RESEARCH ARTICLE
doi: 10.2306/scienceasia1513-1874.2018.44.027

ScienceAsia 44 (2018): **27–33**

A class of nonmonotone trust region algorithm for solving unconstrained nonlinear optimization problems

Fulan Ye, Yang You, Zhen Chen, Baoguo Chen*

Research Centre for Science Technology and Society,
Fuzhou University of International Studies and Trade, Fuzhou 350202, China

*Corresponding author, e-mail: chenbg123@163.com

Received 10 Jun 2017
Accepted 15 Oct 2017

ABSTRACT: Based on the nonmonotone line search technique proposed by Gu and Mo a nonmonotone trust region algorithm is proposed for solving unconstrained nonlinear optimization problems. The new algorithm is resets the ratio ρ_k for evaluating whether the trial step d_k is acceptable. The global and superlinear convergence of the algorithm are proved under suitable conditions. Numerical results show that the new algorithm is effective.

KEYWORDS: global convergence, superlinear convergence, numerical experiment

MSC2010: 90C30

INTRODUCTION

We consider the unconstrained nonlinear optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f:\mathbb{R}^n\to\mathbb{R}$ is a continuously differentiable nonlinear function. The line search method and trust region method are the two principal methods for solving (1). The line search method produces a sequence x_0, x_1, ..., where x_{k+1} is generated from the current approximate solution x_k, and the specific direction d_k and a step size $\alpha_k>0$ by the rule $x_{k+1}=x_k+\alpha_k d_k$. The trust region methods obtain a trial step d_k by solving the quadric program subproblem

$$\min_{d\in\mathbb{R}^n}\ \phi_k(d)=g_k^{\mathrm{T}}d+\frac{1}{2}d^{\mathrm{T}}B_k d \text{ s.t. } \|d\| \leqslant \Delta_k, \tag{2}$$

where $g_k=\nabla f(x_k)$, $B_k\in\mathbb{R}^{n\times n}$ 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

$$\rho_k=\frac{f(x_k)-f(x_k+d_k)}{\phi_k(0)-\phi_k(d_k)}. \tag{3}$$

The trial step d_k is accepted whenever ρ_k is greater than a positive constant μ. Then we can set the new point $x_{k+1}=x_k+d_k$ and enlarge the trust region radius. Otherwise, the traditional trust region methods resolve the subproblem (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, and a much lower computational cost than the traditional trust region methods[1]. Some theoretical and numerical results of these trust region methods with line search are also interesting. However, all these methods enforce a monotonic decrease in the objective function values at each iteration and this may slow the convergence rate in the minimization process[2]. To overcome the shortcomings, the earliest nonmonotone line search framework was developed by Grippo et al[3] for Newton's method. In their approach, parameters λ_1, λ_2, σ and β are introduced, where $0<\lambda_1<\lambda_2$, β, $\sigma\in(0,1)$ 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

$$f(x_k+d_k)\leqslant \max_{0\leqslant j\leqslant m_k} f(x_{k-j})+\beta\alpha_k\nabla f(x_k)^{\mathrm{T}}d_k, \tag{4}$$

the memory variable m_k is a nondecreasing integer by setting

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

where M is a prefixed nonnegative integer. From then on, studies in nonlinear optimization have paid great attentions to it[4,5]. Deng et al[2] made some changes in the common ratio (3) by resetting the rule as follows:

$$\hat{\rho}_k = \frac{\max_{0\leqslant j\leqslant m_k} f(x_{k-j}) - f(x_k+d_k)}{\phi_k(0)-\phi_k(d_k)}. \quad (5)$$

The ratio (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[6,7].

Although the nonmonotone technique (4) has many advantages, Zhang and Hager[8] proposed a new nonmonotone line search algorithm, which had the same general form as the scheme of Grippo et al[3] except that their 'max' was replaced by an average of function values. The nonmonotone line search found a step length β to satisfy the inequality

$$f(x_k+\beta d_k) \leqslant C_k + \delta\beta\nabla f(x_k)^{\mathrm{T}} d_k, \quad (6)$$

where

$$C_k = \begin{cases} f(x_k), & k=0, \\ (\eta_{k-1}Q_{k-1}C_{k-1} + f(x_k))/Q_k, & k\geqslant 1, \end{cases} \quad (7)$$

and

$$Q_k = \begin{cases} 1, & k=0, \\ \eta_{k-1}Q_{k-1}+1, & k\geqslant 1, \end{cases} \quad (8)$$

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

Observe that C_{k+1} is a convex combination of 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 $f(x_0)$, $f(x_1),\ldots,f(x_k)$. From (7), the degree of nonmonotonicity and (8) depend on the choice η_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 η_k and Q_k at each k in practice. Recently Gu and Mo[9] developed an algorithm that combines a new nonmonotone technique and trust region method for unconstrained optimization problems. The new nonmonotone line search is as follows:

$$f(x_k+\beta d_k) \leqslant D_k + \delta\beta\nabla f(x_k)^{\mathrm{T}} d_k, \quad (9)$$

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

$$D_k = \begin{cases} f(x_k), & k=0, \\ \eta D_{k-1} + (1-\eta) f(x_k), & k\geqslant 1, \end{cases} \quad (10)$$

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

In this paper, we develop an algorithm which resets the ratio ρ_k in the trust region method for unconstrained optimization problems. The algorithm does not restrict one to having a monotonic decrease in 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.

NEW NONMONOTONE TRUST REGION ALGORITHM

For convenience, we denote $f(x_k)$ by f_k and $g(x_k)$ by g_k, where $g(x_k)\in\mathbb{R}^n$ is the gradient of f evaluated at x_k. The trial step d_k is obtained by solving (2) at each iteration. We solve (2) such that $\|d_k\| \leqslant \Delta_k$ and

$$\phi_k(0)-\phi_k(d_k) \geqslant \tau\|g_k\|\min\{\Delta_k, \|g_k\|/\|B_k\|\}, \quad (11)$$

where $\tau\in(0,1)$ is a constant. Clearly if B_k is a symmetric and positive definite diagonal matrix, we can obtain the solution d_k easily. More precisely, if $\|B_k^{-1}g_k\| \leqslant \Delta_k$, then $d_k = -B_k^{-1}g_k$ is the optimal solution of (2); otherwise, if $\|B_k^{-1}g_k\| > \Delta_k$, we choose the optimal solution

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

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 ρ_k between the actual reduction, $D_k - f(x_k+d_k)$, and the predicted reduction, $\phi_k(0)-\phi_k(d_k)$, as follows:

$$\rho_k = \frac{D_k - f(x_k+d_k)}{\phi_k(0)-\phi_k(d_k)}, \quad (12)$$

where D_k is computed from (10). If $\rho_k \geqslant \mu$, 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 Δ_k. Otherwise we set $x_{k+1} = x_k$, reduce the trust region radius, and re-solve (2).

We now propose the following new nonmonotone trust region algorithm.

Algorithm 1

Step 1: Choose parameters $\eta \in (0,1)$, $\mu \in (0,1)$, $\Delta_0 > 0$, $0 < c_1 < 1 < c_2$. Given an arbitrary point $x_0 \in \mathbb{R}^n$ and a symmetric matrix $B_0 \in \mathbb{R}^{n\times n}$. Set $k := 0$.

Step 2: Compute g_k. If $\|g_k\| = 0$, stop. Otherwise, go to Step 2.

Step 3: Compute an approximate solution d_k so that $\|d_k\| \leqslant \Delta_k$ and (11) is satisfied.

Step 4: Compute D_k by (10), and ρ_k by (12).

Step 5: Set

$$x_{k+1} = \begin{cases} x_k + d_k, & \rho_k \geqslant \mu, \\ x_k, & \text{otherwise.} \end{cases} \tag{13}$$

Step 6: Compute Δ_{k+1} as

$$\Delta_{k+1} = \begin{cases} c_1\|d_k\| & if\ \ \rho_k < \mu, \\ c_2\|d_k\| & if\ \ \rho_k \geqslant \mu. \end{cases} \tag{14}$$

Step 7: Update B_k. Set $k := k+1$ and go to Step 2.

GLOBAL CONVERGENCE

In this section, we discuss the global convergence of Algorithm 1. Suppose an infinite sequence of iterations $\{x_k\}$ is obtained from Algorithm 1. Some common assumptions are as follows:

(A_1) The level set $\Omega_0 = \{x \in \mathbb{R}^n \mid f(x) \leqslant f(x_0)\}$ is bounded.

(A_2) The gradient function of $g(x)$ is Lipschitz continuous in Ω_0.

(A_3) The matrix sequence $\{B_k\}$ is uniformly bounded.

For simplicity, we define two index sets as follows:

$$I = \{k \mid \rho_k \geqslant \mu\},\ \ J = \{k \mid \rho_k < \mu\}.$$

Lemma 1 *Suppose that the sequence $\{x_k\}$ is generated by Algorithm 1. Then the following inequality holds for all k:*

$$f_{k+1} \leqslant D_{k+1} \leqslant D_k. \tag{15}$$

Proof: Firstly we prove that (15) holds for all $k \in I$, i.e.,

$$f_{k+1} \leqslant D_{k+1} \leqslant D_k \quad \forall k \in I. \tag{16}$$

For $k \in I$, according to $\rho_k \geqslant \mu$, (11) and (12), we know that

$$f_{k+1} \leqslant D_k - \mu\tau\|g_k\| \min\{\Delta_k, \|g_k\|/\|B_k\|\}. \tag{17}$$

From (10) and (17), we obtain

$$\begin{aligned} D_{k+1} &= \eta D_k + (1-\eta) f(x_{k+1}) \\ &\leqslant \eta D_k + (1-\eta) D_k \\ &\quad - \mu\tau\|g_k\| \min\{\Delta_k, \|g_k\|/\|B_k\|\} \\ &= D_k - \mu\tau\|g_k\| \min\{\Delta_k, \|g_k\|/\|B_k\|\}. \end{aligned} \tag{18}$$

By (10), if $\eta \neq 0$, we have

$$D_{k+1} - D_k = \frac{(1-\eta)(f_{k+1} - D_{k+1})}{\eta}, \tag{19}$$

and if $\eta = 0$, we have

$$D_{k+1} = f_{k+1}. \tag{20}$$

Combining (18)–(20), it follows that (16) holds.

Secondly, we prove that (15) holds for all $k \in J$. From Step 4 of Algorithm 1, we obtain $x_{k+1} = x_k$ and $f_{k+1} = f_k$ for all $k \in J$. First we prove that $f_{k+1} \leqslant D_{k+1}$. We consider two cases, respectively.

(i) If $k-1 \in I$. According to (16), we have $f_k \leqslant D_k$. Following from (10) and $f_{k+1} = f_k$, we can deduce that

$$\begin{aligned} D_{k+1} &= \eta D_k + (1-\eta) f_{k+1} \\ &\geqslant \eta f_{k+1} + (1-\eta) f_{k+1} = f_{k+1}. \end{aligned} \tag{21}$$

(ii) If $k-1 \in J$. In this case, we define an index set $\mathscr{F} = \{i \mid 1 < i \leqslant k, k-i \in I\}$. If $\mathscr{F} = \varnothing$, by Step 4 of Algorithm 1 we obtain $f_0 = f_{k-j} = f_{k+1}$, $j = 0, 1, \ldots, k-1$. From (10) we obtain

$$D_{k+1} = D_k = f_{k+1}. \tag{22}$$

We now suppose that $\mathscr{F} \neq \varnothing$. Let $s = \min\{i : i \in \mathscr{F}\}$. Then we have

$$f_{k+1} = f_k = f_{k-j},\ j = 0, 1, \ldots, s-1. \tag{23}$$

By (10) we have

$$D_k = \eta D_{k-1} + (1-\eta) f_k,\ k \geqslant 1. \tag{24}$$

Using (24) repeatedly we obtain

$$\begin{aligned} \eta D_k + (1-\eta) f_{k+1} &= \eta^s D_{k-s+1} \\ &\quad + \sum_{i=0}^{s-2} \eta^{i+1}(1-\eta) f_{k-i} + (1-\eta) f_{k+1}. \end{aligned} \tag{25}$$

According to the definition of $\mathscr{F}$, s and (16), we have $k-s \in I$ and $D_{k-s+1} \geqslant f_{k-s+1}$. Combining (23) and (25) we deduce that

$$\begin{aligned}
&\eta D_k + (1-\eta) f_{k+1} \\
&\quad \geqslant \eta^s f_{k-s+1} + \sum_{i=0}^{s-2} \eta^{i+1}(1-\eta) f_{k-i} + (1-\eta) f_{k+1} \\
&\quad = \Big[\eta^s + \sum_{i=0}^{s-2} \eta^{i+1}(1-\eta) + (1-\eta)\Big] f_{k+1} \\
&\quad = f_{k+1}. \qquad (26)
\end{aligned}$$

Hence it follows from (10) and (26) that

$$D_{k+1} = \eta D_k + (1-\eta) f_{k+1} \geqslant f_{k+1}. \qquad (27)$$

By (21), (22) and (27), we conclude that

$$f_{k+1} \leqslant D_{k+1}, \quad \forall k \in J. \qquad (28)$$

If $\eta \neq 0$, by (19) and (28) we obtain that $f_{k+1} \leqslant D_{k+1} \leqslant D_k$. If $\eta = 0$, then, by (10) and $k \in J$, we obtain $D_{k+1} = f_{k+1} = f_k$. Combining $k-1 \in J$ and (28), we obtain that $f_k \leqslant D_k$. Thus (15) holds for all $k \in J$. □

Lemma 2 *Suppose that* A_1 *holds. Then the sequence* $\{x_k\}$ *generated by Algorithm 1 is contained in the level set* Ω_0.

Proof: From Lemma 1, A_1 and $D_0 = f_0$, we can easily obtain the assertion. □

For convenience, we say an iteration point is successful if $x_{k+1} = x_k + d_k$, and unsuccessful if $x_{k+1} = x_k$.

Lemma 3 (Ref. 10) *Suppose that* A_2 *and* A_3 *hold, the sequence* $\{x_k\}$ *is generated by Algorithm 1, and the following inequality holds for all k:*

$$\|g_k\| \geqslant \epsilon, \qquad (29)$$

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 1.

Theorem 1 *Suppose that* A_1–A_3 *hold. Then the sequence* $\{x_k\}$ *generated by Algorithm 1 satisfies*

$$\liminf_{k\to\infty} \|g_k\| = 0. \qquad (30)$$

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

$$\|g_k\| \geqslant \epsilon. \qquad (31)$$

Firstly, we prove that

$$\lim_{k\to\infty,\, k\in I} \Delta_k = 0. \qquad (32)$$

From the proof of Lemma 3 in Ref. 10, we know that I is an infinite set. If $k \in I$, then by (18) and (31), we obtain

$$D_{k+1} \leqslant D_k - \mu\tau\epsilon(1-\eta)\min\{\Delta_k, \epsilon/\|B_k\|\}. \qquad (33)$$

From Lemma 1, we know that $\{D_k\}$ is nonincreasing and $f_{k+1} \leqslant D_{k+1}$ for all $k \geqslant 0$. By A_1, Lemma 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 (33), we have

$$\lim_{k\to\infty,\, k\in I} \min\{\Delta_k, \epsilon/\|B_k\|\} = 0. \qquad (34)$$

By A_3 and (34) we see that (32) holds.

Next, we prove that

$$\lim_{k\to\infty} \Delta_k = 0. \qquad (35)$$

(i) If J is a finite set, then (32) holds, which implies that (35) holds.

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

$$i_1 = \min\{j \mid j \in J\},$$

and

$$i_{k+1} = \min\{j \in J \mid j-1 \in I, j-1 > i_k\}, \quad \forall k \geqslant 1.$$

According to Lemma 3, we know that $\mathscr{F}_1$ is an infinite set. For $k \geqslant 1$, by the definition of i_k we know that $i_k - 1 \in I$. According to Step5 of Algorithm 1, we have

$$\Delta_{i_k} \leqslant c_2 \Delta_{i_k-1}. \qquad (36)$$

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

$$i_k + l < i_{k+1} - 1, \quad i_k + l \in J. \qquad (37)$$

Let l_k be the maximum integer satisfying (37). It follows from Step 5 of Algorithm 1 that

$$\Delta_{i_k+l} \geqslant \Delta_{i_k+l+1}, \quad l = 0, 1, \ldots, l_k, \qquad (38)$$

and

$$\Delta_{i_k+l} \leqslant \Delta_{i_k+l+1}, \quad l = l_k+1, l_k+2, \ldots, i_{k+1}-i_k-1.$$

From (32), we see that $\Delta_{i_k-1} \to 0$ as $k \to \infty$. This fact combined with (36) and (38) implies that

$$\lim_{k\to\infty,\, k\in J} \Delta_k = 0. \tag{39}$$

Hence, it follows from (32) and (39) that (35) holds.

We now prove the theorem. From the Taylor expansion of $f(x)$, A_2, $\|d_k\| \leqslant \Delta_k$, and (35), we obtain

$$\begin{aligned} &\left|f_k - f(x_k+d_k) - (\phi_k(0)-\phi_k(d_k))\right| \\ &\quad = \left|\frac{1}{2} d_k^{\mathrm{T}} B_k d_k - \int_0^1 [g(x_k+\xi d_k) - g_k]^{\mathrm{T}} d_k \,\mathrm{d}\xi\right| \\ &\quad \leqslant O(\Delta_k^2 \|B_k\|) + o(\Delta_k). \end{aligned} \tag{40}$$

By (11), (31) and (40), it follows that

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

Combining the above inequality, (35) and A_3, we deduce that

$$\lim_{k\to\infty} \frac{f_k - f(x_k+d_k)}{\phi_k(0)-\phi_k(d_k)} = 1. \tag{41}$$

It follows from (12) and (15) that

$$\rho_k = \frac{D_k - f(x_k+d_k)}{\phi_k(0)-\phi_k(d_k)} \geqslant \frac{f_k - f(x_k+d_k)}{\phi_k(0)-\phi_k(d_k)}. \tag{42}$$

Thus for k large enough, according to $\mu \in (0,1)$, (41) and (42), we have that

$$\rho_k \geqslant \mu.$$

From Step 5 of Algorithm 1, we know that $\Delta_{k+1} \geqslant \Delta_k$ holds for sufficiently large k, which contradicts (35). □

LOCAL SUPERLINEAR CONVERGENCE

In this section, we analyse the superlinear convergence of Algorithm 1 under suitable conditions. First we present the following assumptions.

(A_4) $f(x)$ is twice continuously differentiable.

(A_5) The matrix B_k is invertible, $\|B_k^{-1} g_k\| \leqslant \Delta_k$, and Algorithm 1 chooses the step $d_k = -B_k^{-1} g_k$ for all k.

Theorem 2 *Suppose that A_1, A_3, A_4, A_5 hold. Suppose that the sequence $\{x_k\}$ is generated by Algorithm 1 and converges to a point x^*, where $\nabla^2 f(x^*)$ is positive definite and $\nabla^2 f(x)$ is Lipschitz continuous on a neighbourhood of x^*. If*

$$\lim_{k\to\infty} \frac{\|(B_k - \nabla^2 f(x_k))d_k\|}{\|d_k\|} = 0, \tag{43}$$

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

Proof: According to A_2 and Theorem 1, we know that there exists a constant $L_1 > 0$ such that

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

and the sequence $\{x_k\}$ is generated by Algorithm 1 and converges to a point x^*. It follows that

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

This implies that

$$\lim_{k\to\infty} \|g_k\| = \|g(x^*)\| = 0. \tag{44}$$

Then it means that x^* is a strict local minimizer. Suppose that $\Omega = \{x \mid \|x - x^*\| \leqslant \Delta\}$, where $\Delta > 0$ is a sufficiently small constant such that $x_k \in \Omega$ for all $k \geqslant k_0$, where k_0 is a positive integer. From A_4, we know that there exist two positive constants m and M, such that

$$m\|d\|^2 \leqslant d^{\mathrm{T}} \nabla^2 f(x) d \leqslant M\|d\|^2, \quad \forall d \in \mathbb{R}^n, \tag{45}$$

for all $x \in \Omega$. For sufficiently large $k > k_0$, from (2), it follows that

$$\phi_k(0) - \phi_k(d_k) = \frac{1}{2} d_k^{\mathrm{T}} (B_k - \nabla^2 f(x_k)) d_k + \frac{1}{2} d_k^{\mathrm{T}} \nabla^2 f(x_k) d_k. \tag{46}$$

From A_3 we have that $d_k \to 0$ as $k \to \infty$. Combining $d_k \to 0$, (43), (45), and (46), we know that

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

This implies that

$$\phi_k(0) - \phi_k(d_k) = O(\|d_k\|^2). \tag{47}$$

By the mean-value theorem, it follows that

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

where $\xi \in (0,1)$. For large enough k, from the Lipschitz continuity of $\nabla^2 f(x)$, (43) and $d_k \to 0$, it follows that

$$f_k - f(x_k + d_k) - (\phi_k(0) - \phi_k(d_k)) = o(\|d_k\|^2). \quad (48)$$

Combining (47) and (48), we know that

$$\left|\frac{f_k - f(x_k + d_k)}{\phi_k(0) - \phi_k(d_k)} - 1\right| = \left|\frac{f_k - f(x_k + d_k) - \phi_k(0) + \phi_k(d_k)}{\phi_k(0) - \phi_k(d_k)} - 1\right| \leqslant \frac{o(\|d_k\|^2)}{O(\|d_k\|^2)}. \quad (49)$$

By (49), we know that (41) holds. Combining (41) and (42) we know that $\rho_k \geqslant \mu$ for sufficiently large k. Hence Algorithm 1 reduces to the standard quasi-Newton method when k is sufficiently large and we can obtain the superlinear convergence result by using the standard results of the quasi-Newton method. □

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 1, the parameter η has a wide scope. If we take $\eta = 0$, then we can obtain the usual trust region methods described in Ref. 2 (denoted by UTR). We also compare Algorithm 1 with the nonmonotone trust region (NTR) method proposed by Mo et al[10].

The mentioned algorithms were coded in MATLAB 7.1. All numerical computation were conducted using an Intel Core 2 Duo CPU 2.20 GHz computer with 2 GB of RAM. For Algorithm 1 we used $\Delta_0 = 2$, $\mu = 0.25$, $c_1 = 0.25$, $c_2 = 1.25$.

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

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

where

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

For each test, we choose the initial matrix $B_0 = |f_0|E$, where E is the unit matrix.

Table 1 Numerical results for Example 1.

Dim	NNTR ($\eta = 0.2$)				
	Iter	FV	TCPU	NF	NG
32	44	2.54×10^{-16}	0.0559	89	84
64	46	4.99×10^{-17}	0.0891	93	90
128	42	1.64×10^{-16}	0.1874	85	83
256	47	3.01×10^{-16}	0.7310	95	93
512	45	1.65×10^{-19}	3.4084	91	91

Iter = number of iterations; Dim = number of dimensions; FV = final value of $f(x_k)$; TCPU = CPU time (s); number of function evaluations; number of gradient evaluations

Table 2 Numerical results for Example 2.

Dim	NNTR ($\eta = 0.2$)				
	Iter	FV	TCPU	NF	NG
32	50	2.60×10^{-11}	0.0617	101	101
64	50	5.44×10^{-10}	0.0766	101	101
128	62	4.86×10^{-13}	0.2241	125	125
256	62	1.43×10^{-10}	0.9593	125	125
512	68	1.24×10^{-9}	5.0646	137	137

Example 1 Extended Rosenbrock function. The test function is the 21st example of Ref. 11. Let

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

The minimum of the problem is $f(\min) = 0$. The standard starting point is $x_0 = (-1.2, 1, \ldots, -1.2, 1)$.

Example 2 Extended Powell singular function. The test function is the 22nd example of Ref. 11. Let

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

The minimum of the problem is $f_{\min} = 0$. The standard starting point is $x_0 = (3, -1, 0, 1, \ldots, 3, -1, 0, 1)$.

Example 3 Extended Dixon test function. The test function is problem 4.5 of Ref. 12. Let

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

Table 3 Numerical results for Example 3.

Dim	NNTR ($\eta = 0.2$)				
	Iter	FV	TCPU	NF	NG
32	80	7.40×10^{-16}	0.0475	161	160
64	85	4.38×10^{-16}	0.1415	171	171
128	106	1.09×10^{-15}	0.3623	213	211
256	114	1.87×10^{-16}	1.8159	229	229
512	130	1.38×10^{-15}	9.9293	261	261

Table 4 Numerical results for Example 4.

Dim	NNTR ($\eta = 0.2$)				
	Iter	FV	TCPU	NF	NG
32	33	4.38×10^{-16}	0.0182	67	67
64	28	7.47×10^{-15}	0.0724	57	57
128	37	8.04×10^{-15}	0.1233	75	75
256	55	1.01×10^{-14}	0.8374	111	111
512	81	8.00×10^{-15}	5.9847	163	163

The minimum of the problem is $f(\min) = 0$. The standard starting point is $x_0 = (-2, -2, \ldots, -2, -2)$.

Example 4 Broyden tridiagonal function. The test function is the 30th example of Ref. 11. Let

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

The minimum of the problem is $f(\min) = 0$. The standard starting point is $x_0 = (-1, -1, \ldots, -1, -1)$.

In tables 1–4 we give some test results about five large scale unconstrained optimization problems to show whether the parameter η has an impact on Algorithm 1. We test the five problems with two cases. The dimensions of the problems are chosen from 32–512.

CONCLUSIONS

In this paper, we proposed a new nonmonotone trust region algorithm based on the nonmonotone line search proposed by Gu and Mo[9]. 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 is proved. Preliminary numerical experiments indicate that our algorithm is quite effective for large scale unconstrained optimization problems.

Acknowledgements: This work was supported by National Social Science Foundation of China (Grant No. 16BKS132). The authors thank the anonymous referees for their constructive comments and suggestions.

REFERENCES

1. Gertz EM (1999) *Combination Trust-region Line Search Methods for Unconstrained Optimization*, Univ of California, San Diego.
2. Deng NY, Xiao Y, Zhou FJ (1993) Nonmonotic trust region algorithm. *J Optim Theor Appl* **76**, 259–85.
3. Grippo L, Lampariello F, Lucidi S (1986) A nonmonotone line search technique for Newton's method. *SIAM J Numer Anal* **23**, 707–16.
4. Sun WY, Han JY, Sun J (2002) On the global convergence of nonmonotone descent methods. *J Comput Appl Math* **146**, 89–98.
5. Grippo L, Sciandrone M (2002) Nonmonotone globalization techniques for the Barzilai-Borwein gradient method. *Comput Optim Appl* **23**, 143–69.
6. Dai YH (2002) On the nonmonotone line search. *J Optim Theor Appl* **112**, 315–30.
7. Toint PL (1996) An assessment of non-monotone line search techniques for unconstrained optimization. *SIAM J Sci Stat Comput* **17**, 725ïij⬡39.
8. Zhang H, Hager WW (2004) A nonmonotone line search technique and its application to unconstrained optimization. *SIAM J Optim* **14**, 1043–56.
9. Gu NZ, Mo JT (2008) Incorporating nonmonotone strategies into the trust region method for unconstrained optimization. *Appl Math Comput* **55**, 2158–72.
10. Mo JT, Liu CY, Yan SC (2007) A nonmonotone trust region method based on nonincreasing technique of weighted average of the successive function values. *J Comput Appl Math* **209**, 97–108.
11. More JJ, Garbow BS, Hillstrome KE (1981) Testing unconstrained optimization software. *ACM Trans Math Software* **7**, 17–41.
12. Touati-Ahmed D, Storey C (1990) Efficient hybrid conjugate gradient techniques. *J Optim Theor Appl* **64**, 379–97.