Journal of Computational Mathematics
Vol.32, No.4, 2014, 476–490.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1401-m3975

# A NEW NONMONOTONE TRUST REGION ALGORITHM FOR SOLVING UNCONSTRAINED OPTIMIZATION PROBLEMS*

Jinghui Liu and Changfeng Ma
*School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350007, China*
*Email: macf@fjnu.edu.cn*

**Abstract**

Based on the nonmonotone line search technique proposed by Gu and Mo (Appl. Math. Comput. 55, (2008) pp. 2158-2172), a new nonmonotone trust region algorithm is proposed for solving unconstrained optimization problems in this paper. The new algorithm is developed by resetting the ratio $\rho_k$ for evaluating the trial step $d_k$ whenever acceptable. The global and superlinear convergence of the algorithm are proved under suitable conditions. Numerical results show that the new algorithm is effective for solving unconstrained optimization problems.

*Mathematics subject classification:* 90C30.
*Key words:* Unconstrained optimization problems; Nonmonotone trust region method; Global convergence; Superlinear convergence.

## 1. Introduction

We consider the following unconstrained optimization problem

$$\min_{x\in\Re^n} f(x), \tag{1.1}$$

where $f:\Re^n\to\Re$ is a continuously differentiable function.

Line search method and trust region method are two principal methods for solving unconstrained optimization problem (1.1). Line search method produce a sequence $x_0, x_1, \cdots$, where $x_{k+1}$ is generated from the current approximate solution $x_k$, the specific direction $d_k$ and a stepsize $\alpha_k > 0$ by the rule $x_{k+1} = x_k + \alpha_k d_k$. On the other hand, the trust region methods obtain a trial step $d_k$ by solving the following quadric program subproblem:

$$\begin{array}{ll}\min\limits_{d\in\Re^n} & \phi_k(d) = g_k^T d + \dfrac{1}{2} d^T B_k d, \\ s.t. & \|d\| \le \Delta_k,\end{array} \tag{1.2}$$

where $g_k = \nabla f(x_k), B_k \in \Re^{n\times n}$ is a symmetric matrix which is either the exact Hessian matrix of $f$ at $x_k$ or an approximation for it, $\Delta_k > 0$ is the trust region radius and $\|\cdot\|$ denotes the Euclidean norm. The traditional trust region methods evaluate the trial step $d_k$ by the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)}. \tag{1.3}$$

The trial step $d_k$ is accepted whenever $\rho_k$ is greater than a positive constant $\mu$, then we can set the new point $x_{k+1} = x_k + d_k$ and enlarge the trust region radius. Otherwise, the traditional

* Received December 14, 2011 / Revised version received August 16, 2013 / Accepted January 15, 2014 / Published online July 3, 2014 /

trust region methods resolve the subproblem (1.2) by reducing the trust region radius until an acceptable step is found. Solving the region subproblems may lead us to solve one or more quadric program problems and increase the total cost of computation for large scale problems. Compared with line search techniques, new trust region methods have a strong convergence property, its computational cost is much lower than the traditional trust region methods (e.g. [1-3]). Some theoretical and numerical results of these trust region methods with line search are also interesting.

However, all these methods enforce monotonically decreasing of the objective function values at each iteration may slow the convergence rate in the minimization process, see [4,8]. In order to overcome the shortcomings, the earliest nonmonotone line search framework was developed by Grippo, Lampariello and Lucidi in [5] for Newton's method, in which their approach was: parameters $\lambda_1, \lambda_2, \sigma$ and $\beta$ are introduced, where $0 < \lambda_1 < \lambda_2$, $\beta, \sigma \in (0,1)$ and $\alpha_k = \bar{\alpha}_k \sigma^{h_k}$, where $\bar{\alpha}_k \in (\lambda_1, \lambda_2)$ is the trial step and $h_k$ is the smallest nonnegative integer such that

$$f(x_k + d_k) \leq \max_{0 \leq j \leq m_k} f(x_{k-j}) + \beta \alpha_k \nabla f(x_k)^T d_k, \tag{1.4}$$

the memory $m_k$ is a nondecreasing integer by setting

$$m_k = \begin{cases} 0, & k = 0, \\ 0 < m_k \leq \min\{m_{k-1} + 1, M\}, & k > 0, \end{cases}$$

where $M$ is a prefixed nonnegative integer.

From then on, researches in nonlinear optimization area have paid great attentions to it, see [7,8,10-14]. Deng et al. [4] made some changes in the common ratio (1.3) by resetting the rule as follows:

$$\hat{\rho}_k = \frac{\max\limits_{0 \leq j \leq m_k} f(x_{k-j}) - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)}. \tag{1.5}$$

The ratio (1.5) which assesses the agreement between the quadratic model and the objective function in trust region methods is the most common nonmonotone ratio, some researchers showed that the nonmonotone method can improve both the possibility of finding the global optimum and the rate of convergence when a monotone scheme is forced to creep along the bottom of a narrow curved valley (see [4, 9, 17]).

Although the nonmonotone technique (1.4) has many advantages, Zhang and Hager [18] proposed a new nonmonotone line search algorithm, which had the same general form as the scheme of Grippo et al. [5], except that their "max" was replaced by an average of function values. The nonmonotone line search found a step length $\beta$ to satisfy the following inequality:

$$f(x_k + \beta d_k) \leq C_k + \delta \beta \nabla f(x_k)^T d_k, \tag{1.6}$$

where

$$C_k = \begin{cases} f(x_k), & k = 0, \\ \big(\eta_{k-1} Q_{k-1} C_{k-1} + f(x_k)\big)/Q_k, & k \geq 1, \end{cases} \tag{1.7}$$

$$Q_k = \begin{cases} 1, & k = 0, \\ \eta_{k-1} Q_{k-1} + 1, & k \geq 1, \end{cases} \tag{1.8}$$

$\eta_{k-1} \in [\eta_{\min}, \eta_{\max}]$, where $\eta_{\min} \in [0,1)$ and $\eta_{\max} \in [\eta_{\min}, 1]$ are two chosen parameters. Numerical results showed that the new nonmonotone can improve the efficiency of the nonmonotone trust region methods.

Observing that $C_{k+1}$ is a convex combination between $C_k$ and $f(x_{k+1})$. Since $C_0 = f(x_0)$, we see that $C_k$ is a convex combination of the function values $f(x_0), f(x_1), \cdots, f(x_k)$ from (1.7), the degree of nonmonotonicity and (1.8) depend on the choice $\eta_k$. If $\eta_k = 0$ for each $k$, then the line search is the usual Armijo line search. If $\eta_k = 1$ for each $k$, then $C_k = A_k$, where

$$A_k = \frac{1}{k+1}\sum_{i=0}^{k} f_i, \quad f_i = f(x_i),$$

is the average function value.

However, it becomes an encumbrance to update $\eta_k$ and $Q_k$ at each $k$ in practice. Recently Gu and Mo [6] developed an algorithm that combining a new nonmonotone technique and trust region method for unconstrained optimization problems. The new nonmonotone line search is as follows:

$$f(x_k + \beta d_k) \leq D_k + \delta\beta\nabla f(x_k)^T d_k, \tag{1.9}$$

where the parameter $\eta \in (0, 1)$ or a variable $\eta_k$ and

$$D_k = \begin{cases} f(x_k), & k = 0, \\ \eta D_{k-1} + (1-\eta) f(x_k), & k \geq 1, \end{cases} \tag{1.10}$$

is a simple convex combination of the previous $D_{k-1}$ and $f_k$.

In this paper, we develop an algorithm by resetting the ratio $\rho_k$ in the trust region method for unconstrained optimization problems. The algorithm does not restrict monotonically decreasing of the objective function values at each iteration. Under suitable assumptions, we establish the global and superlinear convergence of the new algorithm. Numerical experiments show that our algorithm is quite effective.

Our paper is organized as follows: In Section 2, a new nonmonotone trust region algorithm is described. Under certain conditions, the global and superlinear convergence of the algorithm are proved in Sections 3 and 4, respectively. Numerical results are provided in Section 5 which showed that the new method is quite effective for unconstrained optimization problems. Finally, some concluding remarks are given in Section 6.

## 2. New Nonmonotone Trust Region Algorithm

For convenience, we denote $f(x_k)$ by $f_k$ and $g(x_k)$ by $g_k$, where $g(x_k) \in \Re^n$ is the gradient of $f$ evaluated at $x_k$. The trial step $d_k$ is obtained by solving the subproblem (1.2) at each iteration. In this paper, we solve (1.2) such that $\|d_k\| \leq \Delta_k$ and

$$\phi_k(0) - \phi_k(d_k) \geq \tau\|g_k\| \min\{\Delta_k, \|g_k\|/\|B_k\|\}, \tag{2.1}$$

where $\tau \in (0, 1)$ is a constant.

Obviously if $B_k$ is a symmetric and positive definite diagonal matrix, we can obtained the solution $d_k$ easily. More precisely, if $\|B_k^{-1} g_k\| \leq \Delta_k$, then $d_k = -B_k^{-1} g_k$ is the optimal solution of the subproblem (1.2); otherwise, if $\|B_k^{-1} g_k\| > \Delta_k$, we choose the optimal solution

$$d_k = -\frac{\Delta_k}{\|B_k^{-1} g_k\|} B_k^{-1} g_k.$$

In order to decide whether the obtained trial step $d_k$ will be accepted or not, and how to adjust the new trust region radius, we compute the ratio $\rho_k$ between the actual reduction,

$$Ared_k = D_k - f(x_k + d_k),$$

and the predicted reduction as follows:

$$Pred_k = \phi_k(0) - \phi_k(d_k),$$

i.e.,

$$\rho_k = \frac{Ared_k}{Pred_k} = \frac{D_k - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)}, \tag{2.2}$$

where $D_k$ is computed by (1.10). If $\rho_k \geq \mu$, where $\mu \in (0, 1)$ is a constant, we accept the trial step $d_k$, set $x_{k+1} = x_k + d_k$ and enlarge the trust region radius $\Delta_k$. Otherwise we set $x_{k+1} = x_k$, reduce the trust region radius and resolve the subproblem (1.2).

Now, we propose a new nonmonotone trust region algorithm as follows:

**Algorithm 2.1.** (*New nonmonotone trust region algorithm*)
**Step 0** *Choose parameters* $\eta \in (0, 1)$, $\mu \in (0, 1)$, $\Delta_0 > 0$, $0 < c_1 < 1 < c_2$. *Given an arbitrary point* $x_0 \in \Re^n$ *and a symmetric matrix* $B_0 \in \Re^{n\times n}$. *Set* $k := 0$.
**Step 1** *Compute* $g_k$. *If* $\|g_k\| = 0$, *stop. Otherwise, go to Step 2.*
**Step 2** *Compute an approximate solution* $d_k$ *so that* $\|d_k\| \leq \Delta_k$ *and (2.1) is satisfied.*
**Step 3** *Compute* $D_k$ *by (1.10), and* $\rho_k$ *by (2.2).*
**Step 4** *Set*

$$x_{k+1} = \begin{cases} x_k + d_k & if \quad \rho_k \geq \mu, \\ x_k & otherwise. \end{cases} \tag{2.3}$$

**Step 5** *Compute* $\Delta_{k+1}$ *as*

$$\Delta_{k+1} = \begin{cases} c_1\|d_k\| & if \quad \rho_k < \mu, \\ c_2\|d_k\| & if \quad \rho_k \geq \mu. \end{cases} \tag{2.4}$$

**Step 6** *Update* $B_k$. *Set* $k := k + 1$ *and go to Step 1.*

## 3. Global Convergence

In this section, we discuss the global convergence of Algorithm 2.1. Suppose an infinite sequence of iterations $\{x_k\}$ is obtained from Algorithm 2.1. Some common assumptions are as follows:

**Assumption 3.1.** *The level set* $\Omega_0 = \{x \in \Re^n | f(x) \leq f(x_0)\}$ *is bounded.*

**Assumption 3.2.** *The gradient function of* $g(x)$ *is Lipschitz continuous in* $\Omega_0$.

**Assumption 3.3.** *The matrix sequence* $\{B_k\}$ *is uniformly bounded.*

For simplicity, we define two index sets as follows:

$$I = \{k \mid \rho_k \geq \mu\} \quad \text{and} \quad J = \{k \mid \rho_k < \mu\}.$$

**Lemma 3.1.** *Suppose that the sequence $\{x_k\}$ is generated by Algorithm* 2.1. *Then the following inequality holds for all k:*

$$f_{k+1} \leq D_{k+1} \leq D_k. \tag{3.1}$$

*Proof.* Firstly we prove that (3.1) holds for all $k \in I$, i.e.,

$$f_{k+1} \leq D_{k+1} \leq D_k \quad \forall k \in I. \tag{3.2}$$

For $k \in I$, according to $\rho_k \geq \mu$, (2.1) and (2.2), we know that

$$f_{k+1} \leq D_k - \mu\tau\|g_k\| \min\{\Delta_k, \|g_k\|/\|B_k\|\}. \tag{3.3}$$

Following from (1.10) and (3.3), we obtain

$$\begin{aligned} D_{k+1} &= \eta D_k + (1-\eta) f(x_{k+1}) \\ &\leq \eta D_k + (1-\eta) D_k - \mu\tau\|g_k\| \min\{\Delta_k, \|g_k\|/\|B_k\|\} \\ &= D_k - \mu\tau\|g_k\| \min\{\Delta_k, \|g_k\|/\|B_k\|\}. \end{aligned} \tag{3.4}$$

By (1.10), if $\eta \neq 0$, we have

$$D_{k+1} - D_k = \frac{(1-\eta)(f_{k+1} - D_{k+1})}{\eta}, \tag{3.5}$$

and if $\eta = 0$, we have

$$D_{k+1} = f_{k+1}. \tag{3.6}$$

Combining (3.4) and (3.6), yields (3.2).

Secondly we prove that (3.1) holds for all $k \in J$. From Step 4 of Algorithm 2.1, we get $x_{k+1} = x_k$ and $f_{k+1} = f_k$ for all $k \in J$. First we prove that $f_{k+1} \leq D_{k+1}$. We consider two cases, respectively.

(i) If $k-1 \in I$. According to (3.2), we have $f_k \leq D_k$. Following from (1.10) and $f_{k+1} = f_k$, we can deduce that

$$D_{k+1} = \eta D_k + (1-\eta) f_{k+1} \geq \eta f_{k+1} + (1-\eta) f_{k+1} = f_{k+1}. \tag{3.7}$$

(ii) If $k-1 \in J$. In this case, we define a index set $\mathcal{F} = \{i | 1 < i \leq k, k-i \in I\}$. If $\mathcal{F} = \emptyset$, by Step 4 of Algorithm 2.1 we get $f_0 = f_{k-j} = f_{k+1}, j = 0, 1, \cdots, k-1$. Following from (1.10) we get

$$D_{k+1} = D_k = f_{k+1}. \tag{3.8}$$

Now, we suppose that $\mathcal{F} \neq \emptyset$. Let $s = \min\{i : i \in \mathcal{F}\}$. Then we have

$$f_{k+1} = f_k = f_{k-j}, \quad j = 0, 1, \cdots, s-1. \tag{3.9}$$

By (1.10), we know

$$D_k = \eta D_{k-1} + (1-\eta) f_k, \quad k \geq 1. \tag{3.10}$$

Using (3.10) repeatedly, we get

$$\eta D_k + (1-\eta) f_{k+1} = \eta^s D_{k-s+1} + \sum_{i=0}^{s-2} \eta^{i+1}(1-\eta) f_{k-i} + (1-\eta) f_{k+1}. \tag{3.11}$$

According to the definition of $\mathcal{F}$, $s$ and (3.2), we have $k-s \in I$ and $D_{k-s+1} \geq f_{k-s+1}$. Combining (3.9) and (3.11) together, we can deduce that

$$\begin{aligned}\eta D_k + (1-\eta) f_{k+1} &\geq \eta^s f_{k-s+1} + \sum_{i=0}^{s-2} \eta^{i+1}(1-\eta) f_{k-i} + (1-\eta) f_{k+1} \\ &= \big[\eta^s + \sum_{i=0}^{s-2} \eta^{i+1}(1-\eta) + (1-\eta)\big] f_{k+1} \\ &= f_{k+1}. \end{aligned} \tag{3.12}$$

Hence, it follows from (1.10) and (3.12) that

$$D_{k+1} = \eta D_k + (1-\eta) f_{k+1} \geq f_{k+1}. \tag{3.13}$$

By (3.7), (3.8) and (3.13), we conclude that

$$f_{k+1} \leq D_{k+1}, \quad \forall k \in J. \tag{3.14}$$

If $\eta \neq 0$, by (3.5) and (3.14) we obtain that $f_{k+1} \leq D_{k+1} \leq D_k$.

If $\eta = 0$, then, by (1.10) and $k \in J$, we get $D_{k+1} = f_{k+1} = f_k$. Combining $k-1 \in J$ and (3.14), we obtain that $f_k \leq D_k$. Thus (3.1) holds for all $k \in J$. The proof is completed. □

**Lemma 3.2.** *Suppose that Assumption* 3.1 *holds, then the sequence* $\{x_k\}$ *generated by Algorithm* 2.1 *is contained in the level set* $\Omega_0$.

*Proof.* Following from Lemma 3.1, Assumption 3.1 and $D_0 = f_0$, we can easily obtain the assertion. □

For convenience's sake, we refer to a successful iteration point $x_{k+1} = x_k + d_k$, and an unsuccessful iteration point if $x_{k+1} = x_k$.

**Lemma 3.3.** ([15]) *Suppose that Assumptions* 3.2 *and* 3.3 *hold, the sequence* $\{x_k\}$ *is generated by Algorithm* 2.1, *and the following inequality holds for all* $k$*:*

$$\|g_k\| \geq \epsilon, \tag{3.15}$$

*where* $\epsilon \in (0,1)$ *is a constant. Then for each* $k$, *there is a nonnegative integer* $m$ *such that* $x_{k+m+1}$ *is a successful iteration point.*

Based on the above lemmas, we establish the global convergence of Algorithm 2.1.

**Theorem 3.1.** *Suppose that Assumptions* $3.1-3.3$ *hold. Then the sequence* $\{x_k\}$ *generated by Algorithm* 2.1 *satisfies*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{3.16}$$

*Proof.* By contradiction, we suppose that there exists a constant $\epsilon \in (0,1)$ such that the following inequality holds for all $k$:

$$\|g_k\| \geq \epsilon. \tag{3.17}$$

Firstly, we prove that

$$\lim_{k\to\infty, k\in I} \Delta_k = 0. \tag{3.18}$$

From the prove of Lemma 3.3 in [15], we know that $I$ is an infinite set. If $k \in I$, then by (3.4) and (3.17), we get

$$D_{k+1} \leq D_k - \mu\tau\epsilon(1-\eta)\min\{\Delta_k, \epsilon/\|B_k\|\}. \tag{3.19}$$

From Lemma 3.1, we know that $\{D_k\}$ is nonincreasing and $f_{k+1} \leq D_{k+1}$ for all $k \geq 0$. By Assumption 3.1, Lemma 3.2 and the continuity of $f$, we know that the sequence $\{f_k\}$ is bounded below, and $\{D_k\}$ is convergent. By taking limits as $k \to \infty$ and $k \in I$ in both sides of (3.19), we have

$$\lim_{k\to\infty, k\in I} \min\{\Delta_k, \epsilon/\|B_k\|\} = 0. \tag{3.20}$$

By Assumption 3.3 and (3.20) that (3.18) holds.

Next, we prove that

$$\lim_{k\to\infty} \Delta_k = 0. \tag{3.21}$$

(i) If $J$ is a finite set, then (3.18) holds, which implies that (3.21) holds.

(ii) If $J$ is an infinite set, we define $\mathcal{F}_1 = \{i_k \mid k = 0, 1, \cdots\}$ which is a subset of $J$ satisfies:

$$\begin{aligned} &i_1 = \min\{j \mid j \in J\}, \\ &i_{k+1} = \min\{j \in J \mid j - 1 \in I, j - 1 > i_k\}, \quad \forall k \geq 1. \end{aligned}$$

According to Lemma 3.3, we know that $\mathcal{F}_1$ is an infinite set. For $k \geq 1$, by the definition of $i_k$ we know that $i_k - 1 \in I$. According to Step 5 of Algorithm 2.1, we have

$$\Delta_{i_k} \leq c_2 \Delta_{i_k - 1}. \tag{3.22}$$

The definition of $i_{k+1}$ implies that there exists at least an integer $l$ such that

$$i_k + l < i_{k+1} - 1 \quad \text{and} \quad i_k + l \in J. \tag{3.23}$$

Let $l_k$ be the maximum integer satisfies (3.23). It follows from Step 5 of Algorithm 2.1 that

$$\Delta_{i_k+l} \geq \Delta_{i_k+l+1}, \quad l = 0, 1, \cdots, l_k, \tag{3.24a}$$

$$\Delta_{i_k+l} \leq \Delta_{i_k+l+1}, \quad l = l_k + 1, l_k + 2, \cdots, i_{k+1} - i_k - 1. \tag{3.24b}$$

Following from (3.18), we see that $\Delta_{i_k-1} \to 0$ as $k \to \infty$. This fact combined with (3.22) and (3.24) implies that

$$\lim_{k\to\infty, k\in J} \Delta_k = 0. \tag{3.25}$$

Hence, it follows from (3.18) and (3.25) that (3.21) holds.

Based on the above work, we prove the theorem now. By the Taylor expansion, Assumption 3.2, $\|d_k\| \leq \Delta_k$ and (3.21), we get

$$\begin{aligned} &\left|f_k - f(x_k + d_k) - (\phi_k(0) - \phi_k(d_k))\right| \\ &= \left|\frac{1}{2} d_k^T B_k d_k - \int_0^1 [g(x_k + \xi d_k) - g_k]^T d_k d\xi\right| \\ &\leq O(\Delta_k^2 \|B_k\|) + o(\Delta_k). \end{aligned} \tag{3.26}$$

By (2.1), (3.17) and (3.26), it follows that

$$\left|\frac{f_k - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)} - 1\right| \leq \frac{O(\Delta_k^2 \|B_k\|) + o(\Delta_k)}{\tau\epsilon \min\{\Delta_k, \epsilon/\|B_k\|\}}.$$

Combining the above inequality, (3.21) and Assumption 3.3, we can deduce that

$$\lim_{k\to\infty} \frac{f_k - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)} = 1. \tag{3.27}$$

It follows from (2.2) and (3.1) that

$$\rho_k = \frac{D_k - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)} \geq \frac{f_k - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)}. \tag{3.28}$$

Thus, for $k$ large enough, according to $\mu \in (0, 1)$, (3.27) and (3.28), we have that

$$\rho_k \geq \mu.$$

From Step 5 of Algorithm 2.1, we know that $\Delta_{k+1} \geq \Delta_k$ holds for sufficiently large $k$, which contradicts (3.21). The theorem is true. □

## 4. Local Superlinear Convergence

In this section, we analyze the superlinear convergence for Algorithm 2.1 under suitable conditions. First we present the following assumptions.

**Assumption 4.1.** *$f(x)$ is twice continuously differentiable.*

**Assumption 4.2.** *The matrix $B_k$ is invertible, $\|B_k^{-1} g_k\| \leq \Delta_k$ and Algorithm 2.1 chooses the step $d_k = -B_k^{-1} g_k$ for all $k$.*

**Theorem 4.1.** *Suppose that Assumptions* 3.1, 3.3, 4.1, *and* 4.2 *hold. Suppose that the sequence $\{x_k\}$ is generated by Algorithm* 2.1 *convergence to a point $x^*$, where $\nabla^2 f(x^*)$ is positive definite and $\nabla^2 f(x)$ is Lipschitz continuous on a neighborhood of $x^*$. If*

$$\lim_{k\to\infty} \frac{\|\big(B_k - \nabla^2 f(x_k)\big) d_k\|}{\|d_k\|} = 0, \tag{4.1}$$

*then the sequence $\{x_k\}$ converges to $x^*$ superlinearly.*

*Proof.* According to Assumption 3.2 and Theorem 3.1, we know that there exists a constant $L_1 > 0$ such that

$$\|g(x_k) - g(x^*)\| \leq L_1 \|x_k - x^*\|,$$

the sequence $\{x_k\}$ is generated by Algorithm 2.1 and converges to a point $x^*$, it follows that

$$\lim_{k\to\infty} \|g(x_k) - g(x^*)\| \leq 0.$$

This implies that

$$\lim_{k\to\infty} \|g_k\| = \|g(x^*)\| = 0. \tag{4.2}$$

Then it means that $x^*$ is a strict local minimizer. Suppose that $\Omega = \{x \big| \|x - x^*\| \leq \Delta\}$, where $\Delta > 0$ is a sufficiently small constant such that $x_k \in \Omega$ for all $k \geq k_0$, where $k_0$ is a positive integer. From Assumption 4.1, we know that there exist two positive constants $m$ and $M$, such that

$$m\|d\|^2 \leq d^T \nabla^2 f(x) d \leq M\|d\|^2, \quad \forall d \in \Re^n, \tag{4.3}$$

for all $x \in \Omega$. For sufficiently large $k > k_0$, from (1.2), it follows that

$$\phi_k(0) - \phi_k(d_k) = \frac{1}{2} d_k^T \big(B_k - \nabla^2 f(x_k)\big) d_k + \frac{1}{2} d_k^T \nabla^2 f(x_k) d_k. \tag{4.4}$$

From Assumption 3.3 we have that $d_k \to 0$ as $k \to \infty$. Combining $d_k \to 0$, (4.1), (4.3) and (4.4), we know that

$$\lim_{k\to\infty} \frac{\phi_k(0) - \phi_k(d_k)}{\|d_k\|^2} \le M.$$

This implies that

$$\phi_k(0) - \phi_k(d_k) = O(\|d_k\|^2). \tag{4.5}$$

By Mean-value Theorem, it follows that

$$\begin{aligned} & f_k - f(x_k + d_k) - \big(\phi_k(0) - \phi_k(d_k)\big) \\ = & \frac{1}{2} d_k^T \big(\nabla^2 f(x_k) - \nabla^2 f(x_k + \xi d_k)\big) d_k + \frac{1}{2} d_k^T \big(B_k - \nabla^2 f(x_k)\big) d_k, \end{aligned}$$

where $\xi \in (0, 1)$. For large enough $k$, due to the Lipshitz continuity of $\nabla^2 f(x)$, (4.1) and $d_k \to 0$, it follows that

$$f_k - f(x_k + d_k) - \big(\phi_k(0) - \phi_k(d_k)\big) = o(\|d_k\|^2). \tag{4.6}$$

Combining (4.5) and (4.6), we know that

$$\left| \frac{f_k - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)} - 1 \right| = \left| \frac{f_k - f(x_k + d_k) - \phi_k(0) + \phi_k(d_k)}{\phi_k(0) - \phi_k(d_k)} - 1 \right| \le \frac{o(\|d_k\|^2)}{O(\|d_k\|^2)}. \tag{4.7}$$

By (4.7), we know (3.27) holds. Combining (3.27) and (3.28) we know that $\rho_k \ge \mu$ for sufficiently large $k$. Therefore, Algorithm 2.1 reduces to the standard quasi-Newton method when $k$ is sufficiently large and we can obtain the superlinear convergent result by following from the stand results of quasi-Newton method. The theorem is true. □

## 5. Computational Experiments

In this section, we provide some preliminary numerical experiments to show the performance of our proposed algorithm. The new nonmonotone trust region algorithm is denoted by NNTR from now on. In Algorithm 2.1, the parameter $\eta$ has a wide scope. If we take $\eta = 0$, then we can obtain the usual trust region methods described in [4](denoted by UTR). Besides, we compare Algorithm 2.1 with NTR method proposed by Mo et al. [15].

The mentioned algorithms are coded in Matlab 7.10.0 environment. All numerical computation were conducted using an Intel(R) Core(TM) 2 Duo CPU 2.20 GHZ computer with 2GB of RAM. We exploit the parameters of Algorithm 2.1 as follows:

$$\Delta_0 = 2,\ \mu = 0.25,\ c_1 = 0.25,\ c_2 = 1.25.$$

In all tests the maximum number of iterations is 300, and the termination condition is $\|g_k\| \le 10^{-6}$. In all algorithms, $B_k$ is updated by the following BFGS formula:

$$B_{k+1} = B_k + \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k^* (y_k^*)^T}{(y_k^*)^T s_k},$$

where

$$y_k^* = \frac{y_k^T s_k}{|y_k^T s_k|} y_k, \quad s_k = x_{k+1} - x_k, \quad y_k = g_{k+1} - g_k.$$

For each test, we choose the initial matrix $B_0 = |f_0|E$, where $E$ is the unit matrix. For simplicity, in the tables of test results, " Iter" denotes the numbers of iterations and " Dim" denotes the dimensions of the problems, " FV"," GV" denote the final value of $f(x_k)$ and $g(x_k)$ respectively when the algorithm terminates, " TCPU" represents the CPU time in second for solving each problem and " NF"," NG" denote the function evaluations and gradient evaluations in sequence respectively. we choose some test functions from [22] and [23].

**Example 5.1.** Extended Rosenbrock function. The test function is the 21th example of [22]. Let

$$f(x) = \sum_{i=1}^{n/2} \Big(100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2\Big).$$

The minima of the problem is $f(min) = 0$, and the standard starting point $x_0 = (-1.2, 1, \cdots, -1.2, 1)^T$.

Table 5.1: Numerical results for Example 5.1.

| Dim | NNTR($\eta = 0.2$) | | | | | NNTR($\eta = 0.5$) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Iter | FV | TCPU | NF | NG | Iter | FV | TCPU | NF | NG |
| 32 | 44 | 2.54e-016 | 0.0559 | 89 | 84 | 48 | 5.08e-018 | 0.0626 | 97 | 91 |
| 64 | 46 | 4.99e-017 | 0.0891 | 93 | 90 | 46 | 4.99e-017 | 0.0924 | 93 | 90 |
| 128 | 42 | 1.64e-016 | 0.1874 | 85 | 83 | 43 | 1.11e-018 | 0.1932 | 87 | 84 |
| 256 | 47 | 3.01e-016 | 0.7310 | 95 | 93 | 48 | 2.56e-021 | 0.7844 | 97 | 97 |
| 512 | 45 | 1.65e-019 | 3.4084 | 91 | 91 | 45 | 1.65e-019 | 3.3937 | 91 | 91 |

**Example 5.2.** Extended Powell singular function. The test function is the 22th example of [22]. Let

$$f(x) = \sum_{i=1}^{n/4} \Big((x_{4i-1} + 10x_{4i-2})^2 + 5(x_{4i-2} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^2 + 10(x_{4i-3} - x_{4i})^4\Big).$$

The minima of the problem is $f(min) = 0$, the standard starting point $x_0 = (3, -1, 0, 1, \cdots, 3, -1, 0, 1)^T$.

Table 5.2: Numerical results for Example 5.2.

| Dim | NNTR($\eta = 0.2$) | | | | | NNTR($\eta = 0.5$) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Iter | FV | TCPU | NF | NG | Iter | FV | TCPU | NF | NG |
| 32 | 50 | 2.60e-011 | 0.0617 | 101 | 101 | 50 | 2.60e-011 | 0.0596 | 101 | 101 |
| 64 | 50 | 5.44e-010 | 0.0766 | 101 | 101 | 50 | 5.44e-010 | 0.0976 | 101 | 101 |
| 128 | 62 | 4.86e-013 | 0.2241 | 125 | 125 | 62 | 4.86e-013 | 0.1811 | 125 | 125 |
| 256 | 62 | 1.43e-010 | 0.9593 | 125 | 125 | 62 | 1.43e-010 | 0.9602 | 125 | 125 |
| 512 | 68 | 1.24e-009 | 5.0646 | 137 | 137 | 68 | 1.24e-009 | 5.0746 | 137 | 137 |

**Example 5.3.** Extended Dixon test function. The test function is the Problem 4.5 of [23]. Let

$$f(x) = \sum_{i=1}^{n/10} \Big( (1 - x_{10i-9})^2 + (1 - x_{10i})^2 + \sum_{j=10i-9}^{10i-1} (x_j^2 - x_{j+1})^2 \Big).$$

The minima of the problem is $f(min) = 0$, the standard starting point $x_0 = (-2, -2, \cdots, -2, -2)^T$.

Table 5.3: Numerical results for Example 5.3.

| Dim | NNTR($\eta = 0.2$) | | | | | NNTR($\eta = 0.5$) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Iter | FV | TCPU | NF | NG | Iter | FV | TCPU | NF | NG |
| 32 | 80 | 7.40e-016 | 0.0475 | 161 | 160 | 82 | 1.02e-014 | 0.0819 | 165 | 165 |
| 64 | 85 | 4.38e-016 | 0.1415 | 171 | 171 | 85 | 4.38e-014 | 0.1378 | 171 | 171 |
| 128 | 106 | 1.09e-015 | 0.3623 | 213 | 211 | 100 | 2.97e-015 | 0.3024 | 201 | 201 |
| 256 | 114 | 1.87e-016 | 1.8159 | 229 | 229 | 114 | 1.87e-016 | 1.7932 | 229 | 229 |
| 512 | 130 | 1.38e-015 | 9.9293 | 261 | 261 | 130 | 1.38e-015 | 9.8413 | 261 | 261 |

**Example 5.4.** Broyden tridiagonal function. The test function is the 30th example of [22]. Let

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

The minima of the problem is $f(min) = 0$, the standard starting point $x_0 = (-1, -1, \cdots, -1, -1)^T$.

Table 5.4: Numerical results for Example 5.4.

| Dim | NNTR($\eta = 0.2$) | | | | | NNTR($\eta = 0.5$) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Iter | FV | TCPU | NF | NG | Iter | FV | TCPU | NF | NG |
| 32 | 33 | 4.38e-016 | 0.0182 | 67 | 67 | 33 | 4.38e-016 | 0.0572 | 67 | 67 |
| 64 | 28 | 7.47e-015 | 0.0724 | 57 | 57 | 28 | 7.47e-015 | 0.0706 | 57 | 57 |
| 128 | 37 | 8.04e-015 | 0.1233 | 75 | 75 | 37 | 8.04e-015 | 0.1712 | 75 | 75 |
| 256 | 55 | 1.01e-014 | 0.8374 | 111 | 111 | 55 | 1.01e-014 | 0.8589 | 111 | 111 |
| 512 | 81 | 8.00e-015 | 5.9847 | 163 | 163 | 81 | 8.00e-015 | 5.9419 | 163 | 163 |

**Example 5.5.** Trigonometric function. The test function is the 26th example of [22]. Let

$$f(x) = \sum_{i=1}^{n} \Big( n - \sum_{j=1}^{n} \cos x_j + i(1 - \cos x_i) - \sin x_i \Big)^2.$$

The minima of the problem is $f(min) = 0$, $x_0 = (1/2n, 1/2n, \cdots, 1/2n)^T$ is the standard starting point.

In Tables 5.1 to 5.5 we give some test results about five large scale unconstrained optimization problems to show that whether the parameter $\eta$ has impact on Algorithm 2.1, we test the five problems with two cases, say, $\eta = 0.2$ and $\eta = 0.5$. The dimensions of the problems are chosen from 32 to 512. It is shown from the tables that the numbers of iterations, the CPU time, the function evaluations and gradient evaluations in sequence didn't appear to be much

Table 5.5: Numerical results for Example 5.5.

| Dim | NNTR($\eta = 0.2$) | | | | | NNTR($\eta = 0.5$) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Iter | FV | TCPU | NF | NG | Iter | FV | TCPU | NF | NG |
| 32 | 68 | 4.24e-014 | 0.0658 | 137 | 119 | 55 | 1.66e-015 | 0.0991 | 111 | 96 |
| 64 | 86 | 1.76e-013 | 0.2987 | 173 | 152 | 74 | 1.32e-013 | 0.2884 | 149 | 130 |
| 128 | 100 | 1.24e-014 | 0.8947 | 201 | 179 | 80 | 9.20e-015 | 0.7324 | 161 | 142 |
| 256 | 177 | 5.30e-012 | 6.7765 | 355 | 321 | 173 | 5.86e-012 | 6.6022 | 347 | 314 |
| 512 | 183 | 7.52e-013 | 30.502 | 367 | 332 | 196 | 6.30e-013 | 32.668 | 393 | 356 |

Table 5.6: Numerical comparisons about the mentioned algorithms(1).

| Problem | Dim | UTR | | | NTR | | | NNTR | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Iter | FV | GV | Iter | FV | GV | Iter | FV | GV |
| 5.1 | 32 | 46 | 6.39e-017 | 3.53e-007 | 44 | 2.54e-016 | 7.12e-007 | 44 | 2.54e-016 | 7.12e-007 |
| | 64 | 55 | 2.05e-020 | 6.24e-009 | 53 | 1.53e-023 | 9.67e-011 | 46 | 4.99e-017 | 9.83e-008 |
| | 128 | 47 | 5.74e-016 | 9.35e-007 | 47 | 5.74e-016 | 9.35e-007 | 42 | 4.65e-028 | 5.73e-007 |
| | 256 | 49 | 3.10e-018 | 6.50e-008 | 49 | 6.21e-016 | 8.77e-007 | 47 | 6.78e-028 | 3.10e-007 |
| | 512 | 50 | 7.29e-017 | 3.74e-007 | 50 | 7.29e-017 | 3.74e-007 | 45 | 2.20e-025 | 1.65e-008 |
| 5.2 | 32 | 60 | 8.65e-014 | 5.14e-007 | 60 | 8.65e-014 | 5.14e-007 | 50 | 2.60e-011 | 4.20e-007 |
| | 64 | 57 | 9.21e-010 | 9.43e-007 | 57 | 9.21e-010 | 9.43e-007 | 50 | 5.44e-010 | 9.96e-007 |
| | 128 | 69 | 1.28e-013 | 2.53e-007 | 67 | 4.62e-016 | 1.34e-007 | 62 | 4.86e-013 | 2.69e-007 |
| | 256 | 60 | 1.03e-009 | 6.70e-007 | 60 | 1.03e-009 | 6.74e-007 | 62 | 1.43e-010 | 6.92e-007 |
| | 512 | 63 | 6.82e-010 | 5.24e-007 | 63 | 6.82e-010 | 5.24e-007 | 68 | 1.24e-009 | 6.16e-007 |
| 5.3 | 32 | 90 | 1.52e-017 | 1.86e-008 | 85 | 2.00e-015 | 2.28e-007 | 80 | 7.40e-016 | 1.45e-007 |
| | 64 | 97 | 4.81e-015 | 2.82e-007 | 97 | 4.81e-015 | 2.82e-007 | 85 | 4.38e-016 | 9.04e-007 |
| | 128 | 109 | 1.41e-016 | 6.14e-008 | 109 | 1.41e-016 | 6.14e-008 | 106 | 1.09e-015 | 1.15e-007 |
| | 256 | 120 | 1.06e-014 | 4.27e-007 | 120 | 1.06e-014 | 4.27e-007 | 114 | 1.87e-016 | 6.78e-008 |
| | 512 | 124 | 3.98e-014 | 8.44e-007 | 124 | 3.98e-014 | 8.44e-007 | 130 | 1.38e-015 | 1.47e-007 |
| 5.4 | 32 | 43 | 3.36e-015 | 6.07e-007 | 33 | 4.38e-016 | 1.80e-007 | 33 | 4.38e-016 | 1.80e-007 |
| | 64 | 28 | 7.47e-015 | 7.18e-009 | 28 | 7.47e-015 | 7.18e-007 | 28 | 7.47e-015 | 7.18e-007 |
| | 128 | 37 | 8.04e-015 | 6.75e-007 | 37 | 8.04e-015 | 6.75e-007 | 37 | 8.04e-015 | 6.75e-007 |
| | 256 | 55 | 1.01e-014 | 9.71e-007 | 55 | 1.01e-014 | 9.71e-007 | 55 | 1.01e-014 | 9.71e-007 |
| | 512 | 81 | 8.00e-015 | 8.17e-007 | 81 | 8.00e-015 | 8.17e-007 | 81 | 8.00e-015 | 8.17e-007 |
| 5.5 | 32 | 71 | 8.04e-014 | 5.67e-007 | 68 | 4.24e-014 | 4.12e-007 | 68 | 4.23e-014 | 4.12e-007 |
| | 64 | 121 | 9.03e-015 | 1.90e-007 | 86 | 1.76e-013 | 8.40e-007 | 86 | 1.76e-013 | 8.40e-007 |
| | 128 | 231 | 2.48e-013 | 9.96e-007 | 100 | 1.24e-014 | 2.23e-007 | 100 | 1.24e-014 | 2.23e-007 |
| | 256 | 300 | 4.26e-010 | 4.13e-005 | 177 | 5.30e-012 | 6.02e-007 | 177 | 5.30e-012 | 6.02e-007 |
| | 512 | 300 | 2.26e-008 | 2.94e-004 | 183 | 7.52e-013 | 9.46e-007 | 183 | 7.52e-013 | 9.46e-007 |

different from $\eta = 0.2$ to $\eta = 0.5$. The results imply that the parameter values of $\eta$ have slight impact on the unconstrained optimization test problems.

We can compute a variable $\eta_k$ by a formula as follows for further research:

$$\eta_k = \begin{cases} \theta\eta_0, & k = 1, \\ \theta\eta_{k-1} + (1-\theta)\eta_{k-2}, & k \geq 2, \end{cases}$$

where $\eta_0 = 0.2, 0.5$ and $0.8$, the parameter $\theta \in (0, 1)$. Other choice such that $\eta_k = 0.95$ for $k \in [1, 20]$, $\eta_k = 0.5$ for $k \in [21, 40]$, $\eta_k = 0.25$ for $k \in [41, 60]$ and $\eta_k = 0.01$ for $k \geq 61$ in [6] can also be computed. The choice for $\eta_k$ mentioned above was not used in the numerical results of this paper.

The Numerical results about the mentioned three algorithms are shown in Tables 5.6 and

Table 5.7: Numerical comparisons about the mentioned algorithms(2).

| Problem | Dim | UTR | | | NTR | | | NNTR | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | TCPU | NF | NG | TCPU | NF | NG | TCPU | NF | NG |
| 5.1 | 32 | 0.0623 | 93 | 86 | 0.0543 | 89 | 84 | 0.0559 | 89 | 84 |
| | 64 | 0.0869 | 111 | 101 | 0.0959 | 107 | 100 | 0.0891 | 93 | 90 |
| | 128 | 0.1576 | 95 | 89 | 0.1210 | 95 | 89 | 0.1874 | 85 | 83 |
| | 256 | 0.7269 | 99 | 93 | 0.7406 | 99 | 94 | 0.7310 | 95 | 93 |
| | 512 | 3.6349 | 101 | 96 | 3.6587 | 101 | 96 | 3.4084 | 91 | 91 |
| 5.2 | 32 | 0.0746 | 121 | 117 | 0.0765 | 121 | 117 | 0.0617 | 101 | 101 |
| | 64 | 0.1006 | 115 | 110 | 0.1045 | 115 | 110 | 0.0766 | 101 | 101 |
| | 128 | 0.2784 | 139 | 134 | 0.1979 | 135 | 130 | 0.2241 | 125 | 125 |
| | 256 | 0.9529 | 121 | 116 | 0.9691 | 121 | 116 | 0.9593 | 125 | 125 |
| | 512 | 4.7554 | 127 | 125 | 4.8502 | 127 | 125 | 5.0646 | 137 | 137 |
| 5.3 | 32 | 0.0784 | 181 | 177 | 0.0786 | 171 | 169 | 0.0475 | 161 | 160 |
| | 64 | 0.1235 | 195 | 192 | 0.1135 | 195 | 192 | 0.1415 | 171 | 171 |
| | 128 | 0.3312 | 219 | 214 | 0.2522 | 219 | 214 | 0.3623 | 213 | 211 |
| | 256 | 1.8305 | 241 | 236 | 1.9156 | 241 | 236 | 1.8159 | 229 | 229 |
| | 512 | 9.2297 | 249 | 245 | 9.4650 | 249 | 245 | 9.9293 | 261 | 261 |
| 5.4 | 32 | 0.0683 | 87 | 75 | 0.0555 | 67 | 67 | 0.0182 | 67 | 67 |
| | 64 | 0.0768 | 57 | 57 | 0.0802 | 57 | 57 | 0.0724 | 57 | 57 |
| | 128 | 0.1689 | 75 | 75 | 0.1828 | 75 | 75 | 0.1233 | 75 | 75 |
| | 256 | 0.8796 | 111 | 111 | 0.9228 | 111 | 111 | 0.8374 | 111 | 111 |
| | 512 | 6.1910 | 163 | 163 | 6.1125 | 163 | 163 | 5.9847 | 163 | 163 |
| 5.5 | 32 | 0.0995 | 143 | 129 | 0.1246 | 137 | 119 | 0.0658 | 137 | 119 |
| | 64 | 0.3913 | 243 | 219 | 0.2644 | 173 | 152 | 0.2987 | 173 | 152 |
| | 128 | 2.0889 | 463 | 420 | 0.8881 | 201 | 179 | 0.8947 | 201 | 179 |
| | 256 | 11.612 | 601 | 550 | 6.8966 | 355 | 321 | 6.7765 | 355 | 321 |
| | 512 | 51.597 | 601 | 550 | 31.244 | 367 | 332 | 30.502 | 367 | 332 |

5.7, including two nonmonotone trust region methods (NTR and NNTR) and a traditional monotone trust region method, i.e., the UTR method. Considering that the parameter values of $\eta$ have slight impact on the unconstrained optimization test problems, we chosen $\eta = 0.2$ for NNTR method to compare with the other two methods.

Considering that NTR and UTR are two nonmonotone trust region methods without line search, we compare NNTR with a trust region method with line search, which seems more convincingly. The nonmonotone trust region algorithm with Armijo line search is denoted by BLS from now on, i.e., when the direction $d_k$ was not accepted, then we choose the step length to satisfy the Armijo line search rule. The BLS method can regard as the special case of [17]. In our experiments, unless otherwise stated, the other parameters in UTR and NTR methods were set as the NNTR method.

It is shown from Tables 5.6 and 5.7 that NNTR method performs better than UTR and NTR method since the new algorithm needs less numbers of iterations, less function and gradient evaluation in most cases. Besides, NNTR uses the least CPU time for solving almost the test problems. In Table 5.8 BLS seems perform better than NNTR in most cases. Considering that trust region with line search can speed up the rate of convergence and NNTR performs better than BLS sometimes. Therefore, we can deduce that the new nonmonotone algorithm is more robust than the other mentioned algorithms, which indicates that extending trust region method from monotone to nonmonotone is meaningful for solving large scale unconstrained optimization

Table 5.8: Numerical comparisons about the mentioned algorithms.

| Example | Dim | NNTR | | | | BLS | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Iter | FV | TCPU | NF | Iter | FV | TCPU | NF |
| 5.2 | 32 | 50 | 2.60e-011 | 0.0617 | 101 | 49 | 1.33e-015 | 0.0749 | 98 |
| | 64 | 50 | 5.44e-010 | 0.0766 | 101 | 50 | 2.50e-012 | 0.0498 | 95 |
| | 128 | 62 | 4.86e-013 | 0.2241 | 125 | 64 | 2.22e-010 | 0.2445 | 128 |
| | 256 | 62 | 1.43e-010 | 0.9593 | 125 | 57 | 2.94e-014 | 0.9663 | 104 |
| | 512 | 68 | 1.24e-009 | 5.0646 | 137 | 63 | 3.25e-010 | 4.4531 | 119 |
| 5.3 | 32 | 80 | 7.40e-016 | 0.0475 | 161 | 73 | 1.33e-015 | 0.0749 | 164 |
| | 64 | 85 | 4.38e-016 | 0.1415 | 171 | 81 | 2.50e-011 | 0.1498 | 173 |
| | 128 | 106 | 1.09e-015 | 0.3623 | 213 | 101 | 3.35e-012 | 0.3516 | 198 |
| | 256 | 114 | 1.87e-016 | 1.8159 | 229 | 109 | 4.91e-015 | 1.7621 | 227 |
| | 512 | 130 | 1.38e-015 | 9.9293 | 261 | 136 | 5.31e-015 | 9.9442 | 273 |
| 5.5 | 32 | 68 | 4.23e-014 | 0.0658 | 137 | 70 | 5.63e-016 | 0.1179 | 138 |
| | 64 | 86 | 1.76e-013 | 0.2987 | 173 | 78 | 4.45e-012 | 0.2382 | 166 |
| | 128 | 100 | 1.24e-014 | 0.8947 | 201 | 101 | 4.52e-016 | 0.8094 | 178 |
| | 256 | 177 | 5.30e-012 | 6.7765 | 355 | 170 | 4.82e-011 | 6.5065 | 351 |
| | 512 | 183 | 7.52e-013 | 30.502 | 367 | 189 | 3.88e-011 | 32.251 | 375 |

problems. In a word, the proposed new algorithm is efficient and robust for solving large scale unconstrained optimization problems.

## 6. Conclusions

In this paper, we proposed a new nonmonotone trust region algorithm based on the nonmonotone line search proposed by Gu and Mo [6]. By resetting the ratio $\rho_k$ for evaluation the trial step, the new algorithm is developed. Theoretical analysis shows that the new algorithm inherits the global convergence of the traditional trust region method. Under suitable conditions the superlinear convergence of the algorithm are proved. Preliminary numerical experiments indicate that our algorithm is quite effective for large scale unconstrained optimization problems.

**Acknowledgments.** This work has been partially supported by National Natural Science Foundation of China (11071041,11201074), Fujian Natural Science Foundation (2013J01006) and R&D of Key Instruments and Technologies for Deep Resources Prospecting (the National R&D Projects for Key Scientific Instruments) under grant number ZDYZ2012-1-02-04.

## References

[1] M.J.D. Powell, Convergence properties of a class minimization algorithms, in: O.L.Mangasarian, R.R.Meyer, S.M.Robubsib (Eds.), Nonlinear programming, vol. 2, Academic Press, New York, (1975), 1-25.

[2] J. Nocedal, Y. Yuan, Combining trust region and line search techniques, in: Y.Yuan (Ed), Advandances in Nonlinear Programming, Kluwer Academic Publishers, Dordrecht, (1996), 153-175.

[3] E.M. Gertz, Combination trust-region line search methods for unconstrained optimization, University of California, San Diego, 1999.

[4] N.Y. Deng, Y. Xiao, F.J. Zhou, Nonmonotic trust region algorithm, *J. Optim Theory Appl.*, **76**:2 (1993), 259-285.

[5] L. Grippo, F. Lampariello, S. Lucidi, A nonmonotone line search technique for Newton's method, *SIAM J. Numer. Anal.*, **23**:4 (1986), 707-716.
[6] N.Z. Gu, J.T. Mo, Incorporating nonmonotone strategies into the trust region method for unconstrained optimization, *Appl. Math. Comput.*, **55** (2008), 2158-2172.
[7] W.Y. Sun, J.Y. Han, J. Sun, On the global convergence of nonmonotone descent methods, *J. Comput. Appl. Math.*, **146** (2002), 89-98.
[8] Y.H. Dai, A nonmonotone conjugate gradient algorithm for unconstrained optimization, *J. Systems Sci. Complex*, **15**:2 (2002), 139-145.
[9] Y.H. Dai, On the nonmonotone line search, *J. Optim. Theory Appl.*, **112** (2002), 315-330.
[10] L. Grippo, M. Sciandrone, Nonmonotone globalization techniques for the Barzilai-Borwein gradient method, *Comput. Optom. Appl.*, **23** (2002), 143-169.
[11] Changfeng Ma, Xiaohong Chen, The convergence of a one-step smoothing Newton method for $P_0$-NCP based on a new smoothing NCP-function, *Journal of Computational and Applied Mathematics*, **216** (2008), 1-13.
[12] Changfeng Ma, Lihua Jiang, Some research on Levenberg-Marquardt method for the nonlinear equations, *Applied Mathematics and Computation*, **184** (2007), 1032-1040.
[13] Changfeng Ma, Lihua Jiang and Desheng Wang, The convergence of a smoothing damped Gauss-Newton method for nonlinear complementarity problem, Nonlinear Analysis: Real World Applications, **10**(2009), 2072-2087.
[14] G. Liu, J. Han, D. Sun, Global convergence of BFGS algorithm with nonmonotone linesearch, *Optimization*, **34** (1995), 147-159.
[15] J.T. Mo, C.Y. Liu, S.C. Yan, A nonmonotone trust region method based on nonincreasing technique of weighted average of the successive function values, *J. Comput. Appl. Math.*, **209** (2007), 97-108.
[16] W.Y. Sun. Nonmonotone trust region method for solving optimization problems, *Appl. Math. Comput.*, **156**:1 (2004), 159-174.
[17] L. Toint, An assessment of non-monotone line search techniques for unconstrained optimization, *SIAM J. Sci. Statist. Comput.*, **17**:3 (1996), 725-739.
[18] H. Zhang, W.W. Hager, A nonmonotone line search technique and its application to unconstrained optimization, *SIAM J. Optim.*, **14**:4 (2004), 1043-1056.
[19] J. Fu, W. Sun, Nonmotone adaptive trust region method for unconstrained optimization problems, *Appl. Math. Comput.*, **163**:1 (2005), 489-504.
[20] X. Ke, J. Han, A class of nonmotone trust region algorithms for unconstrained optimization, *Sci. China Ser. A*, **41**:9 (1998), 927-932.
[21] M. Ahookhosh, K. Amini, A nonmotone trust region method with adaptive radius for unconstrained optimization, *Comput. Math. Appl.*, **60** (2010), 411-422.
[22] J.J. More, B.S. Garbow, K.E. Hillstrome, Testing unconstrained optimization software, *ACM Trans. Math. Software*, **7**:1 (1981), 17-41.
[23] D. Touati-Ahmed, C. Storey, Efficient hybrid conjugate gradient techniques, *J. Optim. Theory Appl.*, **64**:2 (1990), 379-397.