A Non-Monotone Tensor Method for Unconstrained Optimization Problems

XIANJUN SHI
Tianjin University
School of Science
Department of Mathematics
Weijin Road 92, Tianjin
P.R. China.
shixianjun88@gmail.com

LEI YANG
Tianjin University
School of Science
Department of Mathematics
Weijin Road 92, Tianjin
P.R. China.
ylei@tju.edu.cn

YING ZHANG
Tianjin University
School of Science
Department of Mathematics
Weijin Road 92, Tianjin
P.R. China.
zhangyhit@yahoo.com.cn

Abstract: The tensor method for unconstrained optimization was first introduced by Schnable and Chow [SIAM Journal on Optimization, 1 (1991): 293–315], where each iteration bases upon a fourth order model for the objective function. In this paper, we propose a tensor method with a non-monotone line search scheme for solving the unconstrained optimization problem, and show the convergence of the method. We evaluate the proposed method by several numerical examples, and compare the obtained numerical results with those by the modified Newton method, the tensor method, and the monotone tensor method. Through the numerical results, we can see that the new method is more effective than others for the problems we tested.

Key–Words: unconstrained optimization, non-monotone linear search, non-monotone tensor method.

AMS subject classifications 90C30, 65K05.

1 Introduction

In this paper, we propose a method, called the non-monotone tensor method, for solving the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{for } f : \mathbb{R}^n \to \mathbb{R} \quad (1)$$

where we assume that $f$ is at least twice continuously differentiable and bounded below. One of the most popular methods for solving (1) is the iterative algorithm, where the sequence $\{x_k\}$ is generated by the iterative scheme

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

where $d_k$ is a direction in the $k$-th iteration and $\alpha_k$ is the corresponding step size. Then, in the $k$-th iteration of the algorithm, the key is to find the direction $d_k$ and the step size $\lambda_k$.

At first, one needs to find the iterative direction $d_k$. To achieve it, one of the traditional methods is the Newton method which employs the quadratic approximation of $f$. However, one defect of Newton method is that the Hessian matrix of objective function $f$ must be nonsingular. Because of this, the Newton method is largely restricted when applied to practical problems. To overcome this defect, as we know in [13], Schabel and Chow proposed a tensor method for unconstrained optimization in 1991. The tensor method improves the efficiency and accuracy of solving (1) compared with standard Newton method, especially, when $\nabla^2 f(x_*)$ is singular where $x_*$ stands for the optimal point. In order to avoid the high cost of expanding $f(x)$ to the third or fourth order Taylor expansion, they used the second derivatives to approximate the third and fourth order tensor, and to approach $f(x)$ by the fourth degree polynomial. Then, Chow, Eskow and Schnabel [6] worked out the homologous software package. According to the method proposed in [13], Bouaricha [2] applied it to large and sparse unconstrained optimizations by taking advantage of the sparse of Hessian matrix and some computational techniques. At the same time, he also published the corresponding software package in 1997 [1]. In addition, in 1984, Schnabel and Frank [14] first solved the system of nonlinear equations by taking advantage of tensor to approach function. In their experiments, their algorithms improved the efficiency and reliability over standard linear model obviously. Besides, they also proved that this tensor method has order 1.16 local convergence rate on some special situations. Similarly, when this idea was applied to solve the nonlinear least square problems [4, 3], the tested results indicated that tensor methods are significantly more efficient and robust than standard methods on small and medium-sized problems.
Secondly, one needs to choose a proper iterative step size \( \lambda_k \) by some line search scheme. The ideal line search rule is an exact one that satisfies:

\[
f(x_k + \lambda_k d_k) = \min_{\lambda \geq 0} f(x_k + \lambda d_k).
\]

In fact, the exact value of \( \lambda_k \) above is difficult or even impossible to reach in practice. So researchers focus on the inexact line search scheme, which generally includes monotone line search and non-monotone line search methods. In the monotone line search methods (such as the Goldstein, Armijo and Wolfe line search method), \( \lambda_k \) is chosen to keep \( f(x_k + \lambda_k d_k) < f(x_k) \) for all \( k \), which is possible to restrict the searching points in a creep along the bottom of a narrow curved valley. However, the non-monotone line search overcomes this obstacle to find a global optimum and improve the speed of convergence especially for the cases that the involving function is highly nonconvex and has a valley in a small neighborhood of some points.

The earliest non-monotone line search scheme was proposed by Grippo, Lampariello, and Lucidi for Newton methods [9]. Then, many non-monotone line search schemes were developed [7, 8, 15, 17]. Recently, Hu, Huang and Lu [10] proposed a non-monotone line search algorithm which is to find the step size \( \lambda_k \) such that

\[
\begin{align*}
f(x_k + \lambda_k d_k) & \leq C_k + \frac{\delta \lambda_k \nabla f(x_k)^T d_k}{2} + \nabla f(x_k + \lambda_k d_k)^T d_k \\
& \geq \sigma \nabla f(x_k)^T d_k.
\end{align*}
\]

Here, \( \delta, \sigma, m_k \) are parameters satisfying \( 0 < \delta < \sigma < 1, \ m_k \in \mathbb{N}^+ \) and \( C_k \) is chosen as

\[
Q_k = 1 + 
\eta_k \sum_{i=1}^{m_k-1} \eta_{k-i},
\]

\[
C_k = \frac{\eta_k \sum_{i=1}^{m_k-1} \eta_{k-i} f(x_{k-i}) + f(x_k)}{Q_k},
\]

where \( \eta_k \in [0, 1] \) is chosen at different conditions (Algorithm 3.1 for details) and \( \mathbb{N}^+ \) stands for the set of positive integers. But, they employed this line search scheme in a Newton method. Similar non-monotone line search schemes were also used in [11] for complementarity problems and in [16] for variational inequality problems.

In this paper, we propose a non-monotone tensor method for unconstrained optimization problem (1) by combining the non-monotone line search scheme above with the tensor method given in [13]. We show the convergence of the proposed method under mild assumptions. We evaluate the proposed method by several numerical examples, and compare the obtained numerical results with those by the modified Newton method, the tensor method, and the monotone tensor method. The preliminary numerical results demonstrate that the new method is more effective than others for the problems we tested.

The remainder of this paper is organized as follows. In Section 2, we will briefly introduce the concepts of tensor and its operations, and then review the techniques to propose the tensor model in [13] and solve it. In Section 3, we will propose the concrete algorithm of the non-monotone tensor method and discuss its global convergence. In Section 4, we will report the numerical results for the non-monotone tensor method, tensor method with monotone line search, tensor method in [13] and modified Newton method, respectively.

## 2 Tensor Model for Unconstrained Optimization

In this section, we first introduce some basic concepts and operations for tensor; and then, we introduce the tensor method used to solve the unconstrained optimization problem (1).

### Definition 1

If \( A = (A_{l_1 \ldots l_m}) \) satisfies \( A_{i_1 \ldots i_m} \in \mathbb{R} \), where \( i_j = 1, \ldots, n_j \) for \( j = 1, \ldots, m \), i.e., \( A \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_m} \), we call \( A \) an m-th order real tensor. Furthermore, if \( n_1 = n_2 = \cdots = n_m \), we call \( A \) an m-th order and n-dimensional square tensor.

From Definition 1, when \( m = 1 \) and \( m = 2 \), the tensor \( A \) reduces to a vector and a matrix, respectively. In addition, the \( m \)-th order tensor is also called a high-order matrix. We are about to give the specific definition of the third and fourth order square tensors.

### Definition 2

Suppose that \( T \in \mathbb{R}^{n \times n \times n} \), \( V \in \mathbb{R}^{n \times n \times n \times n} \) and \( \zeta, \mu, \nu, \omega \in \mathbb{R}^n \). Then,

\[
T \cdot \mu \nu \omega = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} T_{ijk} \mu_i \nu_j \omega_k;
\]

\[
(T \cdot \nu \omega)_i = \sum_{j=1}^{n} T_{ijk} \nu_j \omega_k, \quad i = 1, \ldots, n;
\]

\[
V \cdot \zeta \mu \nu \omega = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V_{ijkl} \zeta_i \mu_j \nu_k \omega_l;
\]

\[
(V \cdot \mu \nu \omega)_i = \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V_{ijkl} \mu_j \nu_k \omega_l, \quad i = 1, \ldots, n,
\]

and hence, \( T \cdot \mu \nu \omega \in \mathbb{R}, \ T \cdot \mu \nu \in \mathbb{R}^n, \ V \cdot \zeta \mu \nu \omega \in \mathbb{R} \), and \( V \cdot \mu \nu \omega \in \mathbb{R}^n \). If \( \zeta = \mu = \nu = \omega \) holds, we denote \( T \cdot \mu \nu \omega \) by \( T \cdot \mu^3 \) and \( V \cdot \zeta \mu \nu \omega \) by \( V \cdot \mu^4 \).
Suppose that means their corresponding elements add. Each ele-

\[ T = \sum_{i_1 \cdots i_m} A_{i_1i_2\cdots i_m} x^{i_1} x^{i_2} \cdots x^{i_m}, \]

where the indices \( j_1j_2 \cdots j_m \) is an arbitrary permuta-

\[ \text{Part I: The model of tensor method} \]

\[ T \text{ } \otimes \text{ } V = T_{i_1i_2 \cdots i_p} V_{j_1j_2 \cdots j_q} \]

Since the vector and matrix are special cases of tensor, the multiplication is also appropriate for them. There is a kind of tensor which is special and owns available properties called rank-one tensor. They are easy to be stored and operated with each other. The detailed definition is presented as:

\[ T = (A_{i_1i_2\cdots i_m} x^{i_1} x^{i_2} \cdots x^{i_m}) \]

where \( x_c \) represents the current iterative point, \( \nabla f(x_c) \) and \( \nabla^2 f(x_c) \) denote either these analytic derivatives, or finite difference approximations to them, and \( T \in \mathbb{R}^{n \times n \times n} \) and \( V \in \mathbb{R}^{n \times n \times n \times n} \) are symmetric rank-one tensors. The advantages of this model are provided in [13].

\[ \text{Part II: Solutions of } T \text{ and } V. \]

Actually, in the model (3), we don’t need the third or fourth order derivative of \( f \). It is just the fourth order expansion of \( f \) on the foundation of the second order Taylor expansion. What we need is finding proper tensors \( T \) and \( V \) to approach the value of \( f \) and \( \nabla f(x) \) at previous iterates. While solving \( T \) and \( V \), we re-

\[ \text{Define the unknown quantities } \alpha = (\alpha_k)_{1 \leq k \leq p}, \beta = (\beta_k)_{1 \leq k \leq p} \in \mathbb{R}^p \text{ as} \]

\[ \alpha_k = T \cdot s_k^3, \quad \beta_k = V \cdot s_k^4 \text{ for } k = 1, \ldots, p. \]

Then, from (4) and (5), we have the following systems of two linear equations in two unknowns for each of the \( p \) pairs \( \alpha_k \) and \( \beta_k \):

\[ \frac{1}{2} \alpha_k + \frac{1}{6} \beta_k = \nabla f(x_{-k}) \cdot s_k - \nabla f(x_c) \cdot s_k \]

\[ -\nabla^2 f(x_c) \cdot s_k^2 + \frac{1}{12} T \cdot s_k^3 + \frac{1}{24} V \cdot s_k^4 \]

\[ \frac{1}{6} \alpha_k + \frac{1}{24} \beta_k = f(x_{-k}) - f(x_c) \]

\[ -\nabla f(x_c) \cdot s_k + \frac{1}{6} \nabla^2 f(x_c) \cdot s_k^2 + \frac{1}{12} T \cdot s_k^3 + \frac{1}{24} V \cdot s_k^4 \]

\[ \text{for } k = 1, \ldots, p. \] The system (7) is nonsingular, so \( \alpha_k \) and \( \beta_k \) have unique solution. Combining (6) and (7), we know \( V \) and \( T \) satisfy

\[ V \cdot s_k^4 = \beta_k, \quad T \cdot s_k^2 = a_k, \quad k = 1, \ldots, p, \]
where
\[ a_k = 2(\nabla f(x_k) - \nabla f(x_c) - \nabla^2 f(x_c) \cdot s_k - \frac{1}{6} V \cdot s_k^3), \text{ for } k = 1, \ldots, p. \]

**Theorem 6** Define \( P \in \mathbb{R}^{p \times p} \) by \( P_{ij} = (s_i^T s_j)^{\frac{4}{3}}, 1 \leq i, j \leq p \) and define \( \tau \in \mathbb{R}^p \) by \( \tau = P^{-1} \beta \). Then, the solution to
\[
\min_{V \in \mathbb{R}^{n \times n \times n \times n}} \|V\|_F
\text{ subject to } V \cdot s_k^4 = \beta_k, k = 1, \ldots, p \\
\text{and } V \text{ is symmetric.}
\]
is
\[
V = \sum_{k=1}^{p} \tau_k (s_k \otimes s_k \otimes s_k \otimes s_k), \tag{10}
\]
where \( \otimes \) is defined in Definition 4.

**Theorem 7** Define \( a_k \in \mathbb{R}^n \) (\( k = 1, \ldots, p \)) as (9). The solution to
\[
\min_{V \in \mathbb{R}^{n \times n \times n \times n}} \|T\|_F
\text{ subject to } T \cdot s_i^4 = a_i, \ i = 1 \ldots p \\
\text{and } T \text{ is symmetric.}
\]
is
\[
T = \sum_{k=1}^{p} \left( b_k \otimes s_k \otimes s_k + s_k \otimes b_k \otimes s_k + \right. \\
\left. + s_k \otimes s_k \otimes b_k, \right) \tag{11}
\]
where \( b_k \in \mathbb{R}^n, k = 1, \ldots, p \), \( \{b_k\} \) is the unique sequence of vectors which makes (11) satisfy \( T \cdot s_i^4 = a_i \) (\( i = 1, \ldots, p \)) and \( \otimes \) is defined in Definition 4.

The proofs of Theorems 6 and 7 can be found in [13]. By these two theorems, we can get the concrete forms of \( T \) and \( V \).

**Part III: Solution of the tensor model:**

Substituting (10) and (11) into (3), the tensor model can be expressed as:
\[
f(x_c + d) = f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2} \nabla^2 f(x_c) \cdot d^2 \\
+ \frac{1}{2} \sum_{k=1}^{p} (b_k^T d)(s_k^T d)^2 + \frac{1}{24} \sum_{k=1}^{p} \tau_k (s_k^T d)^4 \tag{12}
\]

Let \( S \in \mathbb{R}^{n \times p} \), whose \( k \)-th column is \( s_k \), and the \( \{s_k\} \) be linearly independent. Before solving (12), we take advantage of QR factorization of \( S \) to transform (12) into two polynomials with \( p \) and \( n - p \) variables, respectively. The concrete form is:
\[
d = W u + Z t \tag{13}
\]
where \( Z \in \mathbb{R}^{n \times (n-p)} \) and \( W \in \mathbb{R}^{n \times p} \) which satisfy \( Z^T S = 0, W^T S = I \), \( \text{rank}(Z) = n - p \) and \( \text{rank}(W) = p \). For convenience, from now on, we use \( g \) and \( H \) instead of \( \nabla f(x_c) \) and \( \nabla^2 f(x_c) \). Then, (12) are transformed into:
\[
f(x_c + W u + Z t) = f(x_c) + g^T W u + g^T Z t + \frac{1}{2} u^T W^T H W u \\
+ u^T W^T H Z t + \frac{1}{2} t^2 Z^T H Z t \\
+ \frac{1}{2} \sum_{k=1}^{p} u_k^2 (b_k^T W u + b_k^T Z t) + \frac{1}{24} \sum_{k=1}^{p} \tau_k u_k^4 \tag{14}
\]
and it is a second degree polynomial about \( t \) when \( u \) is fixed. Here, we assume \( Z^T H Z \) is positive definite. Then, it is obvious that
\[
t = -(Z^T H Z)^{-1} Z^T (g + HW u + \frac{1}{2} \sum_{k=1}^{p} b_k u_k^2) \tag{15}
\]
is the minimum of the function in (14) with respect to the variable \( t \).

Furthermore, substitute (15) into (14) to obtain a fourth degree polynomial of \( u \):
\[
f^0(u) = f(x_c) + \frac{1}{2} u^T W^T H W u + \frac{1}{2} \sum_{i=1}^{p} u_i^2 (b_i^T W u) \\
+ \frac{1}{24} \sum_{i=1}^{p} \tau_i u_i^4 L^T Z (Z^T H Z)^{-1} Z^T L, \tag{16}
\]
where \( L = (g + HW u + \frac{1}{2} \sum_{i=1}^{p} b_i u_i^2) \). Suppose that \( u_s \) is the minimum of the above function. Then, by (15), we get \( t_s \). Thus, we can obtain the optimal point of (12) by
\[
d_s = W u_s + Z t_s. \tag{16}
\]
Particularly, if we set \( u = 0 \), then
\[
d_s = Z t_s = -Z (Z^T H Z)^{-1} Z^T g = -Z (Z^T H Z)^{-1} Z^T \nabla f(x_c). \tag{17}
\]
3 Algorithm and Convergence

In this section, we propose a non-monotone tensor algorithm for unconstrained optimization problem, and show the convergence of the algorithm. In the following part, \( \{x_m\} \) for \( 1 \leq m \leq p \) represents a sequence of past iterative points; \( k, dN \) and \( dT \) mean the number of iteration, the direction of Newton method and the direction of Tensor method correspondingly. \( dT^1 \) is the direction calculated by (16), and if we set \( u = 0 \) in (16), we get the direction \( dT^0 \) by (17). \( \lambda^T_0(k) \) and \( \lambda^N_0(k) \) denote the step size for tensor direction and Newton direction in the \( k \)-th iteration.

Algorithm 3.1 (Non-monotone tensor algorithm for solving (1))

Step 0 Choose \( \delta \) and \( \sigma \) such that \( 0 < \delta < \sigma < 1 \), \( \eta_0 \in (0, 1) \) and \( \epsilon_p > 0 \). Take the initial point \( x_0 \in \mathbb{R}^n \), and set \( C_0 = f(x_0) \). Let \( m_0 = 1 \). Choose an integer \( M \geq 2 \). Set \( k = 0 \).

Step 1 Calculate \( \nabla f(x_k) \) and \( |\nabla f(x_k)| \). If \( |\nabla f(x_k)| \leq \epsilon_p \), stop.

Step 2 Calculate \( \nabla^2 f(x_k) \) and \( f(x_k) \). If \( \det(\nabla^2 f(x_k)) = 0 \), then \( dN = -\nabla f(x_k) \); otherwise, \( dN = -|\nabla^2 f(x_k)|^{-1} \nabla^T f(x_k) \).

Step 3 If \( k = 0 \), then \( d = dT = dN \); otherwise, choose \( p (p \leq n^1/3) \) latest points as the sequence \( \{x_m\} (1 \leq m \leq p) \) and solve \( T \) and \( V \) by (11) and (10). Calculate \( dT^1 \) and \( dT^0 \) through (16) and (17). If \( \nabla f(x_k)T \geq 0 \), then \( dT = dT^1 \); otherwise, \( dT = dT^0 \).

Step 4 If \( k = 0 \), then find \( \lambda^T_0 \) by (2) when \( d = dN \), and set \( \lambda^N_0 = \lambda^T_0 \); otherwise, find \( \lambda^N_0 \) by (2) when \( d = dT \), and \( \lambda^T_0 \) by (2) when \( d = dN \), respectively.

Step 5 If \( f(x_k + \lambda^T_0 dN) \leq f(x_k + \lambda^N_0 dT) \), then \( x_{k+1} = x_k + \lambda^T_0 dN \); and \( d = dN \); otherwise, \( x_{k+1} = x_k + \lambda^N_0 dT \) and \( d = dT \).

Step 6 Set \( k = k + 1 \) and \( m_k = \min\{m_k, 1 + M\} \). If \( \sum_{i=1}^{m_k-1} \eta_{k-i} f(x_{k-i}) \geq \sum_{i=1}^{m_k-1} \eta_{k-i} f(x_k) \), then set \( \eta_k = \eta_0 \); otherwise, set \( \eta_k = 0 \). Set

\[
Q_k = 1 + \eta_k \sum_{i=1}^{m_k-1} \eta_{k-i},
\]

\[
C_k = \frac{\eta_k \sum_{i=1}^{m_k-1} \eta_{k-i} f(x_{k-i}) + f(x_k)}{Q_k}.
\]

Go back to Step 1.

Remark 8 In Step 4, which searching for the non-monotone step size \( \lambda_T \) and \( \lambda_N \), if the step size satisfying scheme (2) does not exist (i.e., \( \lambda < m \) holds, before \( \lambda \) satisfies (2), where \( m \) is a small constant such as \( 10^{-10} \)), we set \( \lambda = 0 \).

Assumption 3.1 The objective function \( f \) is at least twice continuously differentiable and bounded below. Besides, \( Z^T H Z \) is positive definite.

This assumption about continuous differentiability is fundamental in unconstrained optimization solved by derivative-based methods. Even though \( H(H = \nabla^2 f(x_k)) \) may be singular which is the main obstacle for Newton method, \( Z^T H Z \) can be both non-singular and positive definite only if \( \text{rank}(H) \geq n - p \). This is the main superiority of tensor method in [13].

Theorem 9 (Property of Descending) Suppose that Assumption 3.1 is satisfied. Then, the direction \( dT \) defined in Step 3 will be descending, which means \( \nabla f(x_c)^T dT \leq 0 \).

Proof. From Step 3 of Algorithm 3.1 and (17), we get

\[
dT_0^T \nabla f(x_c) = \nabla f(x_c)^T Z(Z^T H Z)^{-1} Z^T \nabla f(x_c),
\]

where \( Z \in \mathbb{R}^{n \times (n-p)} \). By the definition in (13), we have the matrix \( Z \) is column full rank. This, together with \( |\nabla f(x_c)| \neq 0 \) (terminate condition in Step 1), implies that \( Z^T \nabla f(x_c) \neq 0 \). Since, if we set \( F = Z^T \nabla f(x_c) \), (18) can be written as

\[
dT_0^T \nabla f(x_c) = -F^T (Z^T H Z)^{-1} F,
\]

and \( (Z^T H Z)^{-1} \) is positive definite, it follows that \( dT_0^T \nabla f(x_c) < 0 \). By combining the above result with \( \nabla f(x_c)^T dT \leq 0 \) given in Step 3, we complete the proof of this theorem. \( \square \)

By Theorem 9, when \( d = dT \) holds, the non-monotone line search scheme in Step 4 is finitely termination. In Algorithm 3.1, the Newton direction \( dN = -|\nabla^2 f(x_k)|^{-1} \nabla^T f(x_k) \) may not be descent when \( \nabla^2 f(x_k) \) is not positive definite. If we suppose the direction \( dN = -|\nabla^2 f(x_k)|^{-1} \nabla^T f(x_k) \) is not descent, by Remark 8, we will get \( \lambda_N = 0 \). And then the Algorithm 3.1 is well defined.

In the following, we show the convergence of Algorithm 3.1.
Lemma 10 Suppose that \( \{x_k\} \) is generated by Algorithm 3.1. Define \( \gamma_k \) as
\[
\gamma_k = \begin{cases} 
0 & \text{if } \eta_k = 0, \\
1 & \text{otherwise,} 
\end{cases}
\]
Then \( C_k \) is the convex combination of \( \gamma_{k-1} f(x_{k-1}), \gamma_{k-2} f(x_{k-2}), \ldots, \gamma_{k-M+1} f(x_{k-M+1}) \) and \( f(x_k) \).

Proof. From the definition of \( C_k \) in Step 6 of Algorithm 3.1, we know that \( C_k \) is a convex combination of \( f(x_{k-1}), f(x_{k-2}), \ldots, f(x_{k-M+1}) \) and \( f(x_k) \). Taking the situation \( \eta_i = 0 \) \( (k-M+1 \leq i \leq k-1) \) into account, we can conclude that \( C_k \) is a convex combination of \( f(x_j) \) and \( f(x_k) \), where \( j \in \{k-M+1 \leq i \leq k-1; \eta_i \neq 0\} \). Then, the lemma is proven.

Theorem 11 Suppose that \( \{x_k\} \) is generated by Algorithm 3.1. Then, there exists a subsequence \( \{y_k\} \) which allows the corresponding values of objective function to be descending. That is, \( f(y_k) \geq f(y_{k+1}) \) for any \( k \in N^+ \).

Proof. Because \( f \) is bounded below from Assumption 3.1, there exists a constant \( M_f \) \( > 0 \) such that \( f(x) + M_f \geq 0 \) for all \( x \in \mathbb{R}^n \). Without loss of generality, we assume that \( f(x) \geq 0 \) for all \( x \in \mathbb{R}^n \). In the following proof process, it is reasonable for us to suppose \( f(x_k) > 0 \) for \( k \in N^+ f(x_k) = 0 \) states \( x_k \) is the global optimal point) holds. Let \( \{\gamma_k\} \) be defined by Lemma 10. To find the subsequence \( \{y_k\} \), we pick every \( y_k \) from set \( \{x_k, \ldots, x_{k-M+2}\} \) for \( k \in N^+ \), where \( \{x_k\} \) is the sequence generated by Algorithm 3.1. Concretely, for any \( k \in N^+ \), we define \( y_k \) as follows:
\[
y_k = \arg \max_{k \leq j \leq k-M+2} \{\gamma_j f(x_j)\} \quad \text{if} \quad \sum_{i=k}^{k+M-2} \eta_i > 0, \quad (19)
\]
\[
y_k = x_{k+M-2} \quad \text{if} \quad \sum_{i=k}^{k+M-2} \eta_i = 0. \quad (20)
\]
Now let’s prove \( f(y_k) \geq f(y_{k+1}) \) for any \( k \in N^+ \).

When \( \sum_{i=k}^{k+M-2} \eta_i > 0 \) holds, we have
\[
f(y_k) = \max_{k \leq j \leq k-M+2} \{\gamma_j f(x_j)\},
\]
If \( f(y_k) \geq f(x_{k+M-1}) \), by the concrete definition of \( \{y_k\} \), we get
\[
f(y_{k+1}) = \max_{k+1 \leq j \leq k-M+1} \{\gamma_j f(x_j)\} \leq \max_{k \leq j \leq k-M+2} \{\gamma_j f(x_j)\}, f(x_{k+M-1}) \leq \max_{k \leq j \leq k-M+2} \{\gamma_j f(x_j)\}, f(y_k) = f(y_k).
\]
On the contrary, if \( f(y_k) < f(x_{k+M-1}) \), it is easy to conclude
\[
\sum_{i=k}^{M-1} \eta_{k+M-1-i} f(x_{k+M-1-i}) = \sum_{i=k}^{M-1} \eta_{k+M-1-i} (\gamma_{k+M-1-i} f(x_{k+M-1-i})) \leq \sum_{i=1}^{M-1} \eta_{k+M-1-i} f(y_k) < \sum_{i=1}^{M-1} \eta_{k+M-1-i} f(x_{k+M-1}).
\]
From Step 6 of Algorithm 3.1, we know \( \eta_{k+M-1} = 0 \), and then
\[
f(y_{k+1}) = \max_{1 \leq j \leq k+M-2} \{\gamma_j f(x_j)\} \leq f(y_k).
\]
Therefore, we can find a subsequence \( \{y_k\} \) which allows the sequence \( \{f(y_k)\} \) to be descending.

Assumption 3.2 There exist two positive constants \( c_1 \) and \( c_2 \) satisfying \( |\nabla f(x_k)|^2 \geq c_1 \|\nabla f(x_k)\|^2 \|d\| \geq c_2 \|\nabla f(x_k)\| (\text{Here } \|\cdot\| \text{ s-stands for the 2-norm}.)
\]

Theorem 12 Suppose that Assumptions 3.1 and 3.2 are satisfied and set \( \rho = \frac{\delta(1-\sigma)\epsilon^2}{2c_2^2} > 0 \). Then, the inequality \( f(x_{k+1}) \leq C_k - \rho \|\nabla f(x_k)\|^2 \) holds.

Proof. By (2), we get
\[
(\sigma - 1) \|\nabla f(x_k)\|^2 \leq \rho \|\nabla f(x_k)\|^2, \quad \text{from Assumption 3.1}, \|\nabla f(x)\| \text{ is Lipschitz continuous and then}
\]
\[
(\sigma - 1) \|\nabla f(x_k)\|^2 \leq \lambda_k L \|d\|^2,
\]
which, together with \( \|\nabla f(x_k)\|^2 \leq 0 \) and \( \sigma < 1 \), implies that
\[
\lambda_k \geq (1-\sigma)\frac{\|\nabla f(x_k)\|^2 d}{L \|d\|^2}. \quad (21)
\]
Combine Assumption 3.2 and (2); we have
\[ f(x_{k+1}) \leq C_k + \frac{\delta \lambda_k \nabla f(x_k)^T d}{2} \]
\[ \leq C_k - \frac{\delta \lambda_k c_1 \| \nabla f(x_k) \|^2}{2}. \] (22)

Through (21) and (23), we have
\[ \lambda_k \geq \frac{(1 - \sigma)c_1}{Lc_2^2}. \] (24)

Substituting (24) into (22), we get the conclusion stated in the theorem. \qed

**Lemma 13** Suppose that \( k \geq 2M - 1 \) and \( f(x) \geq 0 \). Then, the inequality \( C_k \leq f(y_{k-2M+2}) \) holds.

**Proof.** Let \( \{\gamma_k\} \) be defined by Lemma 10. From the definition of \( \{y_k\} \) and the descending of \( \{f(y_k)\} \) in Theorem 11, there exists \( j \in \{0, 1, \cdots, M-2\} \) which makes
\[ f(y_{k-2M+2}) = f(x_{k-2M+2} + j) \geq \gamma_D f(x_D) \] (25)
holds for all \( D \geq k - 2M + 2 \). Consider \( \eta_k, \eta_{k-1}, \cdots, \eta_{k-M+1} \). They can not be equal 0 at the same time because of Step 6 in Algorithm 3.1.

Now, without losing generality, we consider the worst situation \( \eta_k = \eta_{k-1} = \cdots = \eta_{k-M+2} = 0 \). Then,
\[ \gamma_{k-M+1} f(x_{k-M+1}) = f(x_{k-M+1}) \]
and from (25) and \( k - M + 1 \geq k - 2M + 2 \)
\[ f(y_{k-2M+2}) = f(x_{k-2M+2} + j) \geq f(x_{k-M+1}), \]
where \( j \in \{0, 1, \cdots, M-2\} \). From Lemma 10, we can conclude
\[ f(y_{k-2M+2}) \geq f(x_{k-M+1}) = C_{k-M+1}. \] (26)

On the other hand, because when \( n_k = \eta_{k-1} = \cdots = \eta_{k-M+2} = 0 \), (2) actually equals to monotone linear search scheme, we have
\[ C_k = f(x_k) \leq f(x_{k-1}) \leq f(x_{k-2}) \cdots \leq f(x_{k-M+2}) \leq C_{k-M+1} - \rho \| \nabla f(x_{k-M+1}) \|^2 \]
\[ \leq C_{k-M+1}. \] (27)

Combining (26) and (27), we will get the conclusion of this lemma. \qed

**Theorem 14** If Assumption 3.1 and \( k \geq 2M - 1 \) holds, the sequence \( \{x_k\} \) generated by Algorithm 3.1 satisfies
\[ \lim_{k \to \infty} \inf \| \nabla f(x_k) \| = 0. \]

**Proof.** From the definition of \( \{y_k\} \), it follows that \( y_k = x_{k+j} \) for one positive integer \( j \in \{0, 1, \cdots, M-2\} \). By Theorem 12, we have
\[ f(y_k) = f(x_{k+j}) \leq C_{k+j-1} - \rho \| \nabla f(x_{k+j-1}) \|^2; \]
and by Theorem 11 and Lemma 13, we have
\[ C_{k+j-1} \leq f(y_{k-2M+1}), \quad \forall j \in \{0, 1, \cdots, M-2\}. \]
Thus,
\[ f(y_k) \leq C_{k+j-1} - \rho \| \nabla f(x_{k+j-1}) \|^2 \leq f(y_{k-2M+1}) - \rho \| \nabla f(x_{k+j-1}) \|^2, \]
i.e.,
\[ \rho \| \nabla f(x_{k+j-1}) \|^2 \leq f(y_{k-2M+1}) - f(y_k). \]
Add them together for \( k = 2M-1, 2M, \cdots, \) and note that \( f \) is bounded below and Theorem 11, we have
\[ \sum_{k=2M-1}^{+\infty} \rho \| \nabla f(x_{k+j-1}) \|^2 < +\infty. \]
Since \( \rho \) is a positive constant, we obtain that
\[ \lim_{k \to +\infty} \inf \|\nabla f(x_k)\| = 0 \]
holds. \qed

### 4 Numerical Results

In this section we will report the results of numerical experiments to illustrate performance of Algorithm 3.1 from three aspects. In the first part, we will give a specific example to prove Algorithm 3.1 has a descending subsequence. Secondly, we will provide an example to indicate the advantages of non-montone tensor method comparing with monotone tensor method. At last, sixteen experiments with four methods are completed. All experiments are done at a PC with CPU of 2.4GHz and RAM of 2.0GB, and all codes are executed in MATLAB 7.8.0. In the following numerical experiments, we choose the parameters as \( \delta = 2.0 \times 10^{-4}, \sigma = 0.1, \eta_0 = 0.85, \rho = 1, \) \( eps = 10^{-6} \) and \( M = 5 \).

In all numerical results, we use \( MNTM \) to stand for the non-montone tensor method (i.e., Algorithm
3.1. If we set $\lambda^{(k)} = \lambda^{(k)}_V = 1$ constantly in Algorithm 3.1, then it reduces to the tensor method [13] which is represented by $TM$. Besides, if we set $\eta_k = 0$ for all $k$, then Algorithm 3.1 changes to a monotone tensor method abbreviated as $MMT$. When $d = dN$ and $\eta_k = 0$ for all $k$ hold constantly, the Algorithm 3.1 becomes the modified Newton method which is denoted as $MN$ in the following.

We test problem (1) where the objective functions are respectively defined as follows:

1. **EPF (Extended Penalty Function):**
   
   $$f(x) = 5 \times 10^{-5} \sum_{i=1}^{n} (x_i - 1)^2 + (\sum_{i=1}^{n} x_i^2 - 0.25)^2.$$  
   
   Initial point $x_0 = [1, 2, \cdots, n]^T$.  
   
   $f_{\text{opt}} = f([1.6667, \cdots, 1.6667]) = 1.1249e - 4$ when $n = 4$;  
   
   $f_{\text{opt}} = f([223, \cdots, 223]) = 3.5437e - 4$ when $n = 10$;  
   
   $f_{\text{opt}} = f([\frac{314}{2349}, \cdots, \frac{314}{2349}]) = 5.2539e - 4$ when $n = 14$.

2. **EF&RF (Extended Freudenstein & Roth Function):**
   
   $$f(x) = \sum_{i=1}^{2} \left[(-13x_{2i-1}((5 - x_{2i})x_{2i} - 2)x_{2i})^2 \right.$$  
   $$+((-29 + x_{2i-1} + ((x_{2i} + 1)x_{2i} - 14)x_{2i})^2)].$$  
   
   Initial point $x_0 = [1, 2, 1, 2]^T$.  
   
   $f_{\text{opt}} = 0$.

3. **ETF (Extended Trigonometric Function):**
   
   $$f(x) = \sum_{i=1}^{n} (n - \sum_{j=1}^{n} \cos x_j + i(1 - \cos x_i) - \sin x_i)^2.$$  
   
   Initial point $x_0 = [-0.5, -0.5, \cdots, -0.5]^T$.  
   
   $f_{\text{opt}} = 0$.

4. **RIF (Raydan 1 Function):**

   $$f(x) = \sum_{i=1}^{n} \frac{i}{10} (e^{x_i} - 1).$$

   Initial point $x_0 = [n, n, \cdots, n]^T$.  
   
   $f_{\text{opt}} = \sum_{i=1}^{n} \frac{i}{10}.$

5. **R2F (Raydan 2 Function):**

   $$f(x) = \sum_{i=1}^{14} (e^{x_i} - 1).$$

   Initial point $x_0 = [14, 14, \cdots, 14]^T$.  
   
   $f_{\text{opt}} = n$.

6. **EPF1 (Extended Powell Function 1):**

   $$f(x) = (x_3 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^2 + 10(x_1 - x_4)^4.$$  
   
   Initial point $x_0 = [4, 4, 4, 4]^T$.  
   
   $f_{\text{opt}} = 0$.

7. **EPF2 (Extended Powell Function 2):**

   $$f(x) = (x_1 + 10x_3)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^2 + 10(x_1 - x_4)^4.$$  
   
   Initial point $x_0 = [4, 4, 4, 4]^T$.  
   
   $f_{\text{opt}} = 0$.

8. **EM&CF (Extended Miele & Cantrell Function):**

   $$f(x) = \sum_{i=1}^{n/4} \left[(e^{x_{4i-3} - x_{4i-2} - 2})^2 + 100(x_{4i-2} - x_{4i-1})^6 \right.$$
   $$+ \tan^4(x_{4i-1} - x_{4i}) + x_{4i-3}^6].$$

   Initial point $x_0 = [n, n, \cdots, n]^T$.  
   
   $f_{\text{opt}} = 0$.

9. **BTF (Broyden Tridiagonal Function):**

   $$f(x) = \sum_{i=1}^{n} [(3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1]^2.$$  
   
   Initial point $x_0 = [n, n, \cdots, n]^T$.  
   
   $f_{\text{opt}} = 0$.

**Part I:**

In this part, we use the $NMTM$ to solve the unconstrained optimization whose objective function is defined as EPF ($n = 4$) to show the fact that Algorithm 3.1 generates a descending subsequence. The corresponding results are showed in Table 1, where $k$ represents the number of iteration; $f_k$ stands for the value of $f$ in the $k$th iteration; $\|g_k\|$ represents the norm of gradient $g_k = \nabla f(x_k)$ which determines whether the algorithm terminates; the step with symbol * means that current iterative direction is the tensor direction i.e., $d = dT$.

Through numerical results in Table 1, we notice that the values of objective function are convergent.

And except for the tenth iteration, all values of objective function at iteration points are descending, which confirms the result given in Theorem 11. In the third iteration, the tensor direction is chosen, and this indicates the tensor direction is better than Newton direction on that iteration. Furthermore, in Part III, we will find that the optimal solution of EPF ($n = 4$) can not be get by $MN$. That means the tensor direction is essential to guarantee the convergence and improve the speed of convergence on the foundation of Newton direction.
stands for the value of $f$ in the $k$th iteration and $g_k$ represents the norm of gradient $g_k = \nabla f(x_k)$ which determines whether the algorithm terminates; the step with symbol * means that current iterative direction is the tensor direction, i.e., $d = dT$. As we know, the monotone line search scheme shrinks the scope of step size due to guaranteeing decreasing of objective function value. Here, we aim to show that non-monotone line search scheme extends the searching scope and increases comparing with the seventh step. That is contrary, the value of $f_k$ in the eighth step of $NMTM$ increases comparing with the seventh step. That is the main difference between these two methods. Secondly, in the seventh step, the values of $\|x_k - x^*\|$ are the same, however, in the eighth step, the value of $\|x_k - x^*\|$ by $MTM$ is nearly as ten times larger as that by $NMTM$. This shows the advantage of $NMTM$ comparing with $MTM$. Thirdly, it is obvious that $\{x_k\}$ generated by $NMTM$ converges faster than the one by $MTM$, which is evidently showed in Table 2.

Part III:
All examples given above are calculated by four different methods: $MNLM$, $TM$, $MTM$ and $NMTM$. The numerical results are listed in Tables 3 and 4. In Tables 3 and 4, the dimension of each example is given in column $D$; the total computation time is given in column $Time(s)$; $Iter$ and $TIter$ denote the total iterations and the iterations using tensor direction. We use a symbol "−" to express the method fails to solve this example. The column $Example$ denotes the abbreviations of examples above. In the column $Error$, we report $\|f_k - f_{opt}\|$, where $f_k$ denotes the objective function value in the $k$th iteration and $f_{opt}$ denotes the minimal value of $f$.

From Tables 3 and 4, we find that $MTM$ succeeds to solve several experiments that $MNLM$ fails to solve such as EPF (4) and EPF1(4). However, the key difference between them is using the tensor direction as a descending direction in $MTM$, which indicates the tensor direction is necessary to guarantee the convergence. Especially, compared with the $MNLM$, the $MTM$ for Example EPF(4) uses tensor direction only twice, but this is very pivotal to get the convergent result. In fact, $MNLM$ fails to solve examples EPF(4), EPF1(4), EPF1(4) and EPF2(4), because of singularity of Hessian matrix in searching for Newton directions. This is the superiority of $TM$ mentioned in [13].

By contrasting $TM$ with $MTM$, we are aware of that adding monotone line search accelerates the speed of convergence and reduces the iterations obviously except for the example ETF. Specifically, $TM$ fails to solve examples R1F(14) and R2F(14) but $MTM$ succeed. It is noticeable that when using $TM$ for EF&RF, $\{x_k\}$ does not converge to an optimal point, which means $TM$ also fails actually.

Contrast the results of $NMTM$ with others. We find that $NMTM$ succeeds to solve all examples above. In addition, from the result of example EPF(14), we know adding non-monotone line search improves the efficiency of algorithm with less iterations and less time. Set $EM&CF$ and $BTF$ as examples. What is more important is that through non-monotone line search, $NMTM$ overcomes the weakness of monotone line search that is restricting the searching points in a creep along the bottom of a narrow curved valley. Then, $NMTM$ can find a globally optimal point for the examples, for which others can

<table>
<thead>
<tr>
<th>$k$ iteration</th>
<th>$f_k$</th>
<th>$|g_k|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.9731</td>
<td>23.3199</td>
</tr>
<tr>
<td>2</td>
<td>0.0435</td>
<td>0.5643</td>
</tr>
<tr>
<td>3</td>
<td>0.0011*</td>
<td>0.0585</td>
</tr>
<tr>
<td>4</td>
<td>$1.3494e-4$</td>
<td>0.0075</td>
</tr>
<tr>
<td>5</td>
<td>$1.2081e-4$</td>
<td>1.5902e-4</td>
</tr>
<tr>
<td>6</td>
<td>$1.2078e-4$*</td>
<td>2.1671e-4</td>
</tr>
<tr>
<td>7</td>
<td>$1.2080e-4$*</td>
<td>8.5829e-5</td>
</tr>
<tr>
<td>8</td>
<td>1.2002e-4</td>
<td>0.0028</td>
</tr>
<tr>
<td>9</td>
<td>1.1749e-4</td>
<td>0.0018</td>
</tr>
<tr>
<td>10</td>
<td>1.1764e-4*</td>
<td>6.3558e-5</td>
</tr>
<tr>
<td>11</td>
<td>1.1689e-4</td>
<td>0.0021</td>
</tr>
<tr>
<td>12</td>
<td>1.1506e-4</td>
<td>0.0011</td>
</tr>
<tr>
<td>13</td>
<td>1.1475e-4*</td>
<td>4.2445e-5</td>
</tr>
<tr>
<td>14</td>
<td>1.1356e-4*</td>
<td>0.0017</td>
</tr>
<tr>
<td>15</td>
<td>1.1279e-4</td>
<td>4.9512e-5</td>
</tr>
<tr>
<td>16</td>
<td>1.1279e-4*</td>
<td>1.3127e-4</td>
</tr>
<tr>
<td>17</td>
<td>1.1274e-4</td>
<td>8.7595e-4</td>
</tr>
<tr>
<td>18</td>
<td>1.1253e-4</td>
<td>1.9818e-5</td>
</tr>
<tr>
<td>19</td>
<td>1.1252e-4</td>
<td>2.8725e-4</td>
</tr>
<tr>
<td>20</td>
<td>1.1249e-4</td>
<td>6.2588e-7</td>
</tr>
</tbody>
</table>

Part II:
In this part, we use both $NMTM$ and $MTM$ to solve the unconstrained problem with a objective function EPF ($n = 14$). The numerical results are displayed in Table 2, where $x^* = [\frac{314}{2359}, \ldots, \frac{314}{2359}] \in \mathbb{R}^{14}$ stands for optimal point, $\|x_k - x^*\|$ measures the distance between the current iterative point and the optimal point, $k$ represents the number of iteration; $f_k$ stands for the value of $f$ in the $k$th iteration; $g_k$ represents the norm of gradient $g_k = \nabla f(x_k)$ which determines whether the algorithm terminates; the step with symbol * means that current iterative direction is the tensor direction, i.e., $d = dT$. As we know, the monotone line search scheme shrinks the scope of step size due to guaranteeing decreasing of objective function value. Here, we aim to show that non-monotone line search scheme extends the searching scope and reduces the iterations.

By comparing the results in Table 2, we can get following conclusions. Firstly, it is easy to find that the values of objective function decrease monotonously when EPF ($n=14$) is solved by $MTM$. On the contrary, the value of $f_k$ in the eighth step of $NMTM$ increases comparing with the seventh step. That is the main difference between these two methods. Secondly, in the seventh step, the values of $\|x_k - x^*\|$ are the same, however, in the eighth step, the value of $\|x_k - x^*\|$ by $MTM$ is nearly as ten times larger as that by $NMTM$. This shows the advantage of $NMTM$ comparing with $MTM$. Thirdly, it is obvious that $\{x_k\}$ generated by $NMTM$ converges faster than the one by $MTM$, which is evidently showed in Table 2.
Table 2: The numerical results by NMTM and MTM for EPF \((n = 14)\)

<table>
<thead>
<tr>
<th>(k)</th>
<th>(f_k)</th>
<th>(|g_k|)</th>
<th>(|x_k - x^*|)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12679</td>
<td>4784.8</td>
<td>10.1854</td>
</tr>
<tr>
<td>2</td>
<td>152.8521</td>
<td>175.6347</td>
<td>3.12</td>
</tr>
<tr>
<td>3</td>
<td>1.5060</td>
<td>5.9649</td>
<td>0.81015</td>
</tr>
<tr>
<td>4</td>
<td>5.5128e-4</td>
<td>0.0038</td>
<td>0.24457</td>
</tr>
<tr>
<td>5</td>
<td>5.3807e-4</td>
<td>0.0035</td>
<td>0.16015</td>
</tr>
<tr>
<td>6</td>
<td>5.3236e-4</td>
<td>0.0016</td>
<td>0.12964</td>
</tr>
<tr>
<td>7</td>
<td>5.3168e-4*</td>
<td>9.6659e-5</td>
<td>0.1297</td>
</tr>
<tr>
<td>8</td>
<td>5.7881e-4*</td>
<td>0.0148</td>
<td>0.015867</td>
</tr>
<tr>
<td>9</td>
<td>5.2548e-4</td>
<td>3.0554e-4</td>
<td>0.013667</td>
</tr>
<tr>
<td>10</td>
<td>5.2540e-4</td>
<td>1.4853e-4</td>
<td>0.0013913</td>
</tr>
<tr>
<td>11</td>
<td>5.2539e-4</td>
<td>2.0332e-6</td>
<td>3.9462e-4</td>
</tr>
<tr>
<td>12</td>
<td>5.2539e-4</td>
<td>3.0824e-7</td>
<td>2.119e-6</td>
</tr>
</tbody>
</table>

Table 3: Numerical results by TM and MTM

<table>
<thead>
<tr>
<th>Example</th>
<th>D</th>
<th>Iter</th>
<th>TIter</th>
<th>Error</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPF</td>
<td>4</td>
<td>30</td>
<td>10</td>
<td>1.30e-11</td>
<td>32.81</td>
</tr>
<tr>
<td>EPF</td>
<td>10</td>
<td>40</td>
<td>19</td>
<td>5.85e-14</td>
<td>64.35</td>
</tr>
<tr>
<td>EPF</td>
<td>14</td>
<td>43</td>
<td>26</td>
<td>4.58e-12</td>
<td>77.54</td>
</tr>
<tr>
<td>EF&amp;RF</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>1.64e+3(-)</td>
<td>7.8</td>
</tr>
<tr>
<td>ETF</td>
<td>6</td>
<td>8</td>
<td>0</td>
<td>2.95e-15</td>
<td>10.3</td>
</tr>
<tr>
<td>R1F</td>
<td>6</td>
<td>11</td>
<td>2</td>
<td>1.33e-15</td>
<td>8.8</td>
</tr>
<tr>
<td>R1F</td>
<td>8</td>
<td>13</td>
<td>1</td>
<td>4.44e-16</td>
<td>11.53</td>
</tr>
<tr>
<td>R1F</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R2F</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EPF1</td>
<td>4</td>
<td>21</td>
<td>3</td>
<td>9.57e-11</td>
<td>23.5</td>
</tr>
<tr>
<td>EPF2</td>
<td>4</td>
<td>21</td>
<td>3</td>
<td>9.57e-11</td>
<td>24.4</td>
</tr>
<tr>
<td>EM&amp;CF</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EM&amp;CF</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BTF</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BTF</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BTF</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table: Numerical results by TM and MTM

<table>
<thead>
<tr>
<th>Example</th>
<th>D</th>
<th>Iter</th>
<th>TIter</th>
<th>Error</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPF</td>
<td>4</td>
<td>17</td>
<td>2</td>
<td>1.23e-14</td>
<td>25.45</td>
</tr>
<tr>
<td>EPF</td>
<td>10</td>
<td>18</td>
<td>4</td>
<td>2.48e-13</td>
<td>49.36</td>
</tr>
<tr>
<td>EPF</td>
<td>14</td>
<td>18</td>
<td>3</td>
<td>2.13e-15</td>
<td>63.93</td>
</tr>
<tr>
<td>EF&amp;RF</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ETF</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1.50e-16</td>
<td>10.98</td>
</tr>
<tr>
<td>R1F</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>1.33e-15</td>
<td>6.04</td>
</tr>
<tr>
<td>R1F</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td>4.44e-16</td>
<td>7.09</td>
</tr>
<tr>
<td>R1F</td>
<td>14</td>
<td>8</td>
<td>0</td>
<td>3.55e-15</td>
<td>14.23</td>
</tr>
<tr>
<td>R2F</td>
<td>14</td>
<td>8</td>
<td>1</td>
<td>5.33e-15</td>
<td>34.56</td>
</tr>
<tr>
<td>EPF1</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>7.65e-12</td>
<td>17.60</td>
</tr>
<tr>
<td>EPF2</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>7.65e-12</td>
<td>18.45</td>
</tr>
<tr>
<td>EM&amp;CF</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EM&amp;CF</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BTF</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BTF</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BTF</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Through Algorithm 3.1, \(NMTM\) will cost more than the \(MNM\) because of calculating tensor direction during each iteration. So, it is reasonable that the \(MNM\) costs less time than \(NMTM, TM\) and \(MTM\) with the same iterations. At last, by all discussion- and data of those four tables, it is easy to conclude that combining tensor method and non-monotone line search is meaningful to improve efficiency of solving unconstrained optimizations and makes more uncon- strained optimization problems solvable.
Table 4: Numerical results by MNM and NMTM

<table>
<thead>
<tr>
<th>Example</th>
<th>D</th>
<th>Iter</th>
<th>TIter</th>
<th>Error</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPF</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EPF</td>
<td>10</td>
<td>18</td>
<td>6</td>
<td>2.92e-14</td>
<td>41.91</td>
</tr>
<tr>
<td>EPF</td>
<td>14</td>
<td>12</td>
<td>2</td>
<td>2.32e-14</td>
<td>38.32</td>
</tr>
<tr>
<td>EF&amp;RF</td>
<td>4</td>
<td>7</td>
<td>0</td>
<td>6.31e-30</td>
<td>33.32</td>
</tr>
<tr>
<td>ETF</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1.50e-16</td>
<td>10.55</td>
</tr>
<tr>
<td>R1F</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>8.88e-16</td>
<td>6.32</td>
</tr>
<tr>
<td>EM&amp;CF</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EM&amp;CF</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BTF</td>
<td>10</td>
<td>16</td>
<td>7</td>
<td>9.10e-11</td>
<td>29.50</td>
</tr>
<tr>
<td>BTF</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BTF</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>D</th>
<th>Iter</th>
<th>TIter</th>
<th>Error</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPF1</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EPF2</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EM&amp;CF</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EM&amp;CF</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BTF</td>
<td>10</td>
<td>22</td>
<td>0</td>
<td>4.19e-30</td>
<td>24.50</td>
</tr>
<tr>
<td>BTF</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BTF</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

5 Conclusion

In this paper we focused on the unconstrained optimization problem and proposed a method called a non-monotone tensor method, which is a tensor method combined with the non-monotone line search scheme. This method possesses advantages of both the tensor method and the non-monotone line search scheme. Several numerical experiments show the effectiveness of the proposed method. We believe that this method will show more superiorities when solving the practical problems in financial and engineering areas, which is a topic of further research.

References:


