) \in [-1,+1] & \text{for all } j,r \\
 & \mathbf{C}(k,r) \in [-1,+1] & \text{for all } k,r \\
\hdashline
 & \mathbf{A}(i,r) = a_{ir} & \text{for all } (i,r) \in I_1 \\
(\#2) & \mathbf{B}(j,r) = b_{jr} & \text{for all } (j,r) \in I_2 \\
 & \mathbf{C}(k,r) = c_{kr} & \text{for all } (k,r) \in I_3
\end{array} \\
\hdashline
\text{with variables} \\
\hdashline
\mathbf{A} \in \mathbb{R}^{I\times F} \ , \ \mathbf{B} \in \mathbb{R}^{J\times F} \ , \ \mathbf{C} \in \mathbb{R}^{K\times F} \ . \\
\hline
\end{array}
\quad . \qquad (4.1)
$$

The objective function consists of two terms with distinct purposes:

$$
\|\mathbf{X} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_{\mathrm{L}^2}^2 + \rho\{\|\mathbf{A}\|_{\mathrm{L}^1} + \|\mathbf{B}\|_{\mathrm{L}^1} + \|\mathbf{C}\|_{\mathrm{L}^1}\} \ .
$$

With the first term, we hope to attract the solution to be a F-PD (or not far from being a F-PD) of $\mathbf{X}$. The second term is a "regularization term" and, with it, we hope that, among all the solutions $\mathbf{A},\mathbf{B},\mathbf{C}$ that are close to be a F-PD of $\mathbf{X}$, we will converge to a sparse solution. Using a L^1-regularization is a trick that is often used in statistics and machine learning to induce sparsity in the solution. For more information about the L^1-regularization, we refer the reader to [12].

The method we use to solve problem (4.1) is described in the next subsection. It is a variant of the gradient method for minimizing functions involving a non-smooth convex term. The method is an iterative process and we need to provide an initial iterate/guess. Note that, since the problem is not convex, we have no guarantees on the convergence to a global minimum. Global and local convergence of the process will be discussed in the next subsection as well.

Before describing the method we use to solve the problem, let us first present the procedure for finding SD-sol's. This involves successive "solve's" of (4.1). More precisely, the idea is the following: in *Phase 1*, we start from a random initial guess $\mathbf{A}_0,\mathbf{B}_0,\mathbf{C}_0$ and use the iterative process to find a solution $\mathbf{A}_1,\mathbf{B}_1,\mathbf{C}_1$ to (4.1) with a low ρ. With this, we hope to converge to an accurate F-PD of $\mathbf{X}$, i.e. $[\![\mathbf{A}_1,\mathbf{B}_1,\mathbf{C}_1]\!] \approx \mathbf{X}$.

In *Phase 2*, we increase the value of ρ and try to solve the problem starting from the last solution $\mathbf{A}_1,\mathbf{B}_1,\mathbf{C}_1$ as first iterate for the iterative process. The objective is to obtain a sparse solution $\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2$ which is also not far from being a F-PD of $\mathbf{X}$. The set of constraints (#1) in problem (4.1) is there to help the first solution $\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1$ to be not completely degenerate so that it will be easier to compute a sparse solution $\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2$, starting from $\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1$. The second set of constraints (#2) is not used during Phase 1 and Phase 2, i.e. I_1, I_2, I_3 are empty sets.

Once we have a sparse approximate decomposition, we try to obtain a SD-sol from it. This is where the second set of constraints (#2) is used. The computations of the parameters involved in (#2), i.e. I_1, I_2, I_3 and the corresponding a_{ir}, b_{jr}, c_{kr}, are done during *Pre-phase 3*. The procedure is the following: let $\mathbf{A}_\alpha, \mathbf{B}_\alpha, \mathbf{C}_\alpha$ be a sparse approximate decomposition, choose a parameter $0 < \eta < 1$ and define the following sets of indices:

$$I_1^+ := \left\{ (i,r) \,\middle|\, \mathbf{A}_\alpha (i,r) \geq +\eta \right\} \quad \text{and} \quad I_1^- := \left\{ (i,r) \,\middle|\, \mathbf{A}_\alpha (i,r) \leq -\eta \right\} .$$

In words, the parameter η decides "above which threshold (in absolute value) the entries of $\mathbf{A}_\alpha$ must be considered to be -1 or $+1$". For example, we may impose that all the entries greater than $+0.9$ or lower than -0.9 are fixed to be $+1$ and -1 respectively. The sets I_1^+, I_1^- contain the indices of those entries and we also have to define $a_{ir} := +1$ for all $(i,r) \in I_1^+$ and $a_{ir} := -1$ for all $(i,r) \in I_1^-$ and finally $I_1 := I_1^+ \cup I_1^-$. The sets I_2^+, I_2^- and I_3^+, I_3^- containing indices of $\mathbf{B}_\alpha$ and $\mathbf{C}_\alpha$ respectively and the constraint values b_{jr} and c_{kr} are defined in the same way.

Those constraints are added to (4.1) and then, in *Phase 3*, we try to find a solution $\mathbf{A}_{\alpha+1}, \mathbf{B}_{\alpha+1}, \mathbf{C}_{\alpha+1}$ starting from $\mathbf{A}_\alpha, \mathbf{B}_\alpha, \mathbf{C}_\alpha$ as initial iterate for the iterative process. Pre-phase 3 and Phase 3 are repeated several times until we converge to a SD-sol, if some exists. A schematic of the procedure is represented in Fig. 4.1 and numerical examples will be presented in Sec. 4.2.

4.1.2 Gradient method for composite functions

We introduce a general sub-class of non-smooth optimization problems and we present a method for solving them. The method will not always converge to a global minimum but the convergence to a stationary point is ensured. For this, we rely on [13]. We do not bring anything new in this subsection but merely synthesize the results of [13] that are relevant for our purpose, i.e. a gradient method to solve problem (4.1).

Let $\mathfrak{U}$ be a finite-dimensional vector space over $\mathbb{R}$ with inner product $(u,v) \mapsto \langle u, v \rangle$ and with $u \mapsto \|u\|$ for the associated norm. Suppose $f(x)$ is a smooth function from $\mathfrak{U}$ to $\mathbb{R}$ and $\psi(x)$ is a convex (possibly non-smooth) function from $\mathfrak{U}$ to $\mathbb{R}$. Finally, let $\mathfrak{Q}$ be a convex subset of $\mathfrak{U}$. We suppose that $\psi(x)$ and $\mathfrak{Q}$ are "simple" in a sense that will be defined later. We consider the problem consisting of:

$$\text{minimize} \quad \phi(x) := f(x) + \psi(x) \qquad \text{subject to } x \in \mathfrak{Q}.$$

Let $\bar{x}$ be a point of $\mathfrak{Q}$ and choose a constant $L > 0$. Consider the following "first-order

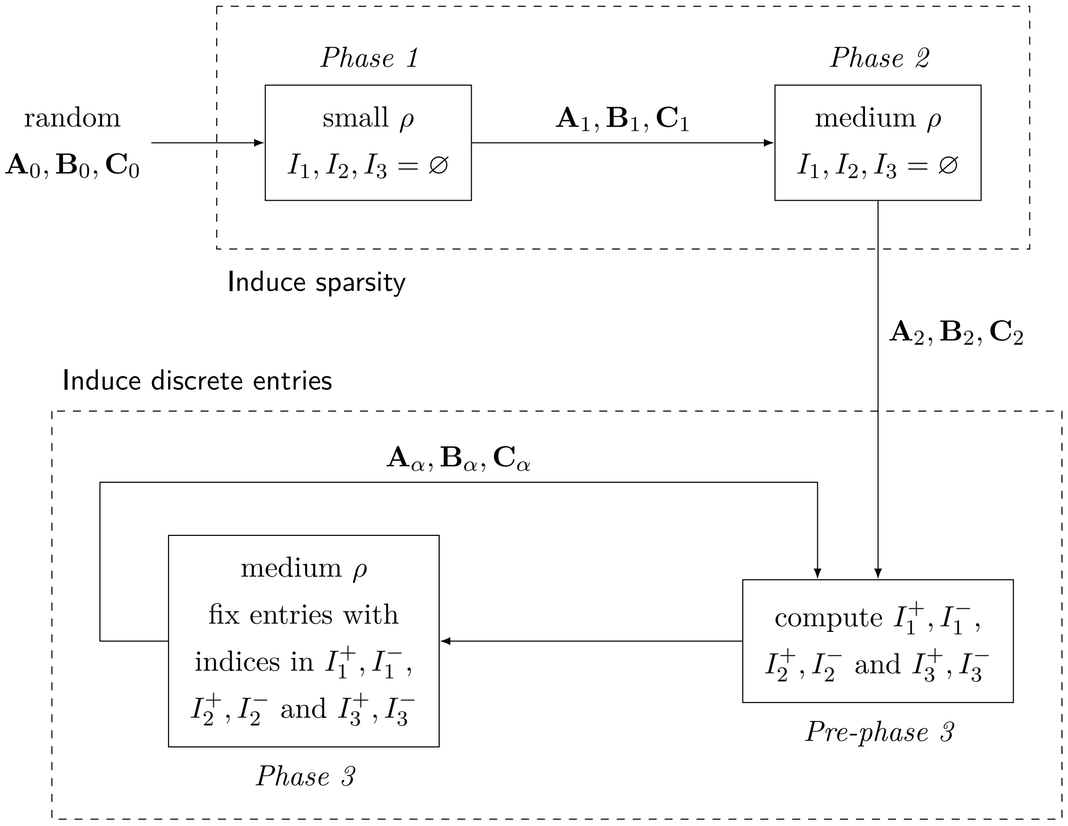

Phase 1
Phase 2
random
$\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0$
small ρ
$I_1, I_2, I_3 = \varnothing$
$\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1$
medium ρ
$I_1, I_2, I_3 = \varnothing$
Induce sparsity
$\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2$
Induce discrete entries
$\mathbf{A}_\alpha, \mathbf{B}_\alpha, \mathbf{C}_\alpha$
medium ρ
fix entries with
indices in I_1^+, I_1^-,
I_2^+, I_2^- and I_3^+, I_3^-
Phase 3
compute I_1^+, I_1^-,
I_2^+, I_2^- and I_3^+, I_3^-
Pre-phase 3

Figure 4.1 – Procedure to compute SD-sol's by solving several times the non-smooth optimization problem using recursively the last obtained solution as initial iterate for the computation of the new solution.

Input: First iterate $x_0 \in \mathcal{Q}$, initial constant $L_0 > 0$ and "step-size stopping criterion" $\epsilon > 0$.

```
[01] :  Set $k \leftarrow 0$ ;
[02] :  Set $x_{k+1} \leftarrow T_{L_k}(x_k)$ ;
[03] :  If $\phi(x_{k+1}) > m_{L_k}(x_{k+1} \,|\, x_k)$ :
[04] :      Set $L_k \leftarrow 2L_k$ ;
[05] :      Go to [02] ;
[06] :  If $\|x_{k+1} - x_k\| \leq \epsilon$ :
[07] :      Exit Function ;
[08] :  Set $L_{k+1} \leftarrow L_k/2$ ;
[09] :  Set $k \leftarrow k+1$ ;
[10] :  Go to [02] ;
```

Output: Last iterate $x_{k+1} \in \mathcal{U}$.

Algorithm 1 – Gradient method for composite functions.

approximation" of $\phi(x)$ around $\bar{x}$:

$$m_L(x \,|\, \bar{x}) := f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x} \rangle + \frac{L}{2} \|x - \bar{x}\|^2 + \psi(x) \ .$$

Observe that $x \mapsto m_L(x \,|\, \bar{x})$ is convex and has thus a unique minimizer:

$$T_L(\bar{x}) := \arg\min \left\{ x \mapsto m_L(x \,|\, \bar{x}) \;\middle|\; x \in \mathcal{Q} \right\} \ .$$

The hypothesis that "$\psi(x)$ and $\mathcal{Q}$ are simple" just means that $T_L(\bar{x})$ has to be easy to compute. Without this assumption, the method would be less interesting since we would have to solve a non-smooth optimization sub-problem at each iteration of the method.

The *composite gradient method* is described in Algorithm 1. The idea of the algorithm is the following: starting from a point x_k, we compute the minimizer $T_L(x_k)$ of the first-order approximation of $\phi(x)$ around x_k. This approximation involves the parameter L. If the exact value $\phi(T_L(x_k))$ is larger than the approximation $m_L(T_L(x_k) \,|\, x_k)$, then it means that the parameter L is too small. Hence, we increase L and we compute $T_L(x_k)$ again with the new L. We do this recursively until the value of the approximation at $T_L(x_k)$ is larger than the exact value $\phi(T_L(x_k))$. In this case, $T_L(x_k)$ becomes the new point x_{k+1} and we repeat the last steps, starting from a smaller L.

Now let us discuss the convergence of the composite gradient method. In the following, we suppose that $f(x)$ has Lipschitz-continuous gradient with parameter M:

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq M \|x_1 - x_2\| \qquad \text{for all } x_1, x_2 \in \mathcal{Q}. \tag{4.2}$$

This implies the following inequality (in fact, the relation is stronger than a simple implication, we prove in Appendix A that the two statements are equivalent):

$$|f(x) - [f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x} \rangle]| \leq \frac{M}{2} \|x - \bar{x}\|^2 \qquad \text{for all } x, \bar{x} \in \mathcal{Q}. \tag{4.3}$$

We analyze the execution of Algorithm 1. Suppose the value of k is k_* when the algorithm stops. We wonder how many times we have computed steps `[02]` and `[03]`. From inequality (4.3), we know that we will satisfy condition `[03]` only if $L_k \leq M$. Hence, when we start executing step `[08]`, we have $L_k \leq 2M$ and thus, after having executed step `[08]`, we must have $L_{k+1} \leq M$. Consider a fixed $k \geq 0$ and let n_k be the number of times we execute step `[02]` for this k. Clearly $n_k \geq 1$ and after having executed step `[08]`, we have $L_{k+1} = 2^{n_k - 2} L_k$. This provides a bound on n_k:

$$n_k = 2 + \log_2 \left(\frac{L_{k+1}}{L_k} \right) .$$

Summing over $k \geq 0$, we obtain how many times we compute steps `[02]` and `[03]`:

$$N_{k_*} := \sum_{k=0}^{k_*} n_k = 2(k_* + 1) + \log_2 \left(\frac{L_{k_*+1}}{L_0} \right) \leq 2(k_* + 1) + \log_2 \left(\frac{M}{L_0} \right) . \tag{4.4}$$

Now, what can we say about the step-size $\|x_{k+1} - x_k\|$? To answer this question, we will first have to analyze the decrease that is guaranteed on $\phi(x_k) - \phi(x_{k+1})$. Therefore, let $\bar{x}$ be fixed and define the convex function

$$\hat{m}(x \,|\, \bar{x}) := f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x} \rangle + \psi(x) = m_L(x \,|\, \bar{x}) - \frac{L}{2} \|x - \bar{x}\|^2 .$$

Since $x \mapsto m_L(x \,|\, \bar{x}) = \hat{m}(x \,|\, \bar{x}) + \frac{L}{2} \|x - \bar{x}\|^2$ is minimal at $T_L(\bar{x})$, the vector

$$h_L(\bar{x}) := -L \cdot (T_L(\bar{x}) - \bar{x})$$

must be a sub-gradient of $x \mapsto \hat{m}(x \,|\, \bar{x})$ at $T_L(\bar{x})$. Hence, we have

$$\hat{m}(\bar{x} \,|\, \bar{x}) \geq \hat{m}(T_L(\bar{x}) \,|\, \bar{x}) + \langle h_L(\bar{x}), \bar{x} - T_L(\bar{x}) \rangle = \hat{m}(T_L(\bar{x}) \,|\, \bar{x}) + L \|T_L(\bar{x}) - \bar{x}\|^2 .$$

Finally, this gives

$$\phi(\bar{x}) - m_L(T_L(\bar{x}) \,|\, \bar{x}) = \hat{m}(\bar{x} \,|\, \bar{x}) - m_L(T_L(\bar{x}) \,|\, \bar{x}) \geq \frac{L}{2} \|T_L(\bar{x}) - \bar{x}\|^2 . \tag{4.5}$$

Using the previous developments, we can prove the following theorem [13, Theorem 3]:

Theorem 4.1. *Let $\phi(x)$ be bounded from below on $\mathcal{Q}$ by a constant ϕ_*. Then, just before starting step* `[06]`*, we have*

$$\min_{0\le i\le k} \frac{L_i}{2}\left\|x_{i+1}-x_i\right\|^2 \le \frac{\phi(x_0)-\phi_*}{k+1}\,. \tag{4.6}$$

Moreover, there exists a sub-gradient $\xi(x_{k+1})$ of $\psi(x)$ at x_{k+1} such that, for any $u=\lambda\cdot(y-x_{k+1})$ for some $y\in\mathcal{Q}$ and some $\lambda\ge 0$ and satisfying $\|u\|=1$, we have

$$\langle\nabla f(x_{k+1})+\xi(x_{k+1}),u\rangle \ge -3M\left\|x_{k+1}-x_k\right\|\,. \tag{4.7}$$

Proof. First let us prove the inequality (4.6). Therefore, observe that, just before we start step `[06]`, we have that $\phi(x_{k+1})\le m_{L_k}(x_{k+1}\,|\,x_k)$. Hence,

$$\phi(x_k)-\phi(x_{k+1})\ge\phi(x_k)-m_L(x_{k+1}\,|\,x_k)\ge\frac{L_k}{2}\left\|x_{k+1}-x_k\right\|^2\,.$$

where the last inequality comes from (4.5). If we sum over $k\ge 0$, we find

$$\phi(x_0)-\phi(x_{k+1})=\sum_{i=0}^{k}\phi(x_i)-\phi(x_{i+1})\ge\sum_{i=0}^{k}\frac{L_i}{2}\left\|x_{i+1}-x_i\right\|^2$$

and thus

$$\phi(x_0)-\phi(x_*)\ge(k+1)\cdot\min_{0\le i\le k}\frac{L_i}{2}\left\|x_{i+1}-x_i\right\|^2\,.$$

In order to prove (4.7), remember that $h_{L_k}(x_k)$ is a sub-gradient of $x\mapsto\hat{m}(x\,|\,x_k)$ at x_{k+1} and thus $\xi(x_{k+1}):=h_{L_k}(x_k)-\nabla f(x_k)$ is a sub-gradient of $\psi(x)$ at x_{k+1}:

$$\langle\nabla f(x_{k+1})+\xi(x_{k+1}),u\rangle=\langle\nabla f(x_{k+1})-\nabla f(x_k),u\rangle+\langle h_{L_k}(x_k),u\rangle\,.$$

Using the definition of $h_{L_k}(x_k)$ and the Lipschitz-continuity of the gradient, we get

$$\langle\nabla f(x_{k+1})+\xi(x_{k+1}),u\rangle\ge -M\left\|x_{k+1}-x_k\right\|-L_k\left\|x_{k+1}-x_k\right\|\,.$$

We conclude with the fact that L_k is always smaller than $2M$. □

From the upper bounds (4.4) and (4.6), we conclude that Algorithm 1 will stop after at most $\mathcal{O}(1/\epsilon^2)$ steps. Moreover, inequality (4.7) tells us that the “descent rate” at x_{k+1} is at most proportional to $\|x_{k+1}-x_k\|$. Hence, if we use a small ϵ, we know that the output x_{k+1} will be an “almost stationary” point of $\phi(x)$. Nevertheless, since the problem is non-convex, we have no guarantee that the stationary point is a global minimum. The algorithm could converge to a local minimum as well. In theory, the method can also converge to a saddle-point or a local maximum, for example if it starts from such a point. However, since the method is a descent method, i.e. the value of $\phi(x_k)$ decreases at every step k, it is unlikely to be attracted to a saddle-point or a maximum. The convergence of the composite gradient method applied to our non-smooth optimization problem (4.1) will be illustrated in the next subsection.

4.1.3 Practical use of the gradient method

Let us come back to our initial problem (4.1). We show that the problem can be solved using the composite gradient method. We also explain in detail how we compute $\nabla f(x)$ and $T_L(x)$. Finally, we show that the theoretical predictions about the convergence and the total number of times we compute steps `[02]` and `[03]` are supported in practice.

In the case of tensor decomposition, the vector space $\mathcal{U}$ consists of the space of triples of matrices $\mathbf{A} \in \mathbb{R}^{I \times F}$, $\mathbf{B} \in \mathbb{R}^{J \times F}$ and $\mathbf{C} \in \mathbb{R}^{K \times F}$. The inner product is defined by

$$\langle (\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1), (\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2) \rangle := \operatorname{trace}\left(\mathbf{A}_1^\top \mathbf{A}_2\right) + \operatorname{trace}\left(\mathbf{B}_1^\top \mathbf{B}_2\right) + \operatorname{trace}\left(\mathbf{C}_1^\top \mathbf{C}_2\right) .$$

The smooth part of the objective function is

$$f(\mathbf{A}, \mathbf{B}, \mathbf{C}) := \|\mathbf{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_{\mathrm{L}^2}^2$$

and the non-smooth convex part is

$$\psi(\mathbf{A}, \mathbf{B}, \mathbf{C}) := \rho \left\{ \|\mathbf{A}\|_{\mathrm{L}^1} + \|\mathbf{B}\|_{\mathrm{L}^1} + \|\mathbf{C}\|_{\mathrm{L}^1} \right\} .$$

The gradient of $f(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is given by (c.f. Appendix B for details)

$$\left\{ \begin{array}{lllll} \nabla_{\mathbf{A}}(f) & = & \dfrac{\partial f}{\partial \mathbf{A}} & = & 2\left[\mathbf{A}(\mathbf{C} \odot \mathbf{B})^\top - \operatorname{unfold}_{[1]}(\mathbf{X})\right](\mathbf{C} \odot \mathbf{B}) \\ \nabla_{\mathbf{B}}(f) & = & \dfrac{\partial f}{\partial \mathbf{B}} & = & 2\left[\mathbf{B}(\mathbf{A} \odot \mathbf{C})^\top - \operatorname{unfold}_{[2]}(\mathbf{X})\right](\mathbf{A} \odot \mathbf{C}) \\ \nabla_{\mathbf{C}}(f) & = & \dfrac{\partial f}{\partial \mathbf{C}} & = & 2\left[\mathbf{C}(\mathbf{B} \odot \mathbf{A})^\top - \operatorname{unfold}_{[3]}(\mathbf{X})\right](\mathbf{B} \odot \mathbf{A}) \end{array} \right\} .$$

Let us mention that a direct computation of the products

$$(\mathbf{C} \odot \mathbf{B})^\top (\mathbf{C} \odot \mathbf{B}) \quad , \quad (\mathbf{A} \odot \mathbf{C})^\top (\mathbf{A} \odot \mathbf{C}) \quad , \quad (\mathbf{B} \odot \mathbf{A})^\top (\mathbf{B} \odot \mathbf{A})$$

involves the multiplication of matrices with sizes at least $(\min[IJ, JK, KI], F)$. Hence, this would require at least $\mathcal{O}(\min[IJ, JK, KI] F^2)$ operations (just to compute a (F, F) matrix). In fact, those products are never computed directly: we use the property that

$$(\mathbf{E}_1 \odot \mathbf{E}_2)^\top (\mathbf{E}_1 \odot \mathbf{E}_2) = \left(\mathbf{E}_1^\top \mathbf{E}_1\right) \circledast \left(\mathbf{E}_2^\top \mathbf{E}_2\right)$$

where $\circledast$ stands for the Hadamard or element-wise product. This reduces the cost for those multiplications to at most $\mathcal{O}(\max[I, J, K] F^2)$ operations.

Finally, we check that $f(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is twice continuously differentiable and the admissible domain is compact so that the Hessian is bounded and thus $\nabla f(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is Lipschitz-continuous.

Let $L > 0$ be fixed and suppose we have computed $\nabla_{\mathbf{A}}(f)$, $\nabla_{\mathbf{B}}(f)$ and $\nabla_{\mathbf{C}}(f)$: how do we compute $T_L(\mathbf{A}, \mathbf{B}, \mathbf{C})$? For simplicity, denote $(\mathbf{A}^+, \mathbf{B}^+, \mathbf{C}^+) := T_L(\mathbf{A}, \mathbf{B}, \mathbf{C})$. Fortunately, the problem is separable, meaning that each entry of $\mathbf{A}^+$, $\mathbf{B}^+$ and $\mathbf{C}^+$ can be computed separately using only the corresponding entry of $\nabla_{\mathbf{A}}(f)$, $\nabla_{\mathbf{B}}(f)$ and $\nabla_{\mathbf{C}}(f)$ respectively. For example, we compute $\mathbf{A}^{+}$ as follows:

$$\mathbf{A}^{+}(i,r) := \arg\min \left\{ a \mapsto \nabla_{\mathbf{A}}(f)(i,r) \cdot a + \frac{L}{2}\left[a - \mathbf{A}(i,r)\right]^2 + \rho\,|a| \;\middle|\; a \in [-1,+1] \right\}.$$

Hence, it boils down to minimizing a single-variable function consisting of a quadratic function and an "absolute value function" defined on the interval $[-1,+1]$. Splitting into the intervals $[-1,0]$ and $[0,+1]$, we just have to compute the minima of two quadratic functions, which is straightforward. The computation of $\mathbf{B}^{+}$ and $\mathbf{C}^{+}$ is identical.

Before we end this section, we would like to check whether the bounds (4.4) and (4.6) are satisfied in practice when applied to our non-smooth optimization problem. For this, let the parameters of (4.1) be defined as follows: $\mathbf{X}$ is the "structural tensor for $\langle 2,2,2\rangle$" and we search the best "rank-at-most-6 approximation" of it, i.e. $\mathbf{X} = \mathbf{T}_{222}$ and $F = 6$. The "regularization factor" ρ is set to 10^{-3}. For the constraints (#2), we use empty sets for I_1, I_2, I_3, i.e. no entries of the variables are completely fixed. We compute 20,000 iterations of the composite gradient method applied on (4.1) with those parameters. Then we compute the "scaled step-size" $\frac{L_k}{2}\left\|x_{k+1} - x_k\right\|^2$ and we compare this with the theoretical upper bound $\frac{\phi(x_0)-\phi_*}{k+1}$. Since the objective function of (4.1) is always non-negative, we use $\phi_* = 0$. We also compute the average value of n_k (the number of times we compute steps `[02]` and `[03]` for a fixed k), i.e. we compute the value of $\frac{N_k}{k+1}$. The results are represented in Fig. 4.2. Note that both the x- and the left y-axis have logarithmic scale. We observe that (4.4) and (4.6) are indeed verified in this example.

4.2 Examples of computation of SD-sol's

We use the method described in the previous section to compute sparse discrete decompositions of several matrix multiplication tensors. We explain in detail how we compute SD-sol's for the $\langle 2,2,2\rangle$, $\langle 2,3,2\rangle$, $\langle 3,2,3\rangle$ and $\langle 3,3,3\rangle$ cases: which "regularization factor" ρ, which "threshold" η and which "step-size stopping criterion" ϵ we use. We also investigate the influence of ρ on the sparsity of the resulting decomposition. For the sake of readability, the figures and tables mentioned in this section are gathered in Appendix C, at the end of the report. Finally, we briefly compare our results with the results presented in two state-of-the-art papers, namely [6] (2013) and [7] (2017).

4.2.1 Sparse discrete decompositions of small tensors

First we start with the $\langle 2,2,2\rangle$ case. It is known [9] that the rank of the associated tensor is 7. Hence, we would like to find a 7-PD of $\mathbf{T}_{222}$. For this, we try to solve (4.1) with $\mathbf{X} = \mathbf{T}_{222}$ and $F = 7$. As initial guess, we use matrices $\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0$ where the entries are chosen randomly uniform between -1 and $+1$. Which values we use for the parameters ρ and ϵ in Phase 1 and Phase 2 of the procedure are detailed in Fig. C.1. In this figure, you also find the quality of the approximation $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$ and the total number of "non-very-small" values in $\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa$ (values smaller than $5 \cdot 10^{-2}$ are considered as "very

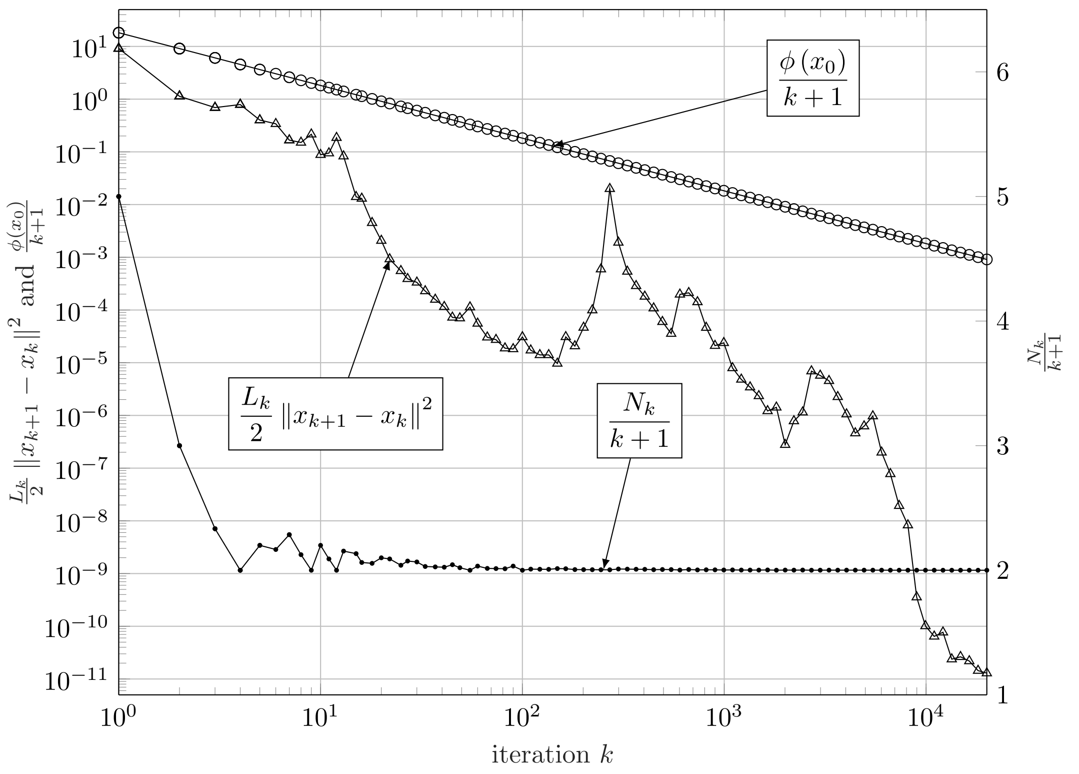


Figure 4.2 – Composite gradient method applied on problem (4.1) to find a "rank-at-most-6 approximation" of $\mathbf{T}_{222}$. We verify that the theoretical decrease (4.6) of the "scaled step-size" is satisfied in practice. The average number of times we compute steps `[02]` and `[03]` for a fixed k tends to a value lower or equal to 2, as predicted by the theory.

small") as a function of the iteration κ (in the previous section, we used "α" to index the decompositions obtained at the end of each phase; the index "κ" is used for indexing the successive iterates inside the composite gradient method). As we can observe in Fig. C.1, at the end of Phase 2, the solution contains only 36 non-zero values, among a total of 84 ($3 \times 4 \times 7$) entries in $\mathbf{A}, \mathbf{B}, \mathbf{C}$. It happens that, for small tensors like in the $\langle 2,2,2\rangle$ case, Phase 1 and Phase 2 are sufficient to compute a SD-sol; Phase 3 (used to induce discrete values) is not needed in this case. The resulting SD-sol is available in Table C.1.

Remark. For tensors larger than $\mathbf{T}_{222}$, we introduce a *Pre-phase 1* to enhance to global convergence of the algorithm, i.e. increase our chances that the algorithm converges to a F-PD (or not far from being a F-PD) of $\mathbf{T}_{mpn}$ starting from a random initial iterate. This pre-phase is easy to explain: suppose that, in Phase 1, we want to solve (4.1) with a certain value for ρ. Then, in Pre-phase 1, we try to solve the same problem (with the same value of ρ) except that the constraints (#1) are replaced with relaxed constraints:

$$\boxed{\begin{array}{lll} \text{subject to} & & \\ \hdashline & \mathbf{A}(i,r) \in [\texttt{lb}, \texttt{ub}] & \text{for all } i, r \\ (\#1') & \mathbf{B}(j,r) \in [\texttt{lb}, \texttt{ub}] & \text{for all } j, r \\ & \mathbf{C}(k,r) \in [\texttt{lb}, \texttt{ub}] & \text{for all } k, r \end{array}}\ .$$

For instance, we use $\texttt{lb} = -10$ and $\texttt{ub} = +10$ in Pre-phase 1 and, during Phase 1, we use constraints (#1) which are clearly equivalent to (#1') with $\texttt{lb} = -1$ and $\texttt{ub} = +1$. The solution $\mathbf{A}_{1'}, \mathbf{B}_{1'}, \mathbf{C}_{1'}$ of Pre-phase 1 is then used as the first iterate for Phase 1. Note that $\mathbf{A}_{1'}, \mathbf{B}_{1'}, \mathbf{C}_{1'}$ does not necessarily satisfies (#1). Hence, we use its projection on the set described by (#1), i.e. the elements of $\mathbf{A}_{1'}, \mathbf{B}_{1'}, \mathbf{C}_{1'}$ larger than $+1$ or lower than -1 are projected to $+1$ and -1 respectively.

Next, we look for a decomposition of $\mathbf{T}_{232}$ with 11 rank-at-most-1 terms. The procedure is the same as in the case of $\langle 2,2,2\rangle$ except that we use Pre-phase 1 introduced in the remark just above. The details are available in Fig. C.2 and the obtained decomposition is described in Table C.2. Like for the $\langle 2,2,2\rangle$ case, it happens that Phase 1 and Phase 2 are sufficient to find a sparse discrete solution. The decomposition contains 48 non-zero values, among a total of 176 ($2 \times 6 \times 11 + 4 \times 11$) entries in $\mathbf{A}, \mathbf{B}, \mathbf{C}$.

Finally, let us also mention that we tested our method on tensors smaller than $\mathbf{T}_{222}$: for example, we computed sparse discrete 2-PD's for the $\langle 1,2,1\rangle$ case and sparse discrete 4-PD's for the $\langle 2,1,2\rangle$ case. We noticed that for these small cases, Phase 2 was sufficient, i.e. if we directly start Phase 2 with a random initial guess, then we converge to a SD-sol. The figure available on the (second) front page illustrates the $\langle 2,1,2\rangle$ case. More precisely, it shows the matrices $\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa$ at four different moments of the composite gradient method applied on (4.1) with $\mathbf{X} = \mathbf{T}_{212}$, $F = 4$ and $\rho = 10^{-2}$.

4.2.2 Sparse discrete decompositions of larger tensors

Now we consider more difficult cases like $\langle 3, 2, 3\rangle$ and $\langle 3, 3, 3\rangle$. Those cases are more interesting since, this time, we need Phase 3 to obtain a discrete decomposition. First we are interested in $\langle 3, 2, 3\rangle$. The details of Phase 1 and Phase 2 are given in Fig. C.3. As "threshold" η for Phase 3, we use the value of 0.8. As we can observe in Table C.3, the resulting solution contains 102 non-zero values to be compared with a total of 315 $(2 \times 6 \times 15 + 9 \times 15)$ entries in $\mathbf{A}, \mathbf{B}, \mathbf{C}$.

Last but not least, we compute a 23-PD for the case of $\langle 3, 3, 3\rangle$. The details of Phase 1 and Phase 2 are given in Fig. C.4. We use a "threshold" η of 0.8 in Phase 3, to get a final decomposition containing 144 non-zero values out of the 621 $(3 \times 9 \times 23)$ possible entries in $\mathbf{A}, \mathbf{B}, \mathbf{C}$. The decomposition is available in Table C.4.

We have seen that using a "medium" value of ρ, like 10^{-2}, in Phase 2, allows us to obtain a decomposition which is both sparse and not far from being a F-PD of $\mathbf{T}_{mpn}$. We wonder what would happen if we use a larger value for ρ, let's say 10^{-1}. Therefore, we solve problem (4.1) with this ρ starting from the decomposition obtained after Pre-phase 1 and Phase 1 for $\mathbf{X} = \mathbf{T}_{333}$ and $F = 23$ (c.f. Fig. C.4) and observe what happens. The results and the details about the parameters are represented in Fig. C.5. We observe that, as expected, the algorithm converges to a solution which is sparser than in the case of a "medium" ρ. Indeed, this time, the factor matrices contain 134 non-zero values only, at the end of Phase 2. The drawback is that the obtained solution is not an "almost accurate" decomposition of $\mathbf{T}_{333}$ since $\|\mathbf{T}_{333} - [\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]\|_{\mathrm{L}^2} \approx 2.26$. Hence, this decomposition is not useful for building a matrix multiplication algorithm.

Finally, let us mention that we also tried to compute sparse discrete decompositions for tensors larger than $\mathbf{T}_{333}$. Unfortunately, we were not able to compute SD-sol's for them. It is even difficult to obtain an accurate decomposition with Pre-phase 1 and Phase 1. For example, in the case of $\langle 4, 3, 4\rangle$ with $F = 40$ (as far as we know, $F = 40$ is the smallest F for which we know [7, p. 7] that a F-PD of $\mathbf{T}_{434}$ exists), the best accuracy we could obtain after Pre-phase 1 and Phase 1 was $\|\mathbf{T}_{434} - [\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]\|_{\mathrm{L}^2} \approx 1$. We also tried a "relaxed problem": finding a sparse discrete decomposition of $\mathbf{T}_{434}$ with 41 rank-at-most-1 terms. In this case, we obtained a decomposition $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ satisfying $\|\mathbf{T}_{434} - [\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]\|_{\mathrm{L}^2} \approx 0.1$ after Pre-phase 1 and Phase 1. However, we did not achieve to obtain a sparse decomposition for which the entries of the factor matrices belong to $\{-1, 0, +1\}$ using Phase 2, Pre-phase 3 and Phase 3 and starting from $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$. Nevertheless, using a modified version of Pre-phase 3 and Phase 3 (c.f. the remark at the end of this section), we were able to compute a sparse 41-PD for the $\langle 4, 3, 4\rangle$ case for which the factor matrices have their entries in the set $\{-1, -1/2, 0, +1/2, +1\}$.

4.2.3 Comparison with two state-of-the-art methods

The results we obtain are not better than the results obtained by Smirnov in [6]. He proposes a decomposition for the $(3, 3)$ case with entries in $\{-1, 0, +1\}$ and 139 non-zero entries while the F-PD we present for the $(3, 3)$ case has 144 non-zero values. The strength of our method is its robustness, we had to use only a few random initial guesses to compute a SD-sol for the $(3, 3)$ case for example. We could thereby hope that, with more tries and a bit of chance, we will be able to find SD-sol's with less non-zero entries. It is difficult to compare the robustness of our procedure with that one of Smirnov's method since he does not give the detail of the computations in his paper. Let us just mention that his method implies at least 7 parameters that you need to update at the successive steps of the method and he does not give any indications on how do it. Hence, it is difficult to reproduce the results obtained by Smirnov.

We also compare our results with the SD-sol's obtained by Tichavský et al. in their paper [7]. The method is briefly described in Sec. 5.1. It seems that the largest case for which they computed a sparse discrete decomposition is the $\langle 3, 2, 3\langle$ case. The decomposition they propose contains values only in $\{-1, 0, +1\}$ and has 94 non-zero values while the decomposition we propose for the $\langle 3, 2, 3\langle$ case has 102 non-zero values. However, Tichavský et al. do not present decompositions for larger tensors.

Remark. We would like to make a final comment before the end of the chapter. We mentioned in the introduction of Sec. 4.1 that the algorithm could be modified to compute sparse decompositions with values in more general sets than $\{-1, 0, +1\}$. For example, if we would like to find sparse F-PD's with entries in $\{-1, -1/2, 0, +1/2, +1\}$, then we would have to modify Pre-phase 3 and Phase 3 in the following way: this time, we do not only fix the values that are close to ± 1 but also the values close to $\pm 1/2$. Nevertheless, we want to be careful when we fix values to $\pm 1/2$. Indeed, it might happen that some entries decrease from a "high value" toward 0 and that, during their decrease, they approach the value of $\pm 1/2$. Hence, we will have to fix value to $\pm 1/2$ only if have a strong intuition that this might be the right value and we should also be prepared to unfix some values if it happens that we cannot converge to an exact F-PD. Hence, the modified versions of Pre-phase 3 and Phase 3 require an "active participation" of the person who executed them and thus cannot be executed in an "automatic fashion" (in their original version, the choice of the "threshold" η totally determined how Pre-phase 3 and Phase 3 would be executed).

Using this modified algorithm, we computed a sparse 41-PD for the $\langle 4, 3, 4\rangle$ case with the entries of the factor matrices having values in $\{-1, -1/2, 0, +1/2, +1\}$. The decomposition is presented in Table C.5, Table C.6 and Table C.7. According to [7], the rank of the associated tensor is 40. Hence, we have one extra rank-at-most-1 term in our decomposition compared to the theoretical minimal decomposition.

Chapter 5

Discretizing decompositions through inv-transformations

5.1 Properties of discretizable decompositions

Recall (c.f. Table 3.1) that a F-PD $[\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$ of a matrix multiplication tensor is *discretizable* if it is (T-scale+T-trace)-equivalent to a discrete decomposition. In the first subsection, we explain why we are interested in discretizable solution or why it is interesting to know whether a solution could be discretized? In a second time, we give a necessary condition to be discretizable.

5.1.1 Motivation $\leftarrow$ the two-steps approach

We already mention why we are interested in discrete decompositions. In Sec. 4.1, we presented a method to compute SD-sol's. This was an "all-at-once" method in the sense that, before the end of the procedure, we have neither an exact decomposition or a completely discrete decomposition. At the end of Phase 1, the decomposition $[\![\mathbf{A}_1,\mathbf{B}_1,\mathbf{C}_1]\!]$ is almost a F-PD of $\mathbf{T}_{mpn}$. However, the "accuracy of the decomposition" is not necessarily preserved during Phase 2 and Phase 3, i.e.

$$\|\mathbf{T}_{mpn} - [\![\mathbf{A}_\alpha,\mathbf{B}_\alpha,\mathbf{C}_\alpha]\!]\|_{\mathrm{L}^2} \leq \epsilon \quad \not\Rightarrow \quad \|\mathbf{T}_{mpn} - [\![\mathbf{A}_{\alpha+1},\mathbf{B}_{\alpha+1},\mathbf{C}_{\alpha+1}]\!]\|_{\mathrm{L}^2} \leq \epsilon \,.$$

The two-steps approach is different. The idea is to first compute a "very accurate" decomposition $[\![\mathbf{A}_1,\mathbf{B}_1,\mathbf{C}_1]\!]$ of the matrix multiplication tensor and then, in a second time, use transformations T-scale and T-trace to transform $[\![\mathbf{A}_1,\mathbf{B}_1,\mathbf{C}_1]\!]$ into a discrete decomposition. A similar approach was used by Tichavský et al. in their paper [7]. First they use a Levenberg-Marquardt method (c.f. Appendix D for a description of the method) to compute an accurate F-PD of $\mathbf{T}_{mpn}$, i.e. $\|\mathbf{T}_{mpn} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_{\mathrm{L}^2} \leq \epsilon$ for some very small ϵ. Then they try to get a sparse decomposition by sequentially solving the following optimization problems:

$$\left\{\begin{array}{lll} \text{minimize} & \left\|\left(\mathbf{I}_p \otimes \mathbf{P}^{-1}\right)\cdot\mathbf{A}\right\|_{\mathrm{L}^1} + \left\|\left(\mathbf{P}^\top \otimes \mathbf{I}_n\right)\cdot\mathbf{C}\right\|_{\mathrm{L}^1} & \text{with variable } \mathbf{P}\in\mathrm{GL}\,(m) \\ \text{minimize} & \left\|\left(\mathbf{I}_n \otimes \mathbf{Q}^{-1}\right)\cdot\mathbf{B}\right\|_{\mathrm{L}^1} + \left\|\left(\mathbf{Q}^\top \otimes \mathbf{I}_m\right)\cdot\mathbf{A}\right\|_{\mathrm{L}^1} & \text{with variable } \mathbf{Q}\in\mathrm{GL}\,(p) \\ \text{minimize} & \left\|\left(\mathbf{I}_m \otimes \mathbf{R}^{-1}\right)\cdot\mathbf{C}\right\|_{\mathrm{L}^1} + \left\|\left(\mathbf{R}^\top \otimes \mathbf{I}_p\right)\cdot\mathbf{B}\right\|_{\mathrm{L}^1} & \text{with variable } \mathbf{R}\in\mathrm{GL}\,(n) \end{array}\right\}.$$

This sequence of optimizations is repeated until convergence is obtained.

We wonder whether we can go further and also obtain discrete values using only inv-transformations. Beside the difficulty to implement an algorithm for doing this, we first wonder how worth it would be. Let us explain what we mean: the processes to compute F-PD's are generally iterative processes and need a initial guess $[\![\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0]\!]$. In general, the starting guesses are chosen randomly. Hence, if the process converges to an accurate decomposition $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ of $\mathbf{T}_{mpn}$, this decomposition will be one decomposition among all possible F-PD's of $\mathbf{T}_{mpn}$. Before starting to try to make $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ discrete with transformations T-scale and T-trace, we would like to know whether there are any chances for it to be discretizable. We give, in the following subsection, a necessary condition for $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ to be discretizable. If the condition is not satisfied, it is pointless to try to transform $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ into a discrete decomposition. In Sec. 5.2, we will also try to answer the question "how often a decomposition $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ will satisfy the necessary condition if this decomposition is computed with the "Tichavský et al.'s LM-method" (abbreviation of the "Tichavský et al.'s Levenberg-Marquardt method to compute F-PD's") and starting from a random initial iterate $[\![\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0]\!]$?".

5.1.2 Characteristic polynomials of a F-PD

In the following, when we use the "reshape operator", we will not specify the size of the reshaped matrix, i.e. "$\mathrm{rshp}_{[\cdot,\cdot]}(\cdot)$" is abbreviated as "$\mathrm{rshp}(\cdot)$", since the size of $\mathrm{rshp}(\cdot)$ will be clear from the situation.

Let $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ be a F-PD of some matrix multiplication tensor $\mathbf{T}_{mpn}$ and suppose it is (T-scale+T-trace)-equivalent to a discrete F-PD $[\![\mathbf{A}^*, \mathbf{B}^*, \mathbf{C}^*]\!]$. Denote $\{\mathbf{a}_1, \dots, \mathbf{a}_F\}$, $\{\mathbf{b}_1, \dots, \mathbf{b}_F\}$ and $\{\mathbf{c}_1, \dots, \mathbf{c}_F\}$ the columns of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ respectively. Similarly, let $\{\mathbf{a}_1^*, \dots, \mathbf{a}_F^*\}$, $\{\mathbf{b}_1^*, \dots, \mathbf{b}_F^*\}$ and $\{\mathbf{c}_1^*, \dots, \mathbf{c}_F^*\}$ denote the columns of $\mathbf{A}^*$, $\mathbf{B}^*$ and $\mathbf{C}^*$ respectively. Then, for all $r \in \{1, \dots, F\}$, we have that

$$\left\{ \begin{array}{c} \mathrm{rshp}(\mathbf{a}_r) = \mathbf{P}^{-1} \cdot \lambda_r \cdot \mathrm{rshp}(\mathbf{a}_r^*) \cdot \mathbf{Q} \\ \\ \mathrm{rshp}(\mathbf{b}_r) = \mathbf{Q}^{-1} \cdot \mu_r \cdot \mathrm{rshp}(\mathbf{b}_r^*) \cdot \mathbf{R} \\ \\ \mathrm{rshp}(\mathbf{c}_r) = \mathbf{R}^{-1} \cdot \nu_r \cdot \mathrm{rshp}(\mathbf{c}_r^*) \cdot \mathbf{P} \end{array} \right\} .$$

for some matrices $\mathbf{P} \in \mathrm{GL}(m)$, $\mathbf{Q} \in \mathrm{GL}(p)$ and $\mathbf{R} \in \mathrm{GL}(n)$ and some scaling coefficients $\{\lambda_1, \dots, \lambda_F\}$, $\{\mu_1, \dots, \mu_F\}$ and $\{\nu_1, \dots, \nu_F\}$ satisfying $\lambda_r \mu_r \nu_r = 1$ for all $r \in \{1, \dots, F\}$. Now define the (mp, mp) matrices $\mathbf{M}_r := \mathrm{rshp}(\mathbf{a}_r) \cdot \mathrm{rshp}(\mathbf{b}_r) \cdot \mathrm{rshp}(\mathbf{c}_r)$ and

$\mathbf{M}_r^* \coloneqq \operatorname{rshp}(\mathbf{a}_r^*) \cdot \operatorname{rshp}(\mathbf{b}_r^*) \cdot \operatorname{rshp}(\mathbf{c}_r^*)$ and observe that

$$\left\{ \begin{array}{c} \operatorname{rshp}(\mathbf{a}_r) \cdot \operatorname{rshp}(\mathbf{b}_r) \cdot \operatorname{rshp}(\mathbf{c}_r) \\ = \\ \left[\mathbf{P}^{-1} \cdot \lambda_r \cdot \operatorname{rshp}(\mathbf{a}_r^*) \cdot \mathbf{Q}\right] \cdot \left[\mathbf{Q}^{-1} \cdot \mu_r \cdot \operatorname{rshp}(\mathbf{b}_r^*) \cdot \mathbf{R}\right] \cdot \left[\mathbf{R}^{-1} \cdot \nu_r \cdot \operatorname{rshp}(\mathbf{c}_r^*) \cdot \mathbf{P}\right] \\ = \\ \mathbf{P}^{-1} \cdot \left[\operatorname{rshp}(\mathbf{a}_r^*) \cdot \operatorname{rshp}(\mathbf{b}_r^*) \cdot \operatorname{rshp}(\mathbf{c}_r^*)\right] \cdot \mathbf{P} \end{array} \right\} .$$

Hence, this shows that, for each $r \in \{1, \ldots, F\}$, $\mathbf{M}_r$ and $\mathbf{M}_r^*$ are similar matrices and thus they must have the same characteristic polynomial. If $\mathbf{A}^*, \mathbf{B}^*, \mathbf{C}^*$ contain only entries belonging to $\{-1, 0, +1\}$ for example, then $\operatorname{charpoly}(\mathbf{M}_r^*)(t)$, the characteristic polynomial of $\mathbf{M}_r^*$, must contain only integer coefficients and thus so does $\operatorname{charpoly}(\mathbf{M}_r)(t)$. Similar comments hold if the entries of $\mathbf{A}^*, \mathbf{B}^*, \mathbf{C}^*$ belong to other sets consisting of a small number of rational values. This must be treated case-by-case. Clearly, the more diverse are those values, the more difficult it becomes to say something about the characteristic polynomial of the $\mathbf{M}_r$'s. But, in general, we only interest in sets like $\{-1, 0, +1\}$ or $\{-1, -1/2, 0, +1/2, +1\}$ for example, then there is a reduced number of possibilities for the coefficients of the $\operatorname{charpoly}(\mathbf{M}_r)(t)$'s.

Remark. In the developments above, we have used the (mp, mp) matrices $\mathbf{M}_r \coloneqq \operatorname{rshp}(\mathbf{a}_r) \cdot \operatorname{rshp}(\mathbf{b}_r) \cdot \operatorname{rshp}(\mathbf{c}_r)$. Would it be interesting to consider the (pn, pn) matrices $\mathbf{M}_r' \coloneqq \operatorname{rshp}(\mathbf{b}_r) \cdot \operatorname{rshp}(\mathbf{c}_r) \cdot \operatorname{rshp}(\mathbf{a}_r)$ or the (nm, nm) matrices $\mathbf{M}_r'' \coloneqq \operatorname{rshp}(\mathbf{c}_r) \cdot \operatorname{rshp}(\mathbf{a}_r) \cdot \operatorname{rshp}(\mathbf{b}_r)$ as well? In fact, this would not bring extra information for the simple reason that $\mathbf{M}_r$, $\mathbf{M}_r'$ and $\mathbf{M}_r''$ have more or less the same characteristic polynomial. This comes from the following property of the characteristic polynomial: if $\mathbf{X} \in \mathbb{R}^{I \times J}$ and $\mathbf{Y} \in \mathbb{R}^{J \times I}$, then

$$t^J \cdot \operatorname{charpoly}(\mathbf{XY})(t) \;=\; t^I \cdot \operatorname{charpoly}(\mathbf{YX})(t) \;.$$

The interested reader can find a proof in [14].

5.2 Distribution of discretizable decompositions

We would like to have an idea of the probability for a "randomly computed" decomposition $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ of $\mathbf{T}_{mpn}$ to be discretizable. By "randomly computed", we mean that: we first define initial guesses $\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0$ with entries chosen randomly uniform between -1 and $+1$. Then we compute $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ with Tichavský et al.'s LM-method.

In the first subsection, we describe a procedure to compute statistics about the characteristic polynomials of "randomly computed" F-PD's. Those will help us to analyze the distribution of discretizable solutions. Before comparing the characteristic polynomials of two different F-PD's, we want to get rid of the (T-perm)-equivalence that might exist between the two decompositions. We explain in the second subsection how we computationally get rid of the permutation invariance without trying every permutations, which would be too expensive. In the last subsection, we present the results and we will draw the following conclusions: for the cases $(1,2,1)$, $(2,1,2)$, $(2,2,2)$ and $(3,2,3)$, 100% of the decompositions satisfy the necessary condition to be discretizable. For the $(2,3,2)$ case and the $(3,3,3)$ case, this percentage drops to respectively 56% and 79% of the decompositions that are eligible to be discretizable.

5.2.1 Statistics on the characteristic polynomials

In order to have an idea of the distribution of discretizable solutions, we make the following procedure: define random starting guesses $\mathbf{A}^{\text{init}}_{\kappa'}, \mathbf{B}^{\text{init}}_{\kappa'}, \mathbf{C}^{\text{init}}_{\kappa'}$ for $\kappa' \in \{1,2,3,\ldots\}$ and apply Tichavský et al.'s LM-method. The method converge to the matrices $\mathbf{A}_{\kappa'}, \mathbf{B}_{\kappa'}, \mathbf{C}_{\kappa'}$ for $\kappa' \in \{1,2,3,\ldots\}$. Sometimes the method does not converge to an exact F-PD. In this case, the solution is rejected. At the end, we keep 100 exact F-PD's of $\mathbf{T}_{mpn}$:

$$\left\{ [\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!] \text{ are } F\text{-PD's of } \mathbf{T}_{mpn} \;\middle|\; \kappa \in \{1,\ldots,100\} \right\} . \tag{5.1}$$

Remark. At this point, the reader might wonder the two following questions: "why we are interested in "randomly computed" decompositions instead of truly random decompositions?" and "why we use the Tichavský et al.'s LM-method to compute the F-PD's instead of another method?". The answer to the first question lies in the fact that, if we want to speak about a "random decomposition", then we first have to define a probability measure on the set of all decompositions of a given tensor. We could define such a probability measure but, since none of them seems very natural, we do not have a real interest to do this. The only probability measure that seems natural for our purposes (computing discrete solutions with the two-steps approach), is the probability measure inherent to the "random computation" of F-PD's with the Tichavský et al.'s LM-method or any other method. This leads us to the second question: here the answer is still more subjective. The fact is that we think that Tichavský et al.'s LM-method is very powerful for computing accurate decompositions. Hence, for the implementation the two-steps approach (c.f. the remark at the end of this section explaining why we do not present an implementation of the two-steps approach), we would most likely use the Tichavský et al.'s LM-method for the first step (the step during which we want to obtain an accurate decomposition).

For each $\kappa \in \{1,\ldots,100\}$, the columns of $\mathbf{A}_\kappa$, $\mathbf{B}_\kappa$ and $\mathbf{C}_\kappa$ are respectively denoted by $\{\mathbf{a}^\kappa_1,\ldots,\mathbf{a}^\kappa_F\}$, $\{\mathbf{b}^\kappa_1,\ldots,\mathbf{b}^\kappa_F\}$ and $\{\mathbf{c}^\kappa_1,\ldots,\mathbf{c}^\kappa_F\}$. Then, for each $\kappa \in \{1,\ldots,100\}$ and for each $r \in \{1,\ldots,F\}$, we define the characteristic polynomial

$$p^\kappa_r(t) := \operatorname{charpoly}\left[\operatorname{rshp}\left(\mathbf{a}^\kappa_r\right) \cdot \operatorname{rshp}\left(\mathbf{b}^\kappa_r\right) \cdot \operatorname{rshp}\left(\mathbf{c}^\kappa_r\right)\right](t) \ .$$

For each $\kappa \in \{1,\ldots,100\}$, we consider the array of characteristic polynomials

$$\mathcal{P}^\kappa := [\ p^\kappa_1(t)\ ,\ p^\kappa_2(t)\ ,\ \ldots\ ,\ p^\kappa_F(t)\ ]$$

associated to the decomposition $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$. We compute how many different $\mathcal{P}^\kappa$'s occur when κ varies in $\{1,\ldots,100\}$. Note that a transformation T-perm of the factor matrices

$\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa$ would result in a permutation of the polynomials of $\mathcal{P}^\kappa$. We get rid of this equivalence by considering that two arrays $\mathcal{P}^{\kappa_1}$ and $\mathcal{P}^{\kappa_2}$ are different if and only if they are different for every permutation of their polynomials. We explain how we computationally achieve this in the next subsection.

Evidently, all our computations are done in floating-point arithmetic. Hence, we need to specify "within what tolerance, is a decomposition considered as exact?" and also "within what tolerance, do we consider two arrays $\mathcal{P}^{\kappa_1}$ and $\mathcal{P}^{\kappa_2}$ as identical?". In our computations, we consider a decomposition as an exact F-PD of $\mathbf{T}_{mpn}$ if the L^∞-norm of the difference is smaller than 10^{-8}, i.e. for all $\kappa \in \{1, \ldots, 100\}$:

$$\|\mathbf{T}_{mpn} - [\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]\|_{\mathrm{L}^\infty} \leq 10^{-8} .$$

For each $\kappa \in \{1, \ldots, 100\}$ and each $r \in \{1, \ldots, F\}$, let $\left\{\alpha^\kappa_{r,0}, \ldots, \alpha^\kappa_{r,m}\right\}$ be the coefficients of the polynomial $p^\kappa_r(t)$. Then, we define the distance between two of those polynomials, let's say $p^{\kappa_1}_{r_1}(t)$ and $p^{\kappa_2}_{r_2}(t)$, as follows:

$$\operatorname{dist}\left(p^{\kappa_1}_{r_1}(t), p^{\kappa_2}_{r_2}(t)\right) := \max\left\{ \left|\alpha^{\kappa_1}_{r_1,q} - \alpha^{\kappa_2}_{r_2,q}\right| \,\middle|\, 0 \leq q \leq m \right\}$$

The distance between two ordered arrays of polynomials is defined as

$$\operatorname{dist}\left(\mathcal{P}^{\kappa_1}, \mathcal{P}^{\kappa_2}\right) := \max\left\{ \operatorname{dist}\left(p^{\kappa_1}_{r}(t), p^{\kappa_2}_{r}(t)\right) \,\middle|\, 1 \leq r \leq F \right\}$$

Let `tol` > 0 be our parameter for the comparison of the $\mathcal{P}^\kappa$'s. We consider that two arrays $\mathcal{P}^{\kappa_1}$ and $\mathcal{P}^{\kappa_2}$ are identical if and only if there exists a permutation of the $p^{\kappa_2}_r(t)$'s such that the "permuted" $\mathcal{P}^{\kappa_2}$ provides $\operatorname{dist}\left(\mathcal{P}^{\kappa_1}, \mathcal{P}^{\kappa_2}\right) \leq$ `tol`. Based on this "being identical" criterion between the arrays of polynomials, the 100 $\mathcal{P}^\kappa$'s will be partitioned into a certain number of classes as follows: the arrays $\mathcal{P}^\kappa$ are the nodes of some graph $\mathcal{G}$ and there is an edge between two nodes $\mathcal{P}^{\kappa_1}$ and $\mathcal{P}^{\kappa_2}$ if and only if they are identical within `tol`. Then the classes correspond to the connected components of $\mathcal{G}$, i.e. $\mathcal{P}^{\kappa_1}$ and $\mathcal{P}^{\kappa_2}$ are in the same class if and only if there exists a sequence of arrays $\mathcal{P}^{\kappa_s}$ starting at $\mathcal{P}^{\kappa_1}$ and ending $\mathcal{P}^{\kappa_2}$ and such that each array is identical (within `tol`) to the next one.

The results of our experiments are gathered in Table 5.1. The first column indicates which multiplication tensor we consider and the number of rank-at-most-1 terms in the polyadic decomposition. The second column contains two sub-columns. The first one specifies which parameter we used for the Tichavský et al.'s LM-method (c.f. Appendix D for a description of the method). Note that, if we are not interested in the Tichavský et al.'s LM-method, this sub-column is not very relevant. Yet we prefer to clarify which parameter we used since this might have an impact on the distribution of the "randomly computed" decompositions $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$. The same comment holds for the third column. This column indicates the number of tries $\mathbf{A}_{\kappa'}, \mathbf{B}_{\kappa'}, \mathbf{C}_{\kappa'}$ that where needed to compute 100 exact decompositions. Hence, this column just gives an idea of the global convergence of the Tichavský et al.'s LM-method for the different cases. The second sub-column of the

Multiplication and number of terms in the F-PD		Parameters		# initial guesses needed	population of the 5 most populated classes
$\langle 1,2,1\rangle$	$F=2$	$\theta=20$	`tol` $=10^{-9}$	100	$\{100\,,0\,,0\,,0\,,0\}$
$\langle 2,1,2\rangle$	$F=4$	$\theta=50$	`tol` $=10^{-9}$	100	$\{100\,,0\,,0\,,0\,,0\}$
$\langle 2,2,2\rangle$	$F=7$	$\theta=100$	`tol` $=10^{-9}$	108	$\{100\,,0\,,0\,,0\,,0\}$
$\langle 2,3,2\rangle$	$F=11$	$\theta=120$	`tol` $=10^{-2}$	100	$\{56\,,2\,,1\,,1\,,1\}$
$\langle 3,2,3\rangle$	$F=15$	$\theta=140$	`tol` $=10^{-9}$	190	$\{94\,,6\,,0\,,0\,,0\}$
$\langle 3,3,3\rangle$	$F=23$	$\theta=150$	`tol` $=10^{-2}$	149	$\{79\,,2\,,1\,,1\,,1\}$

Table 5.1 – Distributions of the characteristic polynomials.

second column indicates which tolerance `tol` we used for deciding whether two arrays of polynomials are considered as equal or not.

Remark. Note that the larger is the tolerance `tol`, the more edges there will be in $\mathcal{G}$ and thus the less number of different classes of arrays of polynomials there will be. Hence, in Table 5.1, we have used two kinds of `tol`: (a) we used small `tol`'s (10^{-9}) when the arrays $\mathcal{P}^{\kappa}$ can be grouped in a small number of classes. The small value of `tol` proves that those classes are effectively very narrow; (b) we used larger values (10^{-2}) for `tol` when there are many classes of $\mathcal{P}^{\kappa}$'s. With those high `tol`'s, we see that, even if characteristic polynomials having a "large" distance between them to are considered as identical, then we still have many different classes of $\mathcal{P}^{\kappa}$'s.

5.2.2 Implementation $\to$ getting rid of T-perm

We explain how we check whether two arrays $\mathcal{P}^{\kappa_1}$ and $\mathcal{P}^{\kappa_2}$ are identical within the tolerance `tol` without trying every permutation of the $p_r^{\kappa}(t)$'s. Define the following (F,F) matrix which computes pair-wise the polynomial distance:

$$\mathbf{G}\,(r_1,r_2) := \operatorname{dist}\left(p_{r_1}^{\kappa_1}(t)\,,p_{r_2}^{\kappa_2}(t)\right)$$

for each $r_1,r_2 \in \{1,\ldots,F\}$. Consider the bipartite graph $\mathcal{G}$ consisting of two distinct sets of nodes $\mathcal{U}^1 := \{u_1^1,\ldots,u_F^1\}$ and $\mathcal{U}^2 := \{u_1^2,\ldots,u_F^2\}$ and where the weight of each edge between a node $u_{r_1}^1 \in \mathcal{U}^1$ and a node $u_{r_2}^2 \in \mathcal{U}^2$ is given by $\mathbf{G}\,(r_1,r_2)$. A *perfect matching* in $\mathcal{G}$ is a subset $\mathcal{M}$ of the edges of $\mathcal{G}$ such that each node is incident to one and only one edge of $\mathcal{M}$. The "max-cost" of the matching is the maximal weight of the edges in $\mathcal{M}$.

Finding a permutation of the polynomials in $\mathcal{P}^{\kappa_2}$ that minimizes the distance between $\mathcal{P}^{\kappa_1}$ and the permuted $\mathcal{P}^{\kappa_2}$ can be rephrased as a problem of finding a perfect matching $\mathcal{M}$ in $\mathcal{G}$ such that the "max-cost" of $\mathcal{M}$ is minimal. This problem is also called the "Linear

Bottleneck Assignment Problem". We solve the "Linear Bottleneck Assignment Problem" with the method proposed in [15]. The method consists of applying several times the "Hungarian Algorithm" on the graph $\mathcal{G}'$ defined as the graph $\mathcal{G}$ where only the edges with the smallest weights are conserved (the other edges get a infinite weight). A comprehensive description of the "Hungarian Algorithm" can be found in [16] and, for the implementation, we relied on [17].

5.2.3 Case-by-case analysis and conclusions

We will analyze in more details each case presented in Table 5.1. For each case, let D be the number of classes of $\mathcal{P}^\kappa$'s, i.e. the number of connected components in the associated graph, we have obtained. Those classes are denoted by $\mathcal{C}_\eta$ where the index η varies in $\{1, \dots, D\}$. For certain classes, we would like to exhibit a $\mathcal{P}^\eta_*$ that will be "representative" for this class. Clearly, being "representative" is not well-defined but we will see, in the case-by-case analysis below, that the most populated classes (c.f. Table 5.1) contain $\mathcal{P}^\kappa$'s for which the polynomials $p^\kappa_r(t)$ have integer coefficients. Hence, in this case, it is not difficult to point out what is a "representative" array $\mathcal{P}^\eta_*$ for the class. Moreover, to show that this $\mathcal{P}^\eta_*$ is indeed "representative" for $\mathcal{C}_\eta$, we also provide the maximal deviation of the class from this "representative" array, i.e. the maximal distance between $\mathcal{P}^\eta_*$ and an array $\mathcal{P}^\kappa$ of $\mathcal{C}_\eta$.

First, let us consider the $\langle 1, 2, 1\rangle$ case with $F = 2$. As indicated in Table 5.1, all the polynomial arrays $\mathcal{P}^\kappa$ are in the same class. Hence, we just have to consider $\mathcal{C}_1$. Since `tol` is very small, this class is very narrow, in the sense that all the $\mathcal{P}^\kappa$'s are identical to each other within a tolerance of $100 \cdot$`tol` (since the longest path between two nodes has a length of at most 100). Concretely, this gives the following results:

$\mathcal{C}_\eta$	polynomials $p_r(t)$ appearing in some $\mathcal{P}^\eta_*$ "representative" for $\mathcal{C}_\eta$	number of $\mathcal{P}^\kappa$'s in $\mathcal{C}_\eta$	maximal distance between $\mathcal{P}^\eta_*$ and any $\mathcal{P}^\kappa$ of $\mathcal{C}_\eta$
$\mathcal{C}_1$	$t - 1$ $(\times 02)$	100	0.000e+00

We observe that the $p^\kappa_r(t)$'s contain integer coefficients. Hence, this suggests (do not forget that the criterion is a necessary condition and not a sufficient condition) that, if we "randomly compute" a decomposition $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ with the Tichavský et al.'s LM-method, then this decomposition will be discretizable with probability one. In fact, we will see in Sec. 6.2 that there is only a unique algorithm (up to inv-equivalence) for computing $\langle 1, 2, 1\rangle$ and this algorithm uses factor matrices with values only in $\{-1, 0, +1\}$. Hence, for this case, we are sure that every decomposition is discretizable. This is also a way to check that our procedure gives exploitable results.

We do the same steps for the $\langle 2, 1, 2\rangle$ case and $F = 4$. Again, there is only one class of $\mathcal{P}^\kappa$'s and we give a "representative" $\mathcal{P}^1_*$ for this class. The polynomials appearing in $\mathcal{P}^1_*$

are listed below:

$\mathcal{C}_\eta$	polynomials $p_r(t)$ appearing in some $\mathcal{P}_*^\eta$ "representative" for $\mathcal{C}_\eta$	number of $\mathcal{P}^\kappa$'s in $\mathcal{C}_\eta$	maximal distance between $\mathcal{P}_*^\eta$ and any $\mathcal{P}^\kappa$ of $\mathcal{C}_\eta$
$\mathcal{C}_1$	$t^2 - t$ $(\times 04)$	100	1.699e-13

This is more interesting since, as we will see in Sec. 6.2, there are more than one algorithm for computing $\langle 2, 1, 2\rangle$. Nevertheless, the results above suggest that all "randomly computed" decompositions have the same characteristic polynomials with probability one. Moreover, those polynomials have integer coefficients so that we may suppose that the decompositions are discretizable with probability one as well.

For $\langle 2, 2, 2\rangle$ with $F = 7$, there is again only one class of $\mathcal{P}^\kappa$'s and they contain the following polynomials (up to the maximal deviation of the class):

$\mathcal{C}_\eta$	polynomials $p_r(t)$ appearing in some $\mathcal{P}_*^\eta$ "representative" for $\mathcal{C}_\eta$	number of $\mathcal{P}^\kappa$'s in $\mathcal{C}_\eta$	maximal distance between $\mathcal{P}_*^\eta$ and any $\mathcal{P}^\kappa$ of $\mathcal{C}_\eta$
$\mathcal{C}_1$	$t^2 - 2t + 1$ $(\times 01)$ $t^2 - t$ $(\times 06)$	100	1.202e-10

The same comment holds as for the $\langle 1, 2, 1\rangle$ case: we will see in Sec. 6.2 that every 7-PD of $\mathbf{T}_{222}$ is unique up to inv-transformations. Considering the decomposition (3.1), we observe that this decomposition is discrete so that every decomposition is discretizable.

The case of $\langle 3, 2, 3\rangle$ with $F = 15$ is more interesting. Indeed, we observe that there are two and only two classes of $\mathcal{P}^\kappa$'s. The first class is the most populated. Let $\mathcal{P}_*^1$ be a "representative" array for the first class and let $\mathcal{P}_*^2$ be a "representative" array for the second class. The polynomials appearing in $\mathcal{P}_*^1$ and $\mathcal{P}_*^2$ as well as the maximal deviation of each class from its “representative” are listed below:

$\mathcal{C}_\eta$	polynomials $p_r(t)$ appearing in some $\mathcal{P}_*^\eta$ “representative” for $\mathcal{C}_\eta$	number of $\mathcal{P}^\kappa$'s in $\mathcal{C}_\eta$	maximal distance between $\mathcal{P}_*^\eta$ and any $\mathcal{P}^\kappa$ of $\mathcal{C}_\eta$
$\mathcal{C}_1$	t^3-2t+t (×03) t^3-t^2 (×12)	094	1.292e-11
$\mathcal{C}_2$	t^3-2t (×01) t^3-2t+t (×02) t^3-t^2 (×12)	006	1.072e-12

Now let us consider the $\langle 2,3,2\rangle$ case with $F = 11$ and also the $\langle 3,3,3\rangle$ case with $F = 23$. We can make more or less the same comment for those two cases: there are many different classes of $\mathcal{P}^\kappa$'s even though we have used a larger value of `tol` (10^{-2}) for considering two $\mathcal{P}^\kappa$'s as belonging to the same class:

Multiplication and number of terms in the F-PD		Number of different classes
$\langle 2,3,2\rangle$	$F=11$	44
$\langle 3,3,3\rangle$	$F=23$	21

Nevertheless, for those two cases, the most populated class, let's say $\mathcal{C}_1$, still contains a “representative” $\mathcal{P}_*^1$ for which the polynomials have integer coefficients only. More precisely, we have for $\langle 2,3,2\rangle$ and $F = 11$, the following $p_r(t)$'s in $\mathcal{P}_*^1$:

$\mathcal{C}_\eta$	polynomials $p_r(t)$ appearing in some $\mathcal{P}_*^\eta$ “representative” for $\mathcal{C}_\eta$	number of $\mathcal{P}^\kappa$'s in $\mathcal{C}_\eta$	maximal distance between $\mathcal{P}_*^\eta$ and any $\mathcal{P}^\kappa$ of $\mathcal{C}_\eta$
$\mathcal{C}_1$	t^2-2t+1 (×01) t^2-t (×10)	056	9.011e-03

Finally, the case of $\langle 3,3,3\rangle$ with $F = 23$ provides

$\mathcal{C}_\eta$	polynomials $p_r(t)$ appearing in some $\mathcal{P}^\eta_*$ "representative" for $\mathcal{C}_\eta$	number of $\mathcal{P}^\kappa$'s in $\mathcal{C}_\eta$	maximal distance between $\mathcal{P}^\eta_*$ and any $\mathcal{P}^\kappa$ of $\mathcal{C}_\eta$
$\mathcal{C}_1$	$t^3 - 2t + t$ (×04) $t^3 - t^2$ (×19)	079	1.395e-02

Clearly, the maximal deviation of the $\mathcal{C}_1$'s from their "representative" $\mathcal{P}^1_*$ is larger than in the previous cases. This comes from the fact that we have used a larger `tol` for considering two $\mathcal{P}^\kappa$'s as identical.

It is difficult to draw further conclusions. Let us just mention that, for the cases when there are many different classes of $\mathcal{P}^\kappa$'s, i.e. the two last considered cases, the arrays $\mathcal{P}^\kappa$ that are not in the most populated class, i.e. the arrays we have not described just above, contains very diverse values that are clearly not integers or something close to coefficients that might emerge if we consider factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ with entries in $\{-1, -1/2, 0, +1/2, +1\}$ for example. Hence, we suppose that those decompositions are not discretizable.

Remark. We have seen that, for the $(1,2,1)$, $(2,1,2)$, $(2,2,2)$ and $(3,2,3)$ cases, the decompositions seem to satisfy the necessary condition to be discretizable with probability one. Hence, a natural question arises: why did we not implement the two-steps approach (for computing SD-sol's), at least for those four cases? The answer is that we tried to implement such an algorithm. For the first step, we used the Tichavský et al.'s LM-method and thus we had to find a way to implement the second step: play on the variables $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ and $\{\lambda_1, \ldots, \lambda_F\}$, $\{\mu_1, \ldots, \mu_F\}$ and $\{\nu_1, \ldots, \nu_F\}$ to transform the F-PD into a sparse discrete decomposition. None of the algorithms we tried gave good enough results so that we chose to not present them in this report.

Hence, we might also wonder why we have kept and presented this chapter since the motivation to compute the characteristic polynomials of the F-PD's was to know whether we should or not implement the two-steps approach. In fact, we think that the results are still interesting: for each of the considered cases, the vast majority of the decompositions has the same characteristic polynomials. We believe that those results might be interesting for further research.

Chapter 6

Analysis of the inv-equivalence classes of decompositions

6.1 Computing the inv-transformations joining two F-PD's

In this section, we first introduce a novel concept: the $\oplus$-rank of a matrix. In Sec. 6.2, we will see that the factor matrices of "randomly computed" F-PD's for tensors larger or with the same size as $\mathbf{T}_{222}$ seem to have a $\oplus$-rank of 1 with probability one. Before that, in the current section, we present an algorithm for deciding whether two such F-PD's (hence, for which the factor matrices have a $\oplus$-rank of 1) $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ are inv-equivalent. The algorithm provides the inv-transformations joining the two F-PD's or conclude that they are not inv-equivalent.

6.1.1 Preliminaries $\rightarrow$ the $\oplus$-rank of a matrix

Let $\mathbf{A}$ be a (m, n) matrix. Let $\{\mathcal{U}_1, \ldots, \mathcal{U}_S\}$ be a family of linearly independent non-trivial sub-spaces of $\mathbb{R}^m$, i.e. the $\mathcal{U}_s$'s are different from $\{0\}$ and $\mathcal{U}_{s_1} \cap \mathcal{U}_{s_2} = \{0\}$ whenever $s_1 \neq s_2$. If each column of $\mathbf{A}$ belongs to one of the $\mathcal{U}_s$'s (in fact, excepting the case where the column contains only zeros, it may belong to at most one of the $\mathcal{U}_s$'s since they are linearly independent), then we say that $\{\mathcal{U}_1, \ldots, \mathcal{U}_S\}$ is a *covering* of $\mathbf{A}$. The maximal integer S^* such that there exists a covering of $\mathbf{A}$ with S^* linearly independent non-trivial sub-spaces is called the *$\oplus$-rank* of a $\mathbf{A}$ and is denoted $\operatorname{rank}_\oplus(\mathbf{A}) = S^*$ (the choice of the symbol "$\oplus$" comes from the fact that, if $\{\mathcal{U}_1, \ldots, \mathcal{U}_S\}$ is a "maximal" covering, then $\mathcal{U}_1 \oplus \cdots \oplus \mathcal{U}_S = \mathbb{R}^m$).

Let $\mathcal{U}_1 := \operatorname{colspan}(\mathbf{A})$ and suppose that the rank of $\mathbf{A}$ is r. Then it is easy to build a covering $\left\{\mathcal{U}_1, \hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_{m-r}\right\}$ where the $\hat{\mathcal{U}}_s$'s have dimension one. Hence, we can conclude that $\operatorname{rank}(\mathbf{A}) + \operatorname{rank}_\oplus(\mathbf{A}) \geq m + 1$.

Suppose that $\mathbf{A}$ has full row-rank, i.e. $\operatorname{rank}(\mathbf{A}) = m$. We give a characterization of the $\oplus$-rank of $\mathbf{A}$ in terms of the connected components of a graph. Denote by $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ the columns of $\mathbf{A}$. Without loss of generality, we may assume that the m first columns of $\mathbf{A}$ span $\mathbb{R}^m$. Those columns will be the nodes of our graph. Let $\mathbf{A}' := \mathbf{A}(:, 1\!:\!m)$ and $\mathbf{A}'' := \mathbf{A}(:, m\!+\!1\!:\!n)$, the columns of $\mathbf{A}''$ will be used to define the edges of the graph. For each column $\mathbf{a}_{j''}$ of $\mathbf{A}''$, compute its coordinates $\mathbf{q}_{j''} := \left(q_1^{j''}, \ldots, q_m^{j''}\right)^\top$ in terms of the m first columns of $\mathbf{A}'$, i.e. $\mathbf{q}_{j''} = [\mathbf{A}']^{-1} \mathbf{a}_{j''}$. For each $j_1', j_2' \in \{1, \ldots, m\}$, draw an edge between the nodes j_1' and j_2' if and only if there is a column $\mathbf{a}_{j''}$ of $\mathbf{A}''$ such that its

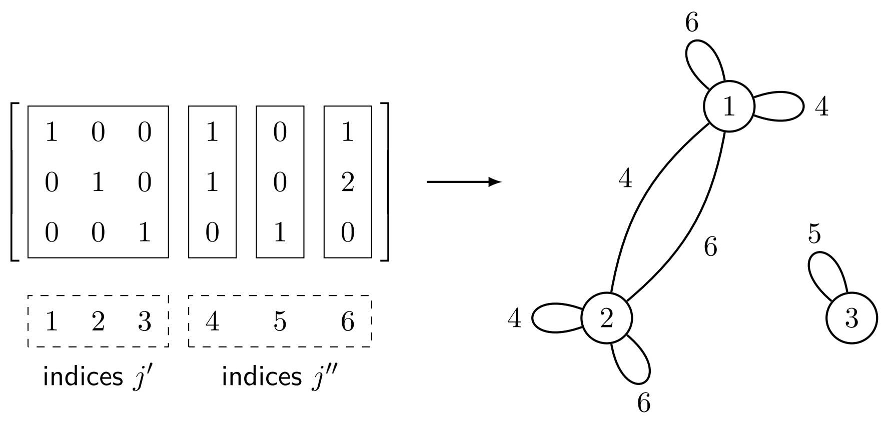


Figure 6.1 – Example of matrix $\mathbf{A}$ and the associated graph $\mathcal{G}$.

coordinates vector has a non-zero component in both $\mathbf{A}\left(:,j_1'\right)$ and $\mathbf{A}\left(:,j_2'\right)$, i.e. if and only if there exists a $j'' \in \{m+1,\ldots,n\}$ such that both $q_{j_1'}^{j''}$ and $q_{j_2'}^{j''}$ are non-zero. Clearly, there might be multiple edges and also loops. Moreover, each edge receives a "label": this "label" is simply j'', the index of the column of $\mathbf{A}''$ that led to this edge. We denote this graph by $\mathcal{G}$. An example is represented on Fig. 6.1. This construction allows us to state the following lemma:

Lemma 6.1. *Let* $\mathbf{A}$ *and* $\mathcal{G}$ *be defined as explained above. Then the* $\oplus$*-rank of* $\mathbf{A}$ *is equal to the number of connected components of* $\mathcal{G}$*.*

Proof. Let $\{\mathcal{G}_1,\ldots,\mathcal{G}_T\}$ be the connected components of $\mathcal{G}$. Now, for each $t \in \{1,\ldots,T\}$, let I_t be the set of all column indices involved in the nodes of $\mathcal{G}_t$. For example, considering the graph in Fig. 6.1, we would have $I_1 = \{1,2\}$ and $I_2 = \{3\}$. For each $t \in \{1,\ldots,T\}$, define the sub-space $\mathcal{U}_t$ as the sub-space spanned by the columns of $\mathbf{A}$ with their index in I_t, i.e. $\mathcal{U}_t := \operatorname{colspan}\left[\mathbf{A}\left(:,I_t\right)\right]$. By hypothesis on $\mathbf{A}'$, i.e. $\mathbf{A}'$ has full rank, the sub-spaces $\mathcal{U}_t$ are linearly independent. We have to show that each column of $\mathbf{A}''$ belongs to at least one of the $\mathcal{U}_t$'s.

Let $j'' \in \{m+1,\ldots,n\}$, we show that the "label" j'' appears in the edges of at most one component $\mathcal{G}_t$. Indeed, if j'' appears in $\mathcal{G}_{t_1}$ and $\mathcal{G}_{t_2}$, then $\mathbf{a}_{j''}$ has a non-zero component in at least one node j_1' of $\mathcal{G}_{t_1}$ and one node j_2' of $\mathcal{G}_{t_2}$. Hence, there must be an edge between j_1' and j_2' and thus $\mathcal{G}_{t_1}$ and $\mathcal{G}_{t_2}$ are not distinct components. Hence, j'' appears in at most one $\mathcal{G}_t$ and thus, by hypothesis on $\mathbf{A}'$ spanning $\mathbb{R}^m$, we have $\mathbf{a}_{j''} \in \mathcal{U}_t$. Hence, we have shown that $\{\mathcal{U}_1,\ldots,\mathcal{U}_T\}$ is a covering of $\mathbb{R}^m$ and thus $\operatorname{rank}_\oplus(\mathbf{A}) \geq T$.

Reversely, let $S := \operatorname{rank}_\oplus(\mathbf{A})$ and let $\{\mathcal{U}_1,\ldots,\mathcal{U}_S\}$ be a covering of $\mathbb{R}^m$. For each $s \in \{1,\ldots,S\}$, let J_s be the set of the indices of the columns of $\mathbf{A}'$ belonging to $\mathcal{U}_s$, i.e. $j' \in J_s$ if and only if $\mathbf{a}_{j'} \in \mathcal{U}_s$ and $j' \in \{1,\ldots,m\}$. Since $\mathbf{A}'$ has full rank, it is clear that $\mathcal{U}_s = \operatorname{colspan}\left[\mathbf{A}\left(:,J_s\right)\right]$. If there is an edge with "label" j'' between two nodes $j_1' \in I_{s_1}$

and $j_2' \in I_{s_2}$, then $\mathbf{a}_{j''}$ has non-zero components in $\mathbf{a}_{j_1'}$ and in $\mathbf{a}_{j_2'}$ and thus it belongs to the sub-spaces $\mathcal{U}_{s_1}$ and $\mathcal{U}_{s_2}$. But we know that the $\mathcal{U}_s$'s are linearly independent. Hence, we must have that s_1 and s_2 are equal. We conclude that there are at least S components in $\mathcal{G}$ and thus $T \geq \operatorname{rank}_{\oplus}(\mathbf{A})$. $\square$

We give a method to compute efficiently the $\oplus$-rank of a given matrix. Therefore, let $\mathbf{A}$ be a (m, n) matrix and consider the following linear problem:

$$\boxed{\begin{array}{c} \text{solve system} \\ \hdashline \mathbf{M} \cdot \mathbf{A} = \mathbf{A} \cdot \operatorname{diag}(\xi_1, \ldots, \xi_n) \\ \hdashline \text{with variables} \\ \hdashline \mathbf{M} \in \mathbb{R}^{m \times m} \quad , \quad \Xi := (\xi_1, \ldots, \xi_n) \in \mathbb{R}^n \ . \end{array}} \ . \tag{6.1}$$

Clearly the system is homogeneous; hence, the trivial solution will always be valid. Let us denote by $\mathcal{S}$ the sub-space of matrices $\mathbf{M}$ and vectors Ξ that satisfy (6.1). We are now able to introduce a useful lemma:

Lemma 6.2. *Let $\mathbf{A}$ have full row-rank and no zero columns and let $\mathcal{S}$ be defined as above. Then the $\oplus$-rank of $\mathbf{A}$ is equal to the dimension of $\mathcal{S}$.*

Proof. First note that, if $\mathbf{A}$ has full row-rank, then n is larger or equal to m. Let $\mathbf{A}'$ and $\mathbf{A}''$ be defined as previously. Without loss of generality, we may again assume that $\mathbf{A}'$ has full rank. Let $(\xi_1, \ldots, \xi_m)$ be fixed. Then we must have

$$\mathbf{M} = \mathbf{A}' \cdot \operatorname{diag}(\xi_1, \ldots, \xi_m) \cdot [\mathbf{A}']^{-1} \ .$$

Hence, $\mathbf{M}$ is completely determined by $(\xi_1, \ldots, \xi_m)$. Reversely, if $\mathbf{M}$ is known, then every $\{\xi_1, \ldots, \xi_n\}$ are also determined since $\mathbf{A}$ has no zero columns. Hence, we only have to check the number of "degrees of freedom" in $(\xi_1, \ldots, \xi_m)$ to compute the dimension of $\mathcal{S}$.

Let $\mathcal{G}$ be the graph associated to $\mathbf{A}$. Suppose that j_1' and j_2' are two nodes that are adjacent to each other with an edge j''. Then, we have

$$\mathbf{M} \cdot \mathbf{a}_{j''} = \left[\mathbf{A}' \cdot \operatorname{diag}(\xi_1, \ldots, \xi_m) \cdot [\mathbf{A}']^{-1}\right] \cdot \mathbf{a}_{j''} = \sum_{j'=1}^{m} \mathbf{a}_{j'} \xi_{j'} q_{j'}^{j''} = \mathbf{a}_{j''} \xi_{j''} \ .$$

Since the $\mathbf{a}_{j'}$'s are linearly independent and since $q_{j_1'}^{j''}$ and $q_{j_2'}^{j''}$ are non-zero, then the only solution provides that $\xi_{j_1'}$ and $\xi_{j_2'}$ are equal to $\xi_{j''}$. Hence, we conclude that, if two nodes $j_1', j_2' \in \{1, \ldots, m\}$ belong to the same component $\mathcal{G}_t$, then the two variables $\xi_{j_1'}$ and $\xi_{j_2'}$ must have the same value. Hence, the dimension of $\mathcal{S}$ is lower or equal to $\operatorname{rank}_{\oplus}(\mathbf{A})$.

On the other hand, let $S := \operatorname{rank}_{\oplus}(\mathbf{A})$ and let $\{\mathcal{U}_1, \ldots, \mathcal{U}_S\}$ be a covering of $\mathbf{A}$. Then, for each $s \in \{1, \ldots, S\}$, let $\mathbf{M}_s$ be the "projection on $\mathcal{U}_s$" operator:

$$\mathbf{M}_s(\mathbf{x} + \mathbf{y}) := \mathbf{x} \qquad \text{for all } \mathbf{x} \in \mathcal{U}_s \text{ and } \mathbf{y} \in \bigoplus_{s' \neq s} \mathcal{U}_{s'} .$$

Let $(\eta_1, \ldots, \eta_S)$ be a fixed vector and define $\mathbf{M} := \sum_{s=1}^{S} \eta_s \mathbf{M}_s$ and for each $j \in \{1, \ldots, n\}$, let $\xi_j := \eta_{s_j}$ where s_j is the unique index in $\{1, \ldots, S\}$ such that $\mathbf{a}_j \in \mathcal{U}_{s_j}$. Then this $\mathbf{M}$ and this Ξ are solutions to (6.1). Hence, the dimension of $\mathcal{S}$ is at least $\operatorname{rank}_{\oplus}(\mathbf{A})$. $\square$

We are now able to prove the main theorem for this subsection:

Theorem 6.3. *Let $\mathbf{A}$ be a (m, n) matrix with rank r and let Z be the number of zero columns in $\mathbf{A}$. Then the following formula links the $\oplus$-rank of $\mathbf{A}$ with the dimension of the solution space $\mathcal{S}$:*

$$\dim(\mathcal{S}) = \operatorname{rank}_{\oplus}(\mathbf{A}) + (m-1)(m-r) + Z\,.$$

Proof. First we suppose that there are no zero columns in $\mathbf{A}$. Let $\mathbf{X} \in \mathrm{GL}(m)$ and observe that the $\oplus$-rank of $\mathbf{XA}$ and the $\oplus$-rank of $\mathbf{A}$ are equal. Define the space $\mathcal{S}'$ as follows:

$$\mathcal{S}' := \left\{ \left(\mathbf{X}^{-1}\mathbf{M}\mathbf{X}, \Xi \right) \,\middle|\, (\mathbf{M}, \Xi) \in \mathcal{S} \right\} .$$

Clearly $\mathcal{S}'$ has the same dimension as $\mathcal{S}$. Hence, we may assume, without loss of generality, that the $m-r$ last rows of $\mathbf{A}$ are zero and that $\tilde{\mathbf{A}} := \mathbf{A}(1\!:\!r, :)$ has full row-rank. In this case, it is easy to check that $\operatorname{rank}_{\oplus}(\mathbf{A}) = \operatorname{rank}_{\oplus}\left(\tilde{\mathbf{A}}\right) + (m-r)$. The matrix $\mathbf{M}$ may be "block-partitioned" into the following sub-matrices:

$$\mathbf{M} = \left[\begin{array}{c|c} \boxed{\mathbf{M}_1} & \\ & \mathbf{M}_3 \\ \boxed{\mathbf{M}_2} & \end{array} \right] \qquad \text{where} \quad \left\{ \begin{array}{lcl} \mathbf{M}_1 & \in & \mathbb{R}^{r \times r} \\ \mathbf{M}_2 & \in & \mathbb{R}^{(m-r) \times r} \\ \mathbf{M}_3 & \in & \mathbb{R}^{m \times (m-r)} \end{array} \right. .$$

From Lemma 6.2, we know that the dimension of the space containing valid $\mathbf{M}_1$'s is $\operatorname{rank}_{\oplus}\left(\tilde{\mathbf{A}}\right)$. On the other hand, $\mathbf{M}_2$ is necessarily zero and, finally, the sub-matrix $\mathbf{M}_3$ might take any value. Hence, the dimension of the space of valid $\mathbf{M}_3$'s is $m(m-r)$. The coefficients $(\xi_1, \ldots, \xi_n)$ are completely determined by $\mathbf{M}_1$ if there are no zero columns.

Finally, observe that appending a zero column to $\mathbf{A}$ does not change its $\oplus$-rank and also does not change the space of valid $\mathbf{M}$'s. The only thing that changes is that the coefficient ξ_{n+1} affected to this zero column might take any value. Hence, the dimension of $\mathcal{S}$ is increased by one. This concludes the proof of the theorem. $\square$

6.1.2 Computation (part 1) $\to$ T-scale + T-trace

Let $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ be two F-PD's of a matrix multiplication tensor $\mathbf{T}_{mpn}$. We would like to know whether the two F-PD's are inv-equivalent and, if they are, to compute the inv-transformations for joining $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ to $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$. First suppose that the two F-PD's are inv-equivalent and also suppose that we know which transformation T-perm is involved in the inv-transformations. We will see in the next subsection how we can compute T-perm without trying every permutation of $\{1, \ldots, F\}$. Hence, it remains to compute the transformations T-scale and T-trace.

First we would like to make two observations: the first observation is that, in general, the factor matrices of "realistic" F-PD's $[\![\mathbf{A}, \mathbf{BC}]\!]$ contain no zero columns. Indeed, if one of the factor matrices contains such a zero column, let's say the r-th column is zero for example, then we can remove the r-th column of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, this will still provide a F-PD of $\mathbf{T}_{mpn}$. Hence, we can easily construct a F-PD with less rank-at-most-1 terms and, in this sense, this F-PD is not "realistic" (c.f. the hardness of the matrix multiplication tensor decomposition problem briefly exposed in Chapter 2). The second observation about $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is summarized in the following theorem:

Theorem 6.4. *Let $\mathbf{T}_{mpn}$ be a matrix multiplication tensor and let $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ be a F-PD of $\mathbf{T}_{mpn}$. Let $I \subseteq \{1, \ldots, F\}$ be a set of indices such that $\#I + n \geq F + 1$. Then the sub-matrix $\mathbf{A}\left(:, I\right)$ has full row-rank.*

Proof. Suppose, on the contrary, that there exists a set $I \subseteq \{1, \ldots, F\}$ with size E such that $E + n \geq F + 1$ and such that $\mathbf{A}\left(:, I\right)$ has not full row-rank. Let $\mathbf{A}' := \mathbf{A}\left(:, I\right)$ and let $\mathbf{B}' := \mathbf{B}\left(:, \{1, \ldots, F\} \setminus I\right)$. Since $\mathbf{A}'$ has not full row-rank, we can find a non-zero vector $\mathbf{x}$ with mp-elements such that $\mathbf{x}^\top \mathbf{A}' = 0$. Denote $\mathcal{Y}$ the space of pn-elements vectors $\mathbf{y}$ such that $\mathbf{y}^\top \mathbf{B}' = 0$. Since the number of columns in $\mathbf{B}'$ is lower or equal to $n - 1$, there are at most $n - 1$ "constraints" on $\mathbf{y}$ and thus the dimension of $\mathcal{Y}$ is at least $pn - n + 1$. Let $\mathbf{X} := \operatorname{rshp}_{[m,p]}\left(\mathbf{x}\right)$ and define the sub-space $\mathcal{V}$ containing pn-elements vectors as

$$\mathcal{V} := \left\{ \operatorname{vec}\left[(\mathbf{ZX})^\top \right] \;\middle|\; \mathbf{Z} \in \mathbb{R}^{n \times m} \right\} = \operatorname{colspan}\left(\mathbf{I}_n \otimes \mathbf{X}^\top \right) .$$

Since $\mathbf{X}$ is non-zero, its rank is at least 1 and thus the dimension of $\mathcal{V}$ is at least n. Now let $\mathbf{v} = \operatorname{vec}\left[(\mathbf{ZX})^\top \right]$ be an element of $\mathcal{V}$ and let $\mathbf{y}$ be an element of $\mathcal{Y}$. Define $\mathbf{z} := \operatorname{vec}\left(\mathbf{Z}\right)$ and, finally, define $\mathbf{Y} := \operatorname{rshp}_{[p,n]}\left(\mathbf{y}\right)$, then observe that

$$\mathbf{v}^\top \mathbf{y} = \operatorname{trace}\left(\mathbf{ZXY}\right) = \sum_{r=1}^{F} \left[\mathbf{x}^\top \mathbf{A}\left(:, r\right) \right] \cdot \left[\mathbf{y}^\top \mathbf{B}\left(:, r\right) \right] \cdot \left[\mathbf{z}^\top \mathbf{C}\left(:, r\right) \right] .$$

We have used the fact that the trace of a product is stable under cycle-shifting of its factors. Hence, we conclude that $\mathbf{v}^\top \mathbf{y} = 0$ so that $\mathcal{Y} \subseteq \mathcal{V}^\perp$. But a simple look at the dimensions of $\mathcal{Y}$ and $\mathcal{V}$ shows that this is a contradiction. $\square$

Note that Theorem 6.4 applies in a similar way on $\mathbf{B}$ and $\mathbf{C}$. To see this, it suffices to note that, if $[\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$ is a F-PD of $\mathbf{T}_{mpn}$, then $[\![\mathbf{B},\mathbf{C},\mathbf{A}]\!]$ and $[\![\mathbf{C},\mathbf{A},\mathbf{B}]\!]$ are F-PD's of $\mathbf{T}_{pnm}$ and $\mathbf{T}_{nmp}$ respectively. Among other conclusions of this theorem, we conclude that each factor matrices in a F-PD of a matrix multiplication tensor must have full row-rank. Indeed, it suffices to apply Theorem 6.4 with $I := \{1,\dots,F\}$.

We now present the algorithm to compute the (T-scale+T-trace)-equivalence between two F-PD's of a matrix multiplication tensor. The algorithm will work provided we make the following hypothesis on the two decompositions:

Hypothesis 6.5. *We suppose that the factor matrices of the two decompositions* $[\![\mathbf{A}_1,\mathbf{B}_1,\mathbf{C}_1]\!]$ *and* $[\![\mathbf{A}_2,\mathbf{B}_2,\mathbf{C}_2]\!]$ *have no zero columns and also that they all have a* $\oplus$*-rank of* 1*.*

As we have already mentioned, the first part of the hypothesis is valid for every "realistic" decomposition. We will see, in Sec. 6.2, that the second part of the hypothesis is verified with probability one for tensors larger or with the same size as $\mathbf{T}_{222}$ and when the F-PD's are "randomly computed". The goal of the algorithm is to compute matrices $\mathbf{P} \in \mathrm{GL}(m)$, $\mathbf{Q} \in \mathrm{GL}(p)$ and $\mathbf{R} \in \mathrm{GL}(n)$ and scaling coefficients $\{\lambda_1,\dots,\lambda_F\}$, $\{\mu_1,\dots,\mu_F\}$ and $\{\nu_1,\dots,\nu_F\}$ such that

$$\left\{\begin{array}{l} \left[\mathbf{Q}^\top \otimes \mathbf{P}^{-1}\right]\cdot \mathbf{A}_1 = \mathbf{A}_2 \cdot \mathrm{diag}\left(\lambda_1,\dots,\lambda_F\right) \\ \left[\mathbf{R}^\top \otimes \mathbf{Q}^{-1}\right]\cdot \mathbf{B}_1 = \mathbf{B}_2 \cdot \mathrm{diag}\left(\mu_1,\dots,\mu_F\right) \\ \left[\mathbf{P}^\top \otimes \mathbf{R}^{-1}\right]\cdot \mathbf{C}_1 = \mathbf{C}_2 \cdot \mathrm{diag}\left(\nu_1,\dots,\nu_F\right) \end{array}\right\} . \tag{6.2}$$

Let $\mathbf{E}_1$ and $\mathbf{E}_2$ be two (I,J) matrices with full row-rank, containing no zero columns and whose $\oplus$-rank is 1. Then consider the following linear problem:

$$\boxed{\begin{array}{l} \text{solve system} \\ \hline \qquad \mathbf{M}\cdot\mathbf{E}_1 = \mathbf{E}_2 \cdot \mathrm{diag}\left(\xi_1,\dots,\xi_J\right) \\ \hline \text{with variables} \\ \hline \qquad \mathbf{M} \in \mathbb{R}^{I\times I} \quad , \quad \Xi = \left(\xi_1,\dots,\xi_J\right) \in \mathbb{R}^{J} \quad . \end{array}} \; . \tag{6.3}$$

This problem is close to problem (6.1) except that we allow distinct $\mathbf{E}_1$ and $\mathbf{E}_2$. Let $\mathcal{S}$ be the space of $\mathbf{M}$'s and Ξ's that are solutions to (6.3). Then consider the following lemma:

Lemma 6.6. *If the solution space $\mathcal{S}$ of (6.3) contains a solution $(\mathbf{M}, \Xi)$ such that $\xi_j \neq 0$ for every $j \in \{1, \ldots, J\}$, then the dimension of $\mathcal{S}$ is one.*

Proof. Let $(\mathbf{M}, \Xi)$ be a solution of (6.3) with $\xi_j \neq 0$ for every $j \in \{1, \ldots, J\}$. We have assumed that $\mathbf{E}_2$ has full row-rank and thus $\mathbf{E}_2 \cdot \operatorname{diag}(\xi_1, \ldots, \xi_J)$ has full row-rank as well. Hence, $\mathbf{M}$ must be invertible. In a similar way as in the proof of Lemma 6.2, we may assume, without loss of generality, that $\mathbf{E}_1(:, 1\!:\!I)$ has full rank. Hence, $\mathbf{E}_2(:, 1\!:\!I)$ has full rank too. We conclude the proof with a similar reasoning as for the first part of the proof of Lemma 6.2. □

In a first time, suppose that $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ are (T-scale+T-trace)-equivalent and let $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ and $\{\lambda_1, \ldots, \lambda_F\}$, $\{\mu_1, \ldots, \mu_F\}$ and $\{\nu_1, \ldots, \nu_F\}$ be some matrices and scaling coefficients satisfying (6.2). Let $\mathcal{S}^1$ be the solution space of (6.3) with $(\mathbf{E}_1, \mathbf{E}_2) = (\mathbf{A}_1, \mathbf{A}_2)$. Similarly, let $\mathcal{S}^2$ and $\mathcal{S}^3$ be the solution spaces of (6.3) with $(\mathbf{E}_1, \mathbf{E}_2) = (\mathbf{B}_1, \mathbf{B}_2)$ and $(\mathbf{E}_1, \mathbf{E}_2) = (\mathbf{C}_1, \mathbf{C}_2)$ respectively. Since $\mathcal{S}^1$, $\mathcal{S}^2$ and $\mathcal{S}^3$ admit the solutions

$$\left\{ \begin{array}{lcl} \left[\, \mathbf{Q}^\top \otimes \mathbf{P}^{-1} \,,\, (\lambda_1, \ldots, \lambda_F) \,\right] & \in & \mathcal{S}^1 \\ \left[\, \mathbf{R}^\top \otimes \mathbf{Q}^{-1} \,,\, (\mu_1, \ldots, \mu_F) \,\right] & \in & \mathcal{S}^2 \\ \left[\, \mathbf{P}^\top \otimes \mathbf{R}^{-1} \,,\, (\nu_1, \ldots, \nu_F) \,\right] & \in & \mathcal{S}^3 \end{array} \right\}, \tag{6.4}$$

the dimension of $\mathcal{S}^1$, $\mathcal{S}^2$ and $\mathcal{S}^3$ must be one. Let $(\mathbf{M}^1, \Xi^1)$, $(\mathbf{M}^2, \Xi^2)$ and $(\mathbf{M}^3, \Xi^3)$ be non-zero solutions belonging to $\mathcal{S}^1$, $\mathcal{S}^2$ and $\mathcal{S}^3$ respectively. From the dimension of $\mathcal{S}^1$, $\mathcal{S}^2$ and $\mathcal{S}^3$ being one, we know that there exist constants $\alpha, \beta, \gamma \in \mathbb{R}$ such that

$$\left\{ \begin{array}{lcll} \mathbf{M}^1 = \alpha \left[\mathbf{Q}^\top \otimes \mathbf{P}^{-1}\right] & \text{and} & \xi^1_r = \alpha\lambda_r & \text{for all } r \in \{1, \ldots, F\} \\ \mathbf{M}^2 = \beta \left[\mathbf{R}^\top \otimes \mathbf{Q}^{-1}\right] & \text{and} & \xi^2_r = \beta\mu_r & \text{for all } r \in \{1, \ldots, F\} \\ \mathbf{M}^3 = \gamma \left[\mathbf{P}^\top \otimes \mathbf{R}^{-1}\right] & \text{and} & \xi^3_r = \gamma\nu_r & \text{for all } r \in \{1, \ldots, F\} \end{array} \right\}.$$

Hence, we can also easily find matrices

$$\left\{ \begin{array}{lcll} \mathbf{S}^1_1, \mathbf{S}^3_2 & \in & \mathbb{R}^{m \times m} & \text{such that } \|\mathbf{S}^1_1\|_{\mathrm{L}^2} = \|\mathbf{S}^3_2\|_{\mathrm{L}^2} = 1 \\ \mathbf{S}^2_1, \mathbf{S}^1_2 & \in & \mathbb{R}^{p \times p} & \text{such that } \|\mathbf{S}^2_1\|_{\mathrm{L}^2} = \|\mathbf{S}^1_2\|_{\mathrm{L}^2} = 1 \\ \mathbf{S}^3_1, \mathbf{S}^2_2 & \in & \mathbb{R}^{n \times n} & \text{such that } \|\mathbf{S}^3_1\|_{\mathrm{L}^2} = \|\mathbf{S}^2_2\|_{\mathrm{L}^2} = 1 \end{array} \right\} \tag{6.5}$$

and satisfying

$$\left\{ \begin{array}{ll} \mathbf{M}^1 = \hat{\alpha}\, [\mathbf{S}^1_2 \otimes \mathbf{S}^1_1] & \text{for some } \hat{\alpha} \in \mathbb{R} \\ \mathbf{M}^2 = \hat{\beta}\, [\mathbf{S}^2_2 \otimes \mathbf{S}^2_1] & \text{for some } \hat{\beta} \in \mathbb{R} \\ \mathbf{M}^3 = \hat{\gamma}\, [\mathbf{S}^2_2 \otimes \mathbf{S}^3_1] & \text{for some } \hat{\gamma} \in \mathbb{R} \end{array} \right\}. \tag{6.6}$$

Hence, we also have that

$$\left\{\begin{array}{llll} \mathbf{S}^1_1 = \alpha'_1 \mathbf{P}^{-1} & \text{and} & \mathbf{S}^1_2 = \alpha'_2 \mathbf{Q}^\top & \text{for some } \alpha'_1, \alpha'_2 \in \mathbb{R} \\ \mathbf{S}^2_1 = \beta'_1 \mathbf{Q}^{-1} & \text{and} & \mathbf{S}^2_2 = \beta'_2 \mathbf{R}^\top & \text{for some } \beta'_1, \beta'_2 \in \mathbb{R} \\ \mathbf{S}^3_1 = \gamma'_1 \mathbf{R}^{-1} & \text{and} & \mathbf{S}^3_2 = \gamma'_2 \mathbf{P}^\top & \text{for some } \gamma'_1, \gamma'_2 \in \mathbb{R} \end{array}\right\} .$$

Now let $\zeta_1 := \alpha'_1\gamma'_2$, $\zeta_2 := \beta'_1\alpha'_2$ and $\zeta_3 := \gamma'_1\beta'_2$ and observe that

$$\mathbf{S}^1_1 \cdot \left[\mathbf{S}^3_2\right]^\top = \zeta_1 \mathbf{I}_m \quad \text{and} \quad \mathbf{S}^2_1 \cdot \left[\mathbf{S}^1_2\right]^\top = \zeta_2 \mathbf{I}_p \quad \text{and} \quad \mathbf{S}^3_1 \cdot \left[\mathbf{S}^2_2\right]^\top = \zeta_3 \mathbf{I}_n \ .$$

Then define

$$\left\{\begin{array}{llll} \tilde{\mathbf{S}}^1_1 := \frac{1}{\sqrt{\zeta_1}} \mathbf{S}^1_1 \ , & \tilde{\mathbf{S}}^1_2 := \frac{1}{\sqrt{\zeta_2}} \mathbf{S}^1_2 & \text{and} \quad \tilde{\xi}^1_r := \frac{1}{\hat{\alpha}\sqrt{\zeta_1\zeta_2}} \xi^1_r & \text{for all } r \in \{1, \ldots, F\} \\ \tilde{\mathbf{S}}^2_1 := \frac{1}{\sqrt{\zeta_2}} \mathbf{S}^2_1 \ , & \tilde{\mathbf{S}}^2_2 := \frac{1}{\sqrt{\zeta_3}} \mathbf{S}^2_2 & \text{and} \quad \tilde{\xi}^2_r := \frac{1}{\hat{\beta}\sqrt{\zeta_2\zeta_3}} \xi^2_r & \text{for all } r \in \{1, \ldots, F\} \\ \tilde{\mathbf{S}}^3_1 := \frac{1}{\sqrt{\zeta_3}} \mathbf{S}^3_1 \ , & \tilde{\mathbf{S}}^3_2 := \frac{1}{\sqrt{\zeta_1}} \mathbf{S}^3_2 & \text{and} \quad \tilde{\xi}^3_r := \frac{1}{\hat{\gamma}\sqrt{\zeta_3\zeta_1}} \xi^3_r & \text{for all } r \in \{1, \ldots, F\} \end{array}\right\} .$$

and let

$$\tilde{\mathbf{P}} := \left[\tilde{\mathbf{S}}^1_1\right]^{-1} \quad \text{and} \quad \tilde{\mathbf{Q}} := \left[\tilde{\mathbf{S}}^2_1\right]^{-1} \quad \text{and} \quad \tilde{\mathbf{R}} := \left[\tilde{\mathbf{S}}^3_1\right]^{-1} .$$

Then we clearly have

$$\left\{\begin{array}{l} \frac{1}{\hat{\alpha}} \mathbf{M}^1 = \frac{1}{\sqrt{\zeta_1\zeta_2}} \left[\tilde{\mathbf{Q}}^\top \otimes \tilde{\mathbf{P}}^{-1}\right] \\ \frac{1}{\hat{\beta}} \mathbf{M}^2 = \frac{1}{\sqrt{\zeta_2\zeta_3}} \left[\tilde{\mathbf{R}}^\top \otimes \tilde{\mathbf{Q}}^{-1}\right] \\ \frac{1}{\hat{\gamma}} \mathbf{M}^3 = \frac{1}{\sqrt{\zeta_3\zeta_2}} \left[\tilde{\mathbf{P}}^\top \otimes \tilde{\mathbf{R}}^{-1}\right] \end{array}\right\} .$$

Finally, note that, since $\alpha = \hat{\alpha}\alpha'_1\alpha'_2$, $\beta = \hat{\beta}\beta'_1\beta'_2$ and $\gamma = \hat{\gamma}\gamma'_1\gamma'_2$, and since the coefficients $\{\lambda_1, \ldots, \lambda_F\}$, $\{\mu_1, \ldots, \mu_F\}$ and $\{\nu_1, \ldots, \nu_F\}$ satisfy $\lambda_r\mu_r\nu_r = 1$ for all $r \in \{1, \ldots, F\}$, we also have that $\tilde{\xi}^1_r\tilde{\xi}^2_r\tilde{\xi}^3_r = 1$ for all $r \in \{1, \ldots, F\}$. Hence, we have shown that the matrices $\tilde{\mathbf{P}}$, $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ together with the scaling coefficients $\left\{\tilde{\xi}^1_1, \ldots, \tilde{\xi}^1_F\right\}$, $\left\{\tilde{\xi}^2_1, \ldots, \tilde{\xi}^2_F\right\}$ and $\left\{\tilde{\xi}^3_1, \ldots, \tilde{\xi}^3_F\right\}$ provide valid transformations T-scale and T-trace for joining $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ to $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$.

We have just seen that, if $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ are two (T-scale+T-trace)-equivalent F-PD's, then the solution spaces $\mathcal{S}^1$, $\mathcal{S}^2$ and $\mathcal{S}^3$ can be used to find transformations T-scale and T-trace for joining them. On the other hand, suppose that we have computed $\mathcal{S}^1$, $\mathcal{S}^2$ and $\mathcal{S}^3$ and, from them, we would like to know whether $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ are (T-scale+T-trace)-equivalent. From Lemma 6.6 and from (6.4), we know that, if one the solution spaces $\mathcal{S}^1$, $\mathcal{S}^2$ or $\mathcal{S}^3$ has dimension different from one, then the decompositions are not (T-scale+T-trace)-equivalent. Hence, we might suppose that the dimension of $\mathcal{S}^1$, $\mathcal{S}^2$ and $\mathcal{S}^3$ is one. As of now, choose non-trivial solutions $(\mathbf{M}^1, \Xi^1)$, $(\mathbf{M}^2, \Xi^2)$ and $(\mathbf{M}^3, \Xi^3)$ from $\mathcal{S}^1$, $\mathcal{S}^2$ and $\mathcal{S}^3$ respectively. Then try to compute (c.f. the remark just below this paragraph) the matrices $(\mathbf{S}^1_1, \mathbf{S}^1_2)$, $(\mathbf{S}^2_1, \mathbf{S}^2_2)$ and $(\mathbf{S}^3_1, \mathbf{S}^3_2)$ and the corresponding coefficients $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ satisfying (6.5) and (6.6). If it is not possible to find such matrices and coefficients, then we can conclude that the F-PD's $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and

$[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ are not (T-scale+T-trace)-equivalent. Now compute $\mathbf{S}_1^1 \cdot [\mathbf{S}_2^3]^\top$, $\mathbf{S}_1^2 \cdot [\mathbf{S}_2^1]^\top$ and $\mathbf{S}_1^3 \cdot [\mathbf{S}_2^2]^\top$ and check whether there exist constants $\zeta_1, \zeta_2, \zeta_3 \in \mathbb{R}$ such that

$$\mathbf{S}_1^1 \cdot \left[\mathbf{S}_2^3\right]^\top = \zeta_1 \mathbf{I}_m \quad \text{and} \quad \mathbf{S}_1^2 \cdot \left[\mathbf{S}_2^1\right]^\top = \zeta_2 \mathbf{I}_p \quad \text{and} \quad \mathbf{S}_1^3 \cdot \left[\mathbf{S}_2^2\right]^\top = \zeta_3 \mathbf{I}_n \ .$$

If this is not the case, then we can conclude again that $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ are not (T-scale+T-trace)-equivalent. On the other hand, if such ζ_1, ζ_2 and ζ_3 do exist, then it suffices to check whether

$$\xi_r^1 \xi_r^2 \xi_r^3 = \left(\hat{\alpha}\hat{\beta}\hat{\gamma}\right) \cdot (\zeta_1 \zeta_2 \zeta_3) \qquad \text{for each } r \in \{1, \dots, F\}. \tag{6.7}$$

If this is the case, then we can define $\tilde{\mathbf{P}}$, $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ and the coefficients $\left\{\tilde{\xi}_1^1, \dots, \tilde{\xi}_F^1\right\}$, $\left\{\tilde{\xi}_1^2, \dots, \tilde{\xi}_F^2\right\}$ and $\left\{\tilde{\xi}_1^3, \dots, \tilde{\xi}_F^3\right\}$ in the same way as above. Those will provide valid transformations T-scale and T-trace. If (6.7) is not satisfied, then $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ are not (T-scale+T-trace)-equivalent.

Remark. We explain how we compute the matrices $(\mathbf{S}_1^1, \mathbf{S}_2^1)$, $(\mathbf{S}_1^2, \mathbf{S}_2^2)$ and $(\mathbf{S}_1^3, \mathbf{S}_2^3)$ and the coefficients $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ satisfying (6.5) and (6.6), especially when we have to take into account the floating-point inaccuracy. The problem can be rephrased as follows: let $\mathbf{M}$ be a (mp, mp) matrix and we search a (m, m) matrix $\mathbf{P}$ and a (p, p) matrix $\mathbf{Q}$, both with L^2-norm equals 1, and a coefficient α such that $\|\mathbf{M} - \alpha\,(\mathbf{P} \otimes \mathbf{Q})\|_{\mathrm{L}^2}$ is minimal.

To solve the problem, first "block-partition" $\mathbf{M}$ in m^2 matrices with size (p, p) and number them with respect to the "column-major order". For example, if $m = 3$, we would have the following block partition:

$$\mathbf{M} = \begin{bmatrix} \boxed{\mathbf{M}_1} & \boxed{\mathbf{M}_4} & \boxed{\mathbf{M}_7} \\ \boxed{\mathbf{M}_2} & \boxed{\mathbf{M}_5} & \boxed{\mathbf{M}_8} \\ \boxed{\mathbf{M}_3} & \boxed{\mathbf{M}_6} & \boxed{\mathbf{M}_9} \end{bmatrix}$$

where the $\mathbf{M}_s$'s are (p, p) matrices. Now let $\mathbf{M}'$ be the (p^2, m^2) matrix defined as

$$\mathbf{M}' := [\,\mathrm{vec}\,(\mathbf{M}_1)\ ,\ \mathrm{vec}\,(\mathbf{M}_2)\ ,\ \dots\ ,\ \mathrm{vec}\,(\mathbf{M}_{m^2})\,]\ .$$

Hence, $\mathbf{M}'$ is a reshaped version of $\mathbf{M}$ and observe that

$$\text{minimize} \quad \|\mathbf{M} - \alpha\,(\mathbf{P} \otimes \mathbf{Q})\|_{\mathrm{L}^2} \qquad \text{with variables } \mathbf{P}, \mathbf{Q}, \alpha$$

is equivalent to

$$\text{minimize} \quad \left\|\mathbf{M}' - \alpha \left[\mathrm{vec}\,(\mathbf{Q})\,\mathrm{vec}\,(\mathbf{P})^\top\right]\right\|_{\mathrm{L}^2} \qquad \text{with variables } \mathbf{P}, \mathbf{Q}, \alpha. \tag{6.8}$$

Solutions to (6.8) can be obtained by considering the singular value decomposition $\mathbf{M}'$, i.e. $\mathbf{M}' = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\boldsymbol{\Sigma}$ is a diagonal matrix with non-negative diagonal elements in decreasing order. A solution is then given by

$$\mathbf{P} := \mathbf{V}\,(\,:, 1) \quad , \quad \mathbf{Q} := \mathbf{U}\,(\,:, 1) \quad \text{and} \quad \alpha := \boldsymbol{\Sigma}\,(1, 1)\ .$$

6.1.3 Computation (part 2) $\to$ getting rid of T-perm

The problem in the computation of the (T-scale+T-trace)-equivalence above is that we do not know what is the permutation that act on the columns of $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$. A straightforward method would be to test every permutation of $\{1, \ldots, F\}$ and, for each permuted decomposition $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$, try to find a (T-scale+T-trace)-equivalence with $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$. The drawback is that this would be prohibitive in running time. For instance, if $F = 7$ (like in the $\langle 2, 2, 2\rangle$ case), we would have 5040 permutations to try. Hence, we have developed more clever method.

Let $\{\mathbf{a}^1_1, \ldots, \mathbf{a}^1_F\}$, $\{\mathbf{b}^1_1, \ldots, \mathbf{b}^1_F\}$ and $\{\mathbf{c}^1_1, \ldots, \mathbf{c}^1_F\}$ be the columns of $\mathbf{A}_1$, $\mathbf{B}_1$ and $\mathbf{C}_1$ respectively. Similarly, let $\{\mathbf{a}^2_1, \ldots, \mathbf{a}^2_F\}$, $\{\mathbf{b}^2_1, \ldots, \mathbf{b}^2_F\}$ and $\{\mathbf{c}^2_1, \ldots, \mathbf{c}^2_F\}$ be the columns of $\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2$ respectively. For each $r \in \{1, \ldots, F\}$, define the following matrices:

$$\left\{ \begin{array}{l} \mathbf{M}^1_r := \mathrm{rshp}\left(\mathbf{a}^1_r\right) \cdot \mathrm{rshp}\left(\mathbf{b}^1_r\right) \cdot \mathrm{rshp}\left(\mathbf{c}^1_r\right) \\ \mathbf{M}^2_r := \mathrm{rshp}\left(\mathbf{a}^2_r\right) \cdot \mathrm{rshp}\left(\mathbf{b}^2_r\right) \cdot \mathrm{rshp}\left(\mathbf{c}^2_r\right) \end{array} \right\} .$$

Suppose that $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ are inv-equivalent and let σ be the permutation involved in T-perm. Similarly, let $\mathbf{P} \in \mathrm{GL}(m)$, $\mathbf{Q} \in \mathrm{GL}(p)$ and $\mathbf{R} \in \mathrm{GL}(n)$ be the matrices involved in T-trace. Then, for each $r \in \{1, \ldots, F\}$, we have that

$$\mathbf{M}^1_r = \mathbf{P}^{-1} \cdot \mathbf{M}^2_{\sigma(r)} \cdot \mathbf{P} .$$

Now consider the two vectors $(r_1, \ldots, r_E)$ and $(s_1, \ldots, s_E)$. We say that those two vectors form a "restriction of σ" if we have $s_\ell = \sigma(r_\ell)$ for each $\ell \in \{1, \ldots, E\}$. In this case, we can check that, for every randomly chosen coefficients $\{\alpha_1, \ldots, \alpha_E\} \subseteq \mathbb{R}$, we have

$$\alpha_1 \cdot \mathbf{M}^1_{r_1} + \ldots + \alpha_E \cdot \mathbf{M}^1_{r_E} = \mathbf{P}^{-1} \cdot \left(\alpha_1 \cdot \mathbf{M}^2_{s_1} + \ldots + \alpha_E \cdot \mathbf{M}^2_{s_E}\right) \cdot \mathbf{P}$$

and thus

$$\mathrm{eig}\left(\alpha_1 \cdot \mathbf{M}^1_{r_1} + \ldots + \alpha_E \cdot \mathbf{M}^1_{r_E}\right) = \mathrm{eig}\left(\alpha_1 \cdot \mathbf{M}^2_{s_1} + \ldots + \alpha_E \cdot \mathbf{M}^2_{s_E}\right) \tag{6.9}$$

where $\mathrm{eig}(\mathbf{E})$ is the function returning the eigenvalues of $\mathbf{E}$.

The idea of the algorithm is the following: first we start from small vectors $(r_1, \ldots, r_E)$ and $(s_1, \ldots, s_E)$. We check whether they are susceptible to form a "restriction of σ" by checking whether (6.9) is satisfied or not. If (6.9) is satisfied, then we try to extend $(r_1, \ldots, r_E)$ and $(s_1, \ldots, s_E)$ to larger vectors $(r_1, \ldots, r_E, r_{E+1})$ and $(s_1, \ldots, s_E, s_{E+1})$ and we check again whether those extended vectors might provide a "restriction of σ" according to (6.9). Otherwise, we try other vectors $\left(r_1, \ldots, r_E, r'_{E+1}\right)$ and $\left(s_1, \ldots, s_E, s'_{E+1}\right)$. If every possible r_{E+1}'s and s_{E+1}'s have been tried and none of them provides a correct "restriction of σ" according to (6.9), then we start from the last $(r_1, \ldots, r_{E-1})$ and $(s_1, \ldots, s_{E-1})$ and test different r_E's and s_E's. On the other hand, when we reach a "complete permutation", i.e. when E is equal to F, then we can try to compute the (T-scale+T-trace)-equivalence

```
Function $[R, S, \text{EquivFound}] \leftarrow$ RecursiveSearch $(R, S, \text{EquivFound})$ :
        If $\text{EquivFound} = \text{True}$ :
                There is no need to search a (T-scale+T-trace)-equivalence further. Hence, return $[R, S, \text{EquivFound}]$ and exit the function.
        If $\text{length}\,(S) = F$ :
                Try to find a (T-scale+T-trace)-equivalence between $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ permuted according to $R$ and $S$. If such a equivalence exists, then return $[R, S, \text{True}]$ and exit the function. Else, return $[R, S, \text{False}]$ and exit the function.
        For each $r^+ \notin R$ and $s^+ \notin S$ :
                Let $R^+ := [R, r^+]$ and $S^+ := [S, s^+]$ and check whether $R^+$ and $S^+$ satisfy (6.9). If they do, then set $[R, S, \text{EquivFound}] \leftarrow$ RecursiveSearch $(R^+, S^+, \text{EquivFound})$. If $\text{EquivFound} = \text{True}$, then return $[R, S, \text{EquivFound}]$ and exit the function. On the other hand, if $\text{EquivFound} = \text{False}$ or if $R^+$ and $S^+$ do not satisfy (6.9), then continue to the next iteration of the "for each" loop. If it is the last iteration of the "for each" loop, then return $[R, S, \text{EquivFound}]$ and exit the function.
```

Algorithm 2 – Recursive function to compute the inv-equivalence.

between $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ permuted with this "complete permutation". If a (T-scale+T-trace)-equivalence exists, then we have found the inv-equivalent between the two F-PD's. If they are not (T-scale+T-trace)-equivalent, then we search another permutation. At the end of the algorithm, we have either found the inv-equivalent between $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ or we can conclude that they are not inv-equivalent. We implemented the algorithm as the recursive function described in Algorithm 2. The recursive function must be called with

$$[R, S, \text{EquivFound}] \leftarrow \texttt{RecursiveSearch}\,(\varnothing, \varnothing, \text{False})\ .$$

If the obtained EquivFound is true, then the two decompositions are inv-equivalent and the transformation T-perm is given by the vectors R and S. On the other hand, if EquivFound is false, then the two F-PD's are not inv-equivalent.

Remark. Strictly speaking, the use of Algorithm 2 supposes that Hypothesis 6.5 is satisfied. We wonder whether we can still obtain some information from Algorithm 2 even if Hypothesis 6.5 is not satisfied. The answer is yes. We modify the algorithm as follows, if the condition length $(S) = F$ is satisfied, then, instead of searching a (T-scale+T-trace)-equivalence between the F-PD's (which is not possible since the $\oplus$-rank of the factor matrices is not 1 anymore), we directly do:

$$\text{Return } [R, S, \text{True}] \text{ and exit the function.}$$

With this modified algorithm, if the call of the function

$$[R, S, \text{EquivFound}] \leftarrow \texttt{RecursiveSearch}\,(\varnothing, \varnothing, \text{False})$$

returns the value of true for EquivFound, then we cannot say anything: the two F-PD's might be inv-equivalent or not. However, if EquivFound is false, then we are sure that the two F-PD's are not inv-equivalent. This modification will be helpful in Sec. 6.2, when we will consider the distribution of the inv-equivalent classes for the $\langle 2, 1, 2\rangle$ case.

6.2 Distribution of the $\oplus$-ranks and inv-equivalence classes

In this section, we would like to apply the algorithms we have developed in Sec. 6.1 on "randomly computed" decompositions. Specifically, we would like to have an idea about the distribution of the $\oplus$-rank of the factor matrices. We also want to inquire "what is the probability for two "randomly computed" decompositions to be inv-equivalent?".

6.2.1 Distribution of the $\oplus$-rank of the factor matrices

Remember that, in Sec. 5.2, we have computed the set (5.1) consisting of 100 decompositions $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$ for the different matrix multiplication tensors between $\mathbf{T}_{121}$ and $\mathbf{T}_{333}$. We reuse those $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$'s and we compute the $\oplus$-rank of the factor matrices $\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa$ for each $\kappa \in \{1, \ldots, 100\}$. For the computation of the $\oplus$-ranks, we use Theorem 6.3. The "$\oplus$-ranks' vector" of the F-PD $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$ is defined by

$$\oplus\text{-vec}_\kappa := \left[\text{rank}_\oplus\left(\mathbf{A}_\kappa\right), \text{rank}_\oplus\left(\mathbf{B}_\kappa\right), \text{rank}_\oplus\left(\mathbf{C}_\kappa\right)\right] \ .$$

For each case described in the two first columns of Table 6.1, we have computed the $\oplus$-vec$_\kappa$'s for the different decompositions $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$. It appears that, in each case (taken separately), the $\oplus$-vec$_\kappa$'s are the same for 100% of the F-PD's. The results are gathered in Table 6.1. Those "$\oplus$-ranks' vectors", which seem to appear with probability one, will be called the "generic $\oplus$-vec's for the F-PD's of $\mathbf{T}_{mpn}$".

Let us analyze, case-by-case, the different situations. The generic $\oplus$-vec for the 2-PD's of $\mathbf{T}_{121}$, is $[2, 2, 1]$. In fact, this is not surprising for the following reasons: $\mathbf{A}$ and $\mathbf{B}$ are $(2, 2)$ matrices and $\mathbf{C}$ is a $(1, 2)$ matrix. From Theorem 6.4, we know that $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ have full row-rank. Hence, with Lemma 6.1, we conclude that there is only one possibility for the $\oplus$-vec's of the 2-PD's of $\mathbf{T}_{121}$: in this case, the generic "$\oplus$-ranks' vector" is also the only possible "$\oplus$-ranks' vector".

For the 4-PD's of $\mathbf{T}_{212}$, the generic "$\oplus$-ranks' vector" is $[1, 1, 4]$. The factor matrix $\mathbf{C}$ is a $(4, 4)$ matrix with full row-rank; hence, by Lemma 6.1, we conclude that its $\oplus$-rank is

Multiplication and number of terms in the F-PD		100% of the $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$'s have the following "$\oplus$-ranks' vector"
$\langle 1,2,1\rangle$	$F=2$	$[\,2\,,\,2\,,\,1\,]$
$\langle 2,1,2\rangle$	$F=4$	$[\,1\,,\,1\,,\,4\,]$
$\langle 2,2,2\rangle$	$F=7$	$[\,1\,,\,1\,,\,1\,]$
$\langle 2,3,2\rangle$	$F=11$	$[\,1\,,\,1\,,\,1\,]$
$\langle 3,2,3\rangle$	$F=15$	$[\,1\,,\,1\,,\,1\,]$
$\langle 3,3,3\rangle$	$F=23$	$[\,1\,,\,1\,,\,1\,]$

Table 6.1 – Generic values of the $\oplus$-ranks of the factor matrices.

necessarily 4. The $\oplus$-rank of $\mathbf{A}$ and the $\oplus$-rank $\mathbf{B}$ seem to be 1 with probability one. Is it possible that they are different $\oplus$-ranks for $\mathbf{A}$ and $\mathbf{B}$, even if this happens with probability zero? To answer this question, consider the following 4-PD of $\mathbf{T}_{mpn}$:

$\mathbf{A}^\top$	$\mathbf{B}^\top$	$\mathbf{C}^\top$
$\begin{bmatrix} +1 & 0 \\ +1 & 0 \\ 0 & +1 \\ 0 & +1 \end{bmatrix}$	$\begin{bmatrix} +1 & 0 \\ 0 & +1 \\ +1 & 0 \\ 0 & +1 \end{bmatrix}$	$\begin{bmatrix} +1 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \end{bmatrix}$

. (6.10)

In this decomposition, both $\mathbf{A}$ and $\mathbf{B}$ have a $\oplus$-rank of 2 contradicting the hypothesis whereby $\operatorname{rank}_\oplus(\mathbf{A})$ and $\operatorname{rank}_\oplus(\mathbf{B})$ are always equal to 1. However, such situations seem to occur with probability zero. As a corollary, we also conclude that the Tichavský et al.'s LM-method seems to have a zero probability to produce a decomposition of $\mathbf{T}_{212}$ in the same inv-equivalence class as (6.10).

For the four last cases, namely $\langle 2,2,2\rangle$ with $F=7$, $\langle 2,3,2\rangle$ with $F=11$, $\langle 3,2,3\rangle$ with $F=15$ and $\langle 3,3,3\rangle$ with $F=23$, 100% of the decompositions $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$ have a $\oplus$-vec equal to $[1,1,1]$. This is good news, since it implies that we can use Algorithm 2 to check whether the decompositions are pair-wise inv-equivalent.

6.2.2 Distribution of the inv-equivalence classes of F-PD's

For each case, except the $\langle 1,2,1\rangle$ case with $F=2$, we use the 100 decompositions $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$ from (5.1). Then we compute 100 random pairs (κ_1^s, κ_2^s) where $\kappa_1^s \neq \kappa_2^s$ and $\kappa_1^s, \kappa_2^s \in \{1,\dots,100\}$ for each $s \in \{1,\dots,100\}$. We use Algorithm 2 on $[\![\mathbf{A}_{\kappa_1^s}, \mathbf{B}_{\kappa_1^s}, \mathbf{C}_{\kappa_1^s}]\!]$ and $[\![\mathbf{A}_{\kappa_2^s}, \mathbf{B}_{\kappa_2^s}, \mathbf{C}_{\kappa_2^s}]\!]$ to check whether they are inv-equivalent or not. The results are

Multiplication and number of terms in the F-PD		Percentage of the random (κ_1^s, κ_2^s)'s such that $[\![\mathbf{A}_{\kappa_1^s}, \mathbf{B}_{\kappa_1^s}, \mathbf{C}_{\kappa_1^s}]\!]$ and $[\![\mathbf{A}_{\kappa_2^s}, \mathbf{B}_{\kappa_2^s}, \mathbf{C}_{\kappa_2^s}]\!]$ are inv-equivalent	Average computing time of Algorithm 2
$\langle 1,2,1\rangle$	$F=2$	`100 %`	`not applic.`
$\langle 2,1,2\rangle$	$F=4$	`000 %`	`2.317e-03 sec`
$\langle 2,2,2\rangle$	$F=7$	`100 %`	`1.032e-02 sec`
$\langle 2,3,2\rangle$	$F=11$	`000 %`	`9.076e-03 sec`
$\langle 3,2,3\rangle$	$F=15$	`000 %`	`3.665e-02 sec`
$\langle 3,3,3\rangle$	$F=23$	`000 %`	`4.559e-02 sec`

Table 6.2 – Distributions of the inv-equivalence classes.

represented in Table 6.2. We have also represented the average computing time (on an ordinary personal computer) of Algorithm 2 for the different cases. Note that a dashed line separates the first two cases from the last four cases. This is because the $\langle 1,2,1\rangle$ case and the $\langle 2,1,2\rangle$ case had to be handled differently because the F-PD's for those cases do not satisfy Hypothesis 6.5.

For the $\langle 1,2,1\rangle$ case with $F=2$, we claim that all decompositions $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$ are inv-equivalent to each other. How do we know that? In fact, we show that if $[\![\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]\!]$ and $[\![\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]\!]$ are two F-PD's (not necessarily "randomly computed") of $\mathbf{T}_{121}$, then they are necessarily inv-equivalent. To see this, observe that the size of $\mathbf{T}_{121}$ is $(2,2,1)$ and thus it might be represented by the following $(2,2)$ matrix:

$$\mathbf{T}_{121} := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and each F-PD $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ of $\mathbf{T}_{121}$ must satisfy

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{A} \cdot \operatorname{diag}\left(\mathbf{C}\left(1,1\right), \mathbf{C}\left(1,2\right)\right) \cdot \mathbf{B}^\top .$$

Using a transformation T-scale if necessary, we may assume, without loss of generality that $\mathbf{C}_2(1,1)$ and $\mathbf{C}_2(1,2)$ are equal to 1. Hence, $\mathbf{A}_1$ and $\mathbf{B}_1^\top$ are inverse of each other and so do $\mathbf{A}_2$ and $\mathbf{B}_2^\top$. Then let $\mathbf{P} := \mathbf{A}_1\mathbf{B}_2^\top$ and observe that

$$\left\{ \begin{array}{l} \mathbf{P}^{-1} \cdot \mathbf{A}_1 = \left[\mathbf{A}_2\mathbf{A}_1^{-1}\right] \cdot \mathbf{A}_1 = \mathbf{A}_2 \\ \mathbf{P}^\top \cdot \mathbf{B}_1 = \left[\mathbf{B}_2\mathbf{B}_1^{-1}\right] \cdot \mathbf{B}_1 = \mathbf{B}_2 \end{array} \right\} .$$

Hence, we have found a transformation T-trace between the F-PD's.

For the $\langle 2,1,2\rangle$ case with $F=4$, we have used the modified version of Algorithm 2 (c.f. the remark at the end of Sec. 6.1). We obtain that the decompositions $[\![\mathbf{A}_{\kappa_1^s},\mathbf{B}_{\kappa_1^s},\mathbf{C}_{\kappa_1^s}]\!]$ and $[\![\mathbf{A}_{\kappa_2^s},\mathbf{B}_{\kappa_2^s},\mathbf{C}_{\kappa_2^s}]\!]$ are never inv-equivalent. Let us link this result with the results of Sec. 5.2. We had obtained that only one class of polynomial arrays $\mathcal{P}^{\kappa}$ exists for the different $[\![\mathbf{A}_{\kappa},\mathbf{B}_{\kappa},\mathbf{C}_{\kappa}]\!]$'s. Hence, we could have thought that there would be only a few different inv-equivalence classes among the decompositions $[\![\mathbf{A}_{\kappa},\mathbf{B}_{\kappa},\mathbf{C}_{\kappa}]\!]$. However, we must observe that this is not the case.

We already mentioned that the 7-PD's of $\mathbf{T}_{222}$ are unique up to inv-transformations, i.e. there is only one inv-equivalence class of 7-PD's of $\mathbf{T}_{222}$ (c.f. [2] for a proof). This result is supported by our computations with Algorithm 2: for each pair (κ_1^s,κ_2^s), $[\![\mathbf{A}_{\kappa_1^s},\mathbf{B}_{\kappa_1^s},\mathbf{C}_{\kappa_1^s}]\!]$ and $[\![\mathbf{A}_{\kappa_2^s},\mathbf{B}_{\kappa_2^s},\mathbf{C}_{\kappa_2^s}]\!]$ are inv-equivalent.

For the three last cases, namely $\langle 2,3,2\rangle$ with $F=11$, $\langle 3,2,3\rangle$ with $F=15$ and $\langle 3,3,3\rangle$ with $F=23$, we have that none of the (κ_1^s,κ_2^s)'s lead to inv-equivalent $[\![\mathbf{A}_{\kappa_1^s},\mathbf{B}_{\kappa_1^s},\mathbf{C}_{\kappa_1^s}]\!]$ and $[\![\mathbf{A}_{\kappa_2^s},\mathbf{B}_{\kappa_2^s},\mathbf{C}_{\kappa_2^s}]\!]$. For the $\langle 2,3,2\rangle$ and $\langle 3,3,3\rangle$ cases, we could argue that this is not surprising since we already had a certain amount of different classes of $\mathcal{P}^{\kappa}$'s and since two decompositions with $\mathcal{P}^{\kappa}$'s belonging to distinct classes cannot be inv-equivalent. The $\langle 3,2,3\rangle$ is probably more surprising since we had only two classes of arrays $\mathcal{P}^{\kappa}$ and thus a similar comment as for the $\langle 2,1,2\rangle$ case holds.

Chapter 7

Conclusions

In this thesis, we tackled the problem of fast matrix multiplication. This problem is quite old but we have seen that only partial results are known so far. We tried to bring some contribution to the field, both for the development of fast matrix multiplication algorithms and for the understanding of the behavior of matrix multiplication tensors.

In Chapter 4, we have presented an algorithm to compute sparse discrete decompositions which might be used to build fast (stable) matrix multiplication algorithms. We have seen that, with this algorithm, we could compute sparse discrete decompositions for tensors up to the $\langle 3,3,3\rangle$ case, which is not, in general, an easy task. Nevertheless, we were not able to compute sparse discrete decompositions for larger tensors (c.f. also the remark concerning the $\langle 4,3,4\rangle$ case, at the end of Sec. 4.2) and, because of that, we do not have faster or more stable matrix multiplication algorithms than some authors like Ballard et al. [8] or Smirnov [6]. Our hope is that our algorithm for computing SD-sol's could be modified or executed on more powerful computers so that we would probably be able to compute sparse discrete decompositions for larger matrix multiplication tensors.

The methodology of Chapter 5 can be rephrased as follows: we try to find an "invariant" for the inv-equivalence classes and we draw conclusions about what this "invariant" must look like if the inv-equivalence class contains a particular decomposition. In our case, the "invariant" is the array $\mathcal{P}$ of characteristic polynomials, i.e. every decomposition in the same inv-equivalence class must have the same $\mathcal{P}$. The particular decompositions we hope to find in the inv-equivalence class are the discrete decompositions, i.e. we hope that every F-PD is inv-equivalent to a discrete F-PD. This implies that the $\mathcal{P}$'s might take only a restricted number of values. For example, if we hope to find a F-PD for which the factor matrices have values only in $\{-1,0,+1\}$, then all the $\mathcal{P}$'s must contain polynomials with integer coefficients only. We presented an algorithm to compute and compare efficiently the different $\mathcal{P}$'s between each other and we concluded that, for four cases on the six considered cases, all the decompositions provide a $\mathcal{P}$ containing polynomials with integer coefficients only. As we already mentioned, this does not necessarily mean that, for those cases, every F-PD can be discretized but this is a good indicator.

In Chapter 6, we addressed the problem of determining whether two F-PD's of a same matrix multiplication tensor are inv-equivalent. To do this, we introduced a new concept which is also an "invariant" of the inv-equivalence classes: the $\oplus$-rank of the factor matrices, i.e. every inv-equivalent decompositions involve factor matrices with mutually equal $\oplus$-ranks. We presented an algorithm for computing the inv-equivalence between two F-PD's or concluding that such an inv-equivalence does not exist. We had to find a clever way to get rid of the (T-perm)-equivalence without trying every possible permutation. We have applied the algorithm on some cases and observed the following results: except for the $\langle 1, 2, 1\rangle$ and $\langle 2, 2, 2\rangle$ cases, where every F-PD's are inv-equivalent to each other, the F-PD's are pair-wise inv-equivalent with probability zero.

To conclude, we would like to say that the problem of fast matrix multiplication remains a challenging field. We hope that our contribution will help to understand the problem better or will be the starting point for some further research. In any case, it was very interesting to treat this subject and discover that there is still so much to do with a problem as old and widely used as matrix multiplication.

Bibliography

[1] V. Strassen, “Gaussian elimination is not optimal,” *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, 1969.

[2] H. de Groote, “On varieties of optimal algorithms for the computation of bilinear mappings II. Optimal algorithms for 2×2-matrix multiplication,” *Theoretical Computer Science*, vol. 7, no. 2, pp. 127–148, 1978.

[3] J. Laderman, “A noncommutative algorithm for multiplying 3×3 matrices using 23 multiplications,” *Bulletin of the American Mathematical Society*, vol. 82, no. 1, pp. 126–128, 1976.

[4] M. Bläser, “On the complexity of the multiplication of matrices of small formats,” *Journal of Complexity*, vol. 19, no. 1, pp. 43–60, 2003.

[5] O. Makarov, “A noncommutative algorithm for multiplying 5×5-matrices using one hundred multiplications,” *USSR Computational Mathematics and Mathematical Physics*, vol. 27, no. 2, pp. 311–315, 1987.

[6] A. Smirnov, “The bilinear complexity and practical algorithms for matrix multiplication,” *Computational Mathematics and Mathematical Physics*, vol. 53, no. 12, pp. 1781–1795, 2013.

[7] P. Tichavský, A.-H. Phan, and A. Cichocki, “Numerical CP decomposition of some difficult tensors,” *Journal of Computational and Applied Mathematics*, vol. 317, pp. 362–370, 2017.

[8] G. Ballard, A. Benson, A. Druinsky, B. Lipshitz, and A. Schwartz, “Improving the numerical stability of fast matrix multiplication algorithms,” *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 4, pp. 1382—-1418, 2016.

[9] R. Brockett and D. Dobkin, “On the optimal evaluation of a set of bilinear forms,” *Linear Algebra and its Applications*, vol. 19, no. 3, pp. 207–235, 1978.

[10] P. Bürgisser, T. Lickteig, M. Clausen, and A. Shokrollahi, *Algebraic Complexity Theory.* Grundlehren der mathematischen Wissenschaften, Springer Berlin Heidelber, 1996.

[11] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *Journal of Symbolic Computation*, vol. 9, no. 3, pp. 251–280, 1990.

[12] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, no. 1, pp. 267–288, 1994.

[13] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[14] R. Howard, "The characteristic polynomial of a product." `http://people.math.sc.edu/howard/Classes/700/charAB.pdf`. Accessed: 2017-May-11.

[15] P. Pundir, S. Porwal, and B. Singh, "A new algorithm for solving linear bottleneck assignment problem," *Journal of Institute of Science and Technology*, vol. 20, no. 2, pp. 101–102, 2015.

[16] M. Golin, "Bipartite matching & the Hungarian method." `http://www.cse.ust.hk/~golin/COMP572/Notes/Matching.pdf`. Accessed: 2017-May-15.

[17] Y. Cao, "Hungarian algorithm for linear assignment problems (V2.3)." `https://www.mathworks.com/matlabcentral/fileexchange/20652-hungarian-algorithm-for-linear-assignment-problems--v2-3-`. Accessed: 2017-May-15.

[18] K. Madsen, H. Nielsen, and O. Tingleff, "Methods for non-linear least squares problems (second ed.)." Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004. Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby.

Appendix A

Lipschitz-continuous gradient and quadratic upper bound

Let $\mathcal{U}$ be a finite-dimensional vector space over $\mathbb{R}$ with inner product $(u, v) \mapsto \langle u, v\rangle$ and associated norm $u \mapsto \|u\|$. We want to prove the following theorem:

Theorem A.1. *Let $f(x)$ be differentiable on $\mathcal{U}$. Then $\nabla f(x)$ satisfies*

$$\|\nabla f(x_1) - \nabla f(x_2)\| \le M \|x_1 - x_2\| \qquad \textit{for all } x_1, x_2 \in \mathcal{U}$$

if and only if

$$|f(x) - [f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x}\rangle]| \le \frac{M}{2}\|x - \bar{x}\|^2 \qquad \textit{for all } x, \bar{x} \in \mathcal{U}. \tag{A.1}$$

The “only if” part is a classical result. Proofs of the “if” part in the convex case are also current in the literature. However, we have not found any proofs of the “if” part for the non-convex case. Hence, the motivation for this appendix.

Proof. First we prove the “only if” part. Let $\bar{x} \in \mathcal{U}$ and $h \in \mathcal{U}$. Then, using the fundamental theorem of integration, we have

$$|f(\bar{x} + h) - [f(\bar{x}) + \langle \nabla f(\bar{x}), h\rangle]| = \left|\int_0^1 \langle \nabla f(\bar{x} + th) - \nabla f(\bar{x}), h\rangle \, \mathrm{d}t\right|$$

and thus

$$|f(\bar{x} + h) - [f(\bar{x}) + \langle \nabla f(\bar{x}), h\rangle]| \le \int_0^1 |\langle \nabla f(\bar{x} + th) - \nabla f(\bar{x}), h\rangle| \, \mathrm{d}t \, .$$

From Cauchy-Schwartz inequality and the Lipschitz-continuity of the gradient, we have

$$|f(\bar{x} + h) - [f(\bar{x}) + \langle \nabla f(\bar{x}), h\rangle]| \le \int_0^1 tM \|h\|^2 \, \mathrm{d}t = \frac{M}{2}\|h\|^2 \, .$$

In order to prove the “if” part, suppose, on the contrary, that there exist $\tilde{x}, y \in \mathcal{U}$ such that $\|\nabla f(\tilde{x}) - \nabla f(y)\| > M \|\tilde{x} - y\|$. Without loss of generality, we may assume that $f(\tilde{x}) = 0$, $\nabla f(\tilde{x}) = 0$ and $\tilde{x} = 0$. Denote $v := \frac{1}{M}\nabla f(y)$. From the assumptions,

$\|v\| > \|y\|$. Suppose that (A.1) is satisfied. From the lower bound emanating from (A.1) with $\bar{x} = 0$ and the upper bound emanating from (A.1) with $\bar{x} = y$, we have

$$\frac{-M}{2} \|z\|^2 \leq f(z) \leq f(y) + \langle Mv, z - y \rangle + \frac{M}{2} \|z - y\|^2$$

for all $z \in \mathfrak{U}$. The difference between the right-hand term minus the left-hand term is minimal at $z := \frac{1}{2}(y - v)$. For this value, we get

$$\frac{-M}{8} \|y - v\|^2 \leq f(y) - \frac{M}{2} \langle v, y + v \rangle + \frac{M}{8} \|y + v\|^2 \ . \tag{A.2}$$

A similar reasoning with the upper bound emanating from (A.1) with $\bar{x} = 0$ and the lower bound emanating from (A.1) with $\bar{x} = y$ provides

$$\frac{M}{2} \|z\|^2 \geq f(z) \geq f(y) + \langle Mv, z - y \rangle - \frac{M}{2} \|z - y\|^2$$

for all $z \in \mathfrak{U}$. The minimizer of the difference between the left-hand term minus the right-hand term is $z := \frac{1}{2}(y + v)$ and, for this value, we have

$$\frac{M}{8} \|y + v\|^2 \geq f(y) - \frac{M}{2} \langle v, y - v \rangle - \frac{M}{8} \|y - v\|^2 \ . \tag{A.3}$$

By subtracting (A.3) minus (A.2), we get

$$\frac{M}{4} \|y + v\|^2 + \frac{M}{4} \|y - v\|^2 \geq M \|v\|^2$$

and thus

$$\frac{M}{2} \|y\|^2 \geq \frac{M}{2} \|v\|^2 \ ,$$

a contradiction $\square$

Appendix B

Gradient of the F-terms polyadic decomposition error

Let $\mathbf{X}$ be a third-order tensor with size (I, J, K) and let $\mathbf{A}, \mathbf{B}, \mathbf{C}$ be matrices with respective sizes (I, F), (J, F) and (K, F). Consider the L^2-error between $\mathbf{X}$ and the F-PD induced by $\mathbf{A}, \mathbf{B}, \mathbf{C}$:

$$f(\mathbf{A}, \mathbf{B}, \mathbf{C}) \coloneqq \| \mathbf{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \|_{\mathrm{L}^2}^2 \ .$$

Let $\mathbf{Y} \coloneqq [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$. We will consider the unfolding of $\mathbf{Y}$ along the 2-nd mode, the other unfoldings follow in the same way. Let (i, j, k) be fixed indices and recall that

$$\mathbf{Y}(i, j, k) = \mathrm{unfold}_{[2]}(\mathbf{Y})(j, [i-1]K + k) \ .$$

But also

$$\mathbf{Y}(i, j, k) = \sum_{r=1}^{F} \mathbf{B}(j, r)\, \mathbf{A}(i, r)\, \mathbf{C}(k, r) = \sum_{r=1}^{F} \mathbf{B}(j, r) \cdot (\mathbf{A} \odot \mathbf{C})([i-1]K + k, r)$$

where we have used the definition of the Khatri-Rao product. Hence, this shows

$$\mathrm{unfold}_{[2]}(\mathbf{Y}) = \mathbf{B}(\mathbf{A} \odot \mathbf{C})^\top \ .$$

Since the unfolding does nothing more than rearranging the entries of $\mathbf{X}$ and $\mathbf{Y}$, the L^2-norm of $\mathbf{X} - \mathbf{Y}$ is the same as the L^2-norm of $\mathrm{unfold}_{[2]}(\mathbf{X} - \mathbf{Y})$, i.e.

$$f(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \left\| \mathrm{unfold}_{[2]}(\mathbf{X}) - \mathbf{B}(\mathbf{A} \odot \mathbf{C})^\top \right\|_{\mathrm{L}^2}^2 \ .$$

Let $\hat{\mathbf{X}} \coloneqq \mathrm{unfold}_{[2]}(\mathbf{X})$ and let $\mathbf{E} \coloneqq (\mathbf{A} \odot \mathbf{C})^\top$, then observe that

$$f(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \mathrm{trace}\left(\hat{\mathbf{X}}^\top \hat{\mathbf{X}}\right) - 2\,\mathrm{trace}\left(\mathbf{E}\hat{\mathbf{X}}^\top \mathbf{B}\right) + \mathrm{trace}\left(\mathbf{E}\mathbf{E}^\top \mathbf{B}^\top \mathbf{B}\right) \ .$$

Hence, we deduce that the gradient of $f(\mathbf{A}, \mathbf{B}, \mathbf{C})$ with respect to $\mathbf{B}$ is

$$\nabla_{\mathbf{B}}(f) = -2\hat{\mathbf{X}}\mathbf{E}^\top + 2\mathbf{B}\mathbf{E}\mathbf{E}^\top \ .$$

Appendix C

Tables and figures for the computation of SD-sol's

$\mathbf{A}^\top$	$\mathbf{B}^\top$	$\mathbf{C}^\top$
+1 +1 0 0	+1 0 +1 0	+1 0 0 0
0 0 +1 0	0 0 −1 −1	0 −1 0 +1
+1 0 −1 0	0 0 +1 0	−1 +1 0 0
0 −1 0 0	−1 −1 0 0	−1 0 +1 0
0 +1 +1 0	0 +1 −1 0	+1 0 0 −1
0 0 +1 +1	0 +1 0 +1	0 0 0 +1
0 +1 0 −1	0 −1 0 0	0 0 +1 −1

Table C.1 – Sparse discrete 7-PD of $\mathbf{T}_{222}$.

$\mathbf{A}^\top$	$\mathbf{B}^\top$	$\mathbf{C}^\top$
0 0 −1 0 0 0	0 0 0 0 +1 0	0 −1 0 0
0 0 0 +1 0 0	0 −1 0 0 0 0	0 0 −1 0
0 −1 0 0 0 0	+1 0 +1 0 0 0	+1 0 −1 0
0 0 0 0 +1 +1	0 0 +1 0 0 −1	0 0 0 −1
+1 +1 0 0 0 0	−1 0 0 +1 0 0	−1 0 0 0
0 −1 0 0 −1 0	0 0 −1 −1 0 0	+1 0 0 +1
0 0 0 0 +1 0	0 0 0 +1 0 +1	0 +1 0 −1
+1 0 0 0 −1 0	0 0 0 −1 0 0	−1 −1 0 0
0 0 0 −1 0 0	0 0 0 0 −1 0	0 0 0 +1
0 0 +1 0 0 0	0 +1 0 0 0 0	+1 0 0 0
0 −1 0 0 0 +1	0 0 −1 0 0 0	0 0 −1 −1

Table C.2 – Sparse discrete 11-PD of $\mathbf{T}_{232}$.

$\mathbf{A}^\top$	$\mathbf{B}^\top$	$\mathbf{C}^\top$
0 +1 0 +1 +1 +1	−1 0 −1 +1 0 0	0 0 0 −1 0 0 0 0 0
0 0 0 −1 −1 −1	+1 −1 +1 −1 0 0	0 +1 0 +1 0 0 0 0 0
0 +1 +1 0 0 0	0 0 −1 0 −1 0	0 0 0 0 0 0 0 −1 0
+1 0 0 +1 0 0	0 +1 0 0 0 −1	0 +1 −1 0 0 0 0 0 0
+1 0 0 +1 +1 +1	+1 −1 +1 0 0 +1	0 +1 0 0 0 0 0 0 0
0 −1 0 0 0 +1	0 0 0 +1 +1 0	0 0 0 0 0 −1 0 +1 0
0 0 0 +1 0 +1	0 −1 0 0 0 0	−1 +1 −1 +1 0 0 0 0 0
+1 0 +1 0 0 0	+1 0 0 0 0 0	0 0 0 0 0 0 +1 0 0
+1 0 0 0 0 −1	0 −1 0 0 −1 +1	−1 +1 −1 0 0 0 +1 0 0
0 −1 0 0 0 0	0 0 +1 −1 0 0	0 0 0 +1 −1 0 0 +1 0
0 0 0 0 0 −1	0 0 0 0 +1 −1	−1 +1 −1 0 0 −1 +1 0 +1
0 0 0 0 −1 −1	0 0 0 +1 0 +1	0 +1 0 0 0 −1 0 0 0
0 −1 0 0 −1 0	0 0 0 +1 0 0	0 0 0 0 −1 +1 0 0 0
0 0 +1 0 0 +1	0 0 0 0 −1 0	0 0 0 0 0 0 0 +1 −1
−1 0 0 0 0 0	−1 +1 0 0 +1 −1	+1 −1 0 0 0 0 −1 0 0

Table C.3 – Sparse discrete 15-PD of $\mathbf{T}_{323}$.

$\mathbf{A}^\top$	$\mathbf{B}^\top$	$\mathbf{C}^\top$
0 0 0 −1 0 0 0 0 −1	0 0 −1 0 0 0 0 −1 0	0 0 +1 0 0 0 +1 0 0
0 0 0 0 0 0 0 +1 0	0 0 0 0 0 0 −1 0 −1	+1 0 −1 +1 0 −1 0 0 0
+1 0 0 0 0 0 0 −1 0	0 0 +1 0 0 0 +1 0 +1	−1 0 +1 −1 0 0 0 0 0
+1 −1 −1 0 −1 −1 0 0 0	0 0 0 0 −1 0 0 0 0	0 0 0 0 0 0 +1 +1 0
+1 −1 −1 0 −1 0 0 0 0	−1 0 0 0 +1 0 0 0 0	0 0 0 0 −1 0 +1 +1 0
−1 0 0 0 0 0 0 0 0	0 0 0 −1 0 −1 0 0 0	0 +1 0 0 +1 0 0 0 0
0 −1 −1 0 0 0 0 +1 +1	0 0 0 0 0 +1 0 0 0	0 0 0 0 0 0 0 +1 +1
0 0 0 +1 0 0 +1 −1 0	0 0 +1 0 0 0 0 0 0	+1 0 −1 0 0 0 0 0 0
−1 0 0 0 0 0 +1 0 −1	0 0 −1 0 0 +1 0 0 −1	0 0 −1 0 0 0 0 0 0
0 +1 0 0 0 0 0 −1 0	0 0 0 0 0 0 +1 0 0	0 0 0 0 +1 +1 0 −1 −1
0 +1 +1 0 0 0 0 −1 0	0 0 0 0 0 −1 −1 0 0	0 0 0 0 +1 0 0 −1 −1
0 0 0 0 +1 0 0 0 0	−1 +1 0 0 0 0 0 0 0	0 0 0 +1 0 0 0 0 0
−1 0 0 0 0 0 0 0 0	+1 0 +1 0 0 0 +1 0 +1	−1 0 0 −1 0 0 0 0 0
0 0 0 0 0 −1 0 0 −1	0 0 0 0 0 0 0 −1 0	0 0 0 0 0 0 −1 0 +1
0 0 0 0 +1 0 0 0 0	0 0 0 0 0 0 0 −1 0	0 0 0 0 0 −1 0 0 0
0 0 0 −1 0 0 0 0 0	0 0 0 0 +1 0 0 0 0	0 −1 0 0 0 0 +1 0 0
0 0 0 +1 0 0 0 0 0	0 +1 −1 0 0 0 0 0 0	+1 0 0 0 0 0 +1 0 0
−1 +1 0 0 +1 0 0 0 0	−1 0 0 0 0 0 0 0 0	0 0 0 −1 −1 0 +1 +1 0
+1 −1 −1 0 0 0 0 0 0	0 0 0 +1 −1 +1 0 0 0	0 0 0 0 −1 0 0 0 0
0 0 0 0 0 0 0 0 +1	0 0 0 0 0 −1 0 −1 +1	0 0 +1 0 0 0 0 0 +1
+1 0 0 0 0 0 −1 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	−1 0 0 +1 0 0 −1 0 0	0 0 0 0 −1 0 0 +1 0
0 0 0 +1 0 −1 0 0 0	0 −1 0 0 +1 0 0 −1 0	0 0 0 0 0 0 +1 0 0

Table C.4 – Sparse discrete 23-PD of $\mathbf{T}_{333}$.

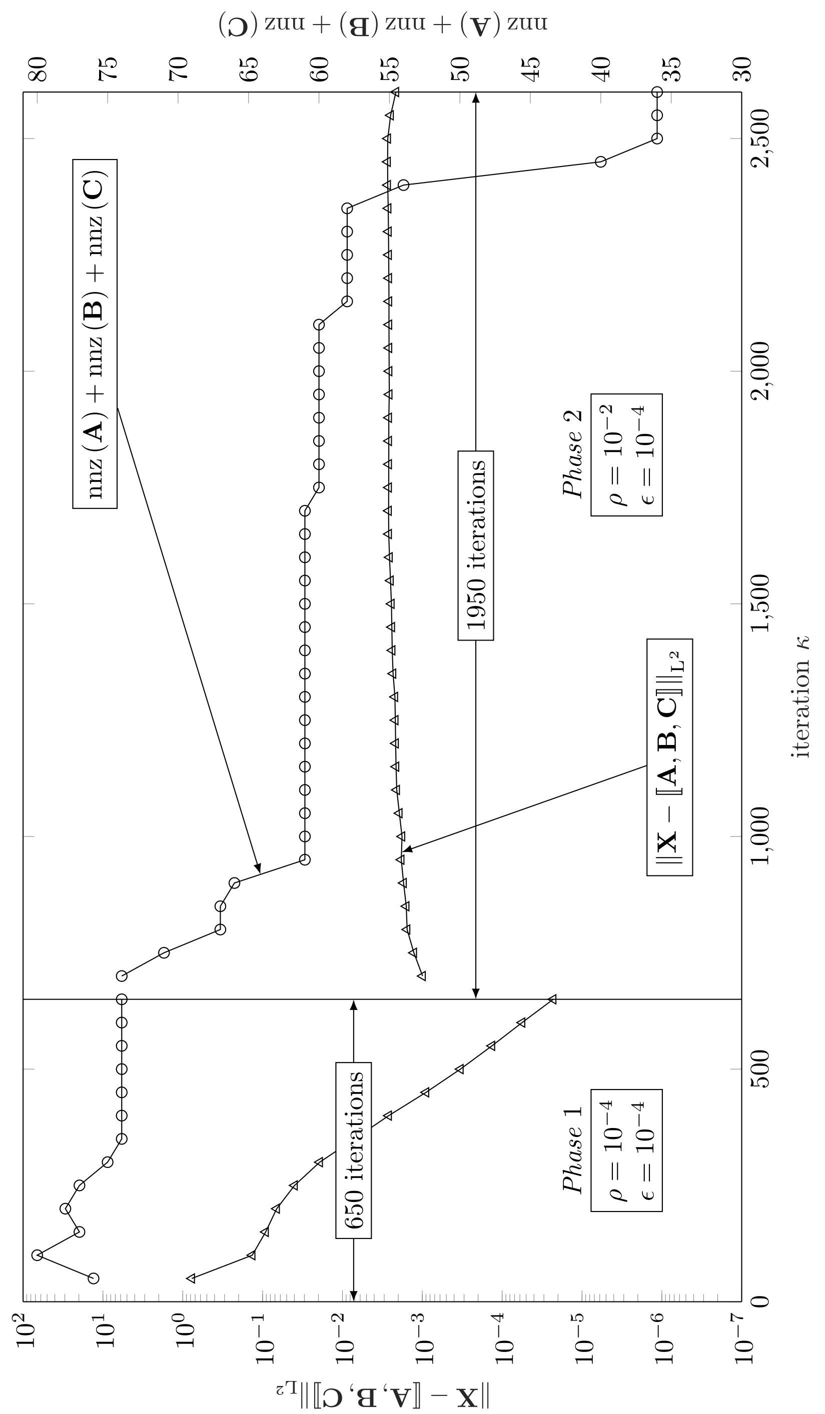


Figure C.1 – Phase 1 and Phase 2 of the procedure for the computation of a 7-PD of $\mathbf{T}_{222}$.

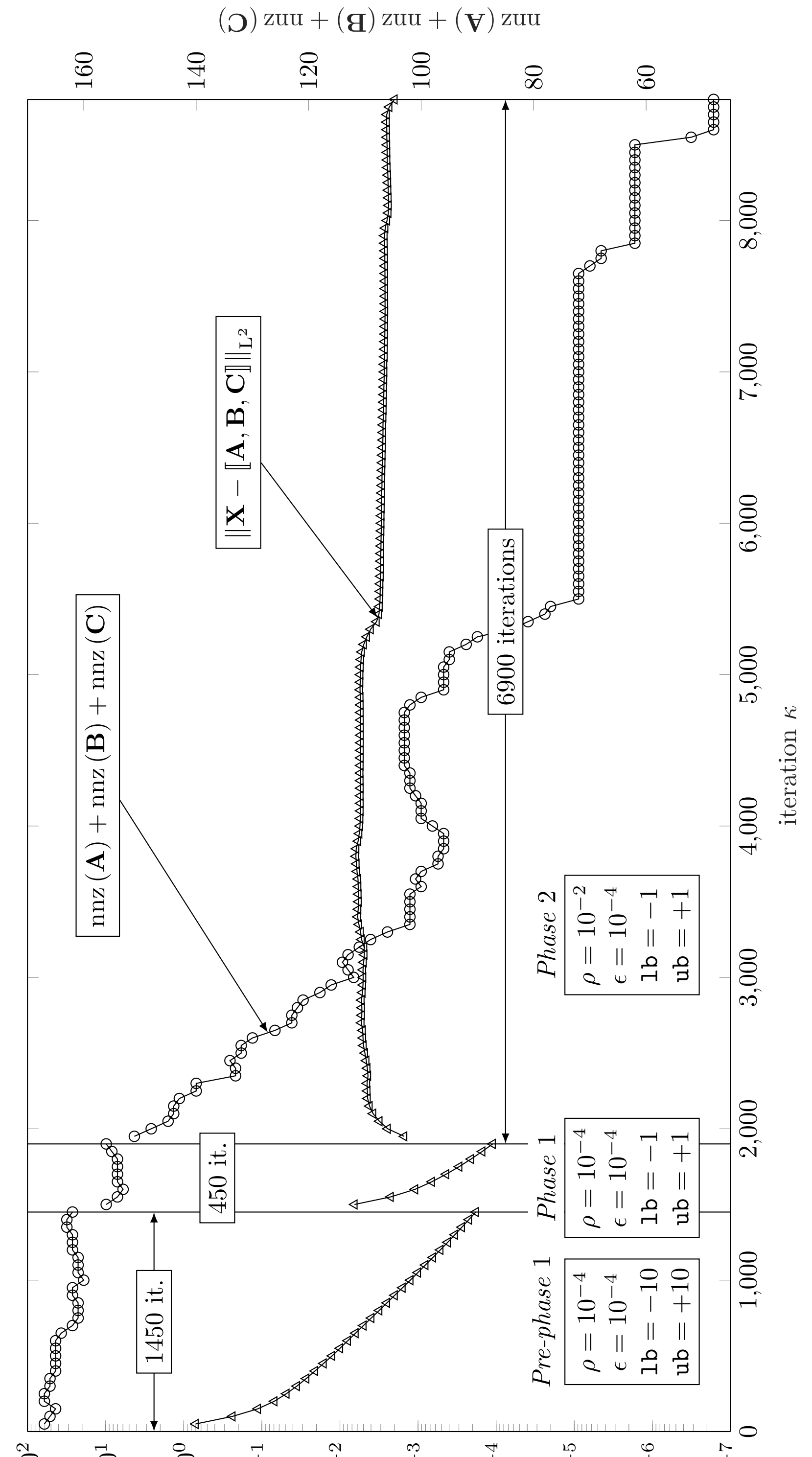


Figure C.2 – Pre-phase 1, Phase 1 and Phase 2 of the procedure for the computation of a 11-PD of $\mathbf{T}_{232}$.

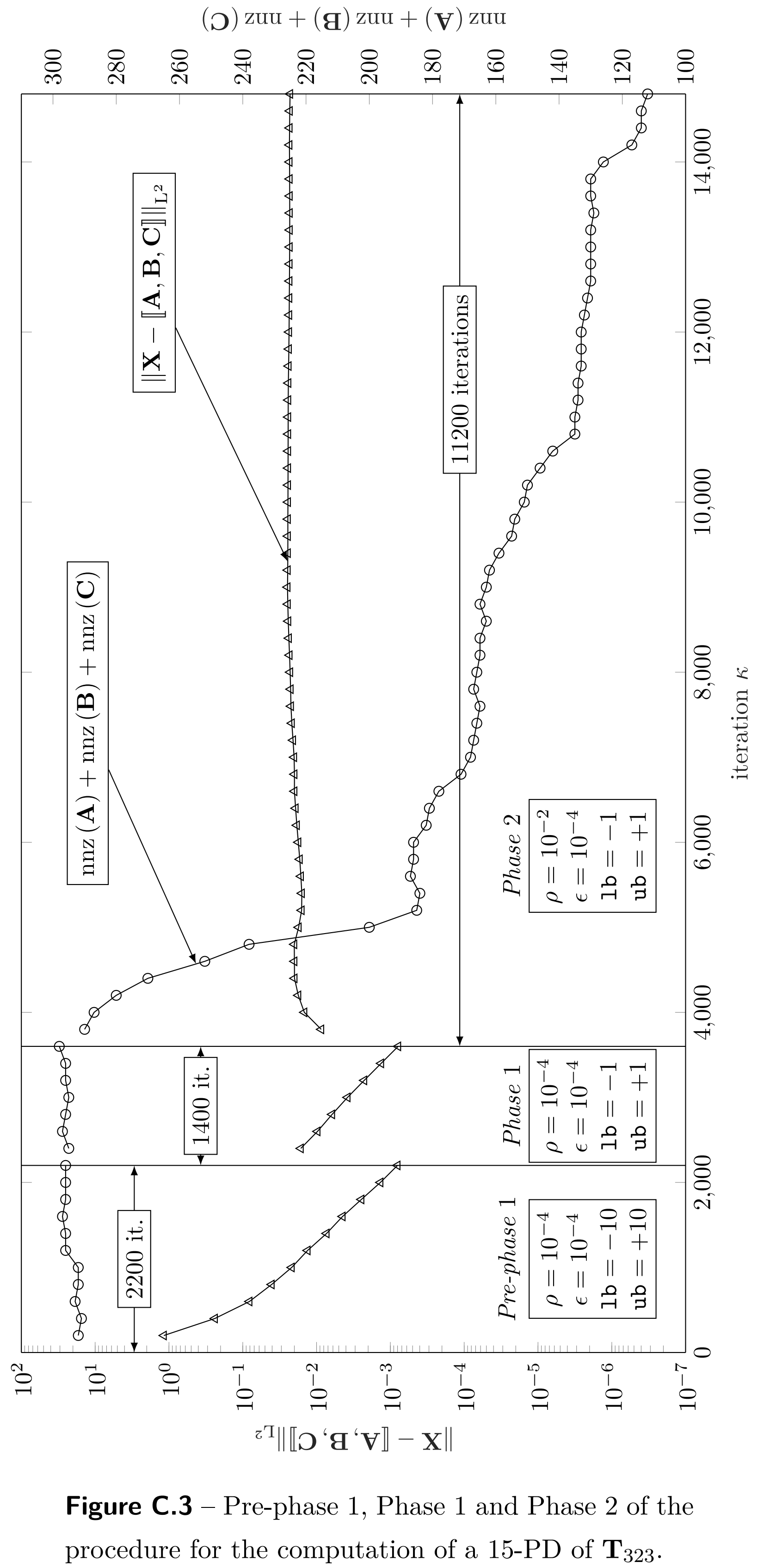


Figure C.3 – Pre-phase 1, Phase 1 and Phase 2 of the procedure for the computation of a 15-PD of $\mathbf{T}_{323}$.

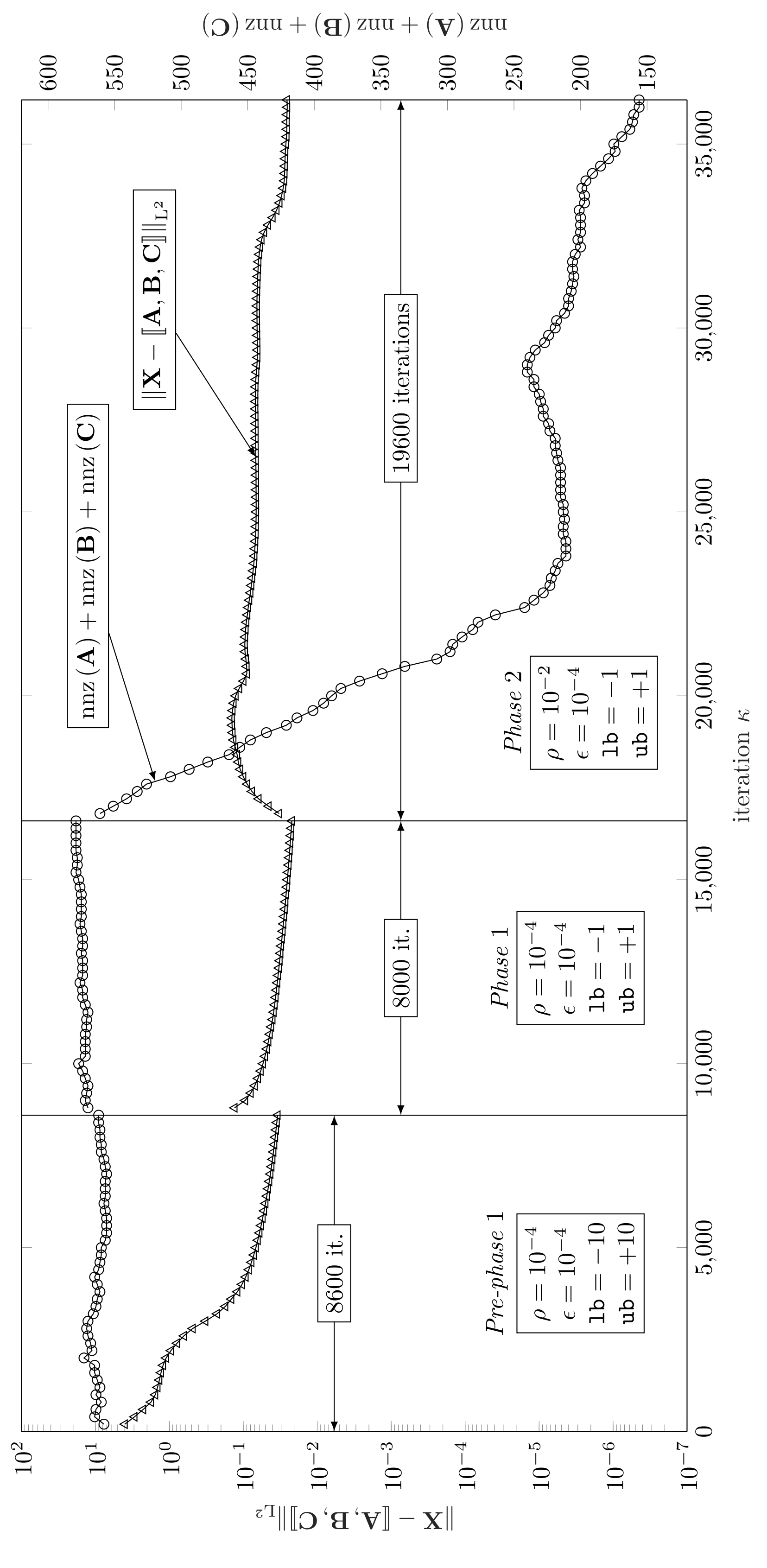


Figure C.4 – Pre-phase 1, Phase 1 and Phase 2 of the procedure for the computation of a 23-PD of $\mathbf{T}_{333}$.

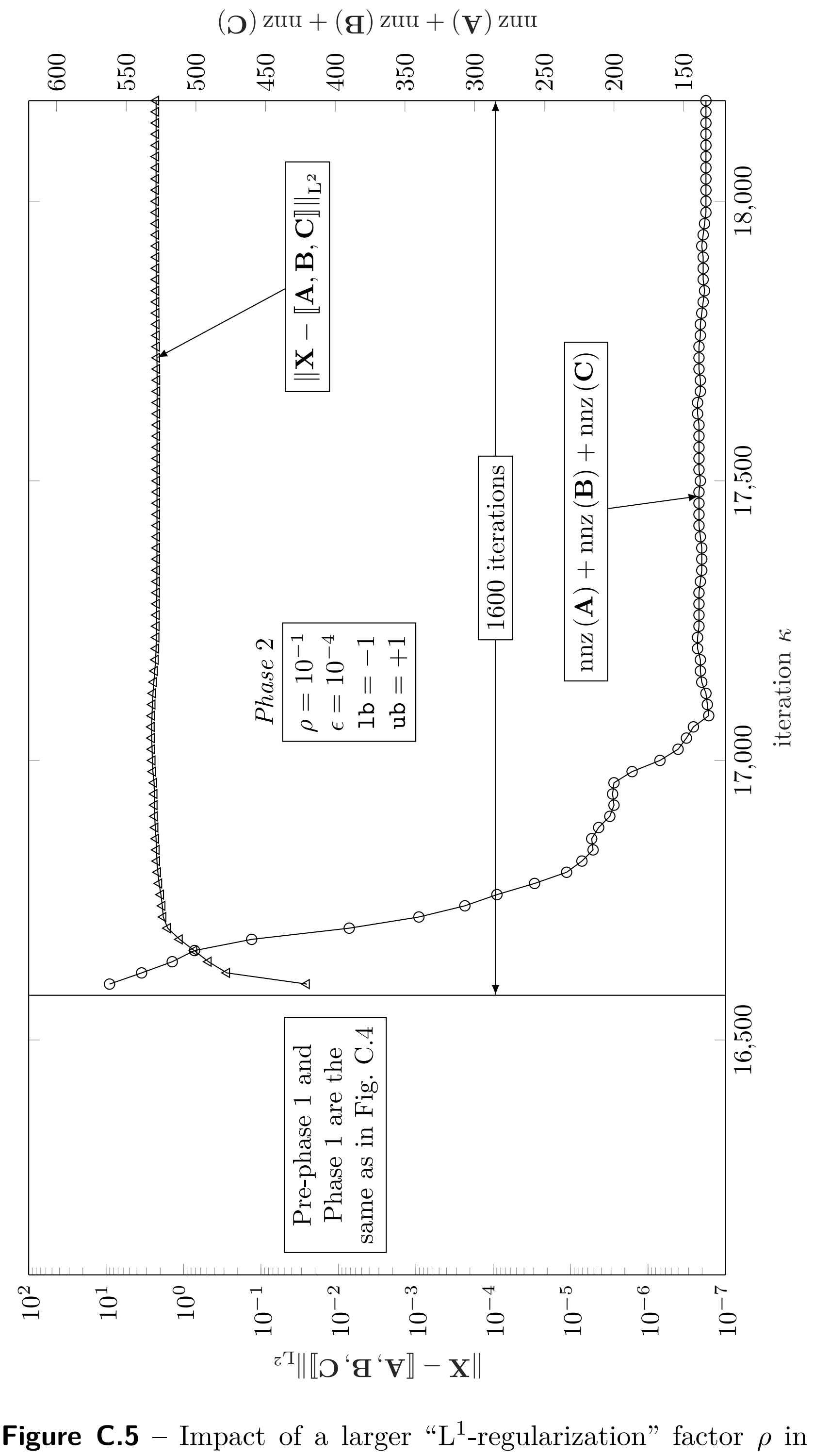


Figure C.5 – Impact of a larger "L^1-regularization" factor ρ in Phase 2 of the procedure for the computation of a 23-PD of $\mathbf{T}_{333}$.

$\mathbf{A}^\top$ in the 41-PD $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ of $\mathbf{T}_{434}$

−1	+1	0	0	0	0	−1/2	0	0	0	−1/2	0
0	0	−1	0	0	0	−1/2	0	0	0	−1/2	0
0	0	0	−1	0	0	0	0	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	0	0	0
+1	−1	0	0	0	0	0	0	+1	−1	0	0
−1	+1	0	−1	0	0	0	0	0	0	0	0
0	0	−1	+1	0	0	0	0	0	0	−1	+1
−1	+1	+1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	−1	+1	0	0	0	0
0	+1	0	0	+1	0	−1	0	+1	0	−1	0
0	0	0	−1	0	0	0	0	0	0	+1	−1
−1	0	0	−1	−1	0	0	−1	0	0	0	0
0	0	0	0	+1	0	0	0	−1	0	0	0
0	−1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	+1	0	−1/2	0	+1/2	0
0	−1	0	0	0	0	0	0	0	−1	0	0
−1	+1	0	0	0	0	0	0	0	+1	0	0
0	0	0	0	0	0	+1	0	0	0	0	0
0	0	0	0	−1	+1	0	−1	0	0	0	0
0	−1	0	0	0	−1	0	0	0	0	0	0
+1	0	0	0	+1	0	+1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	−1	0	0
0	+1	0	0	0	+1	0	0	0	0	0	0
−1	+1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	−1	0	0	−1
0	0	0	−1	0	0	0	0	0	0	0	0
0	0	0	0	+1	0	0	+1	−1	0	0	0
0	0	0	0	0	0	0	0	0	0	+1	−1
0	0	0	0	0	0	0	0	−1	0	−1	0
0	0	0	0	0	0	+1	0	0	0	+1	0
0	0	0	0	+1	0	0	+1	0	0	0	0
0	0	0	0	+1	0	0	+1	−1	0	0	−1
0	0	0	0	0	−1	0	0	0	+1	0	0
0	0	0	0	0	0	0	0	0	−1	0	0
−1	0	0	0	−1	0	0	−1	0	0	0	0
0	+1	0	0	+1	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	−1	0	+1	0
0	0	0	0	−1/2	0	0	−1/2	−1/2	+1	0	0
0	0	0	0	0	0	+1	0	0	0	−1	0

Table C.5 – Sparse discrete 41-PD of $\mathbf{T}_{434}$ (part 1).

$\mathbf{B}^\top$ in the 41-PD $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ of $\mathbf{T}_{434}$

+1	0	0	0	0	0	0	0	0	−1	+1	+1
+1	0	0	0	0	0	0	0	0	−1	0	0
0	0	0	0	0	−1	0	0	+1	0	0	0
0	−1	+1	0	−1/2	0	0	−1/2	+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	+1	+1
+1	0	0	−1	0	0	+1	0	0	−1	0	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	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	+1	0	0	−1	0	0	0
0	0	0	+1	0	0	−1	0	0	+1	0	0
0	0	0	0	0	0	0	0	0	0	−1	0
+1	−1	0	0	0	0	0	0	0	0	0	0
0	+1	−1	0	0	0	0	0	−1	0	0	0
0	0	0	0	0	0	+1	−1	0	−1	0	0
0	0	0	0	0	0	+1	−1	0	−1	+1	+1
0	0	0	0	−1	0	0	+1	0	0	0	0
0	+1	0	0	−1	0	0	+1	0	0	0	0
0	+1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	−1	0	0	0	0
0	0	−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	0	0	0	0	0	0	−1	+1	+1
0	0	−1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	+1	0	0	−1	+1	0	0
0	0	0	0	0	+1	0	0	−1	0	+1	+1
0	0	0	−1	0	+1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	+1	0	0	0
+1/2	0	−1	0	0	0	0	0	−1	−1/2	+1/2	+1/2
0	0	0	−1	+1	+1	+1	−1	−1	−1	+1	+1
0	0	0	0	0	−1	0	0	+1	0	0	−1
0	0	0	0	0	0	0	0	0	0	−1	0
0	0	0	0	0	0	−1	+1	+1	+1	−1	−1
0	0	0	+1	0	0	−1	+1	0	+1	0	0
0	−1	0	+1	0	0	−1	0	0	+1	0	0
0	0	0	0	0	0	−1	+1	0	0	0	0
0	+1	−1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	+1	0	0	−1	0	+1	+1
0	0	0	0	0	0	0	0	0	0	+1	−1

Table C.6 – Sparse discrete 41-PD of $\mathbf{T}_{434}$ (part 2).

$\mathbf{C}^\top$ in the 41-PD $[\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$ of $\mathbf{T}_{434}$															
−1	0	0	0	0	0	0	0	−1	0	0	−1	+1	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	+1	0	0	0	+1	0	−1
0	+1	+1	0	0	0	0	0	0	−1	−1	0	0	−1	−1	0
0	0	0	0	0	0	0	0	0	0	−1	0	0	0	0	0
0	−1	0	+1	0	0	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	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	−1	0	0	0	0
0	−1	0	0	0	0	0	0	0	0	0	0	0	+1	0	0
+1	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0
0	0	0	0	0	0	0	0	0	+1	0	0	0	+1	+1	0
0	0	0	0	0	0	0	0	0	0	0	0	−1	0	+1	0
0	0	0	−1	0	0	0	0	0	0	0	0	0	0	0	+1
−1	0	0	0	−1	0	0	0	0	0	0	0	0	0	0	0
+1	+1	+1	0	0	0	0	0	+1	−1	−1	0	−1	−1	−1	0
0	−1	0	+1	0	−1	0	+1	0	0	0	0	0	0	0	0
0	−1	0	+1	0	0	+1	+1	0	0	0	0	0	0	0	0
0	+1/2	−1/2	0	0	0	0	0	0	−1/2	+1/2	0	0	−1/2	+1/2	0
0	0	0	0	0	−1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	−1	−1	0	0	0	0	0	0	0	0	0	0
0	0	−1	0	0	0	0	0	0	0	0	0	0	0	+1	0
0	0	0	0	+1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	−1	−1	0	0	0	0	0	0	0	0	0
+1	0	0	0	0	0	−1	−1	+1	0	0	+1	−1	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	−1	−1
0	−1	0	0	0	−1/2	0	0	0	0	0	0	0	0	0	+1
0	0	0	0	0	0	0	0	0	0	0	0	0	−1	−1	0
+1/2	−1/2	−1/2	0	0	0	0	0	+1/2	−1/2	−1/2	0	−1/2	−1/2	−1/2	0
−1	0	0	0	0	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
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	0	0	0	−1	−1	0	0	0	0	0	0	0	0	0
0	−1	0	0	0	0	0	0	0	0	0	0	0	0	−1	0
0	0	0	0	0	+1	0	0	0	0	0	0	−1	0	0	0
0	0	−1	−1	0	0	−1	−1	0	0	0	0	0	0	0	0
+1/2	−1/2	−1/2	0	0	0	0	0	−1/2	+1/2	+1/2	0	−1/2	+1/2	+1/2	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	0	+1/2	0	0	0	0

Table C.7 – Sparse discrete 41-PD of $\mathbf{T}_{434}$ (part 3).

Appendix D

Description of the Tichavský et al.'s LM-method

In this appendix, we describe the method used by Tichavský et al. in their paper [7] to compute F-PD's of small tensors. Let us insist on the fact that we do not bring anything new in this appendix. The only purpose is that the reader can understand the quasi-totality of the report without resorting to external resources.

The method proposed by Tichavský et al. works as follows: let θ be a positive parameter and consider the following optimization problem:

minimize
$\|\mathbf{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_{\mathrm{L}^2}^2$
subject to
$\|\mathbf{A}\|_{\mathrm{L}^2}^2 + \|\mathbf{B}\|_{\mathrm{L}^2}^2 + \|\mathbf{C}\|_{\mathrm{L}^2}^2 = \theta^2$
with variables
$\mathbf{A} \in \mathbb{R}^{I\times F} \quad , \quad \mathbf{B} \in \mathbb{R}^{J\times F} \quad , \quad \mathbf{C} \in \mathbb{R}^{K\times F} \quad .$

.

The authors use the Levenberg-Marquardt method to solve this problem. Let us explain how it works. Let $f(x)$ be a function from $\mathbb{R}^n$ to $\mathbb{R}^m$. We consider the problem of

$$\text{minimize} \quad \phi(x) := \|f(x)\|^2 \qquad \text{with variable } x \in \mathbb{R}^n.$$

The Jacobian of $f(x)$ at $\bar{x}$ is denoted by $J(\bar{x})$. Hence, the gradient of $\phi(x)$ at $\bar{x}$ is

$$\nabla\phi(\bar{x}) = 2\left[J(\bar{x})^\top f(\bar{x})\right] .$$

The Levenberg-Marquardt method, which is a quasi-Newton method, consists of approximating the Hessian of $\phi(x)$ at $\bar{x}$ with

$$\mathbf{H}(\bar{x}) := 4\left[J(\bar{x})^\top J(\bar{x})\right] .$$

To avoid singular $\mathbf{H}(\bar{x})$'s, they add a "damping parameter" μ to the approximate Hessian, i.e. $\mathbf{H}_\mu(\bar{x}) := \mathbf{H}(\bar{x}) + \mu\mathbf{I}_n$. The parameter μ is updated at each iteration according to a rule described in [18]. Define the following quadratic approximation the $\phi(x)$ around $\bar{x}$:

$$m_\mu(x \mid \bar{x}) \coloneqq \phi(\bar{x}) + \langle \nabla\phi(\bar{x}), x - \bar{x} \rangle + \frac{1}{2} \langle \mathbf{H}_\mu(\bar{x})[x - \bar{x}], x - \bar{x} \rangle \ .$$

Let $T'_\mu(\bar{x})$ be the minimizer of $m_\mu(x \mid \bar{x})$ on the subset $\{x \in \mathbb{R}^n \mid \langle x - \bar{x}, \bar{x} \rangle = 0\}$. Using the Lagrange multipliers method, we obtain a closed expression for $T'_\mu(\bar{x})$ as follows: if $T'_\mu(\bar{x})$ is optimal, then there exists a constant $\lambda \in \mathbb{R}$ such that

$$\nabla\phi(\bar{x}) + \mathbf{H}_\mu(\bar{x}) \left[T'_\mu(\bar{x}) - \bar{x} \right] = \lambda \bar{x} \, .$$

Now we determine the value of λ with

$$\left\langle T'_\mu(\bar{x}) - \bar{x}, \bar{x} \right\rangle = \left\langle \lambda \mathbf{H}_\mu(\bar{x})^{-1} \bar{x} - \mathbf{H}_\mu(\bar{x})^{-1} \nabla\phi(\bar{x}), \bar{x} \right\rangle = 0 \, .$$

Finally, we find that

$$T'_\mu(\bar{x}) = \bar{x} - \mathbf{H}_\mu(\bar{x})^{-1} \nabla\phi(\bar{x}) + \frac{\left\langle \mathbf{H}_\mu(\bar{x})^{-1} \nabla\phi(\bar{x}), \bar{x} \right\rangle}{\left\langle \mathbf{H}_\mu(\bar{x})^{-1} \bar{x}, \bar{x} \right\rangle} \mathbf{H}_\mu(\bar{x})^{-1} \bar{x} \, .$$

We come back on the ball $\{x \in \mathbb{R}^n \mid \|x\| = \theta\}$ by setting

$$T_\mu(\bar{x}) \coloneqq \left[\theta \left\| T'_\mu(\bar{x}) \right\|^{-1} \right] T'_\mu(\bar{x}) \ .$$

The algorithm starts with a random initial point x_0 satisfying $\|x_0\| = \theta$ and with a initial "damping parameter" $\mu > 0$. For each $k \geq 0$, we compute $T_\mu(x_k)$. If $\phi(T_\mu(x_k))$ is lower than $\phi(x_k)$, then $T_\mu(x_k)$ becomes the new point x_{k+1} and we decrease μ according to the rules in [18]. On the other hand, if $\phi(T_\mu(x_k)) \geq \phi(x_k)$, then we increase μ with respect to the rules in [18] and we do the same steps for the new $T_\mu(x_k)$. We repeat this sequence until convergence is obtained.

About the global convergence of Tichavský et al.'s LM-method, i.e. the chance of converging to an accurate decomposition of $\mathbf{X}$ when starting from a random initial guess $\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0$, we quote the following paragraph from [7]:

"The method works well for small matrices. For example, for decomposition of the $\mathbf{T}_{333}$ and constraint $\theta = 150$ we need only a few random trials to obtain an exact fit solution. On the other hand, for tensor $\mathbf{T}_{444}$ the false local minima are so numerous that it is almost impossible to get an exact fit decomposition when the algorithm is started from random initial conditions."

This is also the reason why we have considered cases up to the $\langle 3, 3, 3 \rangle$ case and not further when we have analyzed the distribution of the discretizable decompositions and the distributions of the $\oplus$-ranks and inv-equivalence classes of F-PD's.

Appendix E

Overview of the Matlab codes attached to the report

The code are available at the address https://drive.google.com/file/d/0B_BURxd4zWUIZjN4U2ZmTVQ1TjQ/view?usp=sharing. The way to use the functions and especially the scripts is explained in their comments. The purpose of this appendix is to give an overview of the different functions and scripts we have implemented so that the reader can easily navigate through them and find the function/script he needs.

Functions
`apply_permx.m` : apply transformation T-perm on F-PD (c.f. Sec. 3.4)
`apply_scale.m` : apply transformation T-scale on F-PD (c.f. Sec. 3.4)
`apply_trace.m` : apply transformation T-trace on F-PD (c.f. Sec. 3.4)
`array_matri.m` : unfold tensor along the n-th mode (c.f. Sec. 3.1)
`bottl_assig.m` : solve the "Linear Bottleneck Assignment Problem" (c.f. Sec. 5.2)
`build_matmu.m` : build matrix multiplication tensor (c.f. Sec. 3.2)
`chpol_array.m` : compute the charpoly's of an array of $\mathbf{M}_r$'s (c.f. Sec. 5.1)
`cpdxx_ticha.m` : compute F-PD with Tichavský et al.'s LM-method (c.f. Sec. 5.2)
`disti_subsp.m` : compute the $\oplus$-rank of a matrix (c.f. Sec. 6.1)
`findx_invar.m` : implementation of Algorithm 2 (c.f. Sec. 6.1)
`hadam_squar.m` : compute $\left(\mathbf{E}_1^\top \mathbf{E}_1\right) \circledast \left(\mathbf{E}_2^\top \mathbf{E}_2\right)$ (c.f. Sec. 4.1)
`hunga_algor.m` : implementation of the "Hungarian Algorithm" (c.f. Sec. 5.2)
`khatr_raoxx.m` : compute the Khatri-Rao product (c.f. Sec. 3.1)
`krone_demul.m` : compute $\mathbf{P}$ and $\mathbf{Q}$ minimizing $\lVert\mathbf{X} - (\mathbf{P} \otimes \mathbf{Q})\rVert_{\mathrm{L}^2}$ (c.f. Sec. 6.1)

`matri_array.m` : recover $\mathbf{X}$ from $\text{unfold}_{[n]}(\mathbf{X})$ (c.f. Sec. 3.1)
`produ_facto.m` : compute the matrices $\mathbf{M}_r$ (c.f. Sec. 5.1)
`rando_facto.m` : generate random factor matrices (c.f. Sec. 3.1)
`rando_scale.m` : generate random coefficients λ_r, μ_r and ν_r (c.f. Sec. 3.4)
`rando_trace.m` : generate random matrices $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$ (c.f. Sec. 3.4)
`resha_facto.m` : reshape factor matrices according to m, p, n (c.f. Sec. 3.4)
`scale_trans.m` : solve system (6.1) (c.f. Sec. 6.1)
`sdsol_nonsm.m` : solve optimization problem (4.1) (c.f. Sec. 4.1)
`simil_check.m` : check whether two arrays of $\mathbf{M}_r$'s satisfy (6.9) (c.f. Sec. 6.1)
`tenso_recom.m` : compute the "polyadic recomposition" $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ (c.f. Sec. 3.1)

Scripts
`script_sparse_discre_comput.m` : implement the procedure described in Fig. 4.1 to compute sparse discrete decompositions (c.f. Sec. 4.1)
`script_tichav_genera_trials.m` : generate 100 F-PD's $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$ with the Tichavský et al.'s LM-method and save them in a file $\boxed{\text{name}}$`.mat` (c.f. Sec. 5.2)
`script_statis_discre_distri.m` : load the F-PD's from $\boxed{\text{name}}$`.mat` and compute statistic about the different classes of arrays $\mathcal{P}^\kappa$ (c.f. Sec. 5.2)
`script_statis_oplusx_rankxx.m` : load the F-PD's from $\boxed{\text{name}}$`.mat` and compute statistic about for the $\oplus$-ranks of the factor matrices $\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa$ (c.f. Sec. 6.2)
`script_statis_equiva_distri.m` : load the F-PD's from $\boxed{\text{name}}$`.mat` and compute statistics about the inv-equivalence classes among the $[\![\mathbf{A}_\kappa, \mathbf{B}_\kappa, \mathbf{C}_\kappa]\!]$'s (c.f. Sec. 6.2)

We also provide examples of the files $\boxed{\text{name}}$`.mat` containing 100 decompositions for the cases $\langle 1, 2, 1\rangle$ to $\langle 3, 3, 3\rangle$. The names of those files are `stat_dec_`$\boxed{m}$`x`$\boxed{p}$`x`$\boxed{n}$`_array.mat` where m, p, n refer to the multiplication $\langle m, p, n\rangle$.