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

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

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 α_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 λ_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 λ_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 λ_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), λ_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 λ_k such that

$$\begin{cases} f(x_k + \lambda_k d_k) \leq C_k + \frac{\delta \lambda_k \nabla f(x_k)^T d_k}{2}, \\ \nabla f(x_k + \lambda_k d_k)^T d_k \geq \sigma \nabla f(x_k)^T d_k. \end{cases} \tag{2}$$

Here, δ, σ, m_k are parameters satisfying $0 < \delta < \sigma < 1$, $m_k \in N^+$ 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 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 $A = (A_{i_1 \dots i_m})$ satisfies $A_{i_1 \dots i_m} \in \mathbb{R}$, where $i_j = 1, \dots, n_j$ for $j = 1, \dots, m$, i.e., $A \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_m}$, we call A an mth order real tensor. Furthermore, if $n_1 = n_2 = \dots = n_m$, 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 mth-order ($m \geq 3$) 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,*

$$\begin{aligned} T \cdot \mu\nu\omega &= \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} T_{ijk} \mu_i \nu_j \omega_k; \\ (T \cdot \nu\omega)_i &= \sum_{j=1}^{n} \sum_{k=1}^{n} T_{ijk} \nu_j \omega_k, \quad i = 1, \cdots, n; \\ V \cdot \zeta\mu\nu\omega &= \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V_{ijkl} \zeta_i \mu_j \nu_k \omega_l; \\ (V \cdot \mu\nu\omega)_i &= \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V_{ijkl} \mu_j \nu_k \omega_l, \ i = 1, \cdots, n, \end{aligned}$$

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 $V \cdot \mu^4$.

From the definition above, it is easy to realize that the square tensor has many similarities with the square matrix. Therefore, we can define the symmetry analogously.

Definition 3 *If A is a tensor defined by Definition 1 and satisfies:*

$$A_{i_1 i_2 \cdots i_m} = A_{j_1 j_2 \cdots j_m},\ 1 \le i_l, j_l \le n \ with\ 1 \le l \le m,$$

where the indices $j_1 j_2 \cdots j_m$ is an arbitrary permutation of the indices $i_1 i_2 \cdots i_m$, we call A a symmetric tensor.

The operations of tensor contain addition, scalar multiplication and tensor multiplication. Addition between two tensors with the same order and dimension means their corresponding elements add. Each element of a tensor multiplies by a scalar is the scalar multiplication between a scalar and a tensor.

Definition 4 *Suppose that $T \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ and $V \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_q}$. Then, $\otimes$ stands for the tensor multiplication and $T \otimes V \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p \times m_1 \times m_2 \times \cdots \times m_q}$ with*

$$(T \otimes V)_{i_1 i_2 \cdots i_p j_1 j_2 \cdots j_q} = T_{i_1 i_2 \cdots i_p} V_{j_1 j_2 \cdots j_q}.$$

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:

Definition 5 *A tensor $T \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$ is called a rank-one tensor, if there exist $\mu^{(i)} \in \mathbb{R}^{n_i}$ ($1 \le i \le p$) satisfying $T_{j_1 \cdots j_p} = \mu^{(1)}_{j_1} \cdots \mu^{(p)}_{j_p}$ ($1 \le j_i \le n_i$ for $1 \le i \le p$), i.e., $T = \mu^{(1)} \otimes \mu^{(2)} \otimes \cdots \otimes \mu^{(p)}$.*

In the following, we introduce the tensor method in [13] used to solve unconstrained optimization (1) briefly, which is divided into the following three parts.

Part I: The model of tensor method

It is necessary to expand $f(x)$ to a higher degree polynomial to approach the objective function more accurately, especially when $\nabla^2 f(x_*)$ is nonsingular. In this paper, the monotone tensor method for unconstrained optimization is based upon the fourth order tensor model

$$\begin{aligned} f(x_c + d) &= f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2} \nabla f(x_c) \cdot d^2 \\ &\quad + \frac{1}{6} T \cdot d^3 + \frac{1}{24} V \cdot d^4 \end{aligned} \tag{3}$$

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].

Part II: Solutions of T 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 restrict T and V to be symmetric and rank-one tensors, which not only shrink the scope of them, but also reduce memory space during the computation. At the same time, T and V satisfy the following system of equations:

$$\left\{ \begin{aligned} f(x_{-k}) &= f(x_c) + \nabla f(x_c) \cdot s_k \\ &\quad + \tfrac{1}{2} \nabla f(x_c) \cdot s_k^2 \\ &\quad + \tfrac{1}{6} T \cdot s_k^3 + \tfrac{1}{24} V \cdot s_k^4, \\ \nabla f(x_{-k}) &= \nabla f(x_c) + \nabla^2 f(x_c) \cdot s_k \\ &\quad + \tfrac{1}{2} T \cdot s_k^2 + \tfrac{1}{6} V \cdot s_k^3, \end{aligned} \right. \tag{4}$$

where $k = 1, \cdots, p$ (here, we set $p \le n^{1/3}$), $\{x_{-k}\}$ is the sequence of past linear independent iterative points and $s_k = x_{-k} - x_c$, $s_k \in \mathbb{R}^n$. Multiplying the second equation of (4) by s_k gives

$$\begin{aligned} \nabla f(x_{-k}) \cdot s_k &= \nabla f(x_c) \cdot s_k + \nabla^2 f(x_c) \cdot s_k^2 \\ &\quad + \frac{1}{2} T \cdot s_k^3 + \frac{1}{6} V \cdot s_k^4. \end{aligned} \tag{5}$$

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

$$\alpha_k = T \cdot s_k^3, \quad \beta_k = V \cdot s_k^4 \ \text{ for } k = 1, \cdots, p. \tag{6}$$

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

$$\left\{ \begin{aligned} \tfrac{1}{2}\alpha_k + \tfrac{1}{6}\beta_k &= \nabla f(x_{-k}) \cdot s_k - \nabla f(x_c) \cdot s_k \\ &\quad - \nabla^2 f(x_c) \cdot s_k^2, \\ \tfrac{1}{6}\alpha_k + \tfrac{1}{24}\beta_k &= f(x_{-k}) - f(x_c) \\ &\quad - \nabla f(x_c) \cdot s_k - \tfrac{1}{2} \nabla^2 f(x_c) \cdot s_k^2 \end{aligned} \right. \tag{7}$$

for $k = 1, \cdots, p$. The system (7) is nonsingular, so α_k and β_k have unique solution. Combining (6) and(7), we know V and T satisfy

$$V \cdot s_k^4 = \beta_k, \quad T \cdot s_k^2 = a_k, \qquad k = 1, \cdots, p, \tag{8}$$

where

$$\begin{aligned} a_k &= 2(\nabla f(x_{-k}) - \nabla f(x_c) - \nabla^2 f(x_c)\cdot s_k \\ &\quad -\frac{1}{6}V\cdot s_k^3), \text{ for } k=1,\cdots,p. \end{aligned} \tag{9}$$

Theorem 6 *Define* $P \in \mathbb{R}^{p\times p}$ *by* $P_{ij} = (s_i^T s_j)^4$, $1 \le i,j \le p$ *and define* $\tau \in \mathbb{R}^p$ *by* $\tau = P^{-1}\beta$. *Then, the solution to*

$$\begin{array}{ll} \min\limits_{V\in\mathbb{R}^{n\times n\times n\times n}} & \|V\|_F \\ subject\ to & V\cdot s_k^4 = \beta_k, k=1\cdots,p \\ & and\ V\ is\ symmetric. \end{array}$$

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* $a_k \in \mathbb{R}^n$ $(k=1,\cdots,p)$ *as* (9). *The solution to*

$$\begin{array}{ll} \min\limits_{V\in\mathbb{R}^{n\times n\times n}} & \|T\|_F \\ subject\ to & T\cdot s_i^2 = a_i,\ \ i=1\cdots p \\ & and\ T\ is\ symmetric. \end{array}$$

is

$$\begin{aligned} T &= \sum_{k=1}^{p} (b_k \otimes s_k \otimes s_k + s_k \otimes b_k \otimes s_k \\ &\quad + s_k \otimes s_k \otimes b_k), \end{aligned} \tag{11}$$

where $b_k \in \mathbb{R}^n$, $k=1,\cdots,p$, $\{b_k\}$ *is the unique sequence of vectors which makes* (11) *satisfy* $T\cdot s_i^2 = a_i$ $(i=1,\cdots,p)$ *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:

$$\begin{aligned} & f(x_c+d) \\ &= f(x_c) + \nabla f(x_c)\cdot d + \frac{1}{2}\nabla f(x_c)\cdot d^2 \\ &\quad + \frac{1}{2}\sum_{k=1}^{p}(b_k^T d)(s_k^T d)^2 + \frac{1}{24}\sum_{k=1}^{p}\tau_k (s_k^T d)^4 \end{aligned} \tag{12}$$

Let $S \in \mathbb{R}^{n\times p}$, whose kth 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 :

$$d = Wu + Zt \tag{13}$$

where $Z \in \mathbb{R}^{n\times(n-p)}$ and $W \in \mathbb{R}^{n\times p}$ which satisfy $Z^T S = 0$, $W^T S = I$, $rank(Z) = n-p$ and $rank(W) = p$. 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:

$$\begin{aligned} & f(x_c + Wu + Zt) \\ &= f(x_c) + g^T Wu + g^T Zt + \frac{1}{2}u^T W^T HWu \\ &\quad + u^T W^T HZt + \frac{1}{2}t^T Z^T HZt \\ &+ \frac{1}{2}\sum_{k=1}^{p} u_k^2 (b_k^T Wu + b_k^T Zt)\frac{1}{24}\sum_{k=1}^{p}\tau_k u_k^4, \end{aligned} \tag{14}$$

and it is a second degree polynomial about t when u is fixed. Here, we assume $Z^T HZ$ is positive definite. Then, it is obvious that

$$t = -(Z^T HZ)^{-1} Z^T (g + HWu + \frac{1}{2}\sum_{k=1}^{p} b_i u_i^2) \tag{15}$$

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:

$$\begin{aligned} & f^0(u) \\ &= f(x_c) + \frac{1}{2}u^T W^T HWu + \frac{1}{2}\sum_{i=1}^{p} u_i^2 (b_i^T Wu) \\ &\quad + \frac{1}{24}\sum_{i=1}^{p}\tau_i u_i^4 \frac{1}{2}L^T Z(Z^T HZ)^{-1} Z^T L, \end{aligned}$$

where $L = (g + HWu + \frac{1}{2}\sum_{i=1}^{p} b_i u_i^2)$. Suppose that u_* is the minimum of the above function. Then, by (15), we get t_*. Thus, we can obtain the optimal point of (12) by

$$d_* = Wu_* + Zt_*. \tag{16}$$

Particularly, if we set $u=0$, then

$$\begin{aligned} d_* &= Zt_* = -Z(Z^T HZ)^{-1} Z^T g \\ &= -Z(Z^T HZ)^{-1} Z^T \nabla f(x_c). \end{aligned} \tag{17}$$

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). $\lambda_T^{(k)}$ and $\lambda_N^{(k)}$ 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* δ *and* σ *such that* $0 < \delta < \sigma < 1$, $\eta_0 \in (0,1)$ *and* $eps > 0$. *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* $\|\nabla f(x_k)\| \leq eps$, *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* $\nabla f(x_k)^T dT_1 \leq 0$, *then* $dT = dT_1$; *otherwise,* $dT = dT_0$.

Step 4 *If* $k = 0$, *then find* λ_N^k *by* (2) *when* $d = dN$, *and set* $\lambda_T^k = \lambda_N^k$; *otherwise, find* λ_T^k *by* (2) *when* $d = dT$, *and* λ_N^k *by* (2) *when* $d = dN$, *respectively.*

Step 5 *If* $f(x_k + \lambda_N^k dN) \leq f(x_k + \lambda_T^k dT)$, *then* $x_{k+1} = x_k + \lambda_N^k dN$ *and* $d = dN$; *otherwise,* $x_{k+1} = x_k + \lambda_T^k dT$ *and* $d = dT$.

Step 6 *Set* $k = k+1$ *and* $m_k = min\{m_{k-1}+1, M\}$. *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* λ_T *and* λ_N, *if the step size satisfying scheme* (2) *does not exist (i.e.,* $\lambda < m$ *holds, before* λ *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,* $Z^T HZ$ *is positive definite.*

This assumption about continuous differentiability is fundamental in unconstrained optimization solved by derivative-based methods. Even though $H(H = \nabla^2 f(x_c))$ may be singular which is the main obstacle for Newton method, $Z^T HZ$ can be both nonsingular and positive definite only if $rank(H) \geq n - p$. 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 $dT_0 = -Z(Z^T HZ)^{-1}Z^T \nabla f(x_c)$. And hence,

$$\begin{aligned} &dT_0^T \nabla f(x_c) \\ &\quad = -\nabla f(x_c)^T [Z(Z^T HZ)^{-1} Z^T] \nabla f(x_c), \end{aligned} \tag{18}$$

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

$$dT_0^T \nabla f(x_c) = -F^T (Z^T HZ)^{-1} F,$$

and $(Z^T HZ)^{-1}$ is positive definite, it follows that $dT_0^T \nabla f(x_c) < 0$.

By combining the above result with $dT_1^T \nabla f(x_c) \leq 0$ given in Step 3, we complete the proof of this theorem. □

By Theorem 9, when $d = dT$ holds, the non-monotone line search scheme in Step 4 is finitely terminational. 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. Definite γ_k as*

$$\gamma_k = \begin{cases} 0 & \eta_k = 0, \\ 1 & \eta_k = \eta_0. \end{cases}$$

Then C_k is the convex combination of $\gamma_{k-1} f(x_{k-1})$, $\gamma_{k-2} f(x_{k-2})$, $\cdots$, $\gamma_{k-M+1} f(x_{k-M+1})$ 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 $f(x_{k-1})$, $f(x_{k-2})$, $\cdots$, $f(x_{k-M+1})$ and $f(x_k)$. Taking the situation $\eta_i = 0$ $(k - M + 1 \le i \le k - 1)$ into account, we can conclude that C_k is a convex combination of $f(x_j)$ and $f(x_k)$, where $j \in \{k - M + 1 \le i \le k - 1;\ \eta_i \ne 0\}$. Then, the lemma is proven. $\square$

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, $f(y_k) \ge f(y_{k+1})$ for any $k \in N^+$.*

Proof. Because f is bounded below from Assumption 3.1, there exists a constant $M_f > 0$ such that $f(x) + M_f \ge 0$ for all $x \in \mathbb{R}^n$. Without loss of generality, we assume $f(x) \ge 0$ for all $x \in \mathbb{R}^n$. In the following proof process, it is reasonable for us to suppose $f(x_k) > 0$ for $k \in N^+ (f(x_*) = 0$ states x_* 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:

$$y_k = \underset{k \le j \le k+M-2}{\operatorname{argmax}} \{\gamma_j f(x_j)\} \text{ if } \sum_{i=k}^{k+M-2} \eta_i > 0, \quad (19)$$

$$y_k = x_{k+M-2} \text{ if } \sum_{i=k}^{k+M-2} \eta_i = 0. \quad (20)$$

Now let's prove $f(y_k) \ge f(y_{k+1})$ for any $k \in N^+$.

- When $\sum_{i=k}^{k+M-2} \eta_i > 0$ holds, we have

$$f(y_k) = \max_{k \le j \le k+M-2} \{\gamma_j f(x_j)\}.$$

If $f(y_k) \ge f(x_{k+M-1})$, by the concrete definition of $\{y_k\}$, we get

$$\begin{aligned} & f(y_{k+1}) = \max\nolimits_{k+1 \le j \le k+M-1} \{\gamma_j f(x_j)\} \\ & \le \max\{\max_{k+1 \le j \le k+M-2} \{\gamma_j f(x_j)\}, f(x_{k+M-1})\} \\ & \le \max_{k \le j \le k+M-2} \{\gamma_j f(x_j), f(y_k)\} = f(y_k). \end{aligned}$$

On the contrary, if $f(y_k) < f(x_{k+M-1})$, it is easy to conclude

$$\begin{aligned} & \sum_{i=1}^{M-1} \eta_{k+M-1-i} f(x_{k+M-1-i}) \\ & = \sum_{i=1}^{M-1} \eta_{k+M-1-i} (\gamma_{k+M-1-i} f(x_{k+M-1-i})) \\ & \le \sum\nolimits_{i=1}^{M-1} \eta_{k+M-1-i} f(y_k) \\ & < \sum_{i=1}^{M-1} \eta_{k+M-1-i} f(x_{k+M-1}). \end{aligned}$$

From Step 6 of Algorithm 3.1, we know $\eta_{k+M-1} = 0$, and then

$$f(y_{k+1}) = \max_{k+1 \le j \le k+M-2} \{\gamma_j f(x_j)\} \le f(y_k).$$

- When $\sum_{i=k}^{k+M-2} \eta_i = 0$ holds, it follows that $\gamma_{k+i} = \eta_{k+i} = 0$ for $i = 1, 2, \cdots, M - 2$. From Step 6 of Algorithm 3.1, we have $\eta_{k+M-1} = \eta_0$, $\gamma_{k+M-2} = 1$ and $C_{k+M-2} = f(x_{k+M-2})$. Thus, by the definition of $\{y_k\}$ (19), we know $f(x_{k+M-1}) = f(y_{k+1})$.

  In this case $(\eta_{k+M-2} = 0)$, (2) is equivalent to monotone linear search which means $f(x_{k+M-2}) > f(x_{k+M-1})$. Through the definition of y_k (20), we have

$$f(y_k) = f(x_{k+M-2}) \text{ and } f(y_k) > f(y_{k+1}).$$

Therefore, we can find a subsequence $\{y_k\}$ which allows the sequence $\{f(y_k)\}$ to be descending. $\square$

Assumption 3.2 *There exist two positive constants c_1 and c_2 satisfying $|\nabla f(x_k)^T d| = -\nabla f(x_k)^T d \ge c_1 \|\nabla f(x_k)\|^2$ and $\|d\| \le c_2 \|\nabla f(x_k)\|$ (Here $\|\cdot\|$ stands for the 2-norm).*

Theorem 12 *Suppose that Assumptions 3.1 and 3.2 are satisfied and set $\rho = \frac{\delta(1-\sigma)c_1^2}{2Lc_2^2} > 0$. Then, the inequality $f(x_{k+1}) \le C_k - \rho \|\nabla f(x_k)\|^2$ holds.*

Proof. By (2), we get

$$(\nabla f(x_k + \lambda_k d) - \nabla f(x_k))^T d \ge (\sigma - 1) \nabla f(x_k)^T d.$$

From Assumption 3.1, $\nabla f(x)$ is Lipschitz continuous and then

$$(\sigma - 1) \nabla f(x_k)^T d \le \lambda_k L \|d\|^2,$$

which, together with $\nabla f(x_k)^T d \le 0$ and $\sigma < 1$, implies that

$$\lambda_k \ge \frac{(1 - \sigma) |\nabla f(x_k)^T d|}{L \|d\|^2}. \quad (21)$$

Combine Assumption 3.2 and (2); we have

$$\begin{aligned} f(x_{k+1}) &\le C_k + \frac{\delta\lambda_k \nabla f(x_k)^T d}{2} \\ &\le C_k - \frac{\delta\lambda_k c_1 \|\nabla f(x_k)\|^2}{2}. \end{aligned} \tag{22}$$

$$\frac{|\nabla f(x_k)^T d|}{L\|d\|^2} \ge \frac{c_1\|\nabla f(x_k)\|^2}{Lc_2^2\|\nabla f(x_k)\|^2} = \frac{c_1}{Lc_2^2}. \tag{23}$$

Through (21) and (23), we have

$$\lambda_k \ge \frac{(1-\sigma)c_1}{Lc_2^2}. \tag{24}$$

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

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

$$f(y_{k-2M+2}) = f(x_{k-2M+2+j}) \ge \gamma_D f(x_D) \tag{25}$$

holds for all $D \ge k-2M+2$. Consider η_k, η_{k-1}, $\cdots$, η_{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 $\eta_{k-M+1} = \eta_0$ and $\eta_k = \eta_{k-1} = \cdots = \eta_{k-M+2} = 0$. Then,

$$\gamma_{k-M+1} f(x_{k-M+1}) = f(x_{k-M+1})$$

and from (25) and $k-M+1 \ge k-2M+2$

$$f(y_{k-2M+2}) = f(x_{k-2M+2+j}) \ge f(x_{k-M+1}),$$

where $j \in \{0, 1, \cdots, M-2\}$. From Lemma 10, we can conclude

$$f(y_{k-2M+2}) \ge f(x_{k-M+1}) = C_{k-M+1}. \tag{26}$$

On the other hand, because when $\eta_k = \eta_{k-1} = \cdots = \eta_{k-M+2} = 0$, (2) actually equals to monotone linear search scheme, we have

$$\begin{aligned} C_k = f(x_k) &\le f(x_{k-1}) \le f(x_{k-2}) \cdots \le f(x_{k-M+2}) \\ &\le C_{k-M+1} - \rho\|\nabla f(x_{k-M+1})\|^2 \\ &\le C_{k-M+1}. \end{aligned} \tag{27}$$

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

Theorem 14 *If Assumption 3.1 and $k \ge 2M-1$ 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

$$f(y_k) = f(x_{k+j}) \le C_{k+j-1} - \rho\|\nabla f(x_{k+j-1})\|^2;$$

and by Theorem 11 and Lemma 13, we have

$$C_{k+j-1} \le f(y_{k-2M+1}), \quad \forall j \in \{0, 1, \cdots, M-2\}.$$

Thus,

$$\begin{aligned} f(y_k) &\le C_{k+j-1} - \rho\|\nabla f(x_{k+j-1})\|^2 \\ &\le f(y_{k-2M+1}) - \rho\|\nabla f(x_{k+j-1})\|^2, \end{aligned}$$

i.e.,

$$\rho\|\nabla f(x_{k+j-1})\|^2 \le f(y_{k-2M+1}) - f(y_k).$$

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 ρ is a positive constant, we obtain that

$$\lim_{k\to\infty} \inf \|\nabla f(x_k)\| = 0$$

holds. □

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-monotone 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 $\delta = 2.0 \times 10^{-4}$, $\sigma = 0.1$, $\eta_0 = 0.85$, $p = 1$, $eps = 10^{-6}$ and $M = 5$.

In all numerical results, we use $MNTM$ to stand for the non-monotone tensor method (i.e., Algorithm

3.1). If we set $\lambda_T^{(k)} = \lambda_N^{(k)} = 1$ 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 MTM. 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 MNM 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 $x_0 = [1, 2, \cdots, n]^T$. $f_{opt} = f([\frac{1667}{6667}, \ldots, \frac{1667}{6667}]) = 1.1249\text{e} - 4$ when $n = 4$; $f_{opt} = f([\frac{223}{1410}, \ldots, \frac{223}{1410}]) = 3.5437\text{e} - 4$ when $n = 10$; $f_{opt} = f([\frac{314}{2349}, \ldots, \frac{314}{2349}]) = 5.2539\text{e} - 4$ when $n = 14$.

2. EF&RF (Extended Freudenstein & Roth Function):

$$f(x) = \sum_{i=1}^{2} [(-13x_{2i-1}((5 - x_{2i})x_{2i} - 2)x_{2i})^2 + (-29 + x_{2i-1} + ((x_{2i} + 1)x_{2i} - 14)x_{2i})^2].$$

Initial point $x_0 = [1, 2, 1, 2]^T$, $f_{opt} = 0$.

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 $x_0 = [-0.5, -0.5, \cdots, -0.5]^T$, $f_{opt} = 0$.

4. R1F (Raydan 1 Function):

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

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

5. R2F (Raydan 2 FunctionF):

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

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

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 $x_0 = [4, 4, 4, 4]^T$, $f_{opt} = 0$.

7. EPF2 (Extended Powell Function 2):

$$f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^2 + 10(x_1 - x_4)^4].$$

Initial point $x_0 = [4, 4, 4, 4]^T$, $f_{opt} = 0$.

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

$$f(x) = \sum_{i=1}^{n/4} [(e^{x_{4i-3}} - x_{4i-2})^2 + 100(x_{4i-2} - x_{4i-1})^6 + \tan^4(x_{4i-1} - x_{4i}) + x_{4i-3}^8].$$

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

9. BTF (Broyden Tridiagonal FunctionF):

$$f(x) = \sum_{i=1}^{n} [(3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1]^2.$$

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

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 kth 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 MNM. That means the tensor direction is essential to guarantee the convergence and improve the speed of convergence on the foundation of Newton direction.

Table 1: The numerical results by NMTM for EPF ($n = 4$)

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

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 $x^* = [\frac{314}{2349}, \dots, \frac{314}{2349}] \in \mathbb{R}^{14}$ 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 kth 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 guaranteing 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.

Part III:

All examples given above are calculated by four different methods: MNM, 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 kth 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 MNM 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 MNM, the MTM for Example EPF(4) uses tensor direction only twice, but this is very pivotal to get the convergent result. In fact, MNM fails to solve examples EPF(4), EPF(14), 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 2: The numerical results by NMTM and MTM for EPF ($n = 14$)

k	NMTM		
	f_k	$\|g_k\|$	$\|x_k - x^*\|$
1	12679	4784.8	10.1854
2	152.8521	175.6347	3.12
3	1.5060	5.9649	0.81015
4	5.5128e-4	0.0038	0.24457
5	5.3807e-4	0.0035	0.16015
6	5.3236e-4	0.0016	0.12964
7	5.3168e-4*	9.6659e-5	0.1297
8	5.7881e-4*	0.0148	0.015867
9	5.2548e-4	3.0554e-4	0.013667
10	5.2540e-4	1.4853e-4	0.0013913
11	5.2539e-4	2.0332e-6	3.9462e-4
12	5.2539e-4	3.0824e-7	2.119e-6
13			
14			
15			
16			
17			
18			

k	MTM		
	f_k	$\|g_k\|$	$\|x_k - x^*\|$
1	12679	4784.8	10.1854
2	152.8521	175.6347	3.12
3	1.5060	5.9649	0.81015
4	5.5128e-4	0.0038	0.24457
5	5.3807e-4	0.0035	0.16015
6	5.3236e-4	0.0016	0.12964
7	5.3168e-4*	9.6659e-5	0.1297
8	5.2989e-4	0.0014	0.10403
9	5.2854e-4	0.0027	0.059177
10	5.2739e-4*	5.4680e-5	0.073174
11	5.2685e-4	0.0017	0.04404
12	5.2605e-4	0.0012	0.0282
13	5.2593e-4*	2.8538e-5	0.038056
14	5.2565e-4	6.7753e-4	0.019696
15	5.2542e-4	2.8189e-4	0.0056622
16	5.2540e-4	2.6841e-6	0.0035179
17	5.2539e-4	2.4831e-5	1.3403e-5
18	5.2539e-4	9.9380e-10	3.1258e-7

Table 3: Numerical results by TM and MTM

	Example	D	Iter	TIter	Error	Time(s)
TM	EPF	4	30	10	1.30e-11	32.81
	EPF	10	40	19	5.85e-14	64.35
	EPF	14	43	26	4.58e-12	77.54
	EF&RF	4	4	0	1.64e+3(-)	7.8
	ETF	6	8	0	2.95e-15	10.3
	R1F	6	11	2	1.33e-15	8.8
	R1F	8	13	1	4.44e-16	11.53
	R1F	14	-	-	-	-
	R2F	14	-	-	-	-
	EPF1	4	21	3	9.57e-11	23.5
	EPF2	4	21	3	9.57e-11	24.4
	EM&CF	4	-	-	-	-
	EM&CF	8	-	-	-	-
	BTF	10	-	-	-	-
	BTF	12	-	-	-	-
	BTF	14	-	-	-	-
MTM	EPF	4	17	2	1.23e-14	25.45
	EPF	10	18	4	2.48e-13	49.36
	EPF	14	18	3	2.13e-15	63.93
	EF&RF	4	-	-	-	-
	ETF	6	5	1	1.50e-16	10.98
	R1F	6	6	0	1.33e-15	6.04
	R1F	8	5	0	4.44e-16	7.09
	R1F	14	8	0	3.55e-15	14.23
	R2F	14	8	1	5.33e-15	34.56
	EPF1	4	9	0	7.65e-12	17.60
	EPF2	4	9	0	7.65e-12	18.45
	EM&CF	4	-	-	-	-
	EM&CF	8	-	-	-	-
	BTF	10	-	-	-	-
	BTF	12	-	-	-	-
	BTF	14	-	-	-	-

not find, and improve the speed of convergence simultaneously. In fact, from the system of inequalities (2), we know that the non-monotone line search contains the monotone one (when $\eta_k = 0$ for all k). Therefore, the $NMTM$ method provides us more options.

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 discussions 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 unconstrained optimization problems solvable.

Table 4: Numerical results by MNM and NMTM

	Example	D	Iter	TIter	Error	Time(s)
	EPF	4	-	-	-	-
	EPF	10	19	0	8.07e-11	19.12
	EPF	14	-	-	-	-
	EF&RF	4	-	-	-	-
	ETF	6	7	0	0	7.01
	R1F	6	6	0	8.88e-16	6.32
M	R1F	8	-	-	-	-
N	R1F	14	8	0	0	8.66
M	R2F	14	8	0	1.78e-15	7.39
	EPF1	4	-	-	-	-
	EPF2	4	-	-	-	-
	EM&CF	4	-	-	-	-
	EM&CF	8	-	-	-	-
	BTF	10	22	0	4.19e-30	24.50
	BTF	12	-	-	-	-
	BTF	14	-	-	-	-
	EPF	4	20	7	9.10e-11	29.50
	EPF	10	18	6	2.92e-14	41.91
	EPF	14	12	2	2.32e-14	38.32
	EF&RF	4	7	0	6.31e-30	33.32
	ETF	6	5	1	1.50e-16	10.55
N	R1F	6	6	0	1.33e-15	6.04
M	R1F	8	5	0	4.44e-16	6.12
T	R1F	14	8	0	3.55e-15	14.23
M	R2F	14	8	1	5.33e-15	34.17
	EPF1	4	9	0	7.65e-12	17.60
	EPF2	4	9	0	7.65e-12	18.45
	EM&CF	4	23	7	2.32e-10	85.47
	EM&CF	8	27	4	1.66e-10	148.83
	BTF	10	27	11	1.82e-18	56.87
	BTF	12	24	11	1.17e-22	58.37
	BTF	14	28	10	5.22e-23	82.79

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:

[1] A. Bouaricha, A software package for large, sparse unconstrained optimization using tensor methods, *ACM T. Math. Softw.* 3, 1997, pp. 81–90.

[2] A. Bouaricha, Tensor methods for large sparse unconstrained optimization, *SIAM J. Optim.* 7, 1997, pp. 732–756.

[3] A. Bouaricha and R. B. Schnabel, Algorithm 768: TENSOLVE: A software package for solving systems of nonlinear equations and nonlinear least-squares problems using tensor methods, *ACM T. Math. Softw.* 23, 1997, pp. 147–195.

[4] A. Bouaricha and R. B. Schnabel, Tensor methods for nonlinear least squares problems, *SIAM J. Sci. Comput.* 21, 1999, pp. 1199–1221.

[5] T. T. Chow, *Derivative and Secant Tensor Methods for Unconstrained Optimization*, Ph. D. thesis, University of Colorado, Computer Science Department, 1989.

[6] T. T. Chow, E. Eskow and R. B. Schnabel, A software package for unconstrained optimization using tensor methods, *ACM T. Math. Softw.* 20, 1994 pp. 518–530.

[7] Y. H. Dai, On the nonmontone line search, *J. Optim. Theory Appl.* 112, 2002, pp. 315–330.

[8] Y. H. Dai, A nonmonotone conjucate gradient algorithm for unconstrained optimization, *J. Syst. Sci. Complex.* 15, 2002, pp. 139–145.

[9] L. Grippo, F. Lampariello and S. Lucidi, A nonmontone line search technique for Newton's method, *SIAM J. Numer. Anal.* 23, 1986, pp. 707–716.

[10] S. L. Hu, Z. H. Huang and N. Lu, A nonmonotone line search algorithm for unconstrained optimization, *J. Sci. Comput.* 42, 2010, pp. 38–53.

[11] S. L. Hu, Z. H. Huang and P. Wang, A nonmonotone smoothing Newton algorithm for solving nonlinear complementarity problems, *Optim. Methods Softw.* 24, 2009, pp. 447–460.

[12] N. Jorge, Theory of algorithms for unconstrained optimization, *Acta. Numerica.* 1, 1992, pp. 199–242.

[13] R. B. Schnabel and T. T. Chow, Tensor methods for unconstrained optimization using second derivatives, *SIAM J. Optim.* 1, 1991, pp. 293–315.

[14] R. B. Schnabel and P.D. Frank, Tensor methods for nonlinear equations, *SIAM J. Numer. Anal.* 21, 1984, pp. 815-843.

[15] Z. J. Shi and J. Shen, Convergence of nonmonotone line search method, *J. Computat. Appl. Math.* 193, 2006, pp. 397–412.

[16] N. Zhao and Z. H. Huang, A nonmonotone smoothing Newton algorithm for solving box constrained variational inequalities with a P_0 function, *J. Indus. Manag. Optim.* 7, 2011, pp. 467–482.

[17] H. C. Zhang and W. H. William, A nonmonotone line search technique and its application to unconstrained optimization, *SIAM J. Optim.* 14, 2004, pp. 1043–1056.