COMPUTING TENSOR EIGENPAIRS USING HOMOTOPY METHODS

By

Liangmin Zhou

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Applied Mathematics — Doctor of Philosophy

2015

ABSTRACT

COMPUTING TENSOR EIGENPAIRS USING HOMOTOPY METHODS

By

Liangmin Zhou

Tensor eigenvalue problems have found important applications in automatic control, statistical data analysis, diffusion tensor imaging, image authenticity verification, spectral hypergraph theory and quantum entanglement, etc. The concept of mode-k generalized eigenvalues and eigenvectors of a tensor is introduced and some properties of such eigenpairs are proved. In particular, an upper bound for the number of equivalence classes of generalized tensor eigenpairs using mixed volume is derived. Based on this bound and the structures of tensor eigenvalue problems, two homotopy continuation type algorithms to solve tensor eigenproblems are proposed. With proper implementation, these methods can find all equivalence classes of isolated generalized eigenpairs and some generalized eigenpairs contained in the positive dimensional components (if there are any). An algorithm that combines a straightforward approach and a Newton homotopy method is introduced to extract real generalized eigenpairs from the available complex generalized eigenpairs. A MATLAB software package `TenEig 1.1` has been developed to implement these methods. Numerical results are presented to illustrate the effectiveness and efficiency of `TenEig 1.1` for computing complex or real generalized eigenpairs.

To my parents

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Professor Tien-Yien Li for his well-timed guidance, encouragement and support all the time. Without his help, I could not have been able to survive my Ph.D life. When I failed my first attempt to take qualifying exams and really thought I was such a loser, it was him who used his own story about qualifying exams to remind me in which way I can do better. When I had no clue on my research and felt I was such a dull person, it was him who can always point out my problem from our discussion and give me some small enlightening problems. Working on these problems did deepen my understanding on my own problem and widen my horizon on math. Over the years, His diligent, persistent and optimistic attitude towards research, teaching and life has been inspiring me all the time.

I would like to thank Dr. Chichia Chiu, Dr. Keith Promislow, Dr. Jianliang Qian and Dr. Zhengfang Zhou, for serving as members of my thesis committee. I am grateful to Professor Lixing Han for introducing such an interesting project to me, helping me build a whole view of the development of the tensor eigen problems and giving me a lot of advice on this subject. Special thanks go to Professor Zhonggang Zeng, without his encouragement on implementing a MATLAB version of HOM4PS2.0, Professor Lixing Han would have not been able to find us and we would not have this chance to work on such a wonderful project.

I want to take this opportunity to thank those friends with whom I have shared laughters and tears, ups and downs: Yu Chen and Junshan Lin, for being such reliable friends; Hongli Gao, for being a problem-solver; Qinfeng Gao, for being so humorous and making life so joyful; Xianfeng Hu, for being such a fantastic cook; Qi Tang and Zixuan Wang, for sharing so much together; Xin Yang, for always studying diligently and motivating me; Qiong Zheng, for having such a wonderful chance to be friends to you.

Last but not least, I owe a debt of thanks to my family, whose always being there has been lighting up my way away from home and back into home.

TABLE OF CONTENTS

LIST OF TABLES . . . . **vii**

Chapter 1 Introduction **1**

Chapter 2 Tensor eigenvalues and eigenvectors **4**
2.1 Definition of generalized mode-k tensor eigenvalue and eigenvectors 4
2.2 Properties of generalized mode-k tensor eigenvalues 8

Chapter 3 The upper bound of the number of equivalence classes of tensor eigenpairs **12**
3.1 Introduction 12
3.2 Preliminaries 13
3.3 Main Theorem based on Bernstéin's Theorem 15

Chapter 4 Computing complex tensor eigenpairs by homotopy methods **26**
4.1 Using homotopy continuation methods to solve polynomial systems 26
4.2 Construction of a linear homotopy to solve tensor eigenpairs 31
4.3 Numerical algorithms of using homotopy continuation methods to solve tensor eigenpairs 37

Chapter 5 Computing real tensor eigenpairs by homotopy methods . . . **41**
5.1 Using Newton homotopy method to find real tensor eigenpairs 41
5.2 A straightforward approach to find real tensor eigenpairs 45
5.3 A numerical algorithm of finding real tensor eigenpairs based on the Newton homotopy and the straightforward approach 53

Chapter 6 Implementation and numerical results **54**
6.1 Examples for computing complex eigenpairs 55
6.2 Computing singular tensor eigenpairs 59
6.3 Examples for Computing Real Eigenpairs 64

APPENDIX **68**

BIBLIOGRAPHY **77**

LIST OF TABLES

Table 5.1: Isolated Z-eigenpairs of the tensor in Example 5.2.1 46

Table 6.1: Comparison of `teig` and `eeig` with `PSOLVE` and `NSolve` 57

Table 6.2: Performance of `teig` on computing Qi-eigenpairs of complex random tensors 57

Table 6.3: Performance of `eeig` on computing E-eigenpairs of complex random tensors 58

Table 6.4: Performance of `teig` and `teneig` on computing generalized eigenpairs of complex random tensors 59

Table 6.5: Eigenpairs of the tensor in Example 6.2.1 61

Table 6.6: Z-eigenpairs of the tensor in Example 6.3.1 65

Table 6.7: H-eigenpairs of the tensor in Example 6.3.1 65

Table 6.8: `zeig` vs Algorithm 3.6 ([12]): CPU time 66

Table 6.9: Z-eigenvalues of the tensor in Example 6.3.3 (* denotes that the CPU time used by Algorithm 3.6 ([12]) when it finds the first three largest Z-eigenvalues) 67

Table A.1: Z-eigenpairs of the tensor in Problem 1 69

Table A.2: Z-eigenvalues of the tensor in Problem 3 70

Table A.3: Z-eigenvalues of the tensor in Problem 4 71

Table A.4: Z-eigenpairs of the tensor in Problem 5 72

Table A.5: Z-eigenpairs of the tensor in Problem 7 73

Table A.6: Z-eigenpairs of the tensor in Problem 8 73

Table A.7: Z-eigenpairs of the tensor in Problem 9 74

Table A.8: Z-eigenpairs of the tensor in Problem 10 75

Table A.9: Z-eigenpairs of the tensor in Problem 11 75

Table A.10: Z-eigenpairs of the tensor in Problem 12 76

Chapter 1

Introduction

Eigenvalues of tensors were first introduced by Qi [35] and Lim [29] in 2005. Since then, tensor eigenvalues have found applications in automatic control, statistical data analysis, diffusion tensor imaging, image authenticity verification, spectral hypergraph theory, and quantum entanglement, etc., see for example, [7, 10, 21, 35, 36, 38, 39, 40] and the references therein. The tensor eigenvalue problem has become an important subject of numerical multilinear algebra.

Various definitions of eigenvalues for tensors have been proposed in the literature, including E-eigenvalues and eigenvalues in the complex field, and Z-eigenvalues, H-eigenvalues, and D-eigenvalues in the real field [29, 35, 38]. In [6], Chang, Pearson, and Zhang introduced a notion of general eigenvalues for tensors that unifies several types of eigenvalues. Recently this definition has been further generalized by Cui, Dai, and Nie [12].

Unlike the matrix eigenvalue problem, computing eigenvalues of the third or higher order tensors is still in its infancy [18]. Several algorithms which aim at computing one or some eigenvalues of a tensor have been developed recently. These algorithms are designed for tensors of certain type, such as entry-wise nonnegative or symmetric tensors.

For nonnegative tensors, Ng, Qi, and Zhou [33] proposed a power-type method for computing the largest H-eigenvalue of a nonnegative tensor. Modified versions of the Ni-Qi-Zhou method have been proposed in [31, 48, 49].

For real symmetric tensors, Hu, Huang, and Qi [20] proposed a sequential semidefinite programming method for computing extreme Z-eigenvalues. Kolda and Mayo [23] proposed a shifted power method (SSHOPM) for computing one Z-eigenvalue. They have improved SSHOPM in [24] by updating the shift parameter adaptively. The resulting method can be used to compute a real general eigenvalue. Han [17] proposed an unconstrained optimization method for computing a real general eigenvalue for even order real symmetric tensors. The methods in [17, 23, 24] can find more eigenvalues of a symmetric tensor if they are executed multiple times using different starting points. Recently, Cui, Dai, and Nie [12] proposed a method for computing all real generalized eigenvalues.

In this article, we focus on computing *all* eigenpairs of a general real or complex tensor. As indicated in the next chapter, finding eigenpairs of a tensor actually amounts to solving a system of polynomials. Naturally one would tempt to use methods in algebraic geometry such as the Gröebner basis method and resultant method [9] for this purpose. These methods can obtain symbolic solutions of a polynomial system, which are accurate. However, they are expensive in terms of computational cost and storage requirement. Moreover, they are difficult to be parallelized. A class of numerical methods, the homotopy continuation methods, can release these shortcomings. During the past few decades, remarkable progresses have been made on homotopy continuation methods for solving polynomial systems, see for example, [3, 26, 27, 32, 41].

In this thesis we will compute complex eigenpairs of general tensors by using homotopy continuation methods to solve their corresponding polynomial systems. An attracting feature of the homotopy continuation methods is their capability of finding all isolated solutions of polynomial systems and some solutions in the positive dimensional solution components. We propose two homotopy type algorithms for computing complex eigenpairs of a tensor. These algorithms allow us to find all equivalence classes of isolated eigenpairs of a general tensor and some eigenpairs in positive dimensional eigenspaces (if there are any). We also present a homotopy method and a straightforward approach to compute real eigenpairs based on the available complex eigenpairs. Numerical results exhibited the effectiveness and efficiency of our methods.

This dissertation is organized as follows. In Chapter 2, we define mode-k generalized eigenvalues and eigenvectors which extend the matrix right eigenpairs and left eigenpairs to higher order tensors. Some properties of such eigenpairs are proved. In Chapter 3, an upper bound for the number of equivalence classes of generalized tensor eigenpairs using mixed volume is derived. In Chapter 4, based on the bounds derived in Chapter 3 two homotopy methods are presented to compute mode-k generalized complex eigenpairs. In Chapter 5, a homotopy method and a straightforward approach to compute real mode-k generalized eigenpairs are proposed. Finally, numerical results are presented in Chapter 6.

Chapter 2

Tensor eigenvalues and eigenvectors

2.1 Definition of generalized mode-k tensor eigenvalue and eigenvectors

Let $\mathbb{F} = \mathbb{C}$ or $\mathbb{R}$ be the complex field or the real field. Let $m \geq 2$, $m' \geq 2$, and n be positive integers. Denote the set of all mth-order, n-dimensional tensors on the field $\mathbb{F}$ by $\mathbb{F}^{[m,n]}$. A tensor in $\mathbb{F}^{[m,n]}$ is indexed as

$$\mathcal{A} = (A_{i_1 i_2 \cdots i_m}),$$

where $A_{i_1 i_2 \cdots i_m} \in \mathbb{F}$, for $1 \leq i_1, i_2, \cdots, i_m \leq n$.

For $x \in \mathbb{C}^n$, the tensor $\mathcal{A}$ defines a scalar function

$$\mathcal{A}x^m := \sum_{i_1, \cdots, i_m = 1}^{n} A_{i_1 i_2 \cdots i_m} x_{i_1} x_{i_2} \cdots x_{i_m}. \tag{2.1}$$

For $1 \leq k \leq m$, $\mathcal{A}^{(k)} x^{m-1}$ is an n-vector whose jth entry is defined as

$$(\mathcal{A}^{(k)} x^{m-1})_j = \sum_{i_1, \cdots, i_{k-1}, i_{k+1}, \cdots, i_m = 1}^{n} A_{i_1 \cdots i_{k-1} j i_{k+1} \cdots i_m} x_{i_1} \cdots x_{i_{k-1}} x_{i_{k+1}} \cdots x_{i_m}. \tag{2.2}$$

When $k = 1$, the notation for the vector $\mathcal{A}^{(1)} x^{m-1}$ is simplified as $\mathcal{A}x^{m-1}$.

A real tensor $\mathcal{A} \in \mathbb{R}^{[m,n]}$ is *positive definite* if the scalar function $\mathcal{A}x^m$ is positive for all

$x \in \mathbb{R}^n \backslash \{0\}$. A tensor $\mathcal{A} \in \mathbb{F}^{[m,n]}$ is *symmetric* if its entries $A_{i_1 i_2 \cdots i_m}$ are invariant under any permutations of their indices $i_1, i_2, \cdots, i_m$. A tensor $\mathcal{A} \in \mathbb{F}^{[m,n]}$ is called the mth-order unit tensor if

$$A_{i_1,\ldots,i_m} = \begin{cases} 1, & \text{when } i_1 = \cdots = i_m \\ 0, & \text{otherwise.} \end{cases}$$

We now introduce the following mode-k generalized eigenvalue definition for a general tensor $\mathcal{A}$.

DEFINITION 2.1.1 *Let $\mathcal{A} \in \mathbb{F}^{[m,n]}$ and $\mathcal{B} \in \mathbb{F}^{[m',n]}$. Assume that $\mathcal{B}x^{m'}$ is not identically zero. For $1 \leq k \leq m$, if there exist a scalar $\lambda \in \mathbb{C}$ and a vector $x \in \mathbb{C}^n \backslash \{0\}$ such that*

- *when $m \neq m'$,*

$$\mathcal{A}^{(k)} x^{m-1} = \lambda \mathcal{B} x^{m'-1}, \quad \mathcal{B} x^{m'} = 1; \tag{2.3}$$

- *when $m = m'$,*

$$\mathcal{A}^{(k)} x^{m-1} = \lambda \mathcal{B} x^{m-1}, \tag{2.4}$$

then we call λ a mode-k $\mathcal{B}$-eigenvalue of $\mathcal{A}$ and x a mode-k $\mathcal{B}$-eigenvector associated with λ. Together (λ, x) is called a mode-k $\mathcal{B}$-eigenpair of $\mathcal{A}$.

If $\lambda \in \mathbb{R}, x \in \mathbb{R}^n$, then λ is called a mode-k $\mathcal{B}_R$-eigenvalue of $\mathcal{A}$ and x a mode-k $\mathcal{B}_R$-eigenvector associated with λ, and (λ, x) a mode-k $\mathcal{B}_R$-eigenpair of $\mathcal{A}$.

Denote the set of all mode-k $\mathcal{B}$ eigenvalues of $\mathcal{A}$ by $\sigma_{\mathcal{B}}(\mathcal{A}^{(k)})$.

REMARK 2.1.1 Let (λ, x) be a mode-k $\mathcal{B}$-eigenpair of $\mathcal{A}$. By (2.3) or (2.4), (λ, x) is a solution of $\mathcal{A}^{(k)} x^{m-1} = \lambda \mathcal{B} x^{m'-1}$, so is (λ', x') with $\lambda' = t^{m-m'} \lambda$ and $x' = tx$ for $t \in \mathbb{C} \backslash \{0\}$. Hence the solution space of $\mathcal{A}^{(k)} x^{m-1} = \lambda \mathcal{B} x^{m'-1}$ consists of different equivalence classes.

We denote such an equivalence class by

$$[(\lambda, x)] := \{(\lambda', x') \,|\, \lambda' = t^{m-m'}\lambda, x' = tx, t \in \mathbb{C}\backslash\{0\}\}.$$

When $m \neq m'$, taking arbitrary $(\lambda', x') \in [(\lambda, x)]$ and substituting $x' = tx$ into $\mathcal{B}x^{m'} = 1$ in (2.3) yields $t^{m'} = 1$, which gives m' different values for t. Hence the normalization $\mathcal{B}x^{m'} = 1$ in (2.3) restricts the choices of m' representative solutions from each equivalence class.

In later discussions, we often choose one representative from each equivalence class.

REMARK 2.1.2 If only one representative is needed from each equivalence class of eigenpairs, we can solve $\mathcal{A}^{(k)}x^{m-1} = \lambda\mathcal{B}x^{m'-1}$ augmented with an additional linear equation

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n + b = 0, \tag{2.5}$$

with generic complex numbers $a_1, \ldots, a_n, b$. Then normalize the resulting solutions to satisfy $\mathcal{B}x^{m'} = 1$ in the case $m \neq m'$.

The eigenvalues/eigenvectors defined in [6, 12, 35, 38] are mode-1 eigenvalues/eigenvectors. The tensors considered in those papers are primarily real symmetric tensors. For symmetric tensors, the sets of mode-k $\mathcal{B}$-eigenpairs and mode-1 $\mathcal{B}$-eigenpairs are the same for any k. Therefore, mode-1 eigenvalues serve the purpose of those articles. Nonetheless, nonsymmetric tensors also appear in applications and theoretical studies, see, for example, [4, 5, 14, 33, 45, 46]. In [29], Lim defined mode-k eigenvalues/eigenvectors for nonsymmetric real tensors $\mathcal{A}$ when $\mathcal{B}$ is the m'th order unit tensor for some $m' \geq 2$. Definition 2.1.1 contains more general $\mathcal{A}$ and $\mathcal{B}$.

As in [6, 12], Definition 2.1.1 adapts a unified approach to define tensor eigenvalues. It covers various types of tensor eigenvalues in the literature, including

- If $\mathcal{A} \in \mathbb{R}^{[m,n]}$, $m' = 2$, and $\mathcal{B}$ is the identity matrix $I_n \in \mathbb{R}^{n \times n}$, the mode-1 $\mathcal{B}$-eigenpairs are the E-eigenpairs and the mode-1 $\mathcal{B}_R$-eigenpairs are the Z-eigenpairs defined in [35], which satisfy

$$\mathcal{A}x^{m-1} = \lambda x, \quad x^T x = 1. \tag{2.6}$$

- If $\mathcal{A} \in \mathbb{R}^{[m,n]}$, $m' = 2$ and $\mathcal{B} = D$, where $D \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix, the $\mathcal{B}_R$-eigenpairs are the D-eigenpairs defined in [38], which satisfy

$$\mathcal{A}x^{m-1} = \lambda D x, \quad x^T D x = 1. \tag{2.7}$$

- If $\mathcal{A} \in \mathbb{R}^{[m,n]}$, $m = m'$ and $\mathcal{B} = \mathcal{I}$ is the unit tensor, then mode-1 $\mathcal{B}$-eigenpairs defined in [35] satisfying

$$\mathcal{A}x^{m-1} = \lambda x^{[m-1]}, \tag{2.8}$$

  where $x^{[m-1]} = [x_1^{m-1}, x_2^{m-1}, \cdots, x_n^{m-1}]^T$, is called a Qi-eigenpair.

- If $\mathcal{A} \in \mathbb{R}^{[m,n]}$, $m = m'$ and $\mathcal{B} = \mathcal{I}$ is the unit tensor, then mode-1 $\mathcal{B}_R$-eigenvalues are the H-eigenvalues defined in [35].

2.2 Properties of generalized mode-k tensor eigenvalues

When $m = m' = 2$ and $\mathcal{B} = I_n$ (the $n \times n$ identity matrix), then mode-1 eigenvectors are right eigenvectors of $\mathcal{A}$ and mode-2 eigenvectors are left eigenvectors of $\mathcal{A}$, and the mode-1 and mode-2 eigenvalues are the eigenvalues of matrix $\mathcal{A}$, i.e., $\sigma_{\mathcal{B}}(\mathcal{A}^{(1)}) = \sigma_{\mathcal{B}}(\mathcal{A}^{(2)})$. However, when $m \geq 3$, $\sigma_{\mathcal{B}}(\mathcal{A}^{(k)})$ and $\sigma_{\mathcal{B}}(\mathcal{A}^{(l)})$ are not equal in general when $k \neq l$, unless $\mathcal{A}$ has a certain type of symmetry. The following example illustrates this difference.

EXAMPLE 2.2.1 Let the tensor $\mathcal{A} \in \mathbb{R}^{[3,2]}$ be

$$A_{111} = 1, A_{121} = 2, A_{211} = 3, A_{221} = 4,$$
$$A_{112} = 5, A_{122} = 6, A_{212} = 7, A_{222} = 0.$$

Choose $m' = 2$ and $\mathcal{B} = I_2$ (the 2×2 identity matrix). In this case, if (λ, x) is an $\mathcal{B}$-eigenpair of $\mathcal{A}$, so is $(-\lambda, -x)$. We follow [4], taking (λ, x) and $(-\lambda, -x)$ as the same eigenpair. Then

$$\sigma_{\mathcal{B}}(\mathcal{A}^{(1)}) = \{0.4105, 4.3820, 9.8995\},$$

$$\sigma_{\mathcal{B}}(\mathcal{A}^{(2)}) = \{0.2851, 4.3536, 9.5652\},$$

$$\sigma_{\mathcal{B}}(\mathcal{A}^{(3)}) = \{0.2936, 4.3007, 9.4025\}.$$

Clearly, $\sigma_{\mathcal{B}}(\mathcal{A}^{(k)}) \neq \sigma_{\mathcal{B}}(\mathcal{A}^{(l)})$ when $k \neq l$.

PROPOSITION 2.2.1 *Let $\mathcal{A} \in \mathbb{F}^{[m,n]}$ and $\mathcal{B} \in \mathbb{F}^{[m',n]}$. If (λ, x) is a mode-k $\mathcal{B}$-eigenpair and (μ, x) is a mode-l $\mathcal{B}$-eigenpair of $\mathcal{A}$ such that $\mathcal{B}x^{m'} \neq 0$, then $\lambda = \mu$.*

Proof: Since (λ, x) is a mode-k $\mathcal{B}$-eigenpair of $\mathcal{A}$,

$$\mathcal{A}^{(k)}x^{m-1} = \lambda \mathcal{B}x^{m'-1}. \tag{2.9}$$

Recall that $\mathcal{A}^{(k)}x^{m-1}$ is defined by

$$\mathcal{A}^{(k)}x^{m-1} = \begin{pmatrix} (\mathcal{A}^{(k)}x^{m-1})_1 \\ \vdots \\ (\mathcal{A}^{(k)}x^{m-1})_j \\ \vdots \\ (\mathcal{A}^{(k)}x^{m-1})_n \end{pmatrix}$$

with $(\mathcal{A}^{(k)}x^{m-1})_j$ given by (2.2). Similarly,

$$\mathcal{B}x^{m'-1} = \begin{pmatrix} (\mathcal{B}x^{m'-1})_1 \\ \vdots \\ (\mathcal{B}x^{m'-1})_j \\ \vdots \\ (\mathcal{B}x^{m'-1})_n \end{pmatrix},$$

where

$$(\mathcal{B}x^{m'-1})_j = \sum_{i_2,\ldots,i_{\mathrm{m}'}=1}^{n} B_{ji_2\ldots i_{\mathrm{m}'}} x_{i_2} \ldots x_{i_{\mathrm{m}'}}.$$

Then multiplying both sides of (2.9) by x^T from the left yields

$$x^T\mathcal{A}^{(k)}x^{m-1} = \lambda x^T \mathcal{B}x^{m'-1}.$$

By definition $x^T\mathcal{A}^{(k)}x^{m-1} = \mathcal{A}x^m$ and $x^T\mathcal{B}x^{m'-1} = \mathcal{B}x^{m'}$. So

$$\mathcal{A}x^m = \lambda\mathcal{B}x^{m'}.$$

Similarly, (μ, x) is a mode-l $\mathcal{B}$-eigenpair of $\mathcal{A}$ will imply that $\mathcal{A}x^m = \mu\mathcal{B}x^{m'}$. Therefore, $\lambda\mathcal{B}x^{m'} = \mu\mathcal{B}x^{m'}$. Hence if $\mathcal{B}x^{m'} \neq 0$, then $\lambda = \mu$. □

Let $\mathcal{A} \in \mathbb{F}^{[m,n]}$. For $1 \leq k < l \leq m$, tensor $\mathcal{G} \in \mathbb{F}^{[m,n]}$ is said to be the $\langle k, l\rangle$ *transpose* of $\mathcal{A}$ if

$$\mathcal{G}_{i_1\cdots i_{k-1}i_l i_{k+1}\cdots i_{l-1}i_k i_{l+1}\cdots i_m} = \mathcal{A}_{i_1\cdots i_{k-1}i_k i_{k+1}\cdots i_{l-1}i_l i_{l+1}\cdots i_m},$$

for all $1 \leq i_1, \cdots, i_m \leq m$. Denote the $\langle k, l\rangle$ *transpose* of $\mathcal{A}$ by $\mathcal{A}^{\langle k,l\rangle}$. We say tensor $\mathcal{A}$ is $\langle k, l\rangle$ *partially symmetric* if

$$\mathcal{A}^{\langle k,l\rangle} = \mathcal{A}.$$

It is clear that the following proposition holds.

PROPOSITION 2.2.2 *Let $\mathcal{A} \in \mathbb{F}^{[m,n]}$ and $\mathcal{B} \in \mathbb{F}^{[m',n]}$. Assume $\mathcal{B}x^{m'}$ is not identically zero. Let k, l be integers such that $1 \leq k < l \leq m$. Then*

- *(λ, x) is a mode-k $\mathcal{B}$-eigenpair of $\mathcal{A}$ if and only if it is a mode-l $\mathcal{B}$-eigenpair of $\mathcal{A}^{\langle k,l\rangle}$.*
- *The sets of mode-k $\mathcal{B}$-eigenpairs and mode-l $\mathcal{B}$-eigenpairs are the same if $\mathcal{A}$ is $\langle k, l\rangle$ partially symmetric.*

For a symmetric tensor, the following proposition was shown in [23].

PROPOSITION 2.2.3 *Let $\mathcal{A} \in \mathbb{F}^{[m,n]}$ be a symmetric tensor. Then the gradient of $\mathcal{A}x^m$ is given by*

$$\nabla(\mathcal{A}x^m) = m\mathcal{A}x^{m-1}.$$

REMARK 2.2.1 Theoretical properties of mode-1 eigenvalues of tensors such as the Perron-Frobenius theory ([5, 14, 45, 46]) for nonnegative tensors can be parallelly developed for mode-k eigenvalues. However, as Horn and Johnson announced in [19]: "One should not dismiss left eigenvectors as merely a parallel theoretical alternative to right eigenvectors. Each type of eigenvector can convey different information about a matrix," we believe mode-1 through mode-m eigenpairs can convey different information about a general tensor of order $m \geq 3$.

Chapter 3

The upper bound of the number of equivalence classes of tensor eigenpairs

In this chapter, we present the derivation of an upper bound for the number of equivalence classes of mode-k eigenpairs.

3.1 Introduction

The number of equivalence classes of tensor eigenpairs has been discussed in many papers. For Qi-eigenvalue problems in (2.8), if (λ, x) is a Qi-eigenpair, so is (λ, tx) for any $t \in \mathbb{C}\backslash\{0\}$. Qi [35] first showed that the Qi-eigenvalues are roots of a univariate polynomial and proved the following theorem.

THEOREM 3.1.1 *Let $\mathcal{A} \in \mathbb{R}^{[m,n]}$ be a symmetric tensor and m be an even integer. Then $\mathcal{A}$ has exactly $n(m-1)^{n-1}$ Qi-eigenvalues.*

For E-eigenvalue problems as defined in (2.6), the following theorem was proved in [4] using techniques from toric variety [11, 15].

THEOREM 3.1.2 *If a tensor* $\mathcal{A} \in \mathbb{C}^{[m,n]}$ *has finitely many equivalence classes of E-eigenpairs over* $\mathbb{C}$ *then their number, counting multiplicities, is equal to*

$$\frac{(m-1)^n - 1}{m-2}.$$

If the entries of $\mathcal{A}$ *are sufficiently generic, then all multiplicities are equal to 1, so there are exactly* $((m-1)^n - 1)/(m-2)$ *equivalence classes of eigenpairs.*

As in Definition 2.1.1, Remark 2.1.1 and Remark 2.1.2, the number of equivalence classes of mode-k generalized $\mathcal{B}$-eigenpairs for general tensors $\mathcal{A} \in \mathbb{C}^{[m,n]}$ and $\mathcal{B} \in \mathbb{C}^{[m',n]}$ is equal to the number of isolated solutions of the following system of polynomials

$$T(\lambda, x) = \begin{pmatrix} (\mathcal{A}^{(k)} x^{m-1})_1 - \lambda(\mathcal{B} x^{m'-1})_1 \\ \vdots \\ (\mathcal{A}^{(k)} x^{m-1})_n - \lambda(\mathcal{B} x^{m'-1})_n \\ a_1 x_1 + a_2 x_2 + \cdots + a_n x_n + b \end{pmatrix} = 0, \tag{3.1}$$

where λ and $x := (x_1, \cdots, x_n)^T$ are the unknowns, $a_1, \ldots, a_n, b$ are random complex numbers. We may therefore use the classic results on the number of solutions of a polynomial system to study the number of equivalence classes of tensor eigenpairs.

3.2 Preliminaries

In the first place, we introduce some commonly used notations and definitions. Let $P(x) := (p_1(x), \ldots, p_n(x))^T$ be a polynomial system with $x := (x_1, \ldots, x_n)^T$. For $\alpha := (\alpha_1, \ldots, \alpha_n) \in$

$(\mathbb{Z}^n_{\geq 0})^T$, write $x^\alpha := x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ and $|\alpha| = \alpha_1 + \cdots + \alpha_n$. Consider the polynomial system

$$P(x) := \begin{pmatrix} p_1(x) := \sum_{\alpha \in S_1} c_{1,\alpha} x^\alpha \\ \vdots \\ p_n(x) := \sum_{\alpha \in S_n} c_{n,\alpha} x^\alpha \end{pmatrix}, \tag{3.2}$$

where $S_1, \ldots, S_n$ are given finite subsets of $(\mathbb{Z}^n_{\geq 0})^T$ and $c_{i,\alpha} \in \mathbb{C}^* := \mathbb{C}\backslash\{0\}$. For each $i = 1, \ldots, n$, S_i is called the support of $p_i(x)$ and its convex hull $R_i := \mathrm{conv}(S_i)$ in $\mathbb{R}^n$ is called the Newton polytope of $p_i(x)$. The n-tuple $(S_1, \ldots, S_n)$ is called the support of $P(x)$. For positive variables $\lambda_1, \ldots, \lambda_n$, let $\lambda_1 R_1 + \cdots + \lambda_n R_n$ be the *Minkowski* sum of $\lambda_1 R_1, \ldots, \lambda_n R_n$, i.e.,

$$\lambda_1 R_1 + \cdots + \lambda_n R_n := \{\lambda_1 r_1 + \cdots + \lambda_n r_n \,|\, r_i \in R_i, i = 1, \ldots, n\}.$$

It was shown in [8] that the n-dimensional volume of $\lambda_1 R_1 + \cdots + \lambda_n R_n$, denoted by $\mathrm{Vol}_n(\lambda_1 R_1 + \cdots + \lambda_n R_n)$, is a homogeneous polynomial of degree n in $\lambda_1, \ldots, \lambda_n$. The coefficient of the monomial $\lambda_1 \lambda_2 \cdots \lambda_n$ in this polynomial is called the *mixed volume* of $R_1, \ldots, R_n$, denoted by $\mathrm{MV}_n(R_1, \ldots, R_n)$. Sometimes we call it mixed volume of the supports $S_1, \ldots, S_n$, denoted by $\mathrm{MV}_n(S_1, \ldots, S_n)$, or the mixed volume of $P(x)$ if no ambiguities exist. The following theorem relates the number of isolated solutions in $(\mathbb{C}^*)^n$ of a polynomial system to its mixed volume.

THEOREM 3.2.1 (**Bernstein's Theorem**) [2] *The number of isolated zeros in* $(\mathbb{C}^*)^n$*, counting multiplicities, of a polynomial system* $P(x) = (p_1(x), \ldots, p_n(x))^T$ *with supports* $S_1, \ldots, S_n$ *is bounded above by the mixed volume* $\mathrm{MV}_n(S_1, \ldots, S_n)$*. Moreover, for generic* *choices of the coefficients in* p_i*, the number of isolated zeros in* $(\mathbb{C}^*)^n$ *is exactly equal to* $\mathrm{MV}_n(S_1, \ldots, S_n)$.

A limitation of Theorem 3.2.1 is that it only counts the isolated zeros of a polynomial system in $(\mathbb{C}^*)^n$ rather than $\mathbb{C}^n$. To deal with this issue, Li and Wang gave the following theorem.

THEOREM 3.2.2 [30] *The number of isolated zeros in* $\mathbb{C}^n$*, counting multiplicities, of a polynomial system* $P(x) = (p_1(x), \ldots, p_n(x))^T$ *with supports* $S_1, \ldots, S_n$ *is bounded above by the mixed volume* $\mathrm{MV}_n(S_1 \cup \{0\}, \ldots, S_n \cup \{0\})$.

The following lemma was given as an exercise in [8].

LEMMA 3.2.1 *For a polynomial system* $P(x) = (p_1(x), \ldots, p_n(x))^T$ *with supports* $S_1 = S_2 = \cdots = S_n = S$*,*

$$\mathrm{MV}_n(S, \ldots, S) = n!\mathrm{Vol}_n(\mathrm{conv}(S)).$$

Recall that an n-simplex is the convex hull of $n+1$ points $z_1, \ldots, z_{n+1}$ such that $z_2 - z_1, \ldots, z_{n+1} - z_1$ are linearly independent in $(\mathbb{R}^n)^T$, and by a simple computation

$$\mathrm{Vol}_n(\mathrm{conv}(z_1, z_2, \ldots, z_{n+1})) = \frac{1}{n!}\left|\det\begin{pmatrix} z_2 - z_1 \\ \vdots \\ z_{n+1} - z_1 \end{pmatrix}\right|.$$

3.3 Main Theorem based on Bernstéin's Theorem

An upper bound for the number of equivalence classes of mode-k eigenpairs which generalizes results in [4, 35] is given in the following theorem.

THEOREM 3.3.1 *Let $\mathcal{A} \in \mathbb{C}^{[m,n]}$ and $\mathcal{B} \in \mathbb{C}^{[m',n]}$. Assume $\mathcal{B}x^{m'}$ is not identically zero and k is an integer satisfying $1 \leq k \leq m$. Assume $\mathcal{A}$ has finitely many equivalence classes of mode-k $\mathcal{B}$-eigenpairs over $\mathbb{C}$.*

(a) If $m = m'$, then the number of equivalence classes of mode-k $\mathcal{B}$-eigenpairs, counting multiplicities, is bounded by

$$n(m-1)^{n-1}.$$

If $\mathcal{A}$ and $\mathcal{B}$ are generic tensors, then $\mathcal{A}$ has exactly $n(m-1)^{n-1}$ equivalence classes of mode-k $\mathcal{B}$-eigenpairs, counting multiplicities.

(b) If $m \neq m'$, then the number of equivalence classes of mode-k $\mathcal{B}$-eigenpairs, counting multiplicities, is bounded by

$$\frac{(m-1)^n - (m'-1)^n}{m - m'}.$$

If $\mathcal{A}$ and $\mathcal{B}$ are generic tensors, then $\mathcal{A}$ has exactly $((m-1)^n - (m'-1)^n)/(m-m')$ equivalence classes of mode-k $\mathcal{B}$-eigenpairs, counting multiplicities.

Proof: As mentioned before, the number of equivalence classes of mode-k $\mathcal{B}$-eigenpairs of $\mathcal{A}$ is equal to the number of solutions of (3.1). For the random hyperplane $a_1x_1+\cdots+a_nx_n+b=0$ in (3.1), we suppose without loss, $a_n \neq 0$. Then

$$x_n = c_1x_1 + \cdots + c_{n-1}x_{n-1} + d, \tag{3.3}$$

where $c_i = -a_i/a_n$ for $i = 1, \ldots, n-1$ and $d = -b/a_n$, and the number of solutions of (3.1) in $\mathbb{C}^{n+1}$ is the same as the number of solutions in $\mathbb{C}^n$ of the resulting system $T^*(\lambda, x_1, \ldots, x_{n-1})$

by substituting (3.3) into the first n equations of (3.1). Let $z := (\lambda, x_1, \ldots, x_{n-1})$ and let the supports of T^* be $S_1, \ldots, S_n$. We claim that

$$\mathrm{MV}_n(S_1 \cup \{0\}, \ldots, S_n \cup \{0\}) = \begin{cases} n(m-1)^{n-1}, & m = m' \\ \dfrac{(m-1)^n - (m'-1)^n}{m-m'}, & m \neq m'. \end{cases} \tag{3.4}$$

As a consequence, when N is the number of equivalence classes of mode-k $\mathcal{B}$-eigenpairs of $\mathcal{A}$ over $\mathbb{C}$, then

$$N \leq n(m-1)^{n-1}$$

for $m = m'$ and

$$N \leq \frac{(m-1)^n - (m'-1)^n}{m-m'}$$

for $m \neq m'$. When $\mathcal{A}$ and $\mathcal{B}$ are generic, the above equality holds by Theorem 3.2.1 and Theorem 3.2.2.

To prove (3.4), let $\bar{\mathcal{A}} \in \mathbb{C}^{[m,n]}$ and $\bar{\mathcal{B}} \in \mathbb{C}^{[m',n]}$ be generic tensors. Similar to (3.1) the polynomial system corresponding to the eigenproblem $\bar{\mathcal{A}}^{(k)}x^{m-1} = \lambda\bar{\mathcal{B}}x^{m'-1}$ is

$$\bar{T}(\lambda, x) = \begin{pmatrix} (\bar{\mathcal{A}}^{(k)}x^{m-1})_1 - \lambda(\bar{\mathcal{B}}x^{m'-1})_1 \\ \vdots \\ (\bar{\mathcal{A}}^{(k)}x^{m-1})_n - \lambda(\bar{\mathcal{B}}x^{m'-1})_n \\ a_1x_1 + a_2x_2 + \cdots + a_nx_n + b \end{pmatrix} = 0. \tag{3.5}$$

Since $\bar{\mathcal{A}}$ and $\bar{\mathcal{B}}$ are generic, without loss of generality one may assume all monomials

$$\{x_1^{\alpha_1}x_2^{\alpha_2}\ldots x_n^{\alpha_n} \,\big|\, \alpha_i \in \mathbb{Z}_{\geq 0},\, \alpha_1 + \alpha_2 + \cdots + \alpha_n = m-1\}$$

and

$$\{\lambda x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n} \Big| \alpha_i \in \mathbb{Z}_{\geq 0},\, \alpha_1 + \alpha_2 + \dots + \alpha_n = m' - 1\}$$

will appear in each of the first n equations in (3.5). Therefore, after substituting (3.3) into the first n equations of (3.5), each equation of the new system $\bar{T}^*(z) := \bar{T}^*(\lambda, x_1, \dots, x_{n-1})$ contains all monomials

$$\{x_1^{\alpha_1} x_2^{\alpha_2} \dots x_{n-1}^{\alpha_{n-1}} \Big| \alpha_i \in \mathbb{Z}_{\geq 0},\, \alpha_1 + \alpha_2 + \dots + \alpha_{n-1} \leq m - 1\}$$

and

$$\{\lambda x_1^{\alpha_1} x_2^{\alpha_2} \dots x_{n-1}^{\alpha_{n-1}} \Big| \alpha_i \in \mathbb{Z}_{\geq 0},\, \alpha_1 + \alpha_2 + \dots + \alpha_{n-1} \leq m' - 1\}$$

So the supports $\bar{S}_1, \dots, \bar{S}_n$ of $\bar{T}^*$ are all equal to

$$\bar{S} := \{(0, \alpha) \,\big|\, \alpha \in (\mathbb{Z}_{\geq 0}^{n-1})^T, |\alpha| \leq m - 1\} \cup \{(1, \alpha) \,\big|\, \alpha \in (\mathbb{Z}_{\geq 0}^{n-1})^T, |\alpha| \leq m' - 1\}.$$

Let $\bar{Q}$ be the convex hull of $\bar{S}$. Then vertices of $\bar{Q}$ are given by the following points in $(\mathbb{Z}^n)^T$:

$$\begin{aligned}
z_0 &= (0, 0, \dots, 0), \\
z_1 &= (0, 0, \dots, 0, m - 1), \\
z_2 &= (0, 0, \dots, 0, m - 1, 0), \\
&\vdots \\
z_{n-1} &= (0, m - 1, 0, \dots, 0),
\end{aligned}$$

$$
\begin{aligned}
z_n &= (1, 0, \ldots, 0), \\
z_{n+1} &= (1, 0, \ldots, 0, m' - 1), \\
z_{n+2} &= (1, 0, \ldots, 0, m' - 1, 0), \\
&\vdots \\
z_{2n-1} &= (1, m' - 1, 0, \ldots, 0).
\end{aligned}
$$

Denote the i-th unit vector in $(\mathbb{R}^n)^T$ by e_i for $i = 1, \ldots, n$. Then

$$
z_i = \begin{cases}
0, & i = 0 \\
(m-1)e_{n+1-i}, & 1 \leq i \leq n-1 \\
e_1, & i = n \\
e_1 + (m'-1)e_{2n+1-i}, & n+1 \leq i \leq 2n-1
\end{cases}
$$

To compute the volume of $\bar{Q}$, we divide it into following n simplices

$$
\begin{aligned}
\bar{Q}_1 &:= \operatorname{conv}(z_0, z_1, \ldots, z_n), \\
\bar{Q}_2 &:= \operatorname{conv}(z_1, z_2, \ldots, z_{n+1}), \\
&\vdots \\
\bar{Q}_i &:= \operatorname{conv}(z_{i-1}, z_i, \ldots, z_{n+i-1}), \\
&\vdots \\
\bar{Q}_n &:= \operatorname{conv}(z_{n-1}, z_n, \ldots, z_{2n-1}).
\end{aligned}
$$

It follows that

$$
\begin{aligned}
\mathrm{Vol}_n(\bar{Q}_1) &= \frac{1}{n!}\left|\det\begin{pmatrix} z_1 - z_0 \\ z_2 - z_0 \\ \vdots \\ z_n - z_0 \end{pmatrix}\right| = \frac{1}{n!}\left|\det\begin{pmatrix} (m-1)e_n \\ (m-1)e_{n-1} \\ \vdots \\ (m-1)e_2 \\ e_1 \end{pmatrix}\right| \\
&= \frac{(m-1)^{n-1}}{n!}\left|\det\begin{pmatrix} e_n \\ e_{n-1} \\ \vdots \\ e_2 \\ e_1 \end{pmatrix}\right| = \frac{(m-1)^{n-1}}{n!}\left|\det\begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix}\right| \\
&= \frac{(m-1)^{n-1}}{n!}. \qquad (3.6)
\end{aligned}
$$

Obviously that z_n is contained in each $\bar{Q}_i$ for $i = 2, \ldots, n$, so

$$
\mathrm{Vol}_n(\bar{Q}_i) = \frac{1}{n!}\left|\det\begin{pmatrix} z_{i-1} - z_n \\ z_i - z_n \\ \vdots \\ z_{n-1} - z_n \\ z_{n+1} - z_n \\ \vdots \\ z_{n+i-1} - z_n \end{pmatrix}\right| = \frac{1}{n!}\left|\det\begin{pmatrix} (m-1)e_{n+2-i} - e_1 \\ (m-1)e_{n+1-i} - e_1 \\ \vdots \\ (m-1)e_2 - e_1 \\ (m'-1)e_n \\ (m'-1)e_{n-1} \\ \vdots \\ (m'-1)e_{n+2-i} \end{pmatrix}\right|.
$$

Further computation gives

$$\mathrm{Vol}_n(\bar{Q}_i) = \frac{1}{n!}\left|\det\begin{pmatrix} -e_1 \\ (m-1)e_{n+1-i} \\ \vdots \\ (m-1)e_2 \\ (m'-1)e_n \\ (m'-1)e_{n-1} \\ \vdots \\ (m'-1)e_{n+2-i} \end{pmatrix}\right| = \frac{(m-1)^{n-i}(m'-1)^{i-1}}{n!}\left|\det\begin{pmatrix} -e_1 \\ e_{n+1-i} \\ \vdots \\ e_2 \\ e_n \\ e_{n-1} \\ \vdots \\ e_{n+2-i} \end{pmatrix}\right|$$

$$= \frac{(m-1)^{n-i}(m'-1)^{i-1}}{n!}\left|\det\begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix}\right| = \frac{(m-1)^{n-i}(m'-1)^{i-1}}{n!}. \tag{3.7}$$

Comparing (3.6) with (3.7), (3.7) actually also holds for $i = 1$. Thus

$$\mathrm{Vol}_n(\bar{Q}) = \sum_{i=1}^{n} \mathrm{Vol}_n(\bar{Q}_i) = \sum_{i=1}^{n} \frac{(m-1)^{n-i}(m'-1)^{i-1}}{n!}.$$

Therefore, by Lemma 3.2.1

$$\mathrm{MV}_n(\bar{S}_1, \ldots, \bar{S}_n) = n!\mathrm{Vol}_n(\bar{Q}) = \sum_{i=1}^{n}(m-1)^{n-i}(m'-1)^{i-1}.$$

Note that for $i = 1, \ldots, n$, $S_i \cup \{0\} \subset \bar{S}_i$. Hence

$$\mathrm{MV}_n(S_1 \cup \{0\}, \ldots, S_n \cup \{0\}) \leq \mathrm{MV}_n(\bar{S}_1, \ldots, \bar{S}_n) = \sum_{i=1}^{n}(m-1)^{n-i}(m'-1)^{i-1}.$$

Consequently,

$$\mathrm{MV}_n(S_1 \cup \{0\}, \ldots, S_n \cup \{0\}) \le \sum_{i=1}^{n} (m-1)^{n-1} = n(m-1)^{n-1} \tag{3.8}$$

for $m = m'$ and

$$\begin{aligned}
\mathrm{MV}_n(S_1 \cup \{0\}, \ldots, S_n \cup \{0\}) &\le \frac{(m-1)^n}{m'-1} \sum_{i=1}^{n} \left(\frac{m'-1}{m-1}\right)^i \\
&= \frac{(m-1)^n}{m'-1} \frac{\frac{m'-1}{m-1}\left(1 - \left(\frac{m'-1}{m-1}\right)^n\right)}{1 - \frac{m'-1}{m-1}} \\
&= (m-1)^n \frac{1 - \left(\frac{m'-1}{m-1}\right)^n}{m - m'} \\
&= \frac{(m-1)^n - (m'-1)^n}{m - m'}
\end{aligned} \tag{3.9}$$

for $m \ne m'$.

On the other hand, for $m = m'$, let the diagonal tensors $\mathcal{A} \in \mathbb{C}^{[m,n]}$ and $\mathcal{B} \in \mathbb{C}^{[m,n]}$ be such that $A_{ii\ldots i} = i$, $B_{ii\ldots i} = 1$ and all other entries zero. Then the number of equivalence classes of mode-k $\mathcal{B}$-eigenpairs of $\mathcal{A}$ is equal to the number of solutions to the following system of polynomials

$$\begin{pmatrix} x_1^{m-1} - \lambda x_1^{m-1} \\ 2x_2^{m-1} - \lambda x_2^{m-1} \\ \vdots \\ n x_n^{m-1} - \lambda x_n^{m-1} \\ x_1 + x_2 + \cdots + x_n - 1 \end{pmatrix} = 0.$$

The zeros of this system are $(\lambda, x_1, \ldots, x_n) = (1, 1, 0, \ldots, 0), (2, 0, 1, 0, \ldots, 0), \ldots, (n, 0, \ldots, 0, 1)$ and each zero has multiplicity $(m-1)^{n-1}$. By Theorem 3.2.2,

$$\mathrm{MV}_n(S_1 \cup \{0\}, \ldots, S_n \cup \{0\}) \geq n(m-1)^{n-1}.$$

Combining with (3.8), we have

$$\mathrm{MV}_n(S_1 \cup \{0\}, \ldots, S_n \cup \{0\}) = n(m-1)^{n-1}.$$

For $m \neq m'$, consider the diagonal tensors $\mathcal{A} \in \mathbb{C}^{[m,n]}$ and $\mathcal{B} \in \mathbb{C}^{[m',n]}$ such that $A_{ii\ldots i} = 1$, $B_{ii\ldots i} = 1$ and all other entries are zero. Assume $m > m'$. Then the number of equivalence classes of mode-k $\mathcal{B}$-eigenpairs of $\mathcal{A}$ is equal to the number of solutions to the following system of polynomials

$$\begin{pmatrix} x_1^{m-1} - \lambda x_1^{m'-1} \\ \vdots \\ x_n^{m-1} - \lambda x_n^{m'-1} \\ a_1x_1 + a_2x_2 + \cdots + a_nx_n + b \end{pmatrix} = \begin{pmatrix} x_1^{m'-1}(x_1^{m-m'} - \lambda) \\ \vdots \\ x_n^{m'-1}(x_n^{m-m'} - \lambda) \\ a_1x_1 + a_2x_2 + \cdots + a_nx_n + b \end{pmatrix} = 0, \qquad (3.10)$$

where $a_1, \ldots, a_n, b$ are random complex numbers. Obviously $x = 0$ cannot be a solution since it fails to satisfy the augmented random hyperplane. As discussed in Remark 2.1.1, the hyperplane is added to ensure that only one representative from each equivalence class can be selected. So we need to find the number of equivalence classes of eigenpairs (λ, x) from the first n equations of (3.10). For each fixed λ, there are $m-1$ choices for each x_i and among those choices, $m'-1$ of them are zeros. Excluding those combinations which make $x = 0$, there are $(m-1)^n - (m'-1)^n$ choices for $x_1, \ldots, x_n$ in total. Furthermore, for each eigenvalue λ, let x be the corresponding solution of the first n equations of (3.10). Then tx with $t^{m-m'} = 1$ is also a solution associated with λ. Thus for each eigenvalue λ, solving the first n equations of (3.10) results in $m - m'$ corresponding solutions of x. By Remark 2.1.1, these $m - m'$ solutions are actually equivalent. Therefore, there should be $((m-1)^n - (m'-1)^n)/(m-m')$ equivalence classes of eigenpairs, i.e., (3.10) must have $((m-1)^n - (m'-1)^n)/(m-m')$ isolated zeros in $\mathbb{C}^{n+1}$ in total. By Theorem 3.2.2,

$$\mathrm{MV}_n(S_1 \cup \{0\}, \ldots, S_n \cup \{0\}) \geq ((m-1)^n - (m'-1)^n)/(m-m').$$

Combining the above inequality with (3.9) gives

$$\mathrm{MV}_n(S_1 \cup \{0\}, \ldots, S_n \cup \{0\}) = ((m-1)^n - (m'-1)^n)/(m-m').$$

$\square$

REMARK 3.3.1

(a) For Qi-eigenpairs, we have $m' = m$. Using (a) in Theorem 3.3.1 the upper bound of the number of equivalence classes of Qi-eigenpairs is $n(m-1)^{n-1}$. This result includes Theorem 3.1.1 proved in [35], in which this bound of the number of Qi-eigenvalues is restricted to a symmetric tensor when m is even.

(b) For E-eigenpairs, $m' = 2$. Using (b) in Theorem 3.3.1 the upper bound of the number of equivalence classes of E-eigenpairs is $((m-1)^n - 1)/(m-2)$, which agrees with Theorem 3.1.2 proved in [4].

(c) The upper bounds provided in Theorem 3.3.1 is very useful in designing effective homotopy methods for computing mode-k generalized eigenpairs. In fact, the homotopy method described in Algorithm 4.3.1 for the case $m = m'$ in the next chapter relies on the bound $n(m-1)^{n-1}$.

Chapter 4

Computing complex tensor eigenpairs by homotopy methods

Let $\mathcal{A} \in \mathbb{C}^{[m,n]}$ and $\mathcal{B} \in \mathbb{C}^{[m',n]}$. As discussed in Chapter 2, the problem of computing mode-k $\mathcal{B}$-eigenpairs of $\mathcal{A}$ in (2.3) is equivalent to solving (3.1), and when $m \neq m'$, we normalize (λ, x) to satisfy $\mathcal{B}x^{m'} = 1$. For polynomial system (3.1), the homotopy continuation method is commonly used to find its numerical solutions.

4.1 Using homotopy continuation methods to solve polynomial systems

The basic idea of using homotopy continuation method to solve a general polynomial system $P(x) = (p_1(x), \ldots, p_n(x))^T = 0$ as defined in (3.2) is to deform $P(x)$ to another polynomial system $Q(x)$ with known solutions in the first place. Then under certain conditions, a smooth curve that emanates from a solution of $Q(x) = 0$ will lead to a solution of $P(x) = 0$.

For the classical linear homotopy [1]:

$$H(x,t) = (1-t)\gamma Q(x) + tP(x) = 0, \quad t \in [0,1], \tag{4.1}$$

where γ is a generic nonzero complex number, if $Q(x)$ is chosen properly, the following properties hold:

- **Property 0** *(Triviality)* The solutions of $Q(x) = 0$ are known or easy to solve.

- **Property 1** *(Smoothness)* The solution set of $H(x,t) = 0$ for $0 < t < 1$ consists of a finite number of smooth paths, each can be parameterized by t in $[0, 1)$.

- **Property 2** *(Accessibility)* Every isolated solution of $H(x, 1) = P(x) = 0$ can be reached by some path emanating from a solution of $H(x, 0) = Q(x) = 0$.

Let $d_1, \ldots, d_n$ be the degrees of polynomials $p_1(x), \ldots, p_n(x)$ respectively. Then $d_1 \times d_2 \times \cdots \times d_n$ is commonly known as the *total degree* or the Bézout number of the system $P(x)$. A typical choice of a starting system $Q(x)$ in (4.1) satisfying Properties 0-2 is

$$Q(x) := \begin{pmatrix} a_1 x_1^{d_1} - b_1 \\ \vdots \\ a_n x_n^{d_n} - b_n \end{pmatrix},$$

where $a_1, \ldots, a_n, b_1, \ldots, b_n$ are random complex numbers. The corresponding linear homotopy is known as the *total degree homotopy*, see [27, 32, 43]. Here all the $d_1 \times d_2 \times \cdots \times d_n$ solutions of $Q(x) = 0$ can be easily solved. By tracking $d_1 \times d_2 \times \cdots \times d_n$ number of solution paths of (4.1) we can find all the isolated solutions of $P(x) = 0$. However, most of the polynomial systems in applications usually have far fewer than $d_1 \times d_2 \times \cdots \times d_n$ isolated solutions. In this case, many of the $d_1 \times d_2 \times \cdots \times d_n$ paths will diverge to infinity as $t \to 1$ resulting in huge wasteful computations.

The *polyhedral* homotopy continuation method [22] based on Bernstein's Theorem [2] makes a major advance in this regard. For this homotopy, the number of paths that need to be traced is the *mixed volume* of the polynomial system, which generally provides a much tighter bound than Bézout's number for the number of isolated zeros of the polynomial system. In most occasions the new method substantially reduces the amount of extraneous paths. The method of using polyhedral homotopy continuation method to solve $P(x) = (p_1(x), \dots, p_n(x))^T$ in (3.2) with supports $S_1, \dots, S_n$ contains the following two major stages (See [26, 27]):

Stage 1. *Mixed Cell Computation.* For each support S_i, let $S_i' := S_i \cup \{0\}$ for $i = 1, \dots, n$. Recall that the number of isolated zeros of (3.2) in $\mathbb{C}^n$ is bounded above by $\mathrm{MV}_n(S_1', \dots, S_n')$ by Theorem 3.2.2. For $i = 1, \dots, n$, let $w_i : S_i' \to \mathbb{R}$ be a real-valued function of randomly chosen images and write $\hat{S}_i' := \{\hat{\alpha} := (\alpha, w_j(\alpha)) \,|\, \alpha \in S_i'\}$. For $\hat{\beta} := (\beta, 1) \in \mathbb{R}^{n+1}$ with $\beta = (\beta_1, \dots, \beta_n)^T \in \mathbb{R}^n$, let $\langle \hat{\alpha}, \hat{\beta} \rangle$ be the Euclidean inner product of $\hat{\alpha}$ and $\hat{\beta}$. A collection of pairs $(\{\alpha_{11}, \alpha_{12}\}, \dots, \{\alpha_{n1}, \alpha_{n2}\})$ with $\{\alpha_{i1}, \alpha_{i2}\} \subseteq S_i'$ for $j = 1, \dots, n$ is called a *mixed cell* of $S_1', \dots, S_n'$ if there exists $\hat{\beta} := (\beta, 1) \in \mathbb{R}^{n+1}$ with $\beta \in \mathbb{R}^n$ such that

$$\langle \hat{\alpha}_{i1}, \hat{\beta} \rangle = \langle \hat{\alpha}_{i2}, \hat{\beta} \rangle < \langle \hat{\alpha}, \hat{\beta} \rangle, \quad \text{for } \alpha \in S_i' \backslash \{\alpha_{i1}, \alpha_{i2}\}, \tag{4.2}$$

where β is called the *inner normal* of this mixed cell.

All the mixed cells with their inner normals in Stage 1 can be found by solving various linear programming problems [25].

Stage 2. *Construction of a polyhedral homotopy and path tracking.* For each mixed cell $(\{\alpha_{11}, \alpha_{12}\}, \dots, \{\alpha_{n1}, \alpha_{n2}\})$ with inner normal β satisfying (4.2) found in Stage 1, let

$$d_i := \min_{\alpha \in S_i'} \langle \hat{\alpha}, \hat{\beta} \rangle, \quad i = 1, \dots, n.$$

Construct the following polyhedral homotopy

$$H(x,t) = \begin{pmatrix} h_1(x,t) := \sum\limits_{\alpha \in S_1'} ((1-t)\bar{c}_{1,\alpha} + tc_{1,\alpha}) x^{\alpha} t^{\langle \hat{\alpha}, \hat{\beta} \rangle - d_1} \\ \vdots \\ h_i(x,t) := \sum\limits_{\alpha \in S_i'} ((1-t)\bar{c}_{i,\alpha} + tc_{i,\alpha}) x^{\alpha} t^{\langle \hat{\alpha}, \hat{\beta} \rangle - d_i} \\ \vdots \\ h_n(x,t) := \sum\limits_{\alpha \in S_n'} ((1-t)\bar{c}_{n,\alpha} + tc_{n,\alpha}) x^{\alpha} t^{\langle \hat{\alpha}, \hat{\beta} \rangle - d_n} \end{pmatrix} = 0, \quad t \in [0,1], \tag{4.3}$$

where each $\bar{c}_{i,\alpha}$ is a randomly chosen nonzero complex number for each $i = 1, \ldots, n$ and $\alpha \in S_i'$. Obviously when $t = 1$, $H(x,1) = P(x)$. When $t = 0$, since

$$\begin{aligned} h_i(x,t) &= \sum_{\alpha \in S_i'} ((1-t)\bar{c}_{i,\alpha} + tc_{i,\alpha}) x^{\alpha} t^{\langle \hat{\alpha}, \hat{\beta} \rangle - d_i} \\ &= \sum_{\substack{\alpha \in S_i' \\ \langle \hat{\alpha}, \hat{\beta} \rangle = d_i}} ((1-t)\bar{c}_{i,\alpha} + tc_{i,\alpha}) x^{\alpha} t^{\langle \hat{\alpha}, \hat{\beta} \rangle - d_i} + \sum_{\substack{\alpha \in S_i' \\ \langle \hat{\alpha}, \hat{\beta} \rangle > d_i}} ((1-t)\bar{c}_{i,\alpha} + tc_{i,\alpha}) x^{\alpha} t^{\langle \hat{\alpha}, \hat{\beta} \rangle - d_i} \\ &= ((1-t)\bar{c}_{i,\alpha_{i1}} + tc_{i,\alpha_{i1}}) x^{\alpha_{i1}} + ((1-t)\bar{c}_{i,\alpha_{i2}} + tc_{i,\alpha_{i2}}) x^{\alpha_{i2}} \\ &\quad + \sum_{\substack{\alpha \in S_i' \\ \langle \hat{\alpha}, \hat{\beta} \rangle > d_i}} ((1-t)\bar{c}_{i,\alpha} + tc_{i,\alpha}) x^{\alpha} t^{\langle \hat{\alpha}, \hat{\beta} \rangle - d_i}, \end{aligned}$$

we have

$$h_i(x,0) = \bar{c}_{i,\alpha_{i1}} x^{\alpha_{i1}} + \bar{c}_{i,\alpha_{i2}} x^{\alpha_{i2}}$$

for each $i = 1, \ldots, n$. Hence

$$H(x,0) = \begin{pmatrix} \bar{c}_{1,\alpha_{11}} x^{\alpha_{11}} + \bar{c}_{1,\alpha_{12}} x^{\alpha_{12}} \\ \vdots \\ \bar{c}_{i,\alpha_{i1}} x^{\alpha_{i1}} + \bar{c}_{i,\alpha_{i2}} x^{\alpha_{i2}} \\ \vdots \\ \bar{c}_{n,\alpha_{n1}} x^{\alpha_{n1}} + \bar{c}_{n,\alpha_{n2}} x^{\alpha_{n2}} \end{pmatrix} = 0.$$

This is a binomial system having

$$k_\beta := \det \left| \begin{pmatrix} \alpha_{11} - \alpha_{12} \\ \vdots \\ \alpha_{i1} - \alpha_{i2} \\ \vdots \\ \alpha_{n1} - \alpha_{n2} \end{pmatrix} \right|$$

nonsingular solutions in $(\mathbb{C}^*)^n$. A very efficient and stable numerical method exists for finding all those solutions [27]. Starting from these solutions to track solution paths of (4.3), we will reach some solutions of $P(x) = 0$. Different mixed cells with their corresponding inner normals β will devote different polyhedral homotopies $H(x,t)$ as defined in (4.3). These different homotopies will reach different isolated zeros of $P(x) = 0$ as proved in [27]. And the total number of paths needed to be tracked here is

$$\sum_{\beta} k_{\beta} = \sum_{\beta} \det \left| \begin{pmatrix} \alpha_{11} - \alpha_{12} \\ \vdots \\ \alpha_{i1} - \alpha_{i2} \\ \vdots \\ \alpha_{n1} - \alpha_{n2} \end{pmatrix} \right|,$$

which agrees with the mixed volume $\mathrm{MV}_n(S'_1, \ldots, S'_n)$.

Although the polyhedral homotopy continuation method usually follows fewer paths than the classical linear homotopy method, the mixed cell computation involved may become very expensive especially for large polynomial systems. Thus if a linear homotopy can be constructed so that only mixed volume number of paths need to be traced, the system should be solved by using a linear homotopy instead of the polyhedral homotopy.

4.2 Construction of a linear homotopy to solve tensor eigenpairs

To solve (3.1), one can certainly use the polyhedral homotopy continuation method implemented in HOM4PS [26], PHCpack [42], PHoM [16], PSOLVE [47] (which is a MATLAB implementation of HOM4PS), or the total degree homotopy continuation method implemented in Bertini [3]. However, using these methods to solve (3.1) directly does not take advantage of the special structures of a tensor eigenproblem. We will introduce two homotopy type algorithms here that utilize such structures.

Theorem 3.3.1 asserts that the mixed volume of (3.1) is $n(m-1)^{n-1}$ for $m=m'$ and $((m-1)^n-(m'-1)^n)/(m-m')$ for $m\neq m'$, which is far less than the Bézout's number, m^n for $m=m'$, and $\max\{(m-1)^n,(m')^n\}$ for $m\neq m'$. From this standpoint, we construct a linear homotopy with which only mixed volume number of paths need to be traced.

For this linear homotopy, the knowledge of multihomogeneous Bézout's number [41] is required. For a polynomial system $P(x)=(p_1(x),\dots,p_n(x))^T$ in (3.2), where $x=(x_1,\dots,x_n)$, we partition the variables $x_1,\dots,x_n$ into k groups $y_1=(x_1^{(1)},\dots,x_{l_1}^{(1)}), y_2=(x_1^{(2)},\dots,x_{l_2}^{(2)}),\dots,y_k=(x_1^{(k)},\dots,x_{l_k}^{(k)})$ with $l_1+\cdots+l_k=n$. Let d_{ij} be the degree of p_i with respect to y_j for $i=1,\dots,n$ and $j=1,\dots,k$. Then the multihomogeneous Bézout's number of $P(x)$ with respect to $(y_1,\dots,y_k)$ is the coefficient of $\alpha_1^{l_1}\alpha_2^{l_2}\dots\alpha_k^{l_k}$ in the product

$$\prod_{i=1}^{n}(d_{i1}\alpha_1+\cdots+d_{ik}\alpha_k).$$

The following theorem will play a very important role in constructing our linear homotopy.

THEOREM 4.2.1 [41] *Let $Q(x)$ be a system of polynomials having the same multihomogeneous structure as $P(x)$ with respect to certain partition of the variables $(x_1,\dots,x_n)$. Assume $Q(x)=0$ has exactly the multihomogeneous Bézout's number of nonsingular solutions with respect to this partition, and let*

$$H(x,t)=(1-t)\gamma Q(x)+tP(x)=0,$$

where $t\in[0,1]$ and $\gamma\in\mathbb{C}^*$. For $\gamma=re^{i\theta}$, Properties 1 and 2 hold for all but finitely many θ.

For (3.1), when $m = m'$ the following polynomial system

$$G(\lambda, x) = \begin{pmatrix} (\mathcal{A}^{(k)}x^{m-1})_1 - \lambda(\mathcal{B}x^{m-1})_1 \\ \vdots \\ (\mathcal{A}^{(k)}x^{m-1})_n - \lambda(\mathcal{B}x^{m-1})_n \\ a_1x_1 + a_2x_2 + \cdots + a_nx_n + b \end{pmatrix} = 0 \tag{4.4}$$

needs to be solved, where λ and $x := (x_1, \cdots, x_n)^T$ are the unknowns, $a_1, \ldots, a_n, b$ are randomly chosen complex numbers. Consider the system

$$Q(\lambda, x) = \begin{pmatrix} (\lambda - \mu_1)(x_1^{m-1} - \beta_1) \\ (\lambda - \mu_2)(x_2^{m-1} - \beta_2) \\ \vdots \\ (\lambda - \mu_n)(x_n^{m-1} - \beta_n) \\ c_1x_1 + \ldots c_nx_n + d \end{pmatrix} = 0, \tag{4.5}$$

where d as well as μ_i, β_i, c_i for $i = 1, \ldots, n$ are generic nonzero complex numbers.

THEOREM 4.2.2 *For $G(\lambda, x)$ and $Q(\lambda, x)$ given above, all isolated zeros (λ, x) of $G(\lambda, x)$ in $\mathbb{C}^{n+1}$ can be found by the homotopy*

$$H(\lambda, x, t) = (1 - t)\gamma Q(\lambda, x) + tG(\lambda, x) = 0, \quad t \in [0, 1] \tag{4.6}$$

for almost all $\gamma \in \mathbb{C}^*$.

Proof: Evidently, with respect to the partition (λ) and $(x_1, x_2, \ldots, x_n)$ of the variables $(\lambda, x_1, x_2, \ldots, x_n)$, both systems (4.4) and (4.5) have degree 1 in (λ) and degree $m - 1$ in

$(x_1, x_2, \ldots, x_n)$ for the first n equations, and degree 0 in (λ) and degree 1 in $(x_1, x_2, \ldots, x_n)$ for the last equation. Hence (4.4) and (4.5) have the same multihomogeneous Bézout's number, that is the coefficient of $\alpha_1 \alpha_2^n$ in the product

$$[1 \cdot \alpha_1 + (m-1)\alpha_2]^n (0 \cdot \alpha_1 + 1 \cdot \alpha_2),$$

which is

$$\binom{n}{1} (m-1)^{n-1} = n(m-1)^{n-1}.$$

We now show $Q(\lambda, x)$ in (4.5) has exactly $n(m-1)^{n-1}$ zeros. In the first place, if λ is equal to none of $\mu_1, \ldots, \mu_n$, then (4.5) becomes a system of $n+1$ equations and n unknowns, which is overdetermined with generic coefficients. It therefore has no solutions. Thus λ must be equal to one of $\mu_1, \ldots, \mu_n$. Assume $\lambda = \mu_1$, then $x_1, \ldots, x_n$ can be determined by

$$\begin{aligned} x_2^{m-1} - \beta_2 &= 0 \\ &\vdots \\ x_n^{m-1} - \beta_n &= 0 \\ c_1 x_1 + \ldots c_n x_n + d &= 0 \end{aligned}$$

Obviously, each x_i for $i = 2, \ldots, n$ can be chosen as one of the $(m-1)$-th root of β_i and x_1 can be solved by substituting the chosen $x_2, \ldots, x_n$ into the last equation. So there are $(m-1)^{n-1}$ solutions corresponding to $\lambda = \mu_1$. This argument holds for λ being any of the μ_i's. Therefore, there are $n(m-1)^{n-1}$ solutions in total.

It remains to prove that each solution of $Q(\lambda, x) = 0$ in (4.5) is nonsingular. As discussed above, any solution (λ^*, x^*) of (4.5) satisfies

$$\begin{aligned}
\lambda^* &= \mu_i \\
(x_1^*)^{m-1} - \beta_1 &= 0 \\
&\vdots \\
(x_{i-1}^*)^{m-1} - \beta_{i-1} &= 0 \\
(x_{i+1}^*)^{m-1} - \beta_{i+1} &= 0 \\
&\vdots \\
(x_n^*)^{m-1} - \beta_n &= 0 \\
c_1 x_1^* + \cdots + c_n x_n^* + d &= 0
\end{aligned} \tag{4.7}$$

Let $DQ(\lambda, x)$ be the Jacobian of $Q(\lambda, x)$ at (λ, x). For nonsingularity of $DQ(\lambda^*, x^*)$, let

$$A_j(\lambda, x) := x_j^{m-1} - \beta_j, \quad B_j(\lambda, x) := (\lambda - \mu_j)(m-1)x_j^{m-2}$$

for $j = 1, \ldots, n$. Then

$$DQ(\lambda, x) = \begin{pmatrix} A_1 & B_1 & & & & & & \\ \vdots & & \ddots & & & & & \\ A_{i-1} & & & B_{i-1} & & & & \\ A_i & & & & B_i & & & \\ A_{i+1} & & & & & B_{i+1} & & \\ \vdots & & & & & & \ddots & \\ A_n & & & & & & & B_n \\ 0 & c_1 & \ldots & c_{i-1} & c_i & c_{i+1} & \ldots & c_n \end{pmatrix}.$$

With

$$A_j(\lambda^*, x^*) = (x_j^*)^{m-1} - \beta_j = 0, \quad j \neq i$$

and

$$B_i(\lambda^*, x^*) = (\lambda^* - \mu_i)(m-1)(x_i^*)^{m-2} = (\mu_i - \mu_i)(m-1)(x_i^*)^{m-2} = 0,$$

we have, by (4.7),

$$DQ(\lambda^*, x^*) = \begin{pmatrix} 0 & B_1^* & & & & & & \\ \vdots & & \ddots & & & & & \\ 0 & & & B_{i-1}^* & & & & \\ A_i^* & & & & 0 & & & \\ 0 & & & & & B_{i+1}^* & & \\ \vdots & & & & & & \ddots & \\ 0 & & & & & & & B_n^* \\ 0 & c_1 & \dots & c_{i-1} & c_i & c_{i+1} & \dots & c_n \end{pmatrix},$$

where $A_j^* := A_j(\lambda^*, x^*)$ and $B_j^* := B_j(\lambda^*, x^*)$. It follows that

$$\det(DQ(\lambda^*, x^*)) = (-1)^{i+1} A_i^* (-1)^{n+i} c_i \prod_{j \neq i} B_j^* \neq 0$$

by (4.7). $\square$

4.3 Numerical algorithms of using homotopy continuation methods to solve tensor eigenpairs

Theorem 4.2.2 suggests that (4.6) can be used to solve (3.1) when $m = m'$. Write $u := (\lambda, x)$, then (4.6) becomes

$$H(u, t) = (1 - t)\gamma Q(u) + tG(u) = 0, \quad t \in [0, 1] \tag{4.8}$$

where Q and G are defined in (4.5) and (4.4) respectively.

We now present our algorithm for computing mode-k generalized eigenpairs when $m = m'$, i.e., solving (4.4).

ALGORITHM 4.3.1 (Compute complex mode-k $\mathcal{B}$-eigenpairs of $\mathcal{A}$, where $\mathcal{A}, \mathcal{B} \in \mathbb{C}^{[m,n]}$, i.e., solving (4.4).)

Step 1. *Compute all solutions of $Q(u)$ as defined in (4.5).*

Step 2. *Path following: Follow the paths from $t = 0$ to $t = 1$ using the prediction-correction strategy. Let $(u_k, t_k) := (u(t_k), t_k)$. For finding the next point (u_{k+1}, t_{k+1}) on the solution path of*

$$H(u, t) = (1 - t)\gamma Q(u) + tG(u) = 0, \quad t \in [0, 1]$$

as defined in (4.8), the following steps are employed:

- *Prediction Step: Compute the tangent vector $\dfrac{du}{dt}$ to $H(u, t) = 0$ at t_k by solving the linear system*

$$H_u(u_k, t_k)\frac{du}{dt} = -H_t(u_k, t_k)$$

for $\dfrac{du}{dt}$. Then compute the approximation $\tilde{u}$ to u_{k+1} by

$$\tilde{u} = u_k + \Delta t\frac{du}{dt}, \quad t_{k+1} = t_k + \Delta t,$$

where Δt is the stepsize. Here u_0 is chosen to be one solution of $Q(u) = 0$.

- *Correction Step: Use Newton's iterations, i.e., for $i = 0, 1, 2, \ldots,$ compute*

$$v_{i+1} = v_i - [H_u(v_i, t_{k+1})]^{-1}H(v_i, t_{k+1}) \quad \textit{with} \quad v_0 = \tilde{u}$$

until $\|H(v_J, t_J)\|$ is very small. Then let $u_{k+1} = v_J$.

Step 3. *End game. When t_N is very close to 1, the corresponding u_N should be very close to a zero u^* of $G(u) = G(\lambda, x)$. So Newton's iterations*

$$u^{(k+1)} = u^{(k)} - [DG(u^{(k)})]^{-1} G(u^{(k)}), \quad k = 0, 1, \ldots$$

will be used again to refine our final approximation $\tilde{u}$ to u^*. If $DG(u^*)$ is nonsingular, then $\tilde{u}$ will be a very good approximation of u^* with multiplicity 1. If $DG(u^*)$ is singular, $\tilde{u}$ is either an isolated singular zero of $G(u)$ with multiplicity $l > 1$ or in a positive dimensional solution component of $G(u) = 0$. We use a strategy provided in Chapter VIII of [27] (see also [41]) to determine whether $\tilde{u}$ is an isolated zero with multiplicity $l > 1$ or in a positive dimensional solution component of $G(u) = 0$.

Step 4. *For each solution $u = (\lambda, x)$ obtained in Step 3, normalize x with $i_0 := \arg\max_{1 \leq i \leq n} |x_i|$:*

$$y = \frac{x}{x_{i_0}} \tag{4.9}$$

a new eigenpair (λ, y) is obtained, since as mentioned in (2.4) and Remark 2.1.1, if (λ, x) is an eigenpair, (λ, tx) for $t \neq 0$ is also an eigenpair.

REMARK 4.3.1 Notice that if x is a real eigenvector associated with a real eigenvalue λ, tx for any $t \in \mathbb{C}\backslash\{0\}$ will be a complex eigenvector associated with the same eigenvalue λ. If at any stage a complex eigenvector like tx is obtained in Step 3 of Algorithm 4.3.1, applying (4.9) to tx will give a new real eigenvector. In this regard, Step 4 is very helpful for detecting real eigenpairs.

To compute mode-k generalized tensor eigenpairs when $m \neq m'$, we use the equivalence class structure of the eigenproblem as described in Remark 2.1.1. We first solve (3.1) by PSOLVE [47] to find a representative (λ, x) from each equivalence class and then find all m' eigenpairs from each equivalence class by simply using $\lambda' = t^{m-m'}\lambda, x' = tx$, where t is a root of $t^{m'} = 1$.

One may solve (2.3) directly for m' eigenpairs from each equivalence class. However, this alternative must follow m' times more paths than the above approach costing more computations.

We now present our algorithm for computing mode-k generalized eigenpairs when $m \neq m'$, i.e., solving (2.3).

ALGORITHM 4.3.2 (Compute complex mode-k $\mathcal{B}$-eigenpairs of $\mathcal{A}$, where $\mathcal{A} \in \mathbb{C}^{[m,n]}, \mathcal{B} \in \mathbb{C}^{[m',n]}$ with $m \neq m'$, i.e., solving (2.3).)

Step 1. *Using PSOLVE to get all solutions* (λ, x) *of (3.1).*

Step 2. *For each* (λ, x) *obtained in Step 1, if* $\mathcal{B}x^{m'} \neq 0$*, normalize it to get an eigenpair* (λ^*, x^*) *by*

$$\lambda^* = \frac{\lambda}{(\mathcal{B}x^{m'})^{(m-m')/m'}}, \quad x^* = \frac{x}{(\mathcal{B}x^{m'})^{1/m'}}$$

to satisfy (2.3).

Step 3. *Compute* m' *equivalent eigenpairs* (λ', x') *of* (λ^*, x^*) *by* $\lambda' = t^{m-m'}\lambda^*$ *and* $x' = tx^*$ *with* t *being a root of* $t^{m'} = 1$.

Chapter 5

Computing real tensor eigenpairs by homotopy methods

In some applications, tensor $\mathcal{A}$ is real and only real eigenpairs (or real eigenvalues) of $\mathcal{A}$ are of interest ([12, 35]). In this situation, only real zeros of the polynomial system (2.3) or (4.4) are needed. Currently there is no effective method to find all real zeros of a polynomial system directly.

5.1 Using Newton homotopy method to find real tensor eigenpairs

For a real tensor $\mathcal{A}$, a real eigenvalue may have complex eigenvectors. To identify real eigenvalues, we may first compute complex zeros (λ, x) of (4.4) by Algorithm 4.3.1 or (2.3) by Algorithm 4.3.2, then classify the real eigenvalues by checking the size of their imaginary parts. Specifically, let (λ^*, x^*) be a computed eigenpair. If

$$|\operatorname{Im}(\lambda^*)| < \delta_0,$$

with threshold $\delta_0 > 0$, then $\operatorname{Re}(\lambda^*)$ will be taken as a real eigenvalue.

For a specific occasion, when $m \neq m'$, if $\frac{m'}{m-m'}$ is a nonzero integer multiple of 4 (for example, $m = 5, m' = 4$ or $m = 10, m' = 8$,) and $\mathcal{A}$ has an eigenpair (λ^*, x^*) with a purely imaginary eigenvalue $\lambda^* = bi$, where $b \in \mathbb{R}$, then one can easily show that $(b, (-i)^{1/(m-m')}x^*)$ and $(-b, i^{1/(m-m')}x^*)$ are eigenpairs with real eigenvalues. Therefore, when $\frac{m'}{m-m'}$ is a nonzero integer multiple of 4, if (λ^*, x^*) is an eigenpair found by Algorithm 4.3.2 such that

$$|\operatorname{Re}(\lambda^*)| < \delta_0,$$

then we take $\operatorname{Im}(\lambda^*)$ and $-\operatorname{Im}(\lambda^*)$ as real eigenvalues, with corresponding eigenvectors $(-i)^{1/(m-m')}x^*$ and $i^{1/(m-m')}x^*$.

When look for real tensor eigenpairs (i.e., both eigenvalues and eigenvectors are real), the situation becomes more complicated. We propose a two-step procedure. First, compute complex zeros (λ, x) of (4.4) by Algorithm 4.3.1 or (2.3) by Algorithm 4.3.2. Then extract all real eigenpairs (λ, x) from the zeros just calculated.

For vector $a = (a_1, \ldots, a_n)^T \in \mathbb{C}^n$, let

$$\operatorname{Im}(a) = (\operatorname{Im}(a_1), \ldots, \operatorname{Im}(a_n))^T, \quad \operatorname{Re}(a) = (\operatorname{Re}(a_1), \ldots, \operatorname{Re}(a_n))^T.$$

Suppose (λ^*, x^*) is an eigenpair found in the first step. There are possibly two cases: (i) (λ^*, x^*) is an isolated eigenpair; (ii) (λ^*, x^*) is an eigenpair contained in a positive dimensional solution component of system (4.4) or (2.3).

When (λ^*, x^*) is an isolated eigenpair, take $(\operatorname{Re}(\lambda^*), \operatorname{Re}(x^*))$ as a real eigenpair if

$$\|\operatorname{Im}(\lambda^*, x^*)\|_2 < \delta_0.$$

If (λ^*, x^*) is an eigenpair in a positive dimensional solution component of system (4.4) or (2.3), in general real eigenvectors induced by which are not warranted even if the corresponding eigenvalue λ^* is real. In this case, we use the following Newton homotopy [1]

$$H(\lambda, x, t) := P(\lambda, x) - (1-t)P(\lambda^*, \mathrm{Re}(x^*)), \quad t \in [0,1] \tag{5.1}$$

to follow homotopy curves of $H(\lambda, x, t) = 0$ in $(\lambda, x) \in \mathbb{R}^{n+1}$ for a real eigenpair. Notice that when following curves in the complex space it is proved in [27] that the solution curves of (5.1) can be parameterized by t, but the solution curves of (5.1) may not be a function of t when restricted in the real space. So a different method to follow curves is needed. In this case parameterizing the solution curves by the arc length s is suggested in [27]. For simplicity, write $y(s) := (\lambda(s), x(s), t(s))$, then (5.1) becomes $H(y(s)) = 0$.

We now summarize our algorithm for computing a real eigenpair from a complex eigenpair (λ^*, x^*) with real λ^*, which is in a positive dimensional solution component of (4.4) or (2.3).

ALGORITHM 5.1.1 (Trace solution paths of (5.1) in the real space to get a real eigenpair.)

Step 1. *Let $y_k := y(s_k)$ and let $y_0 = (\lambda^*, \mathrm{Re}(x^*))$, to find the next point on the solution path of $H(y) = 0$, we use the following strategy:*

- *Prediction Step: Compute the tangent vector $\frac{dy}{ds}$ to $H(y) = 0$ at y_k by solving the system*

$$\begin{aligned} DH(y_k)\frac{dy}{ds} &= 0 \\ \left\|\frac{dy}{ds}\right\|_2 &= 1 \end{aligned}$$

for $\frac{dy}{ds}$. Here $DH(y_k)$ is the Jacobian of H with respect to y evaluated at y_k. Then *compute the approximation $\tilde{y}$ to y_{k+1} by*

$$\tilde{y} = y_k + \Delta s \frac{dy}{ds}, \qquad s_{k+1} = s_k + \Delta s,$$

where Δs is a stepsize.

- *Correction Step: Use Newton's iterations to solve ξ from*

$$F(\xi) := \begin{pmatrix} H(\xi) \\ (\xi - \tilde{y}) \cdot \dfrac{dy}{ds} \end{pmatrix} = 0$$

with the initial point $\xi_0 = \tilde{y}$, i.e., for $i = 0, 1, 2, \ldots,$ compute

$$\xi_{i+1} = \xi_i - [DF(\xi_i)]^{-1} F(\xi_i)$$

until $\|F(\xi_J)\|$ is less than a threshold for some J. Then let $y_{k+1} = \xi_J$.

Step 2. *End game. During Step 1 if after sufficiently many Prediction-Correction Steps t does not approach 1, then the scheme cannot provide a real eigenpair, stop. Otherwise when $t(s_N)$ is very close to 1 for some N, the corresponding $(\lambda(s_N), x(s_N))$ is very close to a real zero $u_R := (\lambda_R, x_R)$ of $P(u) := P(\lambda, x)$. So Newton's iterations*

$$u^{(k+1)} = u^{(k)} - [DP(u^{(k)})]^{-1} P(u^{(k)}), \quad k = 0, 1, \ldots$$

with $u^{(0)} = (\lambda(s_N), x(s_N))$ will be employed to refine our final approximation $\tilde{u}$ to u_R. Take $\tilde{u}$ as a real eigenpair.

5.2 A straightforward approach to find real tensor eigenpairs

An interesting phenomenon is that in some special cases, the real eigenpairs can be obtained in a more straightforward manner from the complex eigenpairs found by Algorithm 4.3.1 or Algorithm 4.3.2, as illustrated in the following example.

EXAMPLE 5.2.1 Let the tensor $\mathcal{A} \in \mathbb{R}^{[3,5]}$ (Example 4.11 in [12], see also [34]) be given by

$$\mathcal{A}_{i,j,k} = \frac{(-1)^i}{i} + \frac{(-1)^j}{j} + \frac{(-1)^k}{k}, \quad i,j,k = 1,\ldots,5.$$

We want the Z-eigenpairs of $\mathcal{A}$. The corresponding polynomial system to solve is

$$\begin{aligned}
-3x_1^2 - 3x_1x_2 - \frac{25}{6}x_1x_3 - \frac{7}{2}x_1x_4 - \frac{22}{5}x_1x_5 - \frac{7}{6}x_2x_3 - \frac{1}{2}x_2x_4 - \frac{7}{5}x_2x_5 & \\
-\frac{7}{6}x_3^2 - \frac{13}{6}x_3x_4 - \frac{181}{60}x_3x_5 - \frac{1}{2}x_4^2 - \frac{19}{10}x_4x_5 - \frac{7}{5}x_5^2 - \lambda x_1 = & \ 0 \\
-\frac{3}{2}x_1^2 - \frac{7}{6}x_1x_3 - \frac{1}{2}x_1x_4 - \frac{7}{5}x_1x_5 + \frac{3}{2}x_2^2 + \frac{4}{3}x_2x_3 + \frac{5}{2}x_2x_4 & \\
+\frac{8}{5}x_2x_5 - \frac{1}{6}x_3^2 + \frac{5}{6}x_3x_4 - \frac{1}{15}x_3x_5 + x_4^2 + \frac{11}{10}x_4x_5 + \frac{1}{10}x_5^2 - \lambda x_2 = & \ 0 \\
-\frac{7}{3}x_1^2 - \frac{5}{3}x_1x_2 - \frac{10}{3}x_1x_3 - \frac{13}{6}x_1x_4 - \frac{46}{15}x_1x_5 + \frac{2}{3}x_2^2 - \frac{1}{3}x_2x_3 + \frac{5}{6}x_2x_4 & \\
-\frac{1}{15}x_2x_5 - x_3^2 - \frac{5}{6}x_3x_4 - \frac{26}{15}x_3x_5 + \frac{1}{6}x_4^2 - \frac{17}{30}x_4x_5 - \frac{11}{15}x_5^2 - \lambda x_3 = & \ 0 \\
-\frac{7}{4}x_1^2 - \frac{1}{2}x_1x_2 - \frac{13}{6}x_1x_3 - x_1x_4 - \frac{19}{10}x_1x_5 + \frac{5}{4}x_2^2 + \frac{5}{6}x_2x_3 + 2x_2x_4 & \\
+x_2x_5 - \frac{5}{12}x_3^2 + \frac{1}{3}x_3x_4 - \frac{17}{30}x_3x_5 + \frac{3}{4}x_4^2 + \frac{3}{5}x_4x_5 - \frac{3}{20}x_5^2 - \lambda x_4 = & \ 0 \\
-\frac{11}{5}x_1^2 - \frac{7}{5}x_1x_2 - \frac{46}{15}x_1x_3 - \frac{19}{10}x_1x_4 - \frac{14}{5}x_1x_5 + \frac{4}{5}x_2^2 - \frac{1}{15}x_2x_3 + \frac{11}{10}x_2x_4 & \\
+\frac{1}{5}x_2x_5 - \frac{13}{15}x_3^2 - \frac{17}{30}x_3x_4 - \frac{22}{15}x_3x_5 + \frac{3}{10}x_4^2 - \frac{3}{10}x_4x_5 - \frac{3}{5}x_5^2 - \lambda x_5 = & \ 0 \\
x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 1 = & \ 0
\end{aligned}$$

Write the above polynomial system as $P(\lambda, x) = 0$. Using Algorithm 4.3.2, 62 solutions of this system are found. Among them, 4 are isolated solutions with $\|(\mathrm{Im}(\lambda), \mathrm{Im}(x)\|$ as small as the machine epsilon. Therefore, these isolated zeros can be classified as the Z-eigenpairs. Note that in this example if (λ, x) is an eigenpair, then so is $(-\lambda, -x)$. Table 5.1 lists two of these four Z-eigenpairs with positive λ.

λ	4.2876	9.9779
x_1	−0.1859	−0.7313
x_2	0.7158	−0.1375
x_3	0.2149	−0.4674
x_4	0.5655	−0.2365
x_5	0.2950	−0.4146

Table 5.1: Isolated Z-eigenpairs of the tensor in Example 5.2.1

For the remaining 58 solutions of $P(\lambda, x) = 0$, each of them has $\lambda = 0$ and is contained in a positive dimensional solution component of $P(\lambda, x) = 0$. For each of these zeros, it can be verified that $(\lambda, \mathrm{Re}(x)/\|\mathrm{Re}(x)\|_2)$ and $(\lambda, \mathrm{Im}(x)/\|\mathrm{Im}(x)\|_2)$ are both solutions of $P(\lambda, x) = 0$. For example, one of these 58 zeros is

$$\begin{pmatrix} \lambda \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ -0.7136 - 0.4086i \\ -0.1425 - 0.0266i \\ 0.9941 - 0.2291i \\ -0.0180 + 0.1180i \\ -0.1200 + 0.5464i \end{pmatrix}.$$

Taking the real and imaginary parts of the above x and let

$$\begin{aligned}\xi &= (-0.7136, -0.1425, 0.9941, -0.0180, -0.1200)^T,\\ \eta &= (-0.4086, -0.0266, -0.2291, 0.1180, 0.5464)^T.\end{aligned}$$

Normalizing ξ and η respectively gives

$$\begin{aligned}v &= (-0.5764, -0.1151, 0.8030, -0.0146, -0.0969)^T,\\ w &= (-0.5599, -0.0365, -0.3139, 0.1617, 0.7487)^T.\end{aligned}$$

Simple computation shows $\|P(0,v)\|_2 \approx 5.8687 \times 10^{-14}$ and $\|P(0,w)\|_2 \approx 5.4921 \times 10^{-14}$. Therefore, $(0,v)$ and $(0,w)$ are both Z-eigenpairs. Accordingly $(\lambda, \mathrm{Re}(x)/\|\mathrm{Re}(x)\|_2)$ and $(\lambda, \mathrm{Im}(x)/\|\mathrm{Im}(x)\|_2)$ are two Z-eigenpairs.

The above example suggests that if (λ^*, x^*) is in a positive dimensional solution component of (2.3) and $\lambda^* \in \mathbb{R}$, then $(\lambda^*, \mathrm{Re}(x^*)/\|\mathrm{Re}(x^*)\|_2)$ and $(\lambda^*, \mathrm{Im}(x^*)/\|\mathrm{Im}(x^*)\|_2)$ may be mode-k $\mathcal{B}_R$ eigenpairs of $\mathcal{A}$. This provides a straightforward approach to find real eigenpairs from eigenpairs belong to positive dimensional components. Actually, this approach has been quite successful for all the examples (e.g., Example 4.8, 4.11, 4.13, 4.14) in [12] when a real Z-eigenvalue has infinitely many real Z-eigenvectors.

This leads to the question: what is the structure of the eigenspace $\{x \in \mathbb{C}^n \mid \mathcal{A}x^{m-1} = \lambda^* \mathcal{B}x^{m'-1}, \mathcal{B}x^{m'} = 1\}$ associated with a real eigenvalue λ^*? Looking into the eigenspace of $\lambda^* = 0$ in Example 5.2.1, i.e., $\{x \in \mathbb{C}^5 \mid P(0,x) = 0\}$, all the 58 eigenvectors obtained from Algorithm 4.3.2 lay on the hyperplane $x_1 + x_2 + x_3 + x_4 + x_5 = 0$. This observation stimulates the following proposition.

PROPOSITION 5.2.1 *Let $\mathcal{A} \in \mathbb{R}^{[m,n]}$ and $\mathcal{B} \in \mathbb{R}^{[m',n]}$. Let k be an integer with $1 \le k \le m$, and $\lambda \in \mathbb{R}$ be a real mode-k $\mathcal{B}$ eigenvalue of $\mathcal{A}$. If $V := \{x \in \mathbb{C}^n \mid \mathcal{A}^{(k)} x^{m-1} = \lambda \mathcal{B} x^{m'-1}\}$ is a complex linear subspace of $\mathbb{C}^n$, then for any $x = \xi + i\eta \in V$ such that $\xi, \eta \in \mathbb{R}^n$ and $\xi \neq 0, \eta \neq 0$,*

- *When $m = m'$, ξ and η are both real mode-k $\mathcal{B}$ eigenvectors of $\mathcal{A}$ associated with λ.*
- *When $m \neq m'$, $\mathcal{B}\xi^{m'} \neq 0$ and $\mathcal{B}\eta^{m'} \neq 0$, the normalized vectors*

$$v := \frac{\xi}{(B\xi^{m'})^{1/m'}}, \quad w := \frac{\eta}{(B\eta^{m'})^{1/m'}}$$

are real mode-k $\mathcal{B}$ eigenvectors of $\mathcal{A}$ associated with λ.

Proof: Let $x \in V$. Then

$$\mathcal{A}^{(k)} x^{m-1} = \lambda \mathcal{B} x^{m'-1}.$$

Taking the conjugate of the above equation yields

$$\bar{\mathcal{A}}^{(k)} \bar{x}^{m-1} = \bar{\lambda} \bar{\mathcal{B}} \bar{x}^{m'-1}.$$

Since λ, $\mathcal{A}$ and $\mathcal{B}$ are all real,

$$\mathcal{A}^{(k)} \bar{x}^{m-1} = \lambda \mathcal{B} \bar{x}^{m'-1}.$$

It follows that $\bar{x} \in V$. Since V is a linear subspace, $\xi = (x + \bar{x})/2$ and $\eta = (x - \bar{x})/(2i)$ are also in V. Thus, when $m = m'$, ξ and η are both real mode-k $\mathcal{B}$ eigenvectors of $\mathcal{A}$ associated with λ. If $m \neq m'$, we have

$$\begin{aligned}
\mathcal{B}v^{m'} &= \sum_{i_1,\cdots,i_{m'}=1}^{n} B_{i_1 i_2 \cdots i_{m'}} v_{i_1} v_{i_2} \cdots v_{i_{m'}} \\
&= \sum_{i_1,\cdots,i_{m'}=1}^{n} B_{i_1 i_2 \cdots i_{m'}} \frac{\xi_{i_1}}{(B\xi^{m'})^{1/m'}} \frac{\xi_{i_2}}{(B\xi^{m'})^{1/m'}} \cdots \frac{\xi_{i_{m'}}}{(B\xi^{m'})^{1/m'}} \\
&= \frac{\sum_{i_1,\cdots,i_{m'}=1}^{n} B_{i_1 i_2 \cdots i_{m'}} \xi_{i_1} \xi_{i_2} \cdots \xi_{i_{m'}}}{B\xi^{m'}} \\
&= \frac{B\xi^{m'}}{B\xi^{m'}} = 1.
\end{aligned}$$

So v is a real mode-k $\mathcal{B}$ eigenvector of $\mathcal{A}$ associated with λ. Similarly, $Bw^{m'} = 1$ and w is also a real mode-k $\mathcal{B}$ eigenvector of $\mathcal{A}$ associated with λ. $\square$

Consequently, when Z-eigenpairs of $\mathcal{A} \in \mathbb{R}^{[m,n]}$ are in concern, let $\lambda \in \mathbb{R}$ be a real E-eigenvalue of $\mathcal{A}$. If $U := \{x \in \mathbb{C}^n \,|\, \mathcal{A}x^{m-1} = \lambda x\}$ contains a complex linear subspace V, then for any $x = \xi + i\eta \in V$ with nonzero $\xi, \eta \in \mathbb{R}^n$ and $\bar{x} \in V$, $\xi/\|\xi\|_2$ and $\eta/\|\eta\|_2$ are both Z-eigenvectors of $\mathcal{A}$ associated with λ.

A natural question is when the eigenspace U defined in Proposition 5.2.1 will contain a linear subspace V. The following propositions suggest some possibilities.

PROPOSITION 5.2.2 *Let $\mathcal{A}$ be a third-order and n-dimensional rank-one tensor, i.e., $\mathcal{A}$ can be written as the outer product of 3 vectors $a, b, c \in \mathbb{C}^n$:*

$$\mathcal{A} := a \circ b \circ c \quad \text{with} \quad \mathcal{A}_{ijk} := (a_i b_j c_k), \quad i, j, k = 1, \ldots, n.$$

Let $W := \operatorname{span}(a, b, c)$ and $V := W^{\perp}$. Then $U := \{x \in \mathbb{C}^n \,|\, \mathcal{A}x^2 = 0\} \supseteq V$, i.e., the *eigenspace of $\mathcal{A}$ corresponding to the eigenvalue 0 contains V.*

Proof: It suffices to show for any $x \in V$, $\mathcal{A}x^2 = 0$. Actually, by definition,

$$\mathcal{A}x^2 = \begin{pmatrix} \sum\limits_{j,k=1}^{n} a_1 b_j c_k x_j x_k \\ \sum\limits_{j,k=1}^{n} a_2 b_j c_k x_j x_k \\ \vdots \\ \sum\limits_{j,k=1}^{n} a_n b_j c_k x_j x_k \end{pmatrix}.$$

Simple computation gives

$$\begin{aligned} \mathcal{A}x^2 &= \begin{pmatrix} a_1 \sum\limits_{j,k=1}^{n} b_j c_k x_j x_k \\ a_2 \sum\limits_{j,k=1}^{n} b_j c_k x_j x_k \\ \vdots \\ a_n \sum\limits_{j,k=1}^{n} b_j c_k x_j x_k \end{pmatrix} = \begin{pmatrix} a_1 \left(\sum\limits_{j=1}^{n} b_j x_j\right)\left(\sum\limits_{k=1}^{n} c_k x_k\right) \\ a_2 \left(\sum\limits_{j=1}^{n} b_j x_j\right)\left(\sum\limits_{k=1}^{n} c_k x_k\right) \\ \vdots \\ a_n \left(\sum\limits_{j=1}^{n} b_j x_j\right)\left(\sum\limits_{k=1}^{n} c_k x_k\right) \end{pmatrix} \\ &= \begin{pmatrix} a_1(b^T x)(c^T x) \\ a_2(b^T x)(c^T x) \\ \vdots \\ a_n(b^T x)(c^T x) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \end{aligned}$$

where the last equality holds since $x \in \operatorname{span}(a, b, c)^{\perp}$. $\square$

PROPOSITION 5.2.3 *Let $\mathcal{A}$ be a third-order and n-dimensional tensor which can be* *decomposed as*

$$\mathcal{A} = \sum_{l=1}^{r} a^{(l)} \circ a^{(l)} \circ a^{(l)},$$

where $r < n$, $a^{(l)} \in \mathbb{C}^n, l = 1, 2, \cdots, r$. *Let* $W = \mathrm{span}(a^{(1)}, a^{(2)}, \cdots, a^{(r)})$ *and let* $V = W^{\perp}$. *Then* $U := \{x \in \mathbb{C}^n \,|\, \mathcal{A}x^2 = 0\} \supseteq V$, *i.e., the eigenspace of* $\mathcal{A}$ *corresponding to the eigenvalue 0 contains* V.

Proof: It is sufficient to show $\mathcal{A}x^2 = 0$ for any $x \in V$. By definition,

$$\mathcal{A}_{ijk} = \sum_{l=1}^{r} \left(a^{(l)} \circ a^{(l)} \circ a^{(l)}\right)_{ijk} = \sum_{l=1}^{r} a_i^{(l)} a_j^{(l)} a_k^{(l)}, \quad i, j, k = 1, \ldots, n.$$

So

$$\mathcal{A}x^2 = \begin{pmatrix} \sum\limits_{j,k=1}^{n} \mathcal{A}_{1jk} x_j x_k \\ \sum\limits_{j,k=1}^{n} \mathcal{A}_{2jk} x_j x_k \\ \vdots \\ \sum\limits_{j,k=1}^{n} \mathcal{A}_{njk} x_j x_k \end{pmatrix} = \begin{pmatrix} \sum\limits_{j,k=1}^{n} \left(\sum\limits_{l=1}^{r} a_1^{(l)} a_j^{(l)} a_k^{(l)} \right) x_j x_k \\ \sum\limits_{j,k=1}^{n} \left(\sum\limits_{l=1}^{r} a_2^{(l)} a_j^{(l)} a_k^{(l)} \right) x_j x_k \\ \vdots \\ \sum\limits_{j,k=1}^{n} \left(\sum\limits_{l=1}^{r} a_n^{(l)} a_j^{(l)} a_k^{(l)} \right) x_j x_k \end{pmatrix}.$$

Then

$$
\begin{aligned}
\mathcal{A}x^2 &= \begin{pmatrix} \sum_{j,k=1}^{n} \sum_{l=1}^{r} a_1^{(l)} a_j^{(l)} a_k^{(l)} x_j x_k \\ \sum_{j,k=1}^{n} \sum_{l=1}^{r} a_2^{(l)} a_j^{(l)} a_k^{(l)} x_j x_k \\ \vdots \\ \sum_{j,k=1}^{n} \sum_{l=1}^{r} a_n^{(l)} a_j^{(l)} a_k^{(l)} x_j x_k \end{pmatrix} = \begin{pmatrix} \sum_{l=1}^{r} \sum_{j,k=1}^{n} a_1^{(l)} a_j^{(l)} a_k^{(l)} x_j x_k \\ \sum_{l=1}^{r} \sum_{j,k=1}^{n} a_2^{(l)} a_j^{(l)} a_k^{(l)} x_j x_k \\ \vdots \\ \sum_{l=1}^{r} \sum_{j,k=1}^{n} a_n^{(l)} a_j^{(l)} a_k^{(l)} x_j x_k \end{pmatrix} \\
&= \begin{pmatrix} \sum_{l=1}^{r} a_1^{(l)} \left(\sum_{j,k=1}^{n} a_j^{(l)} a_k^{(l)} x_j x_k \right) \\ \sum_{l=1}^{r} a_2^{(l)} \left(\sum_{j,k=1}^{n} a_j^{(l)} a_k^{(l)} x_j x_k \right) \\ \vdots \\ \sum_{l=1}^{r} a_n^{(l)} \left(\sum_{j,k=1}^{n} a_j^{(l)} a_k^{(l)} x_j x_k \right) \end{pmatrix} = \begin{pmatrix} \sum_{l=1}^{r} a_1^{(l)} \left(\sum_{j=1}^{n} a_j^{(l)} x_j \right) \left(\sum_{k=1}^{n} a_k^{(l)} x_k \right) \\ \sum_{l=1}^{r} a_2^{(l)} \left(\sum_{j=1}^{n} a_j^{(l)} x_j \right) \left(\sum_{k=1}^{n} a_k^{(l)} x_k \right) \\ \vdots \\ \sum_{l=1}^{r} a_n^{(l)} \left(\sum_{j=1}^{n} a_j^{(l)} x_j \right) \left(\sum_{k=1}^{n} a_k^{(l)} x_k \right) \end{pmatrix} \\
&= \begin{pmatrix} \sum_{l=1}^{r} a_1^{(l)} \left((a^{(l)})^T x \right) \left((a^{(l)})^T x \right) \\ \sum_{l=1}^{r} a_2^{(l)} \left((a^{(l)})^T x \right) \left((a^{(l)})^T x \right) \\ \vdots \\ \sum_{l=1}^{r} a_n^{(l)} \left((a^{(l)})^T x \right) \left((a^{(l)})^T x \right) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix},
\end{aligned}
$$

where the last equality is valid because $x \in \operatorname{span}(a^{(1)}, \ldots, a^{(r)})^{\perp}$. □

5.3 A numerical algorithm of finding real tensor eigenpairs based on the Newton homotopy and the straightforward approach

In this section, we present the algorithm for computing real eigenpairs based on the straightforward approach in Section 5.2 and Algorithm 5.1.1.

ALGORITHM 5.3.1 (Compute real mode-k $\mathcal{B}$-eigenpairs of $\mathcal{A}$, where $\mathcal{A} \in \mathbb{R}^{[m,n]}, \mathcal{B} \in \mathbb{R}^{[m',n]}$)

Step 1. *Compute all complex eigenpairs using Algorithm 4.3.1 or Algorithm 4.3.2. Let K be the set of resulting eigenpairs (λ, x) with $|\operatorname{Im}(\lambda)| < \delta_0$ (chosen threshold).*

Step 2. *For each eigenpair $(\lambda^*, x^*) \in K$: if (λ^*, x^*) is in a positive dimensional solution component of (4.4) or (2.3), go to Step 3. Otherwise, (λ^*, x^*) is an isolated eigenpair. If $\|\operatorname{Im}(x^*)\|_2 < \delta_0$, then take $(\operatorname{Re}(\lambda^*), \operatorname{Re}(x^*))$ as a real eigenpair and stop.*

Step 3. *Set $\lambda = \operatorname{Re}(\lambda^*)$. If $m = m'$, set $v := \operatorname{Re}(x^*)$ (if $\operatorname{Re}(x^*) \neq 0$) and $w := \operatorname{Im}(x^*)$ (if $\operatorname{Im}(x^*) \neq 0$); otherwise, set*

$$v := \frac{\operatorname{Re}(x^*)}{(\mathcal{B}\operatorname{Re}(x^*)^{m'})^{1/m'}} \quad (\text{if } \mathcal{B}\operatorname{Re}(x^*)^{m'} \neq 0),$$

and

$$w := \frac{\operatorname{Im}(x^*)}{(\mathcal{B}\operatorname{Im}(x^*)^{m'})^{1/m'}} \quad (\text{if } \mathcal{B}\operatorname{Im}(x^*)^{m'} \neq 0).$$

If (λ, v) or (λ, w) is a mode-k $\mathcal{B}$-eigenpair of $\mathcal{A}$, then we have obtained a real eigenpair and stop. Otherwise, goto Step 4.

Step 4. *Starting from (λ^*, x^*), use Algorithm 5.1.1 to find a real eigenpair.*

Chapter 6

Implementation and numerical results

Based on the algorithms introduced in Chapter 4 and Chapter 5, a MATLAB package `TenEig 1.1` has been developed. The package `TenEig 1.1` can be downloaded from

`http://www.math.msu.edu/~chenlipi/TenEig.html`

Consider the tensors $\mathcal{A} \in \mathbb{C}^{[m,n]}$ and $\mathcal{B} \in \mathbb{C}^{[m',n]}$. In `TenEig 1.1`, function `teig` computes generalized mode-k $\mathcal{B}$ eigenvalues and eigenvectors of a tensor $\mathcal{A}$ for $m = m'$. The input of this function is: tensor $\mathcal{A}$ or the polynomial form $\mathcal{A}x^m$ if $\mathcal{A}$ is symmetric (for which $\mathcal{A}x^{m-1} = \nabla(\mathcal{A}x^m)/m$ from Proposition 2.2.3), $\mathcal{B}$ (the default is the unit tensor), mode k (the default value is 1), and the output is: mode-k $\mathcal{B}$ eigenvalues and eigenvectors of $\mathcal{A}$. In general, `teig` computes generalized complex mode-k $\mathcal{B}$ eigenvalues and eigenvectors of $\mathcal{A}$ when $m = m'$. But by default, `teig` finds Qi-eigenvalues and Qi-eigenvectors.

The function `teneig` computes generalized mode-k $\mathcal{B}$ eigenvalues and eigenvectors of a tensor $\mathcal{A}$ for $m \neq m'$. The input of this function is: tensor $\mathcal{A}$ or the polynomial form (if $\mathcal{A}$ is symmetric) $\mathcal{A}x^m$, tensor $\mathcal{B}$, mode k (the default value is 1), and the output is: mode-k $\mathcal{B}$ eigenvalues and eigenvectors of $\mathcal{A}$. If $\mathcal{B}$ is chosen as the identify matrix, `teneig` computes E-eigenvalues and E-eigenvectors of $\mathcal{A}$ as defined in Qi [35].

Since E-eigenpairs of a tensor are frequently in demand, our package includes a separate function `eeig`, which only computes E-eigenpairs of a tensor.

The package also includes two functions `heig` and `zeig` to compute real eigenpairs of a tensor: The first one computes H-eigenpairs and second one computes Z-eigenpairs.

In the next two sections, numerical results are reported to illustrate the effectiveness and efficiency of our methods for computing tensor eigenpairs. All the numerical experiments were performed on a Thinkpad T400 Laptop with an Intel(R) dual core CPU at 2.80GHz and 2GB of RAM, running on a Windows 7 operating system. The package `TenEig 1.1` uses MATLAB 2013a. In the examples, we used `teig` or `teneig` to compute generalized tensor eigenpairs, `teig` to compute Qi-eigenpairs, `eeig` to compute E-eigenvalues, `heig` to compute (real) H-eigenpairs, and `zeig` to compute (real) Z-eigenpairs, respectively.

6.1 Examples for computing complex eigenpairs

In this section, some numerical examples exhibiting the performance of `TenEig 1.1` for computing complex tensor eigenpairs are presented.

We will compare our package `TenEig 1.1` with `NSolve`, a function in Mathematica (based on the Gröebner basis) for solving systems of algebraic equations, in computing complex tensor eigenpairs. Let

$$\begin{aligned} T(m,n) &:= n(m-1)^{n-1}, \\ E(m,n) &:= ((m-1)^n - 1)/(m-2), \\ G(m,m',n) &:= ((m-1)^n - (m'-1)^n)/(m-m'). \end{aligned}$$

Recall that Theorem 3.3.1 shows for tensors $\mathcal{A} \in \mathbb{C}^{[m,n]}$ and $\mathcal{B} \in \mathbb{C}^{[m',n]}$, the number of equivalence classes of isolated $\mathcal{B}$-eigenpairs of $\mathcal{A}$ is bounded by $T(m,n)$ for $m = m'$ and $G(m,m',n)$ for $m \neq m'$. In particular, as mentioned in Remark 3.3.1, the number of equivalence classes of isolated Qi-eigenpairs is bounded by $T(m,n)$ and the number of equivalence classes of isolated E-eigenpairs of $\mathcal{A}$ is bounded by $E(m, n)$.

EXAMPLE 6.1.1 In this example, we compare the performance of our `TenEig 1.1` with `NSolve` and `PSOLVE`. More specifically, when computing Qi-eigenpairs of a generic tensor $\mathcal{A} \in \mathbb{C}^{[m,n]}$,

(a) `teig` is based on Algorithm 4.3.1 using the linear homotopy given in Theorem 4.2.2.

(b) `NSolve` and `PSOLVE` solve the polynomial system $\mathcal{A}x^{m-1} = \lambda x^{[m-1]}$ augmented by a random hyperplane defined in (4.4).

To compute E-eigenpairs,

(a) `eeig` is based on Algorithm 4.3.2 which solves the polynomial system $\mathcal{A}x^{m-1} = \lambda x$ augmented with a random hyperplane and then normalize the resulting solutions to satisfy $x^T x = 1$ afterwards.

(b) `NSolve` and `PSOLVE` solve the polynomial system $\mathcal{A}x^{m-1} = \lambda x$ with $x^T x = 1$ appended defined by (2.3).

The tensors $\mathcal{A}$ were generated using $randn(n, \cdots, n) + i * randn(n, \cdots, n)$ in MATLAB. For instance, $randn(2, 2, 2) + i * randn(2, 2, 2)$ generates a third-order and two dimensional complex tensor. The computing results are given in Table 6.1, in which, N denotes the number of equivalence classes of Qi-eigenpairs or E-eigenpairs found by `teig`, `eeig`, `PSOLVE` or `NSolve`, the CPU times are in seconds, ”-” means no results were returned after 12 hours.

From the table, our codes `teig` and `eeig` find all equivalence classes of Qi-eigenpairs or E-eigenpairs in reasonable amount of time for all cases. `NSolve` fails to provide any results in 12 hours for some cases (we terminated it after 12 hours). Although `PSOLVE` successfully finds all equivalence classes in many cases, it does miss a few equivalence classes sometime. Regarding the CPU time, `PSOLVE` is comparable to `teig` but takes more time than `eeig`.

(m, n)	$T(m, n)$	Alg	N	time (s)	$E(m, n)$	Alg	N	time (s)
$(4, 5)$	405	`teig`	405	15.8	121	`eeig`	121	5.4
		`PSOLVE`	404	14.0		`PSOLVE`	121	9.5
		`NSolve`	405	3136.4		`NSolve`	121	486.6
$(5, 5)$	1280	`teig`	1280	73.8	341	`eeig`	341	22.3
		`PSOLVE`	1280	65.5		`PSOLVE`	341	38.6
		`NSolve`	-	-		`NSolve`	341	9264.8
$(5, 6)$	6144	`teig`	6144	606.5	1365	`eeig`	1365	166.5
		`PSOLVE`	6144	694.2		`PSOLVE`	1365	283.6
		`NSolve`	-	-		`NSolve`	-	-
$(6, 6)$	18750	`teig`	18750	3721.3	3906	`eeig`	3906	990.2
		`PSOLVE`	18748	4636.0		`PSOLVE`	3905	1721.0
		`NSolve`	-	-		`NSolve`	-	-

Table 6.1: Comparison of `teig` and `eeig` with `PSOLVE` and `NSolve`

EXAMPLE 6.1.2 In this example we show the effectiveness and efficiency of `teig` for finding all equivalence classes of isolated Qi-eigenpairs of a generic tensor $\mathcal{A} \in \mathbb{C}^{[m,n]}$. Each tensor was generated by $randn(n, \cdots, n) + i * randn(n, \cdots, n)$ in MATLAB. The results are reported in Table 6.2, in which N is the number of equivalence classes of isolated Qi-eigenpairs found by `teig` and $T(m, n)$ represents the bound of the number of equivalence classes of isolated Qi-eigenpairs (see Remark 3.3.1(a)).

(m, n)	$T(m, n)$	N	time(s)	(m, n)	$T(m, n)$	N	time(s)
$(3, 5)$	80	80	2.4	$(3, 6)$	192	192	6.8
$(3, 7)$	448	448	18.3	$(3, 8)$	1024	1024	53.0
$(3, 9)$	2304	2304	145.9	$(3, 10)$	5120	5120	409.2
$(4, 3)$	27	27	0.7	$(4, 4)$	108	108	2.9
$(4, 5)$	405	405	15.8	$(4, 6)$	1458	1458	80.0
$(4, 7)$	5103	5103	385.9	$(4, 8)$	17496	17496	2115.5
$(5, 3)$	48	48	1.2	$(5, 4)$	256	256	8.8
$(5, 5)$	1280	1280	73.8	$(5, 6)$	6144	6144	606.5
$(5, 7)$	28672	28672	5394.2	$(6, 3)$	75	75	2.3
$(6, 4)$	500	500	21.0	$(6, 5)$	3125	3125	287.7
$(6, 6)$	18750	18750	3721.3	$(7, 3)$	108	108	3.6
$(7, 4)$	864	864	51.5	$(7, 5)$	6480	6480	981.3

Table 6.2: Performance of `teig` on computing Qi-eigenpairs of complex random tensors

EXAMPLE 6.1.3 In this example we show the effectiveness and efficiency of `eeig` for finding all equivalence classes of isolated E-eigenpairs of a generic tensor $\mathcal{A} \in \mathbb{C}^{[m,n]}$. Each generic tensor was generated by $randn(n, \cdots, n) + i * randn(n, \cdots, n)$ in MATLAB. The results are reported in Table 6.3, in which N is the number of equivalence classes of E-eigenpairs found by `eeig` and $E(m, n)$ is the bound of the number of equivalence classes of isolated E-eigenpairs (see Remark 3.3.1(b)).

(m, n)	$E(m, n)$	N	time(s)	(m, n)	$E(m, n)$	N	time(s)
$(3, 5)$	31	31	1.4	$(3, 6)$	63	63	3.1
$(3, 7)$	127	127	7.5	$(3, 8)$	255	255	20.3
$(3, 9)$	511	511	48.5	$(3, 10)$	1023	1023	133.9
$(4, 3)$	13	13	0.4	$(4, 4)$	40	40	1.7
$(4, 5)$	121	121	5.4	$(4, 6)$	364	364	26.9
$(4, 7)$	1093	1093	119.5	$(4, 8)$	3280	3280	555.8
$(5, 3)$	21	21	0.7	$(5, 4)$	85	85	4.2
$(5, 5)$	341	341	22.3	$(5, 6)$	1365	1365	166.5
$(5, 7)$	5461	5461	1330.7	$(6, 3)$	31	31	1.2
$(6, 4)$	156	156	9.5	$(6, 5)$	781	781	100.4
$(6, 6)$	3906	3906	990.2	$(7, 3)$	43	43	1.9
$(7, 4)$	259	259	21.3	$(7, 5)$	1555	1555	245.0

Table 6.3: Performance of `eeig` on computing E-eigenpairs of complex random tensors

According to [35], [7], and [4], for a randomly generated tensor $\mathcal{A} \in \mathbb{C}^{[m,n]}$, it has $T(m, n)$ number of equivalence classes of Qi-eigenpairs and $E(m, n)$ number of equivalence classes of E-eigenpairs. Moreover, its Qi-eigenpairs and E-eigenpairs are isolated. From Tables 6.2 and 6.3, our code `teig` and `eeig` can find all equivalence classes of Qi-eigenpairs and E-eigenpairs of such tensors in the examples we tested.

EXAMPLE 6.1.4 The example here exhibits the effectiveness and efficiency of `teig` and `teneig` for finding all equivalence classes of isolated $\mathcal{B}$-eigenpairs of a generic tensor $\mathcal{A}$, where $\mathcal{A} \in \mathbb{C}^{[m,n]}, \mathcal{B} \in \mathbb{C}^{[m',n]}$ are generic tensors. Each generic tensor was generated by

$randn(n, \cdots, n) + i * randn(n, \cdots, n)$ in MATLAB. The results are reported in Table 6.4, in which N denotes the number of equivalence classes of eigenpairs found by `teig` or `teneig`, $T(m, n)$ denotes the bound of the number of equivalence classes of isolated $\mathcal{B}$-eigenpairs for $m = m'$, and $G(m, m', n)$ denotes the bound of the number of equivalence classes of isolated $\mathcal{B}$-eigenpairs for $m \neq m'$ (see Theorem 3.3.1).

teig ($m = m'$)				teneig ($m \neq m'$)			
(m, n)	$T(m, n)$	N	time(s)	(m, m', n)	$G(m, m', n)$	N	time(s)
$(3, 7)$	448	448	23.7	$(3, 2, 7)$	127	127	10.3
$(3, 8)$	1024	1024	68.3	$(3, 4, 6)$	665	665	68.1
$(3, 9)$	2304	2304	210.3	$(3, 5, 5)$	496	496	49.5
$(4, 5)$	405	405	20.8	$(4, 2, 6)$	364	364	28.9
$(4, 6)$	1458	1458	110.4	$(4, 3, 5)$	211	211	13.2
$(4, 7)$	5103	5103	737.5	$(4, 5, 4)$	175	175	9.5
$(5, 5)$	1280	1280	97.9	$(5, 4, 5)$	781	781	83.9
$(5, 6)$	6144	6144	1178	$(5, 6, 3)$	61	61	2.6
$(6, 4)$	500	500	29.9	$(6, 5, 4)$	369	369	30.7
$(6, 5)$	3125	3125	449.4	$(6, 7, 3)$	91	91	6.0
$(7, 3)$	108	108	4.4	$(7, 6, 4)$	671	671	77.1
$(7, 4)$	864	864	77.6	$(7, 8, 3)$	127	127	9.4

Table 6.4: Performance of `teig` and `teneig` on computing generalized eigenpairs of complex random tensors

Evidently, Table 6.4 shows that our `teig` and `teneig` can find all equivalence classes of isolated $\mathcal{B}$-eigenpairs of $\mathcal{A}$ for generic tensors $\mathcal{A}$ and $\mathcal{B}$ in a reasonable amount of time.

6.2 Computing singular tensor eigenpairs

It is known that the homotopy method uses Newton's method in the endgame. While Newton's method converges rapidly at nonsingular solutions of a polynomial system with high accuracy, it may just attain a few correct digits at singular solutions. Therefore, when homotopy methods is used to solve the polynomial system (3.1) for the tensor eigenvalue problem, desired number of significant digits for a singular eigenpair (λ^*, x^*) may not be achievable, as the following example shows.

EXAMPLE 6.2.1 Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[6,3]}$ (Example 4.10 in [12], see also [4]) in the polynomial form

$$\mathcal{A}x^6 = x_1^4x_2^2 + x_1^2x_2^4 + x_3^6 - 3x_1^2x_2^2x_3^2.$$

To compute its Qi-eigenpairs, by Proposition 2.2.3, the corresponding polynomial system is

$$\begin{aligned}
\frac{1}{6}(4x_1^3x_2^2 + 2x_1x_2^4 - 6x_1x_2^2x_3^2) - \lambda x_1^5 &= 0 \\
\frac{1}{6}(2x_1^4x_2 + 4x_1^2x_2^3 - 6x_1^2x_2x_3^2) - \lambda x_2^5 &= 0 \\
\frac{1}{6}(6x_3^5 - 6x_1^2x_2^2x_3) - \lambda x_3^5 &= 0 \\
a_1x_1 + a_2x_2 + a_3x_3 + b &= 0,
\end{aligned}$$

where a_1, a_2, a_3, b are complex random numbers. Let the system be $P(\lambda, x) = 0$. According to Qi [35], the tensor $\mathcal{A}$ should have $3(6-1)^{3-1} = 75$ eigenvalues. Table 6.5 lists all the 75 eigenvalues together with their corresponding eigenvectors. In the table, $\lambda^{(l)}$ means there are l eigenvectors associated with λ counting multiplicities, the multiplicity of each eigenpair as a solution of $P(\lambda, x) = 0$. For instance, $\lambda^{(l)} = -1.1445^{(8)}$ indicates there are 8 eigenvectors associated with -1.1445. The corresponding

$$x^T = (1, \pm(0.4209 - 0.9071i), \pm(0.4447 + 0.6965i))$$

with multiplicity 1 represents each of following four eigenvectors:

$$(1, +(0.4209 - 0.9071i), +(0.4447 + 0.6965i)),$$

$$(1, +(0.4209 - 0.9071i), -(0.4447 + 0.6965i)),$$

$$(1, -(0.4209 - 0.9071i), +(0.4447 + 0.6965i)),$$

$$(1, -(0.4209 - 0.9071i), -(0.4447 + 0.6965i))$$

together with $\lambda = -1.1445$ has multiplicity 1 as a solution of $P(\lambda, x) = 0$. Of course, eigenpairs with multiplicity 1 are nonsingular and all the others are singular.

$\lambda^{(l)}$	x^T	multiplicity
$-1.1445^{(8)}$	$(1, \pm(0.4209 - 0.9071i), \pm(0.4447 + 0.6965i))$	1
	$(1, \pm(0.4209 + 0.9071i), \pm(0.4447 - 0.6965i))$	1
$-\frac{1}{3}^{(6)}$	$(1, -i, 0)$	3
	$(1, i, 0)$	3
$0^{(14)}$	$(0, 1, 0)$	5
	$(1, 0, 0)$	5
	$(-1, 1, -1)$	1
	$(1, 1, -1)$	1
	$(1, 1, 1)$	1
	$(-1, 1, 1)$	1
$0.0555^{(8)}$	$(\pm 0.4568, 1, \pm 0.6856)$	1
	$(1, \pm 0.4568, \pm 0.6856)$	1
$0.5445 - 0.5350i^{(8)}$	$(\pm(0.7318 + 0.6111i), \pm(0.8490 - 0.2286i), 1)$	1
	$(\pm(0.8490 - 0.2286i), \pm(0.7318 + 0.6111i), 1)$	1
$0.5445 + 0.5350i^{(8)}$	$(\pm(0.8490 + 0.2286i), \pm(0.7318 - 0.6111i), 1)$	1
	$(\pm(0.7318 - 0.6111i), \pm(0.8490 + 0.2286i), 1)$	1
$1^{(15)}$	$(0, 0, 1)$	13
	$(1, -1, 0)$	1
	$(1, 1, 0)$	1
$1.5 - 0.8660i^{(4)}$	$(1, \pm 1, \pm(0.5 + 0.8660i))$	1
$1.5 + 0.8660i^{(4)}$	$(1, \pm 1, \pm(0.5 - 0.8660i))$	1

Table 6.5: Eigenpairs of the tensor in Example 6.2.1

By Algorithm 4.3.1, all 75 solutions of the system $P(\lambda, x) = 0$ can be found, and very accurately for the 46 nonsingular solutions. For instance, the approximation to the nonsingular solution $(\lambda^*, x^*) = (0, 1, 1, -1)^T$:

$$\begin{pmatrix} \lambda^* \\ x_1^* \\ x_2^* \\ x_3^* \end{pmatrix} = \begin{pmatrix} 0.000000000000000 + 0.000000000000000i \\ 1.000000000000000 + 0.000000000000000i \\ 1.000000000000000 + 0.000000000000000i \\ -1.000000000000000 - 0.000000000000000i \end{pmatrix}$$

achieves accuracy up to the machine precision. However, the algorithm cannot achieve so many significant digits for a singular solution. For example, the approximation to the singular solution $(\lambda^*, x^*) = (-1/3, 1, i, 0)^T$ is

$$\begin{pmatrix} \lambda^* \\ x_1^* \\ x_2^* \\ x_3^* \end{pmatrix} = \begin{pmatrix} -0.333333333338482 - 0.000000000005507i \\ 1.000000000000000 + 0.000000000000000i \\ -0.000002043599517 + 0.999999061292030i \\ 0.000000000000000 + 0.000000000000000i \end{pmatrix}.$$

To improve the accuracy of singular eigenpairs we further use the so-called deflation method (see [13]).

Let $(\hat{\lambda}, \hat{x})$ be a singular zero of $P(\lambda, x)$. The deflation method starts from the observation that $DP(\hat{\lambda}, \hat{x}) \in \mathbb{C}^{n+1,n+1}$ is singular. Assume $DP(\hat{\lambda}, \hat{x}) \in \mathbb{C}^{n+1,n+1}$ is deficient of rank d. Then for almost all $d \times (n+1)$ random matrix R, the matrix

$$\begin{bmatrix} DP(\hat{\lambda}, \hat{x}) \\ R \end{bmatrix}$$

is of full column rank. Let $e_1 := (1, 0, \ldots, 0)^T \in \mathbb{R}^d$. Then it is clear that the linear system

$$\begin{bmatrix} DP(\hat{\lambda}, \hat{x}) \\ \\ R \end{bmatrix} y = \begin{bmatrix} 0 \\ \\ e_1 \end{bmatrix}$$

has a unique solution $y = \hat{y}$ in $\mathbb{C}^{n+1}$. We now construct a new $(2(n+1)+d) \times (n+1)$ system

$$Q(\lambda, x, y) := \begin{bmatrix} P(\lambda, x) \\ \begin{bmatrix} DP(\lambda, x) \\ \\ R \end{bmatrix} y - \begin{bmatrix} 0 \\ \\ e_1 \end{bmatrix} \end{bmatrix} = 0.$$

If $\hat{z} := (\hat{\lambda}, \hat{x}, \hat{y})$ is a simple zero of $Q(z) := Q(\lambda, x, y)$, $DQ(\hat{z})$ must be of full rank. Denote

$$(DQ(z))^{\dagger} := [(DQ(z))^T (DQ(z))]^{-1} (DQ(z))^T.$$

Then the Gaussian-Newton iterations

$$z^{(j+1)} = z^{(j)} - (DQ(z^{(j)}))^{\dagger} Q(z^{(j)}) \quad \text{for } j = 0, 1, \ldots$$

with $z^{(0)} := (\lambda^*, x^*, \hat{y})$ can be used, until the residue $\|Q(z^{(j+1)})\|_2$ is within the desired accuracy. This will lead to a much more accurate $(\tilde{\lambda}^*, \tilde{x}^*)$.

If $\hat{z}$ is a multiple zero of $Q(z) := Q(\lambda, x, y)$, the deflation procedure can be repeated on $Q(z)$ until a satisfactory $(\tilde{\lambda}^*, \tilde{x}^*)$ is achieved.

Returning to Example 6.2.1, when the deflation method is used in conjunction with Algorithm 4.3.1, the eigenpair $(\lambda^*, x^*) = (-1/3, 1, i, 0)^T$ can be computed remarkably accurate:

$$\begin{pmatrix} \tilde{\lambda}^* \\ \tilde{x}_1^* \\ \tilde{x}_2^* \\ \tilde{x}_3^* \end{pmatrix} = \begin{pmatrix} -0.333333333333333 + 0.000000000000000i \\ 1.000000000000000 + 0.000000000000000i \\ 0.000000000000000 + 1.000000000000000i \\ -0.000000000000000 + 0.000000000000000i \end{pmatrix}.$$

6.3 Examples for Computing Real Eigenpairs

In this section, numerical examples are presented to illustrate the effectiveness and efficiency of `zeig` or `heig` for computing real Z-eigenpairs or H-eigenpairs of a tensor $\mathcal{A} \in \mathbb{R}^{[m,n]}$. By Definition 2.1.1, (λ, x) is a Z-eigenpair if and only if $((-1)^{m-2}\lambda, -x)$ is a Z-eigenpair, and (λ, x) is an H-eigenpair if and only if (λ, tx) is an H-eigenpair for any nonzero $t \in \mathbb{R}$. Only one representative from each equivalence class of eigenpairs will be listed in our tables. The notation $\lambda^{(l)}$ represents l eigenvectors are found for the eigenvalue λ. In the following tables, the multiplicity of an eigenpair means the multiplicity of this eigenpair as a zero of the corresponding polynomial system. For conciseness, the polynomial system that solves the tensor eigenvalue problem will be omitted.

EXAMPLE 6.3.1 Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[6,3]}$ whose corresponding polynomial form is the Motzkin polynomial:

$$\mathcal{A}x^6 = x_3^6 + x_1^4 x_2^2 + x_1^2 x_2^4 - 3x_1^2 x_2^2 x_3^2.$$

Stated in Example 5.9 of [4], this tensor has 25 equivalence classes of Z-eigenpairs as shown in Table 6.6. Exactly 25 equivalence classes of Z-eigenpairs are found by `zeig`, agreeing with the results of [4]. It takes `zeig` about 0.9 seconds for the entire computation.

λ	x^T	multiplicity
$0^{(14)}$	$(0.5774, \pm 0.5774, \pm 0.5774)$	1
	$(1, 0, 0)$	5
	$(0, 1, 0)$	5
$0.0156^{(8)}$	$(0.8253, \pm 0.2623, \pm 0.5000)$	1
	$(0.2623, \pm 0.8253, \pm 0.5000)$	1
$0.2500^{(2)}$	$(0.7071, \pm 0.7071, 0)$	1
1	$(0, 0, 1)$	1

Table 6.6: Z-eigenpairs of the tensor in Example 6.3.1

As shown in Table 6.7 all the H-eigenpairs in Example 4.10 of [12] are also found by `heig`. The entire computation time is 1.7 seconds.

λ	x^T	multiplicity
$0^{(14)}$	$(\pm 1, 1, \pm 1)$	1
	$(1, 0, 0)$	5
	$(0, 1, 0)$	5
$0.0555^{(8)}$	$(\pm 0.4568, 1, \pm 0.6856)$	1
	$(1, \pm 0.4568, \pm 0.6856)$	1
$1^{(15)}$	$(0, 0, 1)$	13
	$(1, \pm 1, 1)$	1

Table 6.7: H-eigenpairs of the tensor in Example 6.3.1

For computing all real eigenvalues of a symmetric tensor the only available method at this time is Algorithm 3.6 in [12]. In the next two examples, we compare the performance of our methods with Algorithm 3.6 in [12].

EXAMPLE 6.3.2 In this example, our `zeig` is used to compute the Z-eigenvalues of 12 symmetric tensors from [12]. The test problems and numerical results are given in the Appendix. From the numerical results, our `zeig` finds all the Z-eigenvalues found by Algorithm

3.6 ([12]) on this set of problems. The CPU times[1] (in seconds) used by `zeig` and by Algorithm 3.6 ([12]) are shown in Table 6.8. (The CPU times by Algorithm 3.6 are from [12]).

Problem	`zeig` time(s)	Algorithm 3.6 ([12]) time(s)
1	0.3	9
2	4	400
3	0.3–0.4	5–20
4	0.1	1
5	0.6	9
6	1.8	10870
7	15.7	280
8	6.1	320
9	0.3	1
10	6.3	370
11	27.3	170
12	4.5	420

Table 6.8: `zeig` vs Algorithm 3.6 ([12]): CPU time

EXAMPLE 6.3.3 Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[4,n]}$ (Example 4.16 in [12]) in the polynomial form

$$\begin{aligned}\mathcal{A}x^4 &= (x_1 - x_2)^4 + \cdots + (x_1 - x_n)^4 + (x_2 - x_3)^4 + \cdots + (x_2 - x_n)^4 \\ &\quad + \cdots + (x_{n-1} - x_n)^4.\end{aligned}$$

Shown in Table 6.9, our `zeig` found all the Z-eigenvalues found by Algorithm 3.6 in [12] for different n. Notice that when $n = 8, 9, 10$, our `zeig` can find all the Z-eigenvalues in a reasonable amount of time, but [12] reports that Algorithm 3.6 can only find the first three

[1]The computer used in [12] is a Thinkpad W520 laptop with an Intel dual core CPU at 2.20GHz and 8 GB RAM. The computer used in this paper is a Thinkpad T400 laptop with an Intel dual core CPU at 2.80GHz and 2GB RAM. The reader should be cautious when comparing the CPU time of the methods since different computers were used.

largest Z-eigenvalues. The CPU times used by `zeig` and Algorithm 3.6 ([12]) are listed in the table. (The CPU times by Algorithm 3.6 ([12]) are from [12][2].) The corresponding Z-eigenvectors are not displayed.

n	λ	time(s)	
		`zeig`	Algorithm 3.6 ([12])
4	0.0000 4.0000 5.0000 5.3333	1.7	3.6
5	0.0000, 4.1667, 4.2500, 5.5000, 6.2500	5.4	274.5
6	0.0000, 4.0000, 4.5000, 6.0000, 7.2000	15.5	280.2
7	0.0000, 4.0833, 4.1667, 4.7500, 4.8846, 4.9000, 6.5000, 8.1667	58.3	9565.6
8	0.0000, 4.0000, 4.2667, 4.2727, 4.3333, 5.0000, 5.2609, 5.3333, 7.0000, 9.1429	244.1	938.2*
9	0.0000, 4.0500, 4.1250, 4.5000, 5.2500, 5.6250, 5.7857, 7.5000, 10.1250	788.0	4173.8*
10	0.0000, 4.0000, 4.1667, 4.1818, 4.2500, 4.6667, 4.7500, 4.7593, 4.7619, 5.5000, 5.9808, 6.2500, 8.0000, 11.1111	2665.6	15310.5*

Table 6.9: Z-eigenvalues of the tensor in Example 6.3.3 (* denotes that the CPU time used by Algorithm 3.6 ([12]) when it finds the first three largest Z-eigenvalues)

[2] Again, we should be cautious when comparing the CPU times used by the two methods because of different computers were used.

APPENDIX

PROBLEM 1 Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[4,3]}$ (Example 4.1 in [12], see also [35]) with the polynomial form

$$\mathcal{A}x^4 = x_1^4 + 2x_2^4 + 3x_3^4.$$

Our `zeig` obtains all the Z-eigenpairs found in [12] (see Table A.1), taking about 0.3 seconds to carry out the entire computation, while Algorithm 3.6 ([12]) needs 9 seconds[3] for the computation.

λ	x^T	multiplicity
$0.5455^{(4)}$	$(0.7385, \pm 0.5222, \pm 0.4264)$	1
$0.6667^{(2)}$	$(0.8165, \pm 0.5774, 0)$	1
$0.7500^{(2)}$	$(0.8660, 0, \pm 0.5000)$	1
1	$(1, 0, 0)$	1
$1.2^{(2)}$	$(0, 0.7746, \pm 0.6325)$	1
2	$(0, 1, 0)$	1
3	$(0, 0, 1)$	1

Table A.1: Z-eigenpairs of the tensor in Problem 1

PROBLEM 2 For diagonal tensor $\mathcal{D} \in \mathbb{R}^{[5,4]}$ (Example 4.2 in [12]) where $\mathcal{D}x^5 = x_1^5 + 2x_2^5 - 3x_3^5 - 4x_4^5$, consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[5,4]}$ such that $\mathcal{A}x^5 = \mathcal{D}(Qx)^5$ where

$$Q = (I - 2w_1w_1^T)(I - 2w_2w_2^T)(I - 2w_3w_3^T)$$

and w_1, w_2, w_3 are randomly generated unit vectors. Our `zeig` takes about 4.0 seconds to reach all the 30 Z-eigenpairs found in [12], in which Algorithm 3.6 spends 400 seconds to

[3]All the CPU times by Algorithm 3.6 ([12]) are from [12]. The computer used in [12] is a Thinkpad W520 laptop with an Intel dual core CPU at 2.20GHz and 8 GB RAM. The computer used in this paper is a Thinkpad T400 laptop with an Intel dual core CPU at 2.80GHz and 2GB RAM. The reader should be cautious when comparing the CPU time of the methods since different computers were used.

complete the computation. The 15 nonnegative Z-eigenvalues are

$$0.2518, \quad 0.3261, \quad 0.3466, \quad 0.3887, \quad 0.4805, \quad 0.5402, \quad 0.5550, \quad 0.6057,$$

$$0.8543, \quad 0.9611, \quad 1.0000, \quad 1.2163, \quad 2.0000, \quad 3.0000, \quad 4.0000.$$

PROBLEM 3 Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[4,3]}$ (Example 4.3 in [12], see also [35]) with the polynomial form

$$\mathcal{A}x^4 = 2x_1^4 + 3x_2^4 + 5x_3^4 + 4ax_1^2x_2x_3,$$

where a is a parameter. For different values of a, our `zeig` found all Z-eigenvalues that appeared in [12], as shown in Table A.2. The last column of the table is the CPU time used

a	λ	time(s)
0	$0.9677^{(4)}$, $1.2000^{(2)}$, $1.4286^{(2)}$, $1.8750^{(2)}$, 2, 3, 5	0.4
0.25	$0.8464^{(2)}$, $1.0881^{(2)}$, $1.2150^{(2)}$, $1.4412^{(2)}$, $1.8750^{(2)}$, 2, 3, 5	0.4
0.5	$0.7243^{(2)}$, $1.2069^{(2)}$, $1.2593^{(2)}$, $1.4783^{(2)}$, $1.8750^{(2)}$, 2, 3, 5	0.4
1	$0.4787^{(2)}$, $1.6133^{(2)}$, $1.8750^{(2)}$, 2, 3, 5	0.3
3	$-0.5126^{(2)}$, $1.8750^{(2)}$, 2, $2.2147^{(2)}$, 3, 5	0.3

Table A.2: Z-eigenvalues of the tensor in Problem 3

by `zeig` for each a. As one can see `zeig` needs 0.3 to 0.4 seconds to find all eigenpairs for different values of a, but this takes Algorithm 3.6 ([12]) 5 to 20 seconds.

PROBLEM 4 Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[4,2]}$ (Example 4.4 in [12], see also [35]) in the polynomial form

$$\mathcal{A}x^4 = 3x_1^4 + x_2^4 + 6ax_1^2x_2^2,$$

where a is a parameter. Table A.3 listed all Z-eigenvalues found by both [12] and `zeig`.

a	λ	time(s)
-1	$-0.6000^{(2)}$, 1, 3	0.1
0	$0.7500^{(2)}$, 1, 3	0.1
0.25	$0.9750^{(2)}$, 1, 3	0.1
0.5	1, 3	0.1
2	1, 3, $4.1250^{(2)}$	0.1
3	1, 3, $5.5714^{(2)}$	0.1

Table A.3: Z-eigenvalues of the tensor in Problem 4

For each a, it takes `zeig` 0.1 second, but Algorithm 3.6 ([12]) 1 second.

PROBLEM 5 Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[4,3]}$ (Example 4.5 in [12], see also [23] or [34]) such that

$$A_{1111} = 0.2883, A_{1112} = -0.0031, A_{1113} = 0.1973, A_{1122} = -0.2485,$$

$$A_{1123} = -0.2939, A_{1133} = 0.3847, A_{1222} = 0.2972, A_{1223} = 0.1862,$$

$$A_{1233} = 0.0919, A_{1333} = -0.3619, A_{2222} = 0.1241, A_{2223} = -0.3420,$$

$$A_{2233} = 0.2127, A_{2333} = 0.2727, A_{3333} = -0.3054.$$

All the Z-eigenpairs found in [12] listed in Table A.4 can also be found by `zeig`. For the entire computation, `zeig` takes about 0.6 seconds but Algorithm 3.6 ([12]) 9 seconds.

PROBLEM 6 Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[3,6]}$ (Example 4.6 in [12], see also [37]) such that $A_{iii} = i$ for $i = 1, \ldots, 6$ and $A_{i,i,i+1} = 10$ for $i = 1, \ldots, 5$ and zero otherwise. Again `zeig` did not miss any Z-eigenvalues found in [12]. The 19 nonnegative Z-eigenvalues

λ	x^T	multiplicity
-1.0954	(-0.5915, 0.7467, 0.3043)	1
-0.5629	(-0.1762, 0.1796, -0.9678)	1
-0.0451	(0.7797, 0.6135, 0.1250)	1
0.1735	(0.3357, 0.9073, 0.2531)	1
0.2433	(-0.9895, -0.0947, 0.1088)	1
0.2628	(-0.1318, 0.4425, 0.8870)	1
0.2682	(0.6099, 0.4362, 0.6616)	1
0.3633	(0.2676,0.6447, 0.7160)	1
0.5105	(-0.3598,0.7780,-0.5150)	1
0.8169	(-0.8412,0.2635,-0.4722)	1
0.8893	(-0.6672, -0.2471, 0.7027)	1

Table A.4: Z-eigenpairs of the tensor in Problem 5

are:

3.9992	4.0225	4.2464	4.3358	5.1402	5.4817	5.5218	5.5668
5.5674	6.0000	7.2165	8.1889	8.5979	8.6596	8.7347	10.9711
15.4298	15.4552	16.2345					

It takes `zeig` about 1.8 seconds, while Algorithm 3.6 ([12]) as much as 10870 seconds, for the entire computation.

PROBLEM 7 Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[4,6]}$ (Example 4.7 in [12], see also [28]) with the polynomial form

$$
\begin{aligned}
-\mathcal{A}x^4 \;\; = \;\; & (x_1 - x_2)^4 + (x_1 - x_3)^4 + (x_1 - x_4)^4 + (x_1 - x_5)^4 + (x_1 - x_6)^4 \\
& + (x_2 - x_3)^4 + (x_2 - x_4)^4 + (x_2 - x_5)^4 + (x_2 - x_6)^4 \\
& + (x_3 - x_4)^4 + (x_3 - x_5)^4 + (x_3 - x_6)^4 \\
& + (x_4 - x_5)^4 + (x_4 - x_6)^4 + (x_5 - x_6)^4.
\end{aligned}
$$

All 5 Z-eigenvalues found in [12] listed in Table A.5 are also found by `zeig`. As mentioned

λ	x^T	multiplicity
$-7.2000^{(6)}$	$(0.1826, 0.1826, 0.1826, 0.1826, 0.1826, -0.9129)$	1
$-6.0000^{(15)}$	$(0.7071, 0, 0, 0, 0, -0.7071)$	1
$-4.5000^{(\star)}$	$(1.5703, 1.5703, -0.4923, -0.4923, -1.0780, -1.0780)$	-
$-4.0000^{(10)}$	$(0.4082, 0.4082, 0.4082, -0.4082, -0.4082, -0.4082)$	1
$0^{(\star)}$	$(0.4088, 0.4088, 0.4083, 0.4083, 0.4076, 0.4076)$	-

Table A.5: Z-eigenpairs of the tensor in Problem 7

in [12], every permutation of a Z-eigenvector is also a Z-eigenvector. We only listed the Z-eigenvector with $x_1 \geq x_2 \geq \cdots \geq x_6$ corresponding to one Z-eigenvalue. Actually the Z-eigenpairs corresponding to Z-eigenvalues 0 and -4.5 form a positive dimensional solution component of the corresponding polynomial system. Therefore, there are infinitely many Z-eigenvectors associated with 0 and -4.5. `zeig` finds 484 Z-eigenvectors associated with 0 and 180 Z-eigenvectors associated with -4.5. Only one of these Z-eigenvectors for each case is listed in Table A.6. `zeig` takes about 15.7 seconds to do the entire computation, while Algorithm 3.6 ([12]) spends 280 seconds.

PROBLEM 8 Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[4,5]}$ (Example 4.8 in [12], see also [44]) in the polynomial form

$$\mathcal{A}x^4 = (x_1 + x_2 + x_3 + x_4)^4 + (x_2 + x_3 + x_4 + x_5)^4.$$

All the 3 Z-eigenvalues found in [12] are also found by `zeig`, which are shown in Table A.6.

λ	x^T	multiplicity
$0^{(\star)}$	$(0.9736, -0.4533, -0.5063, -0.0131, 0.9712)$	-
0.5000	$(0.7071, 0, 0, 0, -0.7071)$	1
24.5000	$(0.2673, 0.5345, 0.5345, 0.5345, 0.2673)$	1

Table A.6: Z-eigenpairs of the tensor in Problem 8

For this tensor, the Z-eigenpairs corresponding to Z-eigenvalue 0 form a positive dimensional solution component of the corresponding polynomial system. Thus, there are infinitely many Z-eigenvectors associated with Z-eigenvalue 0. `zeig` finds 234 of them. Only one of them is listed in Table A.6. `zeig` uses about 6.1 seconds, while it takes Algorithm 3.6 ([12]) 320 seconds, for the entire computation.

PROBLEM 9 For the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[3,3]}$ (Example 4.9 in [12], see also [4]) in the polynomial form

$$\mathcal{A}x^3 = 2x_1^3 + 3x_1x_2^2 + 3x_1x_3^2.$$

The Z-eigenpairs corresponding to Z-eigenvalue 2 form a positive dimensional solution component of the corresponding polynomial system. Thus, there are infinitely many Z-eigenvectors associated with Z-eigenvalue 0. `zeig` finds 7 of them. Only one of them is listed in Table A.7. `zeig` uses about 0.3 seconds, while Algorithm 3.6 ([12]) spends 1 second, to do the entire computation.

λ	x^T	multiplicity
2	$(1, 0, 0)$	-

Table A.7: Z-eigenpairs of the tensor in Problem 9

PROBLEM 10 Let the tensor $\mathcal{A} \in \mathbb{R}^{[4,n]}$ (Example 4.12 in [12], see also [34]) be

$$\mathcal{A}_{i_1,\ldots,i_4} = \sin(i_1 + i_2 + i_3 + i_4).$$

When $n = 5$, exhibited in Table A.8 are all the 5 Z-eigenvalues found by `zeig`, which agree with those appeared in [12].

λ	x^T	multiplicity
-8.8463	$(0.5809, 0.3563, -0.1959, -0.5680, -0.4179)$	1
-3.9204	$(-0.1785, 0.4847, 0.7023, 0.2742, -0.4060)$	1
$0^{(\star)}$	$(-0.9914, 0.3771, -0.2946, -0.6360, -0.0534)$	-
4.6408	$(0.5055, -0.1228, -0.6382, -0.5669, 0.0256)$	1
7.2595	$(0.2686, 0.6150, 0.3959, -0.1872, -0.5982)$	1

Table A.8: Z-eigenpairs of the tensor in Problem 10

For this tensor, the Z-eigenpairs corresponding to Z-eigenvalue 0 form a positive dimensional solution component of the corresponding polynomial system. Thus, there are infinitely many Z-eigenvectors associated with 0. `zeig` finds 234 of them. Only one of them is listed in Table A.8. `zeig` takes about 6.3 seconds with Algorithm 3.6 ([12]) using 370 seconds for the entire computation.

PROBLEM 11 Consider the tensor $\mathcal{A} \in \mathbb{R}^{[4,n]}$ (Example 4.13 in [12]) such that

$$\mathcal{A}_{i_1,\dots,i_4} = \tan(i_1) + \cdots + \tan(i_4).$$

When $n = 6$, `zeig` found all the 3 Z-eigenvalues found in [12], which are given in Table A.9.

λ	x^T	multiplicity
-133.2871	$(0.1936, 0.5222, 0.3429, 0.2287, 0.6272, 0.3559)$	1
$0^{(\star)}$	$(-1.9950, -0.5791, 0.2737, 1.6411, 0.1326, 0.5277)$	-
45.5045	$(0.6281, 0.0717, 0.3754, 0.5687, -0.1060, 0.3533)$	1

Table A.9: Z-eigenpairs of the tensor in Problem 11

For this tensor, the Z-eigenpairs corresponding to Z-eigenvalue 0 form a positive dimensional solution component of the corresponding polynomial system. Thus, there are infinitely many Z-eigenvectors associated with 0. `zeig` finds 724 of them. Only one of them is listed in Table A.9. It takes `zeig` about 27.3 seconds, but Algorithm 3.6 ([12]) 170 seconds, to carry out the entire computation.

PROBLEM 12 Consider the tensor $\mathcal{A} \in \mathbb{R}^{[5,n]}$ (Example 4.14 in [12]) such that

$$\mathcal{A}_{i_1,\ldots,i_5} = \ln(i_1) + \cdots + \ln(i_5).$$

For $n = 4$, all the 3 Z-eigenvalues found in [12] are also found by `zeig`, which are shown in Table A.10.

λ	x^T	multiplicity
$0^{(\star)}$	$(0.9914, -1.2262, 0.1847, 0.0523)$	-
0.7074	$(-0.9054, -0.3082, 0.0411, 0.2890)$	1
132.3070	$(0.4040, 0.4844, 0.5319, 0.5657)$	1

Table A.10: Z-eigenpairs of the tensor in Problem 12

For this problem, the Z-eigenpairs corresponding to Z-eigenvalue 0 form a positive dimensional solution component of the corresponding polynomial system. Thus, there are infinitely many Z-eigenvectors associated with 0. `zeig` finds 166 of them. Only one of them is listed in Table A.9. The entire computation takes `zeig` about 4.5 seconds, while Algorithm 3.6 ([12]) uses 420 seconds.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] E.L. Allgower and K. Georg, *Numerical Continuation Methods, an Introduction*, Springer Series in Comput. Math., Vol 13, Springer-Verlag (Berlin, Heidelberg, New York), 1990.

[2] D. N. Bernstein, The number of the roots of a system of equations, *Funct. Anal. Appl.*, 1975, 9:183-185.

[3] D.L. Bates, J.D. Hauenstein, A.J. Sommese and C.W. Wampler, *Numerically Solving Polynomial Systems with Bertini*, Society for Industrial and Applied Mathematics, Philadelphia, 2013.

[4] D. Cartwright and B. Sturmfels, The number of eigenvalues of a tensor, *Linear Algebra and its Applications*, 2013, 438: 942–952.

[5] K.C. Chang, K. Pearson and T. Zhang, Perron-Frobenius theorem for nonnegative tensors, *Commun. Math. Sci.*, 2008, 6(5): 507–520.

[6] K.C. Chang, K. Pearson and T. Zhang, On eigenvalues of real symmetric tensors, *Journal of Mathematical Analysis and Applications*, 2009, 350: 416–422.

[7] K.C. Chang, L. Qi, and T. Zhang, A survey of the spectral theory of nonnegative tensors, *Numerical Linear Algebra with Applications*, 2013, 20: 891–912.

[8] D. A. Cox, J. Little, and D. O'Shea, *Using Algebraic Geometry*, 2nd ed., Springer-Verlag, New York, NY, 2005.

[9] D. A. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, 3rd ed., Springer-Verlag, Secaucus, NJ, USA, 2007.

[10] J. Cooper and A. Duttle, Spectra of uniform hypergrphs, *Linear Algebra and its Applications*, 2012, 436: 3268–3292.

[11] D.A. Cox, J.B. Little and H. Schenck, *Toric varieties*, graduate studies in mathematics, Am. Math. Soc. 2011, 124.

[12] C.-F. Cui, Y.-H. Dai and J. Nie, All real eigenvalues of symmetric tensors, arXiv:1403.3720, 2014.

[13] B. Dayton, T. Li and Z. Zeng, Multiple zeros of nonlinear systems, *Math. Comp.*, 2011, 80: 2143–2168.

[14] S. Friedland, S. Gaubert and L. Han, Perron-Frobenius theorem for nonnegative multilinear forms and extensions, *Linear Algebra and Applications*, 2013, 438: 738–749.

[15] W. Fulton, *e Introduction to toric varieties*, in: William H. Roever (Ed.), in: Lectures in Geometry, Annals of Mathematics Studies, vol. 131, Princeton University Press, Princeton, NJ, 1993.

[16] T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa and T. Mizutani, PHoM - A polyhedral homotopy continuation method, *Computing*, 2004, 73: 53-57.

[17] L. Han, An unconstrained optimization approach for finding real eigenvalues of even order symmetric tensors, *Numerical Algebra, Control and Optimization*, 2013, 3: 583–599.

[18] C. Hillier and L.-H. Lim, Most tensor problems are NP-hard, *Journal of the ACM*, 2013, 60, no. 6, art. 45.

[19] R. Horn and C.R. Johnson, *Matrix Analysis*, 2nd ed., Cambridge University Press, 2013.

[20] S. Hu, Z. Huang, and L. Qi, Finding the extreme Z-eigenvalues of tensors via a sequential semidefinite programming method, *Numerical Linear Algebra with Applications*, 2013, 20: 972–984.

[21] S. Hu, Li. Qi, and B. Zhang, The geometric measure of entanglement of pure states with nonnegative amplitudes and the spectral theory of nonnegative tensors, arXiv:1203.3675, 2012.

[22] B. Huber and B. Sturmfels, A polyhedral method for solving sparse polynomial systems, *Math. Comp.*, 1995, 64: 1541-1555.

[23] T.G. Kolda and J.R. Mayo, Shifted power method for computing tensor eigenpairs, *SIAM Journal on Matrix Analysis and Applications*, 2011, 32: 1095–1124.

[24] T.G. Kolda and J.R. Mayo, An adaptive shifted power method for computing generalized tensor eigenpairs, *SIAM Journal on Matrix Analysis and Applications*, 2014, 35: 1563-1581.

[25] T.L. Lee and T.Y. Li, Mixed Volume Computation in solving polynomial systems, *Contemporary Mathematics*, 2011, 556:97-112.

[26] T.L. Lee, T.Y. Li, and C.H. Tsai, Hom4PS-2.0, a software package for solving polynomial systems by the polyhedral homotopy continuation method, *Computing*, 2008, 83:109–133.

[27] T.Y. Li, Solving polynomial systems by the homotopy continuation method, *Handbook of Numerical Analysis*, XI, 2003, 209–304.

[28] G. Li, L. Qi, and G. Yu, The Z-eigenvalues of a symmetric tensor and its application to spectral hypergraph theory, *Numerical Linear Algebra with Applications*, 2013, 20:1001-1029.

[29] L.-H. Lim, Singular values and eigenvalues of tensors: a variational approach, *Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP'05)*, 2005, 1: 129–132.

[30] T.Y. Li and X. Wang, The BKK root count in $\mathbb{C}^n$, *Math. Comp.*, 1996, 65: 1477-1484.

[31] Y. Liu, G, Zhou, and N. F. Ibrahim, An always convergent algorithm for the largest eigenvalue of an irreducible nonnegative tensor, *Journal of Computational and Applied Mathematics*, 2010, 235: 286–292.

[32] A.P. Morgan, *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*, Society for Industrial and Applied Mathematics, Philadelphia, 2009.

[33] M. Ng, L. Qi, and G. Zhou, Finding the largest eigenvalue of a nonnegative tensor, *SIAM Journal on Matrix Analysis and Applications*, 2009, 31: 1090–1099.

[34] J. Nie and L. Wang, Semidefinite relaxations for best rank-1 tensor approximations, *SIAM Journal on Matrix Analysis and Applications* 2014, 35: 1155–1179.

[35] L. Qi, Eigenvalues of a real supersymmetric tensor, *J. Symb. Comput.*, 2005, 40: 1302–1324.

[36] L. Qi, W. Sun, and Y. Wang, Numerical multilinear algebra and its applications, *Frontiers of Mathematics in China*, 2007, 2: 501–526.

[37] L. Qi, F. Wang, and Y. Wang, Z-eigenvalue methods for a global optimization polynomial optimization problem, *Mathematical Programming*, 2009, 118: 301–306.

[38] L. Qi, Y. Wang, and E.X. Wu, D-eigenvalues of diffusion kurtosis tensors, *Journal of Computational and Applied Mathematics*, 2008, 221: 150–157.

[39] L. Qi, G. Yu, and E.X. Wu, Higher order positive semi-definite diffusion tensor imaging, *SIAM Journal on Imaging Sciences*, 2010, 3: 416–433.

[40] L. Qi, G. Yu, and Y. Xu, Nonnegative diffusion orientation distribution function, *Journal of Mathematical Imaging and Vision*, 2013, 45: 103-113.

[41] A.J. Sommese and W.W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering And Science* World Scientific Pub Co Inc, 2005.

[42] J. Verschelde, Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation, *ACM Trans. Math. Softw.*, 1999, 25:251-276.

[43] A.H. Wright, Finding all solutions to a system of a polynomial equations, *Math. Comp.*, 1985, 44: 125-133.

[44] J. Xie and A. Chang, On the Z-eigenvalues of the signless Laplacian tensor for an even uniform hypergraph, *Numerical Linear Algebra with Applications*, 2013, 20: 1030-1045.

[45] Y. Yang and Q. Yang, Further results for Perron-Frobenius theorem for nonnegative tensors, *SIAM Journal on Matrix Analysis and Applications*, 2010, 31: 2517–2530.

[46] Y. Yang and Q. Yang, Further results for Perron-Frobenius theorem for nonnegative tensors II, *SIAM Journal on Matrix Analysis and Applications*, 2011, 32: 1236–1250.

[47] Z. Zeng and T. Li, NACLab: A Matlab Toolbox for Numerical Algebraic Computation, *ACM Communications in Computer Algebra*, 2013, 47: 170-173.

[48] L. Zhang, L. Qi, and Y. Xu, Finding the largest eigenvalue of an irreducible tensor and linear convergence for weakly positive tensors, *Journal of Computational Mathematics*, 2012, 30: 24–33.

[49] G. Zhou, L. Qi, and S.-Y. Wu, On the largest eigenvalue of a symmetric nonnegative tensor, *Numerical Linear Algebra with Applications*, 2013, 20: 913–928.