A New Modified TR Algorithm with Adaptive Radius to Solve a Nonlinear Systems of Equations

Hasan H. Dwail\(^1\), Mohammed M. Mahdi\(^2\), H.A. Wasi\(^3\), Karrar H. Hashim\(^4\), Nabiha k. Dreeb\(^5\), Hussein A. Hussein\(^6\) and Mushtak A.K. Shiker\(^7\)\(^*\)

Department of Mathematics, Collage of Education for Pure Sciences, University of Babylon, Hilla, Iraq

mathhasan74@gmail.com
mohmath44@gmail.com
Mathman1972@gmail.com
teach-er89karrar@gmail.com
nabiha250222@gmail.com
hussainalseaidi@gmail.com
mmttmmhh@yahoo.com

Abstract. The trust region method (TRM) is a very important technique to solve both of linear and nonlinear systems of equations. In this work, a new modified algorithm of a TRM with adaptive radius is introduced in purpose of solving systems of nonlinear equations. At each iteration, the new algorithm changes the trust region radius (TRR) automatically to reduce the subproblems resolving number when the current radius is rejected. The global convergence results of the new procedure under some appropriate conditions is established. The numerical effects indicate that the suggested algorithm is interesting and robustness.

Keywords. Trust region method; Global convergence; Nonlinear equations.

1. Introduction

Assume that

\[ F(x) = 0 \ldots \ldots (1.1) \]

is a nonlinear system of equations such that

\[ F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T, \]

where \( F : \mathbb{R}^n \rightarrow \mathbb{R}^m \) is a continuously differentiable function and \( x \in \mathbb{R}^n \). This problem is presents in a lot of scientific fields such as medical sciences, optimal control, engineering and further areas of science \([1,2]\). There are many methods to solve (1.1) such as newton method and quasi newton method. The traditional method to solve (1.1) are line search method (LCM) and TRM \([3]\). At each iteration the LCM depends on searching a new iterate alongside a descent direction, but TRM used a new iterate in a center of a ball at the present iterate. At most, LCM takes the form

\[ x_{k+1} = x_k + \alpha_k d_k, k = 0, 1, 2, \ldots \ldots (1.2), \]

Where \( \alpha_k \) is a trial step and \( d_k \) is a descent direction of \( f(x) \) at \( x_k \).

Briefly, we denote
\[ \nabla f(x_k) \text{ by } \nabla f_k, f(x_k) \text{ by } f_k, \nabla^2 f(x_k) \text{ by } G_k \text{ and } f(x^*) \text{ by } f^*. \]

Solving subproblem (1.2) plays an important role for solving (1.1). The search direction \( d_k \) needs to satisfy
\[ d_k = -g_k \quad \ldots \ldots \quad (1.3). \]
That is, guarantees \( d_k \) is a descent direction of \( f(x) \) at \( x_k \) [4, 5]. Sometimes we use the angle property to prove the global convergence, that is
\[ \cos < -g_k, d_k > = - \frac{g_k d_k}{\|g_k\|\|d_k\|} \geq \sigma \quad \ldots \ldots \quad (1.4), \]
such that \( 1 \geq \sigma \geq 0 \).

There exist two types of lines to find a step size at each iteration, the exact line search which is time-consuming, while inexact line search rules, such as Armijo rule [6], usually used in applied computations. Thus, the Armijo rule is helpful and easy to perform in applied computations.

Armijo rule: Assume \( \lambda > 0 \) is a constant \( \rho \in (0, 1) \) and \( \mu \in (0, 1) \), Take \( \alpha_k \) to be the largest \( \alpha \) in \( \{ \lambda, \lambda \rho, \lambda \rho^2, \ldots \} \) such that
\[ f_k - f(\alpha d_k) \geq -\alpha \mu g_k \cdot d_k \|g_k\| \quad \ldots \ldots \quad (1.5). \]

The important thing in Armijo rule, is how to determine \( \lambda \)?

First, if \( \Lambda \) is larger than the number of function computation, then the solving cones will be increased at each iteration.

Second, if \( \Lambda \) be smaller than the number of iterations, that will increase the consequence and the effective will be decrease.

Some problems in line search method depend on method of choice direction \( d_k \) and step size \( \alpha_k \) in every iteration [1], while, TRM require to solve the following sub problem
\[ \min q_k(p) = f_k + g_k^T d + \frac{1}{2} d^T B_k d, \text{ s.t. } \|d\| \leq \Delta_k \quad \ldots \ldots \quad (1.6) \]
where \( B_k \) is an approximation for \( \nabla^2 f(x_k) \) and \( \Delta_k \) is a TRR.

We can confirm whether or not the new point \( x_{k+1} = x_k + d_k \) is agree.

Given \( \mu \in [0, 1/4] \), then
\[ x_{k+1} = \begin{cases}  
 x_k^* \text{ if } r_k \geq \mu \\
 x_k \text{ otherwise} 
\end{cases} \quad \ldots \ldots \quad (1.8), \]
there are some difficult In TRM, first we need to solve subproblem when the dimension of problem is large, other difficulty is method of choice the TRR \( \Delta_k \) at the \( k^{th} \) iteration [2, 3]. If \( \Delta_k \) is larger than \( \mu \), then the numeral of subproblems which need to be solved will be increased and thus the sum of account will be increase. If \( \Delta_k \) is smaller than \( \mu \), so the activity of TRM will be minimize. To overcome these hardness mentioned above, we will find a new kind of TRM with adaptive radius. The principal way is to identify the subproblems into a simple subspace, the secant way at each iteration choose an adaptive TRR and virtually reduce the subproblems which solving [1, 2, 7].

The remain of this paper is: in section 2 we present a new TRM with adaptive radius. The global convergence analysis In section 3, we summarized some numerical results in section 4 and given conclusion remarks in section 5.

2. TRM with Adaptive Radius;

Suppose that
\[ (M_1) \quad f(x) \quad \text{is a continuously differentiable function has lower bound} \]
\[ L_0 = \{ x \in \mathbb{R}^n: f(x) \leq f(x_0) \}, \]
\[ (M_2) \quad \text{The gradient } \nabla f(x) = g(x) \quad \text{is uniformly continuous on an open convex set } B \quad \text{that contains the level set } L_0. \]

In TRM we must resolve the subproblem (1.6), \( d_k^* \text{ is the solution of (1.6)} \) where the minimizer of \( q_k(d) \) in the current iterate with the radius \( \Delta_k \). Thus, we should note that the important first step is who
to choose the TRR. The TRR $\Delta_k$ at every iteration depended on the relationship between the objective function $F$ at the former iterate and the model $q_k$. Then the step $d_k$ is got by reducing model $q_k$. So, if $r_k$ is close to 1, it is safe extend, if $r_k > 0$, but not close to 1, then the TRR do not alter. if $r_k \leq 0$, we shrink the trust region, otherwise, if $r_k < 0$, so the step must be rejected \cite{8, 9}.

**Algorithm 2.1 (Trust Region).**

Given $\Delta' > 0$, $\Delta_0 \in (0, \Delta')$, and $\mu \in [0, 1/4]$; For $k = 0, 1, 2,\ldots$

1. Obtain $d_k$ by solving (1.6);
2. Evaluate $r_k$ from (1.7);
3. If $r_k < 1/4$, then $\Delta(k + 1) = 1/4\|d_k\|$
4. If $r_k > 3/4$ and $\|d_k\| = \Delta_k$, then $\Delta_{k+1} = \min(2\Delta_k, \Delta')$
5. Else $\Delta_{k+1} = \Delta_k$;
6. If $r_k > \mu$, then $x_{k+1} = x_k + d_k$
7. Else $x_{k+1} = x_k$;
8. End

Usually, we require not to answer (1.6) closely, we could find $d_k$ that satisfies

$$q_k(0) - q_k(d_k) \geq c_1\|g_k\| \min [\Delta_k^{\|g_k\|}/\|B_k\|], \quad (2.1)$$

$$\|d_k\| \leq \gamma \Delta_k, \ldots \ldots \ldots (2.2)$$

for $\gamma \geq 1$ and $c_1 \in (0, 1]$. Really, the exact answer $d_k^*$ of (1.7) holds (2.1), (2.2) \cite{10, 11}.

**Lemma 2.2 \cite{1}**

Let $S$ be constant, $\|B_k\| \leq S$ and $\mu = 0$ in Algorithm 2.1, $f$ is continuously differentiable and bounded below on the level set

$$L_0 = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\},$$

and very approximate answers of (1.6) holds inequalities (2.1), (2.2) for constants $c_1, \gamma > 0$, then

$$\lim_{k \to \infty} \|g_k\| = 0 \ldots \ldots (2.3)$$

**Lemma 2.3 \cite{1}**

Let $S > 0$ be constant, $\|B_k\| \leq S$ and $\mu \in (0, 1/4)$ in Algorithm 2.1, $f$ is Lipschitz continuously differentiable and bounded below on the level set $L_0$ and very approximate answers of (1.6) holds inequalities (2.1), (2.2) for constants $c_1, \gamma > 0$, then

$$\lim_{k \to \infty} \inf \|g_k\| = 0 \ldots \ldots (2.4)$$

As it’s known, the TRR in Algorithm 2.1 is controls agreeing to ratio. The hardness method to choice $\Delta'$ in Algorithm (2.1). If $\Delta'$ is large then the number of subproblems increases then increases computational costs. if $\Delta'$ is small then the number of iteration will increases, so the efficiency will reduce.

Therefore, we should select a suitable radius at each iteration, so, we can develop this method with adaptive radius.

**Algorithm 2.2 (Adaptive Trust Region Radius)**

1. For $k = 0, 1, 2,\ldots$ set $k := 0$.
2. Select $d_k$ to satisfy $d_k = -g_k(B_k = B_k + i_k l$ with $i_k$ being the smallest nonnegative integer $i$ such that
3. $d_k^2 B_k = d_k^2 B_k + i \|d_k\| \leq 2 > 0$.
4. Compute $r_k$ from (1.7)
5. $x_{k+1} = x_k + d_k(\Delta_k)$, such that $k$ is the biggest $\Delta$ in $\{s_k, \lambda s_k, \lambda^2 s_k, \ldots\}$ such that
6. $f(x_k) + d(\Delta) \leq f_k + \delta u g_k^\top$ in which $d_k(\Delta)$ is a solution of
7. $\min q_k(p) = f_k + g_k^\top d + \frac{1}{2} d^\top B d$, such that $\|d\| \leq \Delta \ldots \ldots (2.5)$
8. Modify $B_k$ as $B_{k+1}$ by using BFGS formula.
9. Set $k := k + 1$ and go to Step 1.
3. Convergence Analysis

To structure the global convergence for the trust region algorithm (2.2), we must prove the following results.

Lemma 3.1: [1]
Suppose that \((M_1)\) and \((M_2)\) verify and \(\exists S > 0\) such that \(\|B_k\| \leq S\), \(\forall k\). So Algorithm 2.2 is well-defined.
Proof:
Since \((M_1)\) and \((M_2)\) are holds, we have
\[
(f(x_k + d(\Delta))) \leq f_k + \delta \mu g_k
\]
Assume that \(\gamma = \delta \mu\) and using mean value theorem,
\[
f(x_k + d(\Delta))/f_k + \delta \mu g_k = \frac{-g'_k + \frac{1}{2}g_k - g(x_k + d)^tt}{-g'_k - \frac{1}{2}d'B_kd}
\]
for \(\Delta \in (0, \Delta]\), where \(d(\Delta)\) is an answer of (2.5). That is prove that Algorithm (2.2) is well-defined.
Lemma 3.2
\[
\forall k = 0, 1, 2, \ldots \text{ then } q_k(0) - q_k(d_k) \geq -\frac{1}{2} \Delta_k g_k d_k, \text{ such that } d_k \text{ is a answer to (2.5) with } \Delta_k = -\alpha \rho^m s_k, \ldots \ldots \ldots (3.2), \text{ such that } m_k \in \mathbb{Z} +.
\]
Proof:
It’s clear that \(d'_k = -\alpha \rho^m g_k d_k / d'B_kd_k\) is a feasible solution of (2.5).
From (1.3) and (3.2) we have
\[
q_k(0) - q_k(d_k) \geq q_k(0) - q_k(d'_k) = -g_k d'_k - \frac{1}{2}d'_k B_k d'_k
\]
\[
\geq \alpha \rho^m \frac{g_k d_k}{d'B_kd_k} [g_k d_k - \frac{1}{2} \alpha \rho^m g_k d_k]
\]
\[
\geq \alpha \rho^m \frac{g_k d_k}{d'B_kd_k} [g_k d_k - \alpha \frac{1}{2} g_k d_k]
\]
\[
= \alpha \frac{\rho^m (g_k d_k)^2}{2} \frac{d_k d'_k d_k}{d_k d_k} \geq -\frac{1}{2} \alpha \Delta_k \frac{g_k d_k}{\|d_k\|}
\]

Theorem 3.1
Assume that \(M_1, M_2\) verify, \(d_k\) satisfies (1.3), there is \(S > 0\) s.t \(\|B_k\| \leq S\) for all \(k\). Algorithm (2.2) produce infinite sequence \(\{x_k\}\) so
\[
\lim_{k \to \infty} -\frac{g_k d_k}{\|d_k\|} = 0 \ldots \ldots \ldots (3.3).
\]
Proof:
By contradiction, suppose that there is an infinite \( K \subseteq \{1,2,3, \ldots \} \) and \( \varepsilon > 0 \), such that
\[
- \frac{g_k d_k}{\|d_k\|} \geq \varepsilon, k \in K \quad \ldots \quad (3.4)
\]
At each iteration \( \Delta_k = - \alpha \rho^{m_k} \frac{g_k d_k}{d_k B_k d_k} \|d_k\| \), with positive integer \( m_k \),
\[
f_k - f(x_k + d_k) \geq \alpha \mu
\]
from lemma (3.1) and equation (3.4) we get
\[
f_k - f_{k-1} \geq \alpha \mu [q_k(0) - q_k(p_k)] \geq -0.5 \alpha \mu \Delta_k \frac{g_k d_k}{\|d_k\|} \geq 0.5 \alpha \mu \varepsilon \Delta_k, k \in K. \]

From \( M_1 \) we get
\[
\lim_{k \to \infty} \Delta_k = 0 \quad \ldots \quad (3.5)
\]

because \( \|B_k\| \leq S \), we get
\[
d_k^T B_k d_k \leq (2S + 1) \|d_k\|^2 \quad \ldots \quad (3.6)
\]
from (3.4), (3.6) we get
\[
\Delta_k = - \alpha \rho^{m_k} \frac{g_k d_k}{d_k B_k d_k} \|d_k\| \geq - \alpha \rho^{m_k} \frac{g_k d_k}{(2S + 1) \|d_k\|} \geq \frac{\rho^{m_k}}{2S + 1} \varepsilon, k \in K
\]
From (3.5) there exists \( k' \) such that \( m_k > 0, k \in K, k \geq k' \).

By algorithm (2.2), \( x_k + d_k' \) (\( k \in K, K \geq k' \)) not to be agree if \( d_k' \) solution of
\[
\min q_k(p) = f_k + g_k^T d + \frac{1}{2} d^T B_k d
\]
such that \( \|d\| \leq \Delta_k / \rho \), that is
\[
f_k - f(x_k + d_k') < \alpha \mu [q_k(0) - q_k(d_k')] \quad \ldots \quad (3.7)
\]

By using \( M_2 \), (3.4), (3.5) and lemma (3.1) we get
\[
\left| \frac{f_k - f(x_k + d_k') - 1}{q_k(0) - q_k(d_k')} \right| \leq \left| \frac{f_k - f(x_k + d_k') + g_k d_k' + 0.5 d_k' B_k d_k'}{q_k(0) - q_k(d_k')} \right| - 0.5(\Delta k / \rho) g_k d_k \|d_k\|
\]
\[
\leq o(\|d_k\|^2) + 0.5 S \|d_k'\|^2
\]
\[
-0.5 \Delta_k / \rho \frac{g_k d_k}{\|d_k\|} \|d_k\|
\]
\[
\leq o(\|d_k'\|^2) + 0.5 S \|d_k'\|^2
\]
\[
0.5 \varepsilon \Delta_k / \rho
\]
\[
\leq \frac{a_k^{\frac{1}{2}}}{\rho^2} \to 0, (k \in K, K \to \infty)
\]
Hence \( k'' \geq k' \) such that
\[
\frac{f_k - f(x_k + d_k')}{q_k(0) - q_k(d_k')} \geq \alpha \mu
\]
That is contradiction with (3.7), then there is no infinite subset \( K \) such that theorem 3.1 holds.

4. Numerical Results
We will report the numerical tests to compare the results of a new method \( TTRHH \) with three algorithms.

\( TTRzjz \) : This algorithm is produced by Zhen-Jun Shi. Jinhun Gao [1], it used the steepest descent method.

\( TTRhm \) : This algorithm is produced by Hamid Esmaeili and Morteza Kimiae [5], it used a newton method.
TTRzz: This algorithm produced by Zhen-Jun and Shi. Zhiwei Xu [3] it used sufficient descent condition.

This experiment run on a computer with CPU – time 1.65 GHZ and 4.00 GB Ram, every algorithms codes are written in MATLAB R2014a, this study checks the documents for problems in algorithms convergence to identical points.

We take \( \mu_1 = 0.1 \); \( \mu_2 = 0.9 \); c = 0.3; p = 0.3; epsilon = \( 10^{-5} \) and the number of total of iteration exceeds 20000, the problems are:

- **P1**: \( f = 100 \times [(x_2 - (x_1^2))^2 + (1 - x_1)^2]. \)
- **P2**: \( f = [(x_2 - (x_1^2))^2 + (1 - x_1)^2]. \)
- **P3**: \( f = (x_2 - (x_1^2))^2 + (1 - x_1)^2. \)
- **P4**: \( f = [(x_2 - (x_1^2))^{1/2} + (1 - x_1)^{1/2}. \)
- **P5**: \( f = (x_2 - x_1^2) + (1 - x_1)^2. \)
- **P6**: \( f = (x_2 - x_1^2) + (1 - x_1). \)
- **P7**: \( f = (x_2 - x_1^2) + (1 - x_1)^2. \)

The numerical results of all algorithms are listed in tables 4.1, 4.2 and 4.3, where table 4.1 contains the number of iterations, table 4.2 contains the number of functions evaluations and table 4.3 contains the CPU times.

**Table 4.1. Number of Iterations**

|       | TTRHH | TTRhm | TTRzjg | TTRzz |
|-------|-------|-------|--------|--------|
| \( P_1 \) | 2     | 20001 | 10     | 20001 |
| \( P_2 \) | 2     | 4     | 4      | 4      |
| \( P_3 \) | 2     | 4     | 4      | 4      |
| \( P_4 \) | 20001 | 20001 | 20001  | 20001  |
| \( P_5 \) | 2     | 5     | 5      | 5      |
| \( P_6 \) | 4     | 7     | 10     | 7      |
| \( P_7 \) | 20001 | 20001 | 20001  | 20001  |

**Table 4.2. Number of Functions Evaluations**

|       | TTRHH | TTRhm | TTRzjg | TTRzz |
|-------|-------|-------|--------|--------|
| \( P_1 \) | 3     | 20002 | 11     | 20002 |
| \( P_2 \) | 3     | 5     | 5      | 5      |
| \( P_3 \) | 3     | 5     | 5      | 5      |
| \( P_4 \) | 20002 | 20002 | 20002  | 20002  |
| \( P_5 \) | 3     | 6     | 6      | 6      |
| \( P_6 \) | 5     | 8     | 11     | 8      |
| \( P_7 \) | 20002 | 20002 | 20002  | 20002  |

**Table 4.3. CPU times**

|       | TTRHH | TTRhm | TTRzjg | TTRzz |
|-------|-------|-------|--------|--------|
| \( P_1 \) | 0.0033 | 24.468 | 0.0064 | 23.493 |
| \( P_2 \) | 0.0015 | 0.0048 | 0.0026 | 0.0038 |
| \( P_3 \) | 0.0015 | 0.0027 | 0.0025 | 0.0050 |
| \( P_4 \) | 23.030 | 23.197 | 24.021 | 27.354 |
| \( P_5 \) | 0.0176 | 0.0032 | 0.0035 | 0.0051 |
| \( P_6 \) | 0.0027 | 0.0045 | 0.0070 | 0.0058 |
| \( P_7 \) | 21.636 | 22.938 | 23.671 | 24.616 |

From the tables above we can see that the new algorithm (TTRHH) is the best one among all the tasted algorithms because it needs number of iterations and functions evaluation less than the other
algorithms. And the CPU times that the new approaches needs to solve the problems is less than the CPU times that the other approaches need. So, we can say that the algorithm (TTRHH) is more accurate and efficient than other algorithms and it is so promising to solve the nonlinear systems of equations.

5. Conclusion:
The trust region technique is one of the most important approaches to solve the nonlinear systems of equations. In this technique, the important and difficult step is how to choose the initial radius of this region? Because it is known that when the radius increases, $d_k$ hits the boundary of the trust-region. If the step is strictly inside the region, it means that the current $\Delta_k$ is appropriate and it is not changed in the next iteration. That is, when the radius is very large, the number of the subproblems increases and so computational cost to solve the problem increases too. And when the radius is very small, the total number of iterations increases and the efficiency of the algorithm will be possibly reduces.

In this work, these disadvantages of the trust region method will overcome because the trust region radius can be adjusted automatically by the proposed algorithm which satisfies the accuracy and the efficiency of the proposed algorithm.

References
[1] Z. Jun and S. J. Guo. “A new trust region method with adaptive radius”, Comput Optim Appl (2008) 41: 225 - 242.
[2] Z. Sheng, G. yuan and Z. Cui. “A new adaptive trust region algorithms for optimization problems”, Acta Mathematica Scientia, (2018) 38B (2) :479 - 496.
[3] Z. J. Shi and Z. Xu. “The convergence of subspace trust region methods”, Journal of Computational and Applied Mathematics (2009) 23 : 365 - 377.
[4] M. Ahookhosh, K. Amini, M.R. Peyghami. “A nonmonotone trust-region line search method for large-scale unconstrained optimization”, Appl. Math Modelling (2012) 36 : 478 – 487.
[5] H. Esmaeili and M. Kimiaei. “A new adaptive trust-region method for system of nonlinear equations”, Applied Mathematical Modeling (2014) 38 : 3003 – 3015.
[6] J. L. Zhan and Y. Wang. “A new trust region method for nonlinear equations”, Math Meth Oper Res (2003) 58 : 283 – 298.
[7] W. Sun and Y. X. Yuan. “Optimizations Theory and Methods Nonlinear Programming”, Springer Optimization And its Application (2017) 1 : 71 - 74.
[8] K. Amini, M. A. Shiker and M. Kimiaei. "A line search trust-region algorithm with non-monotone adaptive radius for a system of nonlinear equations", 4 OR- Journal of operation (2016) 14 : 133 - 152.
[9] M. A. Shiker and Z. Sahib. "A modified Technique for Solving Unconstrained Optimization", Journal of Engineering and Applied Sciences (2018) 13(22) : 9667 - 9671.
[10] W. Sun and Y. X. Yuan. "Optimizations Theory and Methods Nonlinear programming ", Springer Optimization And its Application (2006) 1 : 303 - 307.
[11] M. A. Shiker and K. Amini, "A New Projection- Based Algorithm for Solving a Large-scale Nonlinear System of Monotone Equations", Croatian Operational Research Review (2018) 9 (1) : 63 - 73.