Journal of Industrial and Management Optimization

https://www.aimspress.com/journal/jimo

JIMO, 22(10): 4978–5000.
DOI:10.3934/jimo.2026172
Received: 07 July 2026
Revised: 01 September 2026
Accepted: 07 September 2026
Published: 11 September 2026

Research article

Modified limited-memory BFGS method with a cautious update for generalized semi-symmetric tensor equations

Yuncheng Xu[1], **Sanyang Liu**[1] **and Huiping Cao**[2,*]

[1] School of mathematics and statistics, Xidian University, Xi'an, Shaanxi, China

[2] School of sciences, Xi'an Polytechnic University, Xi'an, Shaanxi, China

* **Correspondence:** Email: huiping_cao@hnu.edu.cn.

Abstract: For the generalized semi-symmetric tensor equations, this paper reformulates the equations as a nonlinear least-squares problem and proposes a modified L-BFGS method with a cautious update. The method constructs a candidate modified curvature vector using high-order curvature information and employs a cautious criterion to select reliable curvature pairs. Under suitable assumptions, the global convergence of the proposed method is established with a Wolfe line search. Numerical experiments on semi-symmetric tensor test problems with positive and nonuniform coupling show that the proposed method exhibits good stability and competitive performance.

Keywords: generalized tensor equations; semi-symmetric tensors; limited-memory BFGS method; cautious update; global convergence
Mathematics Subject Classification: 90C30, 90C26

1. Introduction

As a higher-order generalization of vectors and matrices, tensors have become a fundamental mathematical tool for solving large-scale problems, with broad applications in signal processing, image restoration, data mining, psychometrics, quantum entanglement, and medical engineering [1–5]. Within this context, tensor equations involving tensor-vector products are extensively used for solving numerical partial differential equations [6], tensor eigenvalue complementarity problems [7], and the stationary probability distribution of higher-order Markov chains [8]. Such equations can generally be formulated as follows

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

where $\mathcal{A} \in \mathbb{R}^{n\times n\times\cdots\times n}$ is an m-th order tensor, $b \in \mathbb{R}^n$. Specifically, $\mathcal{A}x^{m-1}$ denotes an n-dimensional vector whose i-th component is given by

$$(\mathcal{A}x^{m-1})_i = \sum_{i_2=1}^{n} \cdots \sum_{i_m=1}^{n} a_{ii_2\ldots i_m} x_{i_2} \cdots x_{i_m}, \quad i \in [n]. \tag{2}$$

Numerous effective algorithms have been developed for tensor equations. Li et al. [4] systematically extended the Jacobi, Gauss-Seidel, and SOR methods to symmetric tensor equations, and proved their global convergence and local R-linear convergence rate under appropriate conditions. Moreover, they further incorporated Newton's method to develop a faster Newton-Gauss-Seidel hybrid algorithm. For more general semi-symmetric tensor equations, Lv et al. [9] designed a Levenberg-Marquardt (LM) algorithm, which achieves global convergence and local quadratic convergence under a local error bound condition weaker than nonsingularity, with applications to H-eigenvalue problems. Li et al. [10] formulated the solution of a third-order tensor equation as the intersection of quadratic surfaces, proposing a hybrid alternating projection algorithm with local linear convergence. To address large-scale systems, Najafi et al. [11] introduced an iterative block method based on block partitioning of the coefficient tensor, providing a new approach for efficiently solving large systems. Neural-network-based approaches have also been developed for M-tensor and time-varying multilinear systems [12–14].

However, existing research has mostly focused on homogeneous tensor equations, whereas nonhomogeneous tensor equations arise in practical applications such as higher-order Markov chains [8] and Multilinear PageRank [15]. In particular, Multilinear PageRank seeks a probability vector x satisfying

$$x = \alpha P x^{m-1} + (1-\alpha)v,$$

where P is a transition probability tensor, v is a teleportation vector, and $0 < \alpha < 1$. Such nonhomogeneous tensor models motivate the study of Generalized Tensor Equations (GTEs) of the form

$$\mathcal{A}_m x^{m-1} + \mathcal{A}_{m-1} x^{m-2} + \cdots + \mathcal{A}_2 x = b, \tag{3}$$

where $\mathcal{A}_p$ denotes an n-dimensional tensor of order $p\,(p = 2, 3, \cdots, m)$. For such nonhomogeneous tensor equations, if all coefficient tensors are semi-symmetric, they are called Generalized Semi-Symmetric Tensor Equations (GSTEs). Yan et al. [16] systematically analyzed their distinctions from general tensor equations and introduced a new class of Z^+-tensors, which contains all P-tensors as a proper subset. They further proved that if the leading coefficient is a Z^+-tensor, at least one solution exists for any right-hand side vector, and developed an improved LM algorithm for efficient numerical solutions. In related work, Sharma et al. [17] proposed the Generalized Tensor Absolute Value Equation, established its equivalence to the Generalized Tensor Complementary Problem, and derived a solution existence theorem by using degree theory. Jiang et al. [18] proposed a locally convergent slice tensor splitting method, while Wang et al. [19] developed greedy randomized Kaczmarz-type methods with deterministic convergence. Feng et al. [20] developed a tensor-train-based alternating direction method for large-scale complex semi-symmetric tensor equations, and Nie et al. [21] proposed a generalized SOR-type method for generalized tensor absolute value equations.

Newton's method and its variants have demonstrated strong numerical performance in solving tensor equations. For M-tensor equations, Ding et al. [6] developed generalized iterative and Newton-type algorithms. He et al. [22] proved that solving M-tensor equations is equivalent to solving nonlinear

equations with P-functions. Based on this equivalence, they developed a Newton-type method with quadratic convergence. Wang et al. [23] developed a novel quasi-Newton method based on third-order tensor expansion and established its global convergence. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm is widely regarded as one of the most effective quasi-Newton methods for nonlinear optimization problems due to its computational efficiency [24–26].

For large-scale systems, the Limited-memory BFGS (L-BFGS) method [27] was introduced to overcome the storage limitations of conventional quasi-Newton methods. By storing only the most recent $\tilde{m}$ pairs of curvature information $\{s_i, y_i\}$, it constructs an approximation of the inverse Hessian with minimal memory usage while preserving a fast convergence rate, making it one of the most effective methods for large-scale problems. It has been shown that the L-BFGS method converges linearly [28]. Xiao et al. [29] incorporated the modified update formula from [26] into the L-BFGS method, enhancing its generality. To address the limitations of standard L-BFGS in capturing curvature information, Biglari et al. [30] developed a modified L-BFGS method based on a high-order tensor model, which shows improved numerical performance on large-scale unconstrained optimization problems. In summary, while existing research on homogeneous tensor equations has been widely established, studies on GSTEs (3) remain relatively limited. Motivated by the remarkable advantages of L-BFGS methods in solving large-scale problems, this paper proposes a **modified L-BFGS (ML-BFGS)** method with a cautious update for solving the GSTEs (3). First, the GSTEs (3) are reformulated as a nonlinear least-squares problem, where the semi-symmetry of the coefficient tensors is explicitly used in deriving the Jacobian. The high-order curvature quantity introduced in [30] is then adopted as a directional-curvature correction. Based on this quantity, we construct a candidate modified curvature vector through a minimum-norm correction of the standard curvature vector and further employ a cautious criterion to determine whether the candidate pair is retained in the limited-memory update. The convergence of the resulting method under a Wolfe line search is analyzed. Numerical experiments on semi-symmetric tensor test problems with positive and nonuniform coupling are performed, and the results show that the proposed method exhibits good stability and competitive performance.

The remainder of this paper is arranged as follows. Section 2 introduces the preliminaries. In Section 3, a ML-BFGS method with cautious update for solving the GSTEs (3) is presented. Convergence analysis is provided in Section 4. Numerical experiments in Section 5 validate the effectiveness and feasibility of the method. Finally, Section 6 concludes the paper.

2. Preliminaries

This subsection presents an overview of the essential concepts and the problem formulation. For any positive integer q, let $[q] := \{1, 2, \ldots, q\}$. Let $m, n > 2$ be given constants, and denote by $\mathbb{R}^{[m,n]}$ the set of all m-th order n-dimensional real tensors. Let $\mathbb{R}^n$ represent the space of n-dimensional real column vectors, for any vectors $x, y \in \mathbb{R}^n$, their Euclidean inner product is denoted by $x^\top y$, and the Euclidean norm of x is defined as $\|x\| = \sqrt{x^\top x}$. Moreover, we denote $\|A\|$ as the Frobenius norm for matrix $A \in \mathbb{R}^{n\times n}$.

Definition 2.1. [31] Let $\mathcal{A} = (a_{i_1 i_2 \cdots i_m}) \in \mathbb{R}^{[m,n]}$. If the entries satisfy

$$a_{i_1 i_2 \cdots i_m} = a_{\pi(i_1 i_2 \cdots i_m)}, \quad \forall \pi \in \Pi_m,$$

where Π_m is the set of all permutations of the index vector $[1, 2, \cdots, m]$, then the tensor $\mathcal{A}$ is called a

symmetric tensor. If the entries satisfy

$$a_{i_1 i_2 \cdots i_m} = a_{i_1 \pi'(i_2 \cdots i_m)}, \quad \forall \pi' \in \widehat{\Pi}_{m-1},$$

where $\widehat{\Pi}_{m-1}$ is the set of all permutations of the index vector $[2, 3, \cdots, m]$, then the tensor $\mathcal{A}$ is called a semi-symmetric tensor.

Definition 2.2. [32] Let $\mathcal{A} = (a_{i_1 i_2 \cdots i_m})$ and $\mathcal{B} = (b_{i_1 i_2 \cdots i_m}) \in \mathbb{R}^{[m,n]}$, the inner product of $\mathcal{A}$ and $\mathcal{B}$ is defined as

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_m=1}^{n} a_{i_1 i_2 \cdots i_m} b_{i_1 i_2 \cdots i_m}.$$

Definition 2.3. [32] Let $\mathcal{A} = (a_{i_1 i_2 \cdots i_m}) \in \mathbb{R}^{[m,n]}$, its norm is defined as

$$\|\mathcal{A}\| = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle} = \sqrt{\sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_m=1}^{n} a_{i_1 i_2 \cdots i_m}^2}.$$

Building upon the foundational concepts above, to solve the GSTEs (3) effectively, we first reformulate it as a nonlinear least-squares problem, which provides a more tractable framework for numerical computation. Specifically, we define

$$F(x) = \mathcal{A}_m x^{m-1} + \mathcal{A}_{m-1} x^{m-2} + \cdots + \mathcal{A}_2 x - b, \tag{4}$$

and construct the corresponding least squares problem as follows

$$f(x) = \frac{1}{2} \|F(x)\|^2. \tag{5}$$

Since $f(x) \geq 0$, x^* solves the GSTEs (3) if and only if $f(x^*) = 0$. Hence, when (3) is solvable, its solutions are the global minimizers of

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} \left\| \mathcal{A}_m x^{m-1} + \mathcal{A}_{m-1} x^{m-2} + \cdots + \mathcal{A}_2 x - b \right\|^2. \tag{6}$$

This least-squares reformulation provides a natural bridge between (3) and unconstrained optimization. Since $\mathcal{A}_p$ is a semi-symmetric tensor, differentiating $(\mathcal{A}_p x^{p-1})_i$ with respect to x_j produces $p-1$ identical terms. Hence, for $i, j \in [n]$,

$$\frac{\partial (\mathcal{A}_p x^{p-1})_i}{\partial x_j} = (p-1) \sum_{i_3, \ldots, i_p = 1}^{n} a_{iji_3 \cdots i_p}^{(p)} x_{i_3} \cdots x_{i_p} = (p-1)(\mathcal{A}_p x^{p-2})_{ij}.$$

Therefore,

$$\frac{d}{dx}(\mathcal{A}_p x^{p-1}) = (p-1) \mathcal{A}_p x^{p-2}. \tag{7}$$

Therefore, the Jacobian matrix of $F(x)$ is given by

$$J(x) = \sum_{p=2}^{m} (p-1) \mathcal{A}_p x^{p-2} = \sum_{p=0}^{m-2} (p+1) \mathcal{A}_{p+2} x^p, \tag{8}$$

where the (i, j)-th entry of $\mathcal{A}_p x^{p-2} \in \mathbb{R}^{n\times n}$ is given by

$$(\mathcal{A}_p x^{p-2})_{ij} = \sum_{i_3,\ldots,i_p=1}^{n} a^{(p)}_{iji_3\ldots i_p} x_{i_3} \cdots x_{i_p}, \quad \forall i, j \in [n]. \tag{9}$$

The gradient of the objective function $f(x)$ is given by

$$g(x) = \nabla\left(\frac{1}{2}F(x)^\top F(x)\right) = J(x)^\top F(x). \tag{10}$$

For brevity, we denote $F(x_k)$, $J(x_k)$, and $g(x_k)$ by F_k, J_k, and g_k, respectively.

3. Modified L-BFGS method with a cautious update

The classical quasi-Newton method generates the iterative sequence by

$$x_{k+1} = x_k + \alpha_k d_k,$$

where $\alpha_k > 0$ is the step size and the search direction $d_k \in \mathbb{R}^n$ is obtained by solving

$$B_k d_k = -g_k,$$

where $g_k = \nabla f(x_k)$, $B_k \in \mathbb{R}^{n\times n}$ is an approximation of the Hessian matrix $\nabla^2 f(x_k)$ and satisfies the secant equation

$$B_{k+1} s_k = y_k,$$

where $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$. Among various quasi-Newton methods, BFGS method

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}, \tag{11}$$

is one of the most widely used. In practical implementations, the search direction is usually computed as

$$d_k = -H_k g_k,$$

where $H_k = B_k^{-1}$ is the inverse of the BFGS Hessian approximation. Accordingly, by applying the Sherman–Morrison–Woodbury formula twice to (11), we obtain

$$\begin{aligned} H_{k+1} &= \Big(I - \frac{s_k y_k^\top}{y_k^\top s_k}\Big) H_k \Big(I - \frac{s_k y_k^\top}{y_k^\top s_k}\Big)^\top + \frac{s_k s_k^\top}{y_k^\top s_k} \\ &= H_k + \frac{(s_k - H_k y_k) s_k^\top + s_k (s_k - H_k y_k)^\top}{y_k^\top s_k} - \frac{(s_k - H_k y_k)^\top y_k}{(y_k^\top s_k)^2} s_k s_k^\top. \end{aligned} \tag{12}$$

However, for the nonlinear least-squares problem associated with GSTEs (3), the standard curvature vector y_k may not adequately characterize the local curvature variation between two consecutive iterates. To enrich the curvature information used in the L-BFGS update, following Biglari et al. [30], we employ the high-order curvature quantity

$$\phi_k^H = 4(f_k - f_{k+1}) + 2(g_{k+1} + g_k)^\top s_k. \tag{13}$$

This quantity is derived from a fourth-order tensor model of the objective function. In particular,

$$s_k^\top G_{k+1} s_k = s_k^\top y_k + \phi_k^H + \frac{1}{6} s_k^\top (T_{k+1} s_k) s_k + O(\|s_k\|^5),$$

where $G_{k+1} = \nabla^2 f(x_{k+1})$ and T_{k+1} denotes the third-order derivative tensor. Thus, ϕ_k^H supplements the standard curvature $s_k^\top y_k$ with higher-order local information and provides a refined approximation to the directional curvature without requiring explicit Hessian or higher-order derivative evaluations. Using the high-order curvature information in (13), we seek a modified curvature vector that satisfies the corrected directional curvature condition while deviating as little as possible from the standard curvature vector y_k. Specifically, for $s_k \neq 0$, consider

$$\begin{aligned} \min_{v \in \mathbb{R}^n} \quad & \frac{1}{2}\|v - y_k\|^2 \\ \text{s.t.} \quad & s_k^\top v = s_k^\top y_k + \phi_k^H. \end{aligned}$$

The unique solution is

$$\widetilde{y}_k = y_k + \frac{\phi_k^H}{\|s_k\|^2} s_k. \tag{14}$$

Geometrically, $\widetilde{y}_k$ is the orthogonal projection of y_k onto the hyperplane $\{v \in \mathbb{R}^n : s_k^\top v = s_k^\top y_k + \phi_k^H\}$. Hence,

$$s_k^\top \widetilde{y}_k = s_k^\top y_k + \phi_k^H. \tag{15}$$

Therefore, the construction incorporates the high-order directional curvature correction with the minimum modification of y_k: only its component along s_k is changed, while the orthogonal component remains unchanged. In particular, $\widetilde{y}_k = y_k$ when $\phi_k^H = 0$.

Given the potential non-convexity of the objective function associated with GSTEs (3), curvature pairs with nonpositive or insufficient directional curvature may lead to unreliable BFGS updates. Following the cautious-update strategy in [26], we therefore accept the candidate pair $(s_k, \widetilde{y}_k)$ only if $\frac{s_k^\top \widetilde{y}_k}{\|s_k\|^2} \geq \varepsilon_c \|g_k\|^\mu$. This criterion guarantees $s_k^\top \widetilde{y}_k > 0$ for every accepted pair, thereby preserving the positive-curvature condition required by the BFGS update. Moreover, by excluding pairs whose directional curvature is too small relative to $\|s_k\|^2$, the criterion prevents insufficient curvature information from entering the BFGS update and thereby reduces the risk of unreliable curvature corrections. Based on the modified curvature vector in (14) and the cautious update criterion, we define the following cautious BFGS update:

$$H_{k+1} = \begin{cases} (\widetilde{V}_k)^\top H_k \widetilde{V}_k + \widetilde{\rho}_k s_k s_k^\top, & \text{if } \dfrac{s_k^\top \widetilde{y}_k}{\|s_k\|^2} \geq \varepsilon_c \|g_k\|^\mu, \\ H_k, & \text{otherwise,} \end{cases} \tag{16}$$

where $\widetilde{\rho}_k = 1/(s_k^\top \widetilde{y}_k)$, $\widetilde{V}_k = I - \widetilde{\rho}_k s_k^\top \widetilde{y}_k$.

For solving large-scale problems, i.e., minimizing objective functions involving thousands or millions of variables, limited-memory variants of quasi-Newton methods, such as the limited-memory variant of BFGS (known as L-BFGS [27]), have been successful for various applications. In the first m iterations, L-BFGS and BFGS generate the same search directions. But for $k > m$, H_{k+1} is obtained by applying m BFGS updates to H_k using information from the m previous iterations. However, in the present method, only the accepted modified curvature pairs $(s_j, \widetilde{y}_j)$ are stored. Before the number of stored pairs reaches the prescribed memory size $\tilde{m}$, all available accepted pairs are used; once it exceeds $\tilde{m}$, only the

most recent $\tilde{m}$ pairs are retained. To describe the limited-memory update precisely, let $\mathcal{M}_k$ denote the limited-memory set available after the k-th iteration, with $|\mathcal{M}_k| < \tilde{m}$. After the candidate pair $(s_k, \widetilde{y}_k)$ is generated, the limited-memory set is updated by

$$\mathcal{M}_{k+1} = \begin{cases} \mathcal{T}_{\tilde{m}}\left(\mathcal{M}_k \cup \{(s_k, \widetilde{y}_k)\}\right), & \text{if} \dfrac{s_k^\top \widetilde{y}_k}{\|s_k\|^2} \geq \varepsilon_c \|g_k\|^\mu, \\ \mathcal{M}_k, & \text{otherwise}, \end{cases} \tag{17}$$

where $\mathcal{T}_{\tilde{m}}(\cdot)$ retains only the most recent $\tilde{m}$ accepted curvature pairs. Let m_{k+1} denote the number of vector pairs in the set $\mathcal{M}_{k+1}$, that is $m_{k+1} = |\mathcal{M}_{k+1}|$. If $\mathcal{M}_{k+1}$ is nonempty, then $m_{k+1} \leq \tilde{m}$, and we arrange the curvature pairs stored in it in chronological order as

$$\mathcal{M}_{k+1} = \left\{(s_{i_1}, \widetilde{y}_{i_1}), \cdots, (s_{i_{m_{k+1}}}, \widetilde{y}_{i_{m_{k+1}}})\right\}, \quad i_1 < \cdots < i_{m_{k+1}}.$$

For each accepted curvature pair, define

$$\widetilde{\rho}_{i_j} = \frac{1}{s_{i_j}^\top \widetilde{y}_{i_j}}, \quad \widetilde{V}_{i_j} = I - \widetilde{\rho}_{i_j} s_{i_j}^\top \widetilde{y}_{i_j}, \quad j = 1, 2, \cdots, m_{k+1}.$$

Since every accepted curvature pair satisfies $s_{i_j}^\top \widetilde{y}_{i_j} > 0$, $\widetilde{\rho}_{i_j}$ is well defined. Based on the proposed inverse BFGS formula (16), the inverse-Hessian approximation represented by the stored curvature pairs can be written in the following equivalent matrix form:

$$\begin{aligned} H_{k+1} = & \left[\widetilde{V}_{i_{m_{k+1}}}^\top \cdots \widetilde{V}_{i_2}^\top \widetilde{V}_{i_1}^\top\right] H_{k+1}^{(0)} \left[\widetilde{V}_{i_1} \widetilde{V}_{i_2} \cdots \widetilde{V}_{i_{m_{k+1}}}\right] \\ & + \widetilde{\rho}_{i_1} \left[\widetilde{V}_{i_{m_{k+1}}}^\top \cdots \widetilde{V}_{i_3}^\top \widetilde{V}_{i_2}^\top\right] s_{i_1} s_{i_1}^\top \left[\widetilde{V}_{i_2} \widetilde{V}_{i_3} \cdots \widetilde{V}_{i_{m_{k+1}}}\right] \\ & + \cdots \\ & + \widetilde{\rho}_{i_{m_{k+1}-1}} \widetilde{V}_{i_{m_{k+1}}}^\top s_{i_{m_{k+1}-1}} s_{i_{m_{k+1}-1}}^\top \widetilde{V}_{i_{m_{k+1}}} + \widetilde{\rho}_{i_{m_{k+1}}} s_{i_{m_{k+1}}} s_{i_{m_{k+1}}}^\top. \end{aligned} \tag{18}$$

If $\mathcal{M}_{k+1}$ is empty, set $H_{k+1} = H_{k+1}^{(0)}$. We note that the matrices H_k are not formed or stored explicitly, but the $\tilde{m}$ previous values of $\widetilde{y}_{i_j}$ and s_{i_j} are stored separately.

Based on the above analysis, the detailed steps of the ML-BFGS method are presented below.

Algorithm 1 ML-BFGS method with a cautious update for solving GSTEs (3).

Step 1. Input the semi-symmetric tensors $\mathcal{A}_p$. Given initial point $x_0 \in \mathbb{R}^n$ and initial symmetric positive definite matrix H_0. Let $0 < c_1 < c_2 < 1$, $\varepsilon_c > 0$, $\mu > 0$, $\tilde{m} > 0$, $\epsilon > 0$. Set $k := 0$ and $\mathcal{M}_0 := \emptyset$, and $H_0 = H_0^{(0)}$.

Step 2. Compute F_k, f_k, J_k, and g_k by (4), (5), (8), and (10), respectively.

Step 3. If $\|g_k\| \leq \epsilon$, stop. Otherwise, go to Step 4.

Step 4. Compute

$$d_k = -H_k g_k.$$

Step 5. Find a step size $\alpha_k > 0$ satisfying the Wolfe conditions

$$\begin{cases} f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k g_k^\top d_k, \\ g(x_k + \alpha_k d_k)^\top d_k \geq c_2 g_k^T d_k. \end{cases}$$

Step 6. Set $x_{k+1} = x_k + \alpha_k d_k$.
Step 7. Compute $\widetilde{y}_k$ by (14). If $\frac{s_k^\top \widetilde{y}_k}{\|s_k\|^2} \geq \varepsilon_c \|g_k\|^\mu$, then set

$$\mathcal{M}_{k+1} = \mathcal{T}_{\tilde{m}} \left(\mathcal{M}_k \cup \{(s_k, \widetilde{y}_k)\} \right),$$

otherwise, set $\mathcal{M}_{k+1} = \mathcal{M}_k$. Update $H_{k+1}^{(0)}$ to get H_{k+1} by (18). In particular, if $m_{k+1} = 0$, use $H_{k+1}^{(0)}$ directly.
Step 8. Set $k := k + 1$, and go to Step 2.

Remark 3.1. *For Algorithm 1, the candidate curvature pair $(s_k, \widetilde{y}_k)$ is used to update H_k only if $\frac{s_k^\top \widetilde{y}_k}{\|s_k\|^2} \geq \epsilon_c \|g_k\|^\mu$. At any non-terminal iteration, this condition ensures*

$$s_k^\top \widetilde{y}_k > 0. \tag{19}$$

thereby preserving the positive definiteness of H_k.

Remark 3.2. *Since Algorithm 1 uses the two-loop recursion with at most $\widetilde{m}$ stored curvature pairs, the storage requirement and the computational cost of computing each search direction are both $O(\widetilde{m}n)$. The additional operations required for constructing $\widetilde{y}_k$ and evaluating the cautious criterion are $O(n)$. Hence, the proposed modifications do not change the leading-order storage or computational complexity of standard L-BFGS.*

4. Convergence analysis

This section establishes a global convergence result for Algorithm 1 in terms of stationarity of (6). For the convenience of the convergence analysis, we use the corresponding Hessian approximation $B_k = H_k^{-1}$. Under exact arithmetic, the direct BFGS update for B_k and the inverse BFGS update for H_k are algebraically equivalent when initialized by inverse matrices and using the same curvature pairs [33]. Accordingly, for each k, define $B_k^{(0)} = (H_k^{(0)})^{-1}$, and assume that $H_k^{(0)}$ and $B_k^{(0)}$ are uniformly bounded. This assumption can be satisfied by the truncated BB-type scaling, where $H_{k+1}^{(0)} = \gamma_k I$ and $0 < \gamma_{\min} \leq \gamma_k \leq \gamma_{\max}$. Indeed, $B_{k+1}^{(0)} = \gamma_k^{-1} I$, so both matrices are uniformly bounded. Let the limited-memory set obtained after the k-th iteration be given by

$$\mathcal{M}_{k+1} = \left\{ (s_{i_1}, \widetilde{y}_{i_1}), (s_{i_2}, \widetilde{y}_{i_2}), \ldots, (s_{i_{m_{k+1}}}, \widetilde{y}_{i_{m_{k+1}}}) \right\}, \quad i_1 < i_2 < \cdots < i_{m_{k+1}},$$

where $m_{k+1} = |\mathcal{M}_{k+1}| \leq \tilde{m}$. Here, $i_1, \ldots, i_{m_{k+1}}$ represent the indices of the currently stored curvature pairs, arranged in chronological order from the oldest to the newest. Set $B_{k+1}^{(0)} = (H_{k+1}^{(0)})^{-1}$. For $j = 1, 2, \ldots, m_{k+1}$, define

$$B_{k+1}^{(j)} = B_{k+1}^{(j-1)} - \frac{B_{k+1}^{(j-1)} s_{i_j} s_{i_j}^\top B_{k+1}^{(j-1)}}{s_{i_j}^\top B_{k+1}^{(j-1)} s_{i_j}} + \frac{\widetilde{y}_{i_j} \widetilde{y}_{i_j}^\top}{s_{i_j}^\top \widetilde{y}_{i_j}}. \tag{20}$$

and finally set $B_{k+1} = B_{k+1}^{(m_{k+1})}$. If $m_{k+1} = 0$, then $B_{k+1} = B_{k+1}^{(0)}$.

We first state the basic assumptions.

Assumption A. The level set $\mathcal{L}(x_0) = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\}$ is bounded.

Lemma 4.1. *Let $\mathcal{A}_j$ be a jth-order n-dimensional semi-symmetric tensor for $j = 2, \ldots, m$. If $L_0 \subset \mathcal{L}(x_0)$ is a bounded closed set, then both $F(x)$ and $J(x)$ are Lipschitz continuous on L_0.*

Proof. According to [9], $\mathcal{A}_j x^{j-1}$ is Lipschitz continuous on L_0, i.e., there exists a constant $L_j > 0$ such that for any $x, y \in L_0$, the following inequality holds

$$\|\mathcal{A}_j x^{j-1} - \mathcal{A}_j y^{j-1}\| \leq L_j \|x - y\|.$$

Consequently,

$$\|F(x) - F(y)\| = \left\| \sum_{j=2}^{m} \left(\mathcal{A}_j x^{j-1} - \mathcal{A}_j y^{j-1} \right) \right\| \leq \sum_{j=2}^{m} L_j \|x - y\| = L_F \|x - y\|,$$

where $L_F = \sum_{j=2}^{m} L_j$. Hence $F(x)$ is Lipschitz continuous on L_0.

Moreover, $\mathcal{A}_j x^{j-2}$ is Lipschitz continuous on L_0, i.e., there exists a constant $M_j > 0$ such that for any $x, y \in L_0$, the following inequality holds

$$\|\mathcal{A}_j x^{j-2} - \mathcal{A}_j y^{j-2}\| \leq M_j \|x - y\|.$$

According to the definition of Jacobian matrix (8), it follows that

$$\|J(x) - J(y)\| \leq \sum_{j=3}^{m} (j-1) \|\mathcal{A}_j x^{j-2} - \mathcal{A}_j y^{j-2}\| \leq \sum_{j=3}^{m} (j-1) M_j \|x - y\| = L_J \|x - y\|,$$

where $L_J = \sum_{j=3}^{m} (j-1) M_j$. Therefore, $J(x)$ is Lipschitz continuous on L_0.

Corollary 4.1. *Under the conditions of Lemma 4.1, the continuity of $F(x)$ and $J(x)$ on the bounded closed set L_0 implies that they are bounded on L_0. Hence, there exist constants $M_F, M_J > 0$ such that, for any $x \in L_0$,*

$$\|F(x)\| \leq M_F, \quad \|J(x)\| \leq M_J.$$

Lemma 4.2. *If $L_0 \subset L(x_0)$ is a bounded closed set, then the function $f(x)$ is continuously differentiable on L_0, and its gradient $g(x)$ is Lipschitz continuous on L_0. That is, there exists a constant $L > 0$ such that for any $x, y \in L_0$, the following inequality holds*

$$\|g(x) - g(y)\| \leq L \|x - y\|. \tag{21}$$

Proof. For any $x, y \in L_0$, we have

$$\begin{aligned} g(x) - g(y) &= J(x)^\top F(x) - J(y)^\top F(y) \\ &= (J(x)^T - J(y)^\top) F(x) + J(y)^\top (F(x) - F(y)). \end{aligned}$$

Taking norms

$$\|g(x) - g(y)\| \leq \|J(x)^\top - J(y)^\top\| \|F(x)\| + \|J(y)^\top\| \|F(x) - F(y)\|.$$

From Lemma 4.1 and Corollary 4.1, there exists a constant $L > 0$ such that

$$\|g(x) - g(y)\| \leq L \|x - y\|.$$

Using the Lipschitz continuity of the gradient and the Wolfe line search conditions, we can obtain a unified descent estimate.

Lemma 4.3. *Let $\{x_k\}$ be generated by Algorithm 1. If $\|g_k\| \geq \varepsilon$ holds for all k with some constant $\varepsilon > 0$, then for every curvature pair $(s_k, \widetilde{y}_k)$ accepted by the cautious criterion, there exists a constant $M_1 > 0$ such that*

$$\frac{\|\widetilde{y}_k\|^2}{s_k^\top \widetilde{y}_k} \leq M_1. \tag{22}$$

Proof. Let

$$x(t) = x_k + ts_k, \quad t \in [0, 1].$$

we have

$$f_{k+1} - f_k = \int_0^1 g(x_k + ts_k)^\top s_k dt.$$

Thus,

$$\begin{aligned}\phi_k^H &= -4\int_0^1 g(x_k + ts_k)^\top s_k dt + 2(g_{k+1} + g_k)^\top s_k \\ &= 2\int_0^1 \left[g_k + g_{k+1} - 2g(x_k + ts_k)\right]^\top s_k dt.\end{aligned}$$

Hence, by the Cauchy–Schwarz inequality and Lemma 4.2, we obtain

$$\begin{aligned}|\phi_k^H| &\leq 2\int_0^1 \|g_k + g_{k+1} - 2g(x_k + ts_k)\| \, \|s_k\| dt \\ &\leq 2\int_0^1 (\|g_k - g(x_k + ts_k)\| + \|g_{k+1} - g(x_k + ts_k)\|) \, \|s_k\| dt \\ &\leq 2\int_0^1 (Lt\|s_k\| + L(1-t)\|s_k\|) \, \|s_k\| dt \\ &= 2L\|s_k\|^2.\end{aligned}$$

Moreover, Lemma 4.2 gives

$$\|y_k\| = \|g_{k+1} - g_k\| \leq L\|s_k\|.$$

Therefore,

$$\|\widetilde{y}_k\| = \left\| y_k + \frac{\phi_k^H}{\|s_k\|^2} s_k \right\| \leq \|y_k\| + \frac{|\phi_k^H|}{\|s_k\|} \leq L\|s_k\| + 2L\|s_k\| = 3L\|s_k\|,$$

and consequently,

$$\|\widetilde{y}_k\|^2 \leq 9L^2\|s_k\|^2.$$

Furthermore, together with the $\|g_k\| \geq \varepsilon$, for the curvature pairs accepted by the cautious criterion, we have

$$\frac{s_k^\top \widetilde{y}_k}{\|s_k\|^2} \geq \varepsilon_c \|g_k\|^\mu \geq \varepsilon_c \varepsilon^\mu.$$

Therefore

$$\frac{\|\widetilde{y}_k\|^2}{s_k^\top \widetilde{y}_k} \leq \frac{9L^2\|s_k\|^2}{\varepsilon_c \varepsilon^\mu \|s_k\|^2} = \frac{9L^2}{\varepsilon_c \varepsilon^\mu} \triangleq M_1.$$

For Algorithm 1, a candidate modified curvature pair $(s_k, \widetilde{y}_k)$ is stored in the limited-memory set only when it satisfies the cautious update criterion. Hence, all curvature pairs in $\mathcal{M}_{k+1}$ are accepted pairs and satisfy the prescribed curvature condition. If $\mathcal{M}_{k+1}$ is empty, then $B_{k+1} = B_{k+1}^{(0)}$. Therefore, in the following lemma, we only consider the nonempty limited-memory case, namely $m_{k+1} = |\mathcal{M}_{k+1}| \geq 1$.

Lemma 4.4. *Let $\{x_k\}$ be generated by Algorithm 1. Suppose that $\|g_k\| \geq \varepsilon$ for all k, where $\varepsilon > 0$ is a constant. Then there exists a constant $Q_1 > 0$ such that*

$$\operatorname{tr}(B_{k+1}) \leq m_{k+1} Q_1, \tag{23}$$

and

$$\sum_{j=1}^{m_{k+1}} \frac{\left\| B_{k+1}^{(j-1)} s_{i_j} \right\|^2}{s_{i_j}^\top B_{k+1}^{(j-1)} s_{i_j}} \leq m_{k+1} Q_1. \tag{24}$$

Proof. Taking the trace of (20) gives

$$\begin{aligned} \operatorname{tr}\left(B_{k+1}^{(j)}\right) &= \operatorname{tr}\left(B_{k+1}^{(j-1)}\right) - \frac{\operatorname{tr}\left(B_{k+1}^{(j-1)} s_{i_j} s_{i_j}^\top B_{k+1}^{(j-1)}\right)}{s_{i_j}^\top B_{k+1}^{(j-1)} s_{i_j}} + \frac{\operatorname{tr}\left(\widetilde{y}_{i_j} \widetilde{y}_{i_j}^\top\right)}{s_{i_j}^\top \widetilde{y}_{i_j}} \\ &= \operatorname{tr}\left(B_{k+1}^{(j-1)}\right) - \frac{\left\| B_{k+1}^{(j-1)} s_{i_j} \right\|^2}{s_{i_j}^\top B_{k+1}^{(j-1)} s_{i_j}} + \frac{\|\widetilde{y}_{i_j}\|^2}{s_{i_j}^\top \widetilde{y}_{i_j}}. \end{aligned}$$

Summing the above equality over $j = 1, \cdots, m_{k+1}$, we obtain

$$\operatorname{tr}(B_{k+1}) = \operatorname{tr}\left(B_{k+1}^{(0)}\right) - \sum_{j=1}^{m_{k+1}} \frac{\left\| B_{k+1}^{(j-1)} s_{i_j} \right\|^2}{s_{i_j}^\top B_{k+1}^{(j-1)} s_{i_j}} + \sum_{j=1}^{m_{k+1}} \frac{\|\widetilde{y}_{i_j}\|^2}{s_{i_j}^\top \widetilde{y}_{i_j}}.$$

By the boundedness of $B_{k+1}^{(0)}$, there exists a constant $U_B > 0$ such that

$$\operatorname{tr}\left(B_{k+1}^{(0)}\right) \leq U_B.$$

Moreover, by Lemma 4.3, for all accepted curvature pairs, it holds that

$$\frac{\|\widetilde{y}_{i_j}\|^2}{s_{i_j}^\top \widetilde{y}_{i_j}} \leq M_1.$$

Therefore,

$$\operatorname{tr}(B_{k+1}) \leq U_B - \sum_{j=1}^{m_{k+1}} \frac{\left\| B_{k+1}^{(j-1)} s_{i_j} \right\|^2}{s_{i_j}^\top B_{k+1}^{(j-1)} s_{i_j}} + m_{k+1} M_1.$$

Removing the non-positive summation term on the right-hand side yields

$$\operatorname{tr}(B_{k+1}) \leq U_B + m_{k+1} M_1.$$

Since B_{k+1} is a positive definite matrix, we have

$$\operatorname{tr}(B_{k+1}) > 0.$$

It follows from the trace identity above that

$$\sum_{j=1}^{m_{k+1}} \frac{\left\|B_{k+1}^{(j-1)} s_{i_j}\right\|^2}{s_{i_j}^\top B_{k+1}^{(j-1)} s_{i_j}} = \operatorname{tr}\left(B_{k+1}^{(0)}\right) + \sum_{j=1}^{m_{k+1}} \frac{\|\widetilde{y}_{i_j}\|^2}{s_{i_j}^\top \widetilde{y}_{i_j}} - \operatorname{tr}(B_{k+1}) \le U_B + m_{k+1} M_1.$$

Let $Q_1 \triangleq U_B + M_1$. Since $m_{k+1} \ge 1$, we have

$$U_B + m_{k+1} M_1 \le m_{k+1}(U_B + M_1) = m_{k+1} Q_1.$$

This proves (23) and (24).

Lemma 4.5. *Let $\{x_k\}$ be generated by Algorithm 1. If $\|g_k\| \ge \varepsilon$ holds for all k with some constant $\varepsilon > 0$, then there exists a constant $\xi_1 > 0$ such that*

$$\cos\varphi_k \triangleq \frac{s_k^\top B_k s_k}{\|s_k\|\|B_k s_k\|} = -\frac{g_k^\top s_k}{\|g_k\|\|s_k\|} \ge \xi_1. \tag{25}$$

Proof. At the beginning of the kth iteration, B_k is generated from the limited-memory set $\mathcal{M}_k$. Let

$$\mathcal{M}_k = \left\{(s_{i_1}, \widetilde{y}_{i_1}), (s_{i_2}, \widetilde{y}_{i_2}), \cdots, (s_{i_{m_k}}, \widetilde{y}_{i_{m_k}})\right\}, \quad 0 \le m_k \le \widetilde{m}.$$

By the trace estimate in Lemma 4.4 and the boundedness of the initial matrices, there exists a constant $C_1 > 0$ such that

$$\lambda_{\max}\left(B_k^{(j)}\right) \le \operatorname{tr}\left(B_k^{(j)}\right) \le C_1, \quad j = 0, 1, \ldots, m_k.$$

In particular,

$$\lambda_{\max}(B_k) \le C_1.$$

Using the determinant formula, we have

$$\det(B_k) = \det\left(B_k^{(0)}\right) \prod_{j=1}^{m_k} \frac{s_{i_j}^\top \widetilde{y}_{i_j}}{s_{i_j}^\top B_k^{(j-1)} s_{i_j}},$$

Since $B_k^{(0)}$ and $\left(B_k^{(0)}\right)^{-1}$ are bounded and $B_k^{(0)}$ is a symmetric positive definite matrix, there exists $d_0 > 0$ such that

$$\det\left(B_k^{(0)}\right) \ge d_0.$$

Moreover, all pairs in $\mathcal{M}_k$ satisfy the cautious update criterion, we have

$$s_{i_j}^\top \widetilde{y}_{i_j} \ge \varepsilon_c \|g_{i_j}\|^\mu \|s_{i_j}\|^2 \ge \varepsilon_c \varepsilon^\mu \|s_{i_j}\|^2.$$

Together with

$$s_{i_j}^\top B_k^{(j-1)} s_{i_j} \le C_1 \|s_{i_j}\|^2,$$

this gives

$$\frac{s_{i_j}^\top \widetilde{y}_{i_j}}{s_{i_j}^\top B_k^{(j-1)} s_{i_j}} \ge \frac{\varepsilon_c \varepsilon^\mu}{C_1}.$$

Since $m_k \le \widetilde{m}$, there exists a constant $d_1 > 0$ such that $\det(B_k) \ge d_1$. Let the eigenvalues of B_k be $0 < \lambda_1^{(k)} \le \lambda_2^{(k)} \le \cdots \le \lambda_n^{(k)}$. Since $\lambda_i^{(k)} \le C_1$ and

$$\prod_{i=1}^{n} \lambda_i^{(k)} = \det(B_k) \ge d_1,$$

we obtain

$$\lambda_{\min}(B_k) \ge \frac{d_1}{C_1^{n-1}} \triangleq \lambda_* > 0.$$

Therefore,

$$\lambda_* I \le B_k \le C_1 I.$$

Consequently,

$$s_k^\top B_k s_k \ge \lambda_* \|s_k\|^2, \quad \|B_k s_k\| \le C_1 \|s_k\|.$$

It follows that

$$\frac{s_k^\top B_k s_k}{\|s_k\|\|B_k s_k\|} \ge \frac{\lambda_*}{C_1} \triangleq \xi_1 > 0.$$

Finally, since $B_k d_k = -g_k$ and $s_k = \alpha_k d_k$, we have

$$\frac{s_k^\top B_k s_k}{\|s_k\|\|B_k s_k\|} = -\frac{g_k^\top s_k}{\|g_k\|\|s_k\|}.$$

This proves (25).

Lemma 4.6. *Let $\{x_k\}$ be generated by Algorithm 1. If $\|g_k\| \ge \varepsilon$ holds for all k for some constant $\varepsilon > 0$, then*

$$f_{k+1} \le f_k - \beta_0 \|g_k\|^2 \cos^2 \varphi_k, \tag{26}$$

where $\beta_0 = \frac{c_1(1-c_2)}{L} > 0$.

Proof. From (19), we have

$$g_k^\top d_k = -g_k^\top H_k g_k < 0.$$

By the Wolfe curvature condition, it follows that

$$[g(x_k + \alpha_k d_k) - g_k]^\top d_k \ge (c_2 - 1) g_k^\top d_k.$$

Moreover, by Lemma 4.2 and the Cauchy–Schwarz inequality, we obtain

$$\begin{aligned} [g(x_k + \alpha_k d_k) - g_k]^\top d_k &\le \left| [g(x_k + \alpha_k d_k) - g_k]^\top d_k \right| \\ &\le \|g(x_k + \alpha_k d_k) - g_k\| \|d_k\| \\ &\le L \alpha_k \|d_k\|^2. \end{aligned}$$

Since $g_k^\top d_k < 0$ and $0 < c_2 < 1$, we have

$$L \alpha_k \|d_k\|^2 \ge (1 - c_2) |g_k^\top d_k|.$$

and hence

$$\alpha_k \geq \frac{1-c_2}{L}\frac{|g_k^\top d_k|}{\|d_k\|^2}.$$

By the Wolfe sufficient decrease condition, we have

$$f_{k+1} \leq f_k + c_1\alpha_k g_k^\top d_k = f_k - c_1\alpha_k|g_k^\top d_k|.$$

Substituting the above lower bound on the step size into this inequality gives

$$f_{k+1} \leq f_k - \frac{c_1(1-c_2)}{L}\frac{(g_k^\top d_k)^2}{\|d_k\|^2}.$$

Furthermore, since $s_k = \alpha_k d_k$ and $\alpha_k > 0$, we have

$$\cos\varphi_k = -\frac{g_k^\top s_k}{\|g_k\|\|s_k\|} = -\frac{g_k^\top d_k}{\|g_k\|\|d_k\|}.$$

Therefore,

$$\frac{(g_k^\top d_k)^2}{\|d_k\|^2} = \|g_k\|^2\cos^2\varphi_k.$$

Setting $\beta_0 \triangleq \frac{c_1(1-c_2)}{L}$, we obtain (26).

Theorem 4.1. *Let $\{x_k\}$ be generated by Algorithm 1. Then*

$$\liminf_{k\to\infty}\|g_k\| = 0.$$

Proof. Suppose the conclusion does not hold, then there exist a constant $\varepsilon > 0$ and an integer k_0 such that

$$\|g_k\| \geq \varepsilon, \quad \forall k \geq k_0.$$

By Lemma 4.5, there exists a constant $\xi_1 > 0$ such that for all sufficiently large k, we obtain

$$\cos\varphi_k \geq \xi_1.$$

From Lemma 4.6, we have

$$f_{k+1} \leq f_k - \beta_0\|g_k\|^2\cos^2\varphi_k \leq f_k - \beta_0\varepsilon^2\xi_1^2.$$

Let $\delta_1 \triangleq \beta_0\varepsilon^2\xi_1^2 > 0$, then it follows that

$$f_{k+1} \leq f_k - \delta_1,$$

which implies that $f_k \to -\infty$. This contradicts

$$f(x) = \frac{1}{2}\|F(x)\|^2 \geq 0.$$

Therefore, the supposition is false, and hence

$$\liminf_{k\to\infty}\|g_k\| = 0.$$

Remark 4.1. *Theorem 4.1 only guarantees* $\liminf_{k\to\infty}\|g_k\| = 0$*, which does not in general imply* $\liminf_{k\to\infty}\|F(x_k)\| = 0$*. If $J(x^*)$ is nonsingular at a stationary point x^*, then $g(x^*) = 0$ implies $F(x^*) = 0$.*

5. Numerical experiments

This section evaluates the stability and computational performance of the proposed ML-BFGS method for solving GSTEs (3). All experiments were conducted on a Windows 11 system with an Intel(R) Core(TM) i5-10400F CPU @ 2.90 GHz. The software platform was MATLAB R2022b, and the Tensor Toolbox v3.5 developed by Kolda et al. [34] was used for implementation. The gradient-based stopping test in Step 3 of Algorithm 1 is used to detect stationarity of the least-squares problem (6). Since $g(x) = J(x)^\top F(x)$, a small gradient norm does not necessarily imply a small residual when $J(x)$ is singular. Therefore, a run is considered successful only when $\|F(x_k)\| \le 10^{-6}$. If this tolerance is not reached within 3600 s, the run is marked by '–', indicating failure. In the numerical computations, the initial matrix is chosen as $H_{k+1}^{(0)} = \gamma_k I$. Specifically, we first compute two BB-type scaling parameters

$$\gamma_k^{BB1} = \frac{s_k^\top s_k}{s_k^\top \widetilde{y}_k}, \quad \gamma_k^{BB2} = \frac{s_k^\top \widetilde{y}_k}{\widetilde{y}_k^\top \widetilde{y}_k}.$$

To balance efficiency and stability, a convex combination is employed

$$\bar{\gamma}_k = \omega \gamma_k^{BB1} + (1-\omega)\gamma_k^{BB2}, \quad 0 \le \omega \le 1.$$

The final scaling parameter is obtained by

$$\gamma_k = \min\left\{\gamma_{\max}, \max\{\gamma_{\min}, \bar{\gamma}_k\}\right\}, \quad 0 < \gamma_{\min} < \gamma_{\max}.$$

If the candidate curvature pair is rejected, the previous scaling parameter is retained. The common parameters are set as follows: $\tilde{m} = 15$, $c_1 = 10^{-4}$, $c_2 = 0.9$, $\gamma_0 = 1$, $\gamma_{\min} = 10^{-12}$, $\gamma_{\max} = 10^{12}$, $\omega = 0.5$, $\varepsilon_c = 10^{-6}$, $\mu = 1$, $\eta_c = 0.2$, $\beta_{\max} = 10^4$. The algorithms were tested on leading tensor coefficient problems of varying orders and dimensions (m, n), and their performance was compared based on the number of iterations (Ite), the computational error (Error), and the CPU time in seconds.

To verify the effects of the cautious update mechanism and the candidate modified curvature vector, this subsection compares Algorithm 1 with the classical L-BFGS method, cautious L-BFGS method and the modified secant L-BFGS method.

- Classical L-BFGS **(L-BFGS)**: This method uses the standard curvature vector $y_k = g_{k+1} - g_k$. Hence it serves as a baseline method for evaluating the improvement of the proposed modification over the standard curvature update.
- Cautious L-BFGS **(CAL-BFGS)**: This method uses the standard curvature vector $y_k = g_{k+1} - g_k$. The curvature pair (s_k, y_k) is stored in the limited-memory set only if $\frac{s_k^\top y_k}{\|s_k\|^2} \ge \varepsilon_c \|g_k\|^\mu$. Otherwise, the current curvature update is skipped. This method is used to examine the individual effect of the cautious update criterion.
- Modified secant L-BFGS **(MSL-BFGS)**: This method uses the modified curvature vector

$$y_k^C = \beta_k y_k, \quad \beta_k = \min\left\{\beta_{\max}, \max\left\{\eta_c, 1 + \frac{\phi_k^H}{s_k^\top y_k}\right\}\right\}.$$

  It constructs the modified curvature factor β_k from the modified curvature quantity ϕ_k^H, without introducing the cautious update criterion. Therefore, it is used to assess the effect of the modified curvature factor.

- Algorithm 1 **(ML-BFGS)**: This method combines the candidate modified curvature vector

$$\widetilde{y}_k = y_k + \frac{\phi_k^H}{\|s_k\|^2} s_k$$

with the cautious update criterion. It is used to evaluate the numerical performance of combining the modified curvature quantity ϕ_k^H with the cautious update criterion.

The two test problems below are constructed for the present numerical study following numerical testing practices used in studies of tensor equations [9, 16]. An exact solution $\tilde{x}$ is first prescribed, and the right-hand side b is then defined as

$$b = \sum_{p=2}^{m} \mathcal{A}_p \tilde{x}^{p-1},$$

which ensures that $\tilde{x}$ is an exact solution. This construction guarantees the existence of a known solution but not its uniqueness. Since uniqueness for general GTEs requires additional structural assumptions [16], the present experiments focus on the numerical computation of a solution. To control the numerical magnitude in large-dimensional experiments, we set

$$\tilde{x}_i = \frac{2i}{n(n+1)}, \quad i \in [n].$$

For all experiments, the initial point is chosen as $x_0 = (1, 1, \ldots, 1)^\top \in \mathbb{R}^n$. Unless otherwise specified, all compared methods use the same initial point, stopping criterion, Wolfe line search, and truncated BB-type initial scaling strategy. To provide an overall comparison of the methods, we use the performance profiles proposed by Dolan and Moré [35]. Let P denote the set of test problems for which at least one algorithm reaches the prescribed accuracy, and let S denote the set of compared algorithms. For each $p \in P$ and $s \in S$, let $t_{p,s}$ denote the computational resource, measured by either the number of iterations or CPU time, required by algorithm s to solve problem p. The performance ratio is defined by

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,q} : q \in S\}}.$$

If algorithm s fails to solve problem p, we set $r_{p,s} = r_M$, where r_M is a sufficiently large constant exceeding all finite performance ratios. The performance profile of algorithm s is then defined by

$$\rho_s(\tau) = \frac{1}{|P|}|\{p \in P : r_{p,s} \le \tau\}|, \quad \tau \ge 1.$$

We now present the first test problem.

Example 1. Let $\mathcal{I}_p$ denote the pth-order n-dimensional identity tensor, and let $\mathcal{E}_p$ denote the pth-order n-dimensional tensor with all elements equal to one. For each $p = 2, 3, \ldots, m$, define

$$\mathcal{A}_p = \mathcal{I}_p + \alpha_p \mathcal{E}_p, \quad \alpha_p = \frac{1}{p-1}.$$

Equivalently,

$$a^{(p)}_{i_1 i_2 \cdots i_p} = \begin{cases} 1 + \alpha_p, & \text{if } i_1 = i_2 = \cdots = i_p, \\ \alpha_p, & \text{otherwise.} \end{cases}$$

Since $\mathcal{A}_p$ is symmetric with respect to all indices, it is also a semi-symmetric tensor. Moreover, all off-diagonal entries are positive, yielding a positive coupling structure. This example is used to examine the convergence stability and computational efficiency of the methods under positive coupling.

The numerical results obtained by the four methods for Example 1 are reported in Table 1.

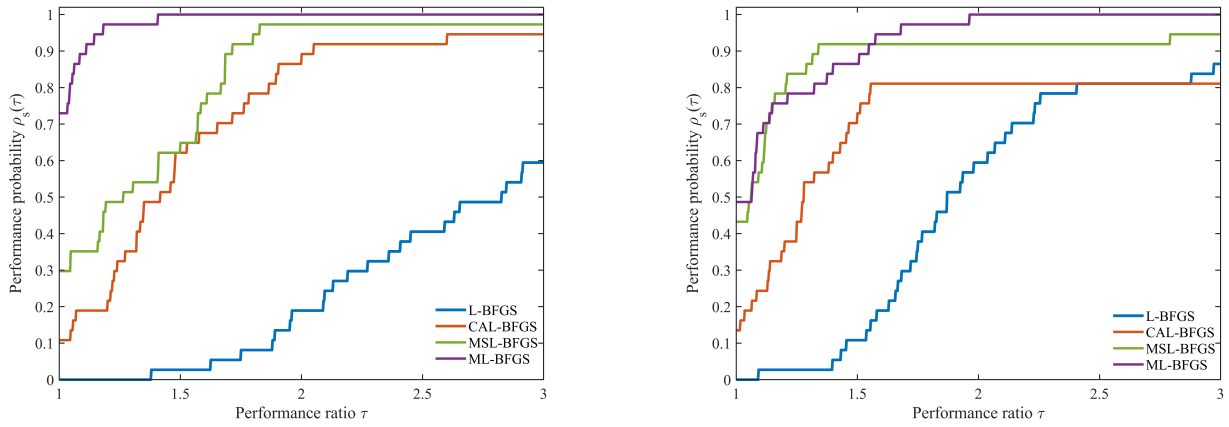
Table 1. Comparison of L-BFGS, CAL-BFGS, MSL-BFGS and ML-BFGS for Example 1.

(m,n)	L-BFGS			CAL-BFGS			MSL-BFGS			ML-BFGS		
	Ite	Error	CPU	Ite	Error	CPU	Ite	Error	CPU	Ite	Error	CPU
(3, 10)	-	-	-	-	-	-	-	-	-	16	9.25e-07	0.0836
(3, 20)	62	5.66e-07	0.0210	29	9.58e-07	0.0094	24	9.99e-07	0.0104	19	5.26e-07	0.0148
(3, 30)	49	6.14e-07	0.0052	27	6.84e-07	0.0042	20	3.70e-08	0.0027	20	3.13e-07	0.0053
(3, 40)	50	7.29e-07	0.0062	27	7.35e-07	0.0033	23	6.00e-07	0.0033	22	5.86e-07	0.0029
(3, 50)	50	7.93e-07	0.0071	28	3.81e-07	0.0059	19	5.63e-07	0.0047	20	9.54e-07	0.0039
(3, 60)	53	6.21e-07	0.0058	29	9.28e-07	0.0164	22	8.19e-07	0.0031	22	5.92e-07	0.0033
(3, 70)	46	7.66e-07	0.0058	29	5.83e-07	0.0048	23	3.49e-07	0.0040	22	5.03e-07	0.0031
(3, 80)	46	7.05e-07	0.0053	28	2.72e-07	0.0034	25	6.63e-07	0.0030	21	6.00e-07	0.0032
(3, 90)	34	2.84e-07	0.0043	19	9.27e-07	0.0032	18	7.14e-07	0.0025	20	6.31e-07	0.0035
(3, 100)	49	5.64e-07	0.0060	31	4.76e-07	0.0041	23	8.76e-07	0.0033	24	8.83e-07	0.0031
(3, 300)	40	4.17e-07	0.3513	31	7.92e-07	0.0297	29	2.35e-07	0.0045	29	1.13e-07	0.0040
(3, 500)	64276	1.61e-07	8.4375	36	1.18e-07	0.1497	30	1.12e-07	0.0173	31	6.63e-07	0.0062
(4, 10)	42	5.88e-07	0.0043	35	3.68e-07	0.0041	24	8.97e-07	0.0028	26	4.82e-07	0.0037
(4, 20)	65	6.93e-07	0.0104	34	2.76e-07	0.0042	30	4.72e-07	0.0039	23	4.80e-07	0.0035
(4, 30)	49	5.33e-07	0.0095	31	3.17e-07	0.0048	25	4.39e-07	0.0033	25	4.26e-07	0.0040
(4, 40)	61	3.22e-07	0.0078	36	4.26e-07	0.0050	32	7.94e-07	0.0046	19	8.63e-07	0.0035
(4, 50)	41	1.00e-07	0.0064	31	3.23e-07	0.0044	33	5.90e-07	0.0048	21	2.90e-07	0.0047
(4, 60)	52	8.70e-07	0.0074	39	8.78e-07	0.0055	32	3.35e-07	0.0044	34	5.54e-07	0.0050
(4, 70)	64	6.74e-07	0.0097	38	5.22e-07	0.0055	32	5.54e-07	0.0052	19	3.74e-07	0.0043
(4, 80)	61	2.78e-07	0.0084	41	2.34e-07	0.0061	36	5.07e-07	0.0054	23	8.07e-07	0.0060
(4, 90)	66	2.37e-07	0.0106	40	9.57e-07	0.0060	35	9.92e-07	0.0058	21	9.30e-07	0.0063
(4, 100)	64	9.63e-07	0.0099	41	7.10e-07	0.0070	36	7.37e-07	0.0058	20	7.12e-07	0.0050
(4, 300)	64	4.02e-07	0.0145	28	2.43e-07	0.0783	31	6.36e-07	0.0911	22	2.76e-07	0.0133
(4, 500)	-	-	-	-	-	-	-	-	-	-	-	-
(5, 10)	97	9.90e-07	0.0114	34	9.62e-07	0.0047	28	6.01e-07	0.0039	24	3.70e-07	0.0034
(5, 20)	71	7.48e-07	0.0091	41	9.23e-07	0.0055	31	5.09e-07	0.0046	22	3.66e-07	0.0044
(5, 30)	64	7.95e-07	0.0091	36	3.95e-07	0.0066	33	1.71e-07	0.0052	21	4.16e-07	0.0052
(5, 40)	74	5.55e-07	0.0106	41	5.90e-07	0.0076	39	7.57e-07	0.0064	26	4.79e-07	0.0088
(5, 50)	18335	1.00e-06	3.7082	23	4.00e-07	0.0059	32	8.78e-07	0.0061	19	4.39e-07	0.0058
(5, 60)	67	8.88e-07	0.0165	38	4.38e-07	0.0081	37	2.72e-07	0.0086	23	3.82e-07	0.0093
(5, 70)	57	7.35e-07	0.0106	23	2.71e-07	0.0065	26	4.21e-07	0.0087	22	6.76e-07	0.0065
(5, 80)	80	3.37e-07	0.0139	6708	1.00e-06	8.0044	42	8.27e-07	0.0088	23	4.36e-07	0.0095
(5, 90)	62	7.62e-07	0.0123	33	9.33e-07	0.0132	39	5.30e-07	0.0088	39	4.81e-07	0.0095
(5, 100)	64	6.62e-07	0.0136	37	5.94e-07	0.0101	36	7.81e-07	0.0095	21	7.16e-07	0.0103
(5, 300)	-	-	-	-	-	-	-	-	-	-	-	-
(6, 10)	59	7.42e-07	0.0075	65	9.99e-07	0.0162	29	5.96e-07	0.0048	25	8.30e-07	0.0043
(6, 20)	44	6.61e-07	0.0101	21	5.09e-07	0.0042	21	7.81e-07	0.0047	24	8.25e-07	0.0065
(6, 30)	81	5.65e-07	0.0120	24	3.53e-07	0.0078	38	2.49e-07	0.0072	25	8.28e-07	0.0121
(6, 40)	85	2.14e-07	0.0171	27	8.02e-07	0.0103	38	6.90e-07	0.0081	38	6.91e-07	0.0122
(6, 50)	-	-	-	-	-	-	-	-	-	-	-	-

Table 1 reports the numerical results for GSTEs (3) with positively coupled coefficient tensors. Except for the test case $(m,n) = (4, 500)$, $(m,n) = (5, 300)$, and $(m,n) = (6, 50)$, where none of the four methods reaches the prescribed accuracy within the specified time limit, the remaining test cases yield the following findings. Regarding the number of iterations, the proposed ML-BFGS method achieves the best performance on 23 test cases. The MSL-BFGS method performs best on 6 test cases, namely $(m,n) = (3, 50)$, $(3, 90)$, $(3, 100)$, $(3, 500)$, $(4, 10)$, and $(4, 60)$, while the CAL-BFGS method performs best on 3 test cases, namely $(m,n) = (5, 90)$, $(6, 30)$, and $(6, 40)$. In addition, MSL-BFGS and ML-BFGS show similar performance on $(m,n) = (3, 30)$, $(3, 60)$, $(4, 30)$ and $(3, 300)$ while CAL-BFGS and MSL-BFGS show similar performance on $(m,n) = (6, 20)$. Regarding the CPU time, MSL-BFGS achieves the

best performance on 15 test cases, ML-BFGS performs best on 16 test cases, and CAL-BFGS performs best on 4 test cases, namely $(m, n) = (3, 20)$, $(4, 50)$, $(5, 60)$, and $(6, 20)$. Moreover, MSL-BFGS and ML-BFGS show similar performance on $(m, n) = (5, 30)$, while CAL-BFGS and ML-BFGS show similar performance on $(5, 70)$. Overall, for GSTEs (3) with positively coupled coefficient tensors, the proposed ML-BFGS method exhibits a clear advantage in terms of the number of iterations. In terms of CPU time, MSL-BFGS and ML-BFGS show comparable overall performance, and both are superior to CAL-BFGS and L-BFGS.

Figure 1 shows the performance profile curves of L-BFGS, CAL-BFGS, MSL-BFGS, and ML-BFGS with respect to the number of iterations and CPU time for GSTEs (3) with positively coupled coefficient tensors as defined in Example 1. The performance profiles show that ML-BFGS has a clear advantage in terms of the number of iterations, achieving the highest probability over most performance ratios. In terms of CPU time, MSL-BFGS and ML-BFGS exhibit comparable performance profiles. In contrast, the L-BFGS curves generally lie below the others, especially for the number of iterations, indicating weaker performance. This behavior may stem from the use of the standard curvature vector y_k , which often fails to adequately capture local curvature variations. More importantly, the global convergence of L-BFGS can be established only under a uniformly convex assumption on the objective function—that is, its theoretical guarantee is inherently restricted to convex problems. By contrast, our ML-BFGS method not only exhibits strong adaptability to non-convex problems, but also achieves considerably better iteration efficiency while maintaining CPU-time performance comparable to that of MSL-BFGS. These results suggest that incorporating high-order curvature information together with a cautious update criterion can effectively improve iteration efficiency without introducing substantial additional computational cost.



(a) Iterations (b) CPU time

Figure 1. Performance profiles of Example 1.

Next, we present the second test problem.

Example 2. Let

$$q_i = -1 + \frac{2(i-1)}{n-1}, \quad i = 1, 2, \cdots, n.$$

For each $p = 2, 3, \ldots, m$, define $\mathcal{A}_p = \mathcal{I}_p + \lambda_p C_p$, here $\lambda_p = \frac{1}{p-1}$. And $C_p = (c^{(p)}_{i_1 i_2 \cdots i_p})$ is given by

$$c^{(p)}_{i_1 i_2 \cdots i_p} = 1 + \eta q_{i_1} q_{i_2} \cdots q_{i_p}, \quad \eta = 1.2.$$

Equivalently,

$$a^{(p)}_{i_1 i_2 \cdots i_p} = \begin{cases} 1 + \lambda_p c^{(p)}_{i_1 i_2 \cdots i_p}, & \text{if } i_1 = i_2 = \cdots = i_p, \\ \lambda_p c^{(p)}_{i_1 i_2 \cdots i_p}, & \text{otherwise.} \end{cases}$$

Since $c^{(p)}_{i_1 \cdots i_p}$ is invariant under any permutation of its indices, both C_p and $\mathcal{A}_p$ are symmetric, and hence semi-symmetric. Moreover, since $|q_i| \le 1$, we have $1 - \eta \le c^{(p)}_{i_1 \cdots i_p} \le 1 + \eta$. Thus, the coupling coefficients vary with the index tuple and can change sign, resulting in a nonuniform coupling structure. This example is used to evaluate the stability and robustness of the methods under nonuniform coupling.

The numerical results obtained by the four methods for Example 2 are reported in Table 2.

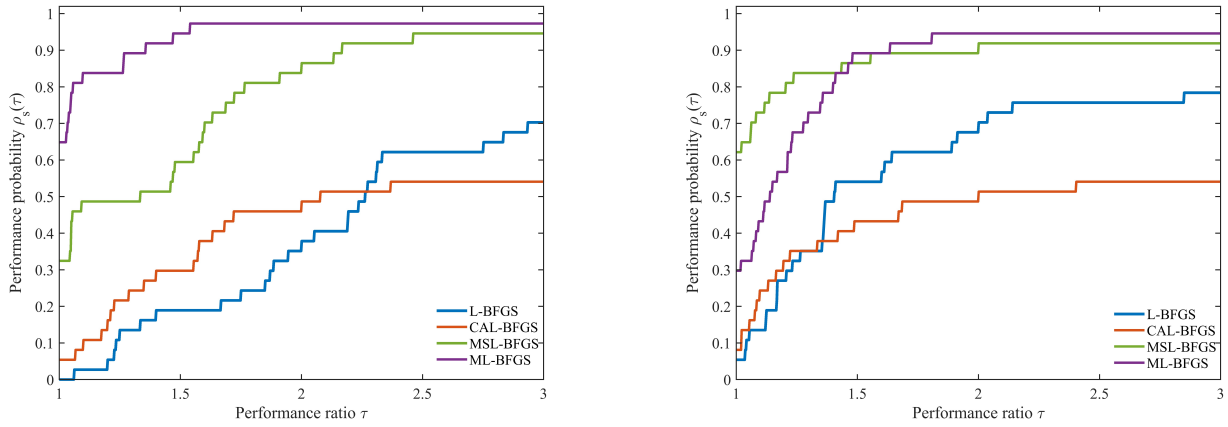
Table 2. Comparison of L-BFGS, CAL-BFGS, MSL-BFGS and ML-BFGS for Example 2.

(m,n)	L-BFGS			CAL-BFGS			MSL-BFGS			ML-BFGS		
	Ite	Error	CPU	Ite	Error	CPU	Ite	Error	CPU	Ite	Error	CPU
(3, 10)	95	9.89e-07	0.0400	28	6.40e-07	0.0543	28	5.44e-07	0.0325	18	8.10e-07	0.0531
(3, 20)	38	8.05e-07	0.0102	23	9.33e-07	0.0103	19	9.28e-07	0.0091	19	4.82e-07	0.0133
(3, 30)	35	8.29e-07	0.0042	27	9.71e-07	0.0037	20	5.30e-07	0.0031	21	9.82e-07	0.0056
(3, 40)	39	1.25e-07	0.0053	308	1.00e-06	0.0849	20	8.35e-07	0.0026	19	5.53e-07	0.0029
(3, 50)	30	8.31e-07	0.0052	27	5.12e-07	0.0055	13	2.45e-09	0.0040	20	2.44e-07	0.0037
(3, 60)	997	4.28e-07	0.1107	27	2.70e-07	0.0044	22	2.07e-07	0.0031	21	1.99e-07	0.0038
(3, 70)	25	3.75e-07	0.0030	18208	1.00e-06	4.8477	15	4.10e-07	0.0022	19	7.09e-07	0.0031
(3, 80)	25	5.67e-07	0.0033	26948	1.00e-06	7.2079	140	2.56e-07	0.0207	20	6.78e-07	0.0033
(3, 90)	1618	9.99e-07	0.2024	7883	1.00e-06	2.2400	17	6.20e-07	0.0025	25	5.13e-07	0.0037
(3, 100)	28	5.42e-08	0.0035	17457	1.00e-06	4.9492	21	1.93e-07	0.0029	23	2.07e-07	0.0033
(3, 300)	24	1.91e-07	0.0056	24	1.81e-07	0.0059	21	2.10e-07	0.0035	20	2.17e-07	0.0049
(3, 500)	27	1.38e-07	0.0045	27	8.46e-08	0.0055	22	7.30e-07	0.0199	23	6.30e-07	0.0061
(4, 10)	43	7.06e-07	0.0051	31	7.18e-07	0.0036	19	9.02e-07	0.0027	24	7.11e-07	0.0035
(4, 20)	46	5.17e-07	0.0055	33	5.22e-07	0.0042	28	4.02e-07	0.0039	21	6.79e-07	0.0039
(4, 30)	35	3.93e-07	0.0048	43	8.27e-07	0.0076	25	9.82e-07	0.0038	26	8.57e-07	0.0046
(4, 40)	38	9.12e-07	0.0049	34	5.72e-07	0.0048	30	7.84e-07	0.0047	17	4.09e-07	0.0054
(4, 50)	178	1.00e-06	0.0217	45	6.22e-07	0.0125	31	4.55e-07	0.0052	19	2.18e-07	0.0070
(4, 60)	37	3.91e-07	0.0079	32	3.67e-07	0.0050	30	4.70e-07	0.0049	31	2.02e-07	0.0053
(4, 70)	37	2.33e-07	0.0088	39759	1.00e-06	18.3827	27	9.40e-07	0.0050	16	3.06e-08	0.0044
(4, 80)	35	1.91e-07	0.0063	53472	1.00e-06	26.4377	33	4.96e-07	0.0056	159	2.81e-08	0.0275
(4, 90)	35	1.10e-08	0.0059	24678	1.00e-06	12.7148	31	1.84e-07	0.0057	18	3.50e-07	0.0061
(4, 100)	35	8.26e-07	0.0067	53023	1.00e-06	29.9371	32	3.25e-08	0.0059	15	3.90e-07	0.0049
(4, 300)	44	2.86e-07	0.0135	21	3.08e-07	0.0128	24	5.96e-07	0.0199	15	2.03e-07	0.0397
(4, 500)	-	-	-	-	-	-	-	-	-	-	-	-
(5, 10)	77	7.42e-07	0.0094	906	9.99e-07	0.2841	24	7.32e-07	0.0035	22	8.88e-07	0.0033
(5, 20)	888	4.90e-08	0.1213	232	1.00e-06	0.0826	31	7.93e-07	0.0046	21	3.75e-07	0.0045
(5, 30)	1502	9.99e-07	0.2059	37	5.89e-07	0.0059	35	8.56e-07	0.0056	22	5.04e-07	0.0057
(5, 40)	57	5.49e-07	0.0084	41	2.99e-07	0.0079	41	2.73e-07	0.0072	26	2.27e-07	0.0092
(5, 50)	51	7.14e-07	0.0087	18	5.67e-07	0.0053	36	6.28e-08	0.0076	19	1.25e-07	0.0062
(5, 60)	57	4.86e-07	0.0106	44826	1.00e-06	32.9928	39	9.79e-07	0.0078	18	3.81e-08	0.0096
(5, 70)	352	9.97e-07	0.0873	31162	1.00e-06	25.0913	25	5.14e-07	0.0071	17	1.15e-07	0.0067
(5, 80)	50	8.08e-07	0.0110	534490	1.00e-06	534.6083	42	6.40e-07	0.0094	22	4.13e-07	0.0114
(5, 90)	262	7.72e-07	0.0538	28	7.82e-07	0.0131	292	9.69e-07	0.0953	38	6.57e-07	0.0121
(5, 100)	47	7.49e-07	0.0111	26616	1.00e-06	28.2751	32	1.91e-07	0.0095	13	9.83e-07	0.0101
(5, 300)	-	-	-	-	-	-	-	-	-	-	-	-
(6, 10)	43	5.34e-07	0.0092	27	9.95e-07	0.0050	24	3.78e-07	0.0048	23	2.39e-07	0.0043
(6, 20)	37	9.11e-07	0.0151	22	4.29e-07	0.0044	21	3.46e-07	0.0088	20	3.58e-07	0.0048
(6, 30)	66	4.63e-07	0.0109	345	1.00e-06	0.2161	35	8.90e-07	0.0099	24	8.49e-07	0.0080
(6, 40)	66	9.88e-07	0.0155	27792	1.00e-06	21.5442	35	6.64e-07	0.0081	36	1.24e-07	0.0090
(6, 50)	-	-	-	-	-	-	-	-	-	-	-	-

For GSTEs (3) with coefficient tensors under nonuniform coupling, none of the four methods reaches the prescribed accuracy within the specified time limit for $(m, n) = (4, 500), (5, 300)$ and $(6, 50)$. For the remaining test cases, the following observations can be made. Regarding the number of iterations,

the proposed ML-BFGS method achieves the best performance on 23 test cases. The MSL-BFGS method performs best on 11 test cases, while the CAL-BFGS method performs best on 2 test cases, namely $(m, n) = (5, 50)$ and $(5, 90)$. In addition, MSL-BFGS and ML-BFGS show similar performance on $(m, n) = (3, 20)$. Regarding the CPU time, MSL-BFGS achieves the best performance on 22 test cases, ML-BFGS performs best on 9 test cases, CAL-BFGS performs best on 3 test cases, and L-BFGS performs best on one test case. Moreover, L-BFGS and ML-BFGS show similar performance on $(m, n) = (3, 80)$, MSL-BFGS and ML-BFGS show similar performance on $(m, n) = (4, 20)$. Overall, for GSTEs (3) with coefficient tensors under nonuniform coupling, the proposed ML-BFGS method shows a clear advantage in terms of the number of iterations, while its CPU time performance remains generally comparable to that of MSL-BFGS.

Figure 2 shows the performance profiles of L-BFGS, CAL-BFGS, MSL-BFGS, and ML-BFGS in terms of iterations and CPU time for GSTEs (3) with nonuniformly coupled coefficient tensors. The performance profiles show that ML-BFGS has a clear advantage in iteration efficiency, while MSL-BFGS performs better in CPU time at small performance ratios. As the performance ratio increases, ML-BFGS remains competitive in CPU time.



(a) Iterations (b) CPU time

Figure 2. Performance profiles of Example 2.

6. Conclusion

This paper has proposed a ML-BFGS method with a cautious update for the GSTEs (3). The method incorporates high-order curvature information into a candidate modified curvature vector and combines it with a cautious update criterion within the limited-memory BFGS framework. Under suitable assumptions, the global convergence of the proposed method with a Wolfe line search is established. Numerical experiments on semi-symmetric tensor test problems with positive and nonuniform coupling show that the proposed method exhibits good stability and competitive performance. Future work will consider validation on application-driven GSTEs (3) and realistic datasets, theoretical conditions ensuring solution uniqueness, and a deeper exploitation of the multilinear structure of semi-symmetric tensors to further improve computational efficiency.

Author contributions

Yuncheng Xu: Methodology, Writing – original draft, Visualization. Sanyang Liu: Supervision, Funding acquisition. Huiping Cao: Writing – review & editing, Funding acquisition.

Use of Generative-AI tools declaration

The author(s) declare(s) they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 12271419), the Natural Science Basic Research Plan in Shaanxi Province of China (Grant No. 2025JC-YBMS-079).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. C. Ling, H. He, L. Qi, On the cone eigenvalue complementarity problem for higher-order tensors, *Comput. Optim. Appl.*, **63** (2016), 143–168. https://doi.org/10.1007/s10589-015-9767-z
2. S. Hu, A note on the solvability of a tensor equation, *J. Ind. Manag. Optim.*, **18** (2022), 4043–4047. https://doi.org/10.3934/jimo.2021146
3. D. Liu, W. Li, M. K. Ng, S. W. Vong, Uniqueness and perturbation bounds for sparse non-negative tensor equations, *Front. Math. China*, **13** (2018), 849–874. https://doi.org/10.1007/s11464-018-0707-y
4. D. H. Li, S. Xie, H. R. Xu, Splitting methods for tensor equations, *Numer. Linear Algebra Appl.*, **24** (2017), e2102. https://doi.org/10.1002/nla.2102
5. Y. Tang, Q. Yang, Two heuristics solving low tensor train rank tensor completion, *J. Ind. Manag. Optim.*, **21** (2025), 925–954. https://doi.org/10.3934/jimo.2024111
6. W. Ding, Y. Wei, Solving multi-linear systems with M-tensors, *J. Sci. Comput.*, **68** (2016), 689–715. https://doi.org/10.1007/s10915-015-0156-7
7. Z. Luo, L. Qi, N. Xiu, The sparsest solutions to Z-tensor complementarity problems, *Optim. Lett.*, **11** (2017), 471–482. https://doi.org/10.1007/s11590-016-1013-9
8. W. Li, M. K. Ng, On the limiting probability distribution of a transition probability tensor, *Linear Multilinear Algebra*, **62** (2014), 362–385. https://doi.org/10.1080/03081087.2013.777436
9. C. Q. Lv, C. F. Ma, A Levenberg-Marquardt method for solving semi-symmetric tensor equations, *J. Comput. Appl. Math.*, **332** (2018), 13–25. https://doi.org/10.1016/j.cam.2017.10.005

10. Z. Li, Y. H. Dai, H. Gao, Alternating projection method for a class of tensor equations, *J. Comput. Appl. Math.*, **346** (2019), 490–504. https://doi.org/10.1016/j.cam.2018.07.013

11. M. Najafi Kalyani, M. Nobakht Kooshkghazi, L. Reichel, Block iterative methods for solving multilinear systems, *Numer. Algorithms*, **100** (2025), 803–829. https://doi.org/10.1007/s11075-025-02077-x

12. X. Wang, M. Che, Y. Wei, Neural networks based approach solving multilinear systems with M-tensors, *Neurocomputing*, **351** (2019), 33–42. https://doi.org/10.1016/j.neucom.2019.03.025

13. X. Wang, C. Mo, S. Qiao, Y. Wei, Predefined-time convergent neural networks for solving the time-varying nonsingular multilinear tensor equations, *Neurocomputing*, **472** (2022), 68–84. https://doi.org/10.1016/j.neucom.2021.11.108

14. R. Zhao, M. Li, T. Liu, S. X. Miao, A noise-tolerant zeroing neural network with fixed-time convergence for solving multilinear systems with nonsingular M tensors, *AIMS Math.*, **10** (2025), 13974–13995. https://doi.org/10.3934/math.2025628

15. D. F. Gleich, L. H. Lim, Y. Yu, Multilinear PageRank, *SIAM J. Matrix Anal. Appl.*, **36** (2015), 1507–1541. https://doi.org/10.1137/140985160

16. W. Yan, C. Ling, L. Ling, H. He, Generalized tensor equations with leading structured tensors, *Appl. Math. Comput.*, **361** (2019), 311–324. https://doi.org/10.1016/j.amc.2019.05.042

17. S. Sharma, K. Palpandi, Some existence results for the generalized tensor absolute value equation, *Filomat*, **37** (2023), 4185–4194. https://doi.org/10.2298/FIL2313185S

18. Z. Jiang, J. Li, Slice tensor splitting method for solving tensor equation, *Appl. Math. Comput.*, **463** (2024), 128367. https://doi.org/10.1016/j.amc.2023.128367

19. J. Wang, Z. Li, Y. Ran, Y. Li, On greedy randomized kaczmarz-type methods for solving the system of tensor equations, *Appl. Math. Lett.*, **158** (2024), 109261. https://doi.org/10.1016/j.aml.2024.109261

20. S. Feng, Y. Wei, R. Xu, E. K. W. Chu, A linearized alternating direction method of multipliers for solving complex semi-symmetric tensor equations and its applications, *Indian J. Pure Appl. Math.*, **56** (2025), 930–959. https://doi.org/10.1007/s13226-025-00812-7

21. Y. Nie, S. Liang, N. Huang, Generalized SOR-type iterative method for solving generalized tensor absolute value equation, *Numer. Algebra Control Optim.*, **19** (2026), 168–188. https://doi.org/10.3934/naco.2026017

22. H. He, C. Ling, L. Qi, G. Zhou, A globally and quadratically convergent algorithm for solving multilinear systems with M-tensors, *J. Sci. Comput.*, **76** (2018), 1718–1741. https://doi.org/10.1007/s10915-018-0689-7

23. Y. Wang, C. Ouyang, L. Lv, G. Yuan, Analysis of a new BFGS algorithm and conjugate gradient algorithms and their applications in image restoration and machine learning, *Appl. Numer. Math.*, **210** (2025), 199–221. https://doi.org/10.1016/j.apnum.2024.12.015

24. Y. H. Dai, Convergence properties of the BFGS algorithm, *SIAM J. Optim.*, **13** (2002), 693–701. https://doi.org/10.1137/S1052623401383455

25. Y. X. Yuan, A modified BFGS algorithm for unconstrained optimization, *IMA J. Numer. Anal.*, **11** (1991), 325–332. https://doi.org/10.1093/imanum/11.3.325

26. D. H. Li, M. Fukushima, On the global convergence of the BFGS method for non-convex unconstrained optimization problems, *SIAM J. Optim.*, **11** (2001), 1054–1064. https://doi.org/10.1137/S1052623499354242

27. J. Nocedal, Updating quasi-Newton matrices with limited storage, *Math. Comp.*, **35** (1980), 773–782. https://doi.org/10.1090/S0025-5718-1980-0572855-7

28. D. C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Program.*, **45** (1989), 503–528. https://doi.org/10.1007/BF01589116

29. Y. H. Xiao, T. H. Li, Z. X. Wei, Global convergence of a modified limited memory BFGS method for non-convex minimization, *Acta Math. Appl. Sin. Engl. Ser.*, **29** (2013), 555–566. https://doi.org/10.1007/s10255-013-0233-3

30. F. Biglari, A. Ebadian, Limited-memory BFGS method based on a high-order tensor model, *Comput. Optim. Appl.*, **60** (2015), 413–422. https://doi.org/10.1007/s10589-014-9678-4

31. L. Qi, H. Chen, Y. Chen, *Tensor eigenvalues and their applications*, Singapore: Springer, 2018. https://doi.org/10.1007/978-981-10-8058-6

32. D. Liu, W. Li, S. W. Vong, The tensor splitting with application to solve multilinear systems, *J. Comput. Appl. Math.*, **330** (2018), 75–94. https://doi.org/10.1016/j.cam.2017.08.009

33. Y. Dai, N. Yamashita, Convergence analysis of sparse quasi-Newton updates with positive definite matrix completion for two-dimensional functions, *Numer. Algebra Control Optim.*, **1** (2011), 61–69. https://doi.org/10.3934/naco.2011.1.61

34. B. W. Bader, T. G. Kolda, Efficient MATLAB computations with sparse and factored tensors, *SIAM J. Sci. Comput.*, **30** (2008), 205–231. https://doi.org/10.1137/060676489

35. E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance profiles, *Math. Program.*, **91** (2002), 201–213. https://doi.org/10.1007/s101070100263

AIMS Press