mathematics

MDPI

Article

An Improved Non-Monotonic Adaptive Trust Region Algorithm for Unconstrained Optimization

Mingming Xu [1], Quanxin Zhu [2] and Hongying Xiao [1,*]

[1] School of Mathematics and Physics, Yibin University, Yibin 644000, China; 2021070027@yibinu.edu.cn
[2] School of Mathematics and Statistics, Hunan Normal University, Changsha 410081, China; zqx22@hunnu.edu.cn
* Correspondence: 2020080001@yibinu.edu.cn

Abstract: The trust region method is an effective method for solving unconstrained optimization problems. Incorrectly updating the rules of the trust region radius will increase the number of iterations and affect the calculation efficiency. In order to obtain an effective radius for the trust region, an adaptive radius updating criterion is proposed based on the gradient of the current iteration point and the eigenvalue of the Hessian matrix which avoids calculating the inverse of the Hessian matrix during radius updating. This approach reduces the computation time and enhances the algorithm's performance. On this basis, we apply adaptive radius and non-monotonic techniques to the trust region algorithm and propose an improved non-monotonic adaptive trust region algorithm. Under proper assumptions, the convergence of the algorithm is analyzed. Numerical experiments confirm that the suggested algorithm is effective.

Keywords: non-monotonic technology; adaptive radius; trust region algorithm

MSC: 49M37; 65K05; 90C30

check for updates

Citation: Xu, M.; Zhu, Q.; Xiao, H. An Improved Non-Monotonic Adaptive Trust Region Algorithm for Unconstrained Optimization. *Mathematics* **2024**, *12*, 3398. https://doi.org/10.3390/math12213398

Academic Editor: Jüri Majak

Received: 9 October 2024
Revised: 24 October 2024
Accepted: 28 October 2024
Published: 30 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Consider the following unconstrained optimization problem:

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

where $h : R^n \to R$ is twice continuously differentiable. There are two types of optimization problems: confined optimization and unconstrained optimization. Constrained optimization problems are typically addressed by being converted into unconstrained optimization problems. Unconstrained optimization problems can be solved using various techniques, including the conjugate gradient approach [1,2], the trust region method [3,4], and the Newton method [5,6]. The conjugate gradient method requires that the coefficient matrix is not only symmetric but also positive definite, which may not be applicable in the case of matrices with non-positive definite coefficients [7]. The Newton method needs to calculate the inverse of the Hessian matrix, and selection of the initial points is difficult in practice [8]. The trust region algorithm has strong convergence and robustness, and it has become one of the effective methods for solving unconstrained optimization problems [9,10].

Many practical application problems which arise in applied mathematics, economics, and engineering can be translated into Equation (1) to be solved [11]. As one of the effective algorithms for solving unconstrained optimization problems, the trust region has many applications in real life. For example, it can be extended to deal with constrained optimization problems, variational inequality problems, and nonlinear complementary problems [12]. Of course, the trust region algorithm can also be extended to reinforcement learning [13] and support vector machines [14] in machine learning, the electrical impedance tomography problem [15], and solving the inverse problem related to seismic spectrum analysis of Pn waves [16] as well as combined with the epitaxial implicit method [17]. Therefore, it is

necessary to propose a trust region algorithm which can effectively solve unconstrained optimization problems.

The underlying principle of trust region (TR) methods is that a trial step t_k is obtained at each iterate point x_k by solving the following subproblem:

$$\begin{aligned} \min \varpi_k(t) =& g_k^T t + \frac{1}{2} t^T B_k t, \\ s.t. & \|t\| \leq \Delta_k, \end{aligned} \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm, $g_k = \nabla h(x_k)$, $B_k \approx \nabla^2 h(x_k)$, and Δ_k is the TR radius [18]. Next, the TR ratio r_k is used to compute the agreement between the predicted and actual reductions. It has the definition

$$r_k = \frac{Aret_k}{Pret_k} = \frac{h(x_k) - h(x_k + t_k)}{\varpi_k(0) - \varpi_k(t_k)},$$

where the predicted reduction $Pret_k$ and the actual reduction $Aret_k$ are given by

$$Aret_k = h(x_k) - h(x_k + t_k),$$

$$Pret_k = \varpi_k(0) - \varpi_k(t_k).$$

The iteration form of the TR is described below:

$$x_{k+1} = \begin{cases} x_k + t_k, & \text{if } r_k \geq \mu, \\ x_k, & \text{otherwise.} \end{cases}$$

If $r_k \geq \mu$ for a specified $0 < \mu < 1$, then the trial step t_k is approved, and $x_{k+1} = x_k + t_k$. The TR radius is adjusted correctly in this instance. In contrast, the t_k is rejected, and the current point is held constant for the following iteration if $r_k < \mu$. The TR radius is suitably decreased in this instance. This procedure is continued until the convergence requirements hold.

In the traditional TR algorithm, it is essential to select the appropriate initial value and update criteria for the TR radius. If the initial radius or radius updating criteria are not selected correctly, then the number of iterations of the trust region algorithm will be increased. Based on this, Sartenaer [19] proposed a criterion that can automatically determine the initial radius which uses gradient information (i.e., $\triangle_0 = \|g_0\|$). Later, Lin et al. [20] proposed a better criterion for selecting the initial radius, which is set to be $\triangle_0 = \gamma\|g_0\|$, where $\gamma > 0$ is an adjustable constant.

It can be noted that when the sequence $\{x_k\}$ generated by the traditional TR algorithm converges to the approximate best point x^*, the TR ratio r_k may converge to one. Therefore, according to the iterative steps of the algorithm, once k is big enough , $\triangle_k$ may be larger than the normal number. Moreover, the information of g_k and B_k generated at the current iteration point x_k is not used to revise the radius $\triangle_k$, which greatly increases the number of subproblems (Equation (2)) to be solved, thus reducing the calculation rate of the algorithm. To avoid the above problems, Zhang [21] proposed some variations of the adaptive TR method based on the following updated formula:

$$\triangle_k = \frac{q^\ell \|g_k\|}{b_k},$$

in which $b_k = \min\{\|B_k\|, 1\}$, $0 < q < 1$ is a constant and $\ell \in \mathbf{N}$. Subsequently, Zhang et al. [22] proposed an adaptive radius formula containing g_k and B_k as follows:

$$\triangle_k = q^\ell \|g_k\| \|\hat{B_k}^{-1}\|,$$

with ℓ and q as defined above and $\hat{B}_k = B_k + iI$ for some nonnegative integer i and the Hessian approximation B_k as a positive definite matrix. To avoid calculating ${\hat{B}_k}^{-1}$ at each iteration point x_k, Shi and Wang [23] proposed

$$\triangle_k = q^{\ell} \frac{\|g_k\|^3}{g_k^T \hat{B}_k g_k},$$

with ℓ, q, and $\hat{B}_k$ as defined above. To avoid counting $\hat{B}_k$, Sheng et al. [24] proposed a variant:

$$\triangle_k = q^{\ell} \frac{\|t_{k-1}\|}{\|y_{k-1}\|} \|g_k\|,$$

where $y_{k-1} = g_k - g_{k-1}$, $0 < q < 1$, and $\ell \in \mathbf{N}$. Recently, Wang et al. [25] employed the new radius updating criterion $\triangle_k = v_k \|g_k\|^{\gamma}$, where $\gamma \in (0, 1)$ is a constant and the updating of v_k depends on the value of the ratio r_k. Building an adaptive TR radius which can speed up the solving of subproblems is vital since the TR algorithm heavily relies on the choice for the TR radius.

Each iteration in the monotone trust region method requires objective function h_k to have a certain reduction. However, it can be seen from some numerical experimental results that the monotonicity of forced h_k may significantly reduce the convergence rate, especially for the objective functions with highly eccentric level curves [26–28]. Therefore, allowing function values to increase to some extent while maintaining global convergence is advantageous. Non-monotonic methods are characterized by the fact that they do not enforce strict monotonicity of the value of the objective function over successive iterations. It has been shown that using non-monotonic techniques can improve the likelihood of finding global optimal values and the rate of convergence of the algorithm [26,29]. Therefore, many researchers combined non-monotonic techniques with trust regions to form a new method for solving optimization problems [28,30–33]. Chamberlain et al. [34] proposed watchdog technology, which belongs to a non-monotonic line search technology. Next, Grippo et al. [26] extended Newton's method with another non-monotonic line search methodology and used it to solve unconstrained optimization problems. The idea of this non-monotonic line search technique is that for a given $0 < \zeta < 1$, one should select the step length α_k such that the following conditions are true:

$$h(x_k + \alpha_k t_k) \leq h_{l(k)} + \zeta \alpha_k g_k^T t_k,$$

where non-monotonic technology $h_{l(k)}$ is defined as

$$h_{l(k)} = h(x_{l(k)}) = \max_{0 \leq j \leq p(k)} \{h_{k-j}\}, \tag{3}$$

where $0 \leq p(k) \leq \min\{p(k-1) + 1, M\}$ and M are positive integers. However, some numerical experiment results indicate that there are still certain issues with Grippo's non-monotonic method [35]. For instance, the current function value cannot be fully used. Moreover, the traditional non-monotonic technique is highly dependent on the selection of parameter M, and thus the numerical results of the algorithm may vary greatly with different selections of M values. To address these shortcomings of non-monotonic techniques, Zhang and Hager [29] proposed another non-monotonic technique as follows:

$$C_k = \begin{cases} h_0, & k = 0, \\ \frac{v_{k-1} Q_{k-1} C_{k-1} + h_k}{Q_k}, & k \geq 1, \end{cases}$$

and

$$Q_k = \begin{cases} 1, & k = 0, \\ v_{k-1} Q_{k-1} + 1, & k \geq 1, \end{cases}$$

where $\nu_{k-1} \in [\nu_{\min}, \nu_{\max}]$, $\nu_{\min} \in [0, 1)$, and $\nu_{\max} \in [\nu_{\min}, 1)$. Because the non-monotonic technique C_k needs to update ν_{k-1} and Q_k in each iteration, which affects the computation rate, Ahookhosh and Amini [35] combined h_k and $h_{l(k)}$ in a convex manner to form another efficient non-monotonic technique R_k:

$$R_k = \vartheta_k h_{l(k)} + (1 - \vartheta_k) h_k, \tag{4}$$

where $\vartheta_{k-1} \in [\vartheta_{\min}, \vartheta_{\max}]$, $\vartheta_{\min} \in [0, 1)$. Compared with $h_{l(k)}$, R_k has the following advantages. It can not only make full use of the current objective function value but also retain the current optimal operation value to a certain extent. Finally, better convergence can be achieved by selecting different parameters. Since the non-monotonic algorithm does not require the objective function to be strictly monotonic during continuous iteration, application of the non-monotonic technique to the trust region algorithm can increase both the likelihood of finding the global optimal solution and the algorithm's computation rate [30,36,37].

Because the convergence rate of the trust region algorithm depends on the renewal of the trust region radius, this paper aims to construct a new formula for the trust region radius. Combined with the non-monotonic technique, a new non-monotonic adaptive trust region algorithm is proposed to solve unconstrained optimization problems.

In this paper, in order to fully utilize the gradient and Hessian matrix of the current iteration points, we propose an improved adaptive radius updating criterion. In the improved adaptive radius-updating formula, the upper bound of the eigenvalue of matrix B_k is solved instead of the inverse of matrix B_k.

The rest of this article is organized as follows. In Section 2, we present a new adaptive trust region radius and propose a non-monotonic adaptive trust region algorithm. In Section 3, we demonstrate this algorithm's global convergence. In Section 4, numerical experiments are carried out, and the new method's effectiveness is demonstrated by the numerical results. Finally, we conclude the paper in Section 5 with a few closing thoughts.

2. The New Algorithm

In this section, an adaptive radius-updating formula is proposed. The improved adaptive radius updating criterion is described in detail below.

At the iterate point x_k, we set λ_{ki} to be the ith eigenvalue of the Hessian matrix B_k. According to the Geršgorin circle theory [12,38], we obtain

$$\lambda_{ki} \leq \sum_{j=1}^{n} |b_{ij}| \triangleq \bar{\lambda}_{ki}, \quad (i \in \mathbf{N}^*) \tag{5}$$

where $\bar{\lambda}_{ki}$ can be thought of as an upper bound on the ith eigenvalue of B_k and the element b_{ij} is the one which is placed at the ith row and jth column in matrix B_k.

Let

$$\lambda_k = \sum_{i=1}^{n} \omega_i \bar{\lambda}_{ki}^{-1}, \tag{6}$$

where $\omega_i > 0$ and $\sum_{i=1}^{n} \omega_i = 1$.

Based on this, the new adaptive trust region radius proposed in this paper is as follows:

$$\Delta_{k+1} = c_{k+1} \|g_{k+1}\| \lambda_{k+1}, \tag{7}$$

where

$$c_{k+1} = \begin{cases} t_1 c_k, & \tilde{r}_k < \mu_1, \\ c_k, & \mu_1 \leq \tilde{r}_k < \mu_2, \\ t_2 c_k, & \tilde{r}_k \geq \mu_2, \end{cases} \tag{8}$$

and $0 < t_1 < 1 < t_2$. Obviously, we adjust the size of c_{k+1} using the trust region ratio $\tilde{r}_k$, which can better use the information of the function to adjust the radius of the trust region so as to effectively decrease the number of solutions for Equation (2).

The adaptive radius updating criterion proposed in this paper not only makes full use of the first- and second-order information of the objective function but also avoids solving the inverse matrix B_k. Based on the non-monotonic technique and the new adaptive trust region radius (Equation (7)), we propose an improved non-monotonic adaptive trust region algorithm (NATR). In our algorithm, the non-monotone technique is defined by R_k, where R_k is given by Equation (4) and the ratio of the trust region $\tilde{r}_k$ is determined by

$$\tilde{r}_k = \frac{R_k - h(x_k + t_k)}{\omega_k(0) - \omega_k(t_k)}, \tag{9}$$

where t_k is the trial step to be calculated by Equation (2).

The procedure of NATR is described by Algorithm 1 .

Algorithm 1 An improved non-monotonic adaptive trust region algorithm (NATR).

Step 0. (Initialization) Start with $x_0 \in R^n$ and the symmetric matrix $B_0 \in R^{n \times n}$. The constants $0 \leq \vartheta_{\min} \leq \vartheta_{\max} \leq 1$, $\vartheta_k \in [\vartheta_{\min}, \vartheta_{\max}]$, $0 < t_1 < 1 < t_2$, $0 < \mu_1 < \mu_2 < 1$, $\varepsilon > 0$, $\theta > 0$, $\Delta_{\max} > 0$, $c_0 > 0, \omega > 0$, and $\Delta_0 > 0$ are also given. Set $k = 0$.
Step 1. If $\|g_k\| \leq \varepsilon$, then stop.
Step 2. Solve the subproblems of Equation (2) to find the trial step t_k.
Step 3. Compute R_k and $\tilde{r}_k$. If $\tilde{r}_k < \mu_1$, then go to Step 2. If $\tilde{r}_k \geq \mu_1$, then set $x_{k+1} = x_k + t_k$, and go to Step 4.
Step 4. Compute λ_{k+1}. If $\lambda_{k+1} \leq \omega$ or $\lambda_{k+1} \geq \frac{1}{\omega}$, then set $\lambda_{k+1} = \theta$; Update $\Delta = c_{k+1}\|g_{k+1}\|\lambda_{k+1}$, where c_{k+1} is given by Equation (8), and set $\Delta_{k+1} = \min\{\Delta, \Delta_{\max}\}$.
Step 5. Update B_k by using a modified quasi-Newton formula. Set $k = k + 1$, and go to Step 1.

From Step 4 of the NATR algorithm, it can be seen that sequence $\{\lambda_k\}$ is bounded; in other words, we have

$$0 < \min\{\omega, \theta\} \leq \lambda_k \leq \max\{\frac{1}{\omega}, \theta\} \tag{10}$$

for any $k \in \mathbf{N}$.

Remark 1. *It can be noted that in this paper, the non-monotone adaptive trust region algorithm uses the gradient and B_k matrix eigenvalues when the radius Δ_k is updated. Compared with [30,39] and [22], the radius in this paper avoids solving B_k^{-1} and makes full use of the ratio of the trust region $\tilde{r}_k$ in each iteration to automatically adjust the radius Δ_k, which can make better use of the information of the current iteration point of the objective function.*

Throughout this paper, we consider the following assumptions in order to analyze the convergence of the new algorithm:

H1: The level set $\tilde{L}(x_0) = \{x \in R^n | h(x) \leq h(x_0)\}$ is a bounded closed set.

H2: There exist constants $M_B > 0$, $M_g > 0$, and $M_G > 0$ such that $\|B_k\| \leq M_B$, $\|g_k\| \leq M_g$, and $\|\nabla^2 h(x)\| \leq M_G$.

H3: Suppose that there is a constant $\bar{c} > 0$ such that $\forall k \in \mathbf{N}$, and $c_k \leq \bar{c}$ is true.

Remark 2. *Similar to [10], for a solution t_k to the subproblem in Equation (2), we have*

$$Pret_k = \omega_k(0) - \omega_k(t_k) \geq \delta \|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\}, \ \delta \in (0,1). \tag{11}$$

3. Convergence Analysis

In order to analyze the convergence properties of Algorithm 1, we also need to present the following lemmas:

Lemma 1. *Based on H1 and H2, suppose that t_k is the solution to Equation (2). Then, we have*

$$h(x_k) - h(x_k + t_k) - [\omega_k(0) - \omega_k(t_k)] \leq o(\|t_k\|^2).$$

Proof. See [31] for the proof of Lemma 1. □

Lemma 2. *Assume that the NATR algorithm produced the sequence $\{x_k\}$. Then, the sequence $\{h_{l(k)}\}$ is a decreasing sequence.*

Proof. See the proof of Lemma 4 in [35]. □

Lemma 3. *Suppose that the sequence $\{x_k\}$ is generated by the NATR algorithm. Then, we have*

$$h_k \leq R_k, \tag{12}$$

and

$$x_k \in \tilde{L}(x_0),$$

for each $k \in \mathbf{N}$.

Proof. From Equaitons (3) and (4), we have

$$h_k = \vartheta_k h_k + (1 - \vartheta_k) h_k \leq \vartheta_k h_{l(k)} + (1 - \vartheta_k) h_k = R_k.$$

Hence, Equation (12) holds. Now let us prove by induction that $x_k \in \tilde{L}(x_0)$ for each $k \in \mathbf{N}$. Obviously, the result holds for $k = 0$. We demonstrate that $x_{k+1} \in \tilde{L}(x_0)$, assuming that $x_k \in \tilde{L}(x_0)$. Based on Lemma 2, we have

$$h_{k+1} \leq h_{l(k+1)} \leq h_{l(k)} \leq h_0,$$

i.e., $x_{k+1} \in \tilde{L}(x_0)$. This completes the proof. □

Lemma 4. *Assume that the NATR algorithm produced the sequence $\{x_k\}$ and $\|g_k\| \geq \varepsilon > 0$. Then, $\forall k \in \mathrm{N}, \exists \varrho \in \mathbf{N}$ such that $x_{k+\varrho+1}$ is a successful iteration (i.e., $\tilde{r}_{k+\varrho} \geq \mu_1$).*

Proof. Assume that there exists an integer constant k such that the point $x_{k+\varrho+1}$ is not a successful point for any ϱ. Hence, we have that $\tilde{r}_{k+\varrho+1} < \mu_1$ for any constant $\varrho = 0, 1, 2, \cdots$. From H2 and Remark 1, we obtain

$$\begin{aligned} \omega_{k+\varrho}(0) - \omega_{k+\varrho}(d_{k+\varrho}) &\geq \delta \|g_{k+\varrho}\| \min\{\Delta_{k+\varrho}, \frac{\|g_{k+\varrho}\|}{\|B_{k+\varrho}\|}\} \\ &\geq \delta \varepsilon \min\{\Delta_{k+\varrho}, \frac{\varepsilon}{M_B}\}. \end{aligned} \tag{13}$$

Thus, from Lemma 1 and Equation (13), we have

$$\begin{aligned} |\frac{h(x_{k+\varrho})-h(x_{k+\varrho}+t_{k+\varrho})}{\varpi_{k+\varrho}(0)-\varpi_{k+\varrho}(t_{k+\varrho})}-1| &= |\frac{h(x_{k+\varrho})-h(x_{k+\varrho}+t_{k+\varrho})-[\varpi_{k+\varrho}(0)-\varpi_{k+\varrho}(t_{k+\varrho})]}{\varpi_{k+\varrho}(0)-\varpi_{k+\varrho}(t_{k+\varrho})}| \\ &\leq \frac{o(\|t_{k+\varrho}\|^2)}{\delta\varepsilon\min\{\Delta_{k+\varrho},\frac{\varepsilon}{M_B}\}}. \end{aligned} \tag{14}$$

When $\varrho \to \infty$, we have $\|t_{k+\varrho}\|^2 \to 0$, and subsequently

$$\lim_{\varrho\to\infty}\frac{h(x_{k+\varrho})-h(x_{k+\varrho}+t_{k+\varrho})}{\varpi_{k+\varrho}(0)-\varpi_{k+\varrho}(t_{k+\varrho})}=1. \tag{15}$$

From Lemma 3 and Equation (15), we obtain

$$\begin{aligned} \tilde{r}_{k+\varrho} &= \frac{R_{k+\varrho}-h(x_{k+\varrho}+t_{k+\varrho})}{\varpi_{k+\varrho}(0)-\varpi_{k+\varrho}(t_{k+\varrho})} \\ &\geq \frac{h(x_{k+\varrho})-h(x_{k+\varrho}+t_{k+\varrho})}{\varpi_{k+\varrho}(0)-\varpi_{k+\varrho}(t_{k+\varrho})} \\ &\geq \mu_2. \end{aligned} \tag{16}$$

Therefore, when ϱ is sufficiently large, $\tilde{r}_{k+\varrho} \geq \mu_2 > \mu_1$. This contradicts the assumption that

$$\tilde{r}_{k+\varrho} < \mu_1,$$

and thus the hypothesis is not valid. □

Lemma 5. *Assuming that H3 is true, then there exists a constant $c_1 > 0$ such that t_k satisfies:*

$$\|t_k\| \leq c_1\|g_k\|.$$

Proof. From H3 and Equations (7) and (10), we obtain

$$\Delta_k \leq \bar{c}\|g_k\|\max\{\frac{1}{\omega},\theta\},$$

and subsequently, through $\|t_k\| \leq \Delta_k$ we have

$$\|t_k\| \leq \bar{c}\|g_k\|\max\{\frac{1}{\omega},\theta\}.$$

Set $c_1 = \bar{c}\max\{\frac{1}{\omega},\theta\}$. Then, we have

$$\|t_k\| \leq c_1\|g_k\|. \tag{17}$$

□

Lemma 6. *Assume that H1 and H2 are true and the sequence $\{x_k\}$ is generated by the NATR algorithm. Then, we have*

$$\lim_{k\to\infty} h_{l(k)} = \lim_{k\to\infty} h(x_k). \tag{18}$$

Proof. See Lemma 7 in [35]. □

Lemma 7. *Assume that the sequence $\{x_k\}$ is generated by the NATR algorithm. Then, we have*

$$\lim_{k\to\infty} R_k = \lim_{k\to\infty} h(x_k). \tag{19}$$

Proof. Using Equations (3) and (4) and Lemma 3, we obtain

$$h_k \leq R_k = \vartheta_k h_{l(k)} + (1-\vartheta_k)h_k \leq \vartheta_k h_{l(k)} + (1-\vartheta_k)h_{l(k)} = h_{l(k)}.$$

According to Lemma 6, we obtain

$$\lim_{k\to\infty} R_k = \lim_{k\to\infty} h(x_k).$$

□

Theorem 1. *(Global Convergence) Assume that H1, H2, and H3 are true, and the sequence* $\{x_k\}$ *is generated by the NATR algorithm. Then, we have*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{20}$$

Proof. By contradiction, we assume that there exists a constant $\varepsilon > 0$ such that

$$\|g_k\| \geq \varepsilon, \tag{21}$$

for any integer k.

When k is sufficiently large, with Equation (9), Lemma 3, and Lemma 4, we have

$$R_k - h(x_k + t_k) \geq \mu_1 Pret_k. \tag{22}$$

Using Equations (11) and (22), we obtain

$$h_{k+1} = h(x_k + t_k) \leq R_k - \mu_1 Pret_k \leq R_k - \mu_1 \delta \|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\}. \tag{23}$$

From H2, Equation (23), Lemma 3, and the definition of R_k, it follows that

$$\begin{aligned} h_{k+1} \leq R_{k+1} &= \vartheta_{k+1} h_{l(k+1)} + (1-\vartheta_{k+1})h_{k+1} \\ &\leq \vartheta_{k+1} h_{l(k)} + (1-\vartheta_{k+1})[R_k - \mu_1 \delta \|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\}] \\ &\leq \vartheta_{k+1} h_{l(k)} + (1-\vartheta_{k+1})[h_{l(k)} - \mu_1 \delta \|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\}] \\ &= \vartheta_{k+1} h_{l(k)} + (1-\vartheta_{k+1})h_{l(k)} - (1-\vartheta_{k+1})\mu_1 \delta \|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\} \\ &= h_{l(k)} - (1-\vartheta_{k+1})\mu_1 \delta \|g_k\| \min\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\} \\ &\leq h_{l(k)} - (1-\vartheta_{k+1})\mu_1 \delta \varepsilon \min\{\Delta_k, \frac{\varepsilon}{M_B}\}. \end{aligned} \tag{24}$$

This implies that

$$h_{l(k)} - h_{k+1} \geq (1-\vartheta_{k+1})\mu_1 \delta \varepsilon \min\{\Delta_k, \frac{\varepsilon}{M_B}\}. \tag{25}$$

Based on Equation (25) and Lemma 6, we write

$$\lim_{k\to\infty} \min\{\Delta_k, \frac{\varepsilon}{M_B}\} = 0, \tag{26}$$

and subsequently, $\lim_{k\to\infty} \Delta_k = 0$; in other words, we have

$$\lim_{k\to\infty} \Delta_k = \lim_{k\to\infty} c_k \|g_k\| \lambda_k = 0. \tag{27}$$

If $\tilde{r}_k \geq \mu_2$—that is, x_{k+1} is a successful iteration point—then there exists a constant $c^+ > 0$ such that

$$c_k \geq c^+,$$

for sufficiently large k values. This information and Equation (10) suggest that

$$\Delta_k = c_k \|g_k\| \lambda_k \geq \varepsilon \min\{\omega, \theta\} c_k. \tag{28}$$

With Equations (27) and (28), we can deduce that

$$\lim_{k \to \infty} c_k = 0,$$

which contradicts the assumption that $c_k \geq c^+ > 0$.

Hence, the proof is completed. □

Remark 3. *It can be seen in Theorem 1 that the NATR algorithm has global convergence; that is, the algorithm is feasible theoretically.*

4. Preliminary Numerical Experiments

In this section, we test and compare the NATR algorithm with the traditional trust region algorithm in [8] (TTR) and the non-monotonic trust region algorithm proposed by Ahookhoosh et al. in [35] (NTR). All unconstrained optimization test problems were selected from [40], and all of the numerical experiments were carried out in the Matlab 2018 environment.

In all algorithms, we set $\Delta_0 = 2, B_0 = I$, $\mu_1 = 0.25$, $\mu_2 = 0.75$, $t_1 = 0.5$, $t_2 = 1.5$, $c_0 = 1$, $M = 5$, $\omega = 10^{-6}$, $\Delta_{\max} = 200$, and $\varepsilon = 10^{-6}$. If $\lambda_{k+1} \leq \omega$, then we set $\theta = \omega$, or if $\lambda_{k+1} \geq \frac{1}{\omega}$, then we set $\theta = \frac{1}{\omega}$. Moreover, we set $\vartheta_k = \begin{cases} \frac{\vartheta_0}{2}, & k = 1, \\ \frac{\vartheta_{k-1} + \vartheta_{k-2}}{2}, & k \geq 2, \end{cases}$ $\vartheta_0 = 0.85$ [35]. The matrix B_k is updated by the BFGS formula in [41], as follows:

$$B_{k+1} = \begin{cases} B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}, & s_k^T y_k > 0, \\ B_k, & s_k^T y_k \leq 0, \end{cases}$$

where $y_k = g_{k+1} - g_k$ and $s_k = x_{k+1} - x_k$. Furthermore, we declare that an algorithm will fail if it has more than 10,000 iterations. All of the algorithms are stopped when $\|g_k\| \leq 10^{-6}$.

In Table 1, the name of the test function is denoted as "function", the dimension of the test function is denoted as "n", the number of iterations of the algorithm is denoted as "k", and the time used by the algorithm to solve the test function is denoted as "Cpu" (in seconds). If the algorithm is iterated more than 10,000 times, then it fails and is marked with "F".

According to the numerical results in Table 1, we can find that when the NATR algorithm was applied to solve these unconstrained optimization problems, it required fewer iterations than the TTR and NTR methods. Especially for the perturbed quadratic diagonal function and diagonal 2 function, our NATR algorithm could solve the problem in fewer iterations. However, the iteration times of the NTR algorithm and TTR algorithm both exceeded 10,000, which means that they could not effectively solve these problems. For the POWER function (CUTE), the CPU time required by the NATR algorithm was only 1/30 and 1/3 of that needed by the NTR and TTR methods, respectively. This is because neither the NTR nor TTR method can correctly adjust the radius of the confidence region, while the radius of the trust region of the NATR algorithm is automatically adjusted according to the gradient of the current iteration point and the eigenvalue of the Hessian matrix, thus reducing the number of iterations and reducing the calculation cost. For most test functions, the three algorithms ended up with the same function value results. For some functions, such as the POWER function (CUTE), the NATR algorithm not only had fewer iterations and fewer iterations, but the function value was also smaller.

Table 1. Numerical results.

Function	n	NATR (k/cpu/h)	NTR (k/cpu/h)	TTR (k/cpu/h)
POWER function (CUTE)	10	$21/0.030242/1.1308 \times 10^{-16}$	$634/0.133546/2.2135 \times 10^{-13}$	$71/24.828385/1.0899 \times 10^{-13}$
Quadratic QF1 function	10	$28/0.033207/-5.0000$	$43/0.040731/-5.0000$	$31/21.791012/-5.0000$
Raydan 1 function	10	21/0.018676/5.5000	59/0.020833/5.5000	87/16.056837/5.5000
Extended DENSCHNB function (CUTE)	10	12/0.035049/5.0000	15/0.031511/5.0000	12/23.596389/5.0000
Diagonal 2 function	100	$95/0.168606/1.5741 \times 10$	$496/0.270396/1.5741 \times 10$	$1364/\ 21.161857/1.5741 \times 10$
QUARTC function(CUTE)	100	$27/0.07209/6.3426 \times 10^{-9}$	$39/0.032488/4.7396 \times 10^{-9}$	F/F/F
HIMMELBG function	100	$21/0.10308/4.5453 \times 10^{-6}$	$32/0.159167/4.5453 \times 10^{-6}$	F/F/F
Diagonal 2 function	500	$193/4.646668/2.6037 \times 10$	$2056/7.205046/2.6037 \times 10$	$6410/257.345146/2.6037 \times 10$
Perturbed quadratic diagonal function	500	$3/0.234107/4.3812 \times 10^{6}$	F/F/F	F/F/F
Diagonal 7 function	500	$8/0.141167/-4.0842 \times 10^{2}$	$17/0.164094/-4.0842 \times 10^{2}$	$8/19.918756/-4.0842 \times 10^{2}$
Diagonal 4 function	900	$9/0.769483/2.2052 \times 10^{2}$	$15/0.831553/2.2052 \times 10^{2}$	$9/23.19987/2.2052 \times 10^{2}$
Diagonal 2 function	1000	$260/44.791428/3.1275 \times 10^{1}$	F/F/F	F/F/F
QUARTC function	1000	$29/3.777523/6.6878 \times 10^{-9}$	$44/0.677226/1.4397 \times 10^{-8}$	F/F/F
Raydan 2 function	1000	$8/1.035779/1.0000 \times 10^{3}$	$23/1.317213/1.0000 \times 10^{3}$	$9/29.141964/1.0000 \times 10^{3}$
Diagonal 5 function	1000	$6/0.642228/6.9315 \times 10^{2}$	$12/0.749363/6.9315 \times 10^{2}$	$7/19.115579/6.9315 \times 10^{2}$
Diagonal 8 function	1000	$10/1.33452/-4.8045 \times 10^{2}$	$12/1.302234/-4.8045 \times 10^{2}$	$10/73.357103/-4.8045 \times 10^{2}$
Raydan 2 function	2000	$8/11.038501/2.0000 \times 10^{3}$	$25/12.024056/2.0000 \times 10^{3}$	$9/72.307226/2.0000 \times 10^{3}$
HIMMELBG function	3000	$24/46.598187/1.5727 \times 10^{-5}$	$47/38.713961/2.2028 \times 10^{-5}$	F/F/F
QUARTC function	3000	$29/41.807088/2.0063 \times 10^{-8}$	$37/24.486504/9.8927 \times 10^{-9}$	F/F/F
Diagonal 5 function	3000	$6/9.334939/2.0794 \times 10^{3}$	$9/4.407844/2.0794 \times 10^{3}$	$7/41.826252/2.0794 \times 10^{3}$
Perturbed quadratic diagonal function	5000	$3/61.283901/3.2506 \times 10^{8}$	F/F/F	F/F/F

For higher-dimension problems, such as the 3000 dimensional test functions of the HIMMELBG function and QUARTC function (CUTE) and the 5000 dimensional test functions of the perturbed quadratic diagonal function, the NATR algorithm could solve them effectively, but the TTR algorithm failed to solve them. This shows that the NATR algorithm has a certain effect when solving higher-dimension problems.

Overall, our new algorithm was effective for most test problems and took fewer iterations, as demonstrated by these numerical findings for both high- and low-dimensional test issues. In general, we can infer that the NATR algorithm is more efficient than the traditional trust region (TTR) algorithm and the non-monotonic trust region (NTR) algorithm in terms of the number of iterations and running time. Therefore, the NATR algorithm can effectively solve unconstrained optimization problems.

5. Conclusions

In this paper, based on the current iteration point gradient g_k and the eigenvalues of the Hessian matrix B_k, we proposed a new improved adaptive radius updating criterion which reduces the effort and computation time and improves the performance of an algorithm. Then, we proposed an improved non-monotonic adaptive trust region (NATR) algorithm by combining the adaptive radius updating criterion with the non-monotonic technique. Under certain common assumptions, the global convergence of the NATR algorithm was demonstrated. Finally, in the same environment, the test function was used to conduct numerical experiments on the three algorithms. The numerical experiments show that the NATR algorithm can effectively solve unconstrained problems, and for some test functions, the iteration time and iteration times of the NATR algorithm were less than those of the NTR algorithm and TTR algorithm, demonstrating an effective algorithm.

In the NATR algorithm, when the trust region ratio did not meet the requirements, the subproblem was recalculated to find the test step d_k which met the conditions. In the future, we will consider combining the trust region algorithm with the line search method to avoid double-computation subproblems. Secondly, the application of the non-monotonic trust region method will be studied further, such as by applying it to nonlinear fractional-order multi-agent systems [42] and cooperative control for linear delta operator systems [43].

Author Contributions: Writing—original draft, M.X.; Writing—review & editing, Q.Z.; Visualization, H.X.; Supervision, H.X. All authors contributed equally to this paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the specialized research fund of YiBin University (Grant No. 412-2021QH027).

Data Availability Statement: The data that support the findings of this study are available on request from the corresponding author, upon reasonable request.

Acknowledgments: The authors would like to thank the Editor and the anonymous referees for their helpful comments and valuable suggestions regarding this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Andrei, N. Another hybrid conjugate gradient algorithm for unconstrained optimization. *Numer. Algorithms* **2008**, *47*, 143–156. [CrossRef]
2. Fatemi, M. A new efficient conjugate gradient method for unconstrained optimization. *J. Comput. Appl. Math.* **2016**, *300*, 207–216. [CrossRef]
3. Kimiaei, M.; Ghaderi, S. A new restarting adaptive trust-region method for unconstrained optimization. *J. Oper. Res. Soc. China* **2017**, *5*, 487–507. [CrossRef]
4. Xue, Y.; Liu, H.; Liu, Z. An improved nonmonotone adaptive trust region method. *Appl. Math.* **2019**, *64*, 1–16. [CrossRef]
5. Gao, H.; Zhang, H.B.; Li, Z.B. A nonmonotone inexact Newton method for unconstrained optimization. *Optim. Lett.* **2017**, *11*, 947–965. [CrossRef]

6. Wei, Z.; Li, G.; Qi, L. New quasi-Newton methods for unconstrained optimization problems. *Appl. Math. Comput.* **2006**, *175*, 1156–1188. [CrossRef]
7. Fletcher, R. Conjugate gradient methods for indefinite systems. In *Numerical Analysis: Proceedings of the Dundee Conference on Numerical Analysis, 1975*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 73–89.
8. Ma, C.F. *Optimization Method and Matlab Program Design*; Science Press: Beijing, China, 2010.
9. Mo, J.; Zhang, K.; Wei, Z. A nonmonotone trust region method for unconstrained optimization. *Appl. Math. Comput.* **2005**, *171*, 371–384. [CrossRef]
10. Nocedal, J.; Yuan, Y. Combining trust region and line search techniques. In *Advances in Nonlinear Programming: Proceedings of the 96 International Conference on Nonlinear Programming*; Springer: New York, NY, USA, 1998; pp. 153–175.
11. Jiang, X.; Huang, Z. An accelerated relaxed-inertial strategy based CGP algorithm with restart technique for constrained nonlinear pseudo-monotone equations to image de-blurring problems. *J. Comput. Appl. Math.* **2024**, *447*, 115887. [CrossRef]
12. Liu, J.; Xu, X.; Cui, X. An accelerated nonmonotone trust region method with adaptive trust region for unconstrained optimization. *Comput. Optim. Appl.* **2018**, *69*, 77–97. [CrossRef]
13. Xu, H.; Xuan, J.; Zhang, G.; Lu, J. Trust region policy optimization via entropy regularization for Kullback CLeibler divergence constraint. *Neurocomputing* **2024**, *589*, 127716. [CrossRef]
14. Yu, Z.; Yuan, Y.; Tian, P. An efficient trust region algorithm with bounded iteration sequence for unconstrained optimization and its application in support vector machine. *J. Comput. Appl. Math.* **2024**, *449*, 115956. [CrossRef]
15. Goharian, M.; Soleimani, M.; Jegatheesan, A.; Moran, G.R. *Regularization of EIT Problem Using Trust Region SubProblem Method*; Springer: Berlin/Heidelberg, Germany, 2007.
16. Weiping, W.; Hongchun, W.; Haofeng, Z.; Xiong, X. A Pn-wave spectral inversion technique based on trust region reflective algorithm. *J. Appl. Geophys.* **2024**, *230*, 105525. [CrossRef]
17. Ceng, L.; Yuan, Q. On a General Extragradient Implicit Method and Its Applications to Optimization. *Symmetry* **2020**, *12*, 124. [CrossRef]
18. Sun, W.; Yuan, Y. *Optimization Theory and Methods: Nonlinear Programming*; Springer: New York, NY, USA, 2006.
19. Sartenaer, A. Automatic Determination Of An Initial Trust Region In Nonlinear Programming. *Siam J. Sci. Comput.* **1997**, *18*, 1788–1803. [CrossRef]
20. Lin, C.; Mor, J. Newton's method for large bound-constrained optimization problems. *Siam J. Optim.* **1999**, *9*, 1100–1127. [CrossRef]
21. Zhang, X. NN models for general nonlinear programming. In *Neural Networks in Optimization*; Springer: Boston, MA, USA, 2000; pp. 273–288.
22. Zhang, X.; Zhang, J.; Liao, L. An adaptive trust region method and its convergence. *Sci. China Ser. Math.* **2002**, *45*, 620–631. [CrossRef]
23. Shi, Z.; Wang, S. Nonmonotone adaptive trust region method. *Eur. J. Oper. Res.* **2011**, *208*, 28–36. [CrossRef]
24. Sheng, Z.; Gonglin, Y.; Zengru, C. A new adaptive trust region algorithm for optimization problems. *Acta Math. Sci.* **2018**, *38*, 479–496. [CrossRef]
25. Wang, X.; Ding, X.; Qu, Q. A new filter nonmonotone adaptive trust region method for unconstrained optimization. *Symmetry* **2020**, *12*, 208. [CrossRef]
26. Grippo, L.; Lampariello, F.; Lucidi, S. A nonmonotone line search technique for Newton's method. *Siam J. Numer. Anal.* **1986**, *23*, 707–716. [CrossRef]
27. Grippo, L.; Lampariello, F.; Lucidi, S. A truncated Newton method with nonmonotone line search for unconstrained optimization. *J. Optim. Theory Appl.* **1989**, *60*, 401–419. [CrossRef]
28. Toint, P.L. Non-monotone trust-region algorithms for nonlinear optimization subject to convex con-straints. *Math. Progr.* **1997**, *77*, 69–94. [CrossRef]
29. Zhang, H.; Hager, W. A nonmonotone line search technique and its application to unconstrained optimization. *Siam J. Optim.* **2004**, *14*, 1043–1056. [CrossRef]
30. Cui, Z.; Wu, B. A new modified nonmonotone adaptive trust region method for unconstrained optimization. *Comput. Optim. Appl.* **2012**, *53*, 795–806. [CrossRef]
31. Reza Peyghami, M.; Ataee Tarzanagh, D. A relaxed nonmonotone adaptive trust region method for solving unconstrained optimization problems. *Comput. Optim. Appl.* **2015**, *61*, 321–341. [CrossRef]
32. Rezaee, S.; Babaie-Kafaki, S. An adaptive nonmonotone trust region algorithm. *Optim. Methods Softw.* **2019**, *34*, 264–277. [CrossRef]
33. Zhang, J.L.; Zhang, X.S. A nonmonotone adaptive trust region method and its convergence. *Comput. Math. Appl.* **2003**, *45*, 1469–1477. [CrossRef]
34. Chamberlain, R.M.; Powell, M.J.D.; Lemarechal, C.; Pedersen, H.C. The watchdog technique for forcing convergence in algorithms for constrained optimization. In *Algorithms for Constrained Minimization of Smooth Nonlinear Functions*; Springer: Berlin/Heidelberg, Germany, 1982; pp. 1–17.
35. Ahookhosh, M.; Amini, K. An efficient nonmonotone trust-region method for unconstrained optimization. *Numer. Algorithms* **2012**, *59*, 523–540. [CrossRef]
36. Ahookhosh, M.; Amini, K. A Nonmonotone trust region method with adaptive radius for unconstrained optimization problems. *Comput. Math. Appl.* **2010**, *60*, 411–422. [CrossRef]

37. Fu, J.; Sun, W. Nonmonotone adaptive trust-region method for unconstrained optimization problems. *Appl. Math. Comput.* **2005**, *163*, 489–504. [CrossRef]
38. Varga, R.S. *Geršgorin and His Circles*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 36.
39. Sang, Z.; Sun, Q. Aself-adaptive trust region method with linesearch based on a simple subproblem model. *J. Comput. Appl. Math.* **2009**, *232*, C514–C522. [CrossRef]
40. Andrei, N. An unconstrained optimization test functions collection. *Adv. Model. Optim.* **2008**, *10*, 147–161.
41. Nocedal, J.; Wright, S.J. *Numerical Optimization*; Springer: New York, NY, USA, 2006.
42. Yang, Y.; Qi, Q.; Hu, J.; Dai, J.; Yang, C. Adaptive Fault-Tolerant Control for Consensus of Nonlinear Fractional-Order Multi-Agent Systems with Diffusion. *Fractal Fract.* **2023**, *7*, 760. [CrossRef]
43. Xue, Y.; Han, J.; Tu, Z.; Chen, X. Stability analysis and design of cooperative control for linear delta operator system. *Aims Math.* **2023**, *8*, 12671–12693. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.