Numerical Algorithms (2019) 80: 1437–1465
https://doi.org/10.1007/s11075-018-0601-4

ORIGINAL PAPER

Alternating iterative methods for solving tensor equations with applications

Maolin Liang[1,2] · Bing Zheng[1] · Ruijuan Zhao[1]

Received: 6 February 2018 / Accepted: 17 September 2018 / Published online: 29 September 2018

Abstract

Recently, the alternating direction method of multipliers (ADMM) and its variations have gained great popularity in large-scale optimization problems. This paper is concerned with the solution of the tensor equation $\mathscr{A}\mathbf{x}^{m-1} = \mathbf{b}$ in which $\mathscr{A}$ is an mth-order and n-dimensional real tensor and $\mathbf{b}$ is an n-dimensional real vector. By introducing certain auxiliary variables, we transform equivalently this tensor equation into a consensus constrained optimization problem, and then propose an ADMM type method for it. It turns out that each limit point of the sequences generated by this method satisfies the Karush-Kuhn-Tucker conditions. Moreover, from the perspective of computational complexity, the proposed method may suffer from the curse-of-dimensionality if the size of the tensor equation is large, and thus we further present a modified version (as a variant of the former) turning to the tensor-train decomposition of the tensor $\mathscr{A}$, which is free from the curse. As applications, we establish the associated inverse iteration methods for solving tensor eigenvalue problems. The performed numerical examples illustrate that our methods are feasible and efficient.

Keywords Tensor equations · ADMM · Tensor-train decomposition · Tensor eigenvalue problems · Inverse iteration methods

Mathematics Subject Classification (2010) 15A69 · 65H10 · 90C30 · 65F15

✉ Bing Zheng
bzheng@lzu.edu.cn

Maolin Liang
liangml2005@163.com

Ruijuan Zhao
zhaobin7755382@163.com

[1] School of Mathematics and Statistics, Lanzhou University, Lanzhou 730000, People's Republic of China

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

1 Introduction

Let $\mathbb{R}$ ($\mathbb{C}$) be the real (complex) field. A tensor of order m and dimension $n_1 \times n_2 \times \cdots \times n_m$ is an m-way array whose entries are addressed via m indices [24, 35]. The set of all mth-order and $n_1 \times n_2 \times \cdots \times n_m$-dimensional tensors over the real field is denoted by $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_m}$. Particularly, if the dimensions of a tensor satisfy $n_1 = n_2 = \cdots = n_m = n$, we call it an mth-order and n-dimensional tensor, and then denote this class of tensors by $\mathbb{R}^{[m,n]}$ for short. For example, a tensor $\mathscr{A} = (a_{i_1 i_2 \ldots i_m}) \in \mathbb{R}^{[m,n]}$, it consists of n^m entries $a_{i_1 i_2 \ldots i_m} \in \mathbb{R}$ with $1 \le i_1, i_2, \ldots, i_m \le n$. Especially, we say that the tensor $\mathscr{A}$ is symmetric if its entries $a_{i_1 i_2 \ldots i_m}$ are invariant under any permutation of their indices $i_1, i_2, \ldots, i_m$.

In this paper, we consider the following tensor equation

$$\mathscr{A}\mathbf{x}^{m-1} = \mathbf{b}, \tag{1.1}$$

where $\mathscr{A} = (a_{i_1 i_2 \ldots i_m})$ is an mth-order and n-dimensional tensor over the real field, $\mathbf{b}$ is a real vector in $\mathbb{R}^n$, and $\mathscr{A}\mathbf{x}^{m-1}$ is a vector in $\mathbb{R}^n$ with entries defined by

$$(\mathscr{A}\mathbf{x}^{m-1})_i = \sum_{i_2, \ldots, i_m} a_{i i_2 \ldots i_m} x_{i_2} \ldots x_{i_m}, \ i = 1, 2, \ldots, n.$$

This equation is sometimes called a multilinear system in literature, since it is linear over $\mathbf{x}$ for each mode of the tensor $\mathscr{A}$, which has practical applications in scientific and engineering fields [35], including data mining [28], numerical partial differential equations [7], and tensor complementarity problems [32].

Specifically, one application of (1.1) is the numerical solution of the following Klein-Gordon equation with Dirichlet's boundary condition [33, 46]

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

where $\triangle = \sum_{k=0}^{d} \frac{\partial^2}{\partial x_k^2}$, $\Omega = [0, 1]^d$ and $m = 3, 4, \ldots$. As stated by Ding and Wei in [7], this equation can be discretized into the following tensor equation

$$\mathscr{L}_h^{(d)} \mathbf{u}^{m-1} = \mathbf{f}, \tag{1.2}$$

in which $h = \frac{1}{n-1}$ and $\mathscr{L}_h^{(d)} = \sum_{k=0}^{d-1} \underbrace{\mathcal{I} \otimes \cdots \otimes \mathcal{I}}_{k} \otimes \mathscr{L}_h \otimes \underbrace{\mathcal{I} \otimes \cdots \otimes \mathcal{I}}_{d-k-1}$ satisfying $(\mathscr{L}_h \mathbf{u}^{m-1})_i = u_i^{m-1} \cdot (\mathbf{L}_h \mathbf{u})_i$ with

$$\mathbf{L}_h = \frac{1}{h^2} \begin{bmatrix} 1 & & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & & 1 \end{bmatrix} \in \mathbb{R}^{n \times n},$$

and $\mathscr{L}_h = ((\mathscr{L}_h)_{i_1 i_2 \ldots i_m}) \in \mathbb{R}^{[m,n]}$ with the entries defined by

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

Particularly, one can refer to [7] for another specific real-life example on a particle's movement under the gravitation.

Recently, some theoretical results and algorithms related to (1.1) have been developed in the sense that $\mathscr{A}$ is a structured tensor. For instance, Ding and Wei [7] proved that the tensor equation (1.1) has only one positive solution when $\mathscr{A}$ is a nonsingular $\mathcal{M}$-tensor and $\mathbf{b}$ is a positive vector (called $\mathcal{M}$-tensor equation for short). They also generalized the classical iterative methods including Jacobi and Gauss-Seidel methods for linear systems to find the unique positive solution. After that, a homotopy method was proposed for searching the positive solution [16]. Furthermore, Liu et al. [31] extended the iterative methods in [7], including the SOR method for linear systems, to the $\mathcal{M}$-tensor equation (1.1) by means of tensor splittings, and proved their linear convergence. Besides, Li et al. [27] also extended the classical iterative methods for linear systems to the $\mathcal{M}$-tensor equation (1.1), which are different from those given in [7, 31]. When the nonsingular $\mathcal{M}$-tensor $\mathscr{A}$ in (1.1) is symmetric, Ding and Wei [7] also established Newton method for it, which is a promising method. Very recently, Xie et al. [43] introduced an algorithm based on the *fast Fourier transform* for the tensor equation (1.1) when the tensor $\mathscr{A}$ is a circulant tensor.

The purpose of this paper is to solve the tensor equation (1.1) under the condition that the tensor $\mathscr{A}$ is a general one. Unfortunately, the aforementioned algorithms are not applicable of solving the class of unstructured tensor equations. On the other hand, those approaches could be intractable if the size of tensor equations is large, because they suffer from the so-called curse-of-dimensionality — the storage and complexity grow exponentially scaling with the dimension of the problems. As a result, it is increasingly desirable to develop certain algorithms that are both rich enough to capture the complexity of data, and scalable enough to process huge variables. We shall propose a class of iterative algorithms sharing such characteristics to address the tensor equation (1.1). As far as we are informed, there are no related works on the general case.

The idea of our approach is based on the well-known *alternating direction method of multipliers* (ADMM) for solving some large-scale optimization problems. This method is firstly proposed by Gabay and Mercier [11] in the 1970s (see Section 2 for details), which and its variants have gained great popularity in modern big data-related problems, especially in image processing, statistical learning, and data mining: see, e.g., the celebrated survey [3] and the references therein. By introducing a series of auxiliary variables, we convert (1.1) into a multi-block optimization problem with consensus constrained conditions (see Section 3 for details), and propose an ADMM type method for it (i.e., Algorithm 3.1). This is a multi-block generalization

of ADMM, and so we denote it by G-ADMM for short. Under some assumptions, it is proved that each limit point of the sequences generated by this method satisfies the Karush-Kuhn-Tucker (KKT) optimality conditions.

Compared with the existing iterative algorithms of solving structured tensor equations, one advantage of G-ADMM is that it is theoretically feasible for any tensor equations, and the other one is that it is easier to be carried out in an efficient and perhaps highly parallel manner. Nevertheless, from the computational complexity point of view, it still suffers from the curse, since the main operations of which lie in the multiplications between the tensor $\mathscr{A}$ and $m-2$ iteration vectors.

To overcome this problem, we apply the tensor-train (TT) format/decomposition of tensors [37] to the tensor-vector multiplications, and then derive a modified version of the G-ADMM method, which is denoted by TT-ADMM for short. It is shown that the amount of calculations corresponding to TT-ADMM is estimated conservatively by $\mathscr{O}((m-1)n^2r^2)$, which is far less than that of the former if m is large, and is free from the curse naturally. The provided numerical examples also illustrate that the methods we propose here are feasible for any tensor equation. Particularly, they are superior to the Newton method [7] and the SOR-type method [31] as long as the involved parameters are selected appropriately.

In addition, as an application of the methods we propose here, two kinds of inverse iteration methods for solving tensor eigenvalue problems (TEP) are given, which are theoretically feasible for any tensor. It is worth mentioning that the inverse iteration stemming from the TT-ADMM method is free from the curse-of-dimensionality as well. Compared with the prevalent shifted symmetric higher-order power method (SS-HOPM) [25] for symmetric tensors, it is shown numerically that our methods have competitive advantages in both the success rate and the computational complexity of finding Z-eigenvalues.

The outline of this paper is as follows. In Section 2, we introduce some notations and definitions, and then review the tensor-train decomposition of tensors and the classical ADMM followed by one variant of which. In Section 3, we propose the alternating iterative methods for solving the tensor equation (1.1), and the convergence analysis of which is given in Section 4. In Section 5, we set up inverse iteration methods for solving tensor eigenvalue problems. In Section 6, we provide some numerical examples to illustrate the effectiveness of our methods. Finally, we conclude this paper with some remarks.

2 Preliminaries

2.1 Notations and definitions

Throughout this paper, scalars are denoted by lower-case letters, e.g., a, b, c; vectors are denoted by boldface lower-case letters, e.g., $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$; matrices are denoted by boldface capital letters, e.g., $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$; tensors are denoted by calligraphic script letters, e.g., $\mathscr{A}$, $\mathscr{B}$, $\mathscr{C}$.

We need the following definitions related to tensors: see, e.g., [24] for details.

Definition 2.1 (The k-mode (vector) product) Let $\mathscr{T} = (t_{i_1 i_2 \ldots i_m}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_m}$ and $\mathbf{v} \in \mathbb{R}^{n_k}$. Then the k-mode (vector) product, denoted by $\mathscr{T} \bullet_k \mathbf{v}$, is an $n_1 \times \cdots \times n_{k-1} \times n_{k+1} \times \cdots \times n_m$ tensor, elementwise,

$$(\mathscr{T} \bullet_k \mathbf{v})_{i_1 \ldots i_{k-1} i_{k+1} \ldots i_m} = \sum_{i_k=1}^{n_k} t_{i_1 \ldots i_k \ldots i_m} v_{i_k}.$$

Definition 2.2 (The k-mode (matrix) product) Let $\mathscr{T} = (t_{i_1 i_2 \ldots i_m}) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_m}$ and $\mathbf{A} = (a_{j_k i_k}) \in \mathbb{R}^{l_k \times n_k}$. Then the k-mode (matrix) product, denoted by $\mathscr{T} \times_k \mathbf{A}$, is an $n_1 \times \cdots \times n_{k-1} \times l_k \times n_{k+1} \times \cdots \times n_m$ tensor, elementwise,

$$(\mathscr{T} \times_k \mathbf{A})_{i_1 \ldots i_{k-1} j_k i_{k+1} \ldots i_m} = \sum_{i_k=1}^{n_k} t_{i_1 \ldots i_k \ldots i_m} a_{j_k i_k}.$$

By Definition 2.1, one can rewrite the symbol $\mathscr{A}\mathbf{x}^{m-1}$ in (1.1) as

$$\mathscr{A}\mathbf{x}^{m-1} = \mathscr{A} \bullet_2 \mathbf{x} \cdots \bullet_m \mathbf{x}.$$

This notation is firstly used when Qi [34] introduced the notion of eigenvalues and eigenvectors of tensors that was also put forward independently by Lim [29] with variational approach.

Definition 2.3 Let $\mathscr{A} \in \mathbb{R}^{[m,n]}$ be a given tensor. Then $(\lambda, \mathbf{x}) \in \mathbb{C} \times (\mathbb{C}^n \setminus \{0\})$ is called an eigenvalue-eigenvector (or simply eigenpair) of the tensor $\mathscr{A}$ if λ and $\mathbf{x}$ satisfy the following homogeneous polynomial equation

$$\mathscr{A}\mathbf{x}^{m-1} = \lambda \mathbf{x}^{[m-1]},$$

where $\mathbf{x}^{[m-1]} \in \mathbb{C}^n$ is the vector defined by

$$\mathbf{x}^{[m-1]} = (x_1^{m-1}, x_2^{m-1}, \ldots, x_n^{m-1})^T.$$

Particularly, we call it an H-eigenpair if they are both real.

Moreover, $(\lambda, \mathbf{x}) \in \mathbb{C} \times (\mathbb{C}^n \setminus \{0\})$ is said to be an E-eigenpair of the tensor $\mathscr{A}$ if λ and $\mathbf{x}$ satisfy the following nonhomogeneous polynomial equations

$$\mathscr{A}\mathbf{x}^{m-1} = \lambda \mathbf{x} \text{ and } \mathbf{x}^T \mathbf{x} = 1.$$

Especially, we call $(\lambda, \mathbf{x})$ a Z-eigenpair if they are both real.

2.2 Low-rank tensor formats

Unlike the matrix case, there are several possibilities to define the rank of a tensor, and thus there are several low-rank tensor formats available [13, 24]. For $\mathscr{A} = (a_{i_1 i_2 \ldots i_m}) \in \mathbb{R}^{[m,n]}$, the CANDECOMP/PARAFAC (CP) format is

$$a_{i_1 i_2 \ldots i_m} = \sum_{k=1}^{r} v_{1,k}(i_1) \cdots v_{m,k}(i_m), \ \mathbf{v}_{l,k} = (v_{l,k}(i_l)) \in \mathbb{R}^n, l = 1, 2, \ldots, m,$$

in which r is the minimal number that the tensor $\mathscr{A}$ can be represented, and is referred to as the CP-rank. This format is data sparse in the sense that storing the factors $\mathbf{v}_{l,k}$ amounts to $\mathscr{O}(mnr)$ storage units, as opposed to the n^m storage units of the full and unstructured tensor $\mathscr{A}$.

In the Tucker format,

$$a_{i_1 i_2 \ldots i_m} = \sum_{k_1}^{l_1} \cdots \sum_{k_m}^{l_m} c_{k_1 k_2 \ldots k_m} v_1(i_1, k_1) \cdots v_m(i_m, k_m),$$

in which $\mathbf{V}_j = (v_j(i_j, k_j)) \in \mathbb{R}^{n \times n}$, $j = 1, 2, \ldots, m$. Since the core tensor $\mathscr{C} = (c_{k_1 k_2 \ldots k_m})$ requires $\prod_{j=1}^{m} l_j$ storage units, this format is limited to small order m, but can be derived by applying the standard matrix approximation techniques to the unfolded tensor.

The low-rank tensor format to be utilized in our approach is the tensor-train (TT) format, which, as a simplified form of the hierarchical scheme, was independently proposed by Hackbusch [15] and later Grasedyck [12] and Oseledets and Tyrtyshnikov [39]. In the TT-format,

$$a_{i_1 i_2 \ldots i_m} = \mathscr{G}_1(i_1)\mathscr{G}_2(i_2) \cdots \mathscr{G}_m(i_m), \tag{2.1}$$

where $\mathscr{G}_k(i_k) \in \mathbb{R}^{r_{k-1} \times r_k}$ is the i_kth lateral slice of the TT-core $\mathscr{G}_k$ with size $r_{k-1} \times n \times r_k$, and r_k $(k = 0, 1, \ldots, m)$ are the tensor-train ranks imposed the "boundary conditions" $r_0 = r_m = 1$.

The TT-format of a tensor can be derived by using the singular value decompositions (SVD) of its matricizations, i.e., the TT-SVD algorithm [37], and the number of operations required by this approach is $O(mnr^3)$ if $r_k \approx r$ with $r = \max_{i=1,2,\ldots,m} r_i$. At this time, the total number of parameters of the TT-format is $O(mnr^2)$. Notably, TT-SVD algorithm returns the quasi-optimal TT-approximation of a tensor. Obviously, this format combines the advantages of both the CP-format and the Tucker format. At present, it has become a powerful tool in scientific computing to address large-scale linear and multilinear algebra problems that would be intractable by using common techniques: see, e.g., [2, 8, 21, 45].

2.3 ADMM

We review the classical ADMM for solving the following minimization problem

$$\begin{cases} \min\limits_{\mathbf{x} \in \mathbb{R}^n, \mathbf{z} \in \mathbb{R}^m} f(\mathbf{x}) + g(\mathbf{z}), \\ \text{s.t.} \quad \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}, \end{cases} \tag{2.2}$$

where $\mathbf{A} \in \mathbb{R}^{l\times n}$, $\mathbf{B} \in \mathbb{R}^{l\times m}$, $\mathbf{c} \in \mathbb{R}^l$, and $f : \mathbb{R}^n \to \mathbb{R}$ is a (smooth) function, and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is a convex (proper) but possibly non-smooth regularization function. Let $\mu > 0$, then the augmented Lagrangian function for (2.2) is defined by

$$L_\mu(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|^2,$$

in which $\mathbf{y} \in \mathbb{R}^l$ is the dual variable. For a chosen initial point $(\mathbf{x}^{(0)}, \mathbf{z}^{(0)}, \mathbf{y}^{(0)})$, ADMM consists of iterating the updates

$$\begin{cases} \mathbf{x}^{(k+1)} = \arg\min\limits_{\mathbf{x}} L_\mu(\mathbf{x}, \mathbf{z}^{(k)}, \mathbf{y}^{(k)}), \\ \mathbf{z}^{(k+1)} = \arg\min\limits_{\mathbf{z}} L_\mu(\mathbf{x}^{(k+1)}, \mathbf{z}, \mathbf{y}^{(k)}), \\ \mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \mu(\mathbf{A}\mathbf{x}^{(k+1)} + \mathbf{B}\mathbf{z}^{(k+1)} - \mathbf{c}). \end{cases}$$

The convergence of the above classical 2-block ADMM was firstly proved by Gabay and Mercier [11] under certain assumptions.

To facilitate the establishment of our methods, we now review a variant of the classical ADMM method for solving the minimization problem involving a separable $f(\mathbf{x}) = \sum\limits_{i=1}^{m} f_i(\mathbf{x})$, namely,

$$\min_{\mathbf{x}\in\mathbb{R}^p} \sum_{i=1}^{m} f_i(\mathbf{x}) + g(\mathbf{x}), \tag{2.3}$$

which covers many important statistical learning problems such as the LASSO problem [41], logistic regression problem [30], and support vector machine problem [17]. In many practical applications, f_i $(i = 1, 2, \ldots m)$ need to be handled by a single agent, such as a thread or a processor. This motivates the following consensus formulation

$$\begin{cases} \min\limits_{\mathbf{x}_i, \mathbf{z}\in\mathbb{R}^p} \sum\limits_{i=1}^{m} f_i(\mathbf{x}_i) + g(\mathbf{z}) \\ \quad \text{s.t. } \mathbf{x}_i = \mathbf{z}, \ i = 1, 2, \ldots, m, \end{cases} \tag{2.4}$$

in which $\mathbf{z}$ is a separate copy of the unknowns, $\mathbf{x}_i$, and $g(\mathbf{z})$ is a regularizer. For example, it can be used to represent a computer network with a star topology [4], in which a master node coordinates the computation of a set of distributed workers (see Fig. 1 for illustration).

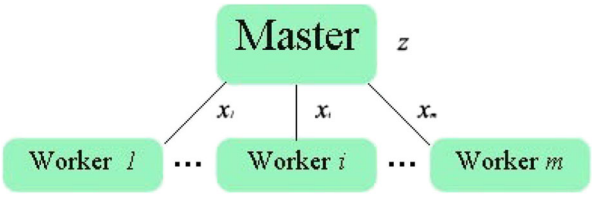

Fig. 1 A star computer cluster with one master and m workers

If set $\mathbf{x} = (\mathbf{x}_1; \ldots; \mathbf{x}_m) \in \mathbb{R}^{pm}$, $\mathbf{A} = \mathbf{I}_{pm} \in \mathbb{R}^{pm \times pm}$, $\mathbf{B} = (\mathbf{I}_p; \ldots; \mathbf{I}_p) \in \mathbb{R}^{pm \times p}$, where $\mathbf{I}_p$ denotes the $p \times p$ identity matrix, then one can observe that the problem (2.4) coincides with the problem (2.2) in the case $\mathbf{c} = \mathbf{0}$. Applying the classical ADMM to (2.4) yields, a multi-block ADMM consists of the iterations:

$$\begin{cases} \mathbf{x}_i^{(k+1)} = \arg\min\limits_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \left(\mathbf{y}_i^{(k)}\right)^T (\mathbf{x}_i - \mathbf{z}^{(k)}) + \dfrac{\mu_i}{2}\|\mathbf{x}_i - \mathbf{z}^{(k)}\|^2, \ i = 1, 2, \ldots, m, \\ \mathbf{z}^{(k+1)} = \arg\min\limits_{\mathbf{z}} \sum\limits_{i=1}^{m} \left(\mathbf{y}_i^{(k)}\right)^T \left(\mathbf{x}_i^{(k+1)} - \mathbf{z}\right) + \sum\limits_{i=1}^{m} \dfrac{\mu_i}{2}\|\mathbf{x}_i^{(k+1)} - \mathbf{z}\|^2, \\ \mathbf{y}_j^{(k+1)} = \mathbf{y}_j^{(k)} + \mu_j \left(\mathbf{x}_j^{(k+1)} - \mathbf{z}^{(k+1)}\right), \ j = 1, 2, \ldots, m, \end{cases} \tag{2.5}$$

in which $\mu_i > 0$ $(i = 1, 2, \ldots, m)$ are penalty parameters, and $\mathbf{y}_j$ $(j = 1, 2, \ldots, m)$ are dual variables.

The above ADMM type method is a multi-block generalization of the classical 2-block ADMM for solving (2.2). The multiple-block iterations often work very well in many cases: see, e.g., [10, 26, 44]; however, the behavior of this class of ADMMs has been largely a mystery [5], especially when the involved functions f_i are nonconvex. Recently, several researchers have made serious attempts in analyzing the convergence of the multi-block case: see, e.g., [19, 20, 22, 40, 42]. In Section 3, we shall transform the tensor equation (1.1) as a consensus constrained optimization problem being similar to (2.4) but involving a nonseparable object.

In addition, the following results are necessary when analyzing the convergence of the algorithms we propose later.

Proposition 2.1 *([36]) Let $D \subseteq \mathbb{R}^n$ be a convex set, and $h : D \longrightarrow \mathbb{R}$ be a differentiable function, then the following expressions are equivalent:*

(a) *h is strongly convex with respect to parameter $\gamma > 0$.*

(b) $h(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq t \cdot h(\boldsymbol{x}) + (1-t)h(\boldsymbol{y}) - \dfrac{\gamma}{2}t(1-t)\|\boldsymbol{x} - \boldsymbol{y}\|^2$ *for any $\boldsymbol{x}, \boldsymbol{y} \in D$ and $t \in [0, 1]$.*

(c) $h(\boldsymbol{x}) - h(\boldsymbol{y}) \geq < \bigtriangledown h(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} > + \dfrac{\gamma}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2$ *holds for any $\boldsymbol{x}, \boldsymbol{y} \in D$.*

(d) $< \bigtriangledown h(\boldsymbol{x}) - \bigtriangledown h(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} > \gamma\|\boldsymbol{x} - \boldsymbol{y}\|^2$ *holds for any $\boldsymbol{x}, \boldsymbol{y} \in D$.*

It is known that if the function h in this proposition is twice continuously differentiable, then it is strongly convex with parameter $\gamma > 0$ if and only if $\bigtriangledown^2 h - \gamma I$ is symmetric positive semi-definite for all $\mathbf{x} \in D$, where $\bigtriangledown^2 h$ is the Hessian matrix. The above analysis indicates that the following result holds true.

Lemma 2.1 *Suppose that $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is full column rank, and $\boldsymbol{b} \in \mathbb{R}^m$, then $\psi(\boldsymbol{x}) = \dfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2$ is strongly convex with $0 < \gamma \leq \lambda_{\min}(\boldsymbol{A}^T\boldsymbol{A})$, where $\lambda_{\min}(\cdot)$ denotes the minimal eigenvalue of a matrix.*

3 Alternating iterative methods for tensor equation (1.1)

In this section, we shall introduce an alternating iterative algorithm (i.e., G-ADMM), following the line of mind of the consensus ADMM (2.5), for approximating the solution of the tensor equation (1.1). After that a modified version of this method will be given by means of the TT-format of the tensor $\mathscr{A}$ (i.e., TT-ADMM). It will be shown that the computational complexity of the later scales linearly with increasing m, that is, it does overcome the curse.

3.1 G-ADMM

We give an equivalent formulation for the tensor equation (1.1). Introducing a set of variables $\{\mathbf{x}_p \in \mathbb{R}^n\}$ with $p = 2, 3, \ldots, m$, then the tensor equation (1.1) can be changed equivalently as

$$\begin{cases} \mathscr{A} \bullet_2 \mathbf{x}_2 \bullet_3 \mathbf{x}_3 \cdots \bullet_m \mathbf{x}_m = \mathbf{b}, \\ \text{s.t. } \mathbf{x}_2 = \mathbf{x}_3 = \cdots = \mathbf{x}_m. \end{cases} \tag{3.1}$$

Then the least-square problem of (1.1) has the following form

$$\begin{cases} \min\limits_{\mathbf{x}_p \in \mathbb{R}^n, p=2,\ldots,m} f(\mathbf{x}_2, \ldots, \mathbf{x}_m) = \dfrac{1}{2} \| \mathscr{A} \bullet_2 \mathbf{x}_2 \bullet_3 \mathbf{x}_3 \cdots \bullet_m \mathbf{x}_m - \mathbf{b} \|^2 \\ \qquad \text{s.t. } (\mathbf{x}_2, \ldots, \mathbf{x}_m) \in \Omega, \end{cases}$$

where $\Omega = \{(\mathbf{x}_2, \ldots, \mathbf{x}_m) \mid \mathbf{x}_2 = \cdots = \mathbf{x}_m, \mathbf{x}_p \in \mathbb{R}^n, \ p = 2, 3, \ldots, m\}$, and it can be rewritten as the consensus constrained optimization problem

$$\begin{cases} \min\limits_{\mathbf{x}_p, \mathbf{z}_p \in \mathbb{R}^n, p=2,\ldots,m} f(\mathbf{x}_2, \ldots, \mathbf{x}_m) + g(\mathbf{z}_2, \ldots, \mathbf{z}_m) \\ \qquad \text{s.t.} \quad \mathbf{x}_p = \mathbf{z}_p, \ \mathbf{x}_p, \mathbf{z}_p \in \mathbb{R}^n, \ p = 2, 3, \ldots, m, \end{cases} \tag{3.2}$$

in which g is the indicator function of Ω, i.e., $g(\mathbf{z}_2, \ldots, \mathbf{z}_m) = 0$ for $(\mathbf{z}_2, \ldots, \mathbf{z}_m) \in \Omega$ and $g(\mathbf{z}_2, \ldots, \mathbf{z}_m) = +\infty$ otherwise.

It is clear that the consensus problem (3.2) coincides with the problem (2.2) if letting $\mathbf{x} = (\mathbf{x}_2; \ldots; \mathbf{x}_m)$, $\widehat{\mathbf{z}} = (\mathbf{z}_2; \ldots; \mathbf{z}_m) \in \mathbb{R}^{n(m-1)}$, $\mathbf{c} = \mathbf{0}$, and $\mathbf{A} = \mathbf{B} = \mathbf{I}_{n(m-1)} \in \mathbb{R}^{n(m-1) \times n(m-1)}$. Furthermore, similar to (2.4), we rewrite (3.2) as

$$\begin{cases} \min\limits_{\mathbf{x}_p, \mathbf{z} \in \mathbb{R}^n, p=2,\ldots,m} f(\mathbf{x}_2, \ldots, \mathbf{x}_m) \\ \qquad \text{s.t.} \quad \mathbf{x}_p = \mathbf{z}, \ \mathbf{x}_p \in \mathbb{R}^n, \ p = 2, 3, \ldots, m, \end{cases} \tag{3.3}$$

in which $\mathbf{z} \in \mathbb{R}^n$ can be viewed as the "central" copy of the unknown variables $\mathbf{x}_p$. Being similar to (2.5), and updating the variables $\mathbf{x}_p$ in a Gauss-Seidel scheme,

we obtain the multi-block ADMM for solving the tensor equation (1.1), denoted by G-ADMM for short, which performs the following updates iteratively:

$$\begin{cases} \mathbf{x}_2^{(k+1)} = \arg\min\limits_{\mathbf{x}_2} L_\mu\left(\mathbf{x}_2, \mathbf{x}_3^{(k)}, \ldots, \mathbf{x}_m^{(k)}, \mathbf{z}^{(k)}, \mathbf{y}_2^{(k)}, \ldots, \mathbf{y}_m^{(k)}\right), \\ \qquad \vdots \\ \mathbf{x}_m^{(k+1)} = \arg\min\limits_{\mathbf{x}_m} L_\mu\left(\mathbf{x}_2^{(k+1)}, \ldots, \mathbf{x}_{m-1}^{(k+1)}, \mathbf{x}_m, \mathbf{z}^{(k)}, \mathbf{y}_2^{(k)}, \ldots, \mathbf{y}_m^{(k)}\right), \\ \mathbf{z}^{(k+1)} = \arg\min\limits_{\mathbf{z}} L_\mu\left(\mathbf{x}_2^{(k+1)}, \ldots, \mathbf{x}_m^{(k+1)}, \mathbf{z}, \mathbf{y}_2^{(k)}, \ldots, \mathbf{y}_m^{(k)}\right), \\ \mathbf{y}_2^{(k+1)} = \mathbf{y}_2^{(k)} + \mu_2\left(\mathbf{x}_2^{(k+1)} - \mathbf{z}^{(k+1)}\right), \\ \qquad \vdots \\ \mathbf{y}_m^{(k+1)} = \mathbf{y}_m^{(k)} + \mu_m\left(\mathbf{x}_m^{(k+1)} - \mathbf{z}^{(k+1)}\right), \end{cases} \tag{3.4}$$

where L_μ is the augmented Lagrangian function, defined by

$$L_\mu(\mathbf{x}_2, \ldots, \mathbf{x}_m, \mathbf{z}, \mathbf{y}_2, \ldots, \mathbf{y}_m) = f(\mathbf{x}_2, \ldots, \mathbf{x}_m) + \sum_{p=2}^{m} \mathbf{y}_p^T(\mathbf{x}_p - \mathbf{z}) + \sum_{p=2}^{m} \frac{\mu_p}{2}\|\mathbf{x}_p - \mathbf{z}\|^2 \tag{3.5}$$

with primal variables $\mathbf{x}_p$ and $\mathbf{z}$, dual variables $\mathbf{y}_p$, and penalty parameters $\mu_p > 0$ for $p = 2, 3, \ldots, m$.

The iteration scheme (3.4) can be written in a slightly different form, which is often more convenient to carry out. As a matter of fact, for each $p \in \{2, 3, \ldots, m\}$, pulling the linear item into the second one, the update $\mathbf{x}_p^{(k+1)}$ becomes

$$\begin{aligned} \mathbf{x}_p^{(k+1)} &= \arg\min_{\mathbf{x}_p}\left(f\left(\mathbf{x}_2^{(k+1)}, \ldots, \mathbf{x}_{p-1}^{(k+1)}, \mathbf{x}_p, \mathbf{x}_{p+1}^{(k)}, \ldots, \mathbf{x}_m^{(k)}\right) + \frac{\mu_p}{2}\|\mathbf{x}_p - \mathbf{z}^{(k)} + \frac{1}{\mu_p}\mathbf{y}_p^{(k)}\|^2\right) \\ &:= \arg\min_{\mathbf{x}_p}\left(\frac{1}{2}\|\mathbf{A}_{\neq p}^k \mathbf{x}_p - \mathbf{b}\|^2 + \frac{\mu_p}{2}\|\mathbf{x}_p - \mathbf{z}^{(k)} + \frac{1}{\mu_p}\mathbf{y}_p^{(k)}\|^2\right), \end{aligned} \tag{3.6}$$

in which

$$\mathbf{A}_{\neq p}^k = \mathscr{A} \bullet_2 \mathbf{x}_2^{(k+1)} \cdots \bullet_{p-1} \mathbf{x}_{p-1}^{(k+1)} \bullet_{p+1} \mathbf{x}_{p+1}^{(k)} \cdots \bullet_m \mathbf{x}_m^{(k)} \in \mathbb{R}^{n\times n}. \tag{3.7}$$

Similarly,

$$\mathbf{z}^{(k+1)} = \arg\min_{\mathbf{z}} \sum_{p=2}^{m} \frac{\mu_p}{2}\|\mathbf{z} - \left(\mathbf{x}_p^{(k+1)} + \frac{1}{\mu_p}\mathbf{y}_p^{(k)}\right)\|^2.$$

By straightforward computation, it holds that

$$\mathbf{z}^{(k+1)} = \sum_{p=2}^{m}\left(\mu_p \mathbf{x}_p^{(k+1)} + \mathbf{y}_p^{(k)}\right) / \sum_{p=2}^{m} \mu_p. \tag{3.8}$$

Denote

$$\overline{\mathbf{x}}^{(k+1)} = \sum_{p=2}^{m} \mu_p \mathbf{x}_p^{(k+1)} / \sum_{p=2}^{m} \mu_p, \quad \overline{\mathbf{y}}^{(k)} = \sum_{p=2}^{m} \mathbf{y}_p^{(k)} / \sum_{p=2}^{m} \mu_p, \tag{3.9}$$

then (3.8) becomes

$$\mathbf{z}^{(k+1)} = \overline{\mathbf{x}}^{(k+1)} + \overline{\mathbf{y}}^{(k)}.$$

On the other hand, the dual updates in (3.4) follow that

$$\overline{\mathbf{y}}^{(k+1)} = \overline{\mathbf{y}}^{(k)} + \overline{\mathbf{x}}^{(k+1)} - \mathbf{z}^{(k+1)}.$$

The last two equalities indicate that $\overline{\mathbf{y}}^{(k+1)} = 0$, then we have, for $k \geq 1$,

$$\mathbf{z}^{(k+1)} = \overline{\mathbf{x}}^{(k+1)}. \tag{3.10}$$

Setting $\mathbf{u}_p^{(k)} = \mathbf{y}_p^{(k)}/\mu_p$, and connecting with (3.6), (3.9), and (3.10), we obtain a scaled version of (3.4), which is summarized in the following algorithm.

Algorithm 3.1 (G-ADMM)

STEP 1: Input $\mathscr{A} \in \mathbb{R}^{[m,n]}$, $\mathbf{b} \in \mathbb{R}^n$, and initial values $\mathbf{x}_p^{(0)}, \mathbf{u}_p^{(0)} \in \mathbb{R}^n$, $\mu_p > 0$ for $p = 2, 3, \ldots, m$. Then compute $\overline{\mathbf{x}}^{(0)}$ by (3.9).

STEP 2: For $p = 2, 3, \ldots, m$

Compute the matrix $\mathbf{A}_{\neq p}^k$ given in (3.7), and then update

$$\mathbf{x}_p^{(k+1)} = \arg\min_{\mathbf{x}_p} \left(\frac{1}{2} \|\mathbf{A}_{\neq p}^k \mathbf{x}_p - \mathbf{b}\|^2 + \frac{\mu_p}{2} \|\mathbf{x}_p - \overline{\mathbf{x}}^{(k)} + \mathbf{u}_p^{(k)}\|^2 \right). \tag{3.11}$$

STEP 3: Compute

$$\overline{\mathbf{x}}^{(k+1)} = \sum_{p=2}^{m} \mu_p \mathbf{x}_p^{(k+1)} / \sum_{p=2}^{m} \mu_p. \tag{3.12}$$

STEP 4: For $p = 2, 3, \ldots, m$, update

$$\mathbf{u}_p^{(k+1)} = \mathbf{u}_p^{(k)} + \mathbf{x}_p^{(k+1)} - \overline{\mathbf{x}}^{(k+1)}. \tag{3.13}$$

STEP 5: If the termination criterion is satisfied, return $\overline{\mathbf{x}}^{(k+1)}$.
Otherwise, let $k = k + 1$, and go to Step 2.

For this algorithm, we have the following comments:

First of all, the primal update $\mathbf{x}_p^{(k+1)}$ in (3.11) is unique. In fact, (3.11) can be reformulated equivalently as

$$\mathbf{x}_p^{(k+1)} = \arg\min_{\mathbf{x}_p} \frac{1}{2} \left\| \begin{bmatrix} \mathbf{A}_{\neq p}^k \\ \sqrt{\mu_p}\, \mathbf{I}_n \end{bmatrix} \mathbf{x}_p - \begin{bmatrix} \mathbf{b} \\ \sqrt{\mu_p}\, (\overline{\mathbf{x}}^{(k)} - \mathbf{u}_p^{(k)}) \end{bmatrix} \right\|^2.$$

Obviously, the coefficient matrix of the above least-squares subproblem is full column rank, thus the solution of which is unique, and can be expressed as

$$\mathbf{x}_p^{(k+1)} = \left((\mathbf{A}_{\neq p}^k)^T \mathbf{A}_{\neq p}^k + \mu_p\, \mathbf{I}_n \right)^{-1} \left((\mathbf{A}_{\neq p}^k)^T \mathbf{b} + \mu_p\, (\overline{\mathbf{x}}^{(k)} - \mathbf{u}_p^{(k)}) \right). \tag{3.14}$$

Notably, the primal updates $\mathbf{x}_p^{(k+1)}$ $(p = 2, 3, \ldots, m)$ in Step 2 could be carried out in parallel manner if all $\mathbf{x}_p^{(k+1)}$ are derived by using the kth iterative values only, thus our method is easier to parallelize than the existing ones.

Secondly, if all the penalty parameters μ_p in (3.12) are identical, then $\overline{\mathbf{x}}^{(k+1)}$ reduces to

$$\overline{\mathbf{x}}^{(k+1)} = \frac{1}{m-1}\sum_{p=2}^{m}\mathbf{x}_p^{(k+1)},$$

the average value of the primal updates $\mathbf{x}_p^{(k+1)}$ for $p = 2, 3, \ldots, m$. In our numerical experiments, we shall choose these parameters as a constant for simplicity. The stopping rule of Algorithm 3.1 is to be discussed in Section 4.

In addition, notice that the main operations of Algorithm 3.1 hinge on the reformulation of the matrices $\mathbf{A}_{\neq p}^{k}$ with $p = 2, 3, \ldots, m$, which means that its computational complexity is about $\mathcal{O}((m-1)n^{m-2})$, suffering from the curse-of-dimensionality. In next subsection, a modified version of G-ADMM will be given by employing the TT-format of the tensor $\mathscr{A}$, which is immune to the curse.

3.2 TT-ADMM

In this subsection, we consider to alleviate the curse of the G-ADMM method for solving the tensor equation (1.1). Since the main operations of Algorithm 3.1 hinge on the computations of the matrices $\mathbf{A}_{\neq p}^{k} \in \mathbb{R}^{n\times n}$ with $p = 2, 3, \ldots, m$, it suffices to change the treatment way to compute them by means of the TT-format of the tensor $\mathscr{A}$.

Let the tensor $\mathscr{A}$ in (1.1) be characterized by the TT-cores $\mathscr{G}_i$ with $i = 1, 2, \ldots, m$, i.e., (2.1). From the definition of tensor-vector multiplication we obtain that, for $i_p \in \{1, 2, \ldots, n\}$,

$$\mathbf{A}_{\neq p}^{k}(:, i_p) = \mathscr{G}_1\left(\mathscr{G}_2 \bullet_2 \mathbf{x}_2^{(k+1)}\right)\cdots\left(\mathscr{G}_{p-1} \bullet_2 \mathbf{x}_{p-1}^{(k+1)}\right)\mathscr{G}_p(i_p)\left(\mathscr{G}_{p+1} \bullet_2 \mathbf{x}_{p+1}^{(k)}\right)\cdots\left(\mathscr{G}_m \bullet_2 \mathbf{x}_m^{(k)}\right). \tag{3.15}$$

Denote

$$\mathbf{A}_{<p}^{k} = \begin{cases} \mathscr{G}_1, & \text{if } p = 2, \\ \mathscr{G}_1\left(\mathscr{G}_2 \bullet_2 \mathbf{x}_2^{(k+1)}\right)\cdots\left(\mathscr{G}_{p-1} \bullet_2 \mathbf{x}_{p-1}^{(k+1)}\right) \in \mathbb{R}^{n\times r_{p-1}}, & \text{if } 2 < p \le m, \end{cases}$$

and

$$\mathbf{A}_{>p}^{k} = \begin{cases} \left(\mathscr{G}_{p+1} \bullet_2 \mathbf{x}_{p+1}^{(k)}\right)\cdots\left(\mathscr{G}_m \bullet_2 \mathbf{x}_m^{(k)}\right) \in \mathbb{R}^{r_p\times 1}, & \text{if } 2 \le p < m, \\ \mathbf{I}_n, & \text{if } p = m, \end{cases}$$

then the matrix $\mathbf{A}_{\neq p}^{k}$ defined in (3.15) can be represented as

$$\mathbf{A}_{\neq p}^{k} = \mathscr{G}_p \times_1 \mathbf{A}_{<p}^{k} \times_3 (\mathbf{A}_{>p}^{k})^T. \tag{3.16}$$

Combining with this formula and Algorithm 3.1, we derive the TT-based multi-block alternating direction method of multipliers for solving the tensor equation (1.1), denoted by TT-ADMM for short, which is stated as follows.

Algorithm 3.2 (TT-ADMM)

STEP 1: Input $\mathscr{A} \in \mathbb{R}^{[m,n]}$ characterized by the TT-cores $\mathscr{G}_i$ with $i = 1, 2, \ldots, m$, and $\mathbf{b} \in \mathbb{R}^n$. Let the initial values be $\mathbf{x}_p^{(0)}, \mathbf{u}_p^{(0)} \in \mathbb{R}^n$, $\mu_p > 0$, $p = 2, 3, \ldots, m$. Then compute $\overline{\mathbf{x}}^{(0)}$ by (3.9).

STEP 2: For $p = 2, 3, \ldots, m$, compute $\mathbf{A}_{\neq p}^k$ by (3.16), and then update $\mathbf{x}_p^{(k+1)}$ by (3.11).

STEP 3: Compute $\overline{\mathbf{x}}^{(k+1)}$ according to (3.12).

STEP 4: For $p = 2, 3, \ldots, m$, update $\mathbf{u}_p^{(k+1)}$ by (3.13).

STEP 5: If the termination criterion is satisfied, return $\overline{\mathbf{x}}^{(k+1)}$. Otherwise, let $k = k + 1$, and go to Step 2.

In what follows, we investigate the computational complexity of Algorithm 3.2. Actually, if we compute all the matrices $\mathbf{A}_{\neq p}^k$ by (3.16), the details of their computational amounts are listed in Table 1, in which we let $r = \max\limits_{1 \le i \le m} r_i$ and suppose $r \le n$. In this case, the amount of operations contained in this algorithm is estimated conservatively by $\mathcal{O}((m-1)n^2r^2)$, and is obviously far less than that of Algorithm 3.1. Therefore, the TT-ADMM algorithm is a promising approach for solving large-scale tensor equations because of its lower computational complexity.

In addition, to guarantee the lower complexity of Algorithm 3.2, the TT-format of the tensor $\mathscr{A}$ should also be employed if some other tensor-vector operations (e.g., the residual of the tensor equation (1.1)) are desired.

Remark 3.1 If the tensor $\mathscr{A}$ is given in CP-format, then the TT-format of which can be exactly obtained [37]. However, for a full tensor, it is not easy to obtain the CP-format of a tensor because it is NP-hard to determine the CP-rank [18]. In contrast, one can use the TT-SVD algorithm to approximate a tensor into the TT-format quasi-optimally [37]. Compared with Algorithm 3.1, it will be shown numerically that the TT-ADMM method is feasible and effective if the error caused by the TT-approximation is small enough.

Table 1 The detail of the amount of computations in Algorithm 3.2

Operations	Amount of calculations	Index p
$\mathbf{B}_i := \mathscr{G}_i \bullet_2 \mathbf{x}_i^{(k+1)}$, $i = 2, 3, \ldots, p-1$	$2nr^2 \times (p-2)$	
$\mathbf{C}_j := \mathscr{G}_j \bullet_2 \mathbf{x}_j^{(k)}$, $j = p+1, \ldots, m$	$2nr^2 \times (m-p-1) + 2nr$	
$\mathbf{A}_1 = \mathscr{G}_1\mathbf{B}_2 \cdots \mathbf{B}_{p-1}$	$2r^3 \times (p-3) + 2nr$	
$\mathbf{A}_2 = (\mathbf{C}_{p+1} \cdots (\mathbf{C}_{m-1}\mathbf{C}_m))$	$2r^3 \times (m-p-2)$	
$\mathbf{A}_{\neq p}^k = \mathscr{G}_p \times_1 \mathbf{A}_1 \times_3 \mathbf{A}_2{}^T$	$2n^2r^2 + 2nr^2$	$p \in \{2, 3, \ldots, m-1\}$
$\mathbf{B}_i := \mathscr{G}_i \bullet_2 \mathbf{x}_i^{(k+1)}$, $i = p, \ldots, m-1$	$2nr^2 \times (m-p)$	
$\mathbf{A}_{\neq m}^k = \mathscr{G}_1\mathbf{B}_2 \cdots \mathbf{B}_{m-1}$	$2r^3 \times (m-3) + 2nr$	$p = m$
$\mathbf{x}_p^{(k+1)}$-update by (3.14)	$2n^2 + 3n + 1$	$p \in \{2, 3, \ldots, m\}$
$\overline{\mathbf{x}}^{(k+1)}$-update by (3.12)	mn	

4 Convergence analysis

Although the alternating iterative methods we propose in preceding section are also the multi-block generalizations of the classical ADMM, but they are very different from the multi-block ADMM (2.5), since the objective function in our problem (3.3) is nonseparable, which means that the convergence analysis of which may be more difficult than that of the later. In this section, we shall give the convergence analysis of Algorithm 3.1 under some hypotheses, which is similar but different from that presented in [44], since the problems we considered here and studied in [44] are different. By the same arguments, we can discuss the convergence analysis of Algorithm 3.2, so it is omitted here.

For ease of expression, we, in this section, use the iteration scheme (3.4) to analyze the convergence, and specially introduce the following notations:

$$\begin{aligned}
\widetilde{\mathbf{x}} &:=(\widetilde{\mathbf{x}}_2,\ldots,\widetilde{\mathbf{x}}_m),\ \widetilde{\mathbf{y}} := (\widetilde{\mathbf{y}}_2,\ldots,\widetilde{\mathbf{y}}_m),\\
\widehat{\mathbf{x}} &:=(\mathbf{x}_2,\ldots,\mathbf{x}_m),\ \widehat{\mathbf{y}} := (\mathbf{y}_2,\ldots,\mathbf{y}_m),\\
\widehat{\mathbf{x}}^{(k+1,p)} &:= \left(\mathbf{x}_2^{(k+1)},\ldots,\mathbf{x}_p^{(k+1)},\mathbf{x}_{p+1}^{(k)}\ldots\mathbf{x}_m^{(k)}\right),\ p=2,\ldots,m-1,\\
\widehat{\mathbf{y}}^{(k+1,p)} &:= \left(\mathbf{y}_2^{(k+1)},\ldots,\mathbf{y}_p^{(k+1)},\mathbf{y}_{p+1}^{(k)}\ldots\mathbf{y}_m^{(k)}\right),\ p=2,\ldots,m-1,\\
\widehat{\mathbf{x}}^{(k+1)} &:=\widehat{\mathbf{x}}^{(k+1,m)} = \left(\mathbf{x}_2^{(k+1)},\ldots,\mathbf{x}_m^{(k+1)}\right),\\
\widehat{\mathbf{y}}^{(k+1)} &:=\widehat{\mathbf{y}}^{(k+1,m)} = \left(\mathbf{y}_2^{(k+1)},\ldots,\mathbf{y}_m^{(k+1)}\right).
\end{aligned}$$

Let us recall the augmented Lagrangian function (3.5)

$$L_\mu(\widehat{\mathbf{x}},\mathbf{z},\widehat{\mathbf{y}}) = f(\widehat{\mathbf{x}}) + \sum_{p=2}^{m}\frac{\mu_p}{2}\left(\|\mathbf{x}_p-\mathbf{z}+\frac{1}{\mu_p}\mathbf{y}_p\|^2-\frac{1}{\mu_p^2}\|\mathbf{y}_p\|^2\right),$$

and the Karush-Kuhn-Tucker (KKT) optimality conditions for the problem (3.3):

$$\begin{cases}
\mathbf{A}_{\neq p}^T(\mathbf{A}_{\neq p}\mathbf{x}_p-\mathbf{b})+\mathbf{y}_p=0,\\
\sum_{p=2}^{m}\mathbf{y}_p=0,\quad p=2,3,\ldots,m,\\
\mathbf{x}_p-\mathbf{z}=0.
\end{cases}\tag{4.1}$$

Then we have the following theorem.

Theorem 4.1 *Let* $\{\widehat{\boldsymbol{x}}^{(k)},\boldsymbol{z}^{(k)},\widehat{\boldsymbol{y}}^{(k)}\}$ *be a sequence generated by Algorithm 3.1, and assume that* $\sum_{k=0}^{\infty}\sum_{p=2}^{m}\frac{1}{\mu_p}\|\boldsymbol{y}_p^{(k+1)}-\boldsymbol{y}_p^{(k)}\|^2<\infty$. *Then* $(\widehat{\boldsymbol{x}}^{(k+1)},\boldsymbol{z}^{(k+1)})-(\widehat{\boldsymbol{x}}^{(k)},\boldsymbol{z}^{(k)})\to 0$ as $k\to\infty$.

Proof For each $p \in \{2, 3, \ldots, m\}$, we rewrite the augmented Lagrangian function as

$$\begin{aligned} L_\mu(\widehat{\mathbf{x}}, \mathbf{z}, \widehat{\mathbf{y}}) &= \frac{1}{2}\|\mathbf{A}_{\neq p}\mathbf{x}_p - \mathbf{b}\|^2 + \frac{\mu_p}{2}\|\mathbf{x}_p - \mathbf{z} + \frac{1}{\mu_p}\mathbf{y}_p\|^2 + c_{\neq p} \\ &= \frac{1}{2}\left\| \begin{bmatrix} \mathbf{A}_{\neq p} \\ \sqrt{\mu_p}\,\mathbf{I}_n \end{bmatrix} \mathbf{x}_p - \begin{bmatrix} \mathbf{b} \\ \sqrt{\mu_p}\,\mathbf{z} - \dfrac{1}{\sqrt{\mu_p}}\mathbf{y}_p \end{bmatrix} \right\|^2 + c_{\neq p}, \end{aligned}$$

where $c_{\neq p}$ is the term being independent with $\mathbf{x}_p$, which, together with Lemma 2.1, indicates that L_μ over $\mathbf{x}_p$ is strongly convex regarding $\alpha_p = \lambda_{\min}(\mathbf{A}_{\neq p}^T\mathbf{A}_{\neq p}) + \mu_p$, namely,

$$\begin{aligned} &L_\mu(\mathbf{x}_2, \ldots, \mathbf{x}_p', \ldots, \mathbf{x}_m, \mathbf{z}, \widehat{\mathbf{y}}) - L_\mu(\mathbf{x}_2, \ldots, \mathbf{x}_p, \ldots, \mathbf{x}_m, \mathbf{z}, \widehat{\mathbf{y}}) \\ &\quad \geq < \frac{\partial}{\partial \mathbf{x}_p} L_\mu(\mathbf{x}_2, \ldots, \mathbf{x}_p, \ldots, \mathbf{x}_m, \mathbf{z}, \widehat{\mathbf{y}}), \mathbf{x}_p' - \mathbf{x}_p > + \frac{\alpha_p}{2}\|\mathbf{x}_p' - \mathbf{x}_p\|^2. \end{aligned}$$

Notice that $\mathbf{x}_p^{(k+1)}$ is the minimizer of L_μ over the variable $\mathbf{x}_p$ in (3.6), then the last inequality implies that

$$\begin{aligned} L_\mu(\widehat{\mathbf{x}}^{(k)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) - L_\mu(\widehat{\mathbf{x}}^{(k+1,2)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) &\geq \frac{\alpha_2'}{2}\|\mathbf{x}_2^{(k)} - \mathbf{x}_2^{(k+1)}\|^2, \\ L_\mu(\widehat{\mathbf{x}}^{(k+1,2)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) - L_\mu(\widehat{\mathbf{x}}^{(k+1,3)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) &\geq \frac{\alpha_3'}{2}\|\mathbf{x}_3^{(k)} - \mathbf{x}_3^{(k+1)}\|^2, \\ &\vdots \\ L_\mu(\widehat{\mathbf{x}}^{(k+1,m-1)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) - L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) &\geq \frac{\alpha_m'}{2}\|\mathbf{x}_m^{(k)} - \mathbf{x}_m^{(k+1)}\|^2, \end{aligned}$$

where $\alpha_p' = \lambda_{\min}((\mathbf{A}_{\neq p}^k)^T\mathbf{A}_{\neq p}^k) + \mu_p$ for $p = 2, 3, \ldots, m$. Summing these inequalities yields

$$L_\mu(\widehat{\mathbf{x}}^{(k)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) - L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) \geq \sum_{p=2}^{m} \frac{\alpha_p'}{2}\|\mathbf{x}_p^{(k)} - \mathbf{x}_p^{(k+1)}\|^2. \tag{4.2}$$

Furthermore, after rearrangement, we obtain another form of the augmented Lagrangian function L_μ as follows:

$$\begin{aligned} L_\mu(\widehat{\mathbf{x}}, \mathbf{z}, \widehat{\mathbf{y}}) &= \sum_{p=2}^{m} \frac{\mu_p}{2}\|\mathbf{x}_p - \mathbf{z} + \frac{1}{\mu_p}\mathbf{y}_p\|^2 + f(\widehat{\mathbf{x}}) - \frac{1}{2}\sum_{p=2}^{m}\frac{1}{\mu_p}\|\mathbf{y}_p\|^2 \\ &= \frac{1}{2}\left\| \begin{bmatrix} \sqrt{\mu_2}\,\mathbf{I}_n & & \\ & \ddots & \\ & & \sqrt{\mu_m}\,\mathbf{I}_n \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \vdots \\ \mathbf{z} \end{bmatrix} - \begin{bmatrix} \sqrt{\mu_2}\,\mathbf{x}_2 + \dfrac{1}{\sqrt{\mu_2}}\mathbf{y}_2 \\ \vdots \\ \sqrt{\mu_m}\,\mathbf{x}_m + \dfrac{1}{\sqrt{\mu_m}}\mathbf{y}_m \end{bmatrix} \right\|^2 \\ &\quad + f(\widehat{\mathbf{x}}) - \frac{1}{2}\sum_{p=2}^{m}\frac{1}{\mu_p}\|\mathbf{y}_p\|^2. \end{aligned}$$

Using Lemma 2.1 once more, we therefore conclude that L_μ over the variable $\mathbf{z}$ is strongly convex with respect to $\beta = \min_{2\le p\le m} \mu_p$, then

$$L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) - L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k)}) \ge \frac{\beta}{2}\|\mathbf{z}^{(k)} - \mathbf{z}^{(k+1)}\|^2. \tag{4.3}$$

In addition, from the definition of L_μ, we obtain

$$\begin{aligned} &L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k)}) - L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k,2)}) \\ &= \mathbf{y}_2^{(k)^T}\left(\mathbf{x}_2^{(k+1)} - \mathbf{z}^{(k+1)}\right) - \mathbf{y}_2^{(k+1)^T}\left(\mathbf{x}_2^{(k+1)} - \mathbf{z}^{(k+1)}\right) \\ &= -\frac{1}{\mu_2}\|\mathbf{y}_2^{(k)} - \mathbf{y}_2^{(k+1)}\|^2. \end{aligned} \tag{4.4}$$

In the same treatment as utilized in (4.4), we have

$$\begin{aligned} L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k,2)}) - L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k,3)}) &= -\frac{1}{\mu_3}\|\mathbf{y}_3^{(k)} - \mathbf{y}_3^{(k+1)}\|^2, \\ &\vdots \\ L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k,m-1)}) - L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k+1)}) &= -\frac{1}{\mu_m}\|\mathbf{y}_m^{(k)} - \mathbf{y}_m^{(k+1)}\|^2. \end{aligned} \tag{4.5}$$

Summing the equalities in (4.4) and (4.5) gives

$$L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k)}) - L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k+1)}) = -\sum_{q=2}^{m}\frac{1}{\mu_q}\|\mathbf{y}_q^{(k+1)} - \mathbf{y}_q^{(k)}\|^2,$$

which, together with the inequalities (4.2) and (4.3), derives that

$$\begin{aligned} &L_\mu(\widehat{\mathbf{x}}^{(k)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) - L_\mu(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k+1)}) \\ &\ge \sum_{p=2}^{m}\frac{\alpha_p'}{2}\|\mathbf{x}_p^{(k+1)} - \mathbf{x}_p^{(k)}\|^2 + \frac{\beta}{2}\|\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\|^2 - \sum_{q=2}^{m}\frac{1}{\mu_q}\|\mathbf{y}_q^{(k+1)} - \mathbf{y}_q^{(k)}\|^2. \end{aligned} \tag{4.6}$$

This formula is essential for completing the proof. As a matter of fact, according to the definition of L_μ, we know that $L_\mu(\widehat{\mathbf{x}}^{(k)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)})$ is bounded below for any positive integer k. Taking summation of the inequalities in (4.6) from $k = 0$ to ∞, we obtain

$$\sum_{k=0}^{\infty}\left(\sum_{p=2}^{m}\frac{\alpha_p'}{2}\|\mathbf{x}_p^{(k+1)} - \mathbf{x}_p^{(k)}\|^2 + \frac{\beta}{2}\|\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\|^2 - \sum_{p=2}^{m}\frac{1}{\mu_q}\|\mathbf{y}_q^{(k+1)} - \mathbf{y}_q^{(k)}\|^2\right) < \infty,$$

which implies from the assumption that

$$\sum_{k=0}^{\infty}\left(\sum_{p=2}^{m}\frac{\alpha_p'}{2}\|\mathbf{x}_p^{(k+1)} - \mathbf{x}_p^{(k)}\|^2 + \frac{\beta}{2}\|\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\|^2\right) < \infty,$$

and then $(\widehat{\mathbf{x}}^{(k+1)}, \mathbf{z}^{(k+1)}, \widehat{\mathbf{y}}^{(k+1)}) - (\widehat{\mathbf{x}}^{(k)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}) \to 0$ as $k \to \infty$. □

By Theorem 4.1, we obtain the following result.

Theorem 4.2 *Let $\{\widehat{\boldsymbol{x}}^{(k)}, \boldsymbol{z}^{(k)}, \widehat{\boldsymbol{y}}^{(k)}\}$ be a sequence generated by Algorithm 3.1. Then any limit point of the above sequence satisfies the KKT equations in (4.1).*

Proof Because $\mathbf{x}_p^{(k+1)}$ is a minimizer of the subproblem (3.6), then we have

$$\begin{aligned}
&\left(\left(\mathbf{A}_{\neq p}^k\right)^T \mathbf{A}_{\neq p}^k + \mu_p \mathbf{I}_n\right)\left(\mathbf{x}_p^{(k+1)} - \mathbf{x}_p^{(k)}\right) \\
&= \left(\mathbf{A}_{\neq p}^k\right)^T \mathbf{b} + \mu_p \mathbf{x}_p^{(k)} - \mathbf{y}_p^{(k)} - \left(\left(\mathbf{A}_{\neq p}^k\right)^T \mathbf{A}_{\neq p}^k + \mu_p \mathbf{I}_n\right)\mathbf{x}_p^{(k)} \qquad (4.7)\\
&= \left(\mathbf{A}_{\neq p}^k\right)^T \left(\mathbf{b} - \mathbf{A}_{\neq p}^k \mathbf{x}_p^{(k)}\right) - \mathbf{y}_p^{(k)}.
\end{aligned}$$

Moreover, it follows from (3.8) that

$$\begin{aligned}
\sum_{p=2}^m \mu_p(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}) &= \sum_{p=2}^m \left(\mathbf{y}_p^{(k)} + \mu_p \mathbf{x}_p^{(k+1)}\right) - \sum_{p=2}^m \mu_p \mathbf{z}^{(k)} \\
&= \sum_{p=2}^m \mathbf{y}_p^{(k)} + \sum_{p=2}^m \mu_p \left(\mathbf{x}_p^{(k+1)} - \mathbf{x}_p^{(k)}\right) + \sum_{p=2}^m \mu_p \left(\mathbf{x}_p^{(k)} - \mathbf{z}^{(k)}\right).
\end{aligned} \qquad (4.8)$$

Rewriting the dual updates in (3.4) as

$$\mathbf{y}_p^{(k+1)} - \mathbf{y}_p^{(k)} = \mu_p \left(\mathbf{x}_p^{(k+1)} - \mathbf{z}^{(k+1)}\right). \qquad (4.9)$$

Then we conclude from Theorem 4.1 that all the right-hand sides in (4.7)–(4.9) tend to zero as $k \to \infty$, i.e.,

$$\begin{cases}
\left(\mathbf{A}_{\neq p}^k\right)^T \left(\mathbf{b} - \mathbf{A}_{\neq p}^k \mathbf{x}_p^{(k)}\right) - \mathbf{y}_p^{(k)} \to 0, \\
\qquad\qquad \sum_{p=2}^m \mathbf{y}_p^{(k)} \to 0, \quad p = 2, 3, \ldots, m, \\
\mathbf{x}_p^{(k+1)} - \mathbf{z}^{(k+1)} \to 0.
\end{cases} \qquad (4.10)$$

On the other hand, for any limit point $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{z}})$ of the sequence $\{\widehat{\mathbf{x}}^{(k)}, \mathbf{z}^{(k)}\}$, that is, there exists a subsequence $\{\widehat{\mathbf{x}}^{(k_j)}, \mathbf{z}^{(k_j)}\}$ such that $\{\widehat{\mathbf{x}}^{(k_j)}, \mathbf{z}^{(k_j)}\} \to (\widetilde{\mathbf{x}}, \widetilde{\mathbf{z}})$ as $j \to \infty$. Moreover, the boundedness of $\{\widehat{\mathbf{y}}^{(k)}\}$ guarantees the existence of a subsequence $\{\widehat{\mathbf{y}}^{(k_j)}\}$ converging to some point $\widetilde{\mathbf{y}}$ as $j \to \infty$. Then it follows that $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{z}}, \widetilde{\mathbf{y}})$ is a limit point of the sequence $\{\widehat{\mathbf{x}}^{(k)}, \mathbf{z}^{(k)}, \widehat{\mathbf{y}}^{(k)}\}$. Replacing k by k_j in (4.10), which means that $\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{z}}$, and $\widetilde{\mathbf{y}}$ constitute a solution pair of the system of (4.1). □

Following this theorem, we can derive the termination criterion of the algorithms proposed in previous section.

Remark 4.1 The convergence analysis suggests that a feasibility termination criterion of Algorithms 3.1 and 3.2 is that both the norm of the primal residual $\mathbf{r}^{(k+1)} = \left(\mathbf{x}_2^{(k+1)} - \overline{\mathbf{x}}^{(k+1)}, \ldots, \mathbf{x}_m^{(k+1)} - \overline{\mathbf{x}}^{(k+1)}\right)$, and the dual residual $\mathbf{s}^{(k+1)} = \overline{\mathbf{x}}^{(k+1)} - \overline{\mathbf{x}}^{(k)}$

are small enough. At this time, $\overline{\mathbf{x}}^{(k+1)}$ is viewed as an approximate solution of the tensor equation (1.1). Certainly, one can also use the residual of the tested tensor equations to terminate the iteration if they are solvable.

5 An application

In this section, we apply the alternating iterative methods given in Section 3 to the solution of tensor eigenvalue problems by establishing the associated inverse iteration methods as in [7].

Here, we consider only the Z-eigenvalue problem of tensors, and the others can be addressed in the similar manner. The inverse iteration method for searching the Z-eigenpairs of a tensor $\mathscr{A} \in \mathbb{R}^{[m,n]}$ is described as follows, which is twisted from Algorithm 3.1 and is denoted by INV-GADMM for short.

Algorithm 5.1 (INV-GADMM)

STEP 1: Input $\mathscr{A} \in \mathbb{R}^{[m,n]}$, and initial unit vector $\mathbf{x}^{(0)}$.

STEP 2: While the terminate criterion is not satisfied, for $k = 1, 2, \ldots$

1) Solve tensor equation $\mathscr{A}(\mathbf{y}^{(k)})^{m-1} = \lambda_{k-1}\mathbf{x}^{(k-1)}$ by Algorithm 3.1;
2) Compute $\mathbf{x}^{(k)} = \mathbf{y}^{(k)}/\|\mathbf{y}^{(k)}\|$;
3) Compute $\lambda_k = \mathscr{A}(\mathbf{x}^{(k)})^m$.

STEP 3: Return $(\lambda_k, \mathbf{x}^{(k)})$ as an approximate Z-eigenpair of $\mathscr{A}$.

Furthermore, depending on Algorithm 3.2, the inverse iteration method for solving the Z-eigenvalue problem of tensors is given below, and is denoted by INV-TTADMM for short.

Algorithm 5.2 (INV-TTADMM)

STEP 1: Input $\mathscr{A} \in \mathbb{R}^{[m,n]}$ characterized by the TT-cores $\mathscr{G}_i$ with $i = 1, 2, \ldots, m$. Let the initial unit vector be $\mathbf{x}^{(0)} \in \mathbb{R}^n$.

STEP 2: While the terminate criterion is not satisfied, for $k = 1, 2, \ldots$

1) Solve tensor equation $\mathscr{A}(\mathbf{y}^{(k)})^{m-1} = \lambda_{k-1}\mathbf{x}^{(k-1)}$ by Algorithm 3.2;
2) Compute $\mathbf{x}^{(k)} = \mathbf{y}^{(k)}/\|\mathbf{y}^{(k)}\|$;
3) Compute $\lambda_k = \mathscr{G}_1 \left\{(\mathscr{G}_2 \bullet_2 \mathbf{x}^{(k)}) \left[(\mathscr{G}_3 \bullet_2 \mathbf{x}^{(k)}) \cdots \left((\mathscr{G}_{m-1} \bullet_2 \mathbf{x}^{(k)})(\mathscr{G}_m \bullet_2 \mathbf{x}^{(k)})\right)\right]\right\}$.

STEP 3: Return $(\lambda_k, \mathbf{x}^{(k)})$ as an approximate Z-eigenpair of $\mathscr{A}$.

Theoretically speaking, both the two algorithms are feasible for solving the eigenvalue problem of any given tensor, because Algorithm 3.1 and Algorithm 3.2 are capable of solving general tensor equations. Moreover, based on the analysis of the computational complexity of Algorithm 3.2, we know that Algorithm 5.2 is also free from the curse-of-dimensionality, which means that Algorithm 5.2 may be promising for solving large-scale tensor eigenvalue problems, while Algorithm 5.1 and the

existing methods presented in, e.g., [7, 25], may be still intractable for large-scale problems.

6 Numerical experiments

In this section, we test the feasibility and effectiveness of the algorithms we propose in this paper for solving tensor equations and tensor eigenvalue problems, respectively. All codes were written in MATLAB (version R2016a) and run on a personal computer with Inter(R) Core(TM) i5-4200M@2.5 GHz and 4.00 G memory, and all the operations were implemented via the tensor toolbox (version 2.6) [1] and the tensor-train toolbox [38]. For the sake of convenience, we use respectively "IT" and "CPU" to represent the number of iteration steps and the elapsed CPU time in seconds, and RES $= \|\mathscr{A}(\bar{\mathbf{x}}^{(k)})^{m-1} - \mathbf{b}\|$, where $\mathbf{x}_p^{(k)}$ and $\bar{\mathbf{x}}^{(k)}$ are the kth iteration values.

6.1 Numerical results related to tensor equations

For the sake of comparison, the tested tensor equations are all consistent, and then the stopping rule is that RES $< \epsilon := 1.0\text{e}-05$, or the iteration number exceeding the prescribed iteration $k_{\max} = 10000$. Particularly, we choose the penalty parameters μ_p ($p = 2, 3, \ldots, m$) in both Algorithms 3.1 and 3.2 as an invariant constant, denoted by μ. In addition, the number of iteration steps, the CPU time, and the residual contained in the tables below are respectively the average of 5 runs from different starting points unless otherwise stated. Besides, if the coefficient tensor of a tested tensor equation is a full tensor, we obtain its TT-format by using the involved function `tt_tensor` in the tensor-train toolbox with accuracy =1.0e−014.

Example 6.1 We consider the solution of the discretized Klein-Gordon equation, i.e., the tensor equation (1.2), and let the right-hand side $\mathbf{f}$ be the vector such that $\mathscr{L}_h^{(d)}(\mathbf{u}^*)^{m-1} = \mathbf{f}$ with $\mathbf{u}^* = 2 * \texttt{ones}(n, 1)$ for simplicity.

For convenience, we consider the case $d = 1$. Since the tensor $\mathscr{L}_h$ in the tensor equation is a nonsymmetric nonsingular $\mathcal{M}$-tensor equation [7], and the SOR-type method [31] has better performance than some others, for instance, Jacobi and Gauss-seidel methods and their variants proposed in [7], then we compared only with the SOR-type method. Starting with randomly chosen initial vector $\mathbf{u}^{(0)} = \texttt{rand}(n, 1) * 10$ and $\mu = \texttt{rand} * 1000$, we ran the three algorithms mentioned above, and reported the numerical results in Table 2, in which the relaxation parameter ω involved in the SOR-type method was chosen randomly by $\omega = 1 + \texttt{rand}$, and particularly, the symbol ''—" means that the residual did not satisfy the tolerance before the number of iteration steps arrives at the maximum.

From Table 2, one can observe that the three algorithms are convergent before the maximal number $k_{\max}$ except the cases $m = 3, n = 100$ and $m = 4, n = 40, 50$ for the SOR-type method and the case $m = 4, n = 20$ for the G-ADMM method.

Table 2 Numerical results for the tensor equation in Example 6.1

Algorithms $[m, n]$		G-ADMM	TT-ADMM	SOR-type [31]
	IT	52	53	419
[3, 10]	CPU	0.3055	0.2815	2.2412
	RES	7.2183e−06	7.4410e−06	9.5562e−06
	IT	67	63	1881
[3, 20]	CPU	0.4078	0.3413	12.1793
	RES	9.5388e−06	7.0937e−06	9.9013e−06
	IT	52	50	9424
[3, 40]	CPU	0.3523	0.2970	65.1846
	RES	9.2479e−06	8.8712e−06	9.9779e−06
	IT	250	566	2066
[3, 80]	CPU	4.9194	9.5337	22.0313
	RES	8.3193e−06	7.1597e−06	9.9587e−06
	IT	56	65	—
[3, 100]	CPU	1.2577	0.8602	—
	RES	1.4017e−06	3.0666e−06	—
	IT	1818	1866	549
[4, 10]	CPU	18.9756	18.5013	3.9798
	RES	9.9875e−06	9.7224e−06	9.9960e−06
	IT	—	9176	4346
[4, 20]	CPU	—	92.9956	41.1742
	RES	—	6.5256e−06	9.9729e−06
	IT	1955	1953	—
[4, 40]	CPU	208.4802	49.0056	—
	RES	8.2304e−06	4.9580e−06	—
	IT	8853	5691	—
[4, 50]	CPU	1489.3642	230.6611	—
	RES	9.9986e−06	9.9889e−06	—
	IT	250	566	2066
[5, 10]	CPU	4.9194	9.5337	22.0313
	RES	8.3193e−06	7.1597e−06	9.9587e−06
	IT	420	1141	4413
[5, 20]	CPU	52.2996	21.1381	279.8183
	RES	3.8554e−06	6.9550e−06	9.9540e−06
	IT	261	285	1512
[6, 10]	CPU	17.8109	6.6072	45.2949
	RES	9.9396e−06	6.6867e−06	9.8433e−06
	IT	558	149	1756
[7, 10]	CPU	186.5865	5.6144	218.8982
	RES	9.8975e−07	4.7903e−06	9.9170e−06

Moreover, the iteration steps, the elapsed CPU time and the residual corresponding to both G-ADMM and TT-ADMM are commonly less than those of the SOR-type method. In addition, this table also reflects that the iteration steps and the elapsed CPU time of G-ADMM and TT-ADMM are always different due to the influence of errors. Particularly, the CPU time costed by TT-ADMM is less than that of G-ADMM in the sense that they have almost the same iteration steps, which coincides with the analysis presented in Section 3.

In what follows, taking the discretized system (1.2) with $d = 1$ as an example, we consider how the error caused by deriving the tensor-train decomposition affects the effectiveness of our algorithms. We produced 100 different TT-formats of the tensor $\mathscr{L}_h$ with different accuracy "EPS", i.e., EPS=[1.0e-01:−1.0e−03:1.0e−016], and ran the G-ADMM method and the TT-ADMM method with the same initial vector $\mathbf{u}^{(0)} = [1 : n]'$ for the case $[m, n] = [4, 10]$. In Fig. 2, we described the norm of the solutions and the corresponding residuals derived by the two proposed methods versus the accuracy EPS, respectively. From the figures, one can observe that the difference between the solutions (or respectively the residuals) decreases as the accuracy increases, and especially becomes very small when the accuracy is higher than EPS=1.8e−02 and EPS=1.0e−02, respectively, which means that the TT-ADMM method is commonly feasible if the error to derive the TT-format is small enough. Moreover, the convergence behavior of the proposed methods was displayed in Fig. 3.

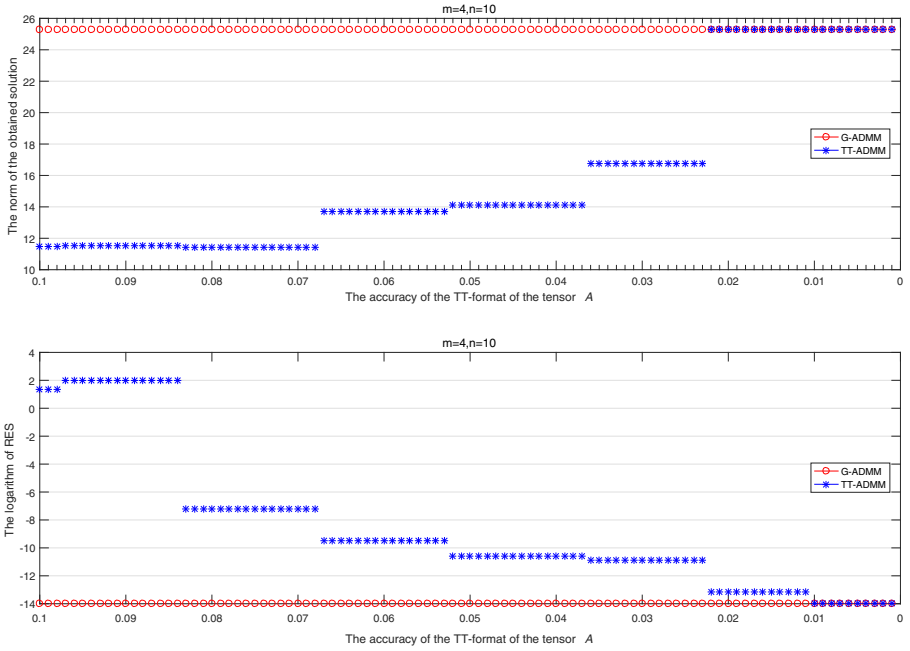

Fig. 2 Comparison of G-ADMM and TT-ADMM for the tensor equation in Example 6.1

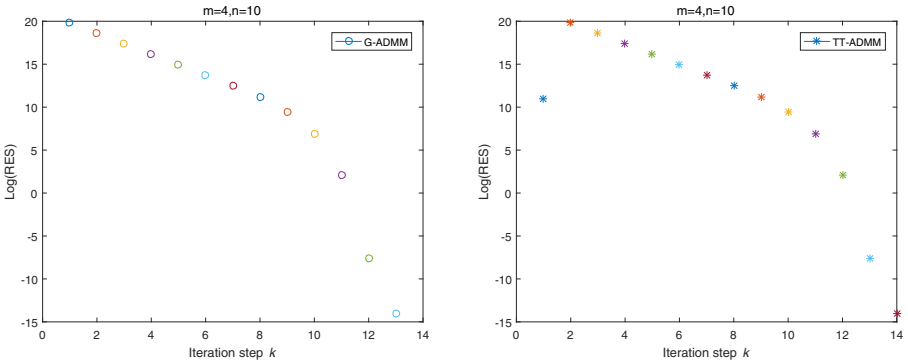

Fig. 3 Comparison on the convergence behavior of the proposed methods for the tensor equation in Example 6.1

Example 6.2 Let the tensor $\mathscr{A}$ in (1.1) be given by

$$\mathscr{A} = s\mathcal{I} - \mathscr{B},\ \mathscr{B} = (b_{i_1 i_2 \ldots i_m}) \in \mathbb{R}^{[m,n]} \text{ with } b_{i_1 i_2 \ldots i_m} = |\tan(i_1 + i_2 + \cdots + i_m)|,$$

where $s = (1+\alpha) \cdot \max_{1 \le i \le n}(\mathscr{B}\mathbf{e}^{m-1})_i$ with $\mathbf{e} = \texttt{ones}(n, 1)$ and $\alpha = 0.01$, and the right-hand side $\mathbf{b}$ be chosen such that $\mathscr{A}(\mathbf{x}^*)^{m-1} = \mathbf{b}$ for $\mathbf{x}^* = 8 * \texttt{ones}(n, 1)$.

The tensor $\mathscr{A}$ defined above is always a symmetric nonsingular $\mathcal{M}$-tensor, which is derived from [16]. When the tensor $\mathscr{A}$ is a nonsingular $\mathcal{M}$-tensor, it was shown [32] that to find a sparsest solutions to the tensor complementarity problem

$$\begin{cases} \min \|\mathbf{x}\|_0 \\ s.t.\ \mathscr{A}\mathbf{x}^{m-1} - \mathbf{b} \ge 0, \mathbf{x} \ge 0, \mathbf{x}^T(\mathscr{A}\mathbf{x}^{m-1} - \mathbf{b}) = 0, \end{cases}$$

is equivalent to search a positive solution of the $\mathcal{M}$-tensor equation (1.1) under the condition that $\mathbf{b}$ is a nonnegative vector, where $\|\cdot\|_0$ denotes the number of nonzero elements of a vector. Therefore, this example can also be regarded as an application of our methods when solving the tensor complementarity problems.

Starting with randomly chosen initial vector $\mathbf{x}^{(0)} = \texttt{rand}(n, 1) * 10$ and $\mu = \texttt{rand}*1000$, we respectively implemented G-ADMM and TT-ADMM, and compared with the Newton method [7] and the SOR-type method [31] for different choices of m and n, because the last two algorithms have better performance, and particularly, the Newton method is theoretically feasible for symmetric $\mathcal{M}$-tensor equations, which is a promising method [7]. The obtained numerical results were reported in Table 3. This table reveals that all the algorithms mentioned above converge to the desired accuracy before reaching the maximum number of iteration steps $k_{\max}$, and in the most cases, the number of iteration steps and the CPU time corresponding to the G-ADMM method and the TT-ADMM method as well as the Newton method are less than those of the SOR-type method. Relatively, the CPU time that the TT-ADMM method spent is also less than that of the G-ADMM method.

Table 3 Numerical results for Example 6.2

Algorithms $[m, n]$		G-ADMM	TT-ADMM	Newton [7]	SOR-type [31]
[3, 10]	IT	8	8	9	226
	CPU	0.0448	0.0611	0.0098	0.9260
	RES	2.7577e−07	2.7567e−07	7.3045E−09	9.3438e−06
[3, 20]	IT	16	16	17	106
	CPU	0.2429	0.1887	0.0620	1.1217
	RES	6.8967e−08	9.5307e−08	3.4022e−08	8.4800e−06
[3, 40]	IT	11	11	12	799
	CPU	0.0803	0.0667	0.0178	4.6079
	RES	2.4467e−07	2.4473e−07	3.3004e−10	9.8501e−06
[3, 80]	IT	12	12	13	114
	CPU	0.1391	0.1093	0.0583	4.3159
	RES	1.7235e−07	1.7508e−07	1.3139e−08	9.0632e−06
[3, 100]	IT	17	17	18	101
	CPU	0.2959	0.2209	0.1278	6.4439
	RES	7.9938e−08	7.8959e−08	4.0291e−09	7.5501e−06
[4, 10]	IT	13	13	14	100
	CPU	0.1470	0.1202	0.0322	0.6227
	RES	2.8662e−08	3.1878e−08	4.6798e−09	7.1084e−06
[4, 20]	IT	322	45	62	155
	CPU	4.5566	0.4711	0.1847	1.4747
	RES	4.0202e−08	8.4469e−08	3.1757e−08	9.5226e−06
[4, 40]	IT	243	439	368	214
	CPU	16.8305	6.7317	6.4615	8.5447
	RES	5.9527e−07	6.8735e−07	2.6610e−07	9.3075e−06
[4, 50]	IT	108	124	168	231
	CPU	16.5617	2.4854	6.6246	19.9548
	RES	7.1131e−07	2.2263e−06	4.4288e−07	9.8645e−06
[5, 10]	IT	17	17	18	162
	CPU	0.3788	0.3109	0.0652	1.5573
	RES	7.3480e−08	1.8428e−07	9.5127e−08	9.3459e−06
[5, 20]	IT	20	20	21	277
	CPU	2.5734	0.4004	0.5628	15.1979
	RES	1.0796e−06	2.3296e−06	7.4171e−07	8.7214e−06
[6, 10]	IT	1409	234	1875	460
	CPU	115.5979	7.4432	25.5439	15.2270
	RES	2.9214e−07	6.5650e−07	2.8681e−07	9.8946e−06
[7, 10]	IT	49	57	76	413
	CPU	0.6073	0.6672	0.1864	3.0254
	RES	1.0443e−04	5.8765e−05	8.3219e−05	9.8075e−04

The performed numerical tests in Examples 6.1 and 6.2 display the superiority of the methods we propose in present paper when they are applied to the tensor equations with special structures, while our approaches are theoretically feasible for any tensor equations, which can also be confirmed numerically by the following example.

Example 6.3 Let the tensor $\mathscr{A}$ in (1.1) be given by $\mathscr{A} = \texttt{tenrand}(\mathbf{v}) \in \mathbb{R}^{[m,n]}$ with $\mathbf{v} = \texttt{ones}(1, m) * n$, and the right-hand side $\mathbf{b}$ be chosen such that $\mathscr{A}(\mathbf{x}^*)^{m-1} = \mathbf{b}$ with $\mathbf{x}^* = 2*\texttt{ones}(n, 1)$, where $\texttt{tenrand}(\mathbf{v})$ returns an mth-order and n-dimensional tensor containing pseudo-random values drawn from the uniform distribution in (0,1).

As chosen in Example 6.2, for randomly initial values and parameters, we respectively performed G-ADMM and TT-ADMM and reported the numerical results in Table 4. From this table, one can see that both of the algorithms are feasible. Particularly, the number of iteration steps and the elapsed CPU time increase as the order m or the dimension n of the tensor $\mathscr{A}$.

6.2 Numerical results related to tensor eigenvalue problems

This subsection is devoted to the computation of Z-eigenpairs of tensors by using the inverse iteration methods twisted from G-ADMM and TT-ADMM, i.e., INV-GADMM and INV-TTADMM. In our tests, the stopping rule is that the difference ERR:=$|\lambda_{k+1} - \lambda_k| < \eta = 1.0\text{e} - 05$, or the number of iteration steps exceeds the maximum $k_{\max} = 100$, while the termination criterion in inner iteration is that the residual RES$< \eta$ or the number of inner iteration steps (denoted by "INT") exceeds the maximum k_{INT}.

Table 4 Numerical results for Example 6.3

Algorithms	G-ADMM			TT-ADMM		
$[m, n]$	IT	CPU	RES	IT	CPU	RES
[3, 10]	42	0.2435	5.1308e−06	35	0.1678	8.9933e−06
[3, 30]	60	0.3131	9.8696e−06	85	0.3918	4.9321e−06
[3, 50]	90	0.5041	9.8612e−06	160	0.7369	9.5644e−06
[3, 100]	178	1.0071	9.5524e−06	154	0.7484	9.3591e−06
[4, 10]	83	0.9306	7.3769e−06	73	0.7378	8.5396e−06
[4, 30]	87	0.9109	9.1836e−06	68	0.7148	9.0353e−06
[4, 50]	156	1.6243	7.2226e−06	114	1.1479	9.7769e−06
[5, 10]	99	1.7509	8.4275e−06	84	1.4720	8.5100e−06
[5, 20]	213	3.6923	9.2637e−06	199	3.4584	9.0893e−06
[6, 10]	115	3.1762	8.3000e−06	99	2.7520	9.3058e−06
[6, 20]	115	3.3001	8.8550e−06	105	3.1587	9.4288e−06
[7, 10]	221	10.1370	5.3943e−06	220	10.0129	6.5320e−06

Example 6.4 Let $\mathscr{A} = (a_{i_1i_2i_3i_4}) \in \mathbb{R}^{[4,3]}$ be the symmetric tensor presented in Example 1 of [23], i.e.,

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

The real Z-eigenvalues of the tensor $\mathscr{A}$ are listed as follows [23]:

$$\begin{aligned}
&0.8893, \ 0.8169, \ 0.5105, \ 0.3633, \ 0.2682, \ 0.2628,\\
&0.2433, \ 0.1735, \ -0.0451, \ -0.5629, \ -1.0954.
\end{aligned}$$

We compared INV-GADMM and INV-TTADMM with the prevalent SS-HOPM method [25], and the codes of the SS-HOPM is from [38] in which the shift is selected randomly. With different initial iterative vectors produced by `randn(3, 1)`, we respectively ran 100 trials of the two algorithms for different choices of $k_{\mathrm{INT}} = 10, 20, 40, 50$, reported their accuracy for computing the Z-eigenpairs of $\mathscr{A}$ in Table 5, and displayed the distribution of the obtained Z-eigenvalues in Fig. 4. From Table 5, one can see that the occurrent number of the obtained Z-eigenvalues varies with k_{INT}, and that the Z-eigenvalues generated by our methods are often more than those of SS-HOPM. The histograms in Fig. 4 reflect that the Z-eigenvalues obtained by our methods lie in the intermediate (by modulus) positions, while the ones obtained by SS-HOPM are at both ends.

It has been shown that the median eigenvalue of a graph plays an important role in mathematical chemistry [9, 14]. Similarly, one can imagine that the median eigenvalue of a hypergraph [6] may also be useful in scientific computing. The above numerical results reveal that the inverse iteration methods we propose here are helpful for finding the median eigenvalue of a tensor.

In addition, as mentioned in Section 5, our inverse iteration methods are able to solve the eigenvalue problem of a general tensor in $\mathbb{R}^{[m,n]}$, which can also be illustrated by the following numerical example.

Table 5 Accuracy of the inverse iteration methods and SS-HOPM for symmetric TEP in Example 6.4

Algorithms	$k_{\mathrm{INT}} = 10$	$k_{\mathrm{INT}} = 20$	$k_{\mathrm{INT}} = 40$	$k_{\mathrm{INT}} = 50$
INV-GADMM	79 %	82 %	84 %	84 %
INV-TTADMM	79 %	82 %	84%	84 %
SS-HOPM	80 %	66 %	82 %	80 %

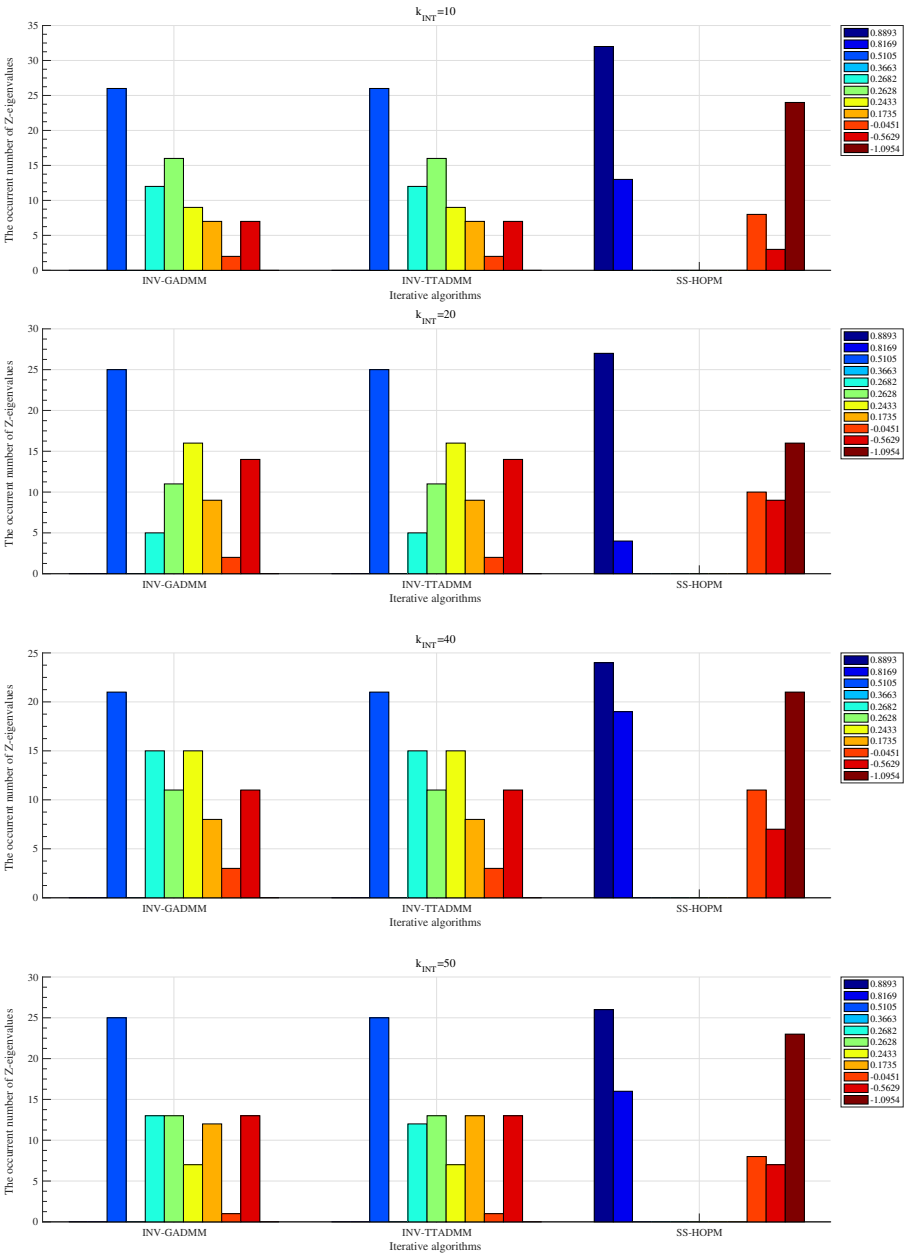

Fig. 4 The distribution of the obtained Z-eigenvalues for the tensor $\mathscr{A}$ in Example 6.4

Example 6.5 Let $\mathscr{A} \in \mathbb{R}^{[4,3]}$ have the same elements as the tensor given in Example 6.4 except the following ones: $a_{2333} = -0.2727$, $a_{3233} = 0.2727$, $a_{3323} = 1.2727$ and $a_{3332} = 2.2727$.

Table 6 Comparison of our inverse iteration methods for nonsymmetric TEP in Example 6.5

Algorithms	EIG	$k_{\text{INT}} = 10$	$k_{\text{INT}} = 20$	$k_{\text{INT}} = 40$	$k_{\text{INT}} = 50$
INV-GADMM	0.2430	38 %	40 %	41 %	37 %
INV-TTADMM	0.2430	38 %	38 %	38 %	34 %

By using Mathematica software, we obtain all the real Z-eigenvalues of the nonsymmetric tensor $\mathscr{A}$ given in this example, namely,

$$1.0896,\ 0.7626,\ 0.5361,\ 0.5101,\ 0.4679,\ 0.3068,$$
$$0.2430,\ 0.1969,\ -0.0426,\ -1.0519,\ -1.1423.$$

We carried out 100 trials of INV-GADMM and INV-TTADMM for randomly initial values generated by $\mathbf{x}^{(0)} = \texttt{randn}(3, 1)$, and reported the obtained Z-eigenvalues (denoted by "EIG") and the corresponding accuracy in Table 6 for $k_{\text{INT}} = 10, 20, 40, 50$, respectively. This table reflects that for each method, the success rate for finding the Z-eigenvalues of $\mathscr{A}$ varies with k_{INT}. Notably, the success rate of the two methods may be different for the same k_{INT} because of the influence of errors. Additionally, an amazed phenomenon, appeared in our experiments and many others trials not listed here, is that only one Z-eigenvalue was found during the performed trials. Therefore, how to improve the efficiency of these algorithms is a problem worthy of consideration.

7 Concluding remarks

In this paper, we proposed an alternating iterative method derived from ADMM for solving the tensor equation (1.1), i.e., Algorithm 3.1. Especially, based on the tensor-train decomposition of tensors, we derived a new version of the above method, i.e., Algorithm 3.2, which is free from the curse-of-dimensionality. Under some assumptions, it has been proved that every limit point of the sequences generalized by our methods satisfies the first-order optimality conditions. Furthermore, we applied these methods to the derivation of the inverse iteration methods for solving tensor eigenvalue problems. Numerical results demonstrate that our methods are feasible and efficient for solving general tensor equations and tensor eigenvalue problems, respectively. Particularly, the proposed methods outperform the existing ones if the penalty parameters are chosen appropriately. During the discussion, we encounter many issues, such as the choice of parameters μ_p, the computation of the median eigenvalue of tensors, and the error of the solution of tensor equations caused by the TT-decomposition, which have not been completely addressed there, and are left as open problems for future studies.

Acknowledgements The authors are very grateful to the editors and two anonymous referees for their constructive comments and valuable suggestions, which greatly improved the original manuscript of this paper. Especially, the first author would like to thank Dr. Yutao Zheng for his selfless help in the process of programming.

Funding information This work was financially supported by the National Natural Science Foundation of China (Grant nos. 11571004 and 11701456). The research of the first author was also financially supported by the Science Foundation of Education Department of Gansu Province (Grant no. 2017A-078) and Tianshui Normal University (Grant no. TAS1603) as well as the Key Discipline Construction Foundation of Tianshui Normal University. The third author was financially supported by the Fundamental Research Funds for the Central Universities (Grant no. lzujbky-2017-it54) as well.

References

1. Bader, B.W., Kolda, T.G., et al.: MATLAB Tensor Toolbox Version 2.6. http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html (2015)
2. Ballani, J., Grasedyck, L.: A projection method to solve linear systems in tensor format. Numer. Linear Algebra Appl. **20**(1), 27–43 (2013)
3. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2010)
4. Chang, T.-H., Hong, M.-Y., Liao, W., Wang, X.-F.: Asynchronous distributed ADMM for large-scale optimization-part i: algorithm and convergence analysis. IEEE Trans. Sig. Process. **64**(12), 3118–3130 (2016)
5. Chen, C.-H., He, B.-S., Yuan, X.-M., Ye, Y.-Y.: The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. Math. Program. **155**(1-2), 57–79 (2016)
6. Cooper, J., Dutle, A.: Spectra of uniform hypergraphs. Linear Algebra Appl. **436**, 3268–3292 (2012)
7. Ding, W.-Y., Wei, Y.-M.: Solving multi-linear systems with $\mathcal{M}$-tensors. J. Sci. Comput. **68**, 689–715 (2016)
8. Dolgov, S.V., Khoromskij, B.N., Oseledets, I.V.: Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker-Planck equation. SIAM J. Sci. Comput. **34**(6), A3016–A3038 (2012)
9. Fowler, P.W., Pisanski, T.: HOMO-LUMO Maps for chemical graphs. MATCH Commun. Math. Comput. Chem. **64**, 373–390 (2010)
10. Forero, P.A., Cano, A., Giannakis, G.B.: Distributed clustering using wireless sensor networks. IEEE J. Selected Topics Sig. Process. **5**, 707–724 (2011)
11. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximation. Comput. Math. Appl. **2**, 17–40 (1976)
12. Grasedyck, L.: Hierarchical singular value decomposition of tensors. SIAM J. Matrix Anal. Appl. **31**(4), 2029–2054 (2010)
13. Grasedyck, L., Kressner, D., Tobler, C.: A literature survey of low-rank tensor approximation techniques. GAMM-Mitt. **36**(1), 53–78 (2013)
14. Gutman, I., Rouvray, D.: An approximate topological formula for the HOMO-LUMO separation in alternate hydrocarbons. Chem. Phys. Lett. **62**, 384–388 (1979)
15. Hackbusch, W., Kuhn, S.: A new scheme for the tensor representation. J. Fourier Anal. Appl. **15**(5), 706–722 (2009)
16. Han, L.-X.: A homotopy method for solving multilinear systems with $\mathcal{M}$-tensors. Appl. Math. Lett. **69**, 49–54 (2017)
17. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference, and prediction. Springer, New York (2001)
18. Hillar, C.J., Lim, L.-H.: Most tensor problems are NP-hard. J. ACM. **60**(6), 1–39 (2013)
19. He, B.-S., Tao, M., Yuan, X.-M.: A splitting method for separable convex programming. IMA J. Numer. Anal. **20**, 1–33 (2014)
20. He, B.-S., Yuan, X.-M.: Linearized alternating direction method of multipliers with Gaussian back substitution for separable convex programming. Numer. Algebra Control Optim. **3**, 247–260 (2013)
21. Holtz, S., Rohwedder, T., Schneider, R.: The alternating linear scheme for tensor optimization in the tensor train format. SIAM J. Sci. Comput. **34**, A683–A713 (2012)
22. Hong, M.-Y., Luo, Z.-Q., Razaviyayn, M.: Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. SIAM J. Optim. **26**(1), 337–364 (2016)
23. Kofidis, E., Regalia, P.A.: On the best rank-1 approximation of higher-order supersymmetric tensors. SIAM J. Matrix Anal. Appl. **23**, 863–884 (2002)

24. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**, 455–500 (2009)
25. Kolda, T.G., Mayo, J.R.: Shifted power method for computing tensor eigenpairs. SIAM J. Matrix Anal. Appl. **32**(4), 1095–1124 (2011)
26. Liavas, A.P., Sidiropoulos, N.D.: Parallel algorithms for constrained tensor factorization via the alternating direction method of multipliers. IEEE Trans. Sig. Process. **63**(20), 5450–5462 (2015)
27. Li, D.-H., Xie, S.-L., Xu, H.-R.: Splitting methods for tensor equations. Numer. Linear Algebra Appl. **24**(5), 1–16 (2017)
28. Li, X.-T., Ng, M.K.: Solving sparse non-negative tensor equations: algorithms and applications. Front. Math. China **10**(3), 649–680 (2015)
29. Lim, L.-H.: Singular values and eigenvalues of tensors: a variational approach. In: proceedings of the 1st IEEE international workshop on computational advances of multi-sensor adaptive processing (CAMSAP), December 13-15, pp. 129–132 (2005)
30. Liu, J., Chen, J., Ye, J.: Large-scale sparse logistic regression. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, June 28-July 1, pp. 547–556 (2009)
31. Liu, D.-D., Li, W., Vong, S.W.: The tensor splitting with application to solve multi-linear systems. J. Comput. Appl. Math. **330**(1), 75–94 (2018)
32. Luo, Z.-Y., Qi, L.-Q., Xiu, N.-H.: The sparsest solutions to Z-tensor complementarity problems. Optim Lett. **11**, 471–482 (2017)
33. 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)
34. Qi, L.-Q.: Eigenvalues of a real supersymmetric tensor. J. Symb. Comput. **40**, 1302–1324 (2005)
35. Qi, L.-Q., Luo, Z.-Y.: Tensor analysis: spectral theory and special tensors. SIAM, Philadelphia (2017)
36. Ortega, J.M., Rheinboldt, W.C.: Iterative solution of nonlinear equations in several variables. Academic Press, New York (1970)
37. Oseledets, I.V.: Tensor-train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011)
38. Oseledets, I.V., et al.: TT-Toolbox. https://github.com/oseledets/TT-Toolbox (2016)
39. Oseledets, I.V., Tyrtyshnikov, E.E.: Breaking the curse of dimensionality, or how to use SVD in many dimensions. SIAM J. Sci. Comput. **31**(5), 3744–3759 (2009)
40. Sun, D.-F., Toh, K.-C., Yang, L.-Q.: A convergent 3-block semiproximal alternating direction method of multipliers for conic programming with 4-type constraints. SIAM J. Optim. **25**(2), 882–915 (2015)
41. Tibshirani, R.: Regression shrinkage and selection via the LASSO. J. Roy. Stat. Soc. B **58**, 267–288 (1996)
42. Wang, Y., Yin, W.-T., Zeng, J.-S.: Global convergence of ADMM in nonconvex nonsmooth optimization. J. Sci. Comput. https://doi.org/10.1007/s10915-018-0757-z (2018)
43. Xie, Z.-J., Jin, X.-Q., Wei, Y.-M.: A fast algorithm for solving circulant tensor systems. Linear Multilinear Algebra **65**(9), 1894–1904 (2017)
44. Xu, Y.-Y., Yin, W.-T., Wen, Z.-W., Zhang, Y.: An alternating direction algorithm for matrix completion with nonnegative factors. Front. Math. China **7**(2), 365–384 (2012)
45. Zhang, J.-Y., Wen, Z.-W., Zhang, Y.: Subspace methods with local refinements for eigenvalue computation using low-rank tensor-train format. J. Sci. Comput. **70**, 478–499 (2017)
46. Zwillinger, D. Handbook of Differential Equations, 3rd edn. Academic Press Inc, Boston (1997)