A Nonmonotone Adaptive Trust Region Method and Its Convergence

JU-LIANG ZHANG*
School of Economics and Management
Tsinghua University
Beijing, 100084, P.R. China

XIANG-SUN ZHANG
Institute of Applied Mathematics
Academy of Mathematics and System Sciences
Chinese Academy of Sciences, P.O. Box 2734
Beijing, 100080, P.R. China

(Received May 2001; revised and accepted August 2002)

Abstract—in this paper, we combine the new trust region subproblem proposed in [1] with the nonmonotone technique to propose a new algorithm for unconstrained optimization—the nonmonotone adaptive trust region method. The local and global convergence properties of the nonmonotone adaptive trust region method are proved. Its efficiency is tested by numerical results. © 2003 Elsevier Science Ltd. All rights reserved.

Keywords—trust region method, Unconstrained optimization, Global convergence, Superlinear convergence, Nonmonotone linesearch technique.

1. INTRODUCTION

Consider the following unconstrained nonlinear programming problem:

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

where $f(x)$ is a twice continuously differentiable function. Throughout the paper, we use the following notations.

- $\| \cdot \|$ is the Euclidean norm.
- $g(x) \in R^n$ is the gradient of $f$ evaluated at $x$.
- $H(x) \in R^{n \times n}$ is the Hessian of $f$ evaluated at $x$.

This work is supported in part by the National Natural Science Foundation of China (Grant No. 39830070, 10171055).

The authors would like to thank Professor Y. X. Yuan for providing the source programs for reference [3]. The authors would like to thank two anonymous referees and the Editor-in-Chief for their valuable suggestions and comments that helped to improve the presentation of this paper.

*Author to whom all correspondence should be addressed.
\( \{x_k\} \) is a sequence of points generated by an algorithm, and \( f_k = f(x_k) \), \( g_k = g(x_k) \), and \( H_k = H(x_k) \).

- \( B_k \) is a symmetric matrix which is either \( H_k \) or an approximation of \( H_k \).
- \( \hat{B}_k \) is a safely positive definite matrix based on Schnabel and Eskow [2] modified Cholesky factorization, \( \hat{B}_k = B_k + E_k \), where \( E_k = 0 \) if \( B_k \) is safely positive definite, and \( E_k \) is a diagonal matrix chosen to make \( \hat{B}_k \) positive definite otherwise.

Trust region methods are based on the following idea. At each iterate point \( x_k \) (nonstationary point), a trial step is usually executed by solving the following subproblem:

\[
\min_{d \in \mathbb{R}^n} \quad g_k^T d + \frac{1}{2} d^T B_k d, \\
\text{s.t.} \quad \|d\| \leq \Delta_k,
\]

where \( \Delta_k \) is the trust radius. A merit function is normally used to test whether the trial step is accepted or the trust radius needs to be adjusted. Comparing with quasi-Newton methods, trust region methods converge to a point which not only is 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 [4-10].

It is well known that the trust radius \( \Delta_k \) is independent of \( g_k \) and \( B_k \). So, at each iterate point \( x_k \) which is far from the optimum, we do not know whether the quasi-Newton step \(-B_k^{-1}g_k\) is feasible; even the test condition of the merit function is satisfied. This situation would decrease the efficiency of these methods. Furthermore, the choice of \( \Delta_0 \) also affects the efficiency of these methods, but there does not exist any general rule on choosing \( \Delta_0 \).

In [1], we proposed a new trust region subproblem as follows:

\[
\min_{d \in \mathbb{R}^n} \quad g_k^T d + \frac{1}{2} d^T B_k d, \\
\text{s.t.} \quad \|d\| \leq \alpha_k,
\]

where \( \alpha_k = c\|g_k\| \tilde{M}_k \), \( 0 < c < 1 \), \( \tilde{M}_k = \|\hat{B}_k^{-1}\| \), and \( p \) is a nonnegative integer. Therefore, instead of adjusting \( \Delta_k \), we adjust \( p \). With this new subproblem, we construct an adaptive trust region method. Numerical experiments show that the algorithm is very efficient.

It is known that the objective function sequences generated by these algorithms are monotonically decreasing; i.e., \( f(x_k) \geq f(x_{k+1}) \), \( k = 0, 1, \ldots \).

In [11], Grippo et al. proposed a nonmonotone line search technique for Newton’s method. Since then, the nonmonotone technique has been studied by many authors. Grippo et al. [12] proposed a nonmonotone truncated Newton method with nonmonotone line search and many authors generalized the nonmonotone technique to trust region methods and proposed nonmonotone trust region methods [13-17]. Theoretic analysis and numerical results show that the algorithms with nonmonotone properties are more efficient than the algorithms with monotone properties.

In this paper, we combine subproblem (3) with nonmonotone technique to propose a nonmonotone adaptive trust region method and study its convergence properties. The efficiency of the method is tested by the numerical results in Section 4.

The rest of the paper is organized as follows. In Section 2, we present the nonmonotone adaptive trust region method model. In Section 3, the global and local convergence properties are studied. Numerical results in Section 4 indicate that the algorithm is very efficient. Finally, some concluding remarks are addressed in Section 5.

### 2. ALGORITHM MODEL

In this section, we give a nonmonotone adaptive algorithm model. First, some definitions are given. At point \( x_k \), we define predict reduction as

\[
\text{pred}_k = -g_k^T d_k + \frac{1}{2} d_k^T B_k d_k,
\]
where \( d_k \) is the solution of (3). Let

\[
\ell(k) = \max_{0 \leq j \leq n(k)} \{ f_{k-j}, \quad k = 0, 1, 2, \ldots, \tag{4}
\]

where \( n(k) = \min\{ N, k \} \), \( N \geq 0 \) is an integer constant. Now we give our algorithm model as follows.

**ALGORITHM MODEL.**

**Step 0.** Choose \( 0 < c < 1, \ \epsilon > 0, \ 1 > \eta > 0, \ x_0 \in \mathbb{R}^n, \) a symmetric matrix \( B_0 \in \mathbb{R}^{n \times n}, \) an integer \( N \geq 0. \) Let \( p = 0, \ k = 0. \)

**Step 1.** Compute \( f_k, g_k. \) If \( \|g_k\| < \epsilon, \) stop.

**Step 2.** Compute \( d_k \) by solving (3) and calculate \( n(k), \ell(k), \) pred\(_k\), and \( r_k \), where \( r_k \) is defined as follows:

\[
r_k = \frac{\ell(k) - f(x_k + d)}{\text{pred}_k}.
\]

If \( r_k < \eta, \) then \( p := p + 1, \) go to Step 2. Otherwise, go to Step 3.

**Step 3.** \( x_{k+1} = x_k + d_k, \) generate \( B_{k+1}, \) set \( p = 0, \ k := k + 1, \) go to Step 1.

**REMerk.**

(i) \( B_k \) can be obtained by quasi-Newton iterate formula.

(ii) If \( N = 0, \) this algorithm reduces to the adaptive trust region algorithm in [1].

(iii) In this algorithm, the procedure of “Step 2—Step 2—Step 2” is named as inner cycle.

3. ANALYSIS OF CONVERGENCE

In this section, we discuss the convergence properties of the algorithm. Before we address some theoretical issues, we would like to make the following assumptions.

**Assumption 3.1.**

(i) The level set \( L(x_0) = \{ x \mid f(x) \leq f(x_0) \} \) is bounded for any given \( x_0 \in \mathbb{R}^n \) and \( f(x) \) is continuously differentiable in \( L(x_0) \) for any given \( x_0 \in \mathbb{R}^n. \)

(ii) Matrices \( \{B_k\} \) are uniformly bounded.

Assumption 3.1(ii) implies that there exists an \( M > 0 \) such that

\[
\|B_k\| \leq M, \quad \text{for all } k. \tag{5}
\]

If \( B_k \) is invertible, from \( \|B^{-1}\|\|B\| \geq 1, \) there exists a positive number \( \bar{M} > 0 \) such that

\[
\|B_k^{-1}\| \geq \bar{M}, \quad \text{for all } k \text{ such that } B_k \text{ is invertible.} \tag{6}
\]

**Lemma 3.1.** (See [1].) Suppose that Assumption 3.1(ii) holds. Then

\[
\text{pred}_k \geq \frac{c^p}{2M_k} \|g_k\|^2, \quad \text{for all } p = 0, 1, 2, \ldots,
\]

where \( M_k = \|\hat{B}_k\|. \)

It is similar to Theorem 4 in [6], and we can prove the following theorem.

**Lemma 3.2.** \( \text{pred}_k \geq (1/2)\|g_k\| \min\{\alpha_k, \|g_k\|/\|B_k\|\}. \)

The following lemma guarantees that the nonmonotone adaptive trust region algorithm does not cycle infinitely in the inner cycle.
LEMMA 3.3. Suppose that Assumption 3.1 holds. Then, the nonmonotone adaptive trust region algorithm is well defined; i.e., the algorithm does not cycle in the inner cycle infinitely.

PROOF. First, we prove that when $p$ is sufficiently large, it holds that
\[
\frac{f(x_k) - f(x_k + d_k)}{\text{pred}_k} \geq \eta. \tag{7}
\]

Let $d_k'$ be the solution of (3) corresponding to $p = i$ at $x_k$ and $\text{pred}_k(i)$ be the predict reduction corresponding to $p = i$ at $x_k$. It follows from Lemma 3.1 that
\[
\text{pred}_k(i) \geq \frac{c^i}{2M_k} \|g_k\|^2.
\]

So,
\[
\left| \frac{f(x_k) - f(x_k + d_k)}{\text{pred}_k} - 1 \right| \leq \frac{O\left(\|d_k\|^2\right)}{(c^i/2M_k) \|g_k\|^2} \leq \frac{O\left(\|g_k\|^2\right)}{(c^i/2M_k) \|g_k\|^2} \to 0, \quad \text{as } i \to \infty,
\]

which implies that (7) holds for $p$ sufficiently large.

The definition of the algorithm implies that
\[
r_k = \frac{f_{l(k)} - f(x_k + d_k)}{\text{pred}_k} \geq \frac{f(x_k) - f(x_k + d_k)}{\text{pred}_k}.
\]

Therefore, when $p$ is sufficiently large, $r_k \geq \eta$. This implies that the algorithm does not cycle in the inner cycle infinitely.

LEMMA 3.4. Suppose that Assumption 3.1 holds and $\{x_k\}$ is generated by the algorithm. Then $\{x_k\} \subseteq L(x_0)$.

PROOF. The result evidently holds for $k = 0$. Assume that $x_k \in L(x_0)$, for $k \geq 0$. By the definition of the algorithm, we get
\[
r_{l(k)} > \eta > 0. \tag{8}
\]

Then
\[
f_{l(k)} \geq f_{k+1} + \eta \text{pred}_k \geq f_{k+1}. \tag{9}
\]

Since $l(k) \leq k$, $f_{l(k)} \leq f_0$, then it follows from (9) that
\[
f_{k+1} \leq f_0,
\]

i.e.,
\[
x_{k+1} \in L(x_0),
\]

which completes the proof.

LEMMA 3.5. Suppose that Assumption 3.1 holds. Then $\{f_{l(k)}\}$ is not increasing monotonically and is convergent.

PROOF. From the definition of the algorithm, we have that
\[
f_{l(k)} \geq f_{k+1}, \quad \forall k. \tag{10}
\]
Now we proceed the proof in the following two cases.

(i) \(k \geq N\). In this case, from the definition of \(f_{i(k)}\) and (10), it holds that

\[
f_{i(k+1)} = \max_{0 \leq j \leq n(k+1)} \{f_{k+1-j}\}
= \max \left\{ \max_{0 \leq j \leq n(k)-1} \{f_{k-j}, f_{k+1}\} \right\}
\leq f_{i(k)}.
\]

(ii) \(k < N\). In this case, by induction, we can prove that

\[
f_{i(k)} = f_0.
\]

So the sequence \(\{f_{i(k)}\}\) is not increasing monotonically. From Assumption 3.1(i) and Lemma 3.4, we know that \(\{f_k\}\) is bounded. Hence, \(\{f_{i(k)}\}\) is convergent.

**Theorem 3.1.** Suppose that Assumption 3.1 holds. If \(\epsilon = 0\), then the algorithm either stops finitely or generates an infinite sequence \(\{x_k\}\) such that

\[
\lim \inf_{k \to \infty} \|g_k\| = 0.
\]

**Proof. Contradiction.** Assume that the theorem is not true. Then there exists a constant \(\epsilon_0 > 0\) such that

\[
\|g_k\| \geq \epsilon_0, \quad \forall k.
\]

Assumption 3.1(ii) and the definition of \(B_k\) imply that there exists \(\tilde{M} > 0\) such that

\[
\|\tilde{B}_k^{-1}\| \geq \tilde{M}.
\]

Therefore, by Assumption 3.1(i), Lemma 3.1, and (12), there exists a constant \(a > 0\) such that

\[
pred_k \geq ac^{p_k},
\]

where \(p_k\) is the value of \(p\) at which the algorithm gets out of the inner cycle at the point \(x_k\).

From Step 2, Step 3, and (13), we know that

\[
f_{i(k)} \geq f_{k+1} + \eta ac^{p_k}.
\]

So

\[
f_{i(k+1)} \leq f_{i(i(k))} - \eta ac^{p_{i(k)}}.
\]

Equation (14) and Lemma 3.5 deduce that

\[
p_{i(k)} \to \infty.
\]

The definition of the algorithm implies that \(\tilde{d}_{i(k)}\) which corresponds to the following subproblem is unacceptable:

\[
\min_{d \in \mathbb{R}^n} \tilde{g}_{i(k)} d \mid \frac{1}{2} d^\top B_{i(k)} d = \tilde{g}_{i(k)}(d),
\]

s.t. \(\|d\| \leq c^{-1} \hat{M}_{i(k)} \|g_{i(k)}\| = \frac{\alpha_{i(k)}}{c}\),

i.e.,

\[
\frac{f_{i(i(k))} - f\left(x_{i(k)} + \tilde{d}_{i(k)}\right)}{-\tilde{\Phi}_{i(k)}\left(\tilde{d}_{i(k)}\right)} < \eta.
\]
On the other hand, from Lemma 3.2, Step 2, and Step 3, it is similar to the proof of (14) that we can prove
\[ f_l(k+1) \leq f_l(k) - \frac{1}{2} \|g_l(k)\| \min \left\{ \alpha_l(k), \frac{\|g_l(k)\|}{B_l(k)} \right\}. \] (18)

Lemma 3.5 and (11) imply that
\[ \alpha_l(k) \to 0. \] (19)

From
\[ f(x_l(k) + \tilde{d}_l(k)) - f(x_l(k)) - \Phi_l(k)\left(\tilde{d}_l(k)\right) = O \left( \|\tilde{d}_l(k)\|^2 \right), \] (20)

and Lemma 3.2 and (20), we obtain
\[ \left| \frac{f(x_l(k) + \tilde{d}_l(k)) - f(x_l(k))}{\Phi_l(k)\left(\tilde{d}_l(k)\right)} - 1 \right| \leq \frac{O \left( \|\tilde{d}_l(k)\|^2 \right)}{(1/2) \|g_l(k)\| \min \{\alpha_l(k)/c, \|g_l(k)\|/B_l(k)\}} \] (21)

Equations (19) and (21) imply that
\[ \frac{f(x_l(k) + \tilde{d}_l(k)) - f(x_l(k))}{\Phi_l(k)\left(\tilde{d}_l(k)\right)} \to 1. \] (22)

It follows from the definition of \( f_l(k) \) that
\[ \frac{f_l(l(k)) - f(x_l(k) + \tilde{d}_l(k))}{-\Phi_l(k)\left(\tilde{d}_l(k)\right)} \geq \frac{f(x_l(k)) - f(x_l(k) + \tilde{d}_l(k))}{-\Phi_l(k)\left(\tilde{d}_l(k)\right)}. \] (23)

From (22) and (23), we have that when \( k \) is sufficiently large, the following formula holds:
\[ \frac{f_l(l(k)) - f(x_l(k) + \tilde{d}_l(k))}{-\Phi_l(k)\left(\tilde{d}_l(k)\right)} > \eta, \quad \forall 0 < \eta < 1. \]

This contradicts (17). The contradiction shows that the theorem is true. \( \square \)

**Theorem 3.2.** Suppose that Assumption 3.1 holds, \( f(x) \) is twice continuously differentiable, \( x_k \to x^* \) and \( \nabla^2 f(x^*) \) is positive definite, and \( \nabla^2 f(x) \) is Lipschitz continuous in a neighborhood of \( x^* \). If \( B_k = \nabla^2 f(x_k), \forall k \), then \( \{x_k\} \) converges to \( x^* \) quadratically.

**Proof.** By assumption, we have that \( B_k = B_k \) and \( B_k \) is a positive matrix, for \( k \) sufficiently large. Moreover, \( \tilde{d}_k = -B_k^{-1}g_k \) is the solution of the following subproblem:
\[ \min_{d \in \mathbb{R}^n} g_k^\top d + \frac{1}{2} d^\top B_k d = \Phi_k(d), \] (24)
\[ \text{s.t.} \quad \|d\| \leq \|g_k\| \tilde{M}_k. \]

We need only to prove that \( \tilde{d}_k \) is acceptable.

In fact, from Theorem 3.1 and the convergence of \( \{x_k\} \), we have that
\[ g_k \to 0, \]
which implies that
\[ \frac{\partial k}{\partial k} \to 0. \] (25)

Since \( \frac{\partial k}{\partial k} = -B_k^{-1} y_k, y_k = -B_k \frac{\partial k}{\partial k}, \)
\[ \Phi\left(\frac{\partial k}{\partial k}\right) - \frac{1}{2} \frac{\partial k}{\partial k} B_k \frac{\partial k}{\partial k} + \frac{1}{2} \frac{\partial k}{\partial k} B_k \frac{\partial k}{\partial k} \]
\[ = -\frac{1}{2} \frac{\partial k}{\partial k} B_k \frac{\partial k}{\partial k}. \] (26)

Because \( \nabla^2 f(x) \) is Lipschitz continuous in a neighborhood of \( x^* \), the following formula holds when \( k \) is sufficiently large:
\[ \left| f(x_k) - f\left( x_k + \frac{\partial k}{\partial k}\right) + \Phi_k\left( \frac{\partial k}{\partial k}\right) \right| \leq \left\| \frac{\partial k}{\partial k} \right\|^2 \left| \int_0^1 \left\| \nabla^2 f(x_k) - \nabla^2 f\left( x_k + \theta \frac{\partial k}{\partial k}\right) \right\| (1 - \theta) \, d\theta \right| \]
\[ \leq \frac{L}{2} \left\| \frac{\partial k}{\partial k} \right\|^3, \] (27)
where \( L \) is the Lipschitz constant.

Following from (25)-(27) and \( B_k \to \nabla^2 f(x^*) \) (\( \nabla^2 f(x^*) \) is positive), we obtain
\[ \left| f(x_k) - f\left( x_k + \frac{\partial k}{\partial k}\right) - \Phi_k\left( \frac{\partial k}{\partial k}\right) \right| - 1 \leq \left( \frac{L}{2} \right) \left( \frac{\partial k}{\partial k} \right)^3 \to 0. \]

Thereby, when \( k \) is sufficiently large, for \( 0 < \eta < 1 \), we have
\[ \frac{f(x_k) - f\left( x_k + \frac{\partial k}{\partial k}\right)}{-\Phi_k\left( \frac{\partial k}{\partial k}\right)} > \eta. \]

The definition of \( f_i(k) \) implies that
\[ \frac{f_i(k) - f\left( x_k + \frac{\partial k}{\partial k}\right)}{-\Phi_k\left( \frac{\partial k}{\partial k}\right)} \geq \frac{f(x_k) - f\left( x_k + \frac{\partial k}{\partial k}\right)}{-\Phi_k\left( \frac{\partial k}{\partial k}\right)} > \eta, \quad \text{for } k \text{ sufficiently large.} \]

So from the definition of the algorithm, we have that \( x_{k+1} = x_k + \frac{\partial k}{\partial k} \) when \( k \) is sufficiently large. Since \( \frac{\partial k}{\partial k} \) is acceptable, the nonmonotone adaptive trust region method is equivalent to the standard Newton method. This completes the proof. \( \square \)

4. NUMERICAL RESULTS

In this section, numerical results are reported on the 18 problems in [3] for both traditional trust region methods with different initial trust region radius and the adaptive trust region method proposed in [1] and the nonmonotone adaptive trust region method. All programs are written in MATLAB with double precision. The stopping criterion used is \( \|g_k\| < \epsilon \), where \( \epsilon = 10^{-11} \). For comparison, the quadratic subproblems are solved precisely and all of the algorithms use the same subroutine to solve the quadratic subproblems.

The traditional trust region method used here is the method described in [9] and \( B_k \) is obtained by the BFGS update. The radius of the trust region in [9] is determined as follows:
\[ \Delta_{k+1} = \begin{cases} \frac{c_3 \| s_k \| + c_4 \Delta_k}{2}, & \text{if } r < c_2, \\ \frac{(1 + c_1) \Delta_k}{2}, & \text{if } r \geq c_2, \end{cases} \]
where \( \eta = 0.1, c_1 = 2, c_2 = 0.25, c_3 = 0.25, \) and \( c_4 = 0.5. \) For the adaptive trust region method and the nonmonotone trust region method, \( B_k \) is also obtained by the BFGS update. In the computation, we choose \( \eta = 0.1, c = 0.5, \) and \( N = 2 \cdot n, \) where \( n \) is the dimension of the problem. However, we found that the choice of \( c \) has little impact on the computational efficiency. The detailed results are summarized in the following Table 1. Table 1 can be read as follows.

- Column 1 represents the problem number (Prob. No.).
- Column 2 shows the problem size or dimension (Prob. Size).
- Columns 3–7 report the numerical results of various algorithms.
- \( \Delta_0 \) denotes the initial trust radius.
- In columns 3–7, I, F, and G represent the numbers of iterations, function evaluations, and gradient evaluations.

Table 1. Numerical results of some test problems.

<table>
<thead>
<tr>
<th>Prob No.</th>
<th>Prob Size</th>
<th>Tradition ( \Delta_0 = 0.01 )</th>
<th>Tradition ( \Delta_0 = 1 )</th>
<th>Tradition ( \Delta_0 = 100 )</th>
<th>Adaptive I-F-G</th>
<th>Nonmonotone Adaptive I-F-G</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>24-25-25</td>
<td>16-17-15</td>
<td>failed</td>
<td>30-31-20</td>
<td>19-19-14</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>50-51-41</td>
<td>56-57-47</td>
<td>146-147-57</td>
<td>107-108-83</td>
<td>24-24-18</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>50-51-50</td>
<td>37-38-37</td>
<td>( &gt; 500 )</td>
<td>42-43-31</td>
<td>45-45-34</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>79-80-75</td>
<td>77-78-70</td>
<td>206-207-141</td>
<td>126-127-87</td>
<td>91-10-80</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>( &gt; 500 )</td>
<td>( &gt; 500 )</td>
<td>( &gt; 500 )</td>
<td>250-251-200</td>
<td>180-180-155</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>34-35-34</td>
<td>35-36-26</td>
<td>21-22-20</td>
<td>27-28-16</td>
<td>53-53-18</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>( &gt; 500 )</td>
<td>70-71-31</td>
<td>83-84-35</td>
<td>223-224-106</td>
<td>84-84-28</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>( &gt; 500 )</td>
<td>( &gt; 500 )</td>
<td>( &gt; 500 )</td>
<td>56-57-56</td>
<td>( &gt; 500 )</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>77-78-65</td>
<td>112-113-94</td>
<td>168-169-130</td>
<td>106-107-90</td>
<td>80-80-75</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>25-26-26</td>
<td>30-31-23</td>
<td>55-56-36</td>
<td>31-32-32</td>
<td>29-29-24</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>138-139-104</td>
<td>113-114-81</td>
<td>126-127-82</td>
<td>143-144-78</td>
<td>113-113-76</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>46-47-33</td>
<td>112-113-80</td>
<td>114-115-78</td>
<td>72-73-32</td>
<td>37-37-31</td>
</tr>
</tbody>
</table>

From the table, we know that the algorithm is efficient, especially for Problems 3, 4, 6, 9, 17 and 18. The nonmonotone adaptive trust region algorithm solves these problems with the least number of iterations. If we compare the nonmonotone adaptive trust region algorithm with the adaptive trust region algorithm, we observe that for most problems the former performs better than the latter.

5. CONCLUDING REMARKS

In this paper, we combine the subproblem proposed in [1] with the nonmonotone technique to propose a nonmonotone adaptive trust region algorithm. Theoretical analysis shows that the method possesses global and superlinear convergence properties, and the numerical results show that the method is very efficient. In this paper, we solve the subproblem exactly. In fact, as pointed in [1,18], we need not solve the subproblem exactly. We can use the approximate algorithms proposed in [1,5,8,18] to solve the subproblem in order to save time.
In this paper, we do not test our algorithm with some large-scale problems. There are two reasons for this. First, although the dimensions of these problems are not large, they are very difficult to solve. Second, because our main contribution in this paper is to propose a new nonmonotone trust region method and to compare the numerical results with the traditional trust region algorithms and monotone adaptive trust region method rather than to propose an algorithm to solve the traditional subproblem approximately and efficiently, we need not solve large-scale problems and compare the results with other results in the literatures.

REFERENCES