Applied Mathematical Modelling 38 (2014) 3003–3015

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

A new adaptive trust-region method for system of nonlinear equations

Hamid Esmaeili [a,*], Morteza Kimiaei [b]

[a] Department of Mathematics, Faculty of Science, Bu-Ali Sina University, Hamedan, Iran
[b] Department of Mathematics, Faculty of Science, Razi University, Kermanshah, Iran

ARTICLE INFO

Article history:
Received 25 March 2012
Received in revised form 28 May 2013
Accepted 22 November 2013
Available online 13 December 2013

Keywords:
Nonlinear equations
Trust-region framework
Adaptive radius
Nonmonotone technique
Convergence theory

ABSTRACT

This study presents a new trust-region procedure to solve a system of nonlinear equations in several variables. The proposed approach combines an effective adaptive trust-region radius with a nonmonotone strategy, because it is believed that this combination can improve the efficiency and robustness of the trust-region framework. Indeed, it decreases the computational cost of the algorithm by decreasing the required number of subproblems to be solved. The global and the quadratic convergence of the proposed approach is proved without any nondegeneracy assumption of the exact Jacobian. Preliminary numerical results indicate the promising behavior of the new procedure to solve systems of nonlinear equations.

Published by Elsevier Inc.

1. Introduction

Let $F : \mathbf{R}^n \to \mathbf{R}^n$ be a continuously differentiable mapping of the form $F(x) = (F_1(x), F_2(x), \ldots, F_n(x))^T$, and consider the following system of nonlinear equations

$$F(x) = 0, \quad x \in \mathbf{R}^n. \tag{1}$$

This class of problems often arises in a wide variety of contexts such as applied mathematics, science, engineering, industry, and is conceptually close to both constrained and unconstrained optimization problems. This means that (1) plays a primary role in the model formulation design and analysis of numerical techniques employed in solving optimization problems, complementarity problems, and variational inequalities. It is noticed that (1) is equivalent to the nonlinear unconstrained least-squares problem

$$\text{Minimize } f(x) = \frac{1}{2}\|F(x)\|^2, \quad \text{subject to } x \in \mathbf{R}^n, \tag{2}$$

where $\|\cdot\|$ denotes the Euclidean norm. Nonlinear least-squares problems have been comprehensively studied by many authors, so there are various iterative procedures to solve it, such as Newton and quasi-Newton method [1–6], Gauss–Newton method [7–10], Levenberg–Marquardt method [11–13], derivative-free method [4], subspace method [14], tensor method [15,16], spectral gradient method [17] and trust-region-based method [18–21,9,22–25].

* Corresponding author. Tel.: +98 8118280444.
E-mail addresses: esmaeili@basu.ac.ir (H. Esmaeili), Morteza.Kimiaei@gmail.com (M. Kimiaei).

0307-904X/$ - see front matter Published by Elsevier Inc.
http://dx.doi.org/10.1016/j.apm.2013.11.023

The trust-region framework for solving the nonlinear least-squares problem (2) is a popular class of iterative procedures. At each iterate x_k, it generates a trial step d_k by computing an exact or approximate solution of the following subproblem

$$\begin{aligned} &\text{Minimize } m_k(x_k+d)=\tfrac{1}{2}\|F_k+J_kd\|^2=f_k+d^TJ_k^TF_k+\tfrac{1}{2}d^TJ_k^TJ_kd,\\ &\text{subject to } d\in\mathbf{R}^n \quad \text{and} \quad \|d\|\leqslant\Delta_k, \end{aligned} \tag{3}$$

where $f_k=f(x_k),\ F_k=F(x_k),\ J_k=F'(x_k)$ is the Jacobian of $F(x)$, and $\Delta_k>0$. The ratio r_k of the actual to the predicted reductions is defined by

$$r_k=\frac{f_k-f(x_k+d_k)}{m_k(x_k)-m_k(x_k+d_k)}. \tag{4}$$

If $r_k\geqslant\mu,\ \mu\in(0,1)$, the trial step d_k will be accepted, leading to $x_{k+1}=x_k+d_k$, so the trust-region radius Δ_k can be updated. Otherwise, the trust-region radius should be diminished and the subproblem (3) will be solved again.

It is known that the traditional trust-region framework is extremely sensitive to the initial radius and its updating process, see [26,18,27]. Therefore, the total number of subproblems to be solved will be increased whenever the trust-region radius Δ_k is remarkably large. This may lead to increasing the computational costs. On the other hand, the intensely small trust-region radius Δ_k causes an increase in the total number of iterates so that the efficiency of the method will possibly be reduced. Hence, many researchers have investigated the best trust-region radius, but no one has actually claimed a general rule to generate it. As a consequence, some adaptive processes for determining radius have been proposed to decrease the total number of subproblems to be solved, see [18,21,13,23,25,28]. For example, Zhang and Wang [25] proposed an adaptive radius by

$$\Delta_k=c^{p_k}\|F_k\|^{\delta}, \tag{5}$$

where $0<c<1$ and $0.5<\delta<1$ are constants, and p_k is a non-negative integer. The technique (5) has some disadvantages: Firstly, the sequence generated by this method is superlinearly convergent of order 2δ. Secondly, efficiency of the method is largely dependent on the choice of δ. Finally, it can not prevent adequately from introducing the intensely small trust-region radius. The theoretical and computational results of the proposed radius from Fan and Pan [21] suggest that $\Delta_k=c^{p_k}M\|F_k\|$ is also another satisfactory radius, in which M is a constant. This choice of the trust-region radius plays an important role in proving the quadratic convergence and also prevents some deal from introducing the intensely small trust-region radius. However, if $\|F_k\|$ is very small, the constant M must be chosen so large that the radius is not too small. For problems in which $\|F_k\|$ is large, $M\|F_k\|$ will be very large and the number of subproblems to be solved may be increased. Thus, the amount of computation and the cost for solving the problem will be increased.

In order to overcome the above disadvantages of these methods, we introduce a modified adaptive radius strategy based on a nonmonotone technique and employ it in a trust-region framework. The new method generates a suitable trust-region radius to decrease the total number of subproblems to be solved. In addition, we investigate the global convergence to first-order stationary points of the proposed algorithm and establish the quadratic convergence properties without any nondegeneracy assumption of the exact Jacobian J_k. To illustrate the efficiency and robustness of the proposed algorithm in practice, we report some numerical experiments.

The rest of this paper is organized as follows. In Section 2, we describe our new algorithm and introduce its properties. Section 3 is devoted to investigating the global and quadratic convergence properties of the proposed algorithm. Numerical results are provided in Section 4 to show that our proposed method is well promising for systems of nonlinear equations. Finally, some concluding remarks are given in Section 5.

2. Motivation and algorithmic structure

A trust-region-based algorithm for solving a system of nonlinear equations will be presented in this section. After proposing an adaptive trust-region radius based on nonmonotone technique, we add it into trust-region framework to construct a more effective procedure.

To guarantee the global convergence of the traditional optimization approaches, it is well-known that we generally need to use a globalization technique, like line search or trust-region. These globalization techniques mostly enforce a monotonicity of the sequence of objective function values which usually result in producing short steps. Due to this fact, a slow numerical convergence is created for highly nonlinear problems, see [29–31,18,32–34]. For example, the traditional trust-region framework exploits the ratio (4) which leads to

$$f_k-f_{k+1}\geqslant m_k(x_k)-m_k(x_k+d_k)>0,$$

so that the sequence $\{f_k\}$ should be monotone. In order to avoid this drawback of the Armijo-type line search globalization techniques, the first nonmonotone strategy was introduced by Grippo et al. [33] for unconstrained optimization problems. In particular, they changed the Armijo rule by

$$f(x_k + \alpha_k d_k) \leqslant f_{l(k)} + \delta \alpha_k g_k^T d_k, \tag{6}$$

where $\delta \in (0,1)$, g_k is the gradient of $f(x)$ in x_k and

$$f_{l(k)} = \max_{0 \leqslant j \leqslant m(k)} \{f_{k-j}\}, \quad k \in \mathbf{N} \cup \{0\}, \tag{7}$$

in which $m(0) = 0$ and $0 \leqslant m(k) \leqslant \min\{m(k-1)+1, N\}$ with $N \geqslant 0$. The theoretical and numerical results show that the new technique has remarkable positive effects on Armijo-type line searches to get faster global convergence properties, especially for highly nonlinear problems. These excellent results are attracting many researchers to investigate more about the effects of these strategies in a wide variety of optimization procedures and propose new nonmonotone techniques, see [29,30,18,33,34]. As a prominent example, the first use of nonmonotone techniques in trust-region framework was introduced and analyzed by Deng et al. [32]. Ahookhosh and Amini [29] and Ahookhosh et al. [31] introduced a new nonmonotone strategy and applied it to both trust-region and line search procedures for unconstrained optimization. These techniques employ the following nonmonotone term

$$R_k = \eta_k f_{l(k)} + (1 - \eta_k) f_k, \tag{8}$$

where $\eta_k \in [\eta_{\min}, \eta_{\max}]$, $\eta_{\min} \in [0,1)$ and $\eta_{\max} \in [\eta_{\min}, 1]$. It is clear that the nonmonotonicity of (8) can be adjusted by selecting an adaptive process for η_k to makes (8) more relaxed for practical usage. Based on (8), the ratio (4) can be changed like

$$\hat{r}_k = \frac{R_k - f(x_k + d_k)}{m_k(x_k) - m_k(x_k + d_k)}. \tag{9}$$

The theoretical analysis and computational experiments support the promising behavior of this nonmonotone technique for unconstrained optimization problems. As discussed in [30,31,27] for an Armijo-type line search, it is generally believed that the best results can be obtained when a stronger nonmonotone term is used far away from the optimum while a weaker one is used close to the optimum. This also means that being far away from the optimum, a larger steplength will be used while being close to it, a smaller one can be employed. We believe that the same idea is true for trust-region radius. Hence, we take the advantage of the nonmonotone technique to introduce a new adaptive radius by

$$\Delta_k = \begin{cases} c^{p_k} \widehat{R}_k, & \text{if } k = 0; \\ c^{p_k} \max\{\widehat{R}_k, \Delta_{k-1}\}, & \text{if } k \geqslant 1, \end{cases} \tag{10}$$

where

$$\widehat{R}_k = \eta_k F_{l(k)} + (1 - \eta_k)\|F_k\|, \tag{11}$$

$\eta_k \in [\eta_{\min}, \eta_{\max}]$, $\eta_{\min} \in [0,1)$, $\eta_{\max} \in [\eta_{\min}, 1]$ and

$$F_{l(k)} = \max_{0 \leqslant j \leqslant m(k)} \{\|F_{k-j}\|\}, \quad k \in \mathbf{N} \cup \{0\}, \tag{12}$$

with $m(0) = 0$, $0 \leqslant m(k) \leqslant \min\{m(k-1)+1, N\}$ and $N \geqslant 0$. Using an adaptive parameter η_k, we can get larger radius when the current point is far away from the optimum and smaller one when being close to it. Because the proposed nonmonotone trust-region methods in [29,31] can not produce an appropriate radius, the total number of iterations and CPU time are increased in general. To overcome this drawback, we introduce the new adaptive radius technique which have advantages of nonmonotone technique provided by Ahookhosh and Amini [29], Ahookhosh et al. [31] and prevent the production of very large or intensely small radius. As a result, the proposed adaptive radius has some benefits as follows:

- Since $\widehat{R}_k$ is greater than $\|F_k\|$ (see (18)) and reduce slowly, it does not introducing the intensely small trust-region radius as possible and prevents from increasing the total number of iterates.
- Due to the decreasing sequence $\{\widehat{R}_k\}$, Δ_k will not stay too large and prevent from increasing the number of subproblems to be solved.

In this way, we are able to produce the suitable adaptive radius, while controlling the size of radius by a similar idea of nonmonotone technique. Now, the new approach can be characterized as a combination of the proposed radius and the trust-region framework for solving system of nonlinear equations. In particular, the approach firstly determines the trust-region radius using (10) and then generates the trial point d_k by exactly or approximately solving the trust-region subproblem. Afterwards, it employs the ratio (4) as a measure of the acceptance of the new point and an updating process of trust-region radius for the next step. On the basis of this discussion, the algorithmic framework of our approach can be outlined as follows:

Algorithm 1. Nonmonotone Adaptive Trust-Region Algorithm (NATR)

Input: An initial point $x_0 \in \mathbf{R}^n$, k_{max}, c, $\mu \in (0,1)$, $\eta_0 \in [\eta_{min}, \eta_{max}]$, $N > 0$ and $\epsilon > 0$.
Begin
$\Delta_0 \leftarrow \widehat{R}_0$; $\widehat{R}_0 \leftarrow \|F_0\|$; $r_0 \leftarrow 0; k \leftarrow 0$;
While ($\|F_k\| \geqslant \epsilon$ **and** $k \leqslant k_{max}$) {**Start of outer loop**}
While ($r_k < \mu$) {**Start of inner loop**}
Step 1: {**Step calculation**}
Specify the trial point d_k by solving the subproblem (3);
Step 2: {**Trial point acceptance**}
Determine the trust-region ratio r_k using (4);
If $r_k < \mu$
Update the trust-region radius by $\Delta_k = c\Delta_k$;
End If
End While {**End of inner loop**}
$x_{k+1} \leftarrow x_k + d_k$;
$F_{k+1} \leftarrow F(x_{k+1})$;
$f_{k+1} \leftarrow 1/2\|F_{k+1}\|^2$;
Step 3: {**Parameters update**}
Update the approximate Jacobin J_{k+1} by a finite differences formula;
Generate η_{k+1} by an adaptive formula;
$m(k+1) \leftarrow \min\{m(k)+1, N\}$;
Calculate $F_{l(k+1)}$ using (12) and then specify $\widehat{R}_{k+1}$ by (11);
Step 4: {**Trust-region radius determination**}
$p \leftarrow 0$;
Determine Δ_{k+1} using (10);
$k \leftarrow k+1$;
End While {**End of outer loop**}
End

Here, the loops containing Steps 1–2 and Steps 1–4 are called inner cycle and outer cycle, respectively.

3. Convergence theory

Since 1970 when the trust-region framework was introduced by Powell [35] for unconstrained optimization, the trust-region approaches have possessed a noteworthy reputation regarding their strong theoretical convergence properties. In this section, we will investigate the global and the quadratic convergence results of the proposed algorithm given in Section 2.

To verify the convergence analysis of the proposed algorithm, the following assumptions are required:

(H1) Let the level set $L(x_0) = \{x \in \mathbf{R}^n | f(x) \leqslant f(x_0)\}$ be bounded for any given $x_0 \in \mathbf{R}^n$, and $F(x)$ be continuously differentiable on a compact convex set Ω containing the level set $L(x_0)$.

(H2) The sequence of matrices $\{J_k\}$ is uniformly bounded above, i.e. there exists a constant $M_1 > 0$ such that

$$\|J_k\| \leqslant M_1, \quad \text{for all } k \in \mathbf{N} \cup \{0\}. \tag{13}$$

Moreover, we assume that $M_0\|F_k\| \leqslant \|J_k^T F_k\|$, for a constant $M_0 > 0$ and all $k \in \mathbf{N} \cup \{0\}$.

(H3) The decrease on the model m_k is at least as much as a fraction of the decrease obtained by the Cauchy point, i.e. there exists a constant $\beta \in (0,1)$ such that

$$m_k(x_k) - m_k(x_k + d_k) \geqslant \beta \|J_k^T F_k\| \min\left\{\Delta_k, \frac{\|J_k^T F_k\|}{\|J_k^T J_k\|}\right\}, \tag{14}$$

for all $k \in \mathbf{N} \cup \{0\}$.

(H4) J_k is Lipschitz continuous on $L(x_0)$, with Lipschitz constant $2\gamma_L$.

Lemma 3.1. *Suppose that d_k is a solution of the subproblem (3) such that*

$$m_k(x_k) - m_k(x_k + d_k) \geqslant m_k(x_k) - m_k(x_k + d_k^{CP}), \tag{15}$$

where d_k^{CP} is the Cauchy point. Then the condition (14) holds.

Proof. To see a proof of this lemma, one can observe the Lemma 2.1 in [22]. □

Lemma 3.2. *Suppose that (H4) holds, the sequence $\{x_k\}$ is generated by Algorithm 1 and d_k is a solution of the subproblem (3) such that $\|F_k + J_k d_k\| \leqslant \|F_k\|$. Then, we have*

$$|f(x_k + d_k) - m_k(x_k + d_k))| \leqslant O(\|d_k\|^2). \tag{16}$$

Proof. Using Taylor's theorem, we can write

$$F(x_k + d_k) = F(x_k) + \int_0^1 J(x_k + td_k)d_k\,dt.$$

So

$$\|F(x_k + d_k)\|^2 = \|F(x_k) + J(x_k)d_k + w(x_k, d_k)\|^2,$$

where

$$w(x_k, d_k) = \int_0^1 [J(x_k + td_k) - J(x_k)]d_k\,dt.$$

From the Lipschitz continuity of $J(x_k)$ and $\|F_k + J_k d_k\| \leqslant \|F_k\|$, we obtain

$$\begin{aligned}
|m_k(x_k + d_k) - f(x_k + d_k)| &= \frac{1}{2}\left|\|F(x_k) + J(x_k)d_k\|^2 - \|F(x_k + d_k)\|^2\right| \\
&= \frac{1}{2}\left|\|F(x_k) + J(x_k)d_k\|^2 - \|F(x_k) + J(x_k)d_k + w(x_k, d_k)\|^2\right| \\
&\leqslant \frac{1}{2}\left(2(F(x_k) + J(x_k)d_k)w(x_k, d_k) + \|w(x_k, d_k)\|^2\right) \\
&\leqslant \|F(x_k) + J(x_k)d_k\|\|w(x_k, d_k)\| + \frac{1}{2}\|w(x_k, d_k)\|^2 \\
&\leqslant \|F_k\|\left(\gamma_L\|d_k\|^2\right) + \frac{1}{2}(\gamma_L\|d_k\|^2)^2 \\
&\leqslant \left(\gamma_L\|F_k\| + \frac{1}{2}\gamma_L^2\|d_k\|^2\right)\|d_k\|^2 = O(\|d_k\|^2). \quad \square
\end{aligned}$$

The following lemma indicates that the inner cycle of the Algorithm 1 can be left in a finite number of internal iterates.

Lemma 3.3. *Suppose that (H2), (H3) and (H4) hold and the sequence $\{x_k\}$ is generated by Algorithm 1. Then, the inner cycle of Algorithm 1 is well-defined.*

Proof. Assume that the inner cycle of Algorithm 1 cycles infinity, i.e., $\Delta_k^p \to 0$ as $p \to \infty$. Using the fact that x_k is not the optimum of (2), we can conclude that there exists a constant $\epsilon > 0$ such that $\|F_k\| \geqslant \epsilon$. These facts, (H2) and (14), suggest

$$\begin{aligned}
m_k(x_k) - m_k(x_k + d_k^p) &\geqslant \beta\|J_k^T F_k\| \min\left\{\Delta_k^p, \frac{\|J_k^T F_k\|}{\|J_k^T J_k\|}\right\} \\
&\geqslant \beta M_0 \epsilon \min\left\{\Delta_k^p, \frac{M_0\epsilon}{M_1^2}\right\} \\
&\geqslant \beta M_0 \epsilon \Delta_k^p,
\end{aligned} \tag{17}$$

where d_k^p is a solution of subproblem (3) corresponding to p in k-th iterate. Now, Lemma 3.2 and (17) lead to

$$\left|\frac{f_k - f(x_k + d_k^p)}{m_k(x_k) - m_k(x_k + d_k^p)} - 1\right| = \left|\frac{f(x_k + d_k^p) - m_k(x_k + d_k^p)}{m_k(x_k) - m_k(x_k + d_k^p)}\right| \leqslant \frac{O(\|d_k^p\|^2)}{\beta M_0 \epsilon \Delta_k^p} \leqslant \frac{O((\Delta_k^p)^2)}{\beta M_0 \epsilon \Delta_k^p} \to 0, \quad \text{as } p \to \infty.$$

Therefore, there exists a sufficiently large p, called p_k, such that

$$r_k = \frac{f_k - f(x_k + d_k^{p_k})}{m_k(x_k) - m_k(x_k + d_k^{p_k})} \geqslant \mu,$$

meaning that p_k is a finite integer number, i.e., the inner cycle of Algorithm 1 is well-defined. □

Lemma 3.4. *Suppose that the sequence $\{x_k\}$ is generated by Algorithm 1. Then, for all $k \in \mathbf{N} \cup \{0\}$, we have $x_k \in L(x_0)$ and $\{F_{l(k)}\}$ is a decreasing convergent sequence.*

Proof. Using the definition of $\widehat{R}_k$ and $F_{l(k)}$, we have

$$\|F_k\| = \eta_k\|F_k\| + (1-\eta_k)\|F_k\| \leqslant \widehat{R}_k = \eta_k F_{l(k)} + (1-\eta_k)\|F_k\| \leqslant \eta_k F_{l(k)} + (1-\eta_k)F_{l(k)} = F_{l(k)}. \tag{18}$$

Assume that x_{k+1} is accepted by Algorithm 1. This fact along with (18) implies that

$$\frac{\widehat{R}_k^2}{2} - \frac{\|F_{k+1}\|^2}{2} \geqslant f_k - f_{k+1} \geqslant \mu(m_k(x_k) - m_k(x_k + d_k)) > 0,$$

so that

$$\|F_{k+1}\| \leqslant \widehat{R}_k \leqslant F_{l(k)} \quad \forall k \in \mathbf{N}. \tag{19}$$

Now, we prove that the sequence $\{F_{l(k)}\}$ is a decreasing sequence. The proof is divided into two following cases.

(i) $k \geqslant N$. In this case, we have $m(k+1) \leqslant m(k)+1$, for all k. So, the definition of $F_{l(k+1)}$ and relationship (19) result in

$$F_{l(k+1)} = \max_{0\leqslant j\leqslant m(k+1)}\{\|F_{k+1-j}\|\} = \max\left\{\max_{0\leqslant j\leqslant m(k)}\{\|F_{k-j}\|\}, \|F_{k+1}\|\right\} \leqslant F_{l(k)}.$$

(ii) $k < N$. In this case, always $m(k) = k$. Since, for any k, $\|F_k\| \leqslant \|F_0\|$, we can see that

$$F_{l(k)} = \|F_0\|, \quad \forall k \in \mathbf{N}.$$

From the definition of $F_{l(k+1)}$, we have

$$\|F_{k+1}\| = \eta_{k+1}\|F_{k+1}\| + (1-\eta_{k+1})\|F_{k+1}\| \leqslant \eta_{k+1}F_{l(k+1)} + (1-\eta_{k+1})\|F_{k+1}\| = \widehat{R}_{k+1}, \quad \forall k \in \mathbf{N}. \tag{20}$$

Obviously, the definition of $\widehat{R}_k$ indicates that $\widehat{R}_0 = \|F_0\|$. By induction, assuming $x_i \in L(x_0)$, for all $i = 1, \ldots, k$, we prove $x_{k+1} \in L(x_0)$. From (11), (12), (20) and the decreasing sequence $F_{l(k)}$, we obtain

$$\|F_{k+1}\| \leqslant \widehat{R}_{k+1} \leqslant F_{l(k+1)} \leqslant F_{l(k)} \leqslant \|F_0\|.$$

Thus, the sequence $\{x_k\}$ is contained in $L(x_0)$. Since $x_k \in L(x_0)$ for all $k \in \mathbf{N} \cup \{0\}$ and $\{F_{l(k)}\}$ is a decreasing sequence, then we conclude that the sequence $\{F_{l(k)}\}$ is convergent. □

Lemma 3.5. *Suppose that (H1), (H2) and (H3) hold and there exists a constant $\kappa > 0$ such that $\kappa\|J_k^T F_k\| > \|d_k\|$. Suppose, furthermore, that the sequence $\{x_k\}$ is generated by Algorithm 1. Then, we have*

$$\lim_{k\to\infty} F_{l(k)} = \lim_{k\to\infty} \|F(x_k)\|.$$

Proof. By Lemma 7 in [29] and $f(x_k) = \frac{1}{2}\|F(x_k)\|^2$, we have

$$\lim_{k\to\infty} f_{l(k)} = \lim_{k\to\infty} f(x_k).$$

Then

$$\lim_{k\to\infty} F_{l(k)} = \lim_{k\to\infty} \|F(x_k)\|. \quad \square$$

According to Lemma 3.5 and relation (18), we can obtain the following corollary.

Corollary 3.1. *Suppose that sequence $\{x_k\}$ is generated by Algorithm 1. Then, we have*

$$\lim_{k\to\infty} \widehat{R}_k = \lim_{k\to\infty} \|F(x_k)\|.$$

In order to establish the global convergence of Algorithm 1, one needs to establish the following results.

Lemma 3.6. *Suppose that (H2) and (H3) hold, the sequence $\{x_k\}$ is generated by Algorithm 1 and d_k is a solution of the subproblem (3). Then, we have*

$$m_k(x_k) - m_k(x_k + d_k) \geqslant L_k\|F_k\|^2, \tag{21}$$

where $L_k = \beta M_0 \min\left\{c^{p_k}, \frac{M_0}{M_1^2}\right\}$.

Proof. Using (H2), (H3) and (10), we have

$$\begin{aligned} m_k(x_k) - m_k(x_k + d_k) &\geqslant \beta \|J_k^T F_k\| \min\left\{\Delta_k, \tfrac{\|J_k^T F_k\|}{\|J_k^T J_k\|}\right\} \\ &= \beta \|J_k^T F_k\| \min\left\{c^{p_k} \max\{\widehat{R}_k, \Delta_{k-1}\}, \tfrac{\|J_k^T F_k\|}{\|J_k^T J_k\|}\right\} \\ &\geqslant \beta \|J_k^T F_k\| \min\left\{c^{p_k} \widehat{R}_k, \tfrac{\|J_k^T F_k\|}{\|J_k^T J_k\|}\right\} \\ &\geqslant \beta M_0 \|F_k\| \min\left\{c^{p_k}(\eta_k F_{l(k)} + (1-\eta_k)\|F_k\|), \tfrac{M_0\|F_k\|}{M_1^2}\right\} \\ &\geqslant \beta M_0 \|F_k\| \min\left\{c^{p_k}(\eta_k \|F_k\| + (1-\eta_k)\|F_k\|), \tfrac{M_0\|F_k\|}{M_1^2}\right\} \\ &\geqslant \beta M_0 \|F_k\| \min\left\{c^{p_k}\|F_k\|, \tfrac{M_0\|F_k\|}{M_1^2}\right\} \\ &= L_k \|F_k\|^2, \end{aligned}$$

where $L_k = \beta M_0 \min\left\{c^{p_k}, \frac{M_0}{M_1^2}\right\}$. Therefore, the proof is completed. □

In this point, based on the mentioned assumptions of this section, the global convergence of Algorithm 1 can be investigated.

Theorem 3.7. *Suppose that (H1)–(H4) hold. Then, Algorithm 1 either stops at a stationary point of $f(x)$ or generates an infinite sequence $\{x_k\}$ such that*

$$\lim_{k\to\infty} \|F_k\| = 0. \tag{22}$$

Proof. By contradiction, for all sufficiently large k, assume that there exists a constant $\epsilon > 0$ and an infinite subset $K \subseteq \mathbf{N} \cup \{0\}$ satisfying

$$\|F_k\| > \epsilon, \quad \text{for all } k \in K. \tag{23}$$

Using (21), (23) and $r_k > \mu$, we can write

$$f_k - f(x_k + d_k) \geqslant \mu[m_k(x_k) - m_k(x_k + d_k)] \geqslant \mu L_k \|F_k\|^2 \geqslant \mu\epsilon^2 L_k.$$

Now, by taking a limit from both sides of this inequality, as $k \to \infty$, we have that $\lim_{k\to\infty} L_k = 0$, i.e., $p_k \to \infty$ for sufficiently large $k \in K$. But this fact is possible only if $p_k \to \infty$, as $k \to \infty$ and $k \in K$. This clearly drives a contradiction with Lemma 3.3. Therefore, the hypothesis (23) is not true and the result of theorem is obtained. □

This theorem simply guarantees that the stopping criterion of Algorithm 1, that is $\|F_k\| < \epsilon$, eventually holds.

To establish the quadratic convergence rate of the sequence generated by Algorithm 1, some additional assumptions are further required. To this end, suppose that x_* is the solution of (1) and $N(x_*, \rho)$ is a neighborhood of it.

(H5) There exist constants $c_0 > 0$ and $\rho \in (0, 1)$ such that

$$\|F(x) - F(y) + J(y)(x - y)\| \leqslant c_0 \|x - y\|^2, \quad \text{for all } x, y \in N(x_*, \rho).$$

(H6) There exists a constant $c_1 \geqslant 1$ such that

$$c_1 \|x - x_*\| \leqslant \|F(x)\| = \|F(x) - F(x_*)\|, \quad \text{for all } x \in N(x_*, \rho).$$

The condition (H5) holds if $F(x)$ is continuously differentiable and $J(x)$ is Lipschitz continuous.

Theorem 3.8. *Suppose that (H1)-(H6) hold and let the sequence of $\{x_k\}$ be generated by the Algorithm 1 converging to x_*. Then, for sufficiently large k, we have*

$$x_{k+1} = x_k + d_k^0,$$

where d_k^0 is a solution of (3) corresponding to $p_k = 0$. Furthermore, the sequence $\{x_k\}$ is converged to x_* quadratically.

Proof. If d_k^0 is a solution of (3) corresponding to $p_k = 0$, then we first show that $x_{k+1} = x_k + d_k^0$, for sufficiently large k. The fact that d_k^0 is feasible for the subproblem (3), Corollary 3.1 and Theorem 3.7, we simply have

$$\|d_k^0\| \leqslant \Delta_k^0 = \max\{\widehat{R}_k, \Delta_{k-1}\} = \widehat{R}_{k_j} \to 0, \quad \text{as } k \to \infty.$$

where $0 \leqslant k_j \leqslant k$. This fact, Lemma 3.2 and (H3) suggest

$$\left|\frac{f_k - f(x_k + d_k^0)}{m_k(x_k) - m_k(x_k + d_k^0)} - 1\right| = \left|\frac{m_k(x_k + d_k^0) - f(x_k + d_k^0)}{m_k(x_k) - m_k(x_k + d_k^0)}\right| \leqslant \frac{O(\|d_k^0\|^2)}{\frac{1}{2}\epsilon\Delta_k^0} \leqslant \frac{O((\Delta_k^0)^2)}{\frac{1}{2}\epsilon\Delta_k^0} \to 0, \quad \text{as } k \to \infty.$$

Thus, for all sufficiently large k, the trial point d_k^0 is accepted by Algorithm 1, i.e. $x_{k+1} = x_k + d_k^0$.

In this point, the quadratic convergence of the sequence $\{x_k\}$. generated by Algorithm 1 is investigated. Paying attention to (H1), one can see that the level set $L(x_0)$ is bounded and $F(x)$ is continuously differentiable on compact convex set Ω containing $L(x_0)$. Therefore, there exists a constant $M_2 > 0$ such that

$$\|J_k\| \leqslant M_2, \quad \text{for all } x \in \Omega. \tag{24}$$

Hence, from (24) and the mean value theorem, one can easily get

$$\|F_k\| \leqslant \|F_k - F(x_*)\| \leqslant \|J(\xi)\|\|x_k - x_*\| \leqslant M_2\|x_k - x_*\|,$$

for all $x_k \in N(x_*, \rho)$ and $\xi \in [x_k, x_*]$. As a result of this fact and from Corollary 3.1, we can write

$$\widehat{R}_k \approx \|F_k\| \leqslant M_2\|x_k - x_*\|,$$

for all sufficiently large k and

$$\|d_k^0\| \leqslant \max\{\widehat{R}_k, \Delta_{k-1}\} = \widehat{R}_{k_j} \leqslant M_2\,\|x_k - x_*\|,$$

where $0 \leqslant k_j \leqslant k$. From (H6), it is clear that

$$\|x_k - x_*\| \leqslant \frac{1}{c_1}\|F_k\| \leqslant \frac{1}{c_1}\widehat{R}_k \leqslant \widehat{R}_k \leqslant \max\{\widehat{R}_k, \Delta_{k-1}\} = \Delta_k^0.$$

This fact directly implies that $x_k - x_*$ is a feasible point for the subproblem (3). Now, it straightforwardly follows from (H5) and (H6) that

$$\begin{aligned}\frac{1}{2}\|F_k + J_k d_k^0\|^2 = m_k(x_k + d_k^0) &\leqslant m_k(x_k + (x_* - x_k)) \leqslant \frac{1}{2}\|F_k + J_k(x_k - x_*)\|^2 + O(\|d_k^0\|)^4 \\ &\leqslant \frac{c_0^2}{2}\|x_k - x_*\|^4 + O(\|x_k - x_*\|)^4 = O(\|x_k - x_*\|^4).\end{aligned} \tag{25}$$

Hence, from (H6) and (25), we can conclude that

$$c_1\|x_{k+1} - x_*\| \leqslant \|F(x_k + d_k^0)\| \leqslant \|F_k + J_k d_k^0\| + O(\|d_k^0\|^2) \leqslant O(\|x_k - x_*\|^2 + O(\|x_k - x_*\|^2) = O(\|x_k - x_*\|^2).$$

Thus, there exists a positive constant κ such that

$$\lim_{k\to\infty}\frac{\|x_{k+1} - x_*\|}{\|x_k - x_*\|^2} = \lim_{k\to\infty}\frac{O(\|x_k - x_*\|^2)}{\|x_k - x_*\|^2} \leqslant \kappa.$$

This shows that the sequence $\{x_k\}$ generated by the Algorithm 1 is quadratically convergent. □

4. Preliminary numerical experiments

We now report the numerical results obtained by running Algorithm 1 (NATR) in comparison with the traditional trust-region algorithm (TTR), the adaptive trust-region algorithm of Zhang and Wang [25] (ATRZ), the adaptive trust-region algorithm of Fan and Pan [21] (ATRF) and the nonmonotone trust-region of Ahookhosh and Amini [29] (NTR) on a set of some systems of nonlinear equations. Test problems are selected from a wide range of literatures with the dimensions from 2 up to 200. In details, while the Problems 1–39 are discussed in Cruz et al. [36], the Problems 40–45 are selected from Lukšan and Vlček [37]. Furthermore, the Problems 46–51 are chosen from Moré et al. [38].

For all of these algorithms, the trust-region subproblems are coded by Steihaug–Toint procedure, see [27]. The Steihaug–Toint algorithm terminates at $x_k + d$ when

$$\|\nabla m_k(x_k + d)\| \leqslant \min\left\{0.1, \|\nabla m_k(0)\|^{\frac{1}{2}}\right\}\|\nabla m_k(0)\| \quad \text{or} \quad \|d\| = \Delta_k,$$

holds. The Jacobian matrix J_k is approximated using finite-differences formula as follows

$$[J_k]_{.j} \sim \frac{1}{h_j}(F(x_k + h_j e_j) - F_k),$$

where $[J_k]_{.j}$ denotes the j-th column of J_k, e_j is the j-th vector of the standard basis and

$$h_j = \begin{cases} \sqrt{\epsilon_m}, & \text{if } x_{k_j} = 0; \\ \sqrt{\epsilon_m}\,sign(x_{k_j})\max\left\{|x_{k_j}|, \frac{\|x_k\|_1}{n}\right\}, & \text{otherwise.}\end{cases}$$

If the point $x_k + h_j e_j$ is not feasible, the backward approximation

$$[J_k]_j \sim \frac{1}{h_j}\left(F(x_k - h_j e_j) - F_k\right),$$

is used.

All codes are written in MATLAB 9 programming environment with double precision format in the same subroutine. In our numerical experiments, the algorithms are stopped whenever

$$\|F_k\| \leqslant 10^{-5},$$

or the total number of iterates exceeds 2000. The latter case is denoted as "F" in Table 1. During the code implementation, we verified whether the different codes converged to the same point, and only provided data for problems where all algorithms converged to the identical point.

Table 1
Numerical results.

Problem name	Dim	TTR			ATRZ			ATRF			NTR			NATR		
		N_i	N_f	C_t	N_i	N_f	C_t	N_i	N_f	C_t	N_i	N_f	C_t	N_i	N_f	C_t
1. Exponential 1	100	6	7	0.0856	7	8	0.1012	6	7	0.0999	6	7	0.2362	6	7	0.0921
2. Exponential 2	100	4	5	0.1002	4	5	0.1081	4	5	0.1088	4	5	0.2146	4	5	0.0990
3. Extended Rosenbrock	100	18	23	0.1805	15	16	0.1823	11	18	0.1383	13	14	0.4803	12	30	0.1337
4. Chandrasekhar's H-equation	100	5	6	0.3023	7	8	0.4316	4	5	0.2490	5	6	0.4215	4	5	0.2424
5. Trigonometric	5	11	16	0.0039	8	12	0.0035	8	20	0.0050	11	16	0.0079	10	15	0.0029
6. Singular	100	19	20	0.2650	69	70	0.8950	35	36	0.4693	16	17	0.6935	15	16	0.2314
7. Logarithmic	100	6	7	0.0615	7	8	0.0779	5	6	0.0564	6	7	0.2424	4	5	0.0410
8. Broyden tridiagonal	100	6	7	0.0655	5	6	0.0635	5	6	0.0617	6	7	0.2145	5	6	0.0555
9. Trigexp	100	F	F	F	10	12	0.1342	12	45	0.2137	16	22	0.7215	12	33	0.1877
10. Variable band 1	200	12	14	1.8940	24	26	4.1660	10	20	1.9744	12	14	2.1185	9	12	1.5166
11. Variable band 2	200	16	20	2.7979	39	41	7.4664	11	24	2.4690	13	15	2.3806	11	14	1.9165
12. Function 15	100	10	11	0.1190	13	14	0.1617	11	12	0.1446	10	11	0.1984	11	12	0.1328
13. Strictly convex 1	100	5	6	0.0494	5	6	0.0568	5	6	0.0568	5	6	0.0757	5	6	0.0568
14. Strictly convex 2	100	9	10	0.2152	12	13	0.2674	8	9	0.2218	9	10	0.4246	8	9	0.2126
15. Function 18	99	20	28	0.2090	17	39	0.2212	13	72	0.2057	19	26	0.3097	15	46	0.1800
16. Zero Jacobian	100	5	6	0.0504	6	7	0.0717	5	6	0.0602	5	6	0.1599	5	6	0.1094
17. Geometric programming	50	15	16	2.8662	429	430	83.0233	242	243	46.6726	15	16	2.9407	15	16	2.8509
18. Function 21	99	7	8	0.0733	16	17	0.1978	7	8	0.0870	7	8	0.1063	7	8	0.0734
19. Linear function-full rank 1	100	8	9	0.0788	36	37	0.4139	1	2	0.0118	8	9	0.2585	1	2	0.0099
20. Linear function-full rank 2	100	2	3	0.0423	2	3	0.0449	2	3	0.0446	2	3	0.0901	2	3	0.0425
21. Penalty I	100	5	6	0.0473	32	33	0.3482	4	7	0.0462	5	6	0.0685	8	9	0.2896
22. Brown almost linear	100	2	3	0.0172	2	3	0.0244	2	3	0.0237	2	3	0.0497	2	3	0.0168
23. Variable dimensioned	100	16	17	0.1509	16	17	0.1775	16	17	0.1788	16	17	0.2964	16	17	0.1509
24. Geometric	50	12	13	2.4865	426	427	72.2884	239	240	40.5793	13	14	2.589	12	13	2.4982
25. Function 27	100	19	20	0.1959	16	17	0.1911	15	16	0.1804	19	20	0.2458	15	16	0.1524
26. Tridimensional valley	99	8	9	0.0949	7	8	0.2044	7	8	0.1574	8	9	0.1460	7	8	0.1313
27. Complementary	100	8	9	0.1183	9	10	0.1428	6	7	0.0959	8	9	0.3255	6	7	0.0895
28. Hanbook	100	4	6	0.2591	4	15	0.2767	3	24	0.2218	4	6	0.3296	4	18	0.2734
29. Tridiagnal system	100	142	220	2.4199	131	209	2.4555	116	427	4.7810	31	30	0.9565	126	223	2.4960
30. Five-diagonal system	5	16	19	0.0045	15	17	0.0045	17	55	0.0128	17	19	0.0074	15	30	0.0069
31. Extended Freud. and Roth	100	7	8	0.0705	8	9	0.1675	6	7	0.1095	7	8	0.1173	6	7	0.1101
32. Extended cragg and levy	100	35	40	0.3781	106	110	1.3125	47	64	0.5963	24	25	0.4730	20	33	0.2363
33. Extended Wood	100	6	7	0.0628	6	7	0.0757	6	7	0.0735	6	7	0.0790	6	7	0.0625
34. Triadiagnal exponential	100	5	6	0.0177	6	7	0.0297	3	4	0.0161	5	6	0.0288	3	4	0.0151
35. Discrete boundary value	100	2	3	0.1970	105	106	7.6429	37	38	3.2107	2	3	0.2403	2	3	0.1970
36. Brent	100	18	20	0.1904	17	20	0.2165	16	23	0.2164	22	23	0.4002	15	32	0.1964
37. Thorech	100	13	18	0.1309	11	16	0.1367	10	28	0.1307	17	18	0.3750	9	27	0.1045
38. Broyden banded	100	7	8	0.1481	7	8	0.1562	7	8	0.1560	7	8	0.3246	7	8	0.1485
39. Discrete integral equation	100	4	5	0.5385	4	5	0.5477	4	5	0.5464	4	5	1.7947	4	5	0.5383
40. Countercurrent reactors 1	120	14	18	0.5206	28	29	0.8702	15	24	0.6736	25	26	1.0774	14	25	0.8851
41. Singular Broyden	100	11	12	0.1250	13	14	0.1711	11	12	0.1494	11	12	1.7946	11	12	0.1269
42. Structured Jacobian	100	10	11	0.1197	14	15	0.1922	10	11	0.1386	10	11	0.2377	10	11	0.1199
43. Extended Powell Singular	100	14	15	0.1486	28	30	0.3450	15	25	0.1961	14	15	0.2436	13	18	0.1446
44. Generalized Broyden banded	100	7	8	0.1327	7	8	0.1402	7	8	0.1415	7	8	0.2593	7	8	0.1332
45. Extended powell badly scaled	100	79	114	0.8393	836	837	9.9276	144	187	1.7603	21	23	0.3259	53	59	0.5473
46. Rosenbrock	2	24	35	0.0048	20	37	0.0056	20	120	0.0157	16	22	0.0061	20	47	0.0062
47. Powell singular	4	14	16	0.0031	12	15	0.0034	11	21	0.0043	12	13	0.0056	11	16	0.0030
48. Powell badley scaled	2	176	269	0.0296	1112	1114	0.1998	470	516	0.0885	25	29	0.0114	131	139	0.0181
49. Helical valley	3	13	16	0.0027	12	18	0.0037	14	46	0.0076	11	12	0.0057	13	27	0.0044
50. Watson	31	15	16	0.1984	119	120	1.8010	61	62	0.9057	15	16	0.3547	15	16	0.1968
51. Chebyquad	4	8	9	0.0036	8	9	0.0038	10	44	0.0133	8	9	0.0048	8	9	0.0048

The NATR algorithm takes advantages of the parameters $\mu = 10^{-6}$, $c = 0.5$. The TTR and NTR algorithms employs the parameters $\mu_1 = 0.1,\ \mu_2 = 0.9$ and generates trust-region radius like [27] by the following formula

$$\Delta_{k+1} = \begin{cases} c_1\|d_k\|, & \text{if } r_k < \mu_1; \\ \Delta_k, & \text{if } \mu_1 \leqslant r_k \leqslant \mu_2; \\ c_2\Delta_k, & \text{if } r_k \geqslant \mu_2, \end{cases}$$

where $c_1 = 0.25$ and $c_2 = 0.3$. The NATR and NTR algorithms updates η_k by the following recursive formula

$$\eta_k = \begin{cases} \eta_0/2, & \text{if } k = 1; \\ (\eta_{k-1} + \eta_{k-2})/2, & \text{if } k \geqslant 2 \end{cases}$$

and employs the parameter $\eta_0 = 0.2$. We also follow the literature [39] in exploiting $\Delta_0 = 1$ as an initial trust-region radius for TTR and NTR. The parameters of ATRZ and ATRF are chosen the same as those proposed in articles [25,21], respectively.

The results for considered algorithms are summarized in Table 1. In this table, N_i, N_f and C_t, respectively, indicate the total number of iterates, the total number of function evaluations and CPU time. It follows from Table 1 that the presented algorithms solve all of test problems successfully and in most cases the total number of iterates, function evaluations, and CPU time of the NATR are less than those of others.

Using the performance profile of Dolan and Moré [40], we can demonstrate the overall behavior of the present algorithms and get more insights about their performance, based on N_i, N_f and C_t, that has been assessed in Figs. 1–3, respectively. Let $\mathcal{S}$ be the set of all algorithms and $\mathcal{P}$ the set of test problems, with $n_s = |\mathcal{S}|$ and $n_p = |\mathcal{P}|$. For each problem p and solver s, let $t_{p,s}$ denote the quantities we want to compare for problem p and solver s. The performance ratio is defined as

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,u} : u \in \mathcal{S}\}}, \tag{26}$$

in order to compare the performance of solver s on problem p with the best performance by any solver in $\mathcal{S}$ on p. If the algorithm s has failed on problem p, we set $r_{p,s} = r_{fail}$, where r_{fail} is strictly larger than any performance ratio (26). For any factor $\tau \geqslant 1$, the overall performance of algorithm s is given by

$$\rho_s(\tau) = \frac{1}{n_p}\ \text{size}\ \{p \in \mathcal{P} : r_{p,s} \leqslant \tau\}.$$

So performance profiles, for every $\tau \geqslant 1$, produce the proportion $\rho_s(\tau)$ of test problems on which each considered code has a performance within a factor τ of the best. Especially, $\rho_s(1)$ is the probability for that the solver s is the best, and $\lim_{\tau \to r_{fail}} \rho_s(\tau)$ gives the fraction of test problems of $\mathcal{P}$ for which the algorithm s succeeded. As a consequence, the values on the left side of

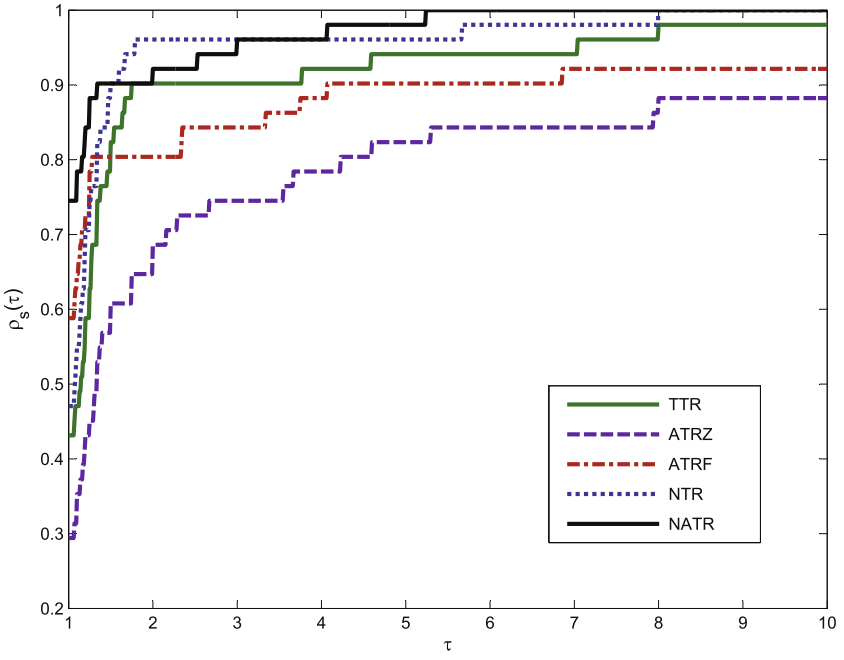

Fig. 1. Performance profile for the number of iterates.

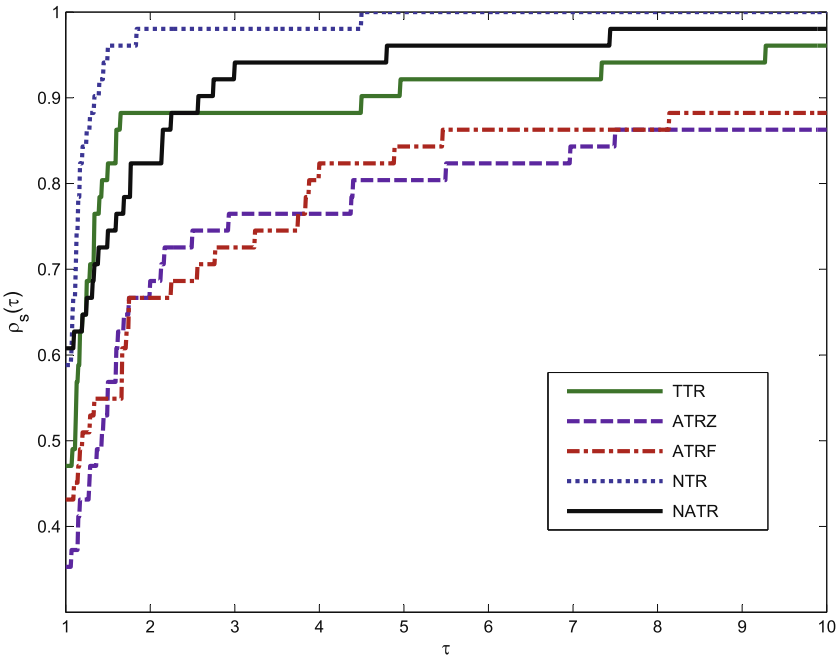

Fig. 2. Performance profile for the number of function evaluations.

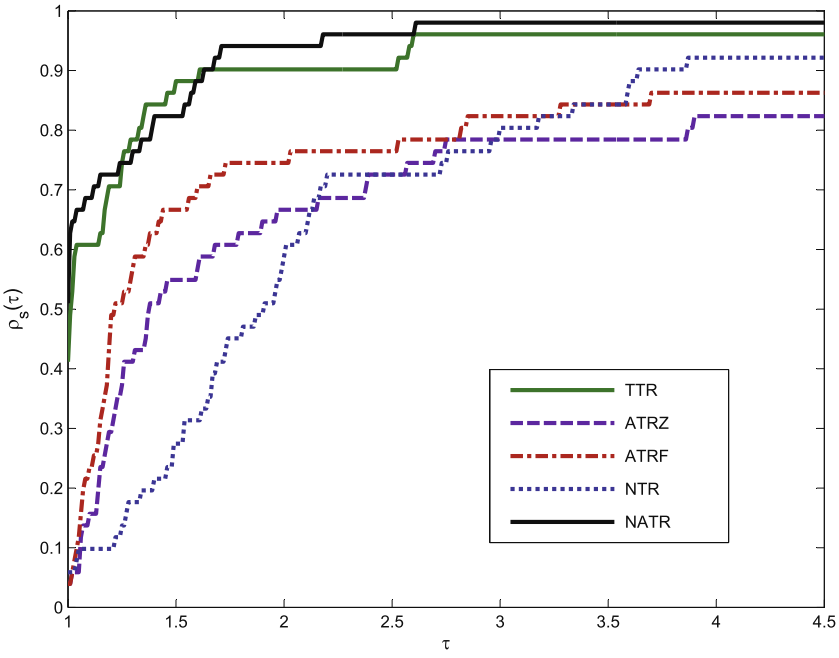

Fig. 3. Performance profile for CPU times.

the figures represent the efficiency of each solver and the values on the right side give information about the robustness of the solvers. This means that the best solver is the highest on the figures.

To compare the CPU times, because of variation of CPU time, each problem is solved five times and then the average of the CPU times is taken into account.

Fig. 1 clearly indicates that NATR outperforms TTR, ATRZ, ATRF and NTR regarding the total number of iterates. In particular, NATR has the most wins in nearly 75% of the tests with the highest efficiency. Meanwhile, in the sense of the ability of completing a run successfully, it is the best among considered algorithms because it grows up faster than others and reaches

1 more rapidly. As illustrated in Fig. 2, NATR implements remarkably better than others where it has most wins in approximately 61% of performed tests concerning the total number of function evaluations. The considered performance profiles in Fig. 3 illustrate that NATR has the best performance, approximately 50% of most wins while reaching 1 faster than other ones. Also, TTR has better results compared with ATRZ, ATRF and NTR in terms of CPU time. As Figs. 1–3 show, the total number of iterations and CPU time are notably decreased due to use of NATR algorithm.

5. Concluding remarks

This paper focuses on introducing and analyzing a trust-region-based algorithm for solving systems of nonlinear equations exploiting an effective adaptive trust-region radius that is produced by nonmonotone strategy. Practical utilization of the trust-region framework has indicated that using adaptive techniques for determining trust-region radius decreases the number of subproblems to be solved. As a result, an effective adaptive trust-region radius is combined in trust-region framework to construct a more promising algorithm for practical usage in nonlinear systems solving. Nevertheless, these modifications in the traditional trust-region procedure are favorably encouraging the global and the quadratic convergence properties of the proposed algorithms which are established without nondegeneracy condition of the exact Jacobian. Preliminary numerical results on a large set of nonlinear systems indicate that the number of iterates and function evaluations are so close to each other that we can immediately conclude that the proposed algorithm has significant profits in computational costs.

Acknowledgements

The authors are grateful for the valuable comments and suggestions of two anonymous referees which considerably improved the presentation of this paper.

References

[1] C.G. Broyden, The convergence of an algorithm for solving sparse nonlinear systems, Math. Comput. 25 (114) (1971) 285–294.
[2] J.E. Dennis, On the convergence of Broyden's method for nonlinear systems of equations, Math. Comput. 25 (115) (1971) 559–567.
[3] A. Griewank, The global convergence of Broyden-like methods with a suitable line search, J. Aust. Math. Soc. Ser. B 28 (1986) 75–92.
[4] L. Grippo, M. Sciandrone, Nonmonotone derivative-free methods for nonlinear equations, Comput. Optim. Appl. 37 (2007) 297–328.
[5] D.H. Li, M. Fukushima, A derivative-free line search and global convergence of Broyden-like method for nonlinear equations, Optim. Methods Softw. 13 (2000) 181–201.
[6] J.M. Martinez, A family of quasi-Newton methods for nonlinear equations with direct secant updates of matrix factorizations, SIAM J. Numer. Anal. 27 (4) (1990) 1034–1049.
[7] G. Fasano, F. Lampariello, M. Sciandrone, A truncated nonmonotone Gauss–Newton method for large-scale nonlinear least-squares problems, Comput. Optim. Appl. 34 (3) (2006) 343–358.
[8] D.H. Li, M. Fukushima, A global and superlinear convergent Gauss–Newton-based BFGS method for symmetric nonlinear equations, SIAM J. Numer. Anal. 37 (1999) 152–172.
[9] L. Lukšan, J. Vlček, Truncated trust-region methods based on preconditioned iterative subalgorithms for large sparse systems of nonlinear equations, J. Optim. Theory Appl. 95 (3) (1997) 637–658.
[10] YU. Nesterov, Modified Gauss–Newton scheme with worst case guarantees for global performance, Optim. Methods Softw. 22 (3) (2007) 469–483.
[11] H. Dan, N. Yamashita, M. Fukushima, Convergence properties of the inexact Levenberg–Marquardt method under local error bound conditions, Optim. Methods Softw. 17 (2002) 605–626.
[12] A. Fischer, P.K. Shukla, M. Wang, On the inexactness level of robust Levenberg–Marquardt methods, Optimization 59 (2) (2010) 273–287.
[13] Z.J. Shi, J.H. Guo, A new trust-region method with adaptive radius, Comput. Optim. Appl. 213 (2008) 509–520.
[14] G.L. Yuan, Z.X. Wei, X.W. Lu, Limited memory BFGS method with backtracking for symmetric nonlinear equations, Math. Comput. Model. 54 (2011) 367–377.
[15] A. Bouaricha, R.B. Schnabel, Tensor methods for large sparse systems of nonlinear equations, Math. Program. 82 (1998) 377–400.
[16] R.B. Schnabel, P.D. Frank, Tensor methods for nonlinear equations, SIAM J. Numer. Anal. 21 (5) (1984) 815–843.
[17] W. La Cruz, M. Raydan, Nonmonotone spectral methods for large-scale nonlinear systems, Optim. Methods Softw. 18 (5) (2003) 583–599.
[18] M. Ahookhosh, H. Esmaeili, M. Kimiaei, An effective trust-region-based approach for symmetric nonlinear systems, Int. J. Comput. Math. 90 (3) (2013) 671–690.
[19] J.Y. Fan, Convergence rate of the trust-region method for nonlinear equations under local error bound condition, Comput. Optim. Appl. 34 (2005) 215–227.
[20] J.Y. Fan, An improved trust-region algorithm for nonlinear equations, Comput. Optim. Appl. 48 (1) (2011) 59–70.
[21] J.Y. Fan, J.Y. Pan, A modified trust-region algorithm for nonlinear equations with new updating rule of trust-region radius, Int. J. Comput. Math. 87 (14) (2010) 3186–3195.
[22] G. Yuan, S. Lu, Z. Wei, A new trust-region method with line search for solving symmetric nonlinear equations, Int. J. Comput. Math. 88 (10) (2011) 2109–2123.
[23] G.L. Yuan, Z.X. Wei, X.W. Lu, A BFGS trust-region method for nonlinear equations, Computing 92 (4) (2011) 317–333.
[24] Y. Yuan, Trust region algorithm for nonlinear equations, Information 1 (1998) 7–21.
[25] J. Zhang, Y. Wang, A new trust-region method for nonlinear equations, Math. Methods Oper. Res. 58 (2003) 283–298.
[26] M. Ahookhosh, K. Amini, A nonmonotone trust-region method with adaptive radius for unconstrained optimization, Comput. Math. Appl. 60 (2010) 411–422.
[27] A.R. Conn, N.I.M. Gould, Ph.L. Toint, Trust-region methods, Society for Industrial and Applied Mathematics SIAM, Philadelphia, 2000.
[28] X.S. Zhang, J.L. Zhang, L.Z. Liao, An adaptive trust-region method and its convergence, Sci. Chin. 45 (2002) 620–631.
[29] M. Ahookhosh, K. Amini, An efficient nonmonotone trust-region method for unconstrained optimization, Numer. Algorithms 59 (4) (2012) 523–540.
[30] M. Ahookhosh, K. Amini, S. Bahrami, A class of nonmonotone Armijo-type line search method for unconstrained optimization, Optimization 61 (4) (2012) 387–404.
[31] M. Ahookhosh, K. Amini, M.R. Peyghami, A nonmonotone trust-region line search method for large-scale unconstrained optimization, Appl. Math. Model. 36 (2012) 478–487.

[32] N.Y. Deng, Y. Xiao, F.J. Zhou, Nonmonotonic trust-region algorithm, J. Optim. Theory Appl. 26 (1993) 259–285.
[33] L. Grippo, F. Lampariello, S. Lucidi, A nonmonotone line search technique for Newton's method, SIAM J. Numer. Anal. 23 (1986) 707–716.
[34] H.C. Zhang, W.W. Hager, A nonmonotone line search technique for unconstrained optimization, SIAM J. Optim. 14 (4) (2004) 1043–1056.
[35] M.J.D. Powell, A new algorithm for unconstrained optimization, in: J.B. Rosen, O.L. Mangassarian, K. Ritter (Eds.), Nonlinear Programming, Academic Press, New York, 2007, pp. 31–66.
[36] W. La Cruz, C. Venezuela, J.M. Martínez, M. Raydan, Spectral residual method without gradient information for solving large-scale nonlinear systems of equations: theory and experiments, Technical, Report RT-04-08, July 2004.
[37] L. Lukšan, J. Vlček, Sparse and partially separable test problems for unconstrained and equality constrained optimization, Techical, Report, No. 767, January 1999.
[38] J.J. Moré, B.S. Garbow, K.E. Hillström, Testing unconstrained optimization software, ACM Trans. Math. Softw. 7 (1981) 17–41.
[39] Ph.L. Toint, Numerical solution of large sets of algebraic nonlinear equations, Math. Comput. 46 (173) (1986) 175–189.
[40] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002) 201–213.