Numerical Algorithms (2024) 97:29–49
https://doi.org/10.1007/s11075-023-01692-w

ORIGINAL PAPER

Iterative methods for solving tensor equations based on exponential acceleration

Maolin Liang[1] · Lifang Dai[1] · Ruijuan Zhao[2]

Received: 18 January 2023 / Accepted: 17 October 2023 / Published online: 13 November 2023

Abstract
The tensor equation $\mathcal{A}\boldsymbol{x}^{m-1} = \boldsymbol{b}$ with the tensor $\mathcal{A}$ of order m and dimension n and the vector $\boldsymbol{b}$, has practical applications in several fields including signal processing, high-dimensional PDEs, high-order statistics, and so on. In this paper, a class of exponential accelerated iterative methods is proposed for solving the tensor equation mentioned above in the sense that the coefficient tensor $\mathcal{A}$ is a symmetric and nonsingular or singular $\mathcal{M}$-tensor. The obtained iterative schemes involve the classical Newton's method as a special case. It is shown that the proposed method for nonsingular case is superlinearly convergent, while for singular cases, it is linearly convergent. The performed numerical experiments demonstrate that our methods outperform some existing ones.

Keywords Tensor equations · Symmetric $\mathcal{M}$-tensors · Newton method · Exponential acceleration

Mathematics Subject Classification (2010) 15A69 · 65H10 · 90C30

1 Introduction

Let $\mathbb{R}$ be the set of all real numbers. For an order m and dimension $n_1 \times n_2 \times \cdots \times n_m$ tensor $\mathcal{A}$, it has $n_1 n_2 \cdots n_m$ entries $\mathcal{A}_{i_1 \ldots i_m}$ indexed by i_j satisfying $1 \leq i_j \leq n_j$,

✉ Maolin Liang
liangml2005@163.com

Lifang Dai
dailf2005@163.com

Ruijuan Zhao
zhaobin7755382@163.com

[1] School of Mathematics and Statistics, Tianshui Normal University, Tianshui 741001, People's Republic of China

[2] School of Information Engineering, Lanzhou University of Finance and Economics, Lanzhou 730101, People's Republic of China

$j = 1, 2, \ldots, m$. The set of all order m and dimension $n_1 \times n_2 \times \cdots \times n_m$ tensors over the real field is denoted by $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_m}$. Particularly, we denote this set by $\mathbb{R}^{[m,n]}$ if $n_1 = \cdots = n_m = n$. We say that $\mathcal{A} \in \mathbb{R}^{[m,n]}$ is a symmetric tensor if its entries $\mathcal{A}_{i_1 \ldots i_m}$ are invariant under any permutation of their indices $\{i_1, i_2, \ldots, i_m\}$. It is not difficult to observe that the tensor $\mathcal{A}$ reduces to a vector of size n_1 when $m = 1$, or becomes a matrix of size $n_1 \times n_2$ in the case that $m = 2$.

Although tensors are a generalized form of matrices, they have significant differences from matrices, For example, unlike matrix situations, there are several definitions of tensor ranks, eigenvalues, and tensor-tensor multiplications based on different application backgrounds including chemometries, signal processing, high-dimensional statistics, and so on, one can refer to the Refs. [1–3] for details.

It is worth mentioning that tensor equations are important models for describing high-dimensional problems. In this paper, we consider the following tensor equation

$$\mathcal{A}\boldsymbol{x}^{m-1} = \boldsymbol{b}, \tag{1.1}$$

where $\mathcal{A} = (\mathcal{A}_{i_1 i_2 \ldots i_m}) \in \mathbb{R}^{[m,n]}$, $\boldsymbol{b} \in \mathbb{R}^n$, and $\mathcal{A}\boldsymbol{x}^{m-1} \in \mathbb{R}^n$ defined by

$$(\mathcal{A}\boldsymbol{x}^{m-1})_i = \sum_{i_2, i_3, \ldots, i_m} \mathcal{A}_{i i_2 \ldots i_m} \boldsymbol{x}(i_2)\boldsymbol{x}(i_3) \ldots \boldsymbol{x}(i_m),$$

herein $\boldsymbol{x}(i)$ stands for the ith-component of the vector $\boldsymbol{x}$. The notation $\mathcal{A}\boldsymbol{x}^{m-1}$ was first exploited by Qi in [4] to define the eigenvalues of a tensor.

Let $\mathcal{A} \in \mathbb{R}^{[m,n]}$. We say that $\lambda \in \mathbb{R}$ is an eigenvalue of $\mathcal{A}$ if there exists a nonzero $\boldsymbol{x} \in \mathbb{R}^n$ such that

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

here $\boldsymbol{x}^{[m-1]} = (\boldsymbol{x}(1)^{m-1}, \boldsymbol{x}(2)^{m-1}, \ldots, \boldsymbol{x}(n)^{m-1})^{\mathrm{T}}$. The spectral radius of $\mathcal{A}$ is the maximum modulus of the eigenvalues, and is denoted by $\rho(\mathcal{A})$. A tensor $\mathcal{A} \in \mathbb{R}^{[m,n]}$ is called a nonsingular (singular) $\mathcal{M}$-tensor [5, 6], if it can be represented as $\mathcal{A} = s\mathcal{I} - \mathcal{B}$ in which $s > \rho(\mathcal{B})$ ($s \geq \rho(\mathcal{B})$), $\mathcal{B} \in \mathbb{R}^{[m,n]}$ is nonnegative, and $\mathcal{I}$ is the identity tensor, that is, $\mathcal{I}_{ii\ldots i} = 1$, and otherwise $\mathcal{I}_{i_1 i_2 \ldots i_m} = 0$.

The tensor equation (1.1) has important applications in many fields, such as information retrieval [7], numerical solution of partial differential equations [8], tensor complement problem [9], higher-order statistics [10], and so on. In recent years, it has been researched deeply in the sense that the coefficient tensor $\mathcal{A}$ has some special structures. For instance, Ding and Wei extended the classical Jacobian and Gauss-Seidel methods for linear equations and the Newton method for nonlinear equations to (1.1) when $\mathcal{A}$ is a nonsingular (symmetric) $\mathcal{M}$-tensor and $\boldsymbol{b}$ is a positive vector [8] (we call it $\mathcal{M}$-tensor equation for ease of expression). After that the homotopy method [11], the tensor method [12], splitting iterative methods [13–15], Newton-type method [16], neural work method [17] were proposed for solving the $\mathcal{M}$-tensor equation mentioned above. Subsequently, the classical Levenberg-Marquardt (LM) method was applied to (1.1) when the tensor $\mathcal{A}$ is a nonsingular semi-symmetric $\mathcal{M}$-tensor [18]. Very recently, the ADMM-type method and the two-step accelerated LM

method were established, respectively, in [19] and [20], for solving the tensor equation (1.1) with a general tensor $\mathcal{A}$.

In essence, the tensor equation under consideration is a nonlinear equation. The iterative algorithms listed above fully considered the particularity of the corresponding tensor equations, and possess better convergence. Nevertheless, no one is suitable for all equations. It is our constant pursuit to establish more efficient iterative algorithms for the tensor equation mentioned above. In present paper, we are interested in the solution of the tensor equation (1.1) whose coefficient tensor $\mathcal{A}$ is a singular or a nonsingular $\mathcal{M}$-tensor. This kind of tensor equations arises from the higher-dimensional PDEs, and one can see [8] for more details. In addition, as is proved that, for any $\mathcal{A} \in \mathbb{R}^{[m,n]}$, there exists a symmetric tensor $\widehat{\mathcal{A}} \in \mathbb{R}^{[m,n]}$ such that $\mathcal{A}\boldsymbol{x}^{m-1} = \widehat{\mathcal{A}}\boldsymbol{x}^{m-1}$ [3]. Therefore, in this paper, we make the following assumptions:

⋆ The tensor $\mathcal{A}$ in (1.1) is a symmetric $\mathcal{M}$-tensor.

⋆ The tensor equation (1.1) is solvable.

The approach to be established here is relying on the exponentially accelerated technique for nonlinear equations, which is an extension of the classical Newton's method [21]. As is well-known, this method is quite efficient and has quadratic convergence under some circumstances, but it relies heavily on initial values, and may fail to converge in the case that the initial guess is far from zero or the derivative of the function in the vicinity of the required root is small. Recently, Chen and Li in [22] proposed a class of exponential iteration approaches (denoted by EAI for short) that has quadratic convergence, and can be applied in the case where the Newton's method is not successful. The EAI method contains the Newton's method as a special case by taking the first order Taylor series expansion, see a short review in Sect. 2.

The under-considered tensor equation (1.1) is a nonlinear equation but possesses special structure. So we attempt to search more efficient iterative methods in two cases: the first case is that the coefficient tensor $\mathcal{A}$ is a symmetric and nonsingular $\mathcal{M}$-tensor, and the second one is that the coefficient tensor $\mathcal{A}$ is a symmetric and singular $\mathcal{M}$-tensor (see Sect. 3 for details). We shall apply the exponential acceleration technique introduced in [22] to the tensor equation (1.1). For ease of expression, we denote EAI-NS as the exponentially accelerated iterative method corresponding to the nonsingular $\mathcal{M}$-tensor $\mathcal{A}$, and EAI-S as the exponentially accelerated iterative method corresponding to the singular $\mathcal{M}$-tensor $\mathcal{A}$. Notably, in view of the singularity of the differential matrix of the vector-valued function, the EAI-S method inherits the characteristics of the LM method. It will be shown that both of them are also the high-dimensional generalizations of the Newton's method proposed in [8]. Moreover, we can prove that the EAI-NS method is suplinearly convergent and the EAI-S method is linearly convergent under the aforementioned hypotheses. Several numerical examples derived from practical applications demonstrate that our methods are promising.

The remainder of this paper is organized as follows. In Sect. 2, we review some basic definitions and conclusions related to tensors and nonlinear equations. In Sect. 3, we present the exponentially accelerated iterative methods for solving the tensor equation (1.1), and the convergence of them will also been analyzed there. In Sect. 4, some numerical examples are given to illustrate the effectiveness of the proposed methods. In Sect. 5, we conclude this paper with some remarks.

2 Preliminaries

2.1 Notations and definitions

First of all, we introduce some necessary notations: scalars are denoted by lower-case letters, e.g., a, b, c; vectors are denoted by boldface lower-case letters, e.g., $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$; matrices are denoted by boldface capital letters, e.g., $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$; tensors are denoted by calligraphic script letters, e.g., $\mathcal{A}, \mathcal{B}, \mathcal{C}$.

Now, we introduce the following definition on the tensor-vector product (see, e.g., [1] for more details).

Definition 2.1 Let $\mathcal{A} = (\mathcal{A}_{i_1 i_2 \ldots i_m}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_m}$ and $\boldsymbol{x} = (\boldsymbol{x}(i)) \in \mathbb{R}^{n_k}$. Then, the k-mode (vector) product, denoted by $\mathcal{A} \bullet_k \boldsymbol{x}$, is an $n_1 \times \cdots \times n_{k-1} \times n_{k+1} \times \cdots \times n_m$ tensor, elementwise,

$$(\mathcal{A} \bullet_k \boldsymbol{x})_{i_1 \ldots i_{k-1} i_{k+1} \ldots i_m} = \sum_{i_k=1}^{n_k} \mathcal{A}_{i_1 \ldots i_k \ldots i_m} \boldsymbol{x}(i_k).$$

Using Definition 2.1, for a given tensor $\mathcal{A} \in \mathbb{R}^{[m,n]}$ and vector $\boldsymbol{x} \in \mathbb{R}^n$, we denote that

$$\mathcal{A}\boldsymbol{x}^{m-k} = \mathcal{A} \bullet_{k+1} \boldsymbol{x} \bullet_{k+2} \boldsymbol{x} \cdots \bullet_m \boldsymbol{x} \in \mathbb{R}^{[k,n]}, k = 1, 2, \ldots, m-1.$$

2.2 Two classical iterative methods for nonlinear equations

In this subsection, we first give a brief review on the classical Newton's method. Let f be a real valued function over the closed and convex set $\Omega \subseteq \mathbb{R}$, and assume that it is continuously differentiable in the domain of a root x^* of $f(x) = 0$. Then, the Newton's method to solve this nonlinear equation can be expressed as follows:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

This method is quite efficient and has quadratic convergence under some circumstances. Nevertheless, it may fail to converge when the initial guess is far from zero or the derivative of the function f in the vicinity of x^* is small.

Due to the aforementioned shortcomings, Chen and Li [22] proposed the EAI method for the nonlinear equation $f(x) = 0$, and the iterative scheme of which consists of the following iteration step:

$$x_{k+1} = x_k \exp\left(-\frac{f(x_k)}{x_k f'(x_k)}\right).$$

In particular, this iterative method reduces to the well-known Newton's method by taking the first order Taylor series expansion of $\exp\left(-\frac{f(x_k)}{x_k f'(x_k)}\right)$. Recently, several

variants of the EAI method were proposed for solving nonlinear equations; see, e.g., [23–25].

Next, we recall the classical Levenberg-Marquardt (LM) method [26, 27]. To do this, let $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^n$ be a continuously differential function, then the LM method consists of computing the trial step at each iteration

$$\boldsymbol{d}_k^{LM} = -(\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \tau_k \boldsymbol{I}_n)^{-1}\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{F}_k,$$

in which $\boldsymbol{F}_k = \boldsymbol{F}(\boldsymbol{x}_k)$, $\boldsymbol{J}_k$ represents the value of the Jacobian $\boldsymbol{J}(\boldsymbol{x}) := \boldsymbol{F}'(\boldsymbol{x})$ at $\boldsymbol{x}_k$, and the LM parameter $\tau_k > 0$ is updated from iteration to iteration. Under the local error bound condition which is weaker than nonsingularity [28], it has been proved that this method has quadratic convergence, and several variants have been developed in the literature; see, e.g., [18, 20, 29] and the references therein.

3 The exponentially accelerated iterative methods

In this section, we shall establish the exponentially accelerated iterative methods to solve the tensor equation (1.1) under the assumption that it is always solvable.

In the sequel, two kinds of exponentially accelerated approaches will be proposed for (1.1) under the two scenarios: The first one is that the coefficient tensor $\mathcal{A}$ in (1.1) is a symmetric and nonsingular $\mathcal{M}$-tensor, and the second one is that the coefficient tensor $\mathcal{A}$ is a symmetric and singular $\mathcal{M}$-tensor. Furthermore, the convergence of the proposed iterative methods will be discussed. We should emphasize that the convergence analysis of the methods given in present paper as well as their iteration schemes are similar but different from that of the iterative method presented in [25].

3.1 The EAI method for (1.1) with nonsingular $\mathcal{M}$-tensor $\mathcal{A}$

As shown by Ding and Wei [8], the tensor equation (1.1) always has a solution when the coefficient tensor $\mathcal{A}$ is a nonsingular one. In order to derive the new iterative scheme, denote

$$\boldsymbol{F}(\boldsymbol{x}) := \mathcal{A}\boldsymbol{x}^{m-1} - \boldsymbol{b} = 0. \tag{3.1}$$

Using (3.1) and Definition 2.1, it follows from the symmetry of the tensor $\mathcal{A}$ that the gradient of $F(\boldsymbol{x})$ is

$$\boldsymbol{J}(\boldsymbol{x}) := \boldsymbol{F}'(\boldsymbol{x}) = (m-1)\mathcal{A}\boldsymbol{x}^{m-2} \in \mathbb{R}^{n\times n}. \tag{3.2}$$

For ease of expression, we shall use $\boldsymbol{F}_k$, and $\boldsymbol{J}_k$ to represent the values of $\boldsymbol{F}(\boldsymbol{x})$ and $\boldsymbol{J}(\boldsymbol{x})$ at $\boldsymbol{x}_k$, respectively.

Following the idea of the exponentially accelerated iterative method given in [22], we obtain the following iterative scheme for (1.1).

$$\begin{cases} \boldsymbol{J}_k \Delta \boldsymbol{x}_k &= -\boldsymbol{F}_k, \\ \boldsymbol{x}_{k+1} &= \mathrm{diag}\left(\exp\left(\dfrac{\Delta \boldsymbol{x}_k(i)}{\boldsymbol{x}_k(i)}\right)\right)\boldsymbol{x}_k, \end{cases} \tag{3.3}$$

in which

$$\operatorname{diag}\left(\exp\left(\frac{\Delta \boldsymbol{x}_k(i)}{\boldsymbol{x}_k(i)}\right)\right) := \begin{pmatrix} \exp\left(\frac{\Delta \boldsymbol{x}_k(1)}{\boldsymbol{x}_k(1)}\right) & & \\ & \ddots & \\ & & \exp\left(\frac{\Delta \boldsymbol{x}_k(n)}{\boldsymbol{x}_k(n)}\right) \end{pmatrix}.$$

Then, the exponentially accelerated iterative method for solving the tensor equation (1.1) in the case that $\mathcal{A}$ is a symmetric and nonsingular $\mathcal{M}$-tensor can be concretely reported as follows:

Algorithm 1 The EAI-NS method for (1.1).

Step 1: Input symmetric tensor $\mathcal{A} \in \mathbb{R}^{[m,n]}$, and $\boldsymbol{b} \in \mathbb{R}^n$.
Let $\boldsymbol{x}_0 \in \mathbb{R}^n$ be an initial guess.
Step 2: Compute $\boldsymbol{F}_k$ and $\boldsymbol{J}_k$ by (3.1), (3.2), respectively.
Step 3: Compute $\boldsymbol{x}_k$ by (3.3).
Step 4: If $\|F_k\| < \epsilon$, stop and output $\boldsymbol{x}_k$. Otherwise, goto Step 2.

We have some comments for this algorithm:

(1) Since $\mathcal{A}$ is a nonsingular $\mathcal{M}$-tensor, the matrix $\boldsymbol{J}_k$ is a nonsingular $\mathbf{M}$-matrix [5]. In this case, the solution to the linear subproblem $\boldsymbol{J}_k \Delta \boldsymbol{x}_k = -\boldsymbol{F}_k$ in (3.3) can be expressed explicitly, i.e., $\Delta \boldsymbol{x}_k = -\boldsymbol{J}_k^{-1} \boldsymbol{F}_k$.

(2) By the definitions of $\operatorname{diag}(\cdot)$ and $\exp(\cdot)$, the iterative scheme in (3.3) is equivalent to

$$\boldsymbol{x}_{k+1}(i) = \boldsymbol{x}_k(i) \exp\left(\frac{\Delta \boldsymbol{x}_k(i)}{\boldsymbol{x}_k(i)}\right), \quad i = 1, 2, \ldots, n. \tag{3.4}$$

In the implementation of Algorithm 1, the matrices J_k may be singular or almost singular due to the influence of computer errors, one can update $\boldsymbol{x}_k$ by the following format:

$$\begin{cases} \Delta \boldsymbol{x}_k &= -\boldsymbol{J}_k^{\dagger} \boldsymbol{F}_k, \\ \boldsymbol{x}_{k+1} &= \operatorname{diag}\left(\exp\left(\frac{\Delta \boldsymbol{x}_k(i)}{\boldsymbol{x}_k(i)}\right)\right) \boldsymbol{x}_k. \end{cases} \tag{3.5}$$

Herein the superscript $'\dagger'$ denotes the Moore-Penrose inverse of a matrix [30]. Particularly, if $\boldsymbol{x}_k(i) = 0$ for some index i, let the corresponding $\boldsymbol{x}_{k+1}(i) = 0$.

(3) From Algorithm 1 and the definition of the tensor-vector product, we know that the main tensor operations is to compute $\boldsymbol{F}_k$ and $\boldsymbol{J}_k$ at each iteration, and then the amount of operations contained in this algorithm is estimated conservatively by $\mathcal{O}(n^{m-1})$.

3.2 Convergence analysis of the EAI-NS method

Using the properties of the function $\boldsymbol{F}(\boldsymbol{x})$ and $\boldsymbol{J}(\boldsymbol{x})$, we can show under some assumptions that Algorithm 1 is superlinearly convergent.

We begin with the following lemmas.

Lemma 3.1 *Let $\boldsymbol{F}(\boldsymbol{x})$ and $\boldsymbol{J}(\boldsymbol{x})$ be two functions defined in (3.1) and (3.2) respectively, and $\Omega \subset \mathbb{R}^n$ be a closed and convex set. Then, for any $\boldsymbol{x}, \boldsymbol{y} \in \Omega$, there exist $L_1 > 0$ and $L_2 > 0$ such that*

$$\begin{aligned} \|\boldsymbol{J}(\boldsymbol{y}) - \boldsymbol{J}(\boldsymbol{x})\| &\leq L_1 \|\boldsymbol{y} - \boldsymbol{x}\|, \\ \|\boldsymbol{F}(\boldsymbol{y}) - \boldsymbol{F}(\boldsymbol{x})\| &\leq L_2 \|\boldsymbol{y} - \boldsymbol{x}\|, \\ \|\boldsymbol{F}(\boldsymbol{y}) - \boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{J}(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x})\| &\leq L_1 \|\boldsymbol{y} - \boldsymbol{x}\|^2. \end{aligned}$$

Especially, $\|\boldsymbol{J}(\boldsymbol{x})\| \leq L_2$.

Proof The proofs of the first two inequalities can be derived following the ones of Corollary 3.1 in [18]. For the third one, because the function F is continuously differential, then there exist one constant $L_1 > 0$ and one vector $\hat{\boldsymbol{x}}_k$ between $\boldsymbol{x}_k$ and $\boldsymbol{x}^*$ such that

$$\begin{aligned} \|\boldsymbol{F}(\boldsymbol{x}_k) - \boldsymbol{F}(\boldsymbol{x}^*) - \boldsymbol{J}(\boldsymbol{x}^*)(\boldsymbol{x}_k - \boldsymbol{x}^*)\| &= \|\boldsymbol{J}(\hat{\boldsymbol{x}}_k)(\boldsymbol{x}_k - \boldsymbol{x}^*) - \boldsymbol{J}(\boldsymbol{x}^*)(\boldsymbol{x}_k - \boldsymbol{x}^*)\| \\ &= \|[\boldsymbol{J}(\hat{\boldsymbol{x}}_k) - \boldsymbol{J}(\boldsymbol{x}^*)](\boldsymbol{x}_k - \boldsymbol{x}^*)\| \\ &\leq \|\boldsymbol{J}(\hat{\boldsymbol{x}}_k) - \boldsymbol{J}(\boldsymbol{x}^*)\| \|\boldsymbol{x}_k - \boldsymbol{x}^*\| \\ &\leq L_1 \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2. \end{aligned}$$

The last inequality can be found in [29]. □

Lemma 3.2 *([12]) Suppose that $\boldsymbol{x}^*$ is a solution of the tensor equation (1.1) with nonsingular M-tensor $\mathcal{A}$. Then, $\boldsymbol{J}(\boldsymbol{x})$ defined in (3.2) is a nonsingular M-matrix for any $\boldsymbol{x} \neq 0$, and there exist positive numbers δ and C such that $\|\boldsymbol{J}(\boldsymbol{x})^{-1}\| \leq C$ for all $\boldsymbol{x}$ satisfying $\|\boldsymbol{x} - \boldsymbol{x}^*\| \leq \delta$.*

By using Lemmas 3.1 and 3.2, we obtain the following theorem.

Theorem 3.3 *Let $\mathcal{A} \in \mathbb{R}^{[m,n]}$ be a symmetric and nonsingular $\mathcal{M}$-tensor, and $\boldsymbol{b} \in \mathbb{R}^n$, and assume that $\boldsymbol{x}^*$ is a solution of the tensor equation (1.1). Then, Algorithm 1 is superlinearly convergent.*

Proof By Algorithm 1 and Lemma 3.1, we have

$$\Delta \boldsymbol{x}_k = -\boldsymbol{J}_k^{-1} \boldsymbol{F}_k = -\boldsymbol{J}_k^{-1}[\boldsymbol{F}_k - \boldsymbol{F}(\boldsymbol{x}^*)],$$

and then

$$\|\Delta \boldsymbol{x}_k\| \leq L_2 \|\boldsymbol{J}_k^{-1}\| \|\boldsymbol{x}_k - \boldsymbol{x}^*\|. \tag{3.6}$$

By the Taylor theorem of the function $\exp(z)$ with the variable $z \in \mathbb{R}$, i.e., $\exp(z) = 1 + z + o(z)$, the equality (3.4) can be concretely expressed as

$$\boldsymbol{x}_{k+1}(i) = \boldsymbol{x}_k(i) + \Delta \boldsymbol{x}_k(i) + o(\Delta \boldsymbol{x}_k(i)), \tag{3.7}$$

then the iterative scheme (3.3) is rewritten as

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \begin{pmatrix} \boldsymbol{x}_k(1) + \Delta\boldsymbol{x}_k(1) + o(\Delta\boldsymbol{x}_k(1)) \\ \vdots \\ \boldsymbol{x}_k(n) + \Delta\boldsymbol{x}_k(n) + o(\Delta\boldsymbol{x}_k(n)) \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{x}_k(1) - \boldsymbol{J}_k^{-1}(1,:)\boldsymbol{F}_k + o(\Delta\boldsymbol{x}_k(1)) \\ \vdots \\ \boldsymbol{x}_k(n) - \boldsymbol{J}_k^{-1}(n,:)\boldsymbol{F}_k + o(\Delta\boldsymbol{x}_k(n)) \end{pmatrix} \\ &= \boldsymbol{x}_k - \boldsymbol{J}_k^{-1}\boldsymbol{F}_k + o(\Delta\boldsymbol{x}_k).\end{aligned}$$

At this time, we obtain

$$\begin{aligned}\boldsymbol{x}_{k+1} - \boldsymbol{x}^* &= \boldsymbol{x}_k - \boldsymbol{x}^* - \boldsymbol{J}_k^{-1}\boldsymbol{F}_k + o(\Delta\boldsymbol{x}_k) \\ &= \boldsymbol{J}_k^{-1}\boldsymbol{J}_k(\boldsymbol{x}_k - \boldsymbol{x}^*) - \boldsymbol{J}_k^{-1}\boldsymbol{F}_k + o(\Delta\boldsymbol{x}_k) \\ &= \boldsymbol{J}_k^{-1}[\boldsymbol{J}_k(\boldsymbol{x}_k - \boldsymbol{x}^*) - \boldsymbol{F}_k] + o(\Delta\boldsymbol{x}_k) \\ &= \boldsymbol{J}_k^{-1}[\boldsymbol{J}_k(\boldsymbol{x}_k - \boldsymbol{x}^*) - \boldsymbol{F}_k + F(\boldsymbol{x}^*)] + o(\Delta\boldsymbol{x}_k).\end{aligned}$$

Using Lemma 3.1 again and noting that the tensor $\mathcal{A}$ is a nonsingular $\mathcal{M}$-tensor, one can derive

$$\begin{aligned}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| &\le \|\boldsymbol{J}_k^{-1}\|\|\boldsymbol{J}_k(\boldsymbol{x}_k - \boldsymbol{x}^*) - \boldsymbol{F}_k + \boldsymbol{F}(\boldsymbol{x}^*)\| + o(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|) \\ &\le L_1\|\boldsymbol{J}_k^{-1}\|\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 + o(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|) \\ &= O(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2) + o(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|) \\ &= o(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|),\end{aligned}$$

which indicates that Algorithm 1 converges suplinearly.

3.3 The EAI method for (1.1) with singular $\mathcal{M}$-tensor $\mathcal{A}$

In this subsection, we are going to establish the exponentially accelerated method for solving the tensor equation (1.1) in the case that the coefficient tensor $\mathcal{A}$ is a symmetric and singular $\mathcal{M}$-tensor. Following the classical LM method [26, 27] and the EAI method proposed in Section 3.1, the main iterative steps of the iteration is constructed by

$$\begin{cases} (\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k\boldsymbol{I}_n)\Delta\boldsymbol{x}_k &= -\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{F}_k, \\ \boldsymbol{x}_{k+1} &= \operatorname{diag}\left(\exp\left(\dfrac{\Delta\boldsymbol{x}_k(i)}{\boldsymbol{x}_k(i)}\right)\right)\boldsymbol{x}_k, \end{cases} \tag{3.8}$$

in which let the corresponding $\boldsymbol{x}_{k+1}(i) = 0$ if $\boldsymbol{x}_k(i) = 0$ for some index i. Then, the new iterative method for solving the tensor equation (1.1) with symmetric and singular $\mathcal{M}$-tensor $\mathcal{A}$ (denoted by EAI-S for short) is described as follows:

Algorithm 2 The EAI-S method for (1.1).

Step 1: Input symmetric tensor $\mathcal{A} \in \mathbb{R}^{[m,n]}$, and $\boldsymbol{b} \in \mathbb{R}^n$.
Let $\mu_0 > 0$ and $\boldsymbol{x}_0 \in \mathbb{R}^n$ be an initial guess.
Step 2: Compute $\boldsymbol{F}_k$ and $\boldsymbol{J}_k$ by (3.1), (3.2), respectively.
Step 3: Compute $\boldsymbol{x}_k$ by (3.8).
Step 4: If $\|\boldsymbol{F}_k\| < \epsilon$, stop and output $\boldsymbol{x}_k$. Otherwise, goto Step 2.

In Algorithm 2, if let $\mu_k = 0$ and J_k be nonsingular, then it reduces to Algorithm 1, that is, it is an extension of the latter. In the case of $\mu_k = 0$ and J_k is singular, one can replace (3.8) with (3.5). Moreover, in the third step of Algorithm 2, we needs to calculate $\Delta\boldsymbol{x}_k$ by solving the equation system $(\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k\boldsymbol{I}_n)\Delta\boldsymbol{x}_k = -\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{F}_k$. The coefficient matrix here is symmetric and positive definite, so it can be solved by using the classical iterative methods (e.g., the CG method [30]) when the size n is large.

In the implementation of this algorithm, as in LM-type methods (e.g., [18]), we can make the parameter μ_k change with iteration steps. Of course, it can also be an invariant positive constant for simplicity. In addition, the computational complexity of Algorithm 2 is the same as that of Algorithm 1.

3.4 Convergence analysis of the EAI-S method

In this subsection, we discuss the convergence of the EAI-S method. The line of the mind to prove the convergence is similar to that of the EAI-NS method.

Theorem 3.4 *Let $\mathcal{A} \in \mathbb{R}^{[m,n]}$ be a symmetric and singular $\mathcal{M}$-tensor, and $\boldsymbol{b} \in \mathbb{R}^n$, and assume that $\boldsymbol{x}^*$ is a solution of the tensor equation (1.1). Then, Algorithm 2 is linearly convergent for $\mu_k > 0$.*

Proof Depending on the Algorithm 2 and the required assumptions, we have

$$\Delta\boldsymbol{x}_k = -(\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k\boldsymbol{I}_n)^{-1}\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{F}_k = -(\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k\boldsymbol{I}_n)^{-1}\boldsymbol{J}_k^{\mathrm{T}}(\boldsymbol{F}_k - \boldsymbol{F}(\boldsymbol{x}^*)),$$

so it follows that

$$\|\Delta\boldsymbol{x}_k\| \le L_2^2\|(\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k\boldsymbol{I}_n)^{-1}\|\|\boldsymbol{x}_k - \boldsymbol{x}^*\|. \tag{3.9}$$

The iterative scheme (3.8) can be componentwise written as the same form as (3.4). Analogously, by the Taylor theorem of the function $\exp(z)$ with the variable $z \in \mathbb{R}$, we can rewrite the iterative scheme (3.8) as (3.7), that is,

$$\boldsymbol{x}_{k+1}(i) = \boldsymbol{x}_k(i) + \Delta\boldsymbol{x}_k(i) + o(\Delta\boldsymbol{x}_k(i)),$$

and so the iterative scheme (3.8) is equivalent to

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k \boldsymbol{I}_n)^{-1}\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{F}_k + o(\Delta \boldsymbol{x}_k). \tag{3.10}$$

Furthermore, we obtain

$$\begin{aligned}
\boldsymbol{x}_{k+1} - \boldsymbol{x}^* &= \boldsymbol{x}_k - \boldsymbol{x}^* - (\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k \boldsymbol{I}_n)^{-1}\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{F}_k + o(\Delta \boldsymbol{x}_k) \\
&= (\boldsymbol{x}_k - \boldsymbol{x}^*) - (\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k \boldsymbol{I}_n)^{-1}\boldsymbol{J}_k^{\mathrm{T}}[\boldsymbol{F}_k - \boldsymbol{F}(\boldsymbol{x}^*)] + o(\Delta \boldsymbol{x}_k) \\
&= (\boldsymbol{x}_k - \boldsymbol{x}^*) - (\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k \boldsymbol{I}_n)^{-1}\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}(\tilde{\boldsymbol{x}}_k)(\boldsymbol{x}_k - \boldsymbol{x}^*) + o(\Delta \boldsymbol{x}_k) \\
&= [\boldsymbol{I}_n - (\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k \boldsymbol{I}_n)^{-1}\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}(\tilde{\boldsymbol{x}}_k)](\boldsymbol{x}_k - \boldsymbol{x}^*) + o(\Delta \boldsymbol{x}_k),
\end{aligned}$$

in which $\tilde{\boldsymbol{x}}_k$ is the Mean Point between $\boldsymbol{x}_k$ and $\boldsymbol{x}^*$.

Noticing that (3.9) and taking the norm at both ends of the above equality, we have

$$\begin{aligned}
\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| &\le \|\boldsymbol{I}_n - (\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k \boldsymbol{I}_n)^{-1}\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}(\tilde{\boldsymbol{x}}_k)\|\|\boldsymbol{x}_k - \boldsymbol{x}^*\| + o(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|) \\
&\le (1 + \|(\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k \boldsymbol{I}_n)^{-1}\|\|\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}(\tilde{\boldsymbol{x}}_k)\|)\|\boldsymbol{x}_k - \boldsymbol{x}^*\| + o(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|) \\
&\le c\|\boldsymbol{x}_k - \boldsymbol{x}^*\| + o(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|) \\
&= O(\|\boldsymbol{x}_k - \boldsymbol{x}^*\|),
\end{aligned} \tag{3.11}$$

where $c := 1 + L_2^2\|(\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k \boldsymbol{I}_n)^{-1}\|$, which derives from Lemma 3.1 and the continuity of $J(\boldsymbol{x})$. The inequality (3.11) reflects that the Algorithm 2 is linearly convergent.

Remark 3.5 *Let the singular values of the matrix J_k be $\sigma_i^{(k)}$ $(i = 1, 2, \ldots, n)$ which satisfy $\sigma_1^{(k)} \ge \sigma_2^{(k)} \ge \ldots \ge \sigma_n^{(k)}$, then*

$$c := 1 + L_2^2\|(\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k + \mu_k \boldsymbol{I}_n)^{-1}\| \le 1 + \frac{L_2^2}{\mu_k + (\sigma_n^{(k)})^2}.$$

Suppose that $\boldsymbol{x}^*$ is a solution of the tensor equation (1.1) with a singular and symmetric $\mathcal{M}$-tensor $\mathcal{A}$, and we assume that the singular values σ_i of the Jacobian $\boldsymbol{J}(\boldsymbol{x}^*)$ satisfy the following unequal relationship:

$$\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_r > 0 = \sigma_{r+1} = \cdots = \sigma_n,$$

then $c \to 1 + \dfrac{L_2^2}{\mu}$ as $k \to \infty$ and $\mu_k \to \mu$.

4 Numerical experiments

In this section, several numerical experiments will be given to illustrate the efficiency of the proposed iterative methods, i.e., Algorithms 1 and 2. All the codes were written

in MATLAB (version R2016a) and run on a personal computer whose specifications are as follows: Intel(R) Core(TM) i7-10510U@1.80GHz and 8.00G memory. The tensor operations appeared in our tests were carried out via the tensor toolbox (version 3.2) [31].

In the numerical results, the symbols "IT" and "CPU" represent the number of iteration steps and the elapsed CPU time in seconds, respectively. The residual of the tensor equation (1.1) at $\boldsymbol{x}_k$ is denoted by "RES", i.e., RES $= \|\mathcal{A}\boldsymbol{x}_k^{m-1} - \boldsymbol{b}\|$. The stopping criteria is when the tolerance $\epsilon \leq 1.0\text{e}-08$, or the iteration number exceeds the prescribed iteration $k_{\max} = 10000$. In the following tests, we let the parameter μ_k be a positive constant chosen by `rand` for simplicity. Additionally, the number of iteration steps, the CPU time, and the residual listed in the tables below are respectively the average of 5 runs from different starting points unless otherwise stated.

Example 4.1 *Let the tensor $\mathcal{A} \in \mathbb{R}^{[m,n]}$ in (1.1) be $\mathcal{A} = s\mathcal{I} - \mathcal{B}$, where $\mathcal{B} \in \mathbb{R}^{[m,n]}$ is a nonnegative tensor with nonnegative entries $\mathcal{B}_{i_1 i_2 \ldots i_m} = |\sin(i_1 + i_2 + \cdots + i_m)|$, and choose the vector $\boldsymbol{b}$ such that $\boldsymbol{x}^* = 8 * \boldsymbol{ones}(n, 1) \in \mathbb{R}^n$ is a solution of the tensor equation mentioned above.*

In view of the definition of the tensor $\mathcal{B}$, the coefficient tensor $\mathcal{A}$ given here is a symmetric $\mathcal{M}$-tensor. In this numerical example, the following two cases were considered:

Case I. Let $s = n^{m-1}$, then the tensor $\mathcal{A}$ is a symmetric and nonsingular $\mathcal{M}$-tensor, and the corresponding tensor equation (1.1) always has a solution [8].

For randomly chosen initial iterative vectors $\boldsymbol{x}_0$ and the parameter $\mu = \texttt{rand}$ (the random function involved in Matlab) in Algorithm 1, we compared the EAI-NS method with the promising iterative algorithms, that is, the steepest descent method (denoted by "SD" for short) [32], the conjugate gradient method (denoted by "CG" for short) [32], the SOR method (denoted by "SOR" for short) [15], the Newton method (denoted by "NT" for short) [8], all those algorithms are feasible for the tensor equations with symmetric coefficient tensors. The numerical results were reported in the Tables 1 and 2, in which the symbol "—" means that $\boldsymbol{x}_k$ does not satisfy the terminated criterion although the number of iteration steps reaches the maximum $k_{\max}$.

From the Tables 1 and 2 one can observe that all the methods converge except the SOR method for the cases $m = 4, 5$ and $[m, n] = [3, 100]$, the reason may be that the relaxation parameter ω there are selected randomly by the function $\omega = \texttt{rand}$, and so they are not optimal. Notably, the SD method and the CG method do have very good convergence. The Newton's method is superior to other methods in terms of the number of iteration steps and the CPU time consumed, but the EAI-NS method and the Newton's method have similar convergence. In addition, the EAI-NS method proposed in this article has better performance than the SD method, the CG method as well as the SOR method.

Furthermore, in order to better show the convergence behavior of the algorithms mentioned in the Table 1, we plotted their convergence curves v.s. the number of iteration steps k in Fig. 1. These curves show that the SD method and the SOR method

Table 1 Nonsingular case (I): numerical results for the tensor equations in Example 4.1

Algs	[m,n]	[3, 10]	[3, 30]	[3, 50]	[3, 100]	[3, 200]
SD	IT	55	60	65	39	41
	CPU	0.1569	0.2311	0.5587	0.4724	2.1249
	RES	7.46e−09	7.56e−09	6.41e−09	5.24e−09	8.59e−09
CG	IT	29	31	40	24	27
	CPU	0.0952	0.1442	0.3143	0.2963	1.4166
	RES	2.29e−09	3.63e−09	5.48e−09	9.16e−09	7.13e−09
SOR	IT	61	68	74	90	—
	CPU	0.2480	0.4486	1.1308	2.5798	—
	RES	8.18e−09	6.30e−09	5.55e−09	6.44e−09	—
NT	IT	16	18	19	13	17
	CPU	0.0409	0.0554	0.0898	0.1299	0.5190
	RES	2.08e−12	7.61e−11	3.75e−10	9.14e−10	3.31e−09
EAI-NS	IT	20	23	25	8	7
	CPU	0.0754	0.1292	0.1750	0.1083	0.3567
	RES	3.22e−12	7.94e−10	2.80e−10	2.45e−09	4.32e−09

have linear convergence, while the other methods have superlinear convergence, which is consistent with the theoretical results presented in Sect. 3.2.

Case II. Let the constant number $s = \rho(\mathcal{B})$ in $\mathcal{A} = s\mathcal{I} - \mathcal{B}$, which will be gained by using the NQZ method [33]) (see Table 3 for details). At this time, the coefficient tensors $\mathcal{A}$ is a singular and symmetric $\mathcal{M}$-tensors.

Table 2 Nonsingular case (II): numerical results for the tensor equations in Example 4.1

Algs	[m,n]	[4, 10]	[4, 30]	[4, 50]	[5, 10]	[5, 20]
SD	IT	75	45	52	66	50
	CPU	0.6215	0.7563	5.1138	0.8060	3.6326
	RES	2.59e−09	9.94e−09	0.00e+00	9.86e−09	0.00e+00
CG	IT	49	14	43	33	39
	CPU	0.4488	0.2258	4.4575	0.3718	2.9504
	RES	5.95e−09	5.89e−09	0.00e+00	2.51e−08	0.00e+00
SOR	IT	94	—	—	—	—
	CPU	0.6338	—	—	—	—
	RES	5.95e−09	—	—	—	—
NT	IT	34	9	—	19	—
	CPU	0.2317	0.1176	—	0.2007	—
	RES	3.05e−10	9.81e−09	—	9.31e−09	—
EAI-NS	IT	40	8	11	16	9
	CPU	0.3640	0.1312	1.1011	0.2074	0.6540
	RES	4.69e−09	8.17e−09	0.00e+00	7.90e−09	0.00e+00

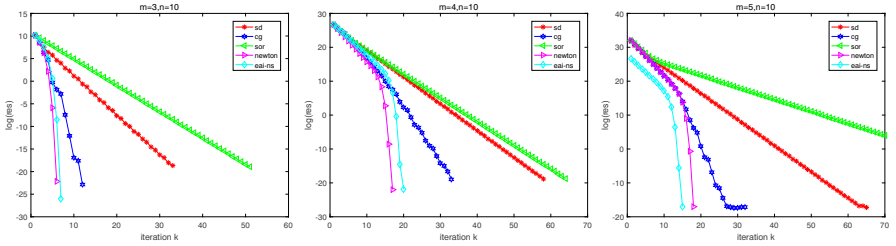

Fig. 1 Nonsingular case: the convergence behavior of the proposed methods in Example 4.1

As is well-known, the Newton method is feasible for the tensor equations with nonsingular coefficient tensors [8], In view of the singularity of the coefficient tensor $\mathcal{A}$, we compared the EAI-S method (i.e., Algorithm 2) with the LM-type method (denoted by "LM" for short) [18], the TALM method (denoted by "TALM" for short) [20] starting from the initial iterative vector $\boldsymbol{x}_0$ and the parameter μ_k chosen randomly, and listed the numerical results in Table 4.

This table reflects that the EAI-S method has the best performance compared with the LM-type method and the TALM method both in the number of iteration steps and the elapsed CPU time. It is worth mentioning that the EAI-S method spends less CPU time when the iterative steps of the two methods are similar. Certainly, the TALM method outperforms the LM method under the environments presented in this example.

In addition, we described the curves of the logarithm of the residual RES of the three methods versus the iteration k in Fig. 2, which displays that the EAI-S method proposed in present paper has better performance. It should be pointed out that we only prove the linear convergence of the EAI-S method in Section 3.4, but from the figures one can observe that this method seems superlinearly convergent. This is an issue that we need to further consider in our future work. Additionally, the TALM method converges cubically [20], and the EAI-S method possesses analogous convergence behavior, so how to prove the convergent rate of this iterative method is an interesting thing.

Moreover, as stated in Sect. 3, the parameter μ_k appeared in Algorithm 2 could be chosen as an arbitrary positive number for simplicity. To numerically verify the influence of the parameter μ_k on the convergence of this algorithm, let the initial vector $\boldsymbol{x}_0 = \texttt{ones}(n, 1)$.

We respectively display the convergence behavior of Algorithm 2 when choosing variable $\mu_k = \|F_k\|$, $\mu_k = \|F_k\|^{1.5}$ and $\mu_k = \|F_k\|^2$ v.s. invariant $\mu_k = 0.05$ (see Fig. 3 for $[m, n] = [3, 10]$, and Fig. 4 for $[m, n] = [5, 10]$). From those figures we can observe that the variable parameter μ_k are beneficial for improving the convergence

Table 3 The spectral radius of the tensor $\mathcal{B}$ in Example 4.1

[m,n]	[3, 10]	[3, 20]	[3, 30]	[3, 40]	[4, 10]	[4, 20]	[4, 30]	[5, 10]
$\rho(\mathcal{B})$	63.6688	254.409	572.9008	1018.3	626.8249	5089.7	17184.0	6355.9

Table 4 Singular case: numerical results for the tensor equations in Example 4.1

Algorithms	LM			TALM			EAI-S		
$[m, n]$	IT	CPU	RES	IT	CPU	RES	IT	CPU	RES
[3, 10]	189	0.8758	2.47e−12	22	0.1244	1.53e−12	16	0.0495	1.20e−12
[3, 20]	269	1.7454	8.33e−12	55	0.4546	5.68e−12	32	0.1463	3.78e−10
[3, 30]	349	2.2546	8.96e−09	106	0.9705	4.66e−11	106	0.5230	9.86e−09
[3, 40]	502	4.9578	2.41e−09	17	0.1082	5.60e−09	41	0.9984	9.97e−09
[4, 10]	144	1.3208	2.32e−10	21	0.2314	2.22e−10	31	0.2150	3.75e−10
[4, 20]	197	2.8867	2.64e−09	37	0.7102	2.74e−09	39	0.4686	2.44e−09
[4, 30]	467	12.6084	9.73e−09	86	2.8206	9.87e−09	38	0.7855	9.92e−09
[5, 10]	112	1.6761	9.76e−09	32	0.5574	9.72e−09	47	0.7368	8.68e−09

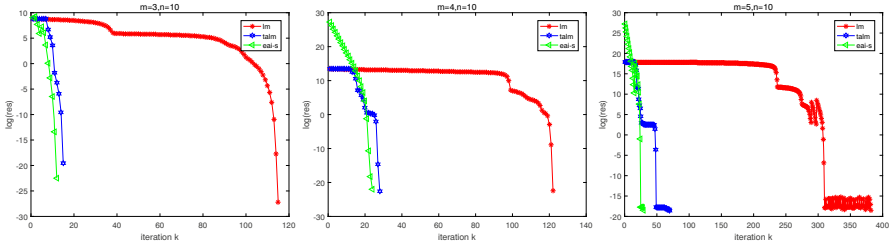

Fig. 2 Singular case: the convergence behavior of the proposed methods in Example 4.1

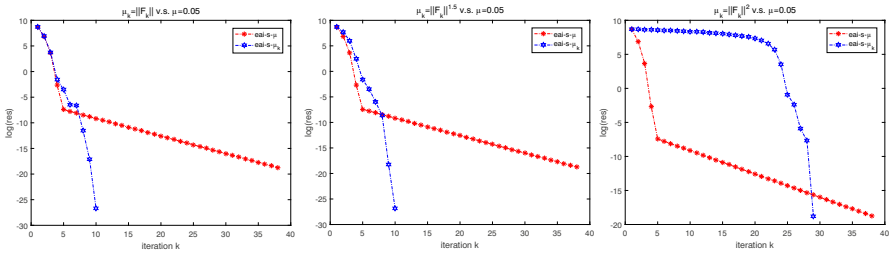

Fig. 3 Comparison for the variable and immutable parameter μ_k when $[m, n] = [3, 10]$

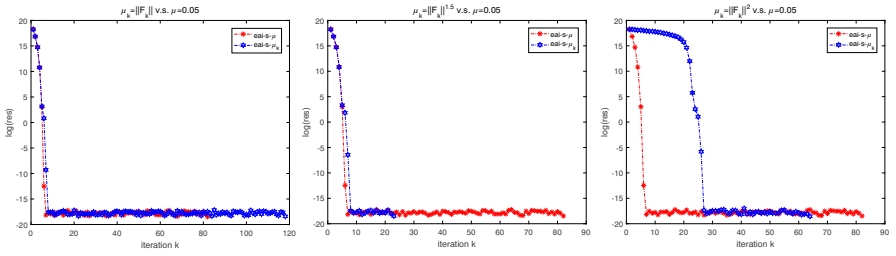

Fig. 4 Comparison for the variable and immutable parameter μ_k when $[m, n] = [5, 10]$

of the algorithm. It is a considerable problem to establish the corresponding theory for seeking the optimal parameter, which will be studied in the further work.

The following two numerical examples are derived from practical application problems.

Example 4.2 *Consider the numerical solution of the following differential equation*

$$\begin{cases} -\max\limits_{(\gamma,\lambda)\in(\Gamma,\Lambda)}\{L^{\lambda}U-\eta U-\frac{1}{2}\alpha\gamma^2U+\beta\gamma\}=0, & \text{on } \Omega,\\ U=g, & \text{on } \partial\Omega, \end{cases}$$

in which $L^{\lambda}U(x)=\frac{1}{2}\sigma(x,\lambda)^2U''(x)+\mu(x,\lambda)U'(x)$, $\Omega=(0,1)$, $\Gamma=[0,+\infty)$, *and* Λ *is a compact metric space.*

Applying "optimize then discretize" approach to the above differential equation, it can be discretized as the 3rd-order Bellman equation [10]

$$\max_{\lambda\in\Lambda_{\Delta x}}\mathcal{A}(\lambda)\mathbf{u}^2=\boldsymbol{b},$$

where $\mathbf{u}=(\mathbf{u}_i)\in\mathbb{R}^{n+1}$, $u_i\approx U(i\Delta x)$ with $\Delta x=\frac{1}{n}$ for $i=0,1,2,\ldots,n$, $\mathcal{A}(\lambda)$ is a 3rd-order and $(n+1)$-dimensional parameterized tensor whose entries are defined as follows:

$$\begin{aligned}
&\mathcal{A}_{i,i-1,i}(\lambda)=\mathcal{A}_{i,i,i-1}(\lambda),\\
&2\mathcal{A}_{i,i,i-1}(\lambda)=-\frac{1}{2}\sigma_i^2(\lambda_i)\frac{1}{(\Delta x)^2}+\mu_i(\lambda_i)\frac{1}{\Delta x}\mathbf{1}_{(-\infty,0)}(\mu_i(\lambda_i)),\\
&\mathcal{A}_{i,i,i}(\lambda)=\frac{1}{2}\sigma_i^2(\lambda_i)\frac{2}{(\Delta x)^2}+|\mu_i(\lambda_i)|\frac{1}{\Delta x}+\eta_i,\\
&2\mathcal{A}_{i,i,i+1}(\lambda)=-\frac{1}{2}\sigma_i^2(\lambda_i)\frac{1}{(\Delta x)^2}-\mu_i(\lambda_i)\frac{1}{\Delta x}\mathbf{1}_{(0,+\infty)}(\mu_i(\lambda_i)),\\
&\mathcal{A}_{i,i+1,i}(\lambda)=\mathcal{A}_{i,i,i+1}(\lambda),\\
&i=1,2,\ldots,n-1,
\end{aligned}$$

here $\mathbf{1}_{\mathbb{S}}$ stands for the indicator function over the set $\mathbb{S}$, $\sigma_i(\lambda)=\sigma(i\Delta x,\lambda)$, $\eta_i=\eta(i\Delta x)$, and $\mathcal{A}_{i,i,i}(\lambda)=1$ with $i=0,n$. The vector $\boldsymbol{b}=(b_i)$ is given by

$$b_i=\begin{cases}\frac{1}{2}\frac{\beta_i^2}{\alpha_i}, & \text{if } i=1,2,\ldots,n-1,\\ g_i^2, & \text{if } i=0,n,\end{cases}$$

in which $\alpha_i=\alpha(i\Delta x)$, $\beta_i=\beta(i\Delta x)$, $g_i=g(i\Delta x)$.

In our tests, let $\sigma(x,\lambda)=0.2$, $\alpha(x)=2-x$, $\eta(x)=0.04$, $u(x,\lambda)=0.04\lambda$, $\beta(x)=1+x$, $g(x)=1$, and $\lambda=-1$. The coefficient tensor $\mathcal{A}$ is a non-symmetric

and nonsingular $\mathcal{M}$-tensor, and thus the derived tensor equation has a nonnegative solution [8].

Although the Newton method mentioned above is very effective as shown in Example 4.1, it is theoretically infeasible for asymmetric cases. Therefore, we compared the EAI-NS method with the SD method [32], the CG method [32], and the SOR method [15] starting from different initial vectors being chosen as the same as in Example 4.1, and displayed the corresponding results in Table 5.

From Table 5, one can see that the number of iteration steps and the corresponding CPU time increase as the n increases from 10 to 60, and all the tested iterative algorithms are convergent for the 3rd-order Bellman tensor equation, in which the CG method takes less iterations and CPU time than the SD method and the SOR method in the majority of cases. Nevertheless, the EAI-NS method has better performance than the CG method.

It should point out that when $n = 70$, except for the EAI-NS method, the other ones fail to stop before reaching the maximum number $k_{\max}$ of iteration steps. Particularly, a large number of unlisted numerical results also reflect similar phenomena, so our algorithm can effectively solve such problems. Moreover, in Fig. 5, we drew the convergence curves of the logarithm of the residual RES of the aforementioned algorithms versus the iteration k. These images also demonstrate the superlinear convergence of Algorithm 1.

Example 4.3 *Consider the numerical solution of the Klein-Gordon equation [34, 35]*

$$\begin{cases} u(\boldsymbol{x})^{m-2} \cdot \triangle u(\boldsymbol{x}) = -f(\boldsymbol{x}), & \text{in } \Omega, \\ u(\boldsymbol{x}) = g(\boldsymbol{x}), & \text{on } \partial\Omega, \end{cases}$$

in which $f(\boldsymbol{x})$ *is a constant function,* $\triangle = \sum_{k=0}^{d} \frac{\partial^2}{\partial x_k^2}$, $\Omega = [0, 1]^d$ *and* $m = 3, 4, \ldots$.

Table 5 Comparison of the proposed methods for the tensor equations in Example 4.2

Algs	[m,n]	[3, 10]	[3, 20]	[3, 30]	[3, 40]	[3, 50]	[3, 60]	[3, 70]
SD	IT	179	905	2473	3535	6003	7839	—
	CPU	1.1862	5.9266	18.8378	11.9631	29.3062	42.0809	—
	RES	8.36e−09	9.83e−09	9.87e−09	9.93e−09	9.95e−09	9.97e−09	—
CG	IT	233	699	1476	479	2639	874	—
	CPU	1.5609	4.6013	11.4889	1.6369	14.5399	4.5512	—
	RES	5.42e−09	8.82e−09	7.10e−09	9.07e−09	9.96e−09	9.44e−09	—
SOR	IT	274	1161	2797	4463	6677	9371	—
	CPU	1.8038	7.6930	26.0691	17.5678	36.6196	49.0069	—
	RES	9.40e−09	9.83e−09	9.99e−09	9.96e−09	9.99e−09	9.99e−09	—
EAI	IT	10	11	12	13	13	14	14
	CPU	0.0584	0.0709	0.1056	0.0596	0.0710	0.0734	0.1021
	RES	8.64e−14	3.46e−10	2.59e−11	2.61e−13	9.56e−11	6.52e−13	9.61e−13

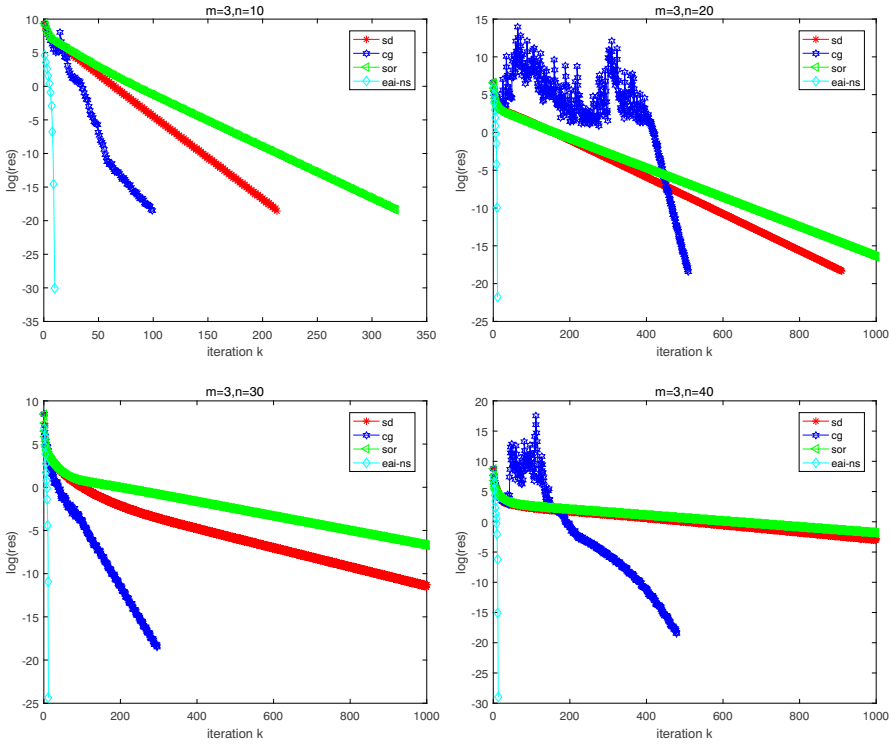

Fig. 5 The convergence behavior of the proposed methods in Example 4.2

When $d = 1$, the above Klein-Gordon equation is discretized as the tensor equation $\mathcal{L}_h \boldsymbol{u}^{m-1} = \boldsymbol{f}$, where $h = 1/(n-1)$, and $\mathcal{L}_h \in \mathbb{R}^{[m,n]}$ is a nonsymmetric and nonsingular $\mathcal{M}$-tensor, i.e.,

$$\begin{aligned}
(\mathcal{L}_h)_{11\ldots1} &= (\mathcal{L}_h)_{nn\ldots n} = 1/h^2, \quad (\mathcal{L}_h)_{ii\ldots i} = 2/h^2,\ i = 2, 3, \ldots, n-1,\\
(\mathcal{L}_h)_{ii-1i\ldots i} &= (\mathcal{L}_h)_{iii-1i\ldots i} = \cdots = (\mathcal{L}_h)_{i\ldots ii-1} = -1/h^2(m-1),\ i = 2, 3, \ldots, n-1,\\
(\mathcal{L}_h)_{ii+1i\ldots i} &= (\mathcal{L}_h)_{iii+1i\ldots i} = \cdots = (\mathcal{L}_h)_{i\ldots ii+1} = -1/h^2(m-1),\ i = 2, 3, \ldots, n-1.
\end{aligned}$$

In the following tests, we choose the vector $\mathbf{f}$ as the vector such that $\mathcal{L}_h(\mathbf{u}^*)^{m-1} = \mathbf{f}$ for $\mathbf{u}^* = 8 * \mathtt{ones}(n, 1)$ for simplicity.

Starting from the randomly initial iterative vectors chosen as the same as in Example 4.1, we performed the EAI-NS method, the SD method [32], the CG method [32], and the SOR method [15], and reported the numerical results in the Table 6.

From this table, we can observe that the CG method has better performance than the SD method and the SOR method for the convergent cases. Moreover, both the CG method and the EAI-NS method converge before the numbers of iteration steps reach the maximum value $k_{\max}$, and especially, our method takes less iteration steps as well as the CPU time.

Table 6 Comparison of the proposed methods for the tensor equations in Example 4.3

Algorithms	SD	CG	SOR	EAI
IT	255	96	658	28
[3, 10] CPU	0.6464	0.2439	1.6520	0.0698
RES	9.41e−09	7.69e−09	9.83e−09	9.25e−09
IT	3437	290	6324	32
[3, 30] CPU	12.9520	1.1527	24.9951	0.1272
RES	9.92e−09	9.14e−09	9.98e−09	4.53e−09
IT	—	588	—	33
[3, 50] CPU	—	2.6803	—	0.2125
RES	—	9.98e−09	—	8.60e−09
IT	—	928	—	36
[3, 100] CPU	—	10.0644	—	0.3845
RES	—	9.63e−09	—	9.05e−09
IT	442	128	995	46
[4, 10] CPU	2.4791	0.7214	4.0046	0.2529
RES	9.26e−09	8.25e−09	9.87e−09	8.96e−09
IT	4227	444	8692	57
[4, 30] CPU	79.9428	8.3941	110.1621	1.1408
RES	9.93e−09	9.95e−09	9.97e−09	8.02e−09
IT	6516	774	—	42
[4, 50] CPU	285.9569	33.5275	—	5.1963
RES	9.72e−09	8.75e−09	—	7.07e−09
IT	338	97	1197	76
[5, 10] CPU	5.3420	1.5261	9.9394	1.2232
RES	9.15e−09	9.20e−09	9.78e−09	7.48e−09
IT	546	151	—	43
[6, 10] CPU	26.3843	7.3154	—	2.1273
RES	7.90e−09	7.79e−09	—	2.61e−09

Additionally, we also described the convergence curves of the logarithm of the residual RES corresponding to the four iterative approaches versus the iteration k in Fig. 6, from which we can see that the EAI-NS method has the best convergent.

5 Conclusions and remarks

Based on the exponential acceleration, in present paper, we develop the exponential accelerated iterative methods for the tensor equation (1.1) under two different cases: One is that the coefficient tensor $\mathcal{A}$ is a symmetric and nonsingular $\mathcal{M}$-tensor, and the other one is that the coefficient tensor $\mathcal{A}$ is a symmetric and singular $\mathcal{M}$-tensor, that is, Algorithms 1 and 2. These two iterative methods are extension of the Newton's

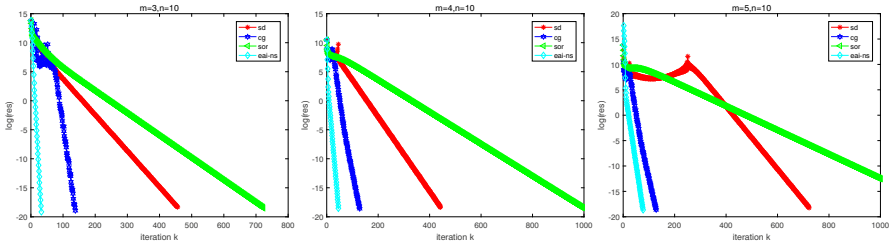

Fig. 6 The convergence behavior of the proposed methods in Example 4.3

method, and Algorithm 1 possesses superlinear convergence, while Algorithm 2 is linearly convergent. The provided numerical results and many other trials that did not list in present paper demonstrate that the proposed methods are effective for solving tensor equations with the form of (1.1), and have better performance than some existing ones.

We should mention that since the key operations in those two algorithms contain the tensor-vector multiplications, which means that the computational amount of the proposed methods grow exponentially as the increasing dimension, that is, they suffer from the so-called "curse-of-dimensionality" [36]. So it is an interesting but important theme to overcome the curse.

Acknowledgements The authors are thankful to the handling editor and the referees for their constructive comments and suggestions, which greatly improve the quality of this paper.

Author contribution M. Liang provided the methodology along with the problem under consideration and re-editing and correcting the manuscript, L. Dai implemented the scheme and edited the manuscript, and R. Zhao completed the numerical experiments of the related algorithms.

Funding This work was supported by National Natural Science Foundation of China (Nos. 11961057, 12201267, 12361081), the Natural Science Foundation of Gansu Province (Nos. 21JR1RE287, 22JR5RA559) the Innovation Foundation of Education Department of Gansu Province (Nos. 2021B-221, 2023B-135), and the Science Foundation of Tianshui Normal University (No. CXJ2021-01).

Data availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Ethical approval Not applicable

Conflict of interest The authors declare no competing interests.

References

1. Kolda, T., Bader, B.: Tensor decompositions and applications. SIAM Rev. **51**, 455–500 (2009)
2. Lathauwer, L., Castaing, J., Cardoso, J.: Fourth-order cumulant-based blind identification of underdetermined mixtures. IEEE Trans. Sig. Proc. **55**, 2965–2973 (2007)
3. Qi, L., Luo, Z.: Tensor analysis: spectral theory and special tensors. SIAM, Philadelphia (2017)

4. Qi, L.: Eigenvalues of a real supersymmetric tensor. J. Symb. Comput. **40**, 1302–1324 (2005)
5. Ding, W., Qi, L., Wei, Y.: $\mathcal{M}$-tensors and nonsingular $\mathcal{M}$-tensors. Linear Algebra Appl. **439**(10), 3264–3278 (2013)
6. Zhang, L., Qi, L., Zhou, G.: $\mathcal{M}$-tensors and some applications. SIAM J. Matrix Anal. Appl. **35**(2), 437–452 (2014)
7. Li, X., Ng, M.: Solving sparse non-negative tensor equations: algorithms and applications. Front. Math. China **10**(3), 649–680 (2015)
8. Ding, W., Wei, Y.: Solving multi-linear systems with $\mathcal{M}$-tensors. J. Sci. Comput. **68**(2), 689–715 (2016)
9. Luo, Z., Qi, L., Xiu, N.: The sparsest solutions to Z-tensor complementarity problems. Optim. Lett. **11**, 471–482 (2017)
10. Azimzadeh, P., Bayraktar, E.: High order Bellman equations and weakly chained diagonally dominant tensors. SIAM J. Matrix Anal. Appl. **40**(1), 276–298 (2019)
11. Han, L.: A homotopy method for solving multilinear systems with M-tensors. Appl. Math. Lett. **69**, 49–54 (2017)
12. Xie, Z., Jin, X., Wei, Y.: Tensor methods for solving symmetric M-tensor systems. J Sci Comput **74**, 412–425 (2018)
13. Li, D., Xie, S., Xu, H.: Splitting methods for tensor equations. Numer. Linear Algebra Appl. **24**(5), e2102 (2017)
14. Cui, L., Li, M., Song, Y.: Preconditioned tensor splitting iterations method for solving multi-linear systems. Appl. Math. Lett. **96**, 89–94 (2019)
15. Liu, D., Li, W., Vong, S.: The tensor splitting with application to solve multi-linear systems. J. Comput. Appl. Math. **330**(1), 75–94 (2018)
16. He, H., Ling, C., Qi, L., Zhou, G.: A globally and quadratically convergent algorithm for solving multilinear systems with $\mathcal{M}$-tensors. J. Sci. Comput. **73**(3), 1718–1741 (2018)
17. Wang, X., Che, M., Wei, Y.: Neural networks based approach solving multi-linear systems with M-tensors. Appl. Math. Lett. **351**, 33–42 (2019)
18. Lv, C., Ma, C.: A Levenberg-Marquardt method for solving semi-symmetric tensor equations. J. Comput. Appl. Math. **332**, 13–25 (2018)
19. Liang, M., Zheng, B., Zhao, R.: Alternating iterative methods for solving tensor equations with applications. Numer. Algor. **80**, 1437–1465 (2019)
20. Liang, M., Zheng, B., Zheng, Y., Zhao, R.: A two-step accelerated Levenberg-Marquardt method for solving multlinear systems in tensor-train format. J. Comput. Appl. Math. **382**, 113069 (2021)
21. Ortega, J., Rheinboldt, W.: Iterative solution of nonlinear equations in several variables. Academic Press, New York (1970)
22. Chen, J., Li, W.: On new exponential quadratically convergent iterative formulae. Appl. Math. Comput. **180**, 242–246 (2006)
23. Chen, J., Li, W.: An exponential regula falsi method for solving nonlinear equations. Numer. Algor. **41**, 327–338 (2006)
24. Kahya, E.: A class of exponential quadratically convergent iterative formulae for unconstrained optimization. Appl. Math. Comput. **186**, 1010–1017 (2007)
25. Smietanski, M.: On a new exponential iterative method for solving nonsmooth equations. Numer. Linear Algebra Appl. **26**(5), e2255 (2019)
26. Levenberg, K.: A method for the solution of certain nonlinear problems in least squares. Quarterly Appl. Math. **2**, 164–168 (1944)
27. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. **11**, 431–441 (1963)
28. Yamashita, N., Fukushima, M.: On the rate of convergence of the Levenberg-Marquardt method. Computing **15**, 239–249 (2001)
29. Zhou, W.: On the convergence of the modified Levenberg-Marquardt method with a nonmonotone second order Armijo type line search. J. Comput. Appl. Math. **239**, 152–161 (2013)
30. Golub, G., Van Loan, C.: Matrix computations, 4th edn. Johns Hopkins University Press, Baltimore (2013)
31. Bader, B., Kolda, T., et al.: Tensor toolbox for MATLAB, Version 3.2. (2021). http://www.tensortoolbox.org
32. Li, T., Wang, Q., Zhang, X.: Gradient based iterative methods for solving symmetric tensor equations. Numer. Linear Algebra Appl. **29**(2), e2414 (2022)

33. Ng, M., Qi, L., Zhou, G.: Finding the largest eigenvalue of a nonnegative tensor. SIAM J. Matrix Anal. Appl. **31**(3), 1090–1099 (2009)
34. Matsuno, Y.: Exact solutions for the nonlinear Klein-Gordon and Liouville equations in four-dimensional Euclidean space. J. Math. Phys. **28**(10), 2317–2322 (1987)
35. Zwillinger, D.: Handbook of differential equations, 3rd edn. Academic Press Inc, Boston (1997)
36. Oseledets, I.: Tensor train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011)