RAIRO Operations Research
RAIRO Oper. Res. **41** (2007) 105–121
DOI: 10.1051/ro:2007007

CONVERGENCE ANALYSIS OF ADAPTIVE TRUST REGION METHODS *

Zhen-Jun Shi[1], Xiang-Sun Zhang[2] and Jie Shen[3]

Abstract. In this paper, we propose a new class of adaptive trust region methods for unconstrained optimization problems and develop some convergence properties. In the new algorithms, we use the current iterative information to define a suitable initial trust region radius at each iteration. The initial trust region radius is more reasonable in the sense that the trust region model and the objective function are more consistent at the current iterate. The global convergence, super-linear and quadratic convergence rate are analyzed under some mild conditions. Numerical results show that some special adaptive trust region methods are available and efficient in practical computation.

Keywords. Adaptive trust region method, unconstrained optimization, global convergence, super-linear convergence.

Mathematics Subject Classification. 90C30, 49M37, 65K05.

1. Introduction

Trust region method is an important technique for solving optimization problems and has wide applications in many fields, such as science, engineering, economy and operations research, etc., due to its strong global convergence and robustness [2–4, 16, 21, 24, 29, 30].

Received March 3, 2005. Accepted October 16, 2006.
* *The work was supported in part by NSF DMI-0514900, USA.*

[1] College of Operations Research and Management, Qufu Normal University, Rizhao, Shandong 276826, P.R. China, and Department of Computer & Information Science, University of Michigan-Dearborn, Michigan MI48128, USA; zjshi@umd.umich.edu or zjshi@qrnu.edu.cn
[2] Institute of Applied Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, P.O. Box 2734, Beijing 100080, P.R. China; zxs@amt.ac.cn
[3] Department of Computer & Information Science, University of Michigan-Dearborn, Michigan MI48128, USA; shen@umich.edu

Consider an unconstrained optimization problem

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

where R^n is an n-dimensional Euclidean space and $f : R^n \to R^1$ is a twice continuously differentiable function.
Throughout this paper we denote

- x_k as the current iterate ($k = 0, 1, 2,...,$);
- $f_k = f(x_k)$, $g_k = g(x_k) = \nabla f(x_k)$, $H_k = H(x_k) = \nabla^2 f(x_k)$;
- $||\cdot||$ as the Euclidean norm;
- B_k as a symmetric matrix and an approximation to H_k in some sense at the point x_k;
- $\hat{B}_k = B_k + iI$ ($I \in R^{n\times n}$ denotes the unit matrix), i is the smallest nonnegative integer such that $\hat{B}_k$ becomes a positive definite matrix.

The existing methods for solving (1) can be divided into two classes: one is line search method and the other is trust region method. Line search method needs to carry out a line search procedure at each iteration and trust region method needs to solve a trust region subproblem at each step. As we have known, trust region methods are based on the following approach. At the iterate x_k (suppose that it is not a stationary point), a trial step is usually obtained by solving the following subproblem

$$\min_{d\in R^n} m_k(d) = g_k^T d + \frac{1}{2} d^T B_k d, \quad s.t. \quad ||d|| \le \Delta_k, \tag{2}$$

where Δ_k is a trust region radius. A merit function is normally used to test whether the trial step is accepted or the trust region radius needs to be adjusted. In comparison with quasi-Newton methods, trust region methods can converge to such a point that is not only a stationary point, but also satisfies second-order necessary conditions. Because of its strong convergence and robustness, trust region methods have been studied by many authors [1, 5, 9–12, 17, 19, 22, 25] and some convergence properties are given in the literature [6–8, 14, 15, 18, 20, 26].

It is well known that the trust region radius Δ_k in the above mentioned subproblem is independent of g_k and B_k. As a result, we do not know whether the quasi-Newton step, $-B_k^{-1} g_k$, is feasible at the k-th step, even when the test condition of the merit function is satisfied. This situation would decrease the efficiency of these methods. Furthermore, the choice of Δ_k also affects the efficiency of these methods.

Sartenaer [19] presented a strategy for determining automatically an initial trust region radius. The basic idea is to determine a maximal initial radius through many repeated trials in the direction, $-g_k$, in order to guarantee a sufficient agreement between the model and the objective function. Zhang *et al.* [27] presented another strategy of determining the trust region radius. Their basic idea originated

from the following subproblem in an artificial neural network research [25, 26],

$$\begin{aligned} \min \quad & m_k(d) = g_k^T d + \tfrac{1}{2} d^T B_k d \\ s.t. \quad & -\alpha(x_k) \le d_i \le \alpha(x_k),\ i = 1, 2, ..., n, \end{aligned}$$

where $\alpha_k = c^p \|g_k\|/m_k, 0 < c < 1, m_k = \min(\|B_k\|, 1)$, and p is a nonnegative integer. In their algorithm, instead of adjusting Δ_k, they adjust p at each iterate. Motivated by this technique, they solved the trust region subproblem (2) with

$$\Delta_k = c^p \|g_k\| \cdot \|\hat{B_k}^{-1}\| \tag{3}$$

and gave a global convergent adaptive trust region method [28], where $c \in (0, 1)$ and p is a nonnegative integer.

However, their method needs to estimate $\|B_k\|$ or $\|\hat{B}_k^{-1}\|$ at each iteration, which leads to some additional cost of computation. As a result, a simple adaptive trust region method was proposed [23], which used the following trust region radius

$$\Delta_k = c^p \|g_k\|^3 / g_k^T \hat{B}_k g_k, \tag{4}$$

where $c \in (0, 1)$, $\hat{B}_k$ is a positive definite matrix and p is a nonnegative integer.

In this paper, we present a new class of adaptive trust region methods for unconstrained optimization problems and develop several convergence properties. At each iteration, the new methods generate a suitable initial trust region radius automatically based on the current iterative information. The new trust region model is more consistent with the objective function at the current iterate. The global convergence and super-linear and quadratic convergence rate of these new methods are proved under some mild conditions. Numerical results show that some special adaptive trust region methods are available and efficient in practical computation.

The rest of this paper is organized as follows. In the next section, we introduce the new class of adaptive trust region methods and give some simple properties. In Sections 3 and 4, the global convergence and super-linear and quadratic convergence rate are investigated. Numerical results are given in Section 5. Conclusions and future research are summarized in Section 6.

2. New algorithms and some properties

The new trust region methods for (1) at the current iterate x_k need to solve the subproblem

$$\min_{d \in R^n} m_k(d) = g_k^T d + \frac{1}{2} d^T B_k d \quad s.t. \quad ||d|| \le \alpha_k, \tag{5}$$

where

$$\alpha_k = -c^p \frac{g_k^T q_k}{q_k^T \hat{B}_k q_k} \|q_k\|, \tag{6}$$

$(0 < c < 1)$, p is a nonnegative integer and q_k satisfies

$$-\frac{g_k^T q_k}{\|g_k\| \cdot \|q_k\|} \geq \tau \tag{7}$$

with $\tau \in (0, 1]$.

We can determine $\hat{B}_k$ by carrying out the procedure: if $q_k^T B_k q_k > 0$ then $\hat{B}_k = B_k$ else test $\hat{B}_k = B_k + iI$ for $i = 1, 2, 3, \cdots$ until $q_k^T \hat{B}_k q_k = q_k^T B_k q_k + i\|q_k\|^2 > 0$.

Define the following parameter

$$Pred_k = m_k(0) - m_k(d_k) = -m_k(d_k) = -\left(g_k^T d_k + \frac{1}{2} d_k^T B_k d_k\right),$$

$$Ared_k = f_k - f(x_k + d_k),$$
$$r_k = \frac{Ared_k}{Pred_k}.$$

We assume that

$(H1)$. The objective function $f(x)$ is twice continuously differentiable on R^n and the level set $L(x_0) = \{x \in R^n | f(x) \leq f(x_0)\}$ is bounded for a given x_0.

$(H2)$. $\{B_k\}$ is uniformly bounded, *i.e.* there exists an M such that $||B_k|| \leq M$ for all k.

Remark 2.1. Since $f(x)$ is a twice continuously differentiable function, $(H1)$ implies that $\{\nabla^2 f(x)\}$ is uniformly continuous and bounded on a bounded open convex set Ω that contains $L(x_0)$. Hence, there exists L such that $\|\nabla^2 f(x)\| \leq L$ and

$$\|g(x) - g(y)\| \leq L\|x - y\|, \ \forall x, y \in \Omega.$$

Remark 2.2. By the generating procedure of $\hat{B}_k$ and $(H2)$, we obtain that $\{\hat{B}_k\}$ is also uniformly bounded. In fact, assume that $\|B_k\| \leq M_0'$ for all k with M_0' being a positive constant, then $\|\hat{B}_k\| = \|B_k + iI\| \leq 2M_0'$ whenever $q_k^T \hat{B}_k q_k = q_k^T B_k q_k + i\|q_k\|^2 > 0$.

Algorithm(A).

Step 0. Set $0 < c < 1, \ \epsilon \geq 0, \ 0 < \eta < 1, x_0 \in R^n$ and $p := 0, k := 0$.

Step 1. If $||g_k|| \leq \epsilon$ then stop else solve (5) to obtain d_k. Set $\bar{x}_{k+1} = x_k + d_k$ and go to Step 2.

Step 2. If $r_k < \eta$ then set $p := p + 1$ and go to Step 1.

Step 3. Set $x_{k+1} = \bar{x}_{k+1}$, modify B_k as B_{k+1}, set $p := 0, \ k := k + 1$ and go to Step 1.

Lemma 2.3. *For $k \geq 1$,*

$$Pred_k \geq \frac{c^p \left(g_k^T q_k\right)^2}{2 q_k^T \hat{B}_k q_k}, \qquad \forall p = 0, 1, 2, \cdots.$$

Proof. Take $d = -\frac{c^p g_k^T q_k}{q_k^T \hat{B}_k q_k} q_k$, and since

$$\begin{aligned} ||d|| &= -\frac{c^p g_k^T q_k}{q_k^T \hat{B}_k q_k} ||q_k|| \\ &= \alpha_k, \end{aligned}$$

it follows that d is a feasible solution to (5). Noting that d_k is an optimal solution to (5), we have $m_k(d_k) \leq m_k(d)$, and thus,

$$\begin{aligned} Pred_k &\geq Pred \\ &= -(g_k^T d + \frac{1}{2} d^T B_k d) \\ &= -\left(-\frac{c^p (g_k^T q_k)^2}{q_k^T \hat{B}_k q_k} + \frac{c^{2p} (g_k^T q_k)^2 q_k^T B_k q_k}{2(q_k^T \hat{B}_k q_k)^2} \right) \\ &\geq -\left(-\frac{c^p (g_k^T q_k)^2}{q_k^T \hat{B}_k q_k} + \frac{c^p (g_k^T q_k)^2 q_k^T \hat{B}_k q_k}{2(q_k^T \hat{B}_k q_k)^2} \right) \\ &= \frac{c^p (g_k^T q_k)^2}{2 q_k^T \hat{B}_k q_k}. \end{aligned}$$

□

Lemma 2.4. *For $k \geq 1$, it holds that*

$$Pred_k \geq \frac{1}{2} \left(-\frac{g_k^T q_k}{\|q_k\|} \right) \min \left\{ \alpha_k, \ -\frac{g_k^T q_k}{q_k^T \hat{B}_k q_k} \|q_k\| \right\}.$$

Proof. In the case of $\alpha_k < -\frac{g_k^T q_k}{q_k^T \hat{B}_k q_k} \|q_k\|$, since $d = \frac{-c^p g_k^T q_k}{q_k^T \hat{B}_k q_k} q_k$ is a feasible solution to (5) and d_k is an optimal solution to (5), we have $m_k(d_k) \leq m_k(d)$, and thus,

$$\begin{aligned} Pred_k &\geq Pred \\ &= -(g_k^T d + \frac{1}{2} d^T B_k d) \\ &= -\left(-\frac{c^p (g_k^T q_k)^2}{g_k^T \hat{B}_k g_k} + \frac{c^{2p} (g_k^T q_k)^2 g_k^T B_k g_k}{2(g_k^T \hat{B}_k g_k)^2} \right) \\ &\geq -\left(-\frac{c^p (g_k^T q_k)^2}{g_k^T \hat{B}_k g_k} + \frac{c^p (g_k^T q_k)^2 g_k^T \hat{B}_k g_k}{2(g_k^T \hat{B}_k g_k)^2} \right) \\ &= \frac{c^p (g_k^T q_k)^2}{2 g_k^T \hat{B}_k g_k} \\ &= -\frac{1}{2} \cdot \frac{g_k^T q_k}{\|q_k\|} \alpha_k. \end{aligned}$$

In the case of $\alpha_k \geq -\frac{g_k^T q_k}{q_k^T \hat{B}_k q_k}\|q_k\|$, since $d = -\frac{g_k^T q_k}{q_k^T \hat{B}_k q_k} q_k$ is a feasible solution to (5) and d_k is an optimal solution to (5), we have $m_k(d_k) \leq m_k(d)$, and thus,

$$\begin{aligned} Pred_k &\geq Pred \\ &= -\left(g_k^T d + \frac{1}{2} d^T B_k d\right) \\ &= -\left(-\frac{(g_k^T q_k)^2}{q_k^T \hat{B}_k q_k} + \frac{(g_k^T q_k)^2 q_k^T B_k q_k}{2(q_k^T \hat{B}_k q_k)^2}\right) \\ &\geq -\left(-\frac{(g_k^T q_k)^2}{q_k^T \hat{B}_k q_k} + \frac{(g_k^T q_k)^2 q_k^T \hat{B}_k q_k}{2(q_k^T \hat{B}_k q_k)^2}\right) \\ &= -\frac{1}{2}\frac{g_k^T q_k}{\|q_k\|}\left(-\frac{g_k^T q_k}{q_k^T \hat{B}_k q_k}\|q_k\|\right). \end{aligned}$$

Therefore

$$Pred_k \geq -\frac{1}{2}\frac{g_k^T q_k}{\|q_k\|}\min\left\{\alpha_k,\ -\frac{g_k^T q_k}{q_k^T \hat{B}_k q_k}\|q_k\|\right\}. \qquad \square$$

Lemma 2.5. *If (H1) and (H2) hold, then Algorithm (A) is well-defined, i.e. Algorithm (A) does not circle at Step 1–Step 2 infinitely.*

Proof. Suppose that Algorithm (A) circles at Step 1–Step 2 infinitely and denotes $k(i)$ as the index of k. Then, there exist point sequences $\{x_{k(i)}\}$, $\{p^{k(i)}\}$, $\{d_{k(i)}\}$, $\{\alpha_{k(i)}\}$ and $\{r_{k(i)}\}$, where $x_{k(i)} = x_k, p^{k(i)} = i$, and $r_{k(i)} < \eta, \quad i = 0, 1, 2, \cdots$.

Lemma 2.3 shows that

$$Pred_{k(i)} \geq \frac{c^{p^{k(i)}}(g_k^T q_k)^2}{2q_k^T \hat{B}_k q_k} \quad \forall i = 0, 1, 2, \cdots. \tag{8}$$

By $(H1)$ and the definition of $Pred_k$ and $Ared_k$, we have

$$Ared_k - Pred_k = O(||d_k||^2). \tag{9}$$

By (8) and (9) we obtain

$$\begin{aligned} |r_{k(i)} - 1| = \frac{|Ared_{k(i)} - Pred_{k(i)}|}{|Pred_{k(i)}|} &\leq \frac{O(||d_{k(i)}||^2)}{\frac{c^{p^{k(i)}}(g_k^T q_k)^2}{2q_k^T \hat{B}_k q_k}} \\ &\leq \frac{O(||\alpha_{k(i)}||^2)}{\frac{c^{p^{k(i)}}(g_k^T q_k)^2}{2q_k^T \hat{B}_k q_k}} = \frac{O(||\alpha_{k(i)}||^2) c^{p^{k(i)}} 2\|q_k\|^2}{||\alpha_{k(i)}||^2 (q_k^T \hat{B}_k q_k)} \\ &\to 0 \quad (i \to +\infty), \end{aligned}$$

and thus, for sufficiently large i, we have $r_{k(i)} \geq \eta$. This contradicts $r_{k(i)} < \eta$ and the proof is completed. □

3. Global convergence

Theorem 3.1. *If $(H1)$ and $(H2)$ hold, $\epsilon = 0$ and $g_k^T q_k < 0$, then Algorithm (A) either stops at a stationary point of (1) in finite step or generates an infinite sequence $\{x_k\}$ such that*

$$\lim_{k\to\infty} -\left(\frac{g_k^T q_k}{\|q_k\|}\right) = 0. \tag{10}$$

Proof. Suppose that Algorithm (A) generates an infinite sequence $\{x_k\}$ and $\lim_{k\to\infty}(-\frac{g_k^T q_k}{\|q_k\|}) \neq 0$. Then there exist $\epsilon_0 > 0$ and an infinite subset $K \subseteq N = \{0, 1, 2, ...\}$ such that

$$-\frac{g_k^T q_k}{\|q_k\|} \geq \epsilon_0, \quad \forall k \in K. \tag{11}$$

$(H2)$ implies that there exists an $M_0 > 0$ such that

$$||\hat{B}_k|| \leq M_0, \forall k \in K. \tag{12}$$

By (11), (12), Lemma 2.3 and Algorithm (A), we have

$$\sum_{k\in K}[f_k - f_{k+1}] \geq \sum_{k\in K} \eta Pred_k \geq \sum_{k\in K} \eta \frac{c^{p^k}\epsilon_0^2}{2||\hat{B}_k||}, \tag{13}$$

where p^k is the largest p at the k-th iteration of Algorithm (A). Combining (12) and (13), we have

$$\sum_{k\in K}[f_k - f_{k+1}] \geq \sum_{k\in K} \frac{\eta\epsilon_0^2}{2M_0} c^{p^k}.$$

Since $\{f(x_k)\}$ decreases monotonically and has a bound from below, we obtain

$$\frac{\eta\epsilon_0^2}{2M_0} c^{p^k} \to 0, (k \in K, \quad k \to +\infty).$$

Therefore, $p^k \to +\infty (k \in K, \quad k \to +\infty)$ and we can assume that $p^k \geq 1, \quad \forall k \in K$. Algorithm (A) shows that the solution $\tilde{d_k}$ to the subproblem

$$\min_{d\in R^n} m_k(d) = g_k^T d + \frac{1}{2} d^T B_k d$$

$$s.t. \ ||d|| \leq -c^{p^k - 1} \frac{g_k^T q_k}{g_k^T \hat{B}_k g_k} ||q_k||, \quad (0 < c < 1),$$

cannot be accepted by the algorithm. That is to say, if $\tilde{x}_{k+1} = x_k + \tilde{d}_k$ then we have

$$\frac{f(x_k) - f(\tilde{x}_{k+1})}{-m_k(\tilde{d}_k)} < \eta. \tag{14}$$

On the other hand, by (12), Lemma 2.4 and Algorithm (A), we obtain

$$\begin{aligned}
\sum_{k\in K}[f_k - f_{k+1}] &\geq \sum_{k\in K} \eta Pred_k \\
&\geq -\sum_{k\in K} \eta \frac{1}{2}\frac{g_k^T q_k}{\|q_k\|} \min\left\{\alpha_k,\ -\frac{||q_k|| g_k^T q_k}{q_k^T \hat{B}_k q_k}\right\} \\
&\geq \sum_{k\in K} \eta \frac{\epsilon_0}{2} \min\left\{\alpha_k,\ \frac{||q_k||^2 \epsilon_0}{q_k^T \hat{B}_k q_k}\right\} \\
&\geq \sum_{k\in K} \frac{\eta\epsilon_0}{2} \min\left\{\alpha_k,\ \frac{\epsilon_0}{M_0}\right\}.
\end{aligned}$$

By $(H1)$, $(H2)$ and the monotonicity of $\{f(x_k)\}$, we have

$$\alpha_k \to 0 (k \in K, k \to +\infty). \tag{15}$$

Since

$$f(\tilde{x}_{k+1}) - f(x_k) - m_k(\tilde{d}_k) = O(||\tilde{d}_k||^2),$$

and by Lemma 2.4, (11), (12) and (15), we have

$$\begin{aligned}
|r_k - 1| &= |\frac{f(x_k) - f(\tilde{x}_{k+1})}{-m_k(\tilde{d}_k)} - 1| \\
&\leq \frac{O(||\tilde{d}_k||^2)}{\frac{\epsilon_0}{2}\min\left\{\alpha_k,\ \frac{\epsilon_0}{M_0}\right\}} \\
&\leq \frac{O(\alpha_k^2)}{\frac{1}{2}\epsilon_0\alpha_k} \\
&\to 0 (k \in K\ , k \to +\infty),
\end{aligned}$$

and thus, for sufficiently large k, we have

$$\frac{f(x_k) - f(\tilde{x}_{k+1})}{-m_k(\tilde{d}_k)} \geq \eta.$$

This is a contradiction to (14). Therefore, there is no such an infinite subset K and $\epsilon_0 > 0$ such that (11) holds. This shows that (10) holds. □

Corollary 3.2. *If the conditions in Theorem 3.1 hold and q_k satisfies (7), then*

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

Proof. Since q_k satisfies (7), by Theorem 3.1 we have

$$\|g_k\|\tau \leq \frac{-g_k^T q_k}{\|g_k\| \cdot \|q_k\|}\|g_k\| = -\frac{g_k^T q_k}{\|q_k\|} \to 0(k \to \infty).$$

This shows the conclusion. □

4. Convergence rate

Theorem 4.1. *Assume that* $(H1)$ *and* $(H2)$ *hold and Algorithm (A) generates an infinite sequence* $\{x_k\}$ *such that* $x_k \to x^*(k \to +\infty)$. $H(x)$ *is Lipschitz continuous on a neighborhood* $N(x^*, \epsilon)$ *of* x^* *and* $H(x^*)$ *and* B_k *are positive definite matrices such that*

$$\lim_{k\to\infty} \frac{||[B_k - H(x^*)]q_k||}{||q_k||} = 0, \tag{16}$$

where $q_k = -B_k^{-1}g_k$. *Then* $\{x_k\}$ *converges to* x^* *super-linearly.*

Proof. Because $\hat{B_k} = B_k$ for sufficiently large k, we assert that $\hat{d_k} = \frac{-g_k^T q_k}{q_k^T B_k q_k} q_k$ is an optimal solution to the subproblem

$$\min_{d\in R^n} m_k(d) = g_k^T d + \frac{1}{2} d^T B_k d$$

$$s.t. \quad ||d|| \leq -\frac{g_k^T q_k}{q_k^T \hat{B}_k q_k}||q_k||.$$

In the sequel, we will prove that

$$\frac{f(x_k) - f(x_k + \hat{d_k})}{-m_k(\hat{d_k})} > \eta,$$

for sufficiently large k. By (16) we have

$$g_k + H(x^*)\hat{d_k} = o(||\hat{d_k}||),$$

i.e.

$$\hat{d_k} = -H(x^*)^{-1}g_k + o(||\hat{d_k}||),$$

thus

$$||\hat{d_k}|| \leq ||H(x^*)^{-1}|| \cdot ||g_k|| + o(||\hat{d_k}||),$$

and consequently

$$\frac{||g_k||}{||\hat{d_k}||} \geq \frac{1}{||(H(x^*))^{-1}||} + \frac{o(||\hat{d_k}||)}{||\hat{d_k}||}.$$

By Corollary 3.2 we have $g_k \to 0$ $(k \to +\infty)$, and therefore $\hat{d}_k \to 0$ $(k \to +\infty)$. By Lemma 2.3 and noting that $-g_k^T q_k = q_k^T B_k q_k$ we have

$$-m_k(\hat{d}_k) \geq \frac{(g_k^T q_k)^2}{2q_k^T \hat{B}_k q_k} \geq \frac{q_k^T B_k q_k}{2}. \tag{17}$$

By (16) and noting that $q_k = -B_k^{-1} g_k = \hat{d}_k$ we have

$$f(x_k + \hat{d}_k) - f(x_k) - m_k(\hat{d}_k) = o(||\hat{d}_k||^2). \tag{18}$$

By (17), (18) and (12), we have

$$\begin{aligned}
\left| \frac{f(x_k) - f(x_k + \hat{d}_k)}{-m_k(\hat{d}_k)} - 1 \right| &\leq \frac{o(||\hat{d}_k||^2)}{\frac{q_k^T B_k q_k}{2M_0}} \\
&\leq \frac{2o(\|\hat{d}_k\|^2)}{M_0 \|q_k\|^2} \\
&\leq \frac{2o(\|\hat{d}_k\|^2)}{M_0 \|\hat{d}_k\|^2} \\
&\to 0 \quad (k \to +\infty),
\end{aligned}$$

which implies that

$$\frac{f(x_k) - f(x_k + \hat{d}_k)}{-m_k(\hat{d}_k)} > \eta,$$

for sufficiently large k. Therefore $x_{k+1} = x_k + \hat{d}_k$ for sufficiently large k. This shows that Algorithm (A) reduces to a quasi-Newton method for sufficiently large k. We can complete the rest proof by citing [2, 5, 15] or other related literature. □

Theorem 4.2. *Assume that* $(H1)$ *and* $(H2)$ *hold and Algorithm (A) generates an infinite sequence* $\{x_k\}$ *such that* $x_k \to x^*$ $(k \to +\infty)$. $H(x)$ *is Lipschitz continuous on a neighborhood* $N(x^*, \epsilon)$ *of* x^* *and* $H(x^*)$ *and* B_k *are positive definite matrices with* $B_k = H(x_k)$ *and* $q_k = -B_k^{-1} g_k$. *Then* $\{x_k\}$ *converges to* x^* *quadratically.*

Proof. Since $B_k = H(x_k)$ implies that all conditions of Theorem 4.1 hold, we have $\hat{d}_k = q_k \to 0$ $(k \to \infty)$. Therefore, there exists a k' such that

$$x_k + tq_k \in N(x^*, \epsilon),\ k \geq k',\ t \in [0, 1]. \tag{19}$$

Because $H(x)$ is Lipschitz continuous on the neighborhood $N(x^*, \epsilon)$ of x^*, there must exist an $L(\epsilon) > 0$ such that

$$\|H(x) - H(y)\| \leq L(\epsilon)\|x - y\|,\ \forall x, y \in N(x^*, \epsilon). \tag{20}$$

In this case, Algorithm (A) reduces to Newton method for sufficiently large k. The rest proof can also be seen from [2, 5, 15]. □

Theorem 4.3. *Assume that* $(H1)$ *and* $(H2)$ *hold and* $B_k = H_k$, $\alpha_k = -c^p \frac{\|q_k\| g_k^T q_k}{q_k^T B_k q_k}$, *where* $0 < c < 1$ *and* p *is a nonnegative integer. If* $\{x_k\}$ *converges to* x^*, *then* $H(x^*)$ *is a semi-positive definite matrix, i.e.* x^* *satisfies the second order necessary condition.*

Proof. Denote λ_k^1 and λ^* as the smallest eigenvalue of H_k and $H(x^*)$ respectively. Let z_k be a normal eigenvector ($\|z_k\| = 1$) of H_k corresponding to the eigenvalue λ_k^1 and $z_k^T g_k \leq 0$, and then $H_k z_k = \lambda_k^1 z_k$. Suppose that $H(x^*)$ is not a positive semi-definite matrix, then $\lambda^* < 0$ and thus $\lambda_k^1 < 0$ for sufficiently large k.

Because $\|\alpha_k z_k\| = \alpha_k$, it follows that $\alpha_k z_k$ is a feasible solution to (5). Therefore,

$$Pred_k \geq -\left(\alpha_k g_k^T z_k + \frac{1}{2}\alpha_k^2 z_k^T B_k z_k\right) \geq -\frac{1}{2}\alpha_k^2 z_k^T B_k z_k$$

$$= -\frac{1}{2}\alpha_k^2 z_k^T H_k z_k = -\frac{1}{2}\alpha_k^2 \lambda_k^1 ||z_k||^2 = -\frac{1}{2}\alpha_k^2 \lambda_k^1. \tag{21}$$

By (21) and Algorithm (A) we have

$$Ared_k \geq \eta Pred_k \geq -\frac{1}{2}\eta \alpha_k^2 \lambda_k^1.$$

Since $\{f(x_k)\}$ is a monotone decreasing sequence and has a bound from below, we have

$$Ared_k = f_{k+1} - f_k \to 0 \ (k \to +\infty),$$

and thus $\alpha_k^2 \lambda_k^1 \to 0$ $(k \to +\infty)$. Noting that $\lambda_k^1 \to \lambda^*$ $(k \to \infty)$, we have

$$\lim_{k\to\infty} \alpha_k = 0.$$

From the definition of Algorithm (A), we can observe that the solution $\tilde{d_k}$ to

$$\min_{d\in R^n} m_k(d) = g_k^T d + \frac{1}{2} d^T B_k d$$

$$s.t. \quad ||d|| \leq \hat{\alpha}_k$$

with $\hat{\alpha}_k = -c^{q_k - 1} \frac{||q_k|| g_k^T q_k}{q_k^T B_k q_k}$ $(0 < c < 1)$, cannot be accepted by the algorithm, *i.e.* if $\tilde{x}_{k+1} = x_k + \tilde{d_k}$, then

$$\frac{f(x_k) - f(\tilde{x}_{k+1})}{-m_k(\tilde{d_k})} < \eta. \tag{22}$$

On the other hand, noting that $B_k = H_k$, by Taylor expansion, we have

$$|f(x_k) - f(\tilde{x}_{k+1}) + m_k(\tilde{d_k})| = o(||\tilde{d_k}||^2). \tag{23}$$

By (21) and (23) we obtain

$$\begin{aligned}\left|\frac{f(x_k)-f(\tilde{x}_{k+1})}{-m_k(\tilde{d}_k)}-1\right| &\leq \frac{o(||\tilde{d}_k||^2)}{-\frac{1}{2}\hat{\alpha}_k^2\lambda_k^1}\\ &\leq \frac{o(\hat{\alpha}_k^2)}{-\frac{1}{2}\hat{\alpha}_k^2\lambda_k^1}\\ &\to 0(k\to\infty).\end{aligned}$$

Since $\lambda_k^1 \to \lambda^* < 0$ $(k \to +\infty)$, we get

$$\left|\frac{f(x_k)-f(\tilde{x}_{k+1})}{-m_k(\tilde{d}_k)}-1\right| \to 0(k\to+\infty),$$

and thus

$$\frac{f(x_k)-f(\tilde{x}_{k+1})}{-m_k(\tilde{d}_k)} \geq \eta$$

for sufficiently large k, which contradicts (22). This shows that $H(x^*)$ is a positive semi-definite matrix. □

5. Numerical results

In adaptive trust region methods, q_k satisfying (7) has a wide scope. Of course, $q_k = -g_k$ is a natural choice, which leads to a recently proposed adaptive trust region method [23]. If we take $q_k = -\hat{B}_k^{-1}g_k$, then we can obtain a new adaptive trust region method. In this case, the trust region radius is $\alpha_k = c^p\|\hat{B}_k^{-1}g_k\|$ at the k-th step. The adaptive trust region methods with $q_k = -g_k$ and $q_k = -\hat{B}_k^{-1}g_k$ are denoted by TRS and TRN respectively. The original trust region method with $\overline{\Delta} = 100$, $\Delta_0 = 50$ and $\eta = 0.01$ is denoted by TRO [15] (p. 68), which is described as follows.

Algorithm 5.1 (trust region)

Given $\overline{\Delta} > 0$, $\Delta_0 \in (0, \overline{\Delta})$, and $\eta \in [0, \frac{1}{4})$;
For $k = 0, 1, 2, ...$
Obtain d_k by (approximately) solving (2);
Evaluate r_k;
if $r_k < \frac{1}{4}$
$\Delta_{k+1} = \frac{1}{4}\|d_k\|$
else
if $r_k > \frac{3}{4}$ and $\|d_k\| = \Delta_k$
$\Delta_{k+1} = \min(2\Delta_k, \overline{\Delta})$
else
$\Delta_{k+1} = \Delta_k$;

TABLE 1. Iterations, function and gradient evaluations and CPU time.

P	n	TRS	TRN	TRI	TRZ	TRO
1	2	26/88/26	18/36/18	28/62/28	26/59/29	36/56/56
2	2	23/45/23	18/34/18	27/58/27	32/32/32	42/57/52
3	2	15/46/15	12/23/12	16/52/16	22/33/22	20/78/60
4	2	15/55/16	12/18/12	17/68/17	34/64/34	28/75/68
5	2	26/48/26	21/26/21	26/68/26	37/87/37	32/78/72
6	2	18/49/18	15/15/15	23/63/23	35/38/35	28/85/78
7	3	43/46/43	36/36/36	52/65/52	61/72/61	48/67/67
8	3	48/62/48	35/35/35	53/58/53	64/68/64	83/75/72
9	3	46/56/46	37/37/37	63/68/63	66/74/66	85/85/81
10	3	28/34/28	23/23/23	25/50/25	56/61/56	65/72/64
11	3	46/62/46	38/38/38	53/57/53	59/78/59	72/83/78
12	3	46/68/46	37/37/37	65/75/65	69/72/69	83/88/83
13	4	56/58/56	45/45/45	68/74/68	81/89/81	91/95/91
14	4	38/45/38	29/29/29	43/48/43	68/78/68	86/92/86
15	4	76/76/76	59/59/59	85/88/85	97/99/97	94/98/94
16	4	29/36/29	18/18/18	43/63/43	78/84/78	93/98/93
17	5	93/98/93	79/79/79	98/98/98	85/96/85	92/123/93
18	6	38/46/38	26/26/26	38/64/38	54/59/54	63/82/78
CPU	–	118 s	59 s	155 s	124 s	195 s

if $r_k > \eta$

$x_{k+1} = x_k + d_k$

else

$x_{k+1} = x_k$;

end(for).

We chose the parameters $c = 0.75,\ \eta = 0.01$, $\epsilon = 10^{-8}$ and $\{B_k\}$ was modified by BFGS formula.

If we set $B_k \equiv I$ and $q_k = -g_k$ we can also obtain a new adaptive trust region method denoted by TRI. Zhang's adaptive trust region method [27] is denoted by TRZ. We chose 18 test problems and their initial points from the literature [13]. For example, P5 means the No. 5 problem and so on. The stop criteria is

$$\|g_k\| \leq \epsilon = 10^{-8}.$$

We used Visual C++ Language to design the program in a portable computer with Pentium IV/ 1.2MHz CUP. Numerical results are reported in Tables 1–4.

TABLE 2. Total CPU time($\Delta_0 = 0.8$ and $\eta = 0$).

TRO	TR0(1)	TRO(5)	TRO(10)	TRO(20)	TRO(30)
CPU	162 s	154 s	178 s	169 s	182 s

TABLE 3. Total CPU time($\Delta_0 = 10$ and $\eta = 0.15$).

TRO	TR0(50)	TRO(100)	TRO(150)	TRO(200)	TRO(300)
CPU	198 s	204 s	212 s	185 s	195 s

TABLE 4. Total CPU time($\Delta_0 = 0.2$ and $\eta = 0.01$).

TRO	TR0(0.3)	TRO(0.5)	TRO(0.8)	TRO(0.85)	TRO(0.9)
CPU	161 s	158 s	168 s	170 s	163 s

In Table 1, "P" denotes the test problem and n denotes the dimension of problems. Each group of three numbers means the iteration number, function evaluations and gradient evaluations in sequence. "T" refers to the total CPU time for solving all the 18 test problems. As we can see, it is difficult to choose an adequate upper bound $\overline{\Delta}$ in the original trust region method. If $\overline{\Delta}$ is too large then the number of solving subproblems will be increased. If $\overline{\Delta}$ is too small then the efficiency of algorithm will be reduced. Thereby, we should choose an adequate initial trust region radius at each iteration. Such problem does not exist in adaptive trust region methods because the initial trust region radius in adaptive trust region method can be adjusted automatically according to the information of iterates.

However, TRO has an advantage that allows $\eta = 0$. In adaptive trust region method, $\eta \in (0, 1)$. If $\eta = 0$, we don't know whether the adaptive trust region method can converge. If we take $\overline{\Delta} = 1, 5, 10, 20, 30$, $\Delta_0 = 0.8$ and $\eta = 0$ in TRO, we have the results in Table 2. TRO(1), TRO(5), TRO(10), TRO(20), TRO(30) denotes the corresponding TRO with $\overline{\Delta} = 1, 5, 10, 20, 30$, respectively. In Table 2, we only list the total CPU time for solving all the 18 problems.

Similarly, we use TRO(m) to denote TRO with $\overline{\Delta} = m$, and the related numerical results are listed in Tables 3 and 4.

Tables 2–4 show that TRO is more efficient when we choose $\overline{\Delta} \in (0.5, 1.5)$ and $\eta \in [0, 0.15]$. This is only a guess. In summary, $\overline{\Delta}$ in TRO is difficult to determine in practical computation. Adaptive trust region method can overcome this difficulty.

It is shown from Table 1 that TRN seems the best adaptive trust region method because it uses the least total CPU time for solving all the 18 test problems. TRS is the second adaptive trust region method that has good numerical performance. The preliminary numerical results show that the adaptive trust region method is a promising method for optimization problems.

TABLE 5. Iterations, function and gradient evaluations and CPU time.

P	n	TRS	TRN	TRI	TRZ	TRO
19	11	32/74/32	26/48/36	31/79/45	35/83/35	58/184/64
20	31	86/132/79	42/97/63	65/99/64	63/84/63	56/179/56
21	20	35/69/35	42/64/42	75/147/75	46/53/46	78/126/78
22	30	54/137/85	38/94/45	69/72/69	53/84/68	59/196/83
23	20	53/64/56	35/47/46	67/84/67	62/68/62	74/198/78
24	20	47/52/68	45/58/45	57/83/68	45/83/63	65/174/79
25	40	64/93/89	62/78/62	78/124/98	78/156/79	63/187/93
CPU	–	104 s	79 s	126 s	158 s	174 s

As we can see, the first 18 problems are all small problems. Thus, seven large problems, Problems 19–25, from [13] were used to test the adaptive trust region methods. We take the parameters $\overline{\Delta} = 1$, $\eta = 0.15$, $c = 0.75$, $\epsilon = 10^{-8}$. Numerical results are listed in Table 5, indicating that TRN is the best adaptive trust region method. We guess that TRN essentially reduces to quasi-Newton method in many situations. Moreover, memory use and matrix computation may play a role in performance comparison beyond the iterative number, function and gradient evaluations. Therefore, CPU time seems a reasonable metric for comparing the numerical performance of algorithms.

6. Conclusions and future research

In this paper, we presented a new class of adaptive trust region methods for unconstrained optimization problems and investigated their global convergence. In these new methods, the trust region radius can be adjusted automatically according to the current iterative information by a simple computational formula. Different choices of q_k determine different adaptive trust region methods. If we take $q_k = -g_k$ then the new method will reduce to the recently proposed method [23]. If we take $q_k = -B_k^{-1} g_k$ with B_k^{-1} being available, then we can obtain some interesting new methods that have convergence properties of quasi-Newton method. We can choose other q_k to obtain many different adaptive trust region methods. Numerical results showed that some adaptive trust region methods were available and efficient in practical computation.

For the future research, we should choose $\hat{B}_k$ and q_k by using other approaches and conduct some numerical experiments when q_k is taken as another special direction that satisfies (7).

Acknowledgements. The authors would like to thank two anonymous referees and the editor for their careful reading and valuable comments and suggestions that greatly improved the paper. The first author is very grateful to Professor Y. Yuan for his meaningful guidance when he was as a postdoctoral fellow in Chinese Academy of Sciences.

References

[1] R.H. Byrd, J. Nocedal and Y.X. Yuan, Global convergence of a class of quasi-Newton methods on convex problems. *SIAM J. Numer. Anal.* **24** (1987) 1171–1190.
[2] A.R. Conn, N.I.M. Gould and P.L. Toint, *Trust-Region Methods,* Society for Industrial and Applied Mathematics (SIAM), Philadelphia PA (2000).
[3] G. Corradi, A trust region algorithm for unconstrained optimization. *Int. J. Comput. Math.* **65** (1997) 109–119.
[4] Y.H. Dai, D.C. Xu, A new family of trust region algorithms for unconstrained optimization. *J. Comput. Math.* **21** (2003) 221–228.
[5] R. Fletcher, *Practical Method of Optimization, Unconstrained Optimization*, Vol. 1, John Wiley, New York (1980).
[6] J.Y. Fan, W.B. Ai and Q.Y. Zhang, A line search and trust region a lgorithm with trust region radius converging to zero. *J. Comput. Math.* **22** (2004) 865–872.
[7] J.H. Fu and W.Y. Sun, Nonmonotone adaptive trust-region method for unconstrained optimization problems. *Appl. Math. Comput.* **163** (2005) 489–504.
[8] E.M. Gertz, A quasi-Newton trust-region method. *Math. Program. Ser. A* **100** (2004) 447–470.
[9] N.I.M. Gould, D. Orban, A. Sartenaer and P.L. Toint, Sensitivity of trust-region algorithms to their parameters. *4OR* **3** (2005) 227–241.
[10] N.I.M. Gould, D. Orban and P.L. Toint, Numerical methods for large-scale nonlinear optimization. *Acta Numer.* **14** (2005) 299–361.
[11] L. Hei, A self-adaptive trust region algorithm. *J. Comput. Math.* **21** (2003) 229–236.
[12] J.J. Moré, Recent developments in algorithms and software for trust region methods, in *Mathematical Programming: The State of the Art*, edited by A. Bachem, M. Grotchel and B. Korte, Springer- Verlag, Berlin (1983) 258–287.
[13] J.J. Moré, B.S. Grabow and K.E. Hillstrom, Testing Unconstrained Optimization software. *ACM Trans. Math. Software* **7** (1981) 17–41.
[14] D. Mauricio and N. Maculan, A trust region method for zero-one nonlinear programming. *RAIRO Rech. Opér.* **31** (1997) 331–341.
[15] J. Nocedal and S.J. Wright, *Numerical Optimization.* Springer-Verlag, New York (1999).
[16] Q. Ni, A globally convergent method of moving asymptotes with trust region technique. *Optim. Methods Softw.* **18** (2003) 283–297.
[17] M.J.D. Powell, Convergence properties of a class minimization algorithms, in *Nonlinear Programming 2*, edited by O.L. Mangasarian, R.R. Meyer and S.M. Robinson, Academic Press, New York (1975) 1–27.
[18] M.J.D. Powell, On the global convergence of trust region algorithms for unconstrained optimization. *Math. Prog.* **29** (1984) 297–303.
[19] A. Sartenaer, Automatic determination of an initial trust region in nonlinear programming. *SIAM J. Sci. Comput.* **18** (1997) 1788–1803.
[20] G.A. Shultz, R.B. Schnabel and R.H. Byrd, A family of trust-region-based algorithms for unconstrained minimization with strong global convergence. *SIAM J. Numer. Anal.* **22** (1985) 47–67.
[21] W.Y. Sun, Nonmonotone trust region method for solving optimization problems. *Appl. Math. Comput.* **156** (2004) 159–174.
[22] J. Sun, On piecewise quadratic Newton and trust region problems. *Math. Programming, Ser. B* **76** (1997) 451–467.

[23] Z.J. Shi and H.Q. Wang, *A new self-adaptive trust region method for unconstrained optimization*. Technical Report, College of Operations Research and Management, Qufu Normal University (2004).
[24] J.Z. Zhang and C.X. Xu, Trust region dogleg path algorithms for unconstrained minimization, Management science in China (Kowloon, 1996). *Ann. Oper. Res.* **87** (1999) 407–418.
[25] X.S. Zhang, Trust region method in neural network. *Acta Math. Appl. Sinica* **12** (1996) 1–10.
[26] X.S. Zhang, Trust region method in neural network. *Acta Math. Appl. Sinica* (English Ser.) **13** (1997) 342–352.
[27] X.S. Zhang, J.L. Zhang and L.Z. Liao, An adaptive trust region method and its convergence. *Science in China* **45** (2002) 620-631.
[28] X.S. Zhang, Z.W. Chen and J.L. Zhang, A self-adaptive trust region method for unconstrained optimization. *OR Transactions* **5** (2001) 53–62.
[29] Y.X. Yuan, A review of trust region algorithms for optimization, in *ICIAM 99 (Edinburgh)*, Oxford Univ. Press, Oxford (2000) 271–282.
[30] Y.X. Yuan, Trust region algorithms for nonlinear equations. *Information* **1** (1998) 7–20.