Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 17, No. 3, pp. 467–485, 2021

# A QUARTICLY CONVERGENT METHOD FOR EIGENVALUES OF GENERAL TENSORS*

Wei-wei Yang, Hao Liu and Qin Ni†

**Abstract:** In this paper, a quarticly convergent method is proposed for solving a system of nonlinear equations, which is a three-step iterative method. This method is used to find the largest H eigenvalue of irreducible nonnegative tensor and the Z eigenvalues of general tensors, where its computational complexity is slightly greater than Newton method. Due to the particular structure of the problem, the computation of three order tensor and four order tensor are implicit, and a economic computing scheme is given in the algorithm. The global and quartic convergence of the new method are proved. Numerical results indicate that the proposed method is competitive and efficient on some tensor problems.

**Key words:** *quartic convergence, tensor eigenvalue, nonlinear equations*

**Mathematics Subject Classification:** *15A18, 15A69, 65F15*

## 1 Introduction

Let $R$ be the real field, and $m$,$n$ be positive integers. An $m$ order $n$ dimensional tensor $\mathcal{A}$ consists of $n^m$ entries in $R$:

$$\mathcal{A} = (a_{i_1 i_2 \cdots i_m}), \quad a_{i_1 i_2 \cdots i_m} \in R, \quad 1 \le i_1, i_2, \cdots, i_m \le n. \tag{1.1}$$

$\mathcal{A}$ is called nonnegative (or positive) [1] if $a_{i_1 i_2 \cdots i_m} \ge 0$ (or $a_{i_1 i_2 \cdots i_m} > 0$). $\mathcal{A}$ is called symmetric [17] if its entries $a_{i_1 i_2 \cdots i_m}$ are invariant under any permutation of their indices.

$\mathcal{A}_s$ is called semi-symmetric [14] if $a_{i i_2 \cdots i_m} = a_{i \sigma(i_2 \cdots i_m)}$, where $1 \le i \le n$ and $\sigma(i_2 \cdots i_m)$ is any permutation of $i_2 \cdots i_m$.

An $m$ order $n$ dimensional tensor $\mathcal{A}$ is called reducible [1] if there exists a nonempty proper index subset $I \subset \{1, 2, \cdots, n\}$ such that

$$a_{i_1 i_2 \cdots i_m} = 0, \quad \forall i_1 \in I, \forall i_2, \cdots, i_m \notin I.$$

If $\mathcal{A}$ is not reducible, then we call $\mathcal{A}$ irreducible.

Since Qi [17] introduced two kinds of eigenvalues, H-eigenvalue and Z-eigenvalue, for even order real symmetric tensor, the tensor eigenvalue problems have become an important part of numerical multilinear algebra.

*This work was supported by the National Natural Science Foundation of China under Grants 11771210, 61661136001.

†Corresponding author.

© 2021 Yokohama Publishers

A real number $\lambda$ is called an H-eigenvalue of $\mathcal{A}$ if it and a nonzero real vector $x$ are solutions of the following homogeneous polynomial equation:

$$\mathcal{A}x^{m-1} = \lambda x^{[m-1]}, \tag{1.2}$$

where

$$\mathcal{A}x^{m-1} = \left( \sum_{i_2,\cdots,i_m=1}^{n} a_{ii_2\cdots i_m} x_{i_2} \cdots x_{i_m} \right)_{1\le i\le n} \tag{1.3}$$

is an $n$-dimensional column vector,

$$(x^{[m-1]})_i = x_i^{m-1},$$

and $x$ is called an H-eigenvector of $\mathcal{A}$ associated with the H-eigenvalue $\lambda$. A real number $\lambda$ and a real vector $x \in R^n$ are called a Z-eigenvalue of $\mathcal{A}$ and a Z-eigenvector of $\mathcal{A}$ associated with the Z-eigenvalue, respectively, if they satisfy:

$$\mathcal{A}x^{m-1} = \lambda x \quad \text{and} \quad x^{\mathrm{T}}x = 1. \tag{1.4}$$

In the H-eigenvalue setting, Lim [12] and Chang et al. [1, 2] defined irreducible tensors and extended Perron-Frobenius theorem to nonnegative irreducible tensors. After that, scholars begun to study the largest H-eigenvalue of a nonnegative irreducible tensor in depth. Ng et al. [13] proposed the NQZ method for the largest H-eigenvalue of a nonnegative irreducible tensor. Pearson [16] and Chang et al. [3] introduced essential positive tensors and primitive tensors, respectively. The linear or R-linear convergence of the NQZ method was established in [3, 27, 9] under primitivity or weak primitivity. Yang and Yang [22, 23] generalized the weak Perron-Frobenius theorem to general nonnegative tensors. Chen et al. [4] provided inexact power-type methods for computing the largest H-eigenvalue of a general nonnegative tensor. Ni and Qi [14] employed Newton method for finding the largest eigenvalue of a nonnegative homogeneous polynomial map, and obtained the quadratic convergence. They also defined the semi-symmetric tensors, and proved that there is an unique semi-symmetric tensor $\mathcal{A}_s$ for any tensor $\mathcal{A}$.

Much of the properties concerning the Perron-Frobenius theorem fail to hold in the Z-eigenvalue setting. It is NP-hard to compute the extreme Z-eigenvalues of higher order tensors. Qi et al. [18] proposed an elimination method for computing the largest Z-eigenvalue when $(n, m) = (2, 3)$. Kolda et al. [10] proposed a shifted power method(SS-HOPM) for computing a Z-eigenvalue and its associated eigenvector for a symmetric tensor. Zeng and Ni [26] proposed a quasi-Newton method for computing Z-eigenpairs of a symmetric tensor. Based on sequential semidefinite programming, Hu et al. [8] proposed a method for computing the extreme Z-eigenvalues. Hao et al. [6] proposed a sequential subspace projection method for computing extreme Z-eigenvalues.

Chang et al. [2] defined generalized eigenvalue which unified the definitions of H-eigenvalue, Z-eigenvalue in Qi [17], and that of the D-eigenvalue in Qi et al. [19]. Then there are a few methods for solving generalized eigenvalues. Han [7] proposed an unconstrained optimization method for even order symmetric tensors, and obtained superlinearly convergence by using BFGS method. Jacobian SDP relaxations in polynomial optimization was used to compute all real eigenvalues of symmetric tensors in [5]. Kolda et al. [11] improved the shifted power method(SS-HOPM), which can choose the shift automatically, and applied to compute generalized tensor eigenvalues. Based on [6], Yu et al. [25] proposed an adaptive gradient method for computing generalized tensor eigenpairs, which is

linearly convergent. Nearly, Zhang et al. [28] studied the properties of semi-symmetric tensor, and proposed a convergent Newton algorithm for computing Z-eigenvalues of an almost nonnegative irreducible tensor.

When the order and dimension of tensor are large, the convergence of general methods for computing tensor eigenvalue will be slow. Hence, methods with rapid convergence need to be investigated. In this paper, we will propose a quarticly convergent method for computing generalized eigenvalues of general tensors. We will transform the solving problem into nonlinear equations and propose a three-step iterative method to solve it. The associated semi-symmetric tensor of a general tensor can be obtained. Due to the particular structure of tensor eigenvalues, the computation of Jacobian matrices, three order tensors and fourth order tensors are implicit, and a economic computing scheme is given in the algorithm. We will also prove the descent property of the quartic direction. The global and quartic convergence can be established.

## 2 Preliminaries

In this section, we will recall some important definitions and theories. The definition of the generalized eigenvalue can be seen in the following.

**Definition 2.1.** [2] Let $\mathcal{A}$ and $\mathcal{B}$ be two $m$ order $n$ dimensional symmetric tensors on $R^{[m,n]}$. Assume that both $\mathcal{A}x^{m-1}$ and $\mathcal{B}x^{m-1}$ are not identical to zero. We say $(\lambda, x) \in R\times(R^n\backslash\{0\})$ is a generalized eigenpair of $(\mathcal{A}, \mathcal{B})$, if the system of equations

$$(\mathcal{A} - \lambda\mathcal{B})x^{m-1} = 0 \tag{2.1}$$

possesses a solution.

**Remark 2.2.** According to Lemma 2.1 in [28], we know that $\mathcal{A}x^{m-1}$ and $\mathcal{B}x^{m-1}$ are not identical to zero if and only if their associated semi-symmtric tensors $\mathcal{A}_s$ and $\mathcal{B}_s$ are not zero. The computational formula to the associated semi-symmtric tensors $\mathcal{A}_s$ and $\mathcal{B}_s$ can refer to [14] and [28].

Assume that $m$ is even. If $\mathcal{B}$ is an unit tensor $\mathcal{U}$, whose entries are

$$u_{i_1i_2\cdots i_m} = \begin{cases} 1 & \text{if } i_1 = i_2 = \cdots = i_m, \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

then the generalized eigenpair is H-eigenpair.

Let $I_2$ be the $n \times n$ unit matrix. If $\mathcal{B} = I_2^l$, where $l = \frac{m}{2}$, which is the tensor product of $l$ copies of the unit matrices $I_2$, then the formula (2.1) becomes

$$\mathcal{A}x^{m-1} = \lambda(x^{\mathrm{T}}x)x.$$

The generalized eigenpair reduces to Z-eigenpair if $x^{\mathrm{T}}x = 1$.

Chang et al. [1] extended Perron-Frobenius theorem to nonnegative irreducible tensors.

**Theorem 2.3** ([1])**.** *If $\mathcal{A}$ is an irreducible nonnegative tensor of order $m$ and dimension $n$, then there exist $\lambda_0 > 0$ and $x_0 > 0, x_0 \in R$ such that*

$$\mathcal{A}x_0^{m-1} = \lambda_0 x_0^{[m-1]},$$

*Moreover, if $\lambda$ is an eigenvalue with nonnegative eigenvector, then $\lambda = \lambda_0$. If $\lambda$ is an eigenvalue of $\mathcal{A}$, then $|\lambda| \le \lambda_0$.*

Clearly, from this result, $\lambda_0$ is the largest eigenvalue of $\mathcal{A}$.

Based on Theorem 2.3, Ng et al. [13] proposed the NQZ method for computing the largest H-eigenvalue of a nonnegative irreducible tensor.

**Algorithm 2.1** (The NQZ method)**.**

Step 0. Choose $x^{(0)} > 0, x^{(0)} \in R^n$. Let $y^{(0)} = \mathcal{A}(x^{(0)})^{m-1}$ and set $k := 0$.

Step 1. Compute

$$x^{(k+1)} = \frac{(y^{(k)})^{[\frac{1}{m-1}]}}{\|(y^{(k)})^{[\frac{1}{m-1}]}\|},$$

$$y^{(k+1)} = \mathcal{A}(x^{(k+1)})^{m-1},$$

$$\underline{\lambda}_{k+1} = \min_{x_i^{(k+1)}>0} \frac{(y^{(k+1)})_i}{(x_i^{(k+1)})^{m-1}},$$

$$\bar{\lambda}_{k+1} = \max_{x_i^{(k+1)}>0} \frac{(y^{(k+1)})_i}{(x_i^{(k+1)})^{m-1}},$$

Step 2. If $\bar{\lambda}_{k+1} = \underline{\lambda}_{k+1}$, stop. Otherwise, replace $k$ by $k+1$ and go to Step 1.

**Definition 2.4.** [28] A tensor $\mathcal{A}$ of order $m$ and dimension $n$ is called almost nonnegative and irreducible, if its associated semi-symmetric tensor $\mathcal{A}_s$ is nonnegative and irreducible.

Based on this, we can solve the eigenvalue problem of a wider range of tensors, as long as $\mathcal{A}$ is an almost nonnegative and irreducible tensor. For detailed description, it can be seen in [28].

For computing Z-eigenpairs of a symmetric tensor, the classical method is the shifted power method(SS-HOPM). It was provided by Kolda et al. [10] based on S-HOPM, where a suitably modified function is proposed to guarantee the convergence.

**Algorithm 2.2** (The SS-HOPM method)**.**

Step 0. Choose $x_0 \in R^n$ with $\|x_0\| = 1$. Let $\lambda_0 = \mathcal{A}x_0^m$. $\alpha \in R, \varepsilon > 0$. Let $k = 0$.

Step 1. If $\alpha \geq 0$,

$$\hat{x}_{k+1} := \mathcal{A}x_k^{m-1} + \alpha x_k,$$

otherwise

$$\hat{x}_{k+1} := -(\mathcal{A}x_k^{m-1} + \alpha x_k).$$

Step 2.

$$x_{k+1} := \frac{\hat{x}_{k+1}}{\|\hat{x}_{k+1}\|}, \quad \lambda_{k+1} := \mathcal{A}x_{k+1}^m.$$

Step 3. If $|\lambda_{k+1} - \lambda_k| < \varepsilon$, stop, otherwise $k := k+1$, go to Step 1.

## 3 A Quarticly Convergent Algorithm for Computing the Eigenvalues of Tensors

At first, we propose a quarticly convergent algorithm for solving general system of nonlinear equations, then use this algorithm to find the generalized eigenvalues of tensors.

### 3.1 A quarticly convergent algorithm for solving general system of nonlinear equations

In order to solve (2.1), we consider a system of nonlinear equations

$$F(\omega) = 0, \tag{3.1}$$

where $F : R^{n+1} \mapsto R^{n+1}$.

There is a Chebyshev method in [21] for solving (3.1) where the $k$-th iteration is

$$\begin{array}{l} F'(\omega_k)a_k = -F(\omega_k), \\ F'(\omega_k)b_k = -F(\omega_k) - \frac{1}{2}F''(\omega_k)a_k^2, \\ \omega_{k+1} = \omega_k + b_k, \quad k = 0, 1, 2, 3, \cdots\cdots . \end{array} \tag{3.2}$$

This method possesses locally cubic convergence.

Inspiring by this idea, we present a new method with higher order convergence where the $k$-th iteration is computed by the following formula

$$\begin{array}{l} F'(\omega_k)a_k = -F(\omega_k), \\ F'(\omega_k)b_k = -F(\omega_k) - \frac{1}{2}F''(\omega_k)a_k^2, \\ F'(\omega_k)c_k = -F(\omega_k) - \frac{1}{2}F''(\omega_k)b_k^2 - \frac{1}{6}F^{(3)}(\omega_k)b_k^3, \\ \omega_{k+1} = \omega_k + c_k, \quad k = 0, 1, 2, 3, \cdots\cdots . \end{array} \tag{3.3}$$

**Remark 3.1.** (1) This new method is meaningful in theory. In the following section, it is proved that the new method possesses locally quartic convergence. (2) It needs huge computation for the second and third derivatives, and is not suitable for solving general system of nonlinear equations.

However, we can give an economical computing scheme of higher derivative of special system of nonlinear equations generated from tensor eigenvalue problems, and can obtain an quartic convergent algorithm for solving tensor eigenvalue problem with appropriate amount of computation.

### 3.2 Some economical computing scheme of higher derivative in tensor eigenvalue problems

In this paper, we consider the general tensors $\mathcal{A}$. Its associated semi-symmetric tensor $\mathcal{A}_s$ can be computed by Lemma 2.1 in [14]. For convenience, we use $\mathcal{A}$ to denote semi-symmetric tensor $\mathcal{A}_s$ in the following. Now we transform (2.1) into a system of nonlinear equations

$$F(\omega) = \begin{pmatrix} (\mathcal{A} - \lambda\mathcal{B})x^{m-1} \\ \frac{1}{m}(1 - \mathcal{B}x^m) \end{pmatrix}, \tag{3.4}$$

where $\omega = (x, \lambda)$, and $\mathcal{A}$ is a semi-symmetric tensor.

By taking the derivative of $F(\omega)$, we can easily obtain the Jacobian matrix $J(\omega)$ as tensors $\mathcal{A}_s, \mathcal{B}$ are at least semi-symmetric,

$$F'(\omega) = \begin{pmatrix} (m-1)(\mathcal{A} - \lambda\mathcal{B})x^{m-2} & -\mathcal{B}x^{m-1} \\ -(\mathcal{B}x^{m-1})^T & 0 \end{pmatrix}, \tag{3.5}$$

where

$$(\mathcal{A}x^{m-2})_{ij} = \sum_{i_3,\cdots,i_m=1}^{n} a_{iji_3\cdots i_m} x_{i_3} \cdots x_{i_m}, i, j = 1, \cdots, n. \tag{3.6}$$

In the derivative process, we use the following formula

$$\frac{\partial(\mathcal{A}x^d)_i}{\partial x_j} = d(\mathcal{A}x^{d-1})_{ij}, \tag{3.7}$$

where $1 \leq d \leq m$, which can be seen in [14]. Then we can also compute $F''(\omega)$ and $F'''(\omega)$. For any $n+1$ dimensional vector $y = (y_1, y_2, \cdots, y_n, y_{n+1})^T = (\bar{y}^T, y_{n+1})^T$, where $\bar{y} = (y_1, y_2, \cdots, y_n)^T$ is for convenience we have

$$F''(\omega)y^2 = (m-1)\begin{pmatrix} [(m-2)Q(x,\lambda,\bar{y}) - y_{n+1}P(x,\bar{y})]x \\ -x^T P(x,\bar{y})\bar{y} \end{pmatrix}, \tag{3.8}$$

$$F^{(3)}(\omega)y^3 = (m-1)(m-2)\begin{pmatrix} [(m-3)Q(x,\lambda,\bar{y}) - 2y_{n+1}P(x,\bar{y})]\bar{y} \\ -\bar{y}^T P(x,\bar{y})\bar{y} \end{pmatrix}, \tag{3.9}$$

where $Q(x,\lambda,\bar{y}) = (\mathcal{A} - \lambda\mathcal{B})x^{m-4}\bar{y}^2$, $P(x,\bar{y}) = \mathcal{B}x^{m-3}\bar{y}$,

$$(\mathcal{A}x^{m-4}\bar{y}^2)_{ij} = \sum_{i_3,\cdots,i_m=1}^{n} a_{iji_3\cdots i_m} x_{i_3}\cdots x_{i_{m-2}}\bar{y}_{i_{m-1}}\bar{y}_{i_m}, i, j = 1, 2, \cdots, n, \tag{3.10}$$

The formulas (3.8)-(3.9) hold when tensor $\mathcal{A}$ is semi-symmetric, because

$$\mathcal{A}x^{m-4}\bar{y}^2 x = \mathcal{A}x^{m-3}\bar{y}^2.$$

From (3.8)-(3.9), it is easy to see that the main computation focuses on $(\mathcal{A}-\lambda\mathcal{B})x^{m-4}$, $(\mathcal{A}-\lambda\mathcal{B})x^{m-3}$, $(\mathcal{A}-\lambda\mathcal{B})x^{m-2}$ and $(\mathcal{A}-\lambda\mathcal{B})x^{m-1}$. By the idea in [20], we first compute $(\mathcal{A}-\lambda\mathcal{B})x^{m-4}$, and then compute

$$(\mathcal{A}-\lambda\mathcal{B})x^{m-3} = (\mathcal{A}-\lambda\mathcal{B})x^{m-4}\cdot x,$$
$$(\mathcal{A}-\lambda\mathcal{B})x^{m-2} = (\mathcal{A}-\lambda\mathcal{B})x^{m-3}\cdot x,$$
$$(\mathcal{A}-\lambda\mathcal{B})x^{m-1} = (\mathcal{A}-\lambda\mathcal{B})x^{m-2}\cdot x,$$

which is an economic computational scheme.

We use the nested scheme as above to multiply tensors by vectors. Let $\mathcal{A}$ be an $m$ order $n$ dimensional tensor, and $\mathcal{A}x^k$ is an $m-k$ order $n$ dimensional tensor. There are

$$u_k = \begin{pmatrix} m-k+n-1 \\ n-1 \end{pmatrix}$$

different elements in $\mathcal{A}x^k$. Suppose that $\mathcal{A}x^{k-1}$ is known, $\mathcal{A}x^k = \mathcal{A}x^{k-1}\cdot x$, each element of $\mathcal{A}x^k$ needs $n$ multiplications. Then we need $u_k \cdot n$ multiplications to get $\mathcal{A}x^k$. So the total multiplications to get $\mathcal{A}x^k$ are $\sum_{i=1}^{k} u_i \cdot n$. Therefore, by the formulas (3.4) and (3.5), we can compute the multiplications to get a Newton direction. It needs $O((n+1)^3)$ multiplications to solve a linear equations by LU-decomposition. And from Table 1, a Newton direction needs $(p_1+p_2)\cdot n + O((n+1)^3)$ multiplications, where

$$p_1 = \sum_{k=1}^{m-1} u_k, \quad p_2 = \sum_{k=1}^{m} u_k.$$

By the formulas (3.8), we can compute the multiplications of a Chebyshev direction. From Table 2, there are $(p_1+p_2)\cdot n$ multiplications to get $F''(\omega_k)a_k^2$. From Table 3, a

Table 1: The main multiplications of a Newton direction

| | $F'(\omega_k)$ | | $F(\omega_k)$ | |
|---|---|---|---|---|
| | $(\mathcal{A}-\lambda\mathcal{B})x_k^{m-2}$ | $\mathcal{B}x_k^{m-1}$ | $(\mathcal{A}-\lambda\mathcal{B})x_k^{m-1}$ | $\mathcal{B}x_k^m$ |
| multiplications | $\sum_{k=1}^{m-2} u_k \cdot n$ | $\sum_{k=1}^{m-1} u_k \cdot n$ | $u_{m-1} \cdot n$ | $u_m \cdot n$ |

Table 2: The multiplications of $F''(\omega_k)a_k^2$

| $(\mathcal{A}-\lambda\mathcal{B})x_k^{m-4}$ | $\mathcal{B}x_k^{m-3}$ | $(\mathcal{A}-\lambda\mathcal{B})x_k^{m-4}a_k^2$ | $\mathcal{B}x_k^{m-3}a_k$ | Others |
|---|---|---|---|---|
| $\sum_{k=1}^{m-4} u_k \cdot n$ | $\sum_{k=1}^{m-3} u_k \cdot n$ | $(u_{m-3}+u_{m-2}) \cdot n$ | $u_{m-2} \cdot n$ | $2n^2+n$ |

Chebyshev direction needs $(p_1+p_2+p_3)\cdot n + O((n+1)^3)$ multiplications, where

$$p_3 = \sum_{k=m-3}^{m-1} u_k + \sum_{k=m-2}^{m} u_k.$$

By the formulas (3.9), we can compute the multiplications of a Quartic direction. From Table 4, a Quartic direction needs $(p_1+p_2+2p_3)\cdot n + n^2 + n + O((n+1)^3)$ multiplications.

**Remark 3.2.** (1) When tensor $\mathcal{B}$ is a special tensor, such as an unit tensor, the computation of $\mathcal{B}x^{m-3}$ is simple. (2) For linear equations, the main computation is LU-decomposition of $F'(\omega_k)$. Hence, the difference between the computation of three directions is very small. (3) When $m = 4$,

$$p_1 = p_2 = p_3 = \alpha(n) + o(\alpha(n)),$$

where $\alpha(n) = n(n+1)(n+2)/6$. The calculation of a Quartic direction is $4/3$ times as Chebyshev direction, and twice as Newton direction. When $m > 4$, the principal term of multiplications is include in $p_1$,$p_2$, then the calculation of the three directions is almost the same.

### 3.3 Quarticly convergent algorithm for tensor eigenvalue problem

In order to guarantee the descent property, we deduce a sufficient descent condition of Quartic direction.

Define the merit function

$$f(\omega) = \frac{1}{2}\|F(\omega)\|^2,$$

then

$$\nabla f(\omega) = F'(\omega)^{\mathrm{T}} F(\omega).$$

**Lemma 3.3.** *Let $\omega_k$ be a current iterate point, $F(\omega_k) \neq 0$, and $a_k$ be a Newton direction which satisfies*

$$F'(\omega_k)a_k = -F(\omega_k),$$

*and $b_k$ satisfies*

$$F'(\omega_k)b_k = -F(\omega_k) - \frac{1}{2}F''(\omega_k)a_k^2.$$

Table 3: The main multiplications of a Chebyshev direction

| $F''(\omega_k)a_k^2$ | $F'(\omega_k)$ | | $F(\omega_k)$ | |
|---|---|---|---|---|
| | $(\mathcal{A}-\lambda\mathcal{B})x_k^{m-2}$ | $\mathcal{B}x_k^{m-1}$ | $(\mathcal{A}-\lambda\mathcal{B})x_k^{m-1}$ | $\mathcal{B}x_k^m$ |
| $(p_1+p_2)\cdot n$ | $(u_{m-3}+u_{m-2})\cdot n$ | $(u_{m-2}+u_{m-1})\cdot n$ | $u_{m-1}\cdot n$ | $u_m\cdot n$ |

Table 4: The main multiplications of a Quartic direction

| $F^{(3)}(\omega_k)b_k^3$ | $F''(\omega_k)b_k^2$ | $F''(\omega_k)b_k^2$ | $F'(\omega_k)$ | $F(\omega_k)$ |
|---|---|---|---|---|
| $(p_1+p_2)\cdot n$ | $n^2+n$ | $p_3\cdot n$ | $p_3\cdot n$ | |

*If*

$$-F(\omega_k)^T F''(\omega_k)b_k^2-\frac{1}{3}F(\omega_k)^T F^{(3)}(\omega_k)b_k^3\le 2\gamma\|F(\omega_k)\|^2 \tag{3.11}$$

*holds, where $\gamma\in(0,1)$, then Quartic direction $c_k$ defined by (3.3) is a descent direction.*

*Proof.* From (3.3), we have that Quartic direction $c_k$ satisfies

$$F'(\omega_k)c_k=-F(\omega_k)-\frac{1}{2}F''(\omega_k)b_k^2-\frac{1}{6}F^{(3)}(\omega_k)b_k^3,$$

and

$$\begin{aligned}\nabla f(\omega_k)^{\mathrm{T}}c_k &= F(\omega_k)^{\mathrm{T}}F'(\omega_k)c_k\\ &= -F(\omega_k)^{\mathrm{T}}F(\omega_k)-\tfrac{1}{2}F(\omega_k)^{\mathrm{T}}F''(\omega_k)b_k^2-\tfrac{1}{6}F(\omega_k)^{\mathrm{T}}F^{(3)}(\omega_k)b_k^3\\ &\le(\gamma-1)\|F(\omega_k)\|^2<0,\end{aligned}$$

where the first inequality is obtained by (3.11). Then Quartic direction defined by (3.3) is a descent direction. □

**Algorithm 3.1** (Quarticly convergent algorithm)**.**

Step 0. Choose starting guess $\omega_0\in R^{n+1}$, and parameters $\gamma\in(0,1)$, $\epsilon>0$. Set $k:=0$, Flag=0.

Step 1. If $\|F(\omega_k)\|<\epsilon$, then stop.

Step 2. Compute an LU-decomposition of $F'(\omega_k)$ by Gauss elimination.

Step 3. Solve the linear systems

$$F'(\omega_k)a_k=-F(\omega_k),$$

$$F'(\omega_k)b_k=-F(\omega_k)-\frac{1}{2}F''(\omega_k)a_k^2.$$

Step 4. If (3.11) is satisfied, solve the linear system

$$F'(\omega_k)c_k=-F(\omega_k)-\frac{1}{2}F''(\omega_k)b_k^2-\frac{1}{6}F^{(3)}(\omega_k)b_k^3,$$

set $d_k=c_k$, go to Step 5; Otherwise, set $d_k=a_k$, go to step 5.

Step 5. Determine $\alpha_k$ satisfying the Wolfe conditions

$$\begin{array}{c} f(\omega_k + \alpha_k d_k) \le f(\omega_k) + c_1 \alpha_k \nabla f(\omega_k)^T d_k, \\ |\nabla f(\omega_k + \alpha_k d_k)^T d_k| \le -c_2 \nabla f(\omega_k)^T d_k, \end{array} \tag{3.12}$$

where $0 < c_1 < c_2 < 0.5$.

Step 6. $\omega_{k+1} = \omega_k + \alpha_k d_k$. Set $k := k + 1$, go to Step 1.

**Remark 3.4.** In Steps 3-4, the coefficient matrices of the three line systems are the same. In Step 4, the line system which is used for solving $d_k$ could have other forms, which do not change the convergence result. For example, $F''(\omega_k)b_k^2$ can be replaced by $F''(\omega_k)a_k b_k$, and $F^{(3)}(\omega_k)b_k^3$ can also be replaced by $F^{(3)}(\omega_k)a_k^3$, $F^{(3)}(\omega_k)a_k^2 b_k$, or $F^{(3)}(\omega_k)a_k b_k^2$.

**Remark 3.5.** If $\mathcal{B}$ is an unit tensor $\mathcal{U}$, then from Theorem 2.3 and Definition 2.4, as long as we keep $x_k$ is positive, we can find the largest H-eigenvalue by Algorithm 3.1 for an almost irreducible nonnegative tensor $\mathcal{A}$. The equations become

$$F(x, \lambda) = \begin{pmatrix} \mathcal{A}x^{m-1} + (\sigma - \lambda)x^{[m-1]} \\ \frac{1}{m}(1 - \sum_{i=1}^{n} x_i^m) \end{pmatrix},$$

where $\sigma$ is a positive number. Then from Lemma 3.2 in [14], $\lambda - \sigma$ is the largest H-eigenvalue of $\mathcal{A}$. Given a positive initial vector $\omega_0$, the step length $\alpha_k$ satisfies both (3.12) and

$$\omega_k + \alpha_k d_k \in R_{++}^{n+1}, \tag{3.13}$$

then $\omega_k$ must be positive.

**Remark 3.6.** If $m$ is even, $\mathcal{B} = I_2^{\frac{m}{2}}$, where $I_2$ is the $n \times n$ unit matrix, the equation becomes

$$F(x, \lambda) = \begin{pmatrix} \mathcal{A}x^{m-1} - \lambda x^{[m-1]} \\ \frac{1}{2}(1 - x^{\mathrm{T}}x) \end{pmatrix},$$

then for different initial values we can find different Z-eigenpairs by Algorithm 3.1.

## 4 Convergence Analysis

**Lemma 4.1.** *Let $\omega_0 \in R^{n+1}$, and the level set $\mathcal{L} = \{\omega \in R^{n+1} | \|F(\omega)\| \le \|F(\omega_0)\|\}$. If $\mathcal{B} = \mathcal{U}$, or $\mathcal{B} = I_2^{\frac{m}{2}}$, then $\mathcal{L}$ is bounded.*

*Proof.* The proof can refer to Lemma 4.1 in [24]. □

**Lemma 4.2.** *Suppose that $\mathcal{N}$ is some open neighborhood of $\mathcal{L}$. Then there exists a constant $L > 0$ such that*

$$\|F(\omega_1) - F(\omega_2)\| \le L\|\omega_1 - \omega_2\|, \forall \omega_1, \omega_2 \in R^{n+1}. \tag{4.1}$$

*Proof.* From the boundness of $\omega$ and Lemma 4.2 in [14], the conclusion can be proved. □

From the formula (3.1) we see that $F(\omega)$ is at least twice continuously differentiable for $m \ge 3$. Since $\{f(\omega_k)\}$ is decreasing, it is clear that the sequence $\{\omega_k\}$ generated by Algorithm 3.1 is contained in $\mathcal{L}$ and hence is bounded. In addition, it is easy to see that there is a constant $\gamma_1 > 0$ such that

$$\|F(\omega)\| \le \gamma_1, \forall \omega \in \mathcal{N}. \tag{4.2}$$

**Theorem 4.3.** *Assume that $\omega^* \in R^{n+1}$ satisfies $F(\omega^*) = 0$ and $F'(\omega^*)$ is nonsingular. Let $\{\omega_k\}$ be generated by Algorithm 3.1. Then*

$$\liminf_{k\to\infty} \|F_k\| = 0. \tag{4.3}$$

*Proof.* From Theorem 3.2 in [15] and Lemmas 3.3, 4.1 and 4.2 we have

$$\lim_{k\to\infty} \nabla f(\omega_k) = 0,$$

where $\nabla f(\omega_k) = F'(\omega_k)^{\mathrm{T}} F(\omega_k)$. Hence, $\{\omega_k\}$ is convergent. As $F'(\omega^*)$ is nonsingular [14], so (4.3) holds. □

**Theorem 4.4.** *Let $\omega^* \in \mathcal{L}$ be a solution of $F(\omega) = 0$, where $F'(\omega^*)$ is nonsingular, and $\{\omega_k\}$ be the convergent sequence of iterates generated by Algorithm 3.1. Assume that $d_k$ is Quartic direction, $\alpha_k = 1$ are obtained for $k \geq k_0$ where $k_0$ is sufficiently great. Then we have*

$$\|\omega_{k+1} - \omega^*\| = O(\|\omega_k - \omega^*\|^4). \tag{4.4}$$

*Proof.* If $\omega$ belongs to some sufficient small neighbourbood of $\omega^*$, then $F'(\omega)$ is nonsingular, and there are positive constants $B_2$ and $B_3$ such that

$$\|F'(\omega)^{-1}\| \leq B_2, \|F''(\omega)\| \leq B_3. \tag{4.5}$$

Let $G_k = F'(\omega_k)$. If Quartic direction $d_k$ in (3.3) is chosen, then we have

$$G_k d_k = -F_k - \frac{1}{2}F''(\omega_k)b_k^2 - \frac{1}{6}F^{(3)}(\omega_k)b_k^3,$$

where $b_k = -G_k^{-1}(F_k + \frac{1}{2}F''(\omega_k)a_k^2)$, $a_k = -G_k^{-1}F_k$, and

$$\begin{aligned}
G_k(\omega_{k+1} - \omega^*) &= G_k(\omega_k + d_k - \omega^*) \\
&= G_k(\omega_k - \omega^*) + G_k d_k \\
&= -F_k - G_k(\omega^* - \omega_k) - \frac{1}{2}F''(\omega_k)b_k^2 - \frac{1}{6}F^{(3)}(\omega_k)b_k^3 \\
&= -F_k - G_k(\omega^* - \omega_k) - \frac{1}{2}F''(\omega_k)(\omega^* - \omega_k)^2 - \frac{1}{6}F^{(3)}(\omega_k)(\omega^* - \omega_k)^3 \\
&\quad + \frac{1}{2}F''(\omega_k)(\omega^* - \omega_k)(\omega^* - \omega_k - b_k) + \frac{1}{2}F''(\omega_k)(\omega^* - \omega_k - b_k)b_k \\
&\quad + \frac{1}{6}F^{(3)}(\omega_k)(\omega^* - \omega_k - b_k)(\omega^* - \omega_k)^2 + \frac{1}{6}F^{(3)}(\omega_k)(\omega^* - \omega_k - b_k)b_k^2 \\
&\quad - \frac{1}{6}F^{(3)}(\omega_k)(\omega^* - \omega_k - b_k)(\omega^* - \omega_k)b_k.
\end{aligned}$$

By doing a Taylor series expansion at $\omega^*$, and combining it with $F(\omega^*) = 0$, we have

$$F_k + G_k(\omega^* - \omega_k) = O(\|\omega^* - \omega_k\|^2),$$

$$F_k + G_k(\omega^* - \omega_k) + \frac{1}{2}F''(\omega_k)(\omega^* - \omega_k)^2 = O(\|\omega^* - \omega_k\|^3),$$

$$F_k + G_k(\omega^* - \omega_k) + \frac{1}{2}F''(\omega_k)(\omega^* - \omega_k)^2 + \frac{1}{6}F^{(3)}(\omega_k)(\omega^* - \omega_k)^3 = O(\|\omega^* - \omega_k\|^4). \tag{4.6}$$

Then by (4.5), we have

$$\begin{aligned}\|\omega^* - \omega_k - a_k\| &= \|\omega^* - \omega_k + G_k^{-1}F_k\| \\ &= \|G_k^{-1}(F_k + G_k(\omega^* - \omega_k))\| \\ &\le \|G_k^{-1}\|\|F_k + G_k(\omega^* - \omega_k)\| \\ &= O(\|\omega_k - \omega^*\|^2),\end{aligned} \tag{4.7}$$

$$\begin{aligned}\|a_k\| = \|G_k^{-1}F_k\| &= \|G_k^{-1}(F_k - F^*)\| \\ &= \|G_k^{-1}G(\omega_k + t\omega^*)(\omega_k - \omega^*)\| \\ &= O(\|\omega_k - \omega^*\|),\end{aligned} \tag{4.8}$$

$$\begin{aligned}\|\omega^* - \omega_k - b_k\| &= \|\omega^* - \omega_k + G_k^{-1}(F_k + \frac{1}{2}F''(\omega_k)a_k^2)\| \\ &= \|G_k^{-1}(F_k + G_k(\omega^* - \omega_k) + \frac{1}{2}F''(\omega_k)(\omega^* - \omega_k)^2 \\ &\quad - \frac{1}{2}F''(\omega_k)(\omega^* - \omega_k)(\omega^* - \omega_k - a_k) - \frac{1}{2}F''(\omega_k)(\omega^* - \omega_k - a_k)a_k)\| \\ &\le \|G_k^{-1}\|\left[\|F_k + G_k(\omega^* - \omega_k) + \frac{1}{2}F''(\omega_k)(\omega^* - \omega_k)^2\|\right. \\ &\quad \left. + \frac{1}{2}\|F''(\omega_k)(\omega^* - \omega_k)(\omega^* - \omega_k - a_k)\| + \frac{1}{2}\|F''(\omega_k)(\omega^* - \omega_k - a_k)a_k\|\right] \\ &= O(\|\omega_k - \omega^*\|^3),\end{aligned} \tag{4.9}$$

and

$$\begin{aligned}\|b_k\| &= \|G_k^{-1}(F_k + \frac{1}{2}F''(\omega_k)a_k^2)\| \\ &= \|G_k^{-1}(F_k + G_k(\omega_k - \omega^*) - G_k(\omega_k - \omega^*) + \frac{1}{2}F''(\omega_k)a_k^2)\| \\ &\le \|G_k^{-1}\|\left[\|F_k + G_k(\omega_k - \omega^*)\| + \|G_k(\omega_k - \omega^*)\| + \|\frac{1}{2}F''(\omega_k)a_k^2)\|\right] \\ &= O(\|\omega_k - \omega^*\|).\end{aligned} \tag{4.10}$$

Then by (4.6), (4.9) and (4.10), we have

$$\begin{aligned}\|\omega_{k+1} - \omega^*\| &= \|G_k^{-1}G_k(\omega_{k+1} - \omega^*)\| \\ &\le \|G_k^{-1}\|\|G_k(\omega_{k+1} - \omega^*)\| \\ &\le \|G_k^{-1}\|\left[\|F_k + G_k(\omega^* - \omega_k) + \frac{1}{2}F''(\omega_k)(\omega^* - \omega_k)^2 + \frac{1}{6}F^{(3)}(\omega_k)(\omega^* - \omega_k)^3\|\right. \\ &\quad + \frac{1}{2}\|F''(\omega_k)(\omega^* - \omega_k)(\omega^* - \omega_k - b_k)\| + \frac{1}{2}\|F''(\omega_k)(\omega^* - \omega_k - b_k)b_k\| \\ &\quad + \frac{1}{6}\|F^{(3)}(\omega_k)(\omega^* - \omega_k - b_k)(\omega^* - \omega_k)^2\| + \frac{1}{6}\|F^{(3)}(\omega_k)(\omega^* - \omega_k - b_k)b_k^2\| \\ &\quad \left. + \frac{1}{6}\|F^{(3)}(\omega_k)(\omega^* - \omega_k - b_k)(\omega^* - \omega_k)b_k\|\right] \\ &= O(\|\omega_k - \omega^*\|^4).\end{aligned}$$

□

## 5 Numerical Results

In this section, we present some preliminary numerical tests. All tests are implemented by using Matlab R2014b on a PC with CPU 2.00 GHz and 8.00 GB RAM. Firstly, we set $\mathcal{B} = \mathcal{U}$ where $\mathcal{U}$ is an unit tensor, and use Algorithm 3.1 to compute the largest H eigenvalues of some nonnegative irreducible tensors.

The test problems TPI and TPII are from [14], where $P(x) = \mathcal{A}x^{m-1}$, $m$ is even.

TPI. $P(x) = \mathcal{A}x^{m-1}$ is defined as

$$P(x) = \begin{pmatrix} (a_1+\gamma)x_1^{m-1} + 0.5(\sum\limits_{i=1}^{n} b_i x_i \sum\limits_{i=1}^{n} c_i x_i)^{\frac{m}{2}-1}(c_1 \sum\limits_{i=1}^{n} b_i x_i + b_1 \sum\limits_{i=1}^{n} c_i x_i) \\ \cdots\cdots \\ (a_j+\gamma)x_j^{m-1} + 0.5(\sum\limits_{i=1}^{n} b_i x_i \sum\limits_{i=1}^{n} c_i x_i)^{\frac{m}{2}-1}(c_j \sum\limits_{i=1}^{n} b_i x_i + b_j \sum\limits_{i=1}^{n} c_i x_i) \\ \cdots\cdots \\ (a_n+\gamma)x_n^{m-1} + 0.5(\sum\limits_{i=1}^{n} b_i x_i \sum\limits_{i=1}^{n} c_i x_i)^{\frac{m}{2}-1}(c_n \sum\limits_{i=1}^{n} b_i x_i + b_n \sum\limits_{i=1}^{n} c_i x_i) \end{pmatrix},$$

where $a_i, b_i, c_i (i = 1, \cdots, n)$ are random number in [0,1], $\gamma > 0$ is a parameter.

TPII. $P(x) = \mathcal{A}x^{m-1}$ is defined as

$$P(x) = \begin{pmatrix} (a_1+\gamma)x_1^{m-1} + 0.5(\sum\limits_{i=1}^{n} b_i x_i)^{\frac{m}{2}} x_1^{\frac{m}{2}-1} + 0.5(\sum\limits_{i=1}^{n} x_i^{\frac{m}{2}})(\sum\limits_{i=1}^{n} b_i x_i)^{\frac{m}{2}-1} b_1 \\ \cdots\cdots \\ (a_j+\gamma)x_j^{m-1} + 0.5(\sum\limits_{i=1}^{n} b_i x_i)^{\frac{m}{2}} x_j^{\frac{m}{2}-1} + 0.5(\sum\limits_{i=1}^{n} x_i^{\frac{m}{2}})(\sum\limits_{i=1}^{n} b_i x_i)^{\frac{m}{2}-1} b_j \\ \cdots\cdots \\ (a_n+\gamma)x_n^{m-1} + 0.5(\sum\limits_{i=1}^{n} b_i x_n)^{\frac{m}{2}} x_n^{\frac{m}{2}-1} + 0.5(\sum\limits_{i=1}^{n} x_i^{\frac{m}{2}})(\sum\limits_{i=1}^{n} b_i x_i)^{\frac{m}{2}-1} b_n \end{pmatrix},$$

where $a_i, b_i (i = 1, \cdots, n)$ are random number in [0,1], $\gamma > 0$ is a parameter.

The numerical results of Algorithm 3.1, Chebyshev's method and Newton method are listed in Table 5 and Table 6. The numerical results of Algorithm 3.1 and Algorithm 2.1 (the power algorithm) can be seen in Table 8 and Table 9. In Tables 5-9, "m", "n" are the number of order and dimension in tensor, "Iter" is the number of iterations, "Term" is the last value of $|\lambda_k - \lambda_{k+1}|$, "Cpu" is cpu time in seconds. In problems TPI and TPII, $\gamma > 0$ is chosen from 10 to $10^7$, in problem TPIII, $\sigma = 10$. In order to compare these algorithms, we choose the same initial point $x_0 = (1, \cdots, 1)^{\mathrm{T}}$, and the same termination conditions

$$|\lambda_k - \lambda_{k+1}| \le 10^{-5}, \| P(x_{k+1}) - \lambda_{k+1} x_{k+1}^{[m-1]} \| \le 10^{-5}.$$

If the termination conditions are not satisfied in 500 iteration, then the algorithm stops.

From Tables 5 and 6, we can see that the number of iteration of Algorithm 3.1 is the least, and the time of Algorithm 3.1 is a little less then the other two methods. For TPII, the precision of Algorithm 3.1 and Chebyshev's method is higher than Newton method.

For TPI, there is only one Quartic direction which is not descent and is in the initial iteration, and the search direction is then chosen as Newton direction. For TPII, when $m = 4$, both Quartic directions and Chebyshev directions are descent in most cases; for the other cases, these directions are not descent in the last few iterations, and the search direction is then chosen as Newton direction. These situations can be seen in Table 7.

Table 5: Numerical results of TPI

| TPI | Algorithm 3.1 | | | Chebyshev's method | | | Newton method | | |
|---|---|---|---|---|---|---|---|---|---|
| m/n/$\gamma$ | Iter | Term | Cpu | Iter | Term | Cpu | Iter | Term | Cpu |
| 4/20/10 | 6 | 1.0759e-06 | 0.0065 | 8 | 1.9280e-07 | 0.0077 | 9 | 1.1087e-06 | 0.0069 |
| $4/20/10^2$ | 6 | 1.6578e-06 | 0.0088 | 8 | 2.0524e-07 | 0.0140 | 9 | 2.0911e-06 | 0.0075 |
| $4/20/10^3$ | 6 | 3.4022e-06 | 0.0072 | 8 | 3.9859e-07 | 0.0104 | 9 | 4.1517e-06 | 0.0074 |
| $4/20/10^4$ | 6 | 1.5409e-06 | 0.0153 | 8 | 1.4950e-07 | 0.0188 | 9 | 1.8029e-06 | 0.0214 |
| $4/20/10^5$ | 6 | 3.7059e-06 | 0.0412 | 8 | 1.7581e-07 | 0.0456 | 9 | 3.0815e-06 | 0.0409 |
| $4/60/10^2$ | 8 | 5.6504e-07 | 0.0515 | 9 | 6.3164e-06 | 0.0655 | 12 | 2.0044e-07 | 0.0616 |
| $4/60/10^3$ | 8 | 1.6306e-07 | 0.0287 | 9 | 2.2150e-06 | 0.0315 | 12 | 1.3420e-07 | 0.0234 |
| $4/60/10^4$ | 8 | 2.9761e-07 | 0.0263 | 9 | 3.7188e-06 | 0.0281 | 12 | 2.0190e-07 | 0.0256 |
| $4/60/10^5$ | 8 | 7.4494e-07 | 0.0438 | 9 | 5.7501e-06 | 0.0479 | 12 | 3.7400e-07 | 0.0474 |
| $4/60/10^6$ | 8 | 2.3406e-07 | 0.1168 | 9 | 2.7370e-06 | 0.1231 | 12 | 1.5730e-07 | 0.1234 |
| $4/100/10^3$ | 9 | 1.4769e-07 | 0.0546 | 10 | 1.8423e-06 | 0.0592 | 13 | 2.2513e-06 | 0.0528 |
| $4/100/10^4$ | 9 | 1.5529e-07 | 0.0522 | 10 | 1.2401e-06 | 0.0623 | 13 | 2.2390e-06 | 0.0562 |
| $4/100/10^5$ | 9 | 1.1736e-07 | 0.0559 | 10 | 9.0957e-07 | 0.0671 | 13 | 1.4478e-06 | 0.0578 |
| $4/100/10^6$ | 9 | 4.1768e-07 | 0.0933 | 10 | 1.0710e-06 | 0.1083 | 13 | 5.3829e-06 | 0.0961 |
| $4/100/10^7$ | 9 | 8.6407e-06 | 0.2128 | 11 | 4.0749e-06 | 0.2233 | 14 | 3.1937e-07 | 0.2211 |
| $6/20/10^3$ | 16 | 7.4161e-07 | 0.0624 | 17 | 5.4144e-06 | 0.0666 | 24 | 8.5985e-08 | 0.0605 |
| $6/20/10^4$ | 16 | 9.3618e-07 | 0.0162 | 17 | 6.2063e-06 | 0.0192 | 24 | 1.0667e-07 | 0.0221 |
| $6/20/10^5$ | 16 | 1.6598e-06 | 0.0686 | 17 | 6.5616e-06 | 0.0646 | 24 | 1.4238e-07 | 0.0636 |
| $6/20/10^6$ | 16 | 3.5723e-06 | 0.0703 | 17 | 1.13088-07 | 0.0692 | 24 | 2.9486e-07 | 0.0710 |
| $6/20/10^7$ | 17 | 5.3842e-07 | 0.1176 | 18 | 6.9337e-06 | 0.1207 | 25 | 1.2273e-06 | 0.1465 |

From Tables 8 and 9 we can see that Algorithm 3.1 performs more stably even though $\gamma$ changes a lot, while Algorithm 2.1 present the opposite case. When $\gamma$ is relative small, Algorithm 2.1 use less time then Algorithm 3.1 , While $\gamma$ increases, Algorithm 3.1 needs the shorter time. The number of iteration of Algorithm 2.1 is much more then Algorithm 3.1.

Secondly, we set $\mathcal{B} = I_2^{\frac{m}{2}}$ where $m$ is even, and use Algorithm 3.1 to compute Z-eigenpair. The test problems are from [5]. We compare the performance of Algorithm 3.1 with Algorithm 2.2.

TPIII. Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[4,3]}$ where

$$\begin{array}{llllllll} a_{1111} = 0.2883, & a_{1112} = -0.0031, & a_{1113} = 0.1973, & a_{1122} = -0.2485, \\ a_{1123} = -0.2939, & a_{1133} = 0.3847, & a_{1222} = 0.2972, & a_{1223} = 0.1862, \\ a_{1133} = 0.0919, & a_{1333} = -0.3619, & a_{2222} = 0.1241, & a_{2223} = -0.3420, \\ a_{2233} = 0.2127, & a_{2333} = 0.2727, & a_{3333} = -0.3054, & \end{array}$$

and the values of other elements can be obtained from symmetry.

We choose an $n+1$-dimensional vector whose elements are uniformly distributed in $(-1, 1)$ randomly as the initial point for TPIII, and perform the two algorithms 50 times, respectively. The numerical results can be seen in Tables 10 and 11, where "Iter" is the average number of iterations in 50 times, "CPU" is the average cpu time in 50 times in seconds, "Occ" is the occurrence probability in the 50 experiments for every eigenvalue. There are a pair of opposite eigenvectors corresponding to each eigenvalue, we only record one of them. In Algorithm 2.2, we set $\alpha = 2$.

Table 6: Numerical results of TPII

| TPII | Algorithm 3.1 | | | Chebyshev's method | | | Newton method | | |
|---|---|---|---|---|---|---|---|---|---|
| m/n/$\gamma$ | Iter | Term | Cpu | Iter | Term | Cpu | Iter | Term | Cpu |
| 4/20/10 | 4 | 1.4863e-10 | 0.0077 | 5 | 4.3027e-13 | 0.0112 | 6 | 3.2707e-06 | 0.0097 |
| $4/20/10^2$ | 4 | 2.7682e-10 | 0.0147 | 5 | 4.1652e-13 | 0.0160 | 6 | 1.9712e-06 | 0.0170 |
| $4/20/10^3$ | 4 | 2.1600e-10 | 0.0142 | 5 | 3.7933e-13 | 0.0187 | 6 | 2.7310e-06 | 0.0177 |
| $4/20/10^4$ | 4 | 4.2058e-10 | 0.0080 | 5 | 1.2205e-12 | 0.0098 | 6 | 4.5939e-06 | 0.0094 |
| $4/100/10^2$ | 5 | 6.7819e-13 | 0.0763 | 6 | 1.3632e-12 | 0.0963 | 8 | 5.5806e-09 | 0.0879 |
| $4/100/10^3$ | 5 | 7.8076e-13 | 0.0805 | 6 | 1.4120e-12 | 0.0996 | 8 | 5.4337e-09 | 0.1197 |
| $4/100/10^4$ | 5 | 7.0997e-13 | 0.0619 | 6 | 1.7481e-12 | 0.0740 | 8 | 6.3194e-09 | 0.0939 |
| $4/100/10^5$ | 5 | 5.6707e-12 | 0.0562 | 6 | 5.6980e-12 | 0.0647 | 8 | 3.9230e-09 | 0.0755 |
| 6/20/10 | 5 | 7.0477e-07 | 0.0089 | 6 | 1.9151e-11 | 0.0099 | 7 | 1.1560e-07 | 0.0088 |
| $6/20/10^2$ | 5 | 7.3097e-07 | 0.0178 | 6 | 2.4924e-11 | 0.0187 | 7 | 1.2873e-07 | 0.0183 |
| $6/20/10^3$ | 5 | 2.0372e-06 | 0.0213 | 6 | 3.0223e-11 | 0.0230 | 7 | 1.1090e-07 | 0.0172 |
| $6/20/10^4$ | 5 | 3.7762e-07 | 0.0141 | 6 | 2.2794e-12 | 0.0206 | 7 | 2.0458e-08 | 0.0199 |
| $6/100/10^2$ | 6 | 1.8554e-09 | 0.0966 | 7 | 1.0525e-09 | 0.0944 | 9 | 2.4459e-10 | 0.1059 |
| $6/100/10^3$ | 6 | 2.3251e-09 | 0.0733 | 7 | 6.7196e-10 | 0.0657 | 9 | 5.2670e-10 | 0.0760 |
| $6/100/10^4$ | 6 | 2.3390e-09 | 0.1001 | 7 | 1.2669e-09 | 0.0862 | 9 | 1.1513e-09 | 0.1065 |
| $6/100/10^5$ | 6 | 1.7547e-09 | 0.0586 | 7 | 9.6412e-10 | 0.0659 | 9 | 7.5397e-10 | 0.0885 |
| 8/20/10 | 6 | 5.5640e-12 | 0.0279 | 7 | 7.5012e-08 | 0.0337 | 8 | 4.9566e-11 | 0.0215 |
| $8/20/10^2$ | 6 | 9.6193e-12 | 0.0095 | 7 | 1.4575e-08 | 0.0098 | 8 | 4.3410e-12 | 0.0073 |
| $8/20/10^3$ | 6 | 1.5210e-11 | 0.0290 | 7 | 5.6325e-08 | 0.0415 | 8 | 3.4184e-11 | 0.0264 |
| $8/20/10^4$ | 6 | 8.1522e-12 | 0.0240 | 7 | 1.6389e-08 | 0.0270 | 8 | 4.7382e-12 | 0.0164 |

Table 7: The number of Newton direction used

| Problem | TPI | | TPII | | | |
|---|---|---|---|---|---|---|
| Scale | m=4 n=20,60,100 | m=6 n=20 | m=4 n=20,100 | m=6 n=20 | m=6 n=100 | m=8 n=20 |
| Algorithm 3.1 | 1 | 1 | 0 | 2 | 2 | 3 |
| Chebyshev's method | 1 | 1 | 0 | 3 | 2 | 3 |

Table 8: Numerical results of TPI

| TPI | Algorithm 3.1 | | | Algorithm 2.1 | | |
|---|---|---|---|---|---|---|
| m/n/$\gamma$ | Iter | Term | Cpu | Iter | Term | Cpu |
| 4/20/10 | 6 | 7.2677e-06 | 0.0065 | 7 | 3.8593e-07 | 0.0033 |
| $4/20/10^2$ | 6 | 1.6578e-06 | 0.0088 | 12 | 9.4410e-06 | 0.0016 |
| $4/20/10^3$ | 6 | 3.4022e-06 | 0.0072 | 44 | 8.2796e-06 | 0.0034 |
| $4/20/10^4$ | 6 | 1.5409e-06 | 0.0153 | 412 | 9.8264e-06 | 0.0311 |
| $4/20/10^5$ | 6 | 3.7059e-06 | 0.0412 | $> 500$ | | |
| $4/60/10^2$ | 8 | 5.6504e-07 | 0.0515 | 8 | 1.8534e-06 | 0.0049 |
| $4/60/10^3$ | 8 | 1.6306e-07 | 0.0287 | 10 | 9.2411e-07 | 0.0040 |
| $4/60/10^4$ | 8 | 2.9761e-07 | 0.0263 | 31 | 7.5659e-06 | 0.0072 |
| $4/60/10^5$ | 8 | 7.4494e-07 | 0.0438 | 214 | 9.8432e-06 | 0.0416 |
| $4/60/10^6$ | 8 | 2.3406e-07 | 0.1168 | $> 500$ | | |
| $4/100/10^3$ | 9 | 1.4769e-07 | 0.0546 | 9 | 5.3419e-07 | 0.0040 |
| $4/100/10^4$ | 9 | 1.5529e-07 | 0.0522 | 14 | 9.3397e-06 | 0.0056 |
| $4/100/10^5$ | 9 | 1.1736e-07 | 0.0559 | 47 | 7.1193e-06 | 0.0170 |
| $4/100/10^6$ | 9 | 4.1768e-07 | 0.0933 | 330 | 9.6937e-06 | 0.1046 |
| $4/100/10^7$ | 9 | 8.6407e-06 | 0.2128 | $> 500$ | | |
| $6/20/10^3$ | 16 | 7.4161e-07 | 0.0624 | 8 | 7.3774e-07 | 0.0141 |
| $6/20/10^4$ | 16 | 9.3618e-07 | 0.0162 | 20 | 7.2058e-06 | 0.0019 |
| $6/20/10^5$ | 16 | 1.6598e-06 | 0.0686 | 83 | 9.3859e-06 | 0.0286 |
| $6/20/10^6$ | 16 | 3.5723e-06 | 0.0703 | 271 | 9.8257e-06 | 0.0884 |
| $6/20/10^7$ | 17 | 5.3842e-07 | 0.1176 | $> 500$ | | |

Table 9: Numerical results of TPII

| TPII | Algorithm 3.1 | | | Algorithm 2.1 | | |
|---|---|---|---|---|---|---|
| m/n/$\gamma$ | Iter | Term | Cpu | Iter | Term | Cpu |
| $4/60/10^2$ | 4 | 2.3506e-06 | 0.0090 | 17 | 1.6932e-10 | 0.0087 |
| $4/60/10^3$ | 4 | 2.9075e-06 | 0.0087 | 32 | 9.0272e-10 | 0.0120 |
| $4/60/10^4$ | 4 | 2.7891e-06 | 0.0102 | 166 | 1.2197e-08 | 0.0613 |
| $4/60/10^5$ | 4 | 2.9736e-06 | 0.0076 | $> 500$ | | |
| $6/60/10^3$ | 6 | 7.6223e-09 | 0.0155 | 23 | 3.9337e-14 | 0.0084 |
| $6/60/10^4$ | 6 | 7.3604e-09 | 0.0149 | 31 | 1.2187e-13 | 0.0126 |
| $6/60/10^5$ | 6 | 6.9093e-09 | 0.0163 | 102 | 9.7514e-13 | 0.0370 |
| $6/60/10^6$ | 6 | 8.9972e-09 | 0.0151 | $> 500$ | | |
| $8/60/10^3$ | 6 | 2.2140e-06 | 0.0180 | 27 | 1.1136e-17 | 0.0105 |
| $8/60/10^4$ | 6 | 4.1664e-06 | 0.0190 | 26 | 2.8303e-17 | 0.0158 |
| $8/60/10^5$ | 6 | 7.7349e-06 | 0.0180 | 33 | 2.0283e-17 | 0.0192 |
| $8/60/10^6$ | 6 | 6.1480e-06 | 0.0181 | 78 | 1.9403e-16 | 0.0315 |
| $10/20/10^3$ | 6 | 3.3169e-08 | 0.0081 | 24 | 6.2283e-17 | 0.0059 |
| $10/20/10^4$ | 7 | 1.8292e-11 | 0.0089 | 29 | 6.0435e-17 | 0.0085 |
| $10/20/10^5$ | 6 | 4.8338e-08 | 0.0059 | 227 | 1.0426e-15 | 0.0197 |
| $10/20/10^6$ | 7 | 9.8427e-11 | 0.0072 | 374 | 4.3409e-15 | 0.0557 |

Table 10: Numerical results of TPIII by Algorithm 3.1

| $\lambda$ | $x^T$ | | | Iter | CPU | Occ |
|---|---|---|---|---|---|---|
| 0.8893 | [0.6672 | 0.2476 | -0.7026] | 22.68 | 0.0453 | 6% |
| 0.8169 | [0.8410 | -0.2639 | 0.4723] | 28.66 | 0.0494 | 12% |
| 0.3633 | [0.2679 | 0.6448 | 0.7159] | 22.57 | 0.0400 | 14% |
| 0.2682 | [0.6099 | 0.4362 | 0.6616] | 21.00 | 0.0525 | 2% |
| 0.2433 | [0.9896 | 0.0945 | -0.1084] | 18.50 | 0.0289 | 8% |
| 0.1735 | [0.3358 | 0.9070 | 0.2541] | 26.50 | 0.0365 | 4% |
| -0.0451 | [0.7797 | 0.6135 | 0.1250] | 6.90 | 0.0130 | 22% |
| -0.5629 | [0.1762 | -0.1795 | 0.9678] | 13.77 | 0.0275 | 18% |
| -1.0954 | [0.5913 | -0.7468 | -0.3043] | 21 | 0.0364 | 14% |

Table 11: Numerical results of TPIII by Algorithm 2.2

| $\lambda$ | $x^T$ | | | Iter | CPU | Occ |
|---|---|---|---|---|---|---|
| 0.8893 | [0.6672 | 0.2477 | -0.7025] | 37.34 | 0.0309 | 52% |
| 0.8169 | [0.8410 | -0.2641 | 0.4723] | 10.00 | 0.0127 | 2% |
| 0.3633 | [0.2684 | 0.6449 | 0.7156] | 20.64 | 0.0182 | 28% |

TPIV. Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[4,n]}$ such that

$$a_{ijkl} = \sin(i + j + k + l) \quad (1 \le i, j, k, l \le n).$$

TPV. Consider the symmetric tensor $\mathcal{A} \in \mathbb{R}^{[4,n]}$ such that

$$a_{ijkl} = \tan(i) + \tan(j) + \tan(k) + \tan(l) \quad (1 \le i, j, k, l \le n).$$

We choose an $n + 1$-dimensional vector whose elements are uniformly distributed in $(-1, 1)$ randomly as the initial point for TPIV and TPV, and perform the two algorithms 30 times, respectively. For TPIV, let $n = 5$, $\alpha = 10$ in Algorithm 2.2. For TPV, let $n = 6$, $\alpha = 2$ in Algorithm 2.2. The relative numerical results are listed in Tables 12-15. There are several eigenvectors corresponding to eigenvalue 0, we only record one of them.

From Table 10 - Table 15, it is easy to see that Algorithm 3.1 can compute much more eigenvalues than Algorithm 2.2. Especially for TPIV, TPV, all the eigenvalues are found by

Table 12: Numerical results of TPIV by Algorithm 3.1

| $\lambda$ | $x^T$ | | | | | Iter | CPU | Occ |
|---|---|---|---|---|---|---|---|---|
| 7.2595 | [ 0.2686 | 0.6150 | 0.3959 | -0.1872 | -0.5982] | 17.62 | 0.1076 | 26.66% |
| 4.6408 | [-0.5055 | 0.1228 | 0.6382 | 0.5669 | -0.0256] | 25.50 | 0.2333 | 6.66% |
| 0.0000 | [-0.6181 | 0.6179 | -0.4213 | -0.2042 | 0.1302] | 17.56 | 0.1154 | 53.33% |
| -3.9204 | [ 0.1785 | -0.4847 | -0.7023 | -0.2742 | 0.4060] | 12.00 | 0.0806 | 3.33% |
| -8.8463 | [-0.5809 | -0.3563 | 0.1959 | 0.5680 | 0.4179] | 11.33 | 0.0883 | 10% |

Table 13: Numerical results of TPIV by Algorithm 2.2

| $\lambda$ | $x^T$ | | | | | Iter | CPU | Occ |
|---|---|---|---|---|---|---|---|---|
| 7.2595 | [ 0.2689 | 0.6152 | 0.3958 | 0.1869 | 0.5980] | 18.82 | 0.0637 | 53.33% |
| 4.6408 | [-0.5061 | 0.1233 | 0.6381 | 0.5664 | -0.0253] | 24.35 | 0.1193 | 46.66% |

Table 14: Numerical results of TPV by Algorithm 3.1

| $\lambda$ | $x^T$ | | | | | | Iter | CPU | Occ |
|---|---|---|---|---|---|---|---|---|---|
| 45.5045 | [-0.6281 | -0.0717 | -0.3754 | -0.5687 | 0.1060 | -0.3533] | 11.42 | 0.1229 | 23.33% |
| 0.0000 | [-0.0904 | 0.4786 | 0.5937 | -0.0878 | -0.3608 | -0.5220] | 23.88 | 0.3090 | 60% |
| -133.2871 | [ 0.1936 | 0.5222 | 0.3429 | 0.2287 | 0.6272 | 0.3559] | 11.2 | 0.1232 | 16.66% |

Table 15: Numerical results of TPV by Algorithm 2.2

| $\lambda$ | $x^T$ | | | | | | Iter | CPU | Occ |
|---|---|---|---|---|---|---|---|---|---|
| 0.0000 | [-0.4921 | -0.3833 | 0.0945 | 0.3987 | -0.6629 | 0.0597] | 5.00 | 0.0162 | 10% |
| -133.2871 | [ 0.1936 | 0.5222 | 0.3429 | 0.2287 | 0.6271 | 0.3559] | 16.59 | 0.1017 | 90% |

Algorithm 3.1. Although the CPU of Algorithm 2.2 is shorter, the confirming procedure of $\alpha$ is long.

From numerical results,we find that Algorithm 3.1 is the least for the number of iteration. This is consistent with the fourth order convergence of Algorithm 3.1. For the precision and time, Algorithm 3.1 is comparable with Chebyshev's method and Newton method. Hence, Algorithm 3.1 is theoretically superior, and numerically slightly better than the existing algorithms.

## References

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

[2] C. Chang, K. Pearson and T. Zhang, On eigenvalue problems of real symmetric tensors, *J. Math. Anal. Appl.* 350 (2009) 416–422.

[3] C. Chang, K. Pearson and T. Zhang, Primitivity, the convergence of the NQZ method, and the largest eigenvalue for nonnegative tensors, *SIAM J. Matrix Anal. Appl.* 32 (2011) 806–819.

[4] Z. Chen, L. Qi, Q. Yang and Y. Yang, The solution methods for the largest eigenvalue (singular value) of nonnegative tensors and convergence analysis, *Linear algebra Appl.* 439 (2013) 3713–3733.

[5] C. Cui, Y. Dai and J. Nie, All real eigenvalues of symmetric tensors, *SIAM J. Matrix Anal. Appl.* 35 (2014) 1582–1601.

[6] C. Hao, C. Cui and Y. Dai, A sequential subspace projection method for extreme Z-eigenvalues of supersymmetric tensors, *Numer. Linear Algebr. Appl.* 22 (2015) 283–298.

[7] L. Han, An unconstrained optimization approach for finding real eigenvalues of even order symmetric tensors, *Numer. Algebr. Control Optim.* 3 (2013) 583–599.

[8] S. Hu, G. Li, L. Qi and Y. Song, Finding the maximum eigenvalue of essentially nonnegative symmetric tensors via sum of squares programming, *J. Optimiz. Theory App.* 158 (2013) 717–738.

[9] S. Hu, Z. Huang and L. Qi, Strictly nonnegative tensors and nonnegative tensor partition, *Sci. China Math.* 57 (2014) 181–195.

[10] G. Kolda and R. Mayo, Shifted power method for computing tensor eigenpairs, *SIAM J. Matrix Anal. Appl.* 32 (2011) 1095–1124.

[11] G. Kolda and R. Mayo, An adaptive Shifted power method for computing generalized tensor eigenpairs, *SIAM J. Matrix Anal. Appl.* 35 (2014) 1563–1581.

[12] H. Lim, Singular values and eigenvalues of tensors, a variational approach, *In: Proceedings of the 1st IEEE International Workshop on Computational Advances of Multitensor Adaptive Processing.* 1 (2005) 129–132.

[13] M. Ng, L. Qi and G. Zhou, Finding the largest eigenvalue of a nonnegative tensor, *SIAM J. Matrix Anal. Appl.* 31 (2009) 1090–1099.

[14] Q. Ni and L. Qi, A quadratically convergent algorithm for finding the largest eigenvalue of nonnegative homogeneous polynomial map, *J. Glob. Optim.* 61 (2015) 627–641.

[15] J. Nocedal and J. Wright, *Numerical Optimization*, Science Press, Beijing 2006.

[16] K. Pearson, Essentially positive tensors, *Int. J. Algebra.* 4 (2010) 421–427.

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

[18] L. Qi, F. Wang and Y. Wang, Z-eigenvalue methods for a global polynomial optimization problem, *Mathematical Programming.* 118 (2009) 301–316.

[19] L. Qi, Y. Wang and E.X. Wu, D-Eigenvalues of diffusion kurtosis tensors, *J. Compu. Appl. Math.* 221 (2008) 150–157.

[20] D. Schatz, M. Low, A. Van De Geijn and G. Kolda, Exploiting symmetry in tensors for high performance: multiplication with symmetric tensors, *SIAM J. Sci. Comput.* 36 (2014) 453–479.

[21] W. Werner, Iterative solution of systems of nonlinear equations based upon quadratic approximations, *Comp. Maths. with Appls.* 12A (1986) 331–343.

[22] Y. Yang and Q. Yang, Further results for Perron-Frobenius Theorem for nonnegative tensors, *SIAM J. Matrix Anal. Appl.* 31 (2010) 2517–2530.

[23] Q. Yang and Y. Yang, Further results for Perron-Frobenius theorem for nonnegative tensors II, *SIAM J. Matrix Anal. Appl.* 32 (2011) 1236–1250.

[24] W. Yang and Q. Ni, A cubically convergent method for solving the largest eigenvalue of a nonnegative irreducible tensor, *Numer. Algorithms.* 77 (2018) 1183–1197.

[25] G. Yu, F. Yu, Y. Xu, Y. Song and Y. Zhou, An adaptive gradient method for computing generalized tensor eigenpairs. *Comput. Optim. Appl.* 65 (2016) 781–797.

[26] M. Zeng and Q. Ni, Quasi-Newton method for computing Z-eigenpairs of a symmetric tensor, *Pacilc J. Optim.* 11 (2009) 279–290.

[27] L. Zhang and L. Qi, Linear convergence of an algorithm for computing the largest eigenvalue of a nonnegative tensor, *Numer. Linear Algebra Appl.* 19 (2012) 830–841.

[28] X. Zhang, Q. Ni and Z. Ge, A convergent Newton algorithm for computing Z-eigenvalues of an almost nonnegative irreducible tensor, *Optim. Method Softw.* 35 (2020) 377–393.

---

*Manuscript received 21 January 2021*
*revised 31 May 2021*
*accepted for publication 18 June 2021*

Wei-wei Yang
School of Physical and Mathematical Sciences
Nanjing Tech University, Nanjing 211816, P.R. China
E-mail address: yangweiwei0810@126.com

Hao Liu
School of Physical and Mathematical Sciences
Nanjing Tech University, Nanjing 211816, P.R. China
E-mail address: lhmath@njtech.edu.cn

Qin Ni
College of Science, NUAA, Nanjing 211106, P.R. China
E-mail address: niqfs@nuaa.edu.cn