Received: 15 January 2019 | Revised: 2 December 2019 | Accepted: 26 December 2019

DOI: 10.1002/nla.2284

RESEARCH ARTICLE

A locally and cubically convergent algorithm for computing $\mathcal{Z}$-eigenpairs of symmetric tensors

Ruijuan Zhao | Bing Zheng | Maolin Liang | Yangyang Xu

School of Mathematics and Statistics, Lanzhou University, Lanzhou, China

Correspondence
Bing Zheng, School of Mathematics and Statistics, Lanzhou University, Lanzhou 730000, China.
Email: bzheng@lzu.edu.cn

Funding information
The National Natural Science Foundation of China, 11571004; the Science Foundation of Education Department of Gansu Province, 2017A-078

Summary
This paper is concerned with computing $\mathcal{Z}$-eigenpairs of symmetric tensors. We first show that computing $\mathcal{Z}$-eigenpairs of a symmetric tensor is equivalent to finding the nonzero solutions of a nonlinear system of equations, and then propose a modified normalized Newton method (MNNM) for it. Our proposed MNNM method is proved to be locally and cubically convergent under some suitable conditions, which greatly improves the Newton correction method and the orthogonal Newton correction method recently provided by Jaffe, Weiss and Nadler since these two methods only enjoy a quadratic rate of convergence. As an application, the unitary symmetric eigenpairs of a complex-valued symmetric tensor arising from the computation of quantum entanglement in quantum physics are calculated by the MNNM method. Some numerical results are presented to illustrate the efficiency and effectiveness of our method.

KEYWORDS
$\mathcal{Z}$-eigenpairs, *US*-eigenpairs, cubical convergence, modified normalized Newton method, Newton correction method, symmetric tensors

1 | INTRODUCTION

The eigenvalue problems of tensors have attracted much attention in recent years because of their applications in high-order Markov chains,[1] magnetic resonance imaging,[2] quantum entanglement,[3,4] automatical control,[5] the best rank-one approximations in statistical data analysis,[6–10] learning binary latent variable models,[11] and so on. Unlike matrices, there are various definitions of eigenvalues for tensors, including $\mathcal{H}$-eigenvalues, $\mathcal{Z}$-eigenvalues, and $\mathcal{D}$–eigenvalues; see, for example, other works[12–18] and the references therein. In this paper, we are mainly concerned with the calculation of $\mathcal{Z}$-eigenvalues and corresponding $\mathcal{Z}$-eigenvectors. This kind of eigenvalues and eigenvectors has practical applications, for example, the smallest $\mathcal{Z}$-eigenvalue can reflect the stability of a nonlinear autonomous system in automatic control, and the principal $\mathcal{Z}$-eigenvector can depict the orientations of nerve fibers in the voxel of white matter of human brain.[5,19]

Computing $\mathcal{Z}$-eigenpairs of a third or higher order tensor is equivalent to finding nontrivial solutions of a system of inhomogeneous polynomial equations in several variables. This implies that the task in the computation of $\mathcal{Z}$-eigenpairs of a tensor will be much more difficult when its order and dimension are very large. For this reason, some iterative methods have been developed to compute one or more $\mathcal{Z}$-eigenpairs of the tensors with special structure, such as nonnegative tensors and symmetric tensors.

Recently, for nonnegative tensors, Guo et al.[20] proposed a modified Newton iteration to compute some nonnegative $\mathcal{Z}$-eigenpairs and showed that this method has local quadratic convergence under appropriate assumptions. Kuo et al.[21] presented a homotopy continuation method for the same purpose. For symmetric tensors, Cui, Dai and

Nie[22] used Jacobian semidefinite relaxations method to compute all real eigenvalues. Chen, Han and Zhou[23] proposed a homotopy continuation method to find all isolated eigenpairs. Although these methods can capture all or most of the eigenvalues, the computations are very expensive (e.g., the work of Jaffe et al.[24]). Several fast algorithms have been established for calculating extreme $\mathcal{Z}$-eigenvalues in the literature.[25,26] Kolda and Mayo[27] proposed a shifted symmetric higher-order power method (SS-HOPM) for computing $\mathcal{Z}$-eigenpairs which depends on the choice of a shift parameter. It seems difficult to choose a reasonable value for the parameter, and an inappropriate choice may increase run-time or lead to a complete lack of convergence. Owing to this motivation, Kolda and Mayo[28] presented an adaptive version, called the adaptive shifted power method (GEAP), for generalized tensor eigenpairs, which is also suitable for finding $\mathcal{Z}$-eigenpairs. Using fixed point theory, they showed that SS-HOPM and GEAP can linearly converge to the positive or negative stable eigenpairs but cannot converge to the unstable ones. To find the unstable eigenpairs, Jaffe et al.[24] provided a Newton correction method (NCM) and an orthogonal Newton correction method (O-NCM) for computing $\mathcal{Z}$-eigenpairs of symmetric tensors, and proved that their convergence is locally quadratic under the condition termed γ-Newton-stability. In particular, the O-NCM method avoids the notable drawback of NCM, that is, NCM may fail to converge to a $\mathcal{Z}$-eigenvector associated with $\mathcal{Z}$-eigenvalue zero. The NCM and O-NCM methods have been shown to be much faster than the SS-HOPM and GEAP methods.

In order to improve the convergence rate of the iterative methods mentioned previously for computing $\mathcal{Z}$-eigenpairs of symmetric tensors, in this paper we first characterize the $\mathcal{Z}$-eigenpairs of a symmetric tensor as the nonzero solutions of a nonlinear system of equations (see Equation (4)), which is a modification of the nonlinear system of equations $g(x) = 0$ given in the work of Jaffe et al.[24] by adding a "regularization" term controlled by a specified parameter $c \in \mathbb{R}$. And then, we propose a modified normalized Newton method (MNNM) for it, which can be regarded as a regularized version of NCM. At each iteration of MNNM, the new approximation is calculated in two steps (as opposed to one step in NCM), especially, the second step can be rapidly computed using the quantities calculated in the first step (i.e., the inverse of the Jacobian is computed only once for both steps). Under the same γ-Newton-stability condition, we prove that MNNM locally converges to a $\mathcal{Z}$-eigenpair at a cubical rate. Compared with the known NCM and O-NCM methods, MNNM enjoys the higher convergence rate with increasing only a small amount of the computational cost, and keeps the runtime low. Meanwhile, similarly to the O-NCM method, MNNM also overcomes the aforementioned shortcomings of NCM. Theoretical analysis and numerical simulation (see Equation (21), Figure 1(B) and the left in Figures 3-5) show that the behavior of MNNM is closely related to the regularization parameter c. Finally, we give an application for computing the unitary symmetric eigenpairs (*US*-eigenpairs) of a complex symmetric tensor, which arises from the computation of quantum entanglement in quantum physics, by reducing the problem to that of finding $\mathcal{Z}$-eigenpairs of an appropriate larger tensor. Numerical examples demonstrate that our method is more competitive and efficient than the existing ones.

The rest of this paper is organized as follows. In Section 2, we introduce some notions and preliminary results related to tensors, and review the NCM method.[24] Section 3 proposes a MNNM for computing $\mathcal{Z}$-eigenpairs of symmetric tensors and analyzes its cubic convergence. In Section 4, we report some numerical results, including computing *US*-eigenpairs. Finally, we make some conclusions to end this paper in Section 5.

2 | PRELIMINARIES

In this section, we recall some definitions and operations related to tensors. Meanwhile, to motivate our approach, we briefly review the NCM[24] for computing $\mathcal{Z}$-eigenpairs of symmetric tensors.

2.1 | Definitions and operations

Let $\mathbb{C}$ ($\mathbb{R}$) be the complex (real) field. $\|\cdot\|$ denotes l_2-norm, and Σ denotes the unit sphere on $\mathbb{R}^n$, that is, $\Sigma = \{x \in \mathbb{R}^n : \|x\| = 1\}$. We define $x^\perp \equiv \{y \in \mathbb{R}^n : x^T y = 0\}$, and use I to denote an identity matrix with appropriate size.

A complex (real) mth-order n-dimensional tensor $\mathcal{A}$ consists of n^m entries in $\mathbb{C}$ ($\mathbb{R}$), which is defined as follows:

$$\mathcal{A} = (a_{i_1 i_2 \dots i_m}), \quad a_{i_1 i_2 \dots i_m} \in \mathbb{C}\ (\mathbb{R}), \quad 1 \le i_1, \quad i_2, \dots, i_m \le n.$$

$\mathcal{A}$ is called symmetric if the value of its entries $a_{i_1 i_2 \dots i_m}$ is invariant under any permutation of their indices $\{i_1, i_2, \dots, i_m\}$. Throughout this paper, we denote the set of all complex (real) symmetric tensors of order m and dimension n by $\mathbb{CS}^{[m,n]}$ ($\mathbb{RS}^{[m,n]}$).

Let $\mathcal{A}$ be an mth-order n-dimensional tensor and x be an n-dimensional vector. For $0 \le r \le m-1$, the $(m-r)$-times product[27] of the tensor $\mathcal{A}$ and the vector x is an $(m-r)$th-order n-dimensional tensor, denoted by $\mathcal{A}x^{m-r}$, which is defined as

$$(\mathcal{A}x^{m-r})_{i_1\dots i_r} = \sum_{i_{r+1},\dots,i_m=1}^{n} a_{i_1 i_2\dots i_m} x_{i_{r+1}} \cdots x_{i_m},\ 1 \le i_1, i_2, \cdots, i_r \le n.$$

Obviously,

$$\mathcal{A}x^{m} = \sum_{i_1,\dots,i_m=1}^{n} a_{i_1 i_2\dots i_m} x_{i_1} \cdots x_{i_m},$$

is a scalar, $\mathcal{A}x^{m-1}$ is an n-dimensional vector whose entries are defined as

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

and $\mathcal{A}x^{m-2}$ is an $n \times n$ matrix whose entries are defined as

$$(\mathcal{A}x^{m-2})_{i_1 i_2} = \sum_{i_3,\dots,i_m=1}^{n} a_{i_1 i_2\dots i_m} x_{i_3} \cdots x_{i_m},\ 1 \le i_1, i_2 \le n.$$

Particularly, when $\mathcal{A}$ is a symmetric tensor, we can get the derivative of the above tensor-vector product

$$(\mathcal{A}x^{m-r})' = (m-r)\mathcal{A}x^{m-r-1}, \quad 0 \le r \le m-1.$$

The following definition of $\mathcal{Z}$-eigenvalues and eigenvectors was independently introduced by Qi[13] and Lim.[14]

Definition 1. Let $\mathcal{A}$ be an mth-order n-dimensional real tensor. If there exist $\lambda \in \mathbb{R}$ and a nonzero vector $x \in \mathbb{R}^n$ satisfying

$$\mathcal{A}x^{m-1} = \lambda x \quad \text{and} \quad x^T x = 1,$$

then λ is called a $\mathcal{Z}$-eigenvalue of $\mathcal{A}$ and x is called the corresponding $\mathcal{Z}$-eigenvector. We call (λ, x) a $\mathcal{Z}$-eigenpair for short.

Note that if (λ_*, x_*) is a $\mathcal{Z}$-eigenpair, then $((-1)^m\lambda_*, -x_*)$ is a $\mathcal{Z}$-eigenpair as well. Following common practice, these two eigenpairs are considered to belong to the same equivalence class.[29]

2.2 | Newton correction method

This subsection gives the NCM for computing $\mathcal{Z}$-eigenpairs of symmetric tensors proposed by Jaffe et al.[24] To describe this method, we need the following notations:

$$\mu(x) := \mathcal{A}x^m, \quad g(x) := \mathcal{A}x^{m-1} - \mu(x)x, \quad A(x) := (m-1)\mathcal{A}x^{m-2} - \mu(x)I - mx(\mathcal{A}x^{m-1})^T. \tag{2}$$

Now, the NCM method is presented in Algorithm 1.

Algorithm 1. Newton correction method

```
1: Given a tensor $\mathcal{A} \in \mathbb{RS}^{[m,n]}$, a tolerance parameter $\delta > 0$ and randomly choose an initial vector $x^{(0)} \in \Sigma$.
2: Set $k = 0$ and $\lambda^{(0)} = \mu(x^{(0)})$.
3: while $\|x^{(k+1)} - x^{(k)}\| > \delta$ do
4:     Solve $A(x^{(k)})\Delta x^{(k)} = -g(x^{(k)})$.
5:     Calculate $x^{(k+1)} = \frac{x^{(k)}+\Delta x^{(k)}}{\|x^{(k)}+\Delta x^{(k)}\|}$.
6:     Set $\lambda^{(k+1)} = \mu(x^{(k+1)})$.
7:     $k \leftarrow k+1$, go to step 3.
8: end while
9: return $(\lambda^{(k+1)}, x^{(k+1)})$.
```

We remark that, different from the derivation in the work of Jaffe et al.,[24] the key steps 4 and 5 in Algorithm 1 are actually the normalized Newton method[30] used to solve the nonlinear system of equations $g(x) = 0$ when it is known that the solution has unit norm, where $g(x)$ is defined in Equation (2).

To prove the quadratic convergence of NCM, Jaffe et al.[24] introduced the following definition.

Definition 2. For $\gamma > 0$, a $\mathcal{Z}$-eigenpair (λ_*, x_*) is called γ-Newton-stable if all eigenvalues of $H_p(x_*)$ in absolute value are at least γ, where

$$H_p(x_*) = U_{x_*}^T \left((m-1)\mathcal{A}x_*^{m-2} - \lambda_* I\right) U_{x_*}, \tag{3}$$

and U_{x_*} is an $n \times (n-1)$ matrix whose columns form an orthonormal basis for $x_*^{\perp}$.

Note that the eigenpair (λ_*, x_*) is γ-Newton-stable for some $\gamma > 0$ if and only if $H_p(x_*)$ is of full rank. Jaffe et al.[24] proved that if (λ_*, x_*) is a $\mathcal{Z}$-eigenpair of the symmetric tensor $\mathcal{A}$ with $\lambda_* \neq 0$, and (λ_*, x_*) is γ-Newton-stable, then the NCM method locally converges to x_* at a quadratic rate. Moreover, their analysis showed that for the case $\lambda_* = 0$, the NCM method may not converge to the corresponding $\mathcal{Z}$-eigenvector. To overcome this shortcoming, they provided a variant of the NCM method, called the O-NCM, and showed that this method also enjoys a quadratic rate of convergence under the γ-Newton-stable condition. In next section we will derive a new method, named the MNNM, which not only cubically converges to the eigenpair (λ_*, x_*) regardless of $\lambda_* = 0$ or not, but also need not to compute a matrix with $n-1$ orthonormal columns at each iteration in the O-NCM method.

3 | A MNNM FOR $\mathcal{Z}$-eigenpairs

In this section, we propose a MNNM for computing $\mathcal{Z}$-eigenpairs of symmetric tensors, and then discuss its cubic convergence under the same γ-Newton-stable condition as that for the NCM and O-NCM methods. We start with the following theorem, which plays a key role in deriving our proposed method.

Theorem 1. *Let $\mathcal{A} \in \mathbb{RS}^{[m,n]}$ and c be a nonzero real number. Then (λ_*, x_*) is a $\mathcal{Z}$-eigenpair of $\mathcal{A}$ if and only if x_* is a nonzero solution of the nonlinear system of equations*

$$F(x;c) := \mathcal{A}x^{m-1} - \lambda_R(x)x + \frac{c}{2}(x^Tx - 1)x = 0, \tag{4}$$

and $\lambda_* = \lambda_R(x_*)$, where $\lambda_R(x) = \frac{\mathcal{A}x^m}{x^Tx}$ $(x \neq 0)$.

Proof. ($\Rightarrow$) Since (λ_*, x_*) is a $\mathcal{Z}$-eigenpair of $\mathcal{A}$, that is, $\mathcal{A}x_*^{m-1} = \lambda_* x_*$ and $x_*^T x_* = 1$, we have that $\lambda_R(x_*) = \frac{x_*^T(\mathcal{A}x_*^{m-1})}{x_*^T x_*} = \lambda_*$ and $F(x_*; c) = 0$.

($\Leftarrow$) Suppose $x_* \neq 0$ and x_* satisfies $F(x_*; c) = 0$, that is,

$$\mathcal{A}x_*^{m-1} - \lambda_R(x_*)x_* + \frac{c}{2}(x_*^T x_* - 1)x_* = 0, \tag{5}$$

then, substituting $\lambda_R(x_*)$ into Equation (5) and left multiplying by x_*^T on the both sides of Equation (5) yield

$$\frac{c}{2}x_*^T x_*(x_*^T x_* - 1) = 0. \tag{6}$$

Note that x_* and c are nonzero, so we get from Equation (6) that $x_*^T x_* = 1$. Thus, it follows from Equation (5) that $\mathcal{A}x_*^{m-1} = \lambda_R(x_*)x_*$, and consequently $\mathcal{A}x_*^{m-1} = \lambda_* x_*$. This implies that (λ_*, x_*) is a $\mathcal{Z}$-eigenpair of the tensor $\mathcal{A}$. ∎

Theorem 1 shows the equivalence between $\mathcal{Z}$-eigenpairs of a tensor and nonzero solutions of the nonlinear system of equations (4). Hence we can compute $\mathcal{Z}$-eigenpairs of a tensor via Equation (4). For a real symmetric tensor $\mathcal{A}$, the derivatives of $\lambda_R(x)$ and $F(x; c)$ can be easily calculated as follows:

$$\lambda_R'(x) = \frac{m\mathcal{A}x^{m-1} - 2\lambda_R(x)x}{x^Tx}, \tag{7}$$

and

$$F'(x;c) = (m-1)\mathcal{A}x^{m-2} - \lambda_R(x)I - x(\lambda_R'(x))^T + \frac{c}{2}(x^Tx - 1)I + cxx^T.$$

In particular, when $\|x\| = 1$, we have that $\lambda_R(x) = \mathcal{A}x^m = \mu(x)$, $F(x;c) = g(x)$ and

$$\begin{aligned} F'(x;c) &= (m-1)\mathcal{A}x^{m-2} - \lambda_R(x)I - mx(\mathcal{A}x^{m-1})^T + (2\lambda_R(x)+c)xx^T \\ &= A(x) + (2\mu(x)+c)xx^T, \end{aligned}$$

where $\mu(x)$, $g(x)$ and $A(x)$ are defined by Equation (2).

For solving the nonlinear equation $f(x) = 0$, where $f : \Omega \subseteq \mathbb{R} \to \mathbb{R}$ and Ω is an open interval, some two-step Newton methods were proposed in the literature,[31–33] which consist of predictor-step and corrector-step and achieve cubical convergence. These methods were obtained by using Taylor's expansion of the function $f(x)$ or some decomposition techniques applied to another nonlinear equation equivalent (or approximate) to the original $f(x) = 0$. In the works of Darvishi et al.,[34] Waseema et al.,[35] and Traub,[36] these techniques were applied to develop a two-step Newton method for solving a system of nonlinear equations, which is an extension of the approach.[33] Besides, a normalized Newton method was proposed by Tapia et al.[30] for the general system of nonlinear equations whose solution x satisfies $\|x\| = 1$. Inspired by the method,[30] for solving the system of nonlinear equations (4), we appropriately modify the two-step Newton method,[34–36] namely, its predictor-step (Newton step) and corrector-step (approximate Newton step) are projected onto the set Σ since the solution is required to be of unit norm. We thus present a MNNM for computing $\mathcal{Z}$-eigenpairs of a symmetric tensor, which is summarized as the following Algorithm 2.

Algorithm 2. MNNM for computing $\mathcal{Z}$-eigenpairs

```
1: Given a tensor $\mathcal{A} \in \mathbb{RS}^{[m,n]}$, a scalar $c \neq 0$, a tolerance parameter $\delta > 0$ and a maximum iterative number $k_{\max}$, and
   randomly choose an initial vector $x^{(0)} \in \Sigma$.
2: Set $k = 0$ and $\lambda^{(0)} = \mu(x^{(0)})$.
3: **while** $k < k_{\max}$ **do**
4:     Solve
           $\left(A(x^{(k)}) + \left(2\mu(x^{(k)}) + c\right) x^{(k)}(x^{(k)})^T\right)\Delta x^{(k)} = -g(x^{(k)})$.
5:     Calculate $y^{(k)} = \frac{x^{(k)}+\Delta x^{(k)}}{\|x^{(k)}+\Delta x^{(k)}\|}$.
6:     Solve
           $\left(A(x^{(k)}) + \left(2\mu(x^{(k)}) + c\right) x^{(k)}(x^{(k)})^T\right)\Delta y^{(k)} = -g(y^{(k)})$.
7:     Normalize $x^{(k+1)} = \frac{y^{(k)}+\Delta y^{(k)}}{\|y^{(k)}+\Delta y^{(k)}\|}$.
8:     Set $\lambda^{(k+1)} = \mu(x^{(k+1)})$.
9:     If $\|x^{(k+1)} - x^{(k)}\| < \delta$, then break and output $(x^{(k+1)}, \lambda^{(k+1)})$; otherwise $k \leftarrow k+1$, go to step 3.
10: **end while**
```

Unlike the two-step approaches[31,32] for solving the classic equation $f(x) = 0$, Algorithm 2 only involves the first derivative F' and, similarly to the work of Chun,[33] the Jacobian $F'(x^{(k)};c)$ at $x^{(k)}$ is used twice in both the fourth and sixth steps, which greatly saves computational cost. The fourth step in Algorithm 2 can be viewed as a regularization of the NCM method.[24] Besides, the cost at per iterative step of Algorithm 2 is dominated by the computation of $\mathcal{A}x^{m-2}$ and $\mathcal{A}y^{m-1}$ whose computational complexity is about $O(n^m)$, and solving a system of n linear equations whose computational complexity is about $O(n^3)$.

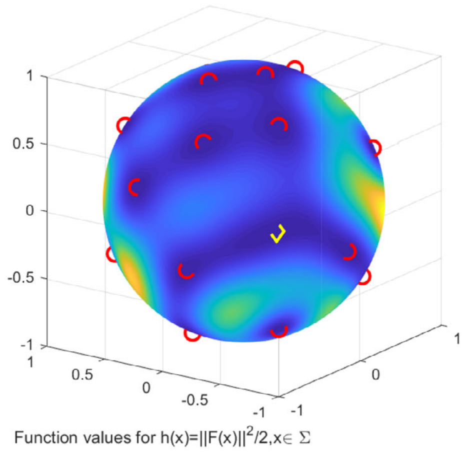
Note that solving the system of nonlinear equations (4) can be viewed as finding the global minima of the nonlinear least-squares problem

$$\min_{x\in\Sigma} h(x) := \frac{1}{2}\|F(x)\|^2.$$

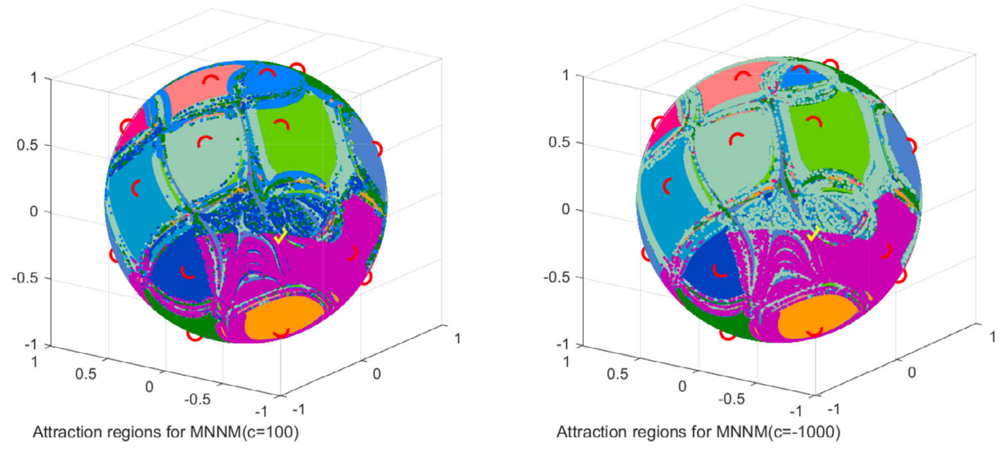
However, $h(x)$ may have local minima which should be avoided. Interestingly enough, our proposed MNNM method can avoid such local minima. Similar to the analysis for NCM,[24] this is because for a local minimum $x_{\text{loc}} \in \Sigma$, $F(x_{\text{loc}})$ is bounded away from 0 and

$$h'(x_{\text{loc}}) = F'(x_{\text{loc}})^T F(x_{\text{loc}}) = (A(x_{\text{loc}}) + (2\mu(x_{\text{loc}}) + c)x_{\text{loc}}x_{\text{loc}}^T)^T F(x_{\text{loc}}) = 0.$$

These imply that the matrix $A(x_{\text{loc}}) + (2\mu(x_{\text{loc}}) + c)x_{\text{loc}}x_{\text{loc}}^T$ is singular. When the approximation point $x^{(k)}$ of MNNM is closer to x_{loc}, the matrix $A(x^{(k)}) + (2\mu(x^{(k)}) + c)x^{(k)}x(k)^T$ becomes more and more singular, which leads to the dramatic increases of $\|\Delta x^{(k)}\|$ and $\|\Delta y^{(k)}\|$ such that $x^{(k+1)}$ is far away from $x^{(k)}$ and x_{loc}. To be clear, we use Example 1 in the work of Kofidis and Regalia[6] to illustrate this phenomenon for our MNNM method. In Figure 1(A), we plot $h(x)$ on the unit sphere and use colors to indicate the value of this function . The circle shape markers denote $\mathcal{Z}$-eigenvectors, and the diamond shape marker represents a local minimum x_{loc} of $h(x)$. Figure 1(B) depicts the attraction regions for MNNM with $c = 100$ and $c = -1000$ by using every point on the unit sphere as the initial point, where every initial point is colored according to which eigenvector it converges to (specifically, the initial point that converges to the same eigenvector is drawn in the same color, otherwise we do it in the different color). From Figure 1(B), we see that MNNM does not converge to the local minimum x_{loc}, and even if the initial point is very close to x_{loc}, MNNM may converge to arbitratily far $\mathcal{Z}$-eigenvectors and thus elegantly avoids the local minimum x_{loc}.



(A)



(B)

FIGURE 1 The value of $h(x)$ on the unit sphere for Example 1 in Reference 6 and the attraction regions for modified normalized Newton method

The following theorem guarantees the convergence of MNNM.

Theorem 2. *Let (λ_*, x_*) be a $\mathcal{Z}$-eigenpair of a real symmetric tensor $\mathcal{A}$, and assume that (λ_*, x_*) is γ-Newton-stable and $c \neq 0$, then the sequence $\{x^{(k)}\}$ generated by Algorithm 2 locally converges to x_* at a cubical rate.*

Proof. Since (λ_*, x_*) is a $\mathcal{Z}$-eigenpair, we know from Theorem 1 that x_* is a nonzero solution of $F(x; c)$, namely, x_* satisfies

$$F(x_*; c) = \mathcal{A}x_*^{m-1} - \lambda_R(x_*)x_* + \frac{c}{2}(x_*^T x_* - 1)x_* = 0,$$

and $\|x_*\| = 1$. Let $\theta_1, \dots, \theta_{n-1}$ denote all eigenvalues of $H_p(x_*)$ defined by Equation (3). Then

$$F'(x_*; c) = (m-1)\mathcal{A}x_*^{m-2} - \lambda_* I + (2\lambda_* + c - m\lambda_*)x_* x_*^T, \tag{8}$$

is symmetric and its all eigenvalues are $c, \theta_1, \dots, \theta_{n-1}$. Note that (λ_*, x_*) is γ-Newton-stable (i.e., $\min\limits_{1\le i\le n-1} |\theta_i| \ge \gamma$ for some $\gamma > 0$) and $c \neq 0$, so $F'(x_*; c)$ is nonsingular and its smallest singular value σ^* satisfies

$$\sigma^* = \sigma_{\min}(F'(x_*; c)) \ge \min\{\gamma, |c|\} > 0.$$

By the continuity of singular values, there exists a $\epsilon > 0$ (which depends on γ and c) such that $\sigma_{\min}(F'(x; c)) \ge \frac{\sigma^*}{2} > 0$ for all x with $\|x - x_*\| < \epsilon$. This implies that $F'(x; c)$ is nonsingular for all x satisfying $\|x - x_*\| < \epsilon$. Hence, we define

$$N(x) = x - F'(x; c)^{-1} F(x; c).$$

Then we can write it as

$$F'(x; c)(x - N(x)) = F(x; c). \tag{9}$$

Computing the derivative on both sides of Equation (9) yields

$$F'(x; c)N'(x) = F''(x; c)\times_2 (x - N(x)), \tag{10}$$

where $\overline{\times}_2$ is the 2-mode tensor-vector product.[37,38] Note that $x_* = N(x_*)$ and $F'(x_*; c)$ is nonsingular, so it follows from Equation (10) that $N'(x_*) = 0$.

Let $S(x) = \frac{N(x)}{\|N(x)\|}$, then

$$\|N(x)\| S'(x) = \left(I - S(x)S(x)^T\right) N'(x), \tag{11}$$

and thus $S'(x_*) = 0$ since $N'(x_*) = 0$ and $\|N(x_*)\| = \|x_*\| = 1$. Further, assume that

$$\tilde{N}(x) = S(x) - F'(x; c)^{-1} F(S(x); c),$$

which can be rewritten as

$$F'(x; c)(S(x) - \tilde{N}(x)) = F(S(x); c). \tag{12}$$

Obviously, $\tilde{N}(x_*) = x_*$ since $S(x_*) = x_*$. Finding the first and second derivatives on both sides of Equation (12) results in the following two equations:

$$F'(x; c)\tilde{N}'(x) = F'(x; c)S'(x) + F''(x; c)\overline{\times}_2 \left(S(x) - \tilde{N}(x)\right) - F'(S(x); c)S'(x), \tag{13}$$

and

$$\begin{aligned}\tilde{N}''(x)\times_1 F'(x; c) = {} & F''(x; c)\times_2 (S'(x) - \tilde{N}'(x))^T + F'''(x; c)\overline{\times}_2 \left(S(x) - \tilde{N}(x)\right) + F''(x; c)\hat{\times}_2 \left(S'(x) - \tilde{N}'(x)\right) \\ & + S''(x)\times_1 (F'(x; c) - F'(S(x); c)) - \left(F''(S(x); c)\times_3 S'(x)^T\right)\times_2 S'(x)^T,\end{aligned} \tag{14}$$

where $\times_k, k = 1, 2, 3$, denotes the k-mode tensor-matrix product,[37,38] and the "product" operation $\hat{\times}_2$ is defined in Appendix. Hence it follows, respectively, from Equations (13) and (14) that $\tilde{N}'(x_*) = 0$ and $\tilde{N}''(x_*) = 0$.

In addition, we define

$$\tilde{S}(x) = \frac{\tilde{N}(x)}{\|\tilde{N}(x)\|}. \tag{15}$$

Similar to the previous discussion, we can obtain that $\tilde{S}'(x_*) = 0$ and $\tilde{S}''(x_*) = 0$. Let

$$V = (I - x_* x_*^T) F'(x_*; c)^{-1}. \tag{16}$$

Then, in Appendix we prove that for any $u \in \mathbb{R}^n$,

$$\tilde{S}'''(x_*)u^3 = V\left(F''(x_*; c)\overline{\times}_3 u\right)\left(F''(x_*; c)\overline{\times}_3 u\right)^T V^T u + 2V\left(F''(x_*; c)\overline{\times}_3 u\right) V\left(F''(x_*; c)\overline{\times}_2 u \overline{\times}_3 u\right), \tag{17}$$

where $\overline{\times}_3$ is the 3-mode tensor-vector product.[37,38] According to Equation (7) and $Vx_* = 0$, we have

$$\begin{aligned} VF''(x_*; c)\overline{\times}_3 u &= V\left((m-1)(m-2)\mathcal{A}x_*^{m-3}\overline{\times}_3 u - \left(\lambda_R'(x_*)^T u\right) I - u\left(\lambda_R'(x_*)\right)^T + c(x_*^T u)I + cux_*^T\right) - V\left(x_*\left(\lambda_R''(x_*)u - cu\right)^T\right) \\ &= V\left((m-1)(m-2)\mathcal{A}x_*^{m-3}\overline{\times}_3 u + (c - (m-2)\lambda_*)\,(x_*^T u)I + (c - (m-2)\lambda_*)\,ux_*^T\right). \end{aligned} \tag{18}$$

and

$$\begin{aligned} VF''(x_*; c)\overline{\times}_2 u \overline{\times}_3 u &= V\left((m-1)(m-2)\mathcal{A}x_*^{m-3}\overline{\times}_2 u \overline{\times}_3 u - 2\left(\lambda_R'(x_*)^T u\right) u + 2c(x_*^T u)u\right) - V\left(\left(u^T \lambda_R''(x_*)u - c\|u\|^2\right) x_*\right) \\ &= V\left((m-1)(m-2)\mathcal{A}x_*^{m-3}\overline{\times}_2 u \overline{\times}_3 u + 2\,(c - (m-2)\lambda_*)\,(x_*^T u)u\right). \end{aligned} \tag{19}$$

Note that the matrix $F'(x_*; c)$ is symmetric, and thus $F'(x_*; c)^{-1}$ is also symmetric. Based on the expression (8) of $F'(x_*; c)$ and its exchangeability with $I - x_* x_*^T$, it is easy to see that V in Equation (16) is also symmetric and all of its eigenvalues are $0, \frac{1}{\theta_1}, \dots, \frac{1}{\theta_{n-1}}$. So

$$\|V\| = \max_{1 \le i \le n-1} \frac{1}{|\theta_i|} \le \frac{1}{\gamma}.$$

On the other hand, there exists a constant $G = G(\mathcal{A}) < \infty$ for any symmetric tensor $\mathcal{A}$ such that $\|\mathcal{A}x_*^{m-3}\overline{\times}_3 u\| \le G\|u\|$ and $\|\mathcal{A}x_*^{m-3}\overline{\times}_2 u \overline{\times}_3 u\| \le G\|u\|^2$. Hence, it follows from Equations (17) to (19) that

$$\begin{aligned} \|\tilde{S}'''(x_*)u^3\| &\le 3\|V\|^2((m-1)(m-2)G + 2|c - (m-2)\lambda_*|)^2\|u\|^3 \\ &\le \frac{3}{\gamma^2}((m-1)(m-2)G + 2|c - (m-2)\lambda_*|)^2\|u\|^3. \end{aligned} \tag{20}$$

Observe that $x_* = \tilde{S}(x_*)$ and $x^{(k+1)} = \tilde{S}(x^{(k)})$ for the MNNM method, therefore, using Taylor's expansion and Equation (20) we have

$$\begin{aligned} \|x^{(k+1)} - x_*\| &= \|\tilde{S}(x^{(k)}) - \tilde{S}(x_*)\| \\ &= \|\tilde{S}'(x_*)(x^{(k)} - x_*) + \frac{1}{2}\tilde{S}''(x_*)(x^{(k)} - x_*)^2 + \frac{1}{6}\tilde{S}'''(x_*)(x^{(k)} - x_*)^3 + O(\|x^{(k)} - x_*\|^4)\| \\ &= \|\frac{1}{6}\tilde{S}'''(x_*)(x^{(k)} - x_*)^3 + O(\|x^{(k)} - x_*\|^4)\| \\ &\le \frac{1}{2\gamma^2}((m-1)(m-2)G + 2|c - (m-2)\lambda_*|)^2\|x^{(k)} - x_*\|^3 + O(\|x^{(k)} - x_*\|^4) \\ &= M\|x^{(k)} - x_*\|^3 + O(\|x^{(k)} - x_*\|^4), \end{aligned} \tag{21}$$

where $M = \frac{1}{2\gamma^2}((m-1)(m-2)G + 2|c - (m-2)\lambda_*|)^2$. Hence, if we choose an initial vector $x^{(0)}$ such that $\|x^{(0)} - x_*\| \le \min(\epsilon, \frac{1}{2\sqrt{M}})$, then the sequence $\{x^{(k)}\}$ converges to x_* with the cubically convergent rate. The proof is completed. ∎

Obviously, the choice of the initial vector $x^{(0)}$ for the MNNM method is closely related to the regulation parameter c, that is, different parameter c may mean the different attraction region of the MNNM method (see Figure 1). Besides, we see from Theorem 2 that for the $\mathcal{Z}$-eigenvector x_* associated with the $\mathcal{Z}$-eigenvalue $\lambda_* = 0$, MNNM

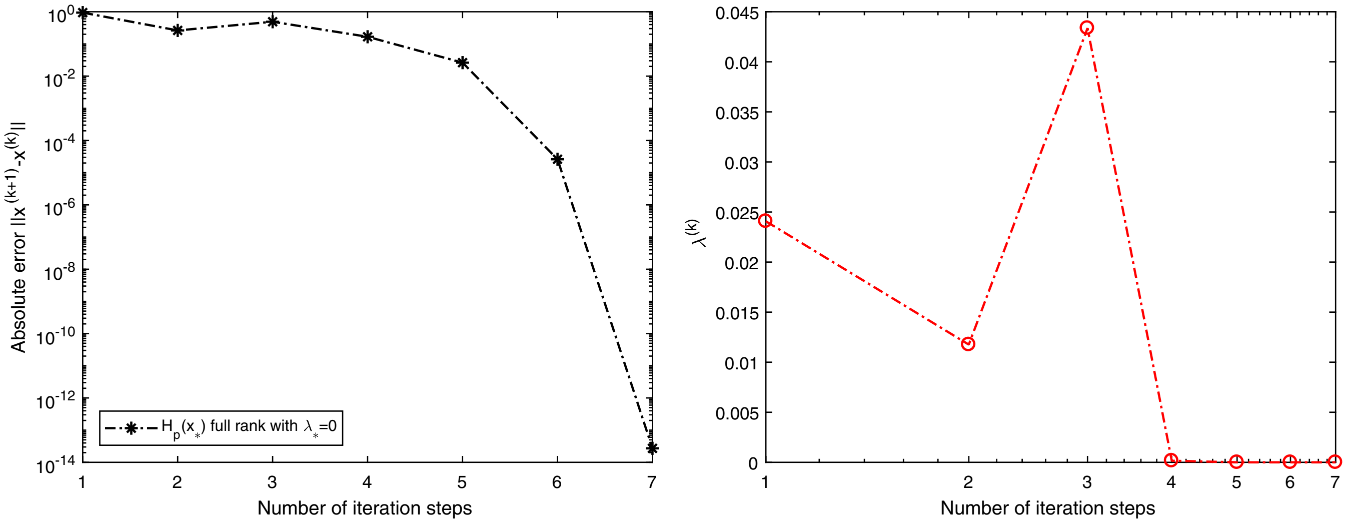


FIGURE 2 Convergence of modified normalized Newton method for Example 1 with the initial vector $x^{(0)} = (-0.8941, 0.4368, 0.0988)^T$

can guarantee its convergence under the assumptions of γ-Newton-stable and $c \neq 0$, which can also be illustrated numerically, see Example 1. It implies that our MNNM method, similar to the O-NCM method,[24] can avoid the defect of the NCM method[24] where $\lambda_* \neq 0$ is required. Jaffe et al.[24] pointed out that for a generic symmetric tensor, its all $\mathcal{Z}$-eigenpairs are γ-Newton-stable, where the meaning of the word "generic" is that there exists a polynomial ϕ in the entries of $\mathcal{A}$ such that if $\phi(\mathcal{A}) \neq 0$ then $\mathcal{A}$ has finitely many $\mathcal{Z}$-eigenvalues. This means that if MNNM is run multiple times using a sufficiently large number of random initial vectors, then it can be guaranteed to find all $\mathcal{Z}$-eigenpairs of a generic symmetric tensor. Meanwhile, Theorem 2 demonstrates theoretically that MNNM has better performance than NCM and O-NCM in the literature[24] since the latters are only quadratically convergent.

The following example shows that MNNM can be used to find a $\mathcal{Z}$-eigenvector x_* associated with the $\mathcal{Z}$-eigenvalue $\lambda_* = 0$.

Example 1. (See the works of Cui et al.[22] and Chen et al.[23])

Consider a symmetric tensor $\mathcal{A} \in \mathbb{RS}^{[6,3]}$ such that its corresponding polynomial form is the Motzkin polynomial:

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

By calculation, the $\mathcal{Z}$-eigenvector $x_* = (-0.5774, 0.5774, -0.5774)^T$ with the $\mathcal{Z}$-eigenvalue $\lambda_* = 0$ is γ-Newton-stable, namely, $H_p(x_*)$ is full rank. For this case, we take the initial vector $x^{(0)} = (-0.8941, 0.4368, 0.0988)^T$ to run MNNM. In Figure 2 (left), we depict that the value of $\|x^{(k)} - x^{(k-1)}\|$ evolves versus the number of iterations, while Figure 2 (right) shows the convergence of the iterative sequence $\{\lambda^{(k)}\}$ to the $\mathcal{Z}$-eigenvalue $\lambda_* = 0$. As one can see from Figure 2, MNNM converges to the $\mathcal{Z}$-eigenpair (λ_*, x_*), which confirms our theoretical analysis. However, NCM converges to the $\mathcal{Z}$-eigenvector $x = (-0.7071, 0.7071, 0)^T$ corresponding to the $\mathcal{Z}$-eigenvalue $\lambda = 0.25$ with the same initial vector. In order to show this superiority of MNNM to NCM, we have also tested NCM with thirty thousand different random initial vectors and found that it all converges to a $\mathcal{Z}$-eigenvector associated with the nonzero $\mathcal{Z}$-eigenvalue, namely, it seems that it is impossible for finding the $\mathcal{Z}$-eigenpair with a zero $\mathcal{Z}$-eigenvalue by the NCM method.

4 | NUMERICAL EXPERIMENTS

In this section, we present some numerical examples to illustrate the effectiveness and efficiency of our proposed method. All tests were performed on a desktop computer with 2.80 GHz CPU processor and 4.00GB RAM memory. We use the software MATLAB R2017a and the **zeig** procedure in the **TenEig** package.[23] In our experiments, the iterations are terminated once the absolute error $\|x^{(k)} - x^{(k-1)}\| < \delta = 10^{-10}$ or the maximum iteration $k_{\max} = 500$ arrives. If the latter case occurs, we declare that the method fails to converge.

4.1 | Numerical results for computing $\mathcal{Z}$-eigenpairs

The following examples show the performance of MNNM for finding all $\mathcal{Z}$-eigenpairs of a real symmetric tensor.

Example 2. (See the work of Jaffe et al.[24])

We consider fourth-order n-dimensional random Gaussian real symmetric tensors, whose entries are all i.i.d. $\mathcal{N}(0,1)$ up to the symmetry constraints. We test the time needed to find all $\mathcal{Z}$-eigenpairs for such random tensors with different dimensions by NCM and O-NCM in the literature[24] and MNNM, respectively.

For each tensor, we first ran **zeig** to obtain all $\mathcal{Z}$-eigenpairs, which would be served as a criterion to judge whether we actually found all tensors' $\mathcal{Z}$-eigenpairs. Next, we ran MNNM, NCM, and O-NCM by choosing different random initial vector $x^{(0)}$ until all $\mathcal{Z}$-eigenpairs of tensors were found. The process was sequential, that is, a new run was started only after the previous one ended. The initial vector $x^{(0)}$ is obtained by normalizing the vector **randn(n,1)** in MATLAB, and we use the same set of random initial vectors for each set of experiments. In addition, for this example, we implemented the MNNM method using $c = -800, -100, 50$, and 100, respectively. For each value of n, we ran 10 independent tensors, respectively.

Figure 3 (left) shows on a logarithmic scale, for each value of n the average time it took to find all $\mathcal{Z}$-eigenpairs of 10 independent tensors by NCM, O-NCM, and MNNM, respectively. Figure 3 (right) plots how the absolute error evolves versus the number of iterations of MNNM, NCM, and O-NCM, respectively, for the random Gaussian symmetric tensor with $m = 4$ and $n = 8$, at the initial vector

$$x^{(0)} = (0.1310, -0.5377, 0.5353, -0.1401, -0.2677, 0.1209, 0.0071, 0.5489)^T.$$

In this case, MNNM, NCM, and O-NCM can all find the $\mathcal{Z}$-eigenvalue $\lambda_* = 0.7583$ and the corresponding $\mathcal{Z}$-eigenvector

$$x_* = (0.1730, -0.2501, 0.3438, -0.1006, -0.5485, 0.2889, -0.0101, 0.6283)^T.$$

We see from Figure 3 that MNNM performs better than NCM and O-NCM. Especially, MNNM found all $\mathcal{Z}$-eigenpairs faster than NCM by approximately one orders of magnitude. The iteration number of MNNM is the least. In addition,

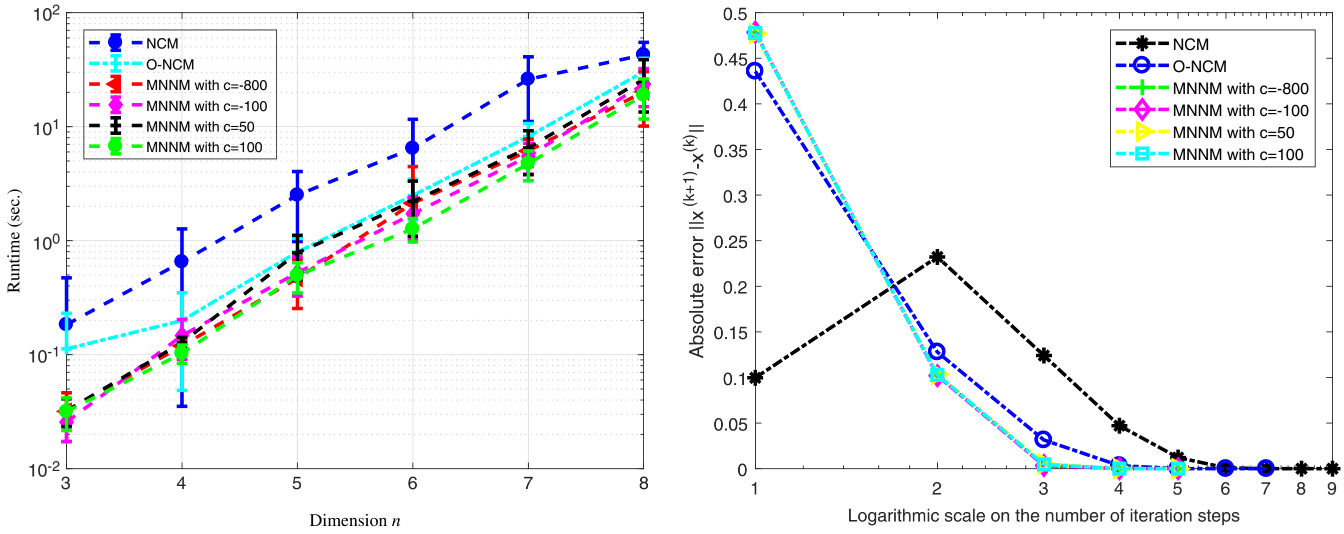


FIGURE 3 Left: The average time needed to compute all $\mathcal{Z}$-eigenpairs via modified normalized Newton method, Newton correction method and orthogonal Newton correction method. Right: The absolute error evolves versus the number of iterations for a random Gaussian symmetric tensor with $m = 4$ and $n = 8$ at the initial vector $x^{(0)} = (0.1310, -0.5377, 0.5353, -0.1401, -0.2677, 0.1209, 0.0071, 0.5489)^T$

from Figure 3, we see that under different values of c, MNNM has the same number of iteration and the runtime is slightly different.

Example 3. (See the work of Kofidis and Regalia[6])

Let $\mathcal{A} \in \mathbb{RS}^{[4,3]}$ be the symmetric tensor defined as

$$\begin{aligned}
&a_{1111} = 0.2883,\ a_{1112} = -0.0031,\ a_{1113} = 0.1973,\ a_{1122} = -0.2485,\ a_{1123} = -0.2939,\\
&a_{1133} = 0.3847,\ a_{1222} = 0.2972,\ a_{1223} = 0.1862,\ a_{1233} = 0.0919,\ a_{1333} = -0.3619,\\
&a_{2222} = 0.1241,\ a_{2223} = -0.3420,\ a_{2233} = 0.2127,\ a_{2333} = 0.2727,\ a_{3333} = -0.3054.
\end{aligned}$$

For this example, we normalize the vector **randn(3,1)** to get the random initial vector $x^{(0)}$. Figure 4 (left) plots the "runtime" of MNNM with different values of c. Here "runtime" denotes the average time in seconds for running 100 trials, where each trial for every algorithm can find all $\mathcal{Z}$-eigenpairs. As one can see, the runtime of MNNM is different as c varies, and is no more than 0.054. On the other hand, we obtain that the runtime of NCM and O-NCM is 0.4354 and 0.0902, respectively. Hence MNNM outperforms NCM and O-NCM.

To compare the convergence in terms of the number of iterations, Figure 4 (right) depicts how the absolute error evolves versus the number of iterations for MNNM, NCM, and O-NCM, respectively, at the initial vector $x^{(0)} = (0.1297, -0.7291, 0.6720)^T$. Here, we respectively set $c = -3500$ and $c = 100$ in the MNNM method. For this case, MNNM, NCM, and O-NCM can all find the $\mathcal{Z}$-eigenvalue $\lambda_* = 0.5105$ and the corresponding $\mathcal{Z}$-eigenvector $x_* = (0.3598, -0.7780, 0.5150)^T$. From Figure 4 (right), we see that MNNM has fewer iterations than NCM and O-NCM, and the number of iterations of MNNM is the same when $c = -3500$ and $c = 100$ (similar results are obtained for other values of c), namely, the number of iterations of MNNM does not vary with different values of c.

Example 4. (See the works of Cui et al.[22] and Chen et al.[23])

Consider the symmetric tensor $\mathcal{A} \in \mathbb{RS}^{[4,3]}$ such that

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

that is, its entries

$$a_{1111} = 1,\ a_{2222} = 2,\ a_{3333} = 3,\ a_{i_1i_2i_3i_4} = 0, \text{otherwise}.$$

For this example, the initial vector $x^{(0)}$ is obtained by normalizing the vector **-1+2rand(3,1)**. Figure 5 (left) plots the "runtime" of MNNM with different values of c, where the representation of "runtime" is the same as in Example 3.

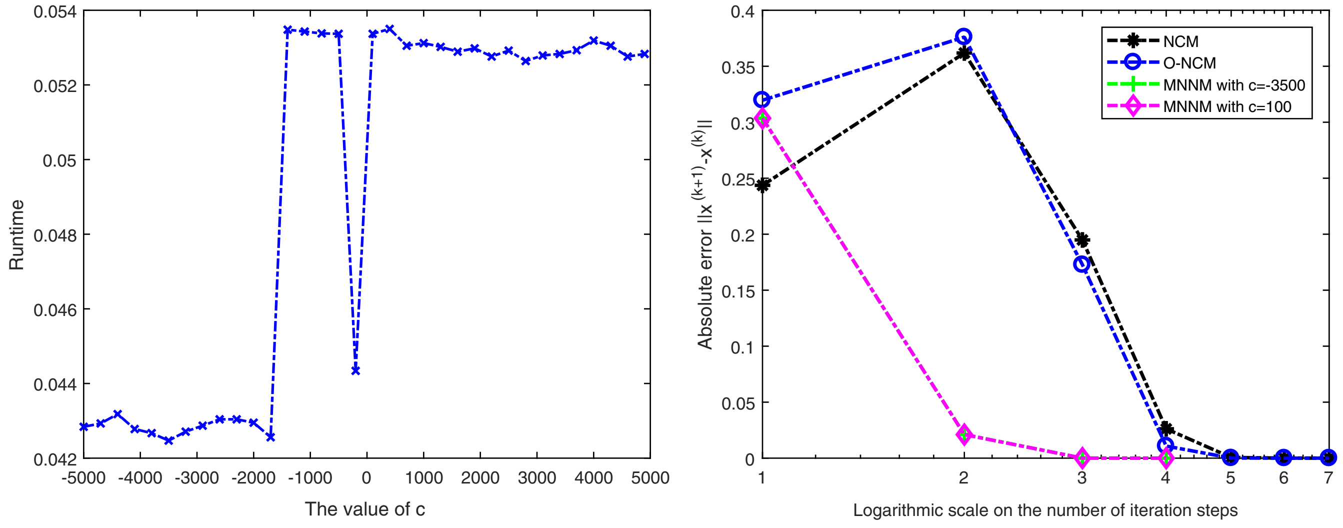


FIGURE 4 Left: the runtime of MNNM with different values of c. Right: the absolute error evolves versus the number of iterations for Example 3 at the initial vector $x^{(0)} = (0.1297, -0.7291, 0.6720)^T$

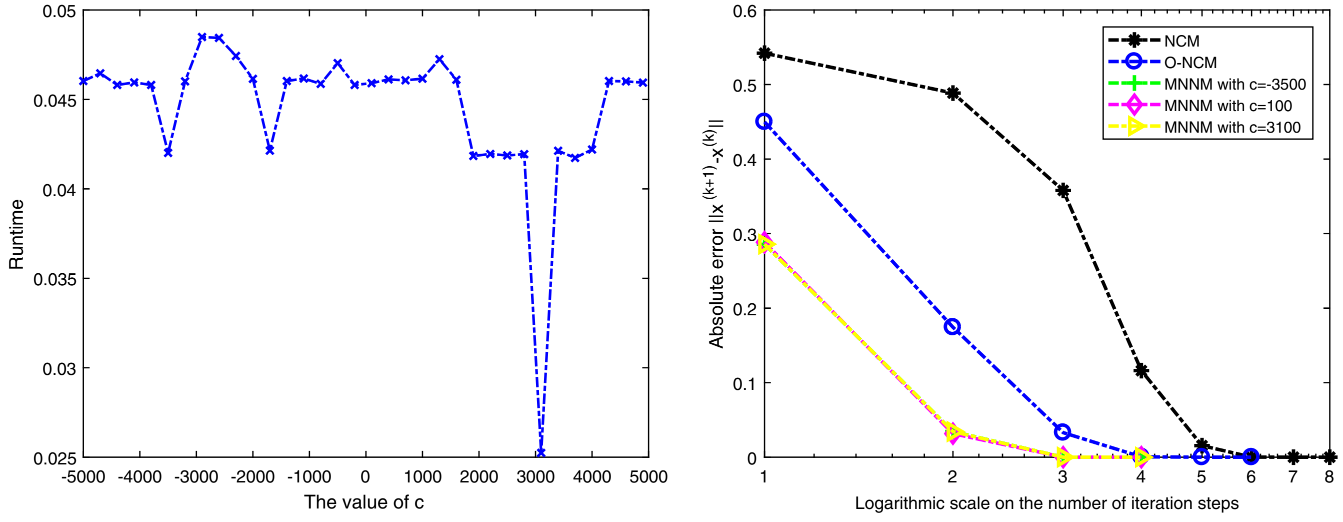


FIGURE 5 Left: the runtime of modified normalized Newton method with different values of c. Right: the absolute error evolves versus the number of iterations for Example 4 at the initial vector $x^{(0)} = (0.0596, 0.5400, 0.8395)^T$

Figure 5 (right) depicts how the absolute error evolves versus the number of iterations for NCM, O-NCM, and MNNM, respectively, at the initial vector $x^{(0)} = (0.0596, 0.5400, 0.8395)^T$. Here we choose $c = -3500, 100$, and 3100, respectively, to run the MNNM method. In this case, MNNM, NCM, and O-NCM can all find the $\mathcal{Z}$-eigenvalue $\lambda_* = 1.2000$ and the corresponding $\mathcal{Z}$-eigenvector $x_* = (0, 0.7746, 0.6325)^T$. As one can see, Figure 5 (left) shows that the runtime of MNNM is different for different values of c, and is no more than 0.05. Besides, we get that the runtime of NCM and O-NCM is 0.0656 and 0.1028, respectively. So MNNM is faster than NCM and O-NCM. Figure 5 (right) indicates that the number of iterations of MNNM is less than that of NCM and O-NCM, and is the same under different values of c (similar conclusion also holds for other c).

4.2 | Numerical results for application to US-eigenpairs

In this subsection, we apply MNNM given in Section 3 to compute the unitary symmetric eigenpairs (*US*-eigenpairs) of a complexed-valued symmetric tensor. The *US*-eigenpair was introduced by Ni et al.,[39] and is closely related to the geometric measure of quantum entanglement and the best complex rank-one approximation of a complexed-valued symmetric tensor.

Ni et al.[39] showed that for an mth-order n-dimensional complexed-valued symmetric tensor $\mathcal{C}$, (μ, x) is its *US*-eigenpair if and only if it satisfies

$$\begin{cases} \mathcal{C}\overline{x}^{m-1} = \mu x, \\ \mu \in \mathbb{R}, \overline{x}^T x = 1, x \in \mathbb{C}^n, \end{cases} \tag{22}$$

where $\overline{x}$ denotes the conjugate of x. Note that (μ, x) is a *US*-eigenpair of $\mathcal{C}$ if and only if $(\mu, \overline{x})$ is a Q-eigenpair of $\mathcal{C}$ defined in the work of Zhang and Qi.[40] Hence, using Theorem 4.2 in the work of Zhang and Qi[40] we can assert that for a tensor $\mathcal{C} \in \mathbb{CS}^{[m,n]}$, there exists a real symmetric tensor $\mathcal{T} \in \mathbb{RS}^{[m,2n]}$ such that (μ, x) is a *US*-eigenpair of $\mathcal{C}$ if and only if (μ, w) is a $\mathcal{Z}$-eigenpair of $\mathcal{T}$, where $x = y - z\sqrt{-1}, y$ and $x \in \mathbb{R}^n$, and $\mathrm{w} = (\mathrm{y}^\mathrm{T}\mathrm{z}^\mathrm{T})^\mathrm{T}$. Further suppose that $\mathcal{C} = \mathcal{A} + \mathcal{B}\sqrt{-1}$, then the entries $\mathcal{T}_{i_1 i_2 \ldots i_m}$ of the tensor $\mathcal{T}$ can be decided by the entries of $\mathcal{A}$ and $\mathcal{B}$, namely,

$$\mathcal{T}_{i_1 i_2 \ldots i_m} = \begin{cases} (-1)^j a_{\hat{i}_1 \hat{i}_2 \ldots \hat{i}_m} & \text{when there exist } 2j \ (0 \le j \le \frac{m}{2}) \text{ indices larger than } n, \\ (-1)^{j+1} b_{\hat{i}_1 \hat{i}_2 \ldots \hat{i}_m} & \text{when there exist } 2j+1 \ (0 \le j \le \frac{m-1}{2}) \text{ indices larger than } n, \end{cases} \tag{23}$$

where $\mathcal{A} = (a_{i_1 i_2 \dots i_m}) \in \mathbb{RS}^{[m,n]}$, $\mathcal{B} = (b_{i_1 i_2 \dots i_m}) \in \mathbb{RS}^{[m,n]}$ and

$$\hat{i}_k = \begin{cases} i_k & \text{if } i_k \le n, \\ n - i_k & \text{if } i_k > n. \end{cases} \quad (k = 1, 2, \dots, m.).$$

This illustrates that we can convert the *US*-eigenpair problem of a complex symmetric tensor to the $\mathcal{Z}$-eigenpair problem of a real symmetric tensor. Thus, MNNM given in Section 3 can be applied to compute *US*-eigenpairs of a complex symmetric tensors. Clearly, we arrange this as the following algorithm scheme which is denoted by MNNM-US for short.

Algorithm 3. (MNNM-US)

```
Step 0. Given an mth-order n-dimensional complex symmetric tensor $C = \mathcal{A} + \mathcal{B}\sqrt{-1}$, a scalar $c \neq 0$ and an initial vector $x_0 \in \mathbb{R}^{2n}$ with $\|x_0\| = 1$.
Step 1. Formulate the real symmetric tensor $\mathcal{T} \in \mathbb{RS}^{[m,2n]}$ by Equation (23).
Step 2. Compute the $\mathcal{Z}$-eigenpair $(\mu, w)$ of $\mathcal{T}$ by Algorithm 2.
Step 3. Set $y = (w_1, w_2, \dots, w_n)^T$, $z = (w_{n+1}, w_{n+2}, \dots, w_{2n})^T$ and $x = y - z\sqrt{-1}$, and output $(\mu, x)$ which is a US-eigenpair of $C$.
```

According to the convergence analysis of Algorithm 2, it is easy to see that MNNM-US is also locally and cubically convergent. Next, we give two examples to illustrate the efficiency and effectiveness of MNNM-US for computing *US*-eigenpairs of complex symmetric tensors. We compare MNNM-US with higher-order power type method (HOPTMC) described by Che et al.,[41] and show that MNNM-US is far superior to HOPTMC. In our tests, we set $c = 100$ to run MNNM-US, the initial vector of HOPTMC was chosen as $\hat{x}_0 = \mathbf{randn(2,1)} + \mathbf{i} * \mathbf{randn(2,1)}$ and the initial vector of MNNM-US was chosen as $x_0 = \begin{pmatrix} \mathbf{real}(\hat{x}_0) \\ \mathbf{imag}(\hat{x}_0) \end{pmatrix}$.

Example 5. (See the works of Ni et al.[39] and Che et al.[41])
Let $C \in \mathbb{CS}^{[3,2]}$ be the symmetric tensor defined as

$$C_{111} = 2, \ C_{112} = 1, \ C_{122} = -1, \ C_{222} = 1.$$

we ran 100 trials of HOPTMC and MNNM-US with different initial vectors. The results are summarized in Table 1. We know that if μ is a *US*-eigenvalue of a complex symmetric tensor, then $-\mu$ is also its *US*-eigenvalue, see Theorem 7 in the work of Ni et al.[39] Therefore, we list only the case that $\mu > 0$ in Table 1. As we see from Table 1, MNNM-US computes fifteen *US*-eigenpairs, while HOPTMV only finds six *US*-eigenpairs, which means that using MNNM-US can compute more *US*-eigenpairs than those by HOPTMC. We compare the number of iteration steps of computing *US*-eigenvalues $\mu_1 = 2.1745$ and $\mu_2 = 2.3547$ by MNNM-US and HOPTMC. The results indicate that we only need 6(4) iterations for computing $\mu_1 = 2.1745$ ($\mu_1 = 2.3547$) by MNNM-US when the predetermined precision $|\mu_{k+1} - \mu_k| < 10^{-10}$ is required, while HOPTMC needs 82 (44) iterations. This shows that MNNM-US converges much faster than HOPTMC. For clarity, Figure 6 shows the comparison results of HOPTMC and MNNM-US on 20 iterations for computing *US*-eigenvalues $\mu_1 = 2.1745$ and $\mu_2 = 2.3547$, respectively.

Next we use MNNM-US to compute the entanglement eigenvalue related to the geometric measure of quantum entanglement for symmetric pure states.

Example 6. (See the work of Wei and Goldbart[42])
Consider the pure state $|W\widetilde{W}(s,\phi)\rangle \equiv \sqrt{s}|W\rangle + \sqrt{1-s}\exp(\sqrt{-1}\phi)|\widetilde{W}\rangle$, where the states

$$|W\rangle = (|001\rangle + |010\rangle + |100\rangle)/\sqrt{3},$$

and

$$|\widetilde{W}\rangle = (|101\rangle + |011\rangle + |110\rangle)/\sqrt{3}.$$

TABLE 1 Finding unitary symmetric(*US*)-eigenpairs for Example 5.

The *US*-eigenpairs computed by HOPTMC	
***US*-eigenvalue**	**The corresponding *US*-eigenvector**
2.1745	$(-0.5071, 0.8619)^T$
2.1745	$(0.2536 - 0.4392\sqrt{-1}, -0.4309 + 0.7464\sqrt{-1})^T$
2.1745	$(0.2536 + 0.4392\sqrt{-1}, -0.4309 - 0.7464\sqrt{-1})^T$
2.3547	$(-0.4863 + 0.8423\sqrt{-1}, -0.1163 + 0.2014\sqrt{-1})^T$
2.3547	$(-0.4863 - 0.8423\sqrt{-1}, -0.1163 - 0.2014\sqrt{-1})^T$
2.3547	$(0.9726, 0.2327)^T$
The *US*-eigenpairs computed by MNNM-US	
***US*-eigenvalue**	**The corresponding *US*-eigenvector**
0.3264	$(0.3765, 0.9264)^T$
0.3264	$(-0.1883 + 0.3261\sqrt{-1}, -0.4632 + 0.8023\sqrt{-1})^T$
0.3264	$(-0.1883 - 0.3261\sqrt{-1}, -0.4632 - 0.8023\sqrt{-1})^T$
2.1213	$(-0.5417 + 0.4545\sqrt{-1}, 0.6645 + 0.2418\sqrt{-1})^T$
2.1213	$(-0.5417 - 0.4545\sqrt{-1}, 0.6645 - 0.2418\sqrt{-1})^T$
2.1213	$(-0.1228 + 0.6964\sqrt{-1}, -0.5417 - 0.4545\sqrt{-1})^T$
2.1213	$(-0.1228 - 0.6964\sqrt{-1}, -0.5417 + 0.4545\sqrt{-1})^T$
2.1213	$(0.6645 - 0.2418\sqrt{-1}, -0.1228 + 0.6964\sqrt{-1})^T$
2.1213	$(0.6645 + 0.2418\sqrt{-1}, -0.1228 - 0.6964\sqrt{-1})^T$
2.1745	$(-0.5071, 0.8619)^T$
2.1745	$(0.2536 - 0.4392\sqrt{-1}, -0.4309 + 0.7464\sqrt{-1})^T$
2.1745	$(0.2536 + 0.4392\sqrt{-1}, -0.4309 - 0.7464\sqrt{-1})^T$
2.3547	$(-0.4863 + 0.8423\sqrt{-1}, -0.1163 + 0.2014\sqrt{-1})^T$
2.3547	$(-0.4863 - 0.8423\sqrt{-1}, -0.1163 - 0.2014\sqrt{-1})^T$
2.3547	$(0.9726, 0.2327)^T$

Abbreviations: HOPTMC, higher-order power type method; MNNM-US, modified normalized Newton method-unitary symmetric.

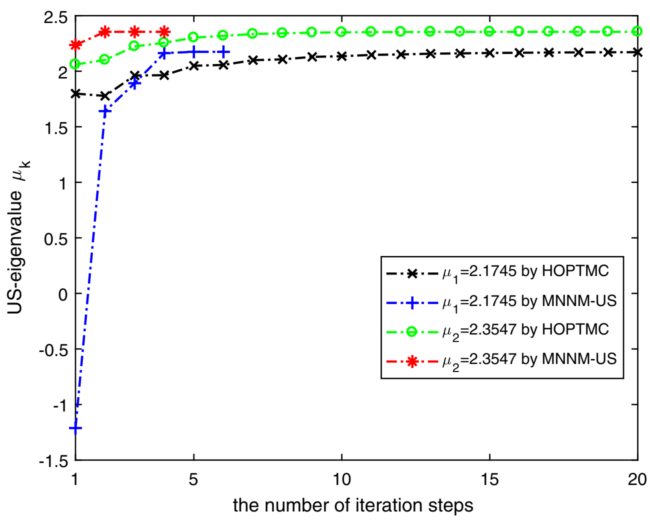


FIGURE 6 The comparison of modified normalized Newton method-unitary symmetric (US) and higher-order power type method for computing the *US*-eigenvalues $\mu_1 = 2.1745$ and $\mu_2 = 3.3547$, respectively.

TABLE 2 The comparison results on entanglement eigenvalues for Example 6

		MNNM-US			HOPTMC		
s	**The entanglement eigenvalue**	**Occ.**	**Its.**	**Time (s)**	**Occ.**	**Its.**	**Time (s)**
0.1	0.7933	15	6.5333	5.4025e-04	100	85.6100	0.0027
0.2	0.8306	21	5.8095	4.6880e-04	100	67.000	0.0021
0.3	0.8514	23	6.6087	5.3714e-04	100	59.6000	0.0019
0.4	0.8625	15	6.4667	5.2709e-04	100	56.6300	0.0018
0.5	0.8660	17	7.5294	6.2371e-04	100	55.5800	0.0018
0.6	0.8625	12	7.000	5.6609e-04	100	56.9400	0.0018
0.7	0.8514	20	6.7500	5.4518e-04	100	59.7100	0.0019
0.8	0.8306	16	6.9375	5.6195e-04	100	67.5600	0.0021
0.9	0.7933	12	8.1667	6.6061e-04	100	85.6000	0.0027

Abbreviations: HOPTMC, higher-order power type method; MNNM-US, modified normalized Newton method-unitary symmetric.

The entanglement is independent of ϕ: the transformation $\{|0\rangle, |1\rangle\} \rightarrow \{|0\rangle, \exp(\sqrt{-1}\phi)|1\rangle\}$ induces $|W\widetilde{W}(s,\phi)\rangle \rightarrow \exp(\sqrt{-1}\phi)|W\widetilde{W}(s,0)\rangle$. For this state, The corresponding tensor $\mathcal{C} \in \mathbb{CS}^{[3,2]}$ is given with the entries

$$C_{112} = C_{121} = C_{211} = \frac{\sqrt{3s}}{3},\ C_{212} = C_{122} = C_{221} = \frac{\sqrt{3-3s}}{3}\exp(\sqrt{-1}\phi).$$

As shown in the work of Ni et al.,[39] computing the entanglement eigenvalue of the state $|W\widetilde{W}(s,\phi)\rangle$ is equivalent to finding the largest *US*-eigenvalue of the complex symmetric tensor $\mathcal{C}$. Hence, we can use MNNM-US and HOPTMC to find the entanglement eigenvalues. For the case that $\phi = \pi$ and $s = 0.1, 0.2, \dots, 0.9$, we, respectively, ran 100 trials of MNNM-US and HOPTMC with different initial vectors. The results are listed in Table 2, where “Occ." refers to the number of occurrences in the 100 experiments, “Its." denotes the average number of iterations until convergence, and “Time (s)" means the average running time for finding the entanglement eigenvalues. As we see from Table 2, the number of occurrences of the entanglement eigenvalues by the MNNM-US method is less than that by the HOPTMC method (HOPTMC is similar to power method for the matrix case and it in general finds the modulus maximum eigenvalue), but MNNM-US performs much better than HOPTMC in terms of the number of iterations and running time.

5 | CONCLUSIONS

In this paper, we proposed a MNNM for computing $\mathcal{Z}$-eigenpairs of symmetric tensors, which can be viewed as a regularized version of the NCM method.[24] The proposed method is established by transforming the problem of computing $\mathcal{Z}$-eigenpairs of a symmetric tensor equivalently into the one of finding nonzero solutions of the nonlinear system of Equations (4). We show that MNNM locally and cubically converges to a $\mathcal{Z}$-eigenpair under the γ-Newton-stable assumption. Compared with the known NCM and O-NCM methods which only enjoy a quadratic rate of convergence, MNNM greatly improves the convergence rate with increasing only a small amount of computational cost. Also, similar to the O-NCM method,[24] MNNM overcomes the drawback of the NCM method that may fail to converge to a $\mathcal{Z}$-eigenpair with $\mathcal{Z}$-eigenvalue zero. We numerically show that the performance of the MNNM method can be affected by the involved parameter c. As a practical application of the MNNM method, we convert the problem of finding *US*-eigenpairs of a complex symmetric tensor to the one of finding $\mathcal{Z}$-eigenpairs of an appropriately larger real symmetric tensor, and then present another method, named MNNM-US, to compute such eigenpairs. We have tested the MNNM and MNNM-US methods with several numerically examples arising from the literature. The experimental results show that the MNNM method can be more efficiently used to find all $\mathcal{Z}$-eigenpairs of a real symmetric tensor than the known NCM and O-NCM methods in the literature[24] through choosing a large number of random initial values, and the resulted MNNM-US method for computing *US*-eigenpairs of a complex symmetric tensor can outperform the HOPTMC method.[41]

ACKNOWLEDGEMENTS

The authors would like to thank anonymous referees for their constructive suggestions and comments which greatly improve the presentation of this paper. The work is supported by the National Natural Science Foundation of China (Grant No. 11571004) and the Science Foundation of Education Department of Gansu Province (Grant No. 2017A-078).

CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

ORCID

Bing Zheng https://orcid.org/0000-0003-0552-835X

REFERENCES

1. Li W, Ng MK. On the limiting probability distribution of a transition probability tensor. Linear Multilinear Algebra. 2014;62:362–385.
2. Qi L, Yu GH, Wu EX. Higher order positive semidefinite diffusion tensor imaging. SIAM J Imag Sci. 2010;3:416–433.
3. Hu SL, Qi L, Zhang GF. The geometric measure of entanglement of pure states with nonnegative amplitudes and the spectral theory of nonnegative tensors. Department of Applied Mathematics, The Hong Kong Polytechnic University, 2012.
4. Che ML, Qi L, Wei Y, Zhang GF. Geometric measures of entanglement in multipartite pure states via complex-valued neural networks. Neurocomputing. 2018;313:25–38.
5. Ni Q, Qi L, Wang F. An eigenvalue method for testing positive definiteness of a multivariate form. IEEE Trans Automat Control. 2008;53:1096–1107.
6. Kofidis E, Regalia PA. On the best rank-1 approximation of higher-order supersymmetric tensors. SIAM J Matrix Anal Appl. 2002;23:863–884.
7. Zhang T, Golub GH. Rank-1 approximation to high order tensors. SIAM J Matrix Anal Appl. 2001;23:534–550.
8. Lathauwer LD, Moor BD, Vandewalle J. On the best rank-1 and rank-$(R_1, R_2, \dots, R_N)$ approximation of higher-order tensors. SIAM J Matrix Anal Appl. 2000;21:1324–1342.
9. Che ML, Cichocki A, Wei Y. Neural networks for computing best rank-one approximations of tensors and its applications. Neurocomputing. 2017;267:114–133.
10. Wang XZ, Che ML, Wei Y. Partial orthogonal rank-one decomposition of complex symmetric tensors based on the Takagi factorization. J Comput Appl Math. 2018;332:56–71.
11. Jaffe A, Weiss R, Carmi S, Kluger Y, Nadler B. Learning binary latent variable models: A tensor eigenpair approach. Paper presented at: Proceedings of the International Conference on Mach Learn; 2018.
12. Chang KC, Pearson K, Zhang T. On eigenvalue problems of real symmetric tensors. J Math Anal Appl. 2009;350:416–422.
13. Qi L. Eigenvalues of a real supersymmetric tensor. J Symb Comput. 2005;40:1302–1324.
14. Lim LH. Singular values and eigenvalues of tensors: a variational approach. In CAMSAP2005:1st IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing; 2005. p. 129–132.
15. Wei Y, Ding WY. Theory and computation of tensors. London, UK: Elsevier/Academic Press, 2016.
16. Qi L, Chen HB, Chen YN. Tensor eigenvalues and their applications. Berlin: Springer, 2018.
17. Ding WY, Wei Y. Generalized tensor eigenvalue problems. SIAM J Matrix Anal Appl. 2015;36:1073–1099.
18. Ding WY, Wei Y. Solving multi-linear systems with $\mathscr{M}$-tensors. J Sci Comput. 2016;68:689–715.
19. Qi L, Yu G, Xu Y. Nonnegative diffusion orientation distribution function. J Math Imaging Vision. 2013;45:103–113.
20. Guo CH, Lin WW, Liu CS. A modified Newton iteration for finding nonnegative $\mathscr{Z}$-eigenpairs of a nonnegative tensor. Numer Algor. 2019;80:595–616.
21. Kuo YC, Lin WW, Liu CS. Continuation methods for computing $\mathscr{Z}$-/$\mathscr{H}$-eigenpairs of nonnegative tensors. J Comput Appl Math. 2018;340:71–88.
22. Cui CF, Dai YH, Nie JW. All real eigenvalues of symmetric tensors. SIAM J Matrix Anal Appl. 2014;35:1582–1501.
23. Chen LP, Han LX, Zhou LM. Computing tensor eigenvalues via homotopy methods. SIAM J Matrix Anal Appl. 2016;37:290–319.
24. Jaffe A, Weiss R, Nadler B. Newton correction methods for computing real eigenpairs of symmetric tensors. SIAM J Matrix Anal Appl. 2018;39:1071–1094.
25. Hao LC, Cui CF, Dai YH. A sequential subspace projection method for extreme $\mathscr{Z}$-eigenvalues of supersymmetric tensors. Numer Linear Algebra Appl. 2015;22:283–298.
26. Hu SL, Huang ZH, Qi L. Finding the extreme $\mathscr{Z}$-eigenvalues of tensors via a sequential semidefinite programming method. Numer Linear Algebra Appl. 2013;20:972–984.
27. Kolda TG, Mayo JR. Shifted power method for computing tensor eigenpairs. SIAM J Matrix Anal Appl. 2011;32:1095–1124.
28. Kolda TG, Mayo JR. An adaptive shifted power method for computing generalized tensor eigenpairs. SIAM J Matrix Anal Appl. 2014;35:1095–1124.
29. Cartwright D, Sturmfels B. The number of eigenvalues of a tensor. Linear Algebra Appl. 2013;438:942–952.

30. Tapia RA, Dennis JE, Schäfermeyer JP. Inverse, shifted inverse, and Rayleigh quotient iteration as Newton's method. SIAM Rev. 2018;60:3–55.
31. Bahgat MSM. New two-step iterative methods for solving nonlinear equations. J Math Res. 2012;4:128–131.
32. Noor MA, Noor KI. Improved iterative methods for solving nonlinear equations. Appl Math Comput. 2007;184:270–275.
33. Chun CB. Iterative methods improving Newton's method by the decomposition method. Comput Math Appl. 2005;50:1559–1568.
34. Darvishi MT, Barati A. A third-order Newton-type method to solve systems of nonlinear equations. Appl Math Comput. 2007;187:630–635.
35. Waseema M, Noor MA, Noor KI. Efficient method for solving a system of nonlinear equations. Appl Math Comput. 2016;275:134–146.
36. Traub JF. Iterative methods for the solution of equations. Englewood Cliffs, NJ: Prentice-Hall, 1964.
37. Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Rev. 2009;51:455–500.
38. Che ML, Wei Y. Randomized algorithms for the approximations of Tucker and the tensor train decompositions. Adv Comput Math. 2019;45:395–428.
39. Ni GY, Qi L, Bai MR. Geometric measure of entanglement and U-eigenvalues of tensors. SIAM J Matrix Anal Appl. 2014;35:73–87.
40. Zhang XZ, Qi L. The quantum eigenvalue problem and $\mathscr{Z}$-eigenvalues of tensors; 2012. Preprint, arXiv:12051342.
41. Che ML, Qi L, Wei Y. Iterative algorithms for computing US- and U-eigenpairs of complex tensors. J Comput Appl Math. 2017;317:547–564.
42. Wei TC, Goldbart PM. Geometric measure of entanglement and applications to bipartite and multipartite quantum states. Phys Rev A. 2003;68:4343–4349.

How to cite this article: Zhao R, Zheng B, Liang M, Xu Y. A locally and cubically convergent algorithm for computing $\mathcal{Z}$-eigenpairs of symmetric tensors. *Numer Linear Algebra Appl*. 2020;e2284. https://doi.org/10.1002/nla.2284

APPENDIX

Let $\mathcal{A} = (a_{i_1 \dots i_m})$ be an mth-order n-dimensional tensor and $B = (b_{ij})$ be an $n \times n$ matrix. In what follows, the entries $a_{i_1 \dots i_m}$ and b_{ij} are also denoted by $\mathcal{A}_{i_1 \dots i_m}$ and B_{ij}, respectively, when necessary. In specific, if the entries of an mth-order n-dimensional tensor $\mathcal{A}$ are a function in variable x, we call it a function tensor, denoted by $\mathcal{A}(x)$. Naturally, its entries are denoted by $\mathcal{A}(x)_{i_1 \dots i_m}$. We use $\mathcal{A}'(x)$ to denote the derivative of $\mathcal{A}(x)$, and the entries of $\mathcal{A}'(x)$ are denoted by $\mathcal{A}'(x)_{i_1 \dots i_m}$. Same notations are also used for the matrix case.

In order to prove Equation (17), we need the following auxiliary lemma about the derivative of the different tensor-vector (-matrix) products.

Lemma 1. *Suppose that x is an n-dimensional vector, $A(x)$ and $B(x)$ are two $n \times n$ function matrices, $\alpha(x)$ is an n-dimensional function vector, $\mathcal{C}(x)$ is a third-order n-dimensional function tensor and $\mathcal{D}(x)$ is a fourth-order n-dimensional function tensor. Then*

$$\begin{aligned}
&(i) && (A(x)B(x))' = A'(x)\times_2 B(x)^T + B'(x)\times_1 A(x);\\
&(ii) && \left(\mathcal{C}(x)\overline{\times}_2\alpha(x)\right)' = \mathcal{C}'(x)\overline{\times}_2\alpha(x) + \mathcal{C}(x)\hat{\times}_2\alpha'(x);\\
&(iii) && (\mathcal{C}(x)\times_1 B(x))' = \mathcal{C}'(x)\times_1 B(x) + \mathcal{C}(x)\hat{\hat{\times}}_1 B'(x);\\
& && (\mathcal{C}(x)\times_2 B(x))' = \mathcal{C}'(x)\times_2 B(x) + \mathcal{C}(x)\hat{\hat{\times}}_2 B'(x);\\
& && \left(\mathcal{C}(x)\hat{\times}_2 B(x)\right)' = \mathcal{C}'(x)\hat{\otimes}_2 B(x) + \mathcal{C}(x)\odot B'(x);\\
&(iv) && \left(\mathcal{D}(x)\overline{\times}_2\alpha(x)\right)' = \mathcal{D}'(x)\overline{\times}_2\alpha(x) + \mathcal{D}(x)\hat{\times}_2\alpha'(x);\\
&(v) && (A(\alpha(x)))' = A'(\alpha(x))\times_3\alpha'(x)^T,
\end{aligned}$$

where $\mathcal{C}(x)\hat{\times}_2\alpha'(x)$ is a third-order n-dimensional tensor with the entries

$$\left(\mathcal{C}(x)\hat{\times}_2\alpha'(x)\right)_{ikl} = \sum_{j=1}^{n} \mathcal{C}(x)_{ijk}\alpha'(x)_{jl},$$

$\mathcal{D}(x)\hat{\times}_2\alpha'(x)$ is a fourth-order n-dimensional tensor with the entries

$$\left(\mathcal{D}(x)\hat{\times}_2\alpha'(x)\right)_{iktl} = \sum_{j=1}^{n} \mathcal{D}(x)_{ijkt}\alpha'(x)_{jl},$$

$\mathcal{C}(x)\hat{\hat{\times}}_1 B'(x)$ is a fourth-order n-dimensional tensor with the entries

$$\left(\mathcal{C}(x)\ \hat{\hat{\times}}_1\ B'(x)\right)_{ljkt} = \sum_{i=1}^{n} \mathcal{C}(x)_{ijk} B'(x)_{lit},$$

$\mathcal{C}(x)\hat{\hat{\times}}_2 B'(x)$ is a fourth-order n-dimensional tensor with the entries

$$\left(\mathcal{C}(x)\hat{\hat{\times}}_2 B'(x)\right)_{ljkt} = \sum_{i=1}^{n} \mathcal{C}(x)_{lik} B'(x)_{jit},$$

$\mathcal{C}'(x)\hat{\otimes}_2 B(x)$ is a fourth-order n-dimensional tensor with the entries

$$\left(\mathcal{C}'(x)\hat{\otimes}_2 B(x)\right)_{kljt} = \sum_{i=1}^{n} \mathcal{C}'(x)_{kilt} B(x)_{ij},$$

and $\mathcal{C}(x) \odot B'(x)$ is a fourth-order n-dimensional tensor with the entries

$$\left(\mathcal{C}(x) \odot B'(x)\right)_{kljt} = \sum_{i=1}^{n} \mathcal{C}(x)_{kil} B'(x)_{ijt}.$$

Proof. We only prove $(\mathcal{C}(x) \times_1 B(x))' = \mathcal{C}'(x) \times_1 B(x) + \mathcal{C}(x)\ \hat{\hat{\times}}_1\ B'(x)$ in (iii), while others' proofs are similar and so they are omitted.

Let $\mathcal{L}(x) = \mathcal{C}(x) \times_1 B(x)$ and $\mathcal{H}(x) = (\mathcal{C}(x) \times_1 B(x))'$. Then

$$\mathcal{H}(x) = (\mathcal{H}(x)_{ljkt}) = (\frac{\partial \mathcal{L}(x)_{ljk}}{\partial x_t}).$$

We set $\mathcal{C}'(x) = \left(\mathcal{C}'(x)_{ijkt}\right) = \left(\frac{\partial \mathcal{C}(x)_{ijk}}{\partial x_t}\right)$ and $B'(x) = \left(B'(x)_{lit}\right) = \left(\frac{\partial B(x)_{li}}{\partial x_t}\right)$, then

$$\begin{aligned}\mathcal{H}(x)_{ljkt} = \frac{\partial \mathcal{L}(x)_{ljk}}{\partial x_t} &= \sum_{i=1}^{n} \frac{\partial \mathcal{C}(x)_{ijk}}{\partial x_t} B(x)_{li} + \sum_{i=1}^{n} \mathcal{C}(x)_{ijk} \frac{\partial B(x)_{li}}{\partial x_t} \\ &= \sum_{i=1}^{n} \mathcal{C}'(x)_{ijkt} B(x)_{li} + \sum_{i=1}^{n} \mathcal{C}(x)_{ijk} B'(x)_{lit} \\ &= \left(\mathcal{C}'(x)\times_1 B(x)\right)_{ljkt} + \left(\mathcal{C}(x)\hat{\hat{\times}}_1 B'(x)\right)_{ljkt}.\end{aligned}$$

Hence, we have $(\mathcal{C}(x)\times_1 B(x))' = \mathcal{C}'(x)\times_1 B(x) + \mathcal{C}(x)\hat{\hat{\times}}_1 B'(x)$. The proof is completed. ∎

We now prove Equation (17).

Proof. We first compute $N''(x_*)$. by using two successive implicit differentiations on Equation (9) we have

$$N''(x)\times_1 F'(x;c) = -F''(x;c)\times_2 N'(x)^T + F'''(x;c)\times_2 (x - N(x)) + F''(x;c)\hat{\times}_2(I - N''(x)).$$

Since $N'(x_*) = 0$, $N(x_*) = x_*$, we get

$$N''(x_*)\times_1 F'(x_*;c) = F''(x_*;c).$$

Note that $F'(x_*; c)$ is nonsingular, then

$$\begin{aligned} N''(x_*) &= N''(x_*)\times_1 \left(F'(x_*; c)^{-1}F'(x_*; c)\right) \\ &= N''(x_*)\times_1 F'(x_*; c)\times_1 F'(x_*; c)^{-1} \\ &= F''(x_*; c)\times_1 F'(x_*; c)^{-1}. \end{aligned} \tag{A1}$$

Computing the first derivatives on both sides of Equation (11) in a similar manner, we have

$$S''(x_*) = N''(x_*)\times_1 (I - x_* x_*^T).$$

Thus it follows from Equation (A1) that

$$\begin{aligned} S''(x_*) &= F''(x_*; c)\times_1 F'(x_*; c)^{-1}\times_1 (I - x_* x_*^T) \\ &= F''(x_*; c)\times_1 \left((I - x_* x_*^T)F'(x_*; c)^{-1}\right). \end{aligned} \tag{A2}$$

Next, we compute $\tilde{N}'''(x_*)$. Finding the third derivatives on both sides of Equation (12) results in the following equation

$$\begin{aligned} \tilde{N}'''(x)\times_1 F'(x; c) = &-\tilde{N}''(x)\hat{\hat{\times}}_1 F''(x; c) + F'''(x; c)\times_2 (S'(x) - \tilde{N}'(x))^T + F''(x; c)\hat{\hat{\times}}_2 (S''(x) - \tilde{N}''(x)) \\ &+ F''''(x; c)\overline{\times}_2 \left(S(x) - \tilde{N}(x)\right) + F'''(x; c)\hat{\times}_2 \left(S'(x) - \tilde{N}'(x)\right) + F'''(x; c)\hat{\otimes}_2 \left(S'(x) - \tilde{N}'(x)\right) \\ &+ F''(x; c) \odot \left(S''(x) - \tilde{N}''(x)\right) + S'''(x)\times_1 (F'(x; c) - F'(S(x); c)) \\ &+ S''(x)\hat{\hat{\times}}_1 (F''(x; c) - F''(S(x); c)\times_3 S'(x)^T) - \left(F''(S(x); c)\times_3 S'(x)^T\right)'\times_2 S'(x)^T \\ &- \left(F''(S(x); c)\times_3 S'(x)^T\right)\hat{\hat{\times}}_2 \left(S''(x)\right)^T. \end{aligned}$$

Hence, using $\tilde{N}'(x_*) = 0$, $\tilde{N}''(x_*) = 0$, $S'(x_*) = 0$ and $\tilde{N}(x_*) = S(x_*) = x_*$ yields

$$\begin{aligned} \tilde{N}'''(x_*) = &F''(x_*; c)\hat{\hat{\times}}_2 S''(x_*)\times_1 F'(x_*; c)^{-1} + F''(x_*; c) \odot S''(x_*)\times_1 F'(x_*; c)^{-1} \\ &+ S''(x_*)\hat{\hat{\times}}_1 F''(x_*; c)\times_1 F'(x_*; c)^{-1}, \end{aligned} \tag{A3}$$

Similarly, we can get

$$\tilde{S}'''(x_*) = \tilde{N}'''(x_*)\times_1 (I - x_* x_*^T).$$

According to Equations (A2) and (A3) and the above equation, we have

$$\begin{aligned} \tilde{S}'''(x_*) &= F''(x_*; c)\hat{\hat{\times}}_2 S''(x_*)\times_1 V + F''(x_*; c) \odot S''(x_*)\times_1 V + S''(x_*)\hat{\hat{\times}}_1 F''(x_*; c) \times_1 V \\ &= F''(x_*; c)\hat{\hat{\times}}_2 \left(F''(x_*; c)\times_1 V\right)\times_1 V + F''(x_*; c) \odot \left(F''(x_*; c)\times_1 V\right)\times_1 V \\ &\quad + \left(F''(x_*; c)\times_1 V\right)\hat{\hat{\times}}_1 F''(x_*; c)\times_1 V, \end{aligned} \tag{A4}$$

where V is given in Equation (16). We now prove

$$F''(x_*; c)\hat{\hat{\times}}_2 \left(F''(x_*; c)\times_1 V\right)\times_1 V = \left(F''(x_*; c)\times_1 V\right)\hat{\hat{\times}}_2 \left(F''(x_*; c) \times_1 V\right). \tag{A5}$$

Let $C = F''(x_*; c)\hat{\hat{\times}}_2 \left(F''(x_*; c)\times_1 V\right)$. Then

$$\begin{aligned} (C\times_1 V)_{i_1 i_2 i_3 i_4} &= \sum_{j=1}^{n} C_{j i_2 i_3 i_4} V_{i_1 j} \\ &= \sum_{j=1}^{n}\sum_{t=1}^{n} F''(x_*; c)_{j t i_3} \left(F''(x_*; c)\times_1 V\right)_{i_2 t i_4} V_{i_1 j} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{t=1}^{n} \left(\sum_{j=1}^{n} F''(x_*;c)_{jti_3} V_{i_1 j} \right) \left(F''(x_*;c)\times_1 V\right)_{i_2 t i_4} \\
&= \sum_{t=1}^{n} \left(F''(x_*;c)\times_1 V\right)_{i_1 t i_3} \left(F''(x_*;c)\times_1 V\right)_{i_2 t i_4} \\
&= \left(\left(F''(x_*;c)\times_1 V\right) \hat{\hat{\times}}_2 \left(F''(x_*;c)\times_1 V\right) \right)_{i_1 i_2 i_3 i_4}.
\end{aligned}
$$

Hence, Equation (A5) holds. Similarly, we can prove that

$$
F''(x_*;c) \odot \left(F''(x_*;c)\times_1 V\right) \times_1 V = \left(F''(x_*;c)\times_1 V\right) \odot \left(F''(x_*;c)\times_1 V\right), \tag{A6}
$$

and

$$
\left(F''(x_*;c)\times_1 V\right) \hat{\hat{\times}}_1 F''(x_*;c)\times_1 V = \left(F''(x_*;c)\times_1 V\right) \hat{\hat{\times}}_1 \left(F''(x_*;c)\times_1 V\right). \tag{A7}
$$

According to Equations (A4) to (A7), we know

$$
\begin{aligned}
\tilde{S}'''(x_*) = {}& \left(F''(x_*;c)\times_1 V\right) \hat{\hat{\times}}_2 \left(F''(x_*;c)\times_1 V\right) + \left(F''(x_*;c)\times_1 V\right) \odot \left(F''(x_*;c)\times_1 V\right) \\
&+ \left(F''(x_*;c)\times_1 V\right) \hat{\hat{\times}}_1 \left(F''(x_*;c)\times_1 V\right).
\end{aligned} \tag{A8}
$$

Finally, we compute $\tilde{S}'''(x_*)u^3$ for any $u \in \mathbb{R}^n$. Let $\mathcal{K} = F''(x_*;c)\times_1 V$. Then

$$
\left(\left(F''(x_*;c)\times_1 V\right) \hat{\hat{\times}}_2 \left(F''(x_*;c)\times_1 V\right) \right) u^3 = \left(\mathcal{K} \hat{\hat{\times}}_1 \mathcal{K} \right) u^3.
$$

Since

$$
\begin{aligned}
\left(\left(\mathcal{K} \hat{\hat{\times}}_2 \mathcal{K} \right) u^3 \right)_i &= \sum_{i_2,i_3,i_4=1}^{n} \left(\mathcal{K} \hat{\hat{\times}}_2 \mathcal{K} \right)_{i i_2 i_3 i_4} u_{i_2} u_{i_3} u_{i_4} \\
&= \sum_{i_2,i_3,i_4=1}^{n} \left(\sum_{j=1}^{n} \mathcal{K}_{ij i_3} \mathcal{K}_{i_2 j i_4} \right) u_{i_2} u_{i_3} u_{i_4} \\
&= \sum_{j=1}^{n} \left(\sum_{i_3=1}^{n} \mathcal{K}_{ij i_3} u_{i_3} \right) \left(\sum_{i_2,i_4=1}^{n} \mathcal{K}_{i_2 j i_4} u_{i_2} u_{i_4} \right) \\
&= \sum_{j=1}^{n} \left(\mathcal{K} \overline{\times}_3 u \right)_{ij} \left(\mathcal{K} \overline{\times}_1 u \overline{\times}_3 u \right)_j \\
&= \left(\left(\mathcal{K} \overline{\times}_3 u \right) \left(\mathcal{K} \overline{\times}_1 u \overline{\times}_3 u \right) \right)_i,
\end{aligned}
$$

we have $\left(\mathcal{K} \hat{\hat{\times}}_2 \mathcal{K} \right) u^3 = \left(\mathcal{K} \overline{\times}_3 u \right) \left(\mathcal{K} \overline{\times}_1 u \overline{\times}_3 u \right)$. Thus,

$$
\begin{aligned}
\left(\left(F''(x_*;c)\times_1 V\right) \hat{\hat{\times}}_2 \left(F''(x_*;c)\times_1 V\right) \right) u^3 &= \left(\left(F''(x_*;c)\times_1 V\right) \overline{\times}_3 u \right) \left(\left(F''(x_*;c)\times_1 V\right) \overline{\times}_1 u \overline{\times}_3 u \right) \\
&= V \left(F''(x_*;c) \overline{\times}_3 u\right) \left(F''(x_*;c) \overline{\times}_3 u\right)^T V^T u.
\end{aligned} \tag{A9}
$$

Similarly, we have

$$
\left(\left(F''(x_*;c)\times_1 V\right) \hat{\hat{\times}}_1 \left(F''(x_*;c)\times_1 V\right) \right) u^3 = V \left(F''(x_*;c) \overline{\times}_3 u\right) V \left(F''(x_*;c) \overline{\times}_2 u \overline{\times}_3 u\right), \tag{A10}
$$

and

$$\left(\left(F''(x_*;c)\times_1 V\right)\odot\left(F''(x_*;c)\times_1 V\right)\right)u^3 = V\left(F''(x_*;c)\overline{\times}_3 u\right)V\left(F''(x_*;c)\overline{\times}_2 u\overline{\times}_3 u\right). \tag{A11}$$

Therefore, it follows from Equations (A8) to (A11) that

$$\tilde{S}'''(x_*)u^3 = V\left(F''(x_*;c)\overline{\times}_3 u\right)\left(F''(x_*;c)\overline{\times}_3 u\right)^T V^T u + 2V\left(F''(x_*;c)\overline{\times}_3 u\right)V\left(F''(x_*;c)\overline{\times}_2 u\overline{\times}_3 u\right).$$

The proof of Equation (17) is completed. ■