Computing tensor Z-eigenpairs via an alternating direction method

Genjiao Zhou[1], Shoushi Wang[1] and Jinhong Huang[1,2]

[1] Gannan Normal University, Ganzhou, China
[2] Gannan Normal University, Key Laboratory of Jiangxi Province for Numerical Simulation and Emulation Techniques, Ganzhou, China

ABSTRACT

Tensor eigenproblems have wide applications in blind source separation, magnetic resonance imaging, and molecular conformation. In this study, we explore an alternating direction method for computing the largest or smallest Z-eigenvalue and corresponding eigenvector of an even-order symmetric tensor. The method decomposes a tensor Z-eigenproblem into a series of matrix eigenproblems that can be readily solved using off-the-shelf matrix eigenvalue algorithms. Our numerical results show that, in most cases, the proposed method converges over two times faster and could determine extreme Z-eigenvalues with 20–50% higher probability than a classical power method-based approach.

INTRODUCTION

The tensor eigenproblem has been of great interest since the seminal works of *Qi (2005)* and *Lim (2005)*. It has numerous applications in several areas, including automatic control (*Ni, Qi & Wang, 2008*), magnetic resonance imaging (*Qi, Yu & Wu, 2010*; *Qi, Yu & Xu, 2013*; *Schultz & Seidel, 2008*), statistical data analysis (*Zhang & Golub, 2001*), image analysis (*Zhang, Zhou & Peng, 2013*), signal processing and navigation (*Kofidis & Regalia, 2001*; *Ashourian & Sharifi-Tehrani, 2022*; *Sharifi-Tehrani & Sabahi, 2022*; *Sharifi-Tehrani, Sabahi & Danaee, 2021*), and higher-order Markov chains (*Li & Ng, 2014*).

Unlike for matrices, there are several definitions of tensor eigenvalues and corresponding eigenvectors. For example, *Qi (2005)* proposed the definition of an H-eigenvalue and Z-eigenvalue as being equivalent to the l^m-eigenvalue and l^2-eigenvalue in *Lim (2005)*, respectively. In *Chang, Pearson & Zhang (2009)*, these definitions were unified by employing a positive definite tensor $\mathcal{B}$ while m was even. In this work, we mainly focus on computing Z-eigenvalues of symmetric tensors.

In general, the calculation of all eigenvalues of a higher-order tensor is very difficult due to the NP-hardness of deciding tensor eigenvalues over $\mathbb{R}$ (*Hillar & Lim, 2013*). Fortunately, one only needs to compute the largest or smallest eigenvalue of a tensor in certain scenarios. For instance, to guarantee the positive definiteness of the diffusivity function in higher-order diffusion tensor imaging, we just need to compute the smallest Z-eigenvalue of the tensor and make sure it is nonnegative. In automatic control (*Ni,*

Corresponding author
Jinhong Huang, hjhmaths@163.com

Qi & Wang, 2008), the smallest Z-eigenvalue of a tensor is used to determine whether a nonlinear autonomous system is stable or not. According to the Perron–Frobenius theory, the spectral radius of a nonnegative tensor is the largest Z-eigenvalue of the tensor (*Chang, Pearson & Zhang, 2008*).

To obtain the extreme eigenvalues of a symmetric tensor, *De Lathauwer, De Moor & Vandewalle (2000)* introduced a symmetric higher-order power method (S-HOMP). However, it was pointed out by *Kofidis & Regalia (2002)* that the S-HOMP method is not guaranteed to converge while the objective function is not convex. To address this problem, *Kolda & Mayo (2011)* presented a shifted S-HOMP (SS-HOMP) for solving the eigenproblem, which is guaranteed to converge to a tensor eigenpair. A major limitation of SS-HOMP is the difficulty in selecting an appropriate shift. Hence, *Kolda & Mayo (2014)* further extended the SS-HOMP method to an adaptive version for computing extreme eigenvalues, called GEAP, which chooses the shift automatically.

Over the past few years, there has been extensive work on handling the extreme eigenvalue problem of symmetric tensors by solving different nonlinearly constrained models. *Hu, Huang & Qi (2013)* proposed a sequential semidefinite relaxations approach to compute extreme Z-eigenvalues. *Han (2012)* employed the BFGS method to solve an unconstrained optimization problem for finding real eigenvalues of even-order symmetric tensors. *Cui, Dai & Nie (2014)* computed all of the real Z-eigenvalues of symmetric tensors using a Jacobian semidefinite relaxation technique. Using the method proposed by *Qi, Wang & Wang (2009)* in which Z-eigenpairs are computed directly in a lower dimensional case, a sequential subspace projection method (SSPM) (*Hao, Cui & Dai, 2015*) was proposed to obtain the extreme Z-eigenvalues of symmetric tensors. All the methods mentioned above converge linearly or superlinearly. To speed up convergence, *Jaffe, Weiss & Nadler (2018)* presented a fast iterative Newton-based method that converges at a locally quadratic rate. Based on the idea of the SSPM method (*Hao, Cui & Dai, 2015*), *Yu et al. (2016)* proposed an adaptive gradient (AG) method in which an inexact line search, rather than an optimal stepsize, was adopted. The experimental results presented in *Yu et al. (2016)* showed that the AG method converges much faster and finds the extreme eigenvalues with a higher probability than those methods using power algorithms. For more related work, we refer readers to *Benson & Gleich (2019)*, *Chen, Han & Zhou (2016)*, *Sheng & Ni (2021)*, *Xiong et al. (2022)*, and references therein.

Despite the fact that the extreme eigenvalue problem has drawn a lot of attention in recent years, there are still some issues to address. For example, all algorithms mentioned above are not guaranteed to converge to the largest or smallest eigenvalue, which is exactly what we want to obtain in some applications (*Chang, Pearson & Zhang, 2008*; *Ni, Qi & Wang, 2008*), and instead only converge to an arbitrary eigenvalue of $\mathcal{A}$ depending on the initial conditions. However, in the case of symmetric matrices, those counterpart algorithms can always converge to the largest or smallest eigenvalue. Motivated by this, we propose to determine extreme eigenvalues by combining the method of solving matrix eigenvalue problems and tensor optimization techniques. To this end, we develop a novel method for computing the largest or smallest Z-eigenvalue of symmetric tensors using the variable splitting method.

The main contributions of this work are listed as follows:

- We reformulate a typical tensor Z-eigenvalue problem as an equivalent multi-variable linearly constrained problem using the variable splitting method, which has a structure similar to the matrix eigenvalue problem.
- We design an efficient algorithm to solve the reformulated problem, in which the tensor Z-eigenproblem is decomposed into a series of matrix eigenproblems that has been extensively studied and can be readily solved using off-the-shelf matrix eigenvalue algorithms.

The remainder of the article is organized as follows. In the next section, we formulate the tensor eigenproblem and introduce some classical methods for solving it. In Section 3, we propose a simple and efficient algorithm for the problem and analyze its convergence property. Section 4 reports some experimental results to show the efficiency of our proposed method. Finally, we conclude this article in Section 5.

PROBLEM FORMULATION AND RELATED WORK

Problem formulation

An mth order n-dimensional real tensor consisting of n^m entries in $\mathbb{R}$:

$$\mathcal{A} = \left(a_{i_1 i_2 \cdots i_m}\right), a_{i_1 i_2 \cdots i_m} \in \mathbb{R}, 1 \leq i_1, i_2, \ldots, i_m \leq n.$$

$\mathcal{A}$ is called symmetric if the value of $a_{i_1 i_2 \cdots i_m}$ is invariant under any permutation of its indices $i_1, i_2, \ldots, i_m$. For convenience, we use $\mathbb{S}^{[m,n]}$ to denote the set of all m th order n-dimensional real symmetric tensors.

Throughout this article, we use $\mathcal{A}x^{m-k}(0 \leq k \leq m)$ to simply denote a k th order n-dimensional tensor defined by

$$\left(\mathcal{A}x^{m-k}\right)_{i_1 i_2 \cdots i_k} = \sum_{i_{k+1}, \ldots, i_m = 1}^{n} a_{i_1 i_2 \cdots i_k i_{k+1} \cdots i_m} x_{i_{k+1}} \cdots x_{i_m}, \tag{1}$$

for all $1 \leq i_1, i_2, \ldots, i_k \leq n$ *and* $1 \leq k \leq m$.

Obviously, $\mathcal{A}x^m$ is a scalar, $\mathcal{A}x^{m-1}$ is a vector, and $\mathcal{A}x^{m-2}$ is a matrix. For brevity, let $\mathcal{A}x^p y^q$ denote the result of $\left(\mathcal{A}x^p\right)y^q = \left(\mathcal{A}x^{m-(m-p)}\right)y^{m-p-(m-p-q)}$, where $0 \leq p, q, p+q \leq m$, and $\mathcal{A}x_1^{p_1} x_2^{p_2} \cdots x_2^{p_k}$ can be computed in a similar way, where $0 \leq p_1, p_2, \ldots, p_k \leq m$, and k is an arbitrary integer such that $0 \leq p_1 + p_2 + \cdots + p_k \leq m$.

Using the definition of Eq. (1), an mth degree homogeneous polynomial function $f(x)$ with real coefficients can be represented by a symmetric tensor $\mathcal{A}$, *i.e.*, $f(x) = \mathcal{A}x^m$. We call $\mathcal{A}$ positive definite tensor if $\mathcal{A}x^m > 0$ for all $x \in \mathbb{R}^n \setminus \{0\}$. It is easy to understand that m must be even in this case.

As mentioned before, there are several definitions of tensor eigenvalues and corresponding eigenvectors. In this work, we mainly focus on computing Z-eigenvalues of symmetric tensors defined as follows.

Definition 1 (*Qi, 2005*). Let $\mathcal{A}$ be an m th order n-dimensional symmetric real tensor. If there exists a nonzero vector $x \in \mathbb{R}^n$ and a scalar $\lambda \in \mathbb{R}$ satisfying

$$\begin{cases} \mathcal{A}x^{m-1} = \lambda x, \\ x^T x = 1, \end{cases} \tag{2}$$

then we call the scalar λ a Z-eigenvalue of $\mathcal{A}$, and the vector x a Z-eigenvector associated with the Z-eigenvalue λ. We also say the pair (λ, x) is a Z-eigenpair of $\mathcal{A}$.

Iterative algorithms to find the largest or smallest eigenvalues and corresponding eigenvectors are usually designed to solve a nonlinearly constrained optimization problem

$$\max f(x) = \mathcal{A}x^m$$
$$s.t. x \in \mathbb{S}^{n-1}, \tag{3}$$

where $\mathbb{S}^{n-1}$ denotes the unit sphere in the Euclidean norm, *i.e.,* $\mathbb{S}^{n-1} = \{x \in \mathbb{R}^n | \|x\|^2 = 1\}$. We can determine the gradient and Hessian of the objective function of Eq. (3) through some simple calculations, as follows:

$$g(x) \equiv \nabla f(x) = m\mathcal{A}x^{m-1} \tag{4}$$

and

$$H(x) \equiv \nabla^2 f(x) = m(m-1)\mathcal{A}x^{m-2} \tag{5}$$

Some existing methods for Z-eigenproblems

In this subsection, we introduce some typical methods for computing Z-eigenpairs by solving the problem Eq. (3) or its variants. From Theorem 3.2 of *Kolda & Mayo (2011)*, we know that (λ, x) is a Z-eigenpair of $\mathcal{A}$ if and only if x is a constrained stationary point of Eq. (3) and $\lambda = \mathcal{A}x^m / \|x\|$. Based on the theorem, *De Lathauwer, De Moor & Vandewalle (2000)* proposed the S-HOPM method for solving the problem Eq. (3) to find the best symmetric rank-1 approximation of a symmetric tensor $\mathcal{A} \in \mathbb{S}^{[m,n]}$, which is equivalent to finding the largest Z-eigenvalue of $\mathcal{A}$ (*Qi, 2005*). The main step of the S-HOPM algorithm is

$$x_{k+1} = \frac{\mathcal{A}x_k^{m-1}}{\|\mathcal{A}x_k^{m-1}\|}, \lambda_{k+1} = \mathcal{A}x_{k+1}^m. \tag{6}$$

Under the assumption of convexity on $\mathcal{A}x^m$, S-HOPM could be convergent for even-order tensors. However, it has been pointed out that S-HOPM can not guarantee to converge globally (*Kofidis & Regalia, 2002*). To address this issue, *Kolda & Mayo (2011)* modified the objective function to

$$\hat{f}(x) = \mathcal{A}x^m + \alpha \|x\|^m, \tag{7}$$

and proposed the SS-HOPM for solving Eq. (3) with the objective function Eq. (7). SS-HOPM has a similar iterative scheme to S-HOPM, but at the same time has a shortcoming

in the choice of the shift α. To overcome the limitation, the same authors proposed an adaptive method, called GEAP, which is monotonically convergent and much faster than the SS-HOPM method due to its adaptive shift choice of the shift. GEAP is originally designed to calculate generalized eigenvalues (*Chang, Pearson & Zhang, 2009*) with a positive definite tensor $\mathcal{B}$. The authors also presented a specialization of the method to the Z-eigenvalue problem, which is equivalent to SS-HOPM except for the adaptive shift. The details of the GEAP specialization are briefly summarized in Algorithm 1.

Algorithm 1. GEAP method for the problem Eq. (3) with the objective function Eq. (7)

Initialization: Given a tensor $\mathcal{A} \in S^{[m,n]}$, an initial vector $x_0 \in \mathbb{R}^n$, and a tolerance $\epsilon > 0$. Let $\beta = 1$ if we want to compute the largest Z-eigenvalue, and let $\beta = -1$ if we want to compute the smallest Z-eigenvalue. Let τ be the tolerance on being positive/negative definite.

1: $x_0 \leftarrow x_0 / \|x_0\|$, and $\lambda_0 \leftarrow \mathcal{A}x_0^m$
For $k = 0, 1, \cdots$ **do**
    2: $H_k \leftarrow m(m-1)\mathcal{A}x_k^{m-2}$
    3: $\alpha_k \leftarrow \beta \max 0, (\tau - \lambda_{min}(\beta H_k))/m$
    4: $x_{k+1} \leftarrow \beta(\mathcal{A}x_k^{m-1} + \alpha x_k)$
    5: $x_{k+1} = x_{k+1}/\|x_{k+1}\|$
    6: $\lambda_{k+1} \leftarrow \mathcal{A}x_{k+1}^m$
    7: Break if $|\lambda_{k+1} - \lambda_k| < \epsilon$
End for
Output: Z-eigenvalue λ and its associated Z-eigenvector x.

GEAP is a simple and effective approach for computing Z-eigenvalues of a symmetric tensor, but it is not guaranteed to determine the largest eigenvalue or the smallest one, which is exactly the goal in some applications. To obtain these extreme eigenvalues with a higher probability, we propose to reformulate the problem Eq. (3) to make its structure similar to a matrix eigenproblem, so that it can be solved by existing methods for matrices that always converge to extreme eigenvalues.

PROPOSED METHOD

An alternating direction method for Z-eigenproblems

Motivated by that algorithms for solving matrix eigenproblem can always converge to the largest or smallest eigenvalue, we propose to compute extreme eigenvalues by combining the method of solving the matrix eigenproblem and tensor optimization techniques. To this end, we adopt a variable splitting strategy in which we introduce some superfluous variables and equality constraints over these variables. Specifically, the term $\mathcal{A}x^m$ with even number m is rewritten as $\mathcal{A}x_1^2 x_2^2 \cdots x_p^2$, where $p = m/2$, with the equality constraints $x_i = x_j (i, j = 1, 2, \ldots, p)$. Therefore, problem Eq. (3) is transformed into the following model:

$$\max \tilde{f}(x) = \mathcal{A}x_1^2 x_2^2 \cdots x_p^2$$
$$s.t. \, x_i = x_j, i, j = 1, 2, \ldots, p \qquad\qquad (8)$$
$$x_i \in \mathbb{S}^{n-1}, i = 1, 2, \ldots, p.$$

When $\mathcal{A}$ is symmetric and conditions $x_i = x_j = x(i, j = 1, 2, \ldots, p)$ hold, we can obtain $\mathcal{A}x_1^2 x_2^2 \cdots x_p^2 = \mathcal{A}x^m$. Using this fact, the equivalence between problems Eqs. (3) and (8) can be easily checked. It is also worthwhile to note that if all variables except x_i are available and those equality constraints are not considered, the problem Eq. (8) reduces to the standard matrix eigenproblem for the matrix $\mathcal{A}x_1^2 \cdots x_{i-1}^2 x_{i+1}^2 \cdots x_p^2$.

Directly solving the problem Eq. (8) may be inefficient because its special structure is not considered, and in doing so, it is easy to converge to a locally optimal point, thus the largest or smallest eigenvalue could not be determined. On the other hand, it is comparatively easy to compute extreme eigenvalues for the matrix cases. In Eq. (8), if all variables except x_i are known and those equality constraints are not considered, solving Eq. (8) can exactly get the largest eigenvalue and the corresponding eigenvector of the matrix $\mathcal{A}x_1^2 \cdots x_{i-1}^2 x_{i+1}^2 \cdots x_p^2$. Following this observation and the fact that there are many efficient algorithms available for tackling matrix eigenproblems, we propose a simple alternating direction scheme between solving different matrix eigenproblems for the problem Eq. (8). The details of this method are given in Algorithm 2.

Algorithm 2 Alternating direction method (ADM) for Eq. (8)

Initialization: Given an even-order tensor $\mathcal{A} \in S^{[m,n]}$, initial unit vectors $x_i \in \mathbb{R}^n, i = 1, 2, \ldots, p$, where $p = m/2$, and $\epsilon > 0$ is the tolerance. Set $x = x_p, \lambda = \mathcal{A}x^m$, and δ as the absolute difference between successive values of λ.

While $\delta > \epsilon$

    For $i = 1, 2, \ldots, p$ do

        1: Compute the matrix $A = \mathcal{A}x_1^2 \cdots x_{i-1}^2 x_{i+1}^2 \cdots x_p^2$.

        2: Find the largest or smallest eigenvalue $\tilde{\lambda}$ and the corresponding unit eigenvector v of A using any eigenvalue algorithm for matrices.

        3: Update the variable $x_i = v$.

    End for

    4: Set $x = x_p$ and $\lambda = \mathcal{A}x^m$.

End while

Output: Z-eigenvalue λ and its associated Z-eigenvector x.

The main computational cost lies in tensor-vector multiplications $\mathcal{A}x_1^2 \cdots x_{i-1}^2 x_{i+1}^2 \cdots x_p^2$ and matrix eigenvalue computations. For an mth order n-dimensional symmetric tensor $\mathcal{A}$, it costs $O(mn^m)$ operations to compute the matrix $\mathbf{A} = \mathcal{A}x_1^2 \cdots x_{i-1}^2 x_{i+1}^2 \cdots x_p^2$. The cost of computing the largest or smallest eigenvalue of the matrix $\mathbf{A}$ is $(4/3)n^3$, which is much less than the products for large n. Compared with GEAP, which has similar computations

to our method but needs to compute tensor multiplication at least twice in each iteration, our method can save about half of the time because it only needs to calculate tensor multiplication once in each iteration. This can be verified in the numerical experiments in Section 4.

Specialization of ADM to fourth-order tensors

The proposed ADM transforms the tensor eigenvalue problem Eq. (3) into a series of matrix eigenvalue problems that are easy to solve. For fourth-order tensors, there are two related variables of x_1 and x_2, and the inner iteration can be omitted because $p = 1$. According to the symmetry property of $\mathcal{A}$, we also have $\mathcal{A}x_1^2 x_2^2 = \mathcal{A}x_2^2 x_1^2$. Therefore, it is not necessary to explicitly write out the variable x_2, and the procedure of Algorithm 2 can be simply described, as shown in Algorithm 3, for fourth-order tensors. To better describe the iterative steps, we use x_k to denote a kth iterate in Algorithm 3, rather than the splitting variable as in Algorithm 2.

Algorithm 3 Specialization of the ADM to fourth-order tensors

Initialization: Given a tensor $\mathcal{A} \in S^{[4,n]}$, initial unit vectors $x_0 \in \mathbb{R}^n$, and $\epsilon > 0$ is the tolerance. Set $\lambda_0 = \mathcal{A}x_0^m$, $k := 0$, and δ as the absolute difference between successive values of λ.

For $k = 0, 1, 2 \cdots$ **do**
    1: Compute the matrix $A_k = \mathcal{A}x_k^2$.
    2: Find the largest or smallest eigenvalue $\tilde{\lambda}$ and the corresponding unit eigenvector v of A_k using any eigenvalue algorithm for matrices.
    3: Update the variable $x_{k+1} = v$ and the eigenvalue $\lambda_{k+1} = \tilde{\lambda}$.
    4: Break if $|\lambda_{k+1} - \lambda_k| < \epsilon$, set $k = k + 1$.
End for
Output: Z-eigenvalue λ_{k+1} and its associated Z-eigenvector x_{k+1}.

Convergence analysis

As shown in the main steps of Algorithm 2 and Algorithm 3, the equality constraints in Eq. (8) are not considered in the process of calculation. A natural question arises about whether the algorithms can converge, and furthermore, whether the algorithms can converge to a Z-eigenvalue of $\mathcal{A}$. In this subsection, we handle these issues using properties of extreme eigenvalues and corresponding eigenvectors of matrices. For simplicity, only the convergence property of Algorithm 3 is analyzed. The convergence property of Algorithm 2 can be analyzed in a similar way.

    Let x_k denote the kth iterate generated by Algorithm 3. According to Steps 2 and 3, in the case of computing the largest Z-eigenvalue of $\mathcal{A}$, x_{k+1} is the largest eigenvalue of the matrix $\mathcal{A}x_k^2$. Therefore, the quadratic function $q(y) = y^T (\mathcal{A}x_k^2) y = \mathcal{A}x_k^2 y^2$ reaches a maximum value λ_{k+1} at the point $y = x_{k+1}$ over the unit sphere $\mathbb{S}^n$, i.e., $\lambda_{k+1} = \mathcal{A}x_k^2 x_{k+1}^2 \geq \mathcal{A}x_k^2 y^2$ for all $y \in \mathbb{S}^n$. At the same time, x_k is the largest eigenvalue of the matrix $\mathcal{A}x_{k-1}^2$. These results give

$$\lambda_{k+1} = \mathcal{A}x_k^2 x_{k+1}^2 \geq \mathcal{A}x_k^2 x_{k-1}^2 = \mathcal{A}x_{k-1}^2 x_k^2 = \lambda_k. \tag{9}$$

Zhou et al. (2023), *PeerJ Comput. Sci.*, DOI 10.7717/peerj-cs.1242

Here, the second equality holds because of the symmetric property of $\mathcal{A}$. From Eq. (9), we know that the sequence λ_k generated by Algorithm 3 is nondecreasing. On the other side, λ_k is computed by $\lambda_k = \mathcal{A}x_{k-1}^2 x_k^2$, where $x_k \in \mathbb{S}^n$. Due to the compactness of the unit sphere $\mathbb{S}^n$, we also know that the sequence λ_k is bounded above. Consequently, λ_k has a unique limit, and we can readily conclude by posing this as a theorem.

Theorem 1. Let λ_k be a sequence generated by Algorithm 3. Then the sequence λ_k is nonincreasing and there exists λ^* such that $\lambda_k \to \lambda^*$.

While Theorem 1 ensures that Algorithm 3 always terminates in finitely many iterations, theoretically, it cannot ensure that the sequence λ_k converges to a Z-eigenvalue of $\mathcal{A}$ because the equality constraints in Eq. (8) are omitted in the implementation of Algorithm 3. One possible result is the occurrence of cyclic solutions, that is, two consecutive iterates x_k and x_{k+1} that satisfy $\mathcal{A}x_k^2 x_{k+1} = \lambda_k x_{k+1}$, $\mathcal{A}x_{k+1}^2 x_k = \lambda_{k+1} x_k$, and $\lambda_k = \lambda_{k+1}$. However, this situation is rarely encountered in the numerical experiments presented in the next section. Additionally, how to theoretically avoid this situation is the subject for future research.

NUMERICAL EXPERIMENTS

In this section, we present some numerical results of the ADM for computing the largest or smallest Z-eigenvalues of tensors. The proposed ADM is compared with the GEAP method, which is an adaptive shifted power method first proposed by *Kolda & Mayo (2014)*, and the AG method (*Yu et al., 2016*), which is an adaptive gradient method with inexact stepsize. All experiments are performed in MATLAB R2017a and the Tensor Toolbox (*Bader & Kolda, 2012*) under a Windows 10 operating system on a laptop with an Intel(R) Core (TM) i7-10510U CPU and 12 GB RAM. In all numerical experiments, we terminate the computation when the absolute difference between successive eigenvalues is less than 10^{-10}, *i.e.,* $|\lambda_{k+1} - \lambda_k| \le 10^{-10}$, or the number of iterations exceeded the maximum number 500.

In our experiments, we use some typical examples from references (*Cui, Dai & Nie, 2014*; *Kofidis & Regalia, 2002*; *Nie & Wang, 2014*) to assess the performance of the proposed method in finding the largest or smallest Z-eigenvalue of a symmetric tensor. All of the largest or smallest Z-eigenvalues in these examples are given in the original literature. Therefore, those desired values are known in advance.

Example 1 (*Kofidis & Regalia, 2002*) Let $\mathcal{A} \in \mathbb{S}^{[4,3]}$ be the symmetric tensor with entries

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

The largest and smallest Z-eigenvalue of the tensor $\mathcal{A}$ are respectively
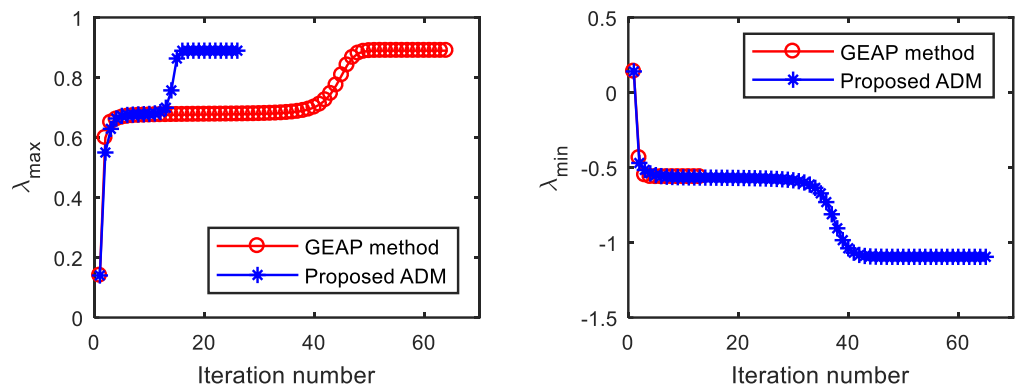
Figure 1 Convergence comparison of the GEAP method and the proposed method for computing the largest and smallest Z-eigenvalue of $\mathcal{A}$ from Example 1.

$$\lambda_{max} = 0.8893, v_{max} = (0.6672, 0.7160, 0.9073)^T;$$
$$\lambda_{min} = -1.0954, v_{min} = (-0.6447, -0.3357, 0.3043)^T.$$

We first test the convergence performance of the proposed ADM in comparison to GEAP. Figure 1 shows the convergence trajectories of the two methods for computing extreme Z-eigenvalues of $\mathcal{A}$ from *Example 1*, with the starting point $x_0 = (0.0417, -0.5618, 0.6848)^T$. As shown on the left in Fig. 1, both GEAP and ADM can find the largest Z-eigenvalue 0.8893. Until the stopping criterion is met, GEAP runs 63 iterations in 0.2188 s, while the proposed ADM runs only 26 iterations in 0.0313 s. When computing the smallest Z-eigenvalue with the same starting point (right in Fig. 1), although ADM runs longer than GEAP, the ADM find the desired value of -1.0954, while GEAP fail.

To test the performance of the proposed algorithm in finding extreme eigenvalues, 1,000 random starting guesses drawn uniformly from [-1, 1] are employed in the experiments. All three methods are implemented 1,000 times, each with the same initial point. We list the number of occurrences of extreme eigenvalues (#Occu.), the average number of iterations (Iter$_{ave}$), and the average running time in seconds (CPU$_{ave}$) for the two types of Z-eigenvalues in Tables 1, 2, 3, 4, 5 and 6. As shown in Tables 1–6, for the cases of computing the largest eigenvalues, the proposed ADM runs a similar or slightly larger number of iterations but in a similar or shorter time compared to both GEAP and AG. For the case of computing the smallest Z-eigenvalue, ADM runs slower for *Example 1* but runs much faster for the other examples. It can also be seen from the fourth columns of Tables 1–6 that GEAP can only obtain the largest Z-eigenvalues with a probability of about 0.55, and the smallest Z-eigenvalue with a probability of about 0.65. AG performs clearly better than GEAP. The proposed ADM performs best in almost all cases, which can reach

Table 1 Comparison results for computing extreme Z-eigenvalues of $\mathcal{A}$ from Example 1.

Type of Z-eigenvalue	Method	λ	#Occu.	Iter$_{ave}$	CPU$_{ave}$
	GEAP	0.8893	51.00%	27.59	0.0356
λ_{max}	AG	0.8893	**57.60%**	12.97	0.0202
	ADM	0.8893	57.10%	28.96	**0.0165**
	GEAP	-1.0953	41.10%	12.18	**0.0121**
λ_{min}	AG	-1.0953	52.50%	8.24	0.0185
	ADM	-1.0953	**100.00%**	43.52	0.0258

Note.
The best results are in bold.

Table 2 Comparison results for computing the extreme Z-eigenvalues of $\mathcal{A}$ from Example 2 ($n = 5$).

Type of Z-eigenvalue	Method	λ	#Occu.	Iter$_{ave}$	CPU$_{ave}$
	GEAP	7.2595	49.80%	49.30	0.0701
λ_{max}	AG	7.2595	60.90%	19.91	**0.0334**
	ADM	7.2595	**100.00%**	117.20	0.0806
	GEAP	-8.8463	51.60%	49.85	0.0692
λ_{min}	AG	-8.8463	77.80%	23.29	**0.0385**
	ADM	-8.8463	**100.00%**	57.41	0.0389

Note.
The best results are in bold.

the largest Z-eigenvalues for Examples 2-5 and the smallest Z-eigenvalues for all examples with a probability of 1.

Example 2 (*Nie & Wang, 2014*). Consider the symmetric tensor $\mathcal{A} \in \mathbb{S}^{[4,n]}$ such that

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

In the case of $n = 5$, the largest and smallest Z-eigenvalues of the tensor $\mathcal{A}$ are respectively

$$\lambda_{max} = 7.2595, v_{max} = (0.2686, 0.6150, 0.3959, -0.1872, -0.5982)^T;$$
$$\lambda_{min} = -8.8463, v_{min} = (-0.5809, -0.3563, 0.1959, 0.5680, 0.4179)^T.$$

Example 3 (*Cui, Dai & Nie, 2014*). Consider the symmetric tensor $\mathcal{A} \in \mathbb{S}^{[4,n]}$ such that

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

In the case of $n = 5$, we can obtain the largest and smallest Z-eigenvalues of the tensor $\mathcal{A}$ from the reference as follows.

$$\lambda_{max} = 34.5304, v_{max} = (0.6665, 0.1089, 0.4132, 0.6070, -0.0692)^T;$$
$$\lambda_{min} = -101.1994, v_{min} = (0.2248, 0.5541, 0.3744, 0.2600, 0.6953)^T.$$

Example 4 (*Nie & Wang, 2014*). Let $\mathcal{A} \in \mathbb{S}^{[4,n]}$ be a symmetric tensor with

$$a_{ijkl} = \arctan\left((-1)^i \frac{i}{n}\right) + \arctan\left((-1)^j \frac{j}{n}\right) + \arctan\left((-1)^k \frac{k}{n}\right) + \arctan\left((-1)^l \frac{l}{n}\right).$$

Table 3 Comparison results for computing the extreme Z-eigenvalues of $\mathcal{A}$ from Example 3 ($n = 5$).

Type of Z-eigenvalue	Method	λ	#Occu.	Iter$_{ave}$	CPU$_{ave}$
	GEAP	34.5304	62.30%	27.52	**0.0341**
λ_{max}	AG	34.5304	92.10%	14.54	0.0350
	ADM	34.5304	**100.00%**	56.26	0.0362
	GEAP	−101.1994	72.00%	14.00	0.0184
λ_{min}	AG	−101.1994	98.90%	9.37	0.0149
	ADM	−101.1994	**100.00%**	13.16	**0.0057**

Note.
The best results are in bold.

Table 4 Comparison results for computing the extreme Z-eigenvalues of $\mathcal{A}$ from Example 4 ($n = 5$).

Type of Z-eigenvalue	Method	λ	#Occu.	Iter$_{ave}$	CPU$_{ave}$
	GEAP	13.0779	63.20%	22.01	0.0263
λ_{max}	AG	13.0779	93.00%	12.80	0.0209
	ADM	13.0779	**100.00%**	30.72	**0.0166**
	GEAP	−23.5741	69.10%	15.21	0.0161
λ_{min}	AG	−23.5741	97.20%	10.09	0.0193
	ADM	−23.5741	**100.00%**	15.32	**0.0030**

Note.
The best results are in bold.

Table 5 Comparison results for computing the extreme Z-eigenvalues of $\mathcal{A}$ from *Example 5* ($n = 5$).

Type of Z-eigenvalue	Method	λ	#Occu.	Iter$_{ave}$	CPU$_{ave}$
	GEAP	9.5821	61.00%	25.59	0.0309
λ_{max}	AG	9.5821	91.70%	14.03	**0.0285**
	ADM	9.5821	**100.00%**	50.69	0.0295
	GEAP	−27.0429	71.60%	13.47	0.0153
λ_{min}	AG	−27.0429	98.40%	9.13	0.0159
	ADM	−27.0429	**100.00%**	12.86	**0.0050**

Note.
The best results are in bold.

In the case of $n = 5$, the largest and smallest Z-eigenvalues of the tensor $\mathcal{A}$ are respectively

$$\lambda_{max} = 13.0779, v_{max} = (0.3174, 0.5881, 0.1566, 0.7260, 0.0418)^T;$$
$$\lambda_{min} = -23.5740, v_{min} = (0.4403, 0.2382, 0.5602, 0.1354, 0.6459)^T.$$

Example 5 (*Nie & Wang, 2014*). Let $\mathcal{A} \in \mathbb{S}^{[4,n]}$ be a symmetric tensor with

$$a_{ijkl} = \frac{(-1)^i}{i} + \frac{(-1)^j}{j} + \frac{(-1)^k}{k} + \frac{(-1)^l}{l}, 1 \le i,j,k,l \le n.$$

Table 6 Comparison results for computing the extreme Z-eigenvalues of $\mathcal{A}$ from *Example 6*.

Type of Z-eigenvalue	Method	λ	#Occu.	Iter$_{ave}$	CPU$_{ave}$
	GEAP	1	51.50%	7.56	0.0077
λ_{max}	AG	1	**82.80%**	6.28	0.0054
	ADM	1	64.70%	12.40	**0.0066**
	GEAP	0	99.90%	231.32	0.3365
λ_{min}	AG	0	**100.00%**	22.73	0.0595
	ADM	0	**100.00%**	12.15	**0.0027**

Note.
The best results are in bold.
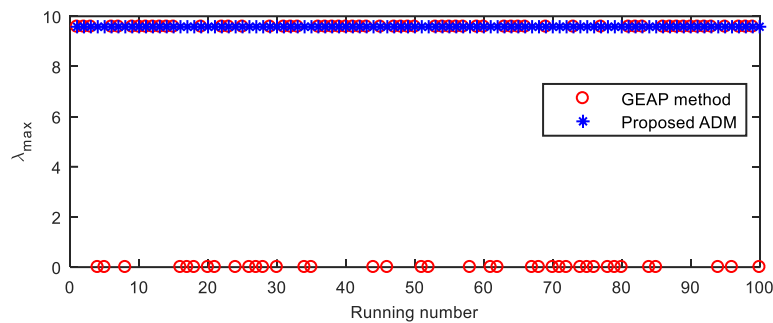
Figure 2 The largest Z-eigenvalues computed by GEAP and ADM in the 100 runs on the tensor $\mathcal{A}$ from Example 5.

For $n = 5$, we can get the largest and smallest Z-eigenvalue of the tensor $\mathcal{A}$ with

$$\lambda_{max} = 9.5821, v_{max} = (-0.1125, 0.7048, 0.2507, 0.5685, 0.3233)^T;$$

$$\lambda_{min} = -27.0429, v_{min} = (-0.6900, -0.1987, -0.4717, -0.2806, -0.4280)^T.$$

Besides the point $\lambda_1 = 9.5821$ of the largest Z-eigenvalue, the tensor $\mathcal{A}$ has another stable eigenvalue $\lambda_2 = 0$. As shown in Figure 2 and Table 5, GEAP falls into the latter point with a probability of around 0.4, while ADM can always converge to the previous point.

Example 6 (*Sheng & Ni, 2021*). Let $\mathcal{A} \in \mathbb{S}^{[6,3]}$ be a tensor with

$$a_{111122} = \frac{1}{15}, a_{112222} = \frac{1}{15}, a_{112233} = -\frac{1}{30}, a_{333333} = 1,$$

and $a_{i_1 \cdots i_6} = 0$ if $(i_1 \cdots i_6)$ is not a permutation of an index in the above. We can get the largest Z-eigenvalue $\lambda_{max} = 1$ and smallest Z-eigenvalue $\lambda_{min} = 0$, and these corresponding eigenvectors are not unique. The comparison results are shown in Table 6, from which we find that GEAP converges very slowly when computing the smallest eigenvalue of $\mathcal{A}$. In contrast, ADM reaches the smallest eigenvalue with only about 12 iterations for each execution.

DISCUSSION

In general, algorithms for computing the largest or smallest eigenvalue of a higher-order tensor are prone to getting stuck in local extrema and then converging to an arbitrary

eigenvalue of the tensor depending on the initial conditions. However, the counterpart for symmetric matrices can always converge to the largest or smallest one. Motivated by this, we propose combining algorithms for matrix eigenproblem and tensor optimization techniques in order to obtain extreme eigenvalues. Specifically, the tensor eigenproblem is split into a series of matrix eigenvalue problems using a variable splitting method, and then an alternating scheme is proposed to solve the problem.

To solve the tensor eigenproblems, many algorithms for matrix eigenproblems are extended to the tensor case. However, these generalizations cannot guarantee the low complexity of these algorithms, and the global convergence to the extreme eigenvalues is also not ensured. In this article, the tensor eigenvalue problem is directly transformed into a series of matrix eigenvalue problems so that its algorithms can be directly used to solve the original tensor eigenvalue problem. This method not only overcomes local minima problems existing in direct generalizations, but also has great potential to speed up the convergence. The experimental results verify the effectiveness and advancement of the proposed algorithm, which converges rapidly in most cases and reaches extreme Z-eigenvalues with a significantly higher probability. In many cases, we determine the extreme eigenvalue with a probability of 1, indicating that we can obtain the extreme eigenvalue under any given initial value in these cases. This demonstrates the significant robustness of the proposed method.

However, the proposed algorithm cannot guarantee global convergence for each type of tensor. For Examples 1 and 6, we could only obtain the largest eigenvalue with a probability of about 0.6. In future research, the question of why this kind of tensor cannot obtain the extreme eigenvalue under any initial point will be discussed in more detail.

CONCLUSION

In this article, we transform a tensor Z-eigenvalue problem into a series of matrix eigenvalue problems using a variable splitting method and propose an alternating scheme for computing the largest or smallest Z-eigenvalue of symmetric tensors. Just like the classical power method, which constantly uses the intermediate iterates to construct a vector, the proposed algorithm uses them to construct a matrix and computes the eigenvalues and corresponding eigenvectors of the matrix. We analyze the convergence properties of the proposed method which is verified in the numerical experiments. The limitations of this work are twofold. First, as the authors note themselves, it can only ensure the convergence of eigenvalues, but not the convergence of eigenvectors due to the possible existence of cyclic solutions. Second, as can be seen from the numerical examples, it cannot obtain the largest eigenvalues with a 100% probability. The numerical results are reported for some testing examples, which showed that the proposed method converged much faster than both GEAP and AG in most cases and could reach the extreme Z-eigenvalues with a significantly higher probability than GEAP.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their insightful comments.

Funding

This research was supported by the National Natural Science Foundation of China (No. 82060328 and 61863001). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
National Natural Science Foundation of China: 82060328, 61863001.

Competing Interests

The authors declare there are no competing interests.

Author Contributions

- Genjiao Zhou conceived and designed the experiments, performed the experiments, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Shoushi Wang performed the experiments, analyzed the data, prepared figures and/or tables, and approved the final draft.
- Jinhong Huang conceived and designed the experiments, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The code is available in Supplementary Files.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.1242#supplemental-information.

REFERENCES

Ashourian M, Sharifi-Tehrani O. 2022. Application of semi-circle law and Wigner spiked-model in GPS jamming confronting. In: *Signal, image and video processing*. 1–8.

Bader BW, Kolda TG. 2012. MATLAB Tensor Toolbox Version 3.4. *Available at https://www.tensortoolbox.org/*.

Benson AR, Gleich DF. 2019. Computing tensor Z-eigenvectors with dynamical systems. *SIAM Journal on Matrix Analysis and Applications* **40**:1311–1324 DOI 10.1137/18M1229584.

Chang K, Pearson K, Zhang T. 2009. On eigenvalue problems of real symmetric tensors. *Journal of Mathematical Analysis and Applications* **350**:416–422 DOI 10.1016/j.jmaa.2008.09.067.

Chang K-C, Pearson K, Zhang T. 2008. Perron–Frobenius theorem for nonnegative tensors. *Communications in Mathematical Sciences* **6**:507–520 DOI 10.4310/CMS.2008.v6.n2.a12.

Chen L, Han L, Zhou L. 2016. Computing tensor eigenvalues via homotopy methods. *SIAM Journal on Matrix Analysis and Applications* **37**:290–319 DOI 10.1137/15M1010725.

Cui C-F, Dai Y-H, Nie J. 2014. All real eigenvalues of symmetric tensors. *SIAM Journal on Matrix Analysis and Applications* **35**:1582–1601 DOI 10.1137/140962292.

De Lathauwer L, De Moor B, Vandewalle J. 2000. On the best rank-1 and rank-(R 1, R 2, …, Rn) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications* **21**:1324–1342 DOI 10.1137/S0895479898346995.

Han L. 2012. An unconstrained optimization approach for finding real eigenvalues of even order symmetric tensors. ArXiv preprint. arXiv:12035150.

Hao CL, Cui CF, Dai YH. 2015. A sequential subspace projection method for extreme Z-eigenvalues of supersymmetric tensors. *Numerical Linear Algebra with Applications* **22**:283–298 DOI 10.1002/nla.1949.

Hillar CJ, Lim L-H. 2013. Most tensor problems are NP-hard. *Journal of the ACM* **60**:45.

Hu S, Huang ZH, Qi L. 2013. Finding the extreme Z-eigenvalues of tensors via a sequential semidefinite programming method. *Numerical Linear Algebra with Applications* **20**:972–984 DOI 10.1002/nla.1884.

Jaffe A, Weiss R, Nadler B. 2018. Newton correction methods for computing real eigenpairs of symmetric tensors. *SIAM Journal on Matrix Analysis and Applications* **39**:1071–1094 DOI 10.1137/17M1133312.

Kofidis E, Regalia PA. 2001. Tensor approximation and signal processing applications. *Contemporary Mathematics* **280**:103–134 DOI 10.1090/conm/280/04625.

Kofidis E, Regalia PA. 2002. On the best rank-1 approximation of higher-order super-symmetric tensors. *SIAM Journal on Matrix Analysis and Applications* **23**:863–884 DOI 10.1137/S0895479801387413.

Kolda TG, Mayo JR. 2011. Shifted power method for computing tensor eigenpairs. *SIAM Journal on Matrix Analysis and Applications* **32**:1095–1124 DOI 10.1137/100801482.

Kolda TG, Mayo JR. 2014. An adaptive shifted power method for computing generalized tensor eigenpairs. *SIAM Journal on Matrix Analysis and Applications* **35**:1563–1581 DOI 10.1137/140951758.

Li W, Ng MK. 2014. On the limiting probability distribution of a transition probability tensor. *Linear and Multilinear Algebra* **62**:362–385 DOI 10.1080/03081087.2013.777436.

Lim L-H. 2005. Singular values and eigenvalues of tensors: a variational approach. In: *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing IEEE*. Piscataway: IEEE, 129–132.

Ni Q, Qi L, Wang F. 2008. An eigenvalue method for testing positive definiteness of a multivariate form. *IEEE Transactions on Automatic Control* **53**:1096–1107 DOI 10.1109/TAC.2008.923679.

Nie J, Wang L. 2014. Semidefinite relaxations for best rank-1 tensor approximations. *SIAM Journal on Matrix Analysis and Applications* **35**:1155–1179 DOI 10.1137/130935112.

Qi L. 2005. Eigenvalues of a real supersymmetric tensor. *Journal of Symbolic Computation* **40**:1302–1324 DOI 10.1016/j.jsc.2005.05.007.

Qi L, Wang F, Wang Y. 2009. Z-eigenvalue methods for a global polynomial optimization problem. *Mathematical Programming* **118**:301–316 DOI 10.1007/s10107-007-0193-6.

Qi L, Yu G, Wu EX. 2010. Higher order positive semidefinite diffusion tensor imaging. *SIAM Journal on Imaging Sciences* **3**:416–433 DOI 10.1137/090755138.

Qi L, Yu G, Xu Y. 2013. Nonnegative diffusion orientation distribution function. *Journal of Mathematical Imaging and Vision* **45**:103–113 DOI 10.1007/s10851-012-0346-y.

Schultz T, Seidel H-P. 2008. Estimating crossing fibers: a tensor decomposition approach. *IEEE Transactions on Visualization and Computer Graphics* **14**:1635–1642 DOI 10.1109/TVCG.2008.128.

Sharifi-Tehrani O, Sabahi MF. 2022. Eigen analysis of flipped Toeplitz covariance matrix for very low SNR sinusoidal signals detection and estimation. *Digital Signal Processing* **129**:103677 DOI 10.1016/j.dsp.2022.103677.

Sharifi-Tehrani O, Sabahi MF, Danaee MR. 2021. Efficient GNSS jamming mitigation using the MarcenkoPastur Law and Karhunen–Loeve decomposition. In: *IEEE Transactions on Aerospace and Electronic Systems.*

Sheng Z, Ni Q. 2021. Computing tensor Z-eigenvalues via shifted inverse power method. *Journal of Computational and Applied Mathematics* **398**:113717 DOI 10.1016/j.cam.2021.113717.

Xiong L, Jiang Z, Liu J, Qin Q. 2022. New Z-eigenvalue localization set for T ensor and its application in entanglement of multipartite quantum states. *Mathematics* **10**:2624 DOI 10.3390/math10152624.

Yu G, Yu Z, Xu Y, Song Y, Zhou Y. 2016. An adaptive gradient method for computing generalized tensor eigenpairs. *Computational Optimization and Applications* **65**:781–797 DOI 10.1007/s10589-016-9846-9.

Zhang F, Zhou B, Peng L. 2013. Gradient skewness tensors and local illumination detection for images. *Journal of Computational and Applied Mathematics* **237**:663–671 DOI 10.1016/j.cam.2012.08.013.

Zhang T, Golub GH. 2001. Rank-one approximation to high order tensors. *SIAM Journal on Matrix Analysis and Applications* **23**:534–550 DOI 10.1137/S0895479899352045.