Computational Optimization and Applications (2022) 82:31–60
https://doi.org/10.1007/s10589-022-00356-0

An adaptive trust-region method without function evaluations

Geovani N. Grapiglia[1] · **Gabriel F. D. Stella**[2]

Received: 20 July 2021 / Accepted: 3 February 2022 / Published online: 2 March 2022

Abstract
In this paper we propose an adaptive trust-region method for smooth unconstrained optimization. The update rule for the trust-region radius relies only on gradient evaluations. Assuming that the gradient of the objective function is Lipschitz continuous, we establish worst-case complexity bounds for the number of gradient evaluations required by the proposed method to generate approximate stationary points. As a corollary, we establish a global convergence result. We also present numerical results on benchmark problems. In terms of the number of calls of the oracle, the proposed method compares favorably with trust-region methods that use evaluations of the objective function.

Keywords Unconstrained optimization · Trust-region method · Global convergence · Worst-case complexity

1 Introduction

In this paper we are interested in the smooth unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and bounded from below. Trust-region methods form an important class of iterative methods for this type of optimization problem (see, e.g., [8, 15, 16, 22, 26, 27, 35]). A model version of the standard trust-region method works in the following way [35]. At the beginning

✉ Geovani N. Grapiglia
geovani.grapiglia@uclouvain.be

[1] Université catholique de Louvain, ICTEAM/INMA, Avenue Georges Lemaître, 4-6/ L4.05.01, 1348 Louvain-la-Neuve, Belgium

[2] Programa de Pós-Graduação em Matemática, Centro Politécnico, Universidade Federal do Paraná, Cx. Postal 19.081, Curitiba, Paraná 81531-980, Brazil

of the kth iteration one has an estimate x_k for the solution of (1.1), a symmetric matrix $B_k \in \mathbb{R}^{n\times n} \setminus \{0\}$, a trust-region radius $\Delta_k > 0$, and constants $0 < \eta_1 < \eta_2 < 1$. When f is twice-differentiable, the matrix B_k may be the Hessian matrix of f at x_k or an approximation of it. A trial step d_k is computed by solving the *trust-region subproblem*

$$\min_{d\in\mathbb{R}^n} \quad m_k(d) \equiv \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d, \tag{1.2}$$

$$\text{s.t.} \quad \|d\| \leq \Delta_k. \tag{1.3}$$

It is not necessary to solve the trust-region subproblem exactly. The vector d_k can be an inexact solution of (1.2)–(1.3) satisfying

$$m_k(0) - m_k(d_k) \geq \frac{1}{2}\|\nabla f(x_k)\| \min\left\{\Delta_k, \frac{\|\nabla f(x_k)\|}{\|B_k\|}\right\}.$$

The quality of d_k is assessed using the ratio between the actual reduction in the objective function and the predicted reduction:

$$\rho_k = \frac{f(x_k) - f(x_k + d_k)}{m_k(0) - m_k(d_k)} \tag{1.4}$$

Specifically, the iterate x_k is updated by the rule

$$x_{k+1} = \begin{cases} x_k + d_k, & \text{if } \rho_k \geq \eta_1, \\ x_k, & \text{otherwise,} \end{cases} \tag{1.5}$$

while the trust-region radius Δ_k is updated by the rule

$$\Delta_{k+1} = \begin{cases} 2\Delta_k, & \text{if } \rho_k \geq \eta_2, \\ \Delta_k, & \text{if } \rho_k \in [\eta_1, \eta_2), \\ \frac{1}{2}\Delta_k, & \text{if } \rho_k < \eta_1. \end{cases} \tag{1.6}$$

From rules (1.4)–(1.6), we see that the assessment of the trial step and the adjustment of the trust-region radius are done at the expense of evaluations of f, which are required to compute ρ_k. In particular, if $\rho_k < \eta_1$ then d_k is rejected and a new trial step must be computed by approximately solving (1.2)–(1.3) with a smaller trust-region radius. Numerous function evaluations may be required until an acceptable trial step is generated, which is an issue in applications where evaluations of the objective function are very expensive. Examples include generalized Nash equilibrium problems [23], full waveform inversion [36], microwave design optimization [24], optical tomography [3], and calibration of ODE models [10, 33], just to mention a few.

In this paper we propose an adaptive trust-region method without function evaluations. Our method relies only on gradient evaluations, and admits two variants: a *conservative variant* and a *flexible variant*. We establish worst-case complexity and

global convergence results assuming that the gradient of the objective function is Lipschitz continuous. Specifically, we show that the conservative variant takes at most $\mathcal{O}\left(\epsilon^{-2}\right)$ gradient evaluations to generate x_k such that $\|\nabla f(x_k)\| \leq \epsilon$ for a given $\epsilon \in (0,1)$, while the flexible variant takes at most $\mathcal{O}\left(|\log(\epsilon)|\epsilon^{-2}\right)$ gradient evaluations. As a consequence of our complexity bounds, we establish a liminf-type global convergence result.

The paper is organized as follows. In Section 2, we describe our adaptive trust-region method and present its worst-case complexity analysis. In Section 3, we report numerical results on benchmark problems. Finally, in Section 4, the obtained results are summarized.

2 Algorithm and its worst-case complexity analysis

Our adaptive trust-region method works in the following way. At the beginning of the kth iteration we have an estimate x_k for the solution of (1.1), a symmetric matrix $B_k \in \mathbb{R}^{n\times n} \setminus \{0\}$ and a scaling coefficient $b_k > 0$. A step d_k is computed as an approximate solution to the trust-region subproblem

$$\min_{d\in\mathbb{R}^n} m_k(d) \equiv \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d, \tag{2.1}$$

$$\text{s.t.} \|d\| \leq \Delta_k \equiv \delta_k \|\nabla f(x_k)\|, \tag{2.2}$$

where $\delta_k = 1/b_k$. The step d_k must satisfy the condition

$$m_k(0) - m_k(d_k) \geq \frac{1}{2}\|\nabla f(x_k)\| \min\left\{\Delta_k, \frac{\|\nabla f(x_k)\|}{\|B_k\|}\right\}. \tag{2.3}$$

Then, the next iterate is defined as $x_{k+1} = x_k + d_k$. The novelty in our method is in the update of δ_k, which is implicitly done by the updating of b_k. If $\|\nabla f(x_{k+1})\|$ is sufficiently small, we choose b_{k+1} less than or equal to b_k, which gives $\delta_{k+1} \geq \delta_k$. Otherwise, we choose b_{k+1} greater than b_k, which gives $\delta_{k+1} < \delta_k$. Specifically, defining $\omega_0 = \|\nabla f(x_0)\|$ and considering parameters $\alpha \in [0,1)$, $b_{\min} > 0$, and $b_0 \geq b_{\min}$, we jointly update b_k and ω_k as follows:

$$\left(b_{k+1}, \omega_{k+1}\right) = \begin{cases} \left(\hat{b}_k, \|\nabla f(x_{k+1})\|\right), & \text{if} \quad \|\nabla f(x_{k+1})\| \leq \alpha\omega_k, \\ \left(b_k + \dfrac{\|\nabla f(x_{k+1})\|^2}{b_k}, \omega_k\right), & \text{otherwise,} \end{cases} \tag{2.4}$$

where

$$\hat{b}_k \in [b_{\min}, b_k]. \tag{2.5}$$

Once we have x_{k+1} and b_{k+1}, we choose a symmetric matrix B_{k+1}, and we repeat the process with $k := k+1$. This method can be summarized as follows.

Algorithm 1. Adaptive Trust-Region (AdaTrust) Method

Step 0. Choose $x_0 \in \mathbb{R}^n$, a symmetric matrix $B_0 \in \mathbb{R}^{n\times n}$, $b_{\min} > 0$, $b_0 \geq b_{\min}$, and $\alpha \in [0, 1)$. Set $\delta_0 = 1/b_0$, $\omega_0 = \|\nabla f(x_0)\|$ and $k := 0$.

Step 1. Compute an approximate solution d_k to (2.1)-(2.2) satisfying condition (2.3), and define $x_{k+1} = x_k + d_k$.

Step 2. Define b_{k+1} and ω_{k+1} by (2.4)-(2.5).

Step 3. Choose a symmetric matrix $B_{k+1} \in \mathbb{R}^{n\times n}$, define $\delta_{k+1} = 1/b_{k+1}$, set $k := k + 1$ and go to Step 1.

Remark 1 Algorithm 1 combines elements of existing methods for smooth unconstrained optimization. In particular, the use of a trust-region radius of the form $\Delta_k = \delta_k\|\nabla f(x_k)\|$ was first proposed by Fan and Yuan [13]. Moreover, the rule

$$b_{k+1} = b_k + \frac{\|\nabla f(x_{k+1})\|^2}{b_k}$$

used when $\|\nabla f(x_{k+1})\| > \alpha\omega_k$ is inspired in the stepsize rule of the (batch) WNGrad method [34].

Remark 2 At each iteration, Algorithm 1 requires only one gradient evaluation. In contrast, the standard trust-region method requires one gradient evaluation and one function evaluation at successful iterations, and one function evaluation at unsuccessful iterations. Therefore, if the cost of computing $f(x)$ is similar to the cost of computing $\nabla f(x)$, then each iteration of Algorithm 1 is computationally cheaper than a successful iteration of the standard TR method.

Remark 3 In Section 3, we give a possible choice for $\hat{b}_k$ in (2.5).

The variant of Algorithm 1 with $\alpha = 0$ will be referred as the *conservative variant*, since in this case rule (2.4) generates a sequence $\{\delta_k\}$ monotonically decreasing. In contrast, the variant of Algorithm 1 with $\alpha \in (0, 1)$ will be referred as the *flexible variant*, since in this case (2.4)–(2.5) allow $\{\delta_k\}$ to be nonmonotone when $\hat{b}_k < b_k$ for some k. Our analysis of both variants of Algorithm 1 will be done under the following assumptions:

A1. The gradient of the objective function $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is L-Lipschitz.

A2. There exists $f_{low} \in \mathbb{R}$ such that $f(x) \geq f_{low}$ for all $x \in \mathbb{R}^n$.

A3. There exists $M > 0$ such that $\|B_k\| \leq M$ for all k.

Given $\ell \in \mathbb{N} \setminus \{0\}$, let us define the set

$$D_\ell = \{1 \leq k \leq \ell \ : \ \|\nabla f(x_k)\| \leq \alpha\omega_{k-1}\}. \tag{2.6}$$

In what follows, given a set A, we will denote its cardinality by $|A|$.

Lemma 1 *Let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1. Suppose that $D_\ell \neq \emptyset$ for some $\ell \in \mathbb{N} \setminus \{0\}$, and denote $D_\ell = \{k_i\}_{i=1}^{|D_\ell|}$ with $k_i < k_j$ whenever $i < j$. Then*

$$\|\nabla f(x_{k_i})\| \leq \alpha^i \|\nabla f(x_0)\| \quad \text{for } i = 1, \ldots, |D_\ell|.$$

Proof By (2.6), (2.4) and $\omega_0 = \|\nabla f(x_0)\|$, we have

$$\omega_k = \omega_0 = \|\nabla f(x_0)\| \quad \text{for } k = 0, \ldots, k_1 - 1, \tag{2.7}$$

$$\omega_k = \omega_{k_i} \quad \text{for } k = k_i, \ldots, k_{i+1} - 1, \tag{2.8}$$

and

$$\omega_{k_i} = \|\nabla f(x_{k_i})\| \leq \alpha \omega_{k_i - 1} \quad \text{for } i = 1, \ldots, |D_\ell|. \tag{2.9}$$

Combining (2.7)–(2.9), we obtain

$$\begin{aligned}
\|\nabla f(x_{k_1})\| &\leq \alpha \omega_{k_1 - 1} = \alpha \omega_0 = \alpha \|\nabla f(x_0)\|, \\
\|\nabla f(x_{k_2})\| &\leq \alpha \omega_{k_2 - 1} = \alpha \omega_{k_1} = \alpha \|\nabla f(x_{k_1})\| \leq \alpha^2 \|\nabla f(x_0)\|, \\
\|\nabla f(x_{k_3})\| &\leq \alpha \omega_{k_3 - 1} = \alpha \omega_{k_2} = \alpha \|\nabla f(x_{k_2})\| \leq \alpha^3 \|\nabla f(x_0)\|,
\end{aligned}$$

and so

$$\|\nabla f(x_{k_i})\| \leq \alpha^i \|\nabla f(x_0)\| \quad \text{for } i = 1, \ldots, |D_\ell|.$$

□

Given $p, q \in \mathbb{N}$ with $p \leq q$, let us define

$$I(p, q) = \{k \in \mathbb{N} \ : \ p \leq k \leq q\}. \tag{2.10}$$

The next lemma provides an upper bound for $|D_T|$ when $\|\nabla f(x_j)\| > \epsilon$ for $j = 0, 1, \ldots, T$.

Lemma 2 *Let $\{x_k\}_{k \geq 0}$ be a sequence generated by Algorithm 1. Given $\epsilon \in (0, 1)$, suppose that*

$$\|\nabla f(x_k)\| > \epsilon, \quad \text{for} \quad k = 0, 1, \ldots, T. \tag{2.11}$$

Then

$$|D_T| = 0 \quad \text{if} \quad \alpha = 0, \tag{2.12}$$

and

$$|D_T| < \frac{\log\left(\|\nabla f(x_0)\| \epsilon^{-1}\right)}{|\log(\alpha)|} \quad \text{if} \quad \alpha \in (0, 1). \tag{2.13}$$

Moreover, denoting $k_0 = 0$ and $k_{|D_T|+1} = T + 1$, we have

$$T + 1 \leq \left(|D_T| + 1\right) \max_{i = 0, \ldots, |D_T|} |I(k_i, k_{i+1} - 1)|. \tag{2.14}$$

Proof Suppose that $\alpha = 0$. Then, by (2.6) and (2.11) we have

$$D_T = \left\{1 \leq k \leq T \ : \ \|\nabla f(x_k)\| \leq 0\right\} = \emptyset.$$

Thus, (2.12) is true. Now, suppose that $\alpha \in (0, 1)$. If $D_T = \emptyset$ then

$$|D_T| = 0 < \frac{\log\left(\|\nabla f(x_0)\|\epsilon^{-1}\right)}{|\log(\alpha)|},$$

that is, (2.13) holds. On the other hand, if $D_T \neq \emptyset$, it follows from Lemma 1 that

$$\|\nabla f(x_{k_{|D_T|}})\| \leq \alpha^{|D_T|}\|\nabla f(x_0)\|. \tag{2.15}$$

Assume by contradiction that

$$|D_T| \geq \frac{\log\left(\|\nabla f(x_0)\|\epsilon^{-1}\right)}{|\log(\alpha)|}.$$

Then we get

$$\alpha^{|D_T|}\|\nabla f(x_0)\| \leq \epsilon, \tag{2.16}$$

which combined with (2.15) implies that

$$\|\nabla f(x_{k_{|D_T|}})\| \leq \epsilon,$$

contradicting (2.11). Thus, in this case, (2.13) is also true.

Finally, using the notation $k_0 = 0$ and $k_{|D_T|+1} = T + 1$, it follows from (2.10) that

$$\{0, 1, \ldots, T\} = \cup_{i=0}^{|D_T|} I(k_i, k_{i+1} - 1).$$

Thus,

$$\begin{aligned} T + 1 =& |\{0, 1, \ldots, T\}| = \left|\cup_{i=0}^{|D_T|} I(k_i, k_{i+1} - 1)\right| \\ =& \sum_{i=0}^{|D_T|} |I(k_i, k_{i+1} - 1)| \\ \leq& \sum_{i=0}^{|D_T|} \max_{i=0,\ldots,|D_T|} |I(k_i, k_{i+1} - 1)| \\ =& (|D_T| + 1) \max_{i=0,\ldots,|D_T|} |I(k_i, k_{i+1} - 1)|, \end{aligned}$$

that is, inequality (2.14) is true. □

Given $\epsilon \in (0, 1)$, the goal of our analysis is to obtain a finite upper bound in terms of ϵ for the first iteration index T such that

$$\|\nabla f(x_{T+1})\| \leq \epsilon.$$

Note that, for such T, (2.11) holds. Thus, in view of Lemma 2, it is enough to derive an upper bound to $\max_{i=0,\ldots,|D_T|} |I(k_i, k_{i+1} - 1)|$. For that, we will need several auxiliary results.

The next lemma establishes that $f(x_{k+1}) < f(x_k)$ when b_k is sufficiently large.

Lemma 3 *Suppose that A1 and A3 hold. If*

$$b_k \geq 2(L + M) \equiv \tilde{L}, \tag{2.17}$$

then

$$f(x_{k+1}) \leq f(x_k) - \frac{\|\nabla f(x_k)\|^2}{4b_k}. \tag{2.18}$$

Proof Since ∇f is L-Lipschitz we have

$$|f(x_k + d_k) - f(x_k) - \nabla f(x_k)^T d_k| \leq \frac{L}{2}\|d_k\|^2.$$

Then, by A3, (2.1) and (2.2), we get

$$\begin{aligned} f(x_{k+1}) - f(x_k) - m_k(d_k) =& f(x_{k+1}) - f(x_k) - \nabla f(x_k)^T d_k - \frac{1}{2} d_k^T B_k d_k \\ \leq& \frac{L}{2}\|d_k\|^2 + \frac{1}{2}\|B_k\|\|d_k\|^2 \\ \leq& \frac{(L+M)}{2}\|d_k\|^2 \\ \leq& \frac{(L+M)}{2}\delta_k^2\|\nabla f(x_k)\|^2. \end{aligned} \tag{2.19}$$

Now, combining (1.4), (2.19), (2.3) and A3, it follows that

$$\begin{aligned} 1 - \rho_k =& \frac{f(x_{k+1}) - f(x_k) - m_k(d_k)}{m_k(0) - m_k(d_k)} \\ \leq& \frac{(L+M)\delta_k^2\|\nabla f(x_k)\|^2}{\|\nabla f(x_k)\| \min\left\{\delta_k\|\nabla f(x_k)\|, \dfrac{\|\nabla f(x_k)\|}{\|B_k\|}\right\}} \\ \leq& \frac{(L+M)\delta_k^2}{\min\left\{\delta_k, \frac{1}{M}\right\}}. \end{aligned}$$

Recall that $\delta_k = 1/b_k$. Thus, by (2.17) we have

$$\delta_k \leq \frac{1}{2(L+M)} < \frac{1}{M}. \tag{2.20}$$

Therefore,

$$1-\rho_k \leq \frac{(L+M)\delta_k^2}{\delta_k} = (L+M)\delta_k \leq (L+M)\frac{1}{2(L+M)} = \frac{1}{2}$$

and so $\rho_k \geq 1/2$. Consequently, by (1.4), (2.3), A3 and (2.20), we have

$$\begin{aligned} f(x_k) - f(x_{k+1}) \geq & \frac{1}{2}\left(m_k(0) - m_k(d_k)\right) \\ \geq & \frac{1}{4}\min\left\{\delta_k, \frac{1}{M}\right\}\|\nabla f(x_k)\|^2 \\ = & \frac{1}{4}\delta_k\|\nabla f(x_k)\|^2 \\ = & \frac{\|\nabla f(x_k)\|^2}{4b_k}. \end{aligned}$$

Hence, (2.18) is true. □

The next two lemmas provide upper bounds for the cardinalities of specific subsets of $I(k_i, k_{i+1}-1)$. For the sake of readability, the proofs are presented in the Appendix.

Lemma 4 *Given $\epsilon \in (0,1)$, let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1 such that (2.11) holds for some $T \geq 1$. Moreover, suppose that A1 and A3 are true. Given $i \in \{0, \ldots, |D_T|\}$ and $q \in I(k_i, k_{i+1}-1)$, if*

$$b_j < \tilde{L} \quad \text{for all} \quad j \in I(k_i, q), \tag{2.21}$$

with $\tilde{L}$ defined in (2.17), then

$$|I(k_i, q)| \leq \tilde{L}^2\epsilon^{-2} + 2. \tag{2.22}$$

Lemma 5 *Given $\epsilon \in (0,1)$, let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1 such that (2.11) holds for some $T \geq 1$, and suppose that A1–A3 are true. Moreover, given $i \in \{0, \ldots, |D_T|\}$, assume*

$$\mathcal{U} = \left\{j \in I(k_i, k_{i+1}-1) \ : \ b_j \geq \tilde{L}\right\} \neq \emptyset, \tag{2.23}$$

and let p be the smallest element of $\mathcal{U}$. If

$$k_i + 2 < k_{i+1} - 2 \quad \text{and} \quad p \in I(k_i, k_{i+1}-2),$$

then

$$\begin{aligned} |I(p, k_{i+1}-1)| \leq & 4\left[b_{k_i} + 19\tilde{L} + \frac{10\tilde{L}^2}{b_{\min}} + \left(\frac{4L^2 + 36b_{\min}^2 + 16b_{\min}L}{2b_{\min}^3}\right)\|\nabla f(x_0)\|^2\right. \\ & \left. + 16(f(x_{k_i}) - f_{low})\right]^2\epsilon^{-2} \end{aligned} \tag{2.24}$$

Combining Lemmas 4 and 5, we can get an upper bound for $|I(k_i, k_{i+1} - 1)|$ in terms of ϵ.

Lemma 6 *Given $\epsilon \in (0, 1)$, let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1 such that (2.11) holds for some $T \geq 1$, and suppose that A1-A3 are true. Then, given $i \in \{0, \ldots, |D_T|\}$ we have*

$$|I(k_i, k_{i+1} - 1)| \leq C_i \epsilon^{-2} \tag{2.25}$$

where

$$C_i = \tilde{L}^2 + 4 + 4\left[b_{k_i} + 19\tilde{L} + \frac{10\tilde{L}^2}{b_{\min}} + \left(\frac{4L^2 + 36b_{\min}^2 + 16b_{\min}L}{2b_{\min}^3}\right)\|\nabla f(x_0)\|^2 + 16(f(x_{k_i}) - f_{low})\right]^2. \tag{2.26}$$

Proof Let $\mathcal{U}$ be the set defined in (2.23). If $\mathcal{U} = \emptyset$, then

$$b_j < \tilde{L} \quad \text{for all} \quad j \in I(k_i, k_{i+1} - 1).$$

In this case, by Lemma 4 with $q = k_{i+1} - 1$, we have

$$|I(k_i, k_{i+1} - 1)| \leq \tilde{L}^2 \epsilon^{-2} + 2 \leq C_i \epsilon^{-2},$$

that is, (2.25) holds. Now, suppose that $\mathcal{U} \neq \emptyset$, and let $\hat{p}$ be the smallest element of $\mathcal{U}$. If $k_i + 2 \geq k_{i+1} - 2$, then $k_{i+1} - 1 \leq k_i + 3$ and so

$$|I(k_i, k_{i+1} - 1)| \leq 4 \leq C_i \epsilon^{-2}.$$

Thus, let us assume that $\mathcal{U} \neq \emptyset$ and $k_i + 2 < k_{i+1} - 2$. Under these conditions, the rest of the proof is reduced to two cases.

Case I $\hat{p} = k_{i+1} - 1$.

By the definition of $\hat{p}$ we have

$$b_j < \tilde{L} \quad \text{for all} \quad j \in I(k_i, \hat{p} - 1).$$

Then, by Lemma 4 with $q = \hat{p} - 1$ we get

$$|I(k_i, k_{i+1} - 1)| = |I(k_i, \hat{p} - 1)| + |\{\hat{p}\}| \leq \tilde{L}^2 \epsilon^{-2} + 3 \leq C_i \epsilon^{-2},$$

that is, (2.25) holds.

Case II $\hat{p} \in I(k_i, k_{i+1} - 2)$.

If $\hat{p} = k_i$, then by Lemma 5 (with $p = \hat{p}$) and (2.26) we have

$$|I(k_i, k_{i+1} - 1)| = |I(\hat{p}, k_{i+1} - 1)| \leq C_i \epsilon^{-2}.$$

On the other hand, if $\hat{p} \in I(k_i + 1, k_{i+1} - 2)$, then it follows from Lemma 4 (with $q = \hat{p} - 1$) and Lemma 5 (with $p = \hat{p}$) that

$$
\begin{aligned}
&|I(k_i, k_{i+1}-1)| \\
&\quad = |I(k_i, \hat{p}-1)| + |I(\hat{p}, k_{i+1}-1)| \\
&\quad \leq \tilde{L}^2\epsilon^{-2} + 2 + 4\left[b_{k_i} + 19\tilde{L} + \frac{10\tilde{L}^2}{b_{\min}} + \left(\frac{4L^2 + 36b_{\min}^2 + 16b_{\min}L}{2b_{\min}^3}\right)\|\nabla f(x_0)\|^2 + 16(f(x_{k_i}) - f_{low})\right]^2 \epsilon^{-2} \\
&\quad \leq C_i\epsilon^{-2}.
\end{aligned}
$$

□

Theorem 1 *Given $\epsilon \in (0,1)$, suppose that A1-A3 hold and let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1 such that*

$$\|\nabla f(x_k)\| > \epsilon \quad \text{for} \quad k = 0, 1, \ldots, T.$$

Then,

$$T < C_0\epsilon^{-2} \tag{2.27}$$

if $\alpha = 0$, and

$$T < \left(\max_{i=0,\ldots,|D_T|} C_i\right)\left(\frac{\log\left(\|\nabla f(x_0)\|\epsilon^{-1}\right)}{|\log(\alpha)|} + 1\right)\epsilon^{-2} \tag{2.28}$$

if $\alpha \in (0,1)$, where C_i is defined in (2.26).

Proof In view of Lemma 2 and Lemma 6, we have

$$
\begin{aligned}
T+1 \leq & \left(|D_T|+1\right) \max_{i=0,\ldots,|D_T|} |I(k_i, k_{i+1}-1)| \\
\leq & \left(|D_T|+1\right) \max_{i=0,\ldots,|D_T|} C_i\epsilon^{-2} \\
= & \left(\max_{i=0,\ldots,|D_T|} C_i\right)\left(|D_T|+1\right)\epsilon^{-2}.
\end{aligned} \tag{2.29}
$$

If $\alpha = 0$, it follows from Lemma 2 that $|D_T| = 0$, which in (2.29) gives

$$T < T + 1 \leq C_0\epsilon^{-2}.$$

Therefore, (2.27) is true. On the other hand, if $\alpha \in (0,1)$, it follows from Lemma 2 that

$$|D_T| < \frac{\log\left(\|\nabla f(x_0)\|\epsilon^{-1}\right)}{|\log(\alpha)|}$$

which in (2.29) gives

$$T < T + 1 \leq \left(\max_{i=0,\ldots,|D_T|} C_i\right)\left(\frac{\log\left(\|\nabla f(x_0)\|\epsilon^{-1}\right)}{|\log(\alpha)|} + 1\right)\epsilon^{-2}.$$

Thus, (2.28) is also true. □

The iteration complexity bound of $\mathcal{O}(\epsilon^{-2})$ obtained for Algorithm 1 with $\alpha = 0$ agrees in order with the complexity bounds obtained in [18–20] for other trust-region methods. As a consequence of Theorem 1 we have the following liminf-type convergence result for Algorithm 1.

Corollary 1 *Suppose that A1-A3 hold and let* $\{x_k\}_{k\geq 0}$ *be a sequence generated by Algorithm 1. Then there exists* $\hat{k}$ *such that* $\nabla f(x_{\hat{k}}) = 0$ *or*

$$\liminf_{k\to+\infty} \|\nabla f(x_k)\| = 0. \tag{2.30}$$

Proof If there exists $\hat{k}$ such that $\nabla f(x_{\hat{k}}) = 0$, then the statement holds. Assume now that

$$\nabla f(x_k) \neq 0 \tag{2.31}$$

for all k. Let $\epsilon_1 = 1/2$. By Theorem 1 there exists $k_1 \in \mathbb{N}$ such that

$$\|\nabla f(x_{k_1})\| \leq \epsilon_1. \tag{2.32}$$

Without loss of generality, we can assume that k_1 is the first iteration index for which (2.31) holds. Now, let

$$\epsilon_2 = \min\left\{\frac{1}{4}, \frac{1}{2}\|\nabla f(x_{k_1})\|\right\}.$$

Then $\epsilon_2 \in (0, \frac{1}{2}\epsilon_1]$. From Theorem 1 there exists $k_2 \in \mathbb{N}$ such that

$$\|\nabla f(x_{k_2})\| \leq \epsilon_2. \tag{2.33}$$

Moreover, since $\epsilon_2 < \|\nabla f(x_{k_1})\|$, we have $k_2 > k_1$. Again, we can assume that k_2 is the first iteration index for which (2.32) holds. Repeating this argument, we obtain a subsequence $\{x_{k_j}\}_{j\geq 1}$ of $\{x_k\}_{k\geq 0}$ such that

$$\|\nabla f(x_{k_j})\| \leq \epsilon_j, \tag{2.34}$$

where

$$\epsilon_j = \begin{cases} 1/2, & \text{for } j = 1, \\ \min\left\{\frac{1}{2j}, \frac{1}{2}\|\nabla f(x_{k_{j-1}})\|\right\}, & \text{for } j \geq 2. \end{cases}$$

Since $\lim_{j\to+\infty} \epsilon_j = 0$, by (2.34) we have

$$\lim_{j\to+\infty} \|\nabla f(x_{k_j})\| = 0.$$

Therefore, (2.30) is true. □

For the conservative variant of Algorithm 1, it is also possible to prove a lim-type convergence result.

Corollary 2 *Suppose that A1-A3 hold and let* $\{x_k\}_{k\geq 0}$ *be a sequence generated by Algorithm 1 choosing* $\alpha = 0$. *Then*

$$\lim_{k\to+\infty} \|\nabla f(x_k)\| = 0. \tag{2.35}$$

Proof We can split the proof in two cases. The first is when the sequence $\{b_k\}$ is bounded by $\tilde{L}$, and the second is when $b_k \geq \tilde{L}$ for some k, where $\tilde{L}$ is defined in Lemma 3.

Case I $b_k < \tilde{L}$ for all k.

Since $\alpha = 0$, it follows from (2.4) that

$$b_{j+1} - b_j = \frac{\|\nabla f(x_{j+1})\|^2}{b_j}.$$

Thus, given $k > 0$, we have

$$b_k - b_0 = \sum_{j=0}^{k-1}(b_{j+1} - b_j) = \sum_{j=0}^{k-1} \frac{\|\nabla f(x_{j+1})\|^2}{b_j}. \tag{2.36}$$

Since $b_k < \tilde{L}$ for all k, we conclude that

$$b_k - b_0 < \tilde{L} \text{ and } \frac{1}{b_j} > \frac{1}{\tilde{L}} \quad \text{for } j = 0, \ldots, k-1. \tag{2.37}$$

The equations (2.36) and (2.37) imply that:

$$\tilde{L} > \sum_{j=0}^{k-1} \frac{\|\nabla f(x_{j+1})\|^2}{b_j} > \sum_{j=0}^{k-1} \frac{\|\nabla f(x_{j+1})\|^2}{\tilde{L}}.$$

Thus,

$$\tilde{L}^2 > \sum_{j=0}^{k-1} \|\nabla f(x_{j+1})\|^2,$$

for any $k > 0$, and so

$$\sum_{j=0}^{+\infty} \|\nabla f(x_{j+1})\|^2 \leq \tilde{L}^2.$$

In particular,

$$\lim_{j\to\infty} \|\nabla f(x_j)\| = 0.$$

Case II $b_{k_0} \geq \tilde{L}$ for some k_0.

In this case, for any $k \geq k_0$ we have $b_k \geq \tilde{L}$, since b_k is increasing. Then, by Lemma 3, for all $k \geq k_0$ we have

$$f(x_{k+1}) \leq f(x_k) - \frac{\|\nabla f(x_k)\|^2}{4b_k}. \tag{2.38}$$

In particular, assuming that $\|\nabla f(x_k)\|$ is nonzero, then $\{f(x_k)\}$ is monotonically decreasing for $k \geq k_0$ and bounded from below, hence this sequence converges to some f^*. Suppose, by contradiction that (2.35) is not true. Then, there exists $\epsilon > 0$ for which the set $J_\epsilon = \{k \in \mathbb{N} : k \geq k_0 \text{ and } \|\nabla f(x_k)\| \geq \epsilon\}$ is infinite. Given $k \in J_\epsilon$, let $\ell_k \geq k$ be the first index such that

$$\|\nabla f(x_{\ell_k})\| \leq \frac{\epsilon}{2}.$$

The existence of ℓ_k is ensured by the Corollary 1. By A1 and $k \in J_\epsilon$ we have:

$$L\|x_k - x_{\ell_k}\| \geq \|\nabla f(x_k) - \nabla f(x_{\ell_k})\| \geq \|\nabla f(x_k)\| - \|\nabla f(x_{\ell_k})\| \geq \frac{\epsilon}{2}.$$

Then

$$\frac{\epsilon}{2L} \leq \|x_k - x_{\ell_k}\| \leq \sum_{j=k}^{\ell_k - 1} \|x_j - x_{j+1}\| \leq \sum_{j=k}^{\ell_k - 1} \frac{\|\nabla f(x_j)\|}{b_j}. \tag{2.39}$$

Since ℓ_k is the first index $j \geq k$ such that

$$\|\nabla f(x_j)\| \leq \frac{\epsilon}{2},$$

it follows that

$$\|\nabla f(x_j)\| > \frac{\epsilon}{2}, \tag{2.40}$$

for $j = k, \ldots, \ell_k - 1$. Thus, multiplying both sides of (2.39) by $\epsilon/2$ and using (2.40), we obtain

$$\frac{\epsilon^2}{4L} \leq \sum_{j=1}^{\ell_k - 1} \frac{\|\nabla f(x_j)\|^2}{b_j}. \tag{2.41}$$

Finally, combining (2.38) and (2.41), it follows that

$$f(x_k) - f(x_{\ell_k}) = \sum_{j=k}^{\ell_k - 1} (f(x_j) - f(x_{j+1})) \geq \sum_{j=k}^{\ell_k - 1} \frac{\|\nabla f(x_j)\|^2}{4b_j} \geq \frac{\epsilon^2}{16L}.$$

But from the convergence of $f(x_k)$, we conclude that

$$\lim_{k \to \infty} f(x_k) - f(x_{\ell_k}) = f^* - f^* = 0,$$

yielding to a contradiction. Therefore, (2.35) must be true. □

Recall that each iteration of Algorithm 1 requires only one gradient evaluation. Thus, it follows from Theorem 1 that the conservative variant of Algorithm 1 takes at most $\mathcal{O}(\epsilon^{-2})$ gradient evaluations to generate x_k such that $\|\nabla f(x_k)\| \leq \epsilon$, while the flexible variant takes at most $\mathcal{O}\big(|\log(\epsilon)|\epsilon^{-2}\big)$ gradient evaluations when $\max_{i=0,\ldots,|D_T|} C_i$ is bounded from above by a constant C independent of T and ϵ. It is possible to obtain such constant C under the following additional assumptions:

A4. The set $\mathcal{F}(x_0) = \big\{x \in \mathbb{R}^n \mid \|\nabla f(x)\| \leq \|\nabla f(x_0)\|\big\}$ is bounded.

A5. There exists $\hat{b}_{\max} > 0$ such that $\hat{b}_k \leq \hat{b}_{max}$ for all k, where $\hat{b}_k$ is used in (2.4).

Corollary 3 *Given $\epsilon \in (0,1)$, suppose that A1-A5 hold and let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1 (with $\alpha \neq 0$) such that*

$$\|\nabla f(x_k)\| > \epsilon \quad \text{for} \quad k = 0, 1, \ldots, T.$$

Moreover, let

$$\tilde{f} = \sup\big\{f(x) \mid x \in \mathcal{F}(x_0)\big\}.$$

Then,

$$T < C\left(\frac{\log\left(\|\nabla f(x_0)\|\epsilon^{-1}\right)}{|\log(\alpha)|} + 1\right)\epsilon^{-2} \tag{2.42}$$

where

$$C = \tilde{L}^2 + 4 + 4\left[b_0 + \hat{b}_{\max} + 19\tilde{L} + \frac{10\tilde{L}^2}{b_{\min}} + \left(\frac{4L^2 + 36b_{\min}^2 + 16b_{\min}L}{2b_{\min}^3}\right)\|\nabla f(x_0)\|^2 + 16(\tilde{f} - f_{low})\right]^2. \tag{2.43}$$

Proof Let $i \in \{0, \ldots, |D_T|\}$. By Lemma 1 we have

$$\|\nabla f(x_{k_i})\| \leq \|\nabla f(x_0)\|.$$

Thus, $x_{k_i} \in \mathcal{F}(x_0)$ and so

$$f(x_{k_i}) \leq \tilde{f}.$$

Moreover, $b_{k_0} = b_0$, and by (2.4) we also have

$$b_{k_i} = \hat{b}_{k_i - 1} \leq \hat{b}_{\max},$$

for $i \geq 1$. Thus, it follows from (2.26) and (2.43) that $C_i \leq C$. Since i was chosen arbitrarily in $\{0, \ldots, |D_T|\}$, we have

$$\max_{i=0,\ldots,|D_T|} C_i \leq C,$$

and then (2.42) follows directly from Theorem 1. □

Notice that $|\log(\epsilon)|$ grows slowly when ϵ approaches to zero. In particular, for the values of ϵ used in practice (such as 10^{-4}), $|\log(\epsilon)|\epsilon^{-2}$ differs from ϵ^{-2} only by a small constant factor.

3 Numerical results

We performed numerical experiments comparing the following Octave implementations:

- **TR**: the standard trust-region method described in the Introduction, with $\eta_1 = 10^{-4}$ and $\eta_2 = 0.25$.
- **FYTR**: the Fan-Yuan trust-region method [13], which differs from TR only by the update rule for the trust-region radius. Specifically, it uses

$$\Delta_k = \delta_k \|\nabla f(x_k)\|,$$

  where δ_k is updated by the rule

$$\delta_{k+1} = \begin{cases} 6\delta_k, & \text{if } \rho_k \geq 0.25 \text{ and } \|d_k\| > \frac{1}{2}\Delta_k, \\ \frac{1}{6}\delta_k, & \text{if } \rho_k < 0.25, \\ \delta_k, & \text{otherwise.} \end{cases}$$

- **AdaTrust1**: the conservative variant of Algorithm 1 ($\alpha = 0$) with $b_{\min} = 10^{-4}$.
- **AdaTrust2**: the flexible variant of Algorithm 1 with $b_{\min} = 10^{-4}$, $\alpha = 0.9$ and, in (2.4),

$$\hat{b}_k = \begin{cases} \min\left\{\hat{b}_{\max}, \max\left\{b_{\min}, \dfrac{b_k}{2}\right\}\right\}, & \text{if } \|d_k\| > \dfrac{1}{2}\Delta_k, \\ \min\left\{\hat{b}_{\max}, b_k\right\}, & \text{otherwise,} \end{cases}$$

  with $\hat{b}_{\max} = \|\nabla f(x_0)\|$.

In all implementations the initial trust-region radius is $\Delta_0 = 1$, which corresponds to the choices $\delta_0 = \frac{1}{\|\nabla f(x_0)\|}$ for FYTR and $b_0 = \|\nabla f(x_0)\|$ for AdaTrust1 and AdaTrust2. The trust-region subproblems are approximately solved using the Steihaug-Toint method [30, 32], while B_k is computed using the BFGS update whenever it is possible, namely

$$B_{k+1} = \begin{cases} B_k + \dfrac{y_k y_k^T}{s_k^T y_k} - \dfrac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}, & \text{if } s_k^T y_k > 0, \\ B_k, & \text{otherwise,} \end{cases}$$

where $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ and $B_0 = I$. All experiments were performed with Octave, version 6.1.0, on a PC with microprocessor Intel(R) Core(TM) i7-8565 (1.99 GHz) and 8 GB of RAM memory.

3.1 Moré-Garbow-Hillstrom test problems

In our first test, we applied the referred codes to the set of 35 problems from [25] of the form

$$\min_{x \in \mathbb{R}^n} f(x) \equiv \sum_{i=1}^{\hat{m}} f_i(x)^2.$$

The choices for n and $\hat{m}$ for each problem were the same used in [6]. First, we used the stopping criterion

$$\|\nabla f(x_k)\| \leq \epsilon \equiv 10^{-4}, \tag{3.1}$$

allowing a maximum of 10, 000 calls of the oracle for each solver, where each call of the oracle corresponds to either one function evaluation or one gradient evaluation. Figure 1 presents the performance profiles [11][1] in terms of the total number of calls of the oracle required by each solver to generate the first iterate x_k satisfying

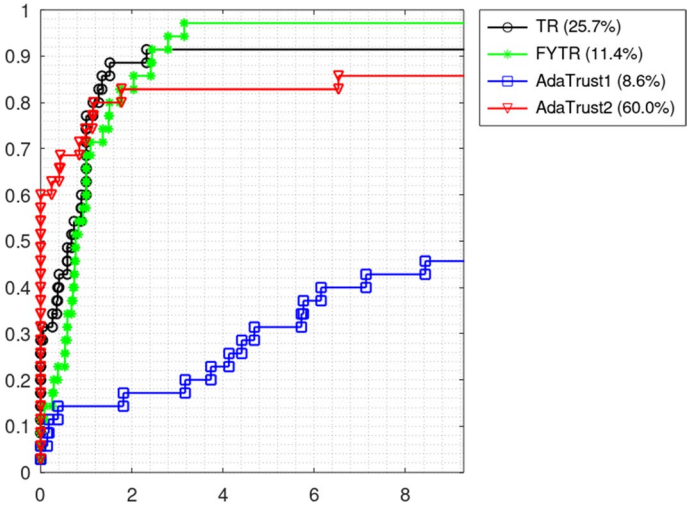

Fig. 1 Evaluation performance profiles ($\log_2$ scale) for the stopping criterion (3.1). For each code, the caption also indicates the percentage of problems in which the code was the best in terms of the number of calls of the oracle

[1] The performance profiles were generated using the code **perf.m** freely available in the website http://www.mcs.anl.gov/~more/cops/.

(3.1). Remarkably, even without using function evaluations, AdaTrust2 was the most efficient solver, requiring less calls of the oracle in 60% of the problems. Moreover, the significant superiority of AdaTrust2 in comparison with AdaTrust1 suggests that the nonmonotone behaviour of $\{\delta_k\}$ (allowed by $\alpha > 0$) is very beneficial. Recall that, by Theorem 1, the complexity bound obtained for the flexible variant of Algorithm 1 (AdaTrust2) is slightly worse than the bound obtained for the conservative variant of Algorithm 1 (AdaTrust1). Thus, our numerical results also indicate that a method with worst-case complexity of $\mathcal{O}(|\log(\epsilon)|\epsilon^{-2})$ is not necessarily worse in practice than a method with complexity of $\mathcal{O}(\epsilon^{-2})$. In fact, the worst-case behavior is usually very pessimistic with respect to the practical performance of the methods. It is also worth noticing that AdaTrust2 was less robust than TR and FYTR in this set of test problems. Specifically, AdaTrust2 failed to find an ϵ-approximate stationary point (with $\epsilon = 10^{-4}$) for problems `Powell badly scaled`, `Brown badly scaled`, `Meyer`, `Osborne 1` and `Chebyquad`[2].

Since Algorithm 1 relies only on gradient evaluations, it does not impose the monotonicity of $\{f(x_k)\}$. Figure 2 shows the nonmonotone behaviour of this sequence for $\{x_k\}$ generated by AdaTrust2 on the problem `Broyden tridiagonal`.

To evaluate the ability of the codes to find points with small function value, we performed another experiment replacing (3.1) by the stopping criterion

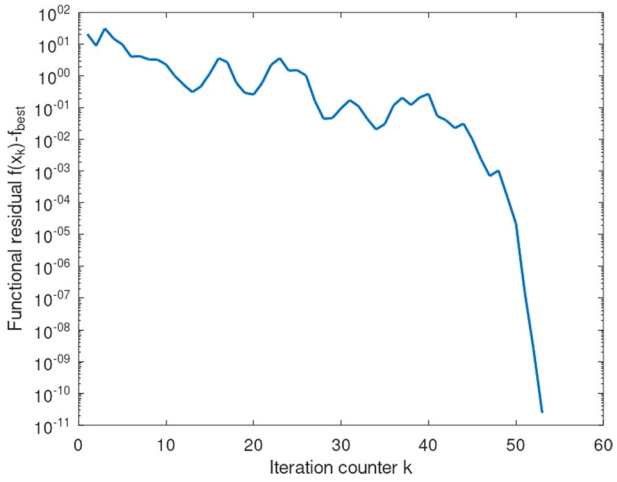

Fig. 2 Functional residuals obtained by AdaTrust2 on problem `Broyden tridiagonal`

[2] Looking closely problems `Powell badly scaled`, `Brown badly scaled` and `Meyer`, we see that for these problems $\|\nabla f(x_0)\| \geq 10^3$. Due to (2.4) and (2.2), large values for $\|\nabla f(x_0)\|$ make Δ_k become extremely small very quickly, which severely slows down the progress of the iterates towards stationary points. This remark suggests that when initializing AdaTrust2, starting points with very large norm of the gradient should be avoided.

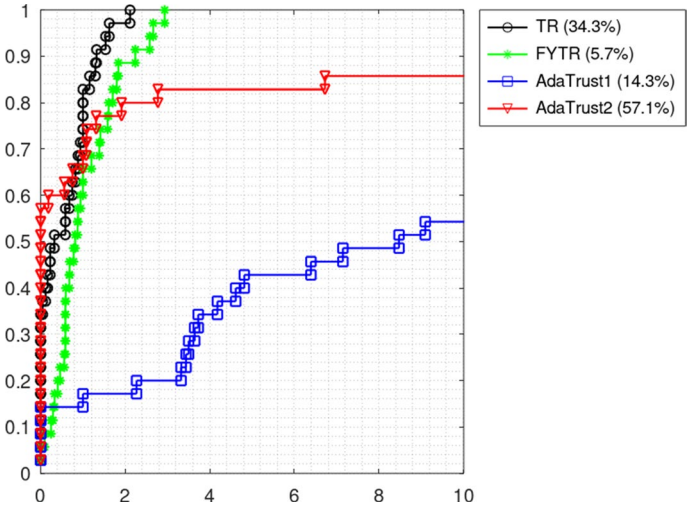

Fig. 3 Evaluation performance profiles ($\log_2$ scale) for the stopping criterion (3.2)

$$\frac{f(x_k) - f_{best}}{\max\left\{1, |f_{best}|\right\}} \leq \epsilon. \tag{3.2}$$

For each problem, f_{best} is the smallest value of the objective function obtained by applying the four codes to the corresponding problem with a budget of 10,000 calls of oracle for each solver.

Figure 3 presents the performance profiles in terms of the numbers of calls of the oracle required by each solver to generate the first iterate x_k satisfying (3.2). As we can see, AdaTrust2 also was the most efficient solver in this case, requiring less calls of the oracle in 57.1% of the problems.

3.2 Nonconvex logistic regression problems

In our second test, we considered nonconvex logistic regression problems of the form

$$\min_{x \in \mathbb{R}^{n+1}} f_\mu(x) := -\sum_{i=1}^{m} \left[b^{(i)} \log\left(m_x(a^{(i)})\right) + (1 - b^{(i)}) \log\left(1 - m_x(a^{(i)})\right)\right] + \mu\psi(x),$$

where $\mu > 0$, $\left\{(a^{(i)}, b^{(i)})\right\}_{i=1}^{m} \subset \mathbb{R}^{n+1} \times \{0, 1\}$ is the dataset (with $a_1^{(i)} = 1$ for $i = 1, \ldots, m$), $m_x(a) := 1/\left(1 + e^{-\langle a, x\rangle}\right)$ is the logistic model, and

$$\psi(x) := \sum_{j=1}^{n+1} \frac{x_j^2}{(1 + x_j^2)}$$

Table 1 Datasets details

Dataset	n	m
Iris [14]	4	150
Breast Cancer Wisconsin [31]	9	683
Wine [2]	12	178
Sonar [17]	60	208
Phishing [1]	9	1353
Ionosphere [29]	34	351
Diabetes [28]	8	768
Musk [9]	15	476
Seeds [7]	7	210
Bank Note Authentication [12]	4	1372

is a nonconvex regularizer. Specifically, we considered $\mu = 5$ and ten datasets obtained from [12] and [28], whose details are given in Table 1. For each dataset, we considered the following choices for the starting points:

$$x_0^{(1)} = \begin{bmatrix} -1 & -1 & \dots & -1 \end{bmatrix}^T, \quad x_0^{(2)} = \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}^T \quad \text{and} \quad x_0^{(3)} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T.$$

Each pair (dataset, starting point) is regarded as a test problem, which results in 30 problems.

In our first experiment we used the stopping criterion (3.1), allowing a maximum of 4,000 calls of the oracle for each solver. As we can see in Fig. 4, AdaTrust2 was the most efficient solver, requiring less calls of the oracle in about 86.6% of the problems. Moreover, for all test problems, AdaTrust2 was able to

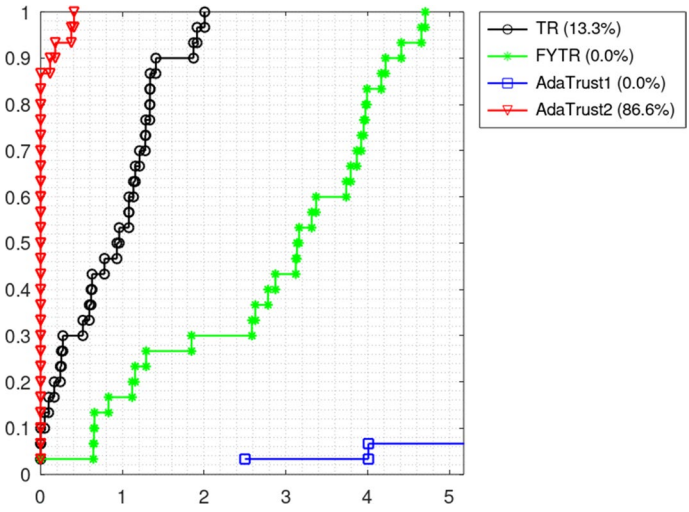

Fig. 4 Evaluation performance profiles ($\log_2$ scale) for the stopping criterion (3.1)

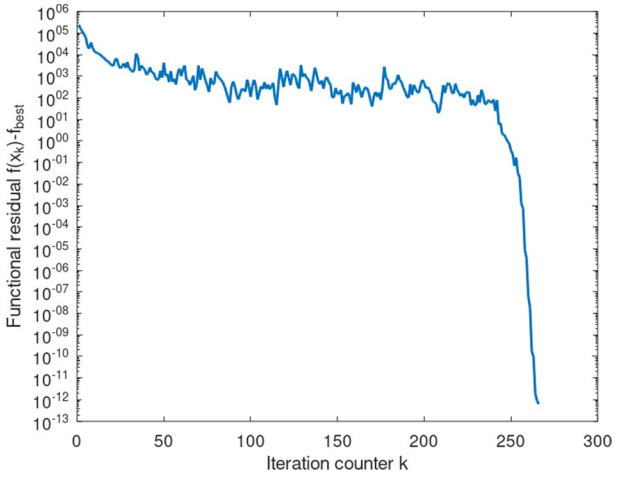

Fig. 5 Functional residuals obtained by AdaTrust2 on problem $(\texttt{Musk}, x_0^{(1)})$

generate a point satisfying (3.1). Figure 5 shows the nonmonotone behaviour of the sequence $\{f(x_k)\}$ for $\{x_k\}$ generated by AdaTrust2 on the problem $(\texttt{Musk}, x_0^{(1)})$.

In our second experiment we used the stopping criterion (3.2), allowing a maximum of 4,000 calls of the oracle. Now, for each problem, f_{best} in (3.2) is the smallest value of the objective function obtained by applying the four solvers to the corresponding problem with a budget of 4,000 calls of the oracle.

Figure 6 presents the performance profile in terms of the number of calls of the oracle. Again, AdaTrust2 was the most efficient solver. It is worth noticing the

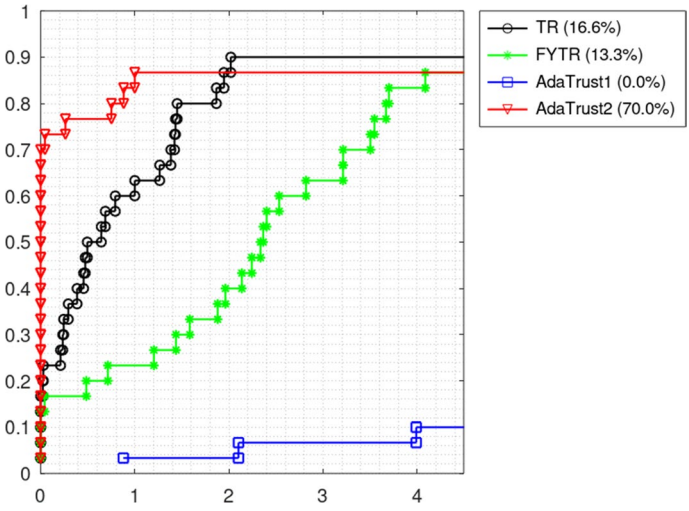

Fig. 6 Evaluation performance profiles ($\log_2$ scale) for the stopping criterion (3.2)

poor performance of AdaTrust1, which highlights the importance of using $\alpha \neq 0$ in Algorithm 1.

3.3 Gradient evaluations more expensive than function evaluations

For both sets of test problems considered in our experiments, the cost of computing gradient is similar to the cost of evaluating function values. However, in some cases, gradient evaluations may be more expensive than function evaluations. In particular, when gradients are computed using reverse mode Automatic Differentiation [21], each gradient evaluations has a cost similar to three function evaluations [4, 5]. To simulate this situation, we generated new performance profiles for both sets of test problems considering

$$CO = FE + 3GE,$$

where CO is the number of calls of the oracle needed to find an ϵ-stationary point, and FE and GE are the corresponding numbers of function evaluations and gradient evaluations, respectively. The resulting graphs are shown in Figs. 7 and 8.

These figures show that in both sets of test problems AdaTrust2 is more efficient than FYTR and is competitive with TR, even if we assume that the cost of each gradient evaluation is equal to the cost of three function evaluations.

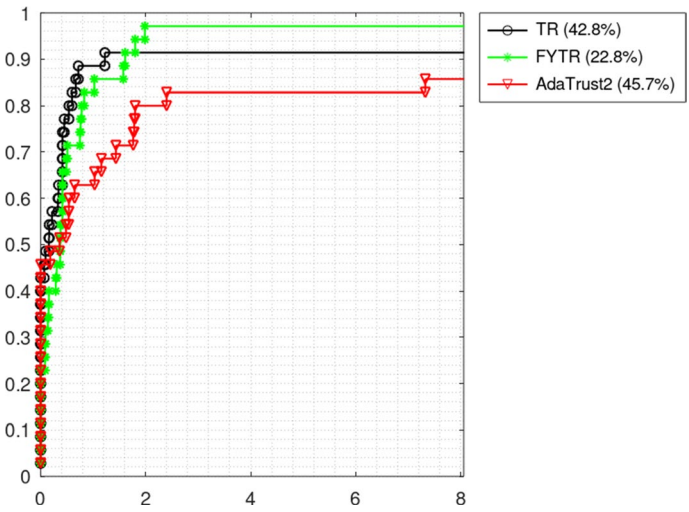

Fig. 7 Evaluation performance profiles ($\log_2$ scale) for problems from [25], using the stopping criterion (3.1)

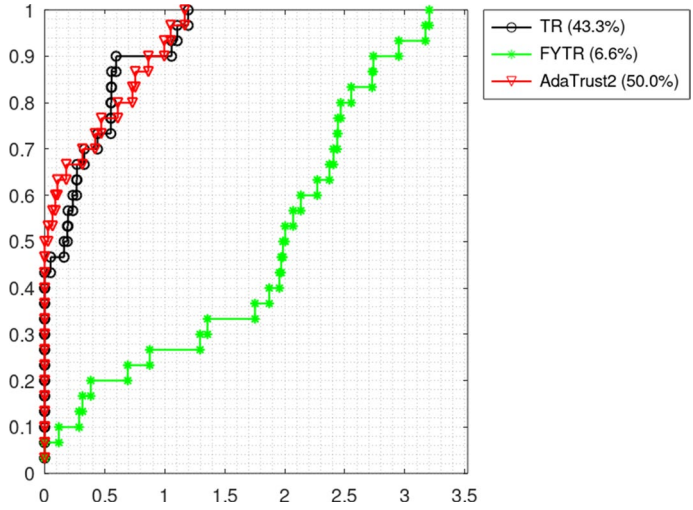

Fig. 8 Evaluation performance profiles ($\log_2$ scale) for the nonconvex logistic regression problems, using the stopping criterion (3.1)

4 Conclusion

In this paper, we proposed an adaptive trust-region method (AdaTrust) for smooth unconstrained optimization. As in the Fan-Yuan trust-region method [13], the trust-region radius at the kth iteration is defined by $\Delta_k = \delta_k \|\nabla f(x_k)\|$. However, our update rule for δ_k relies only on gradient evaluations. The general AdaTrust scheme admits a conservative variant and a flexible variant. In the conservative variant, sequence $\{\delta_k\}$ is monotonically decreasing, while in the flexible variant $\{\delta_k\}$ is allowed to be non-monotone. Assuming that the objective function is bounded from below and that its gradient is Lipschitz continuous, we show that the conservative variant of AdaTrust takes at most $\mathcal{O}(\epsilon^{-2})$ gradient evaluations to generate x_k such that $\|\nabla f(x_k)\| \leq \epsilon$ for a given $\epsilon \in (0, 1)$, while the flexible variant takes at most $\mathcal{O}(|\log(\epsilon)|\epsilon^{-2})$ gradient evaluations. Using this worst-case complexity result, we also proved a liminf-type global convergence result. Finally, we reported numerical results in which a flexible variant of AdaTrust compared favorably in comparison with the standard trust-region method and the Fan-Yuan trust-region method in terms of the number of calls of the oracle required to find points with small norm of the gradient and points with small function value. As a topic for future research, it would be interesting to investigate the development of an stochastic version of AdaTrust for the minimization of finite sums.

Appendix

Proof of Lemma 4

By definition, $k_i \leq q$. If $k_i \geq q - 1$, then $|I(k_i, q)| \leq 2$ and so (2.22) holds. Now, suppose that $k_i < q - 1$. By (2.4) we have

$$b_{j+1} - b_j = \frac{\|\nabla f(x_{j+1})\|^2}{b_j} \quad \text{for} \quad j = k_i, \ldots, q-1.$$

Summing up these equalities, it follows from (2.11) and (2.21) that

$$b_q - b_{k_i} = \sum_{j \in I(k_i, q-1)} \frac{\|\nabla f(x_{j+1})\|^2}{b_j} > |I(k_i, q-1)| \tilde{L}^{-1} \epsilon^2,$$

and so

$$|I(k_i, q-1)| < \tilde{L}\big(b_q - b_{k_i}\big)\epsilon^{-2} < \tilde{L} b_q \epsilon^{-2}.$$

Since $b_q < \tilde{L}$, we obtain

$$|I(k_i, q-1)| < \tilde{L}^2 \epsilon^{-2},$$

which gives

$$|I(k_i, q)| < \tilde{L}^2 \epsilon^{-2} + 1,$$

Therefore, (2.22) also holds in this case. □

Proof of Lemma 5

Let $k \in I(p+1, k_{i+1} - 1)$. Then, by (2.4), $b_j \geq \tilde{L}$ for $j = p, \ldots, k-1$. Consequently, by Lemma 3, we have

$$f(x_j) - f(x_{j+1}) \geq \frac{\|\nabla f(x_j)\|^2}{4 b_j} \quad \text{for all} \quad j = p, \ldots, k-1.$$

Summing up these inequalities we get

$$f(x_p) - f(x_k) \geq \frac{1}{4} \sum_{j=p}^{k-1} \frac{\|\nabla f(x_j)\|^2}{b_j}, \tag{4.1}$$

and so, by A2,

$$\sum_{j=p}^{k-1} \frac{\|\nabla f(x_j)\|^2}{b_j} \leq 4\big(f(x_p) - f(x_k)\big) \leq 4\big(f(x_p) - f_{low}\big). \tag{4.2}$$

On the other hand, by (2.4) and A1, we also have

$$\begin{aligned} b_k - b_p = \sum_{j=p}^{k-1} \left(b_{j+1} - b_j\right) = \sum_{j=p}^{k-1} \frac{\|\nabla f(x_{j+1})\|^2}{b_j} \\ \leq \sum_{j=p}^{k-1} \frac{\left(\|\nabla f(x_j) - \nabla f(x_{j+1})\| + \|\nabla f(x_j)\|\right)^2}{b_j} \\ \leq 2 \sum_{j=p}^{k-1} \frac{\|\nabla f(x_j) - \nabla f(x_{j+1})\|^2 + \|\nabla f(x_j)\|^2}{b_j} \\ \leq 2 \sum_{j=p}^{k-1} \frac{L^2 \|x_j - x_{j+1}\|^2 + \|\nabla f(x_j)\|^2}{b_j}. \end{aligned} \tag{4.3}$$

By (2.2),

$$\|x_j - x_{j+1}\| = \|d_j\| \leq \delta_j \|\nabla f(x_j)\| = \frac{\|\nabla f(x_j)\|}{b_j}. \tag{4.4}$$

Then, combining (4.3) and (4.4) and using $b_j \geq \tilde{L}$, it follows that

$$\begin{aligned} b_k - b_p \leq 2 \sum_{j=p}^{k-1} \frac{L^2 \|\nabla f(x_j)\|^2}{b_j^3} + 2 \sum_{j=p}^{k-1} \frac{\|\nabla f(x_j)\|^2}{b_j} \\ \leq 2 \sum_{j=p}^{k-1} \frac{\tilde{L}^2 \|\nabla f(x_j)\|^2}{b_j^2 b_j} + 2 \sum_{j=p}^{k-1} \frac{\|\nabla f(x_j)\|^2}{b_j} \\ \leq 4 \sum_{j=p}^{k-1} \frac{\|\nabla f(x_j)\|^2}{b_j}. \end{aligned} \tag{4.5}$$

Now, combining (4.5) and (4.2), we obtain

$$b_k - b_p \leq 16\big(f(x_p) - f_{low}\big),$$

and so

$$b_k \leq b_p + 16\big(f(x_p) - f_{low}\big), \quad \forall k \in I(p, k_{i+1} - 1). \tag{4.6}$$

Our first goal is to refine the upper bound for b_k in (4.6). For that, we will break the analysis into a few cases and subcases related with the position of p in the set $I(k_i, k_{i+1} - 2)$.

Case I $p = k_i$.
In this case, it follows from (4.6) that

$$b_k \leq b_{k_i} + 16(f(x_{k_i}) - f_{low}) \quad \forall k \in I(p, k_{i+1} - 1). \tag{4.7}$$

Case II $p \in I(k_i + 1, k_{i+1} - 2)$.
By A1 and the trust-region constraint, we have

$$\begin{aligned} b_p =& b_{p-1} + \frac{\|\nabla f(x_p)\|^2}{b_{p-1}} \\ \leq& b_{p-1} + \frac{\left(\|\nabla f(x_{p-1}) - \nabla f(x_p)\| + \|\nabla f(x_{p-1})\|\right)^2}{b_{p-1}} \\ \leq& b_{p-1} + 2\frac{\|\nabla f(x_{p-1}) - \nabla f(x_p)\|^2 + \|\nabla f(x_{p-1})\|^2}{b_{p-1}} \\ \leq& b_{p-1} + 2\frac{L^2\|x_{p-1} - x_p\|^2 + \|\nabla f(x_{p-1})\|^2}{b_{p-1}} \\ \leq& b_{p-1} + \frac{2L^2\|\nabla f(x_{p-1})\|^2}{b_{p-1}^3} + \frac{2\|\nabla f(x_{p-1})\|^2}{b_{p-1}}. \end{aligned} \tag{4.8}$$

Moreover, given $j \in I(k_i, p-1)$, we also have

$$\begin{aligned} f(x_{j+1}) \leq& f(x_j) + \nabla f(x_j)^T(x_{j+1} - x_j) + \frac{L}{2}\|x_{j+1} - x_j\|^2 \\ \leq& f(x_j) + \|\nabla f(x_j)\|\|d_j\| + \frac{L}{2}\|d_j\|^2 \\ \leq& f(x_j) + \frac{\|\nabla f(x_j)\|^2}{b_j} + \frac{L}{2b_j}\frac{\|\nabla f(x_j)\|^2}{b_j} \\ =& f(x_j) + \left(1 + \frac{L}{2b_j}\right)\frac{\|\nabla f(x_j)\|^2}{b_j} \\ \leq& f(x_j) + \left(1 + \frac{L}{2b_{\min}}\right)\frac{\|\nabla f(x_j)\|^2}{b_j}. \end{aligned} \tag{4.9}$$

Case II(a) $p = k_i + 1$.
In this case, by (4.8) we have

$$b_p \leq b_{k_i} + \frac{2L^2\|\nabla f(x_{k_i})\|^2}{b_{k_i}^3} + \frac{2\|\nabla f(x_{k_i})\|^2}{b_{k_i}}. \tag{4.10}$$

On the other hand, by (4.9) with $j = k_i$, we get

$$f(x_p) \leq f(x_{k_i}) + \left(1 + \frac{L}{2b_{\min}}\right)\frac{\|\nabla f(x_{k_i})\|^2}{b_{k_i}}. \tag{4.11}$$

Thus, combining (4.6), (4.10) and (4.11), it follows that

$$\begin{aligned} b_k \leq & b_{k_i} + \frac{2L^2\|\nabla f(x_{k_i})\|^2}{b_{k_i}^3} + \frac{2\|\nabla f(x_{k_i})\|^2}{b_{k_i}} \\ & + 16\left[\left(1 + \frac{L}{2b_{\min}}\right)\frac{\|\nabla f(x_{k_i})\|^2}{b_{k_i}} + f(x_{k_i}) - f_{low}\right] \\ = & b_{k_i} + \left(\frac{4L^2 + 36b_{\min}^2 + 16b_{min}L}{2b_{\min}^3}\right)\|\nabla f(x_{k_i})\|^2 + 16\big(f(x_{k_i}) - f_{low}\big) \end{aligned} \tag{4.12}$$

for all $k \in I(p, k_{i+1} - 1)$.

Case II(b) $p \in I(k_i + 2, k_{i+1} - 2)$.

In this case, $b_{p-1} \geq b_{p-2}$ and so, by (4.8), we get

$$\begin{aligned} b_p \leq & b_{p-1} + \frac{2L^2\|\nabla f(x_{p-1})\|^2}{b_{p-1}^2 b_{p-2}} + \frac{2\|\nabla f(x_{p-1})\|^2}{b_{p-2}} \\ = & b_{p-1} + \frac{2L^2(b_{p-1} - b_{p-2})}{b_{p-1}^2} + \frac{2\|\nabla f(x_{p-1})\|^2}{b_{p-2}} \\ < & b_{p-1} + \frac{2L^2}{b_{p-1}} + 2(b_{p-1} - b_{p-2}) \\ < & 3b_{p-1} + \frac{2L^2}{b_{p-1}}. \end{aligned}$$

Since $b_{\min} \leq b_{p-1} < \tilde{L}$, it follows that

$$b_p \leq 3\tilde{L} + \frac{2L^2}{b_{\min}}. \tag{4.13}$$

On the other hand, by (4.9) we have

$$
\begin{aligned}
f(x_p) - f(x_{k_i}) &= \sum_{j=k_i}^{p-1} f(x_{j+1}) - f(x_j) \\
&\leq \left(1 + \frac{L}{2b_{\min}}\right) \sum_{j=k_i}^{p-1} \frac{\|\nabla f(x_j)\|^2}{b_j} \\
&= \left(1 + \frac{L}{2b_{\min}}\right) \frac{\|\nabla f(x_{k_i})\|^2}{b_{k_i}} + \left(1 + \frac{L}{2b_{\min}}\right) \sum_{j=k_i+1}^{p-1} \frac{\|\nabla f(x_j)\|^2}{b_j} \\
&\leq \left(1 + \frac{L}{2b_{\min}}\right) \frac{\|\nabla f(x_{k_i})\|^2}{b_{k_i}} + \left(1 + \frac{L}{2b_{\min}}\right) \sum_{j=k_i+1}^{p-1} \frac{\|\nabla f(x_j)\|^2}{b_{j-1}} \\
&= \left(1 + \frac{L}{2b_{\min}}\right) \frac{\|\nabla f(x_{k_i})\|^2}{b_{k_i}} + \left(1 + \frac{L}{2b_{\min}}\right) \sum_{j=k_i+1}^{p-1} (b_j - b_{j-1}) \\
&= \left(1 + \frac{L}{2b_{\min}}\right) \frac{\|\nabla f(x_{k_i})\|^2}{b_{k_i}} + \left(1 + \frac{L}{2b_{\min}}\right) (b_{p-1} - b_{k_i}) \\
&< \left(1 + \frac{L}{2b_{\min}}\right) \frac{\|\nabla f(x_{k_i})\|^2}{b_{k_i}} + \left(1 + \frac{L}{2b_{\min}}\right) b_{p-1} \\
&< \left(1 + \frac{L}{2b_{\min}}\right) \frac{\|\nabla f(x_{k_i})\|^2}{b_{k_i}} + \left(1 + \frac{L}{2b_{\min}}\right) \tilde{L}.
\end{aligned}
$$

and so

$$
f(x_p) \leq f(x_{k_i}) + \left(1 + \frac{L}{2b_{\min}}\right) \frac{\|\nabla f(x_{k_i})\|^2}{b_{\min}} + \left(1 + \frac{L}{2b_{\min}}\right) \tilde{L}. \tag{4.14}
$$

Thus, combining (4.6), (4.13) and (4.14), it follows that

$$
\begin{aligned}
b_k &\leq 3\tilde{L} + \frac{2L^2}{b_{\min}} + 16 \left[\left(1 + \frac{L}{2b_{\min}}\right) \frac{\|\nabla f(x_{k_i})\|^2}{b_{\min}} + \left(1 + \frac{L}{2b_{\min}}\right) \tilde{L} + f(x_{k_i}) - f_{low} \right] \\
&\leq 19\tilde{L} + \frac{10\tilde{L}^2}{b_{\min}} + 16 \left(1 + \frac{L}{2b_{\min}}\right) \frac{\|\nabla f(x_{k_i})\|^2}{b_{\min}} + 16(f(x_{k_i}) - f_{low}),
\end{aligned} \tag{4.15}
$$

for all $k \in I(p, k_{i+1} - 1)$.

Summarizing all cases and subcases above, it follows from (4.7), (4.12), and (4.15) that

$$
b_k \leq b_{k_i} + 19\tilde{L} + \frac{10\tilde{L}^2}{b_{\min}} + \left(\frac{4L^2 + 36b_{\min}^2 + 16b_{\min}L}{2b_{\min}^3} \right) \|\nabla f(x_{k_i})\|^2 + 16(f(x_{k_i}) - f_{low}) \tag{4.16}
$$

for all $k \in I(p, k_{i+1}-1)$, regardless of the position of p in the set $I(k_i, k_{i+1}-2)$. Finally, by (2.4) and Lemma 3,

$$f(x_j) - f(x_{j+1}) \geq \frac{\|\nabla f(x_j)\|^2}{4b_j} \quad \text{for } j = p, \ldots, k_{i+1}-1.$$

Summing up these inequalities, it follows from A2, (2.11) and (4.16) that

$$\begin{aligned} f(x_p) - f_{low} \geq & f(x_p) - f(x_{k_{i+1}}) \\ \geq & \sum_{j=p}^{k_{i+1}-1} \frac{\|\nabla f(x_j)\|^2}{4b_j} \\ \geq & \frac{|I(p, k_{i+1}-1)|\epsilon^2}{4\left[b_{k_i} + 19\tilde{L} + \frac{10\tilde{L}^2}{b_{\min}} + \left(\frac{4L^2 + 36b_{\min}^2 + 16b_{\min}L}{2b_{\min}^3}\right)\|\nabla f(x_{k_i})\|^2 + 16(f(x_{k_i}) - f_{low})\right]} \end{aligned} \tag{4.17}$$

By (4.11) and (4.14), we also have

$$\begin{aligned} f(x_p) - f_{low} \leq & \left(1 + \frac{L}{2b_{\min}}\right)\tilde{L} + \left(1 + \frac{L}{2b_{\min}}\right)\frac{\|\nabla f(x_{k_i})\|^2}{b_{\min}} + (f(x_{k_i}) - f_{low}) \\ \leq & \tilde{L} + \frac{\tilde{L}^2}{2b_{\min}} + \left(1 + \frac{L}{2b_{\min}}\right)\frac{\|\nabla f(x_{k_i})\|^2}{b_{\min}} + (f(x_{k_i}) - f_{low}) \\ \leq & b_{k_i} + 19\tilde{L} + \frac{10\tilde{L}^2}{b_{\min}} + \left(\frac{4L^2 + 36b_{\min}^2 + 16b_{\min}L}{2b_{\min}^3}\right)\|\nabla f(x_{k_i})\|^2 + 16(f(x_{k_i}) - f_{low}). \end{aligned} \tag{4.18}$$

Thus, combining (4.17), (4.18) and using $\|\nabla f(x_{k_i})\| \leq \|\nabla f(x_0)\|$ (by Lemma 1), we get

$$|I(p, k_{i+1}-1)| \leq 4\left[b_{k_i} + 19\tilde{L} + \frac{10\tilde{L}^2}{b_{\min}} + \left(\frac{4L^2 + 36b_{\min}^2 + 16b_{\min}L}{2b_{\min}^3}\right)\|\nabla f(x_{k_i})\|^2 + 16(f(x_{k_i}) - f_{low})\right]^2 \epsilon^{-2}.$$

□

Acknowledgements The authors are very grateful to the two anonymous referees, whose comments helped to improve the paper.

Funding G. N. Grapiglia was partially supported by the National Council for Scientific and Technological Development (CNPq) - Brazil (Grant 312777/2020-5). G.F.D. Stella was supported by the Coordination for the Improvement of Higher Education Personnel (CAPES) - Brazil.

Data availability The data used to form the test problems in Subsection 3.2 are freely available in [12, 28].

References

1. Abdelhamid, N., Ayesh, A., Thabtah, F.: Phishing detection based Associative Classification data mining. Expert Syst. Appl. **41**, 5948–5959 (2014)

2. Aeberhard, S., Coomans, D., de Vel, O.: Comparison of Classifiers in High Dimensional Settings, Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland
3. Balima, O., Boulanger, J., Charette, A., Marceau, D.: New developments in frequency domain optical tomography. Part II: application with a L-BFGS associated to an inexact line search. J. Quant. Spectrosc. Radiat. Transf. **112**, 1235–1240 (2011)
4. Bartholomew-Biggs, M., Brown, S., Christianson, B., Dixon, L.: Automatic differentiation of algorithms. J. Comput. Appl. Math. **124**, 171–190 (2000)
5. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: a survey. J. Mach. Learn. Res. **18**, 1–43 (2018)
6. Birgin, E.G., Gardenghi, J.L., Martínez, J.M., Santos, S.A.: On the use of third-order models with fourth-order regularization for unconstrained optimization. Optim. Lett. **14**, 815–838 (2020)
7. Charytanowicz, M., Niewczas, J., Kulczycki, P., Kowalski, P.A., Lukasik, S., Zak, S.: A complete gradient clustering algorithm for features analysis of X-ray images. In: Pietka, E., Kawa, J. (eds.) Information Technologies in Biomedicine, pp. 15–24. Springer, Berlin (2010)
8. Conn, A.R., Gould, N.I.M., Toint, Ph.L.: Trust-Region Methods. SIAM, Philadelphia (2000)
9. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple-instance problem with axis-parallel rectangles. Artif. Intell. **89**, 31–71 (1997)
10. Ding, J., Pan, Z., Chen, L.: Parameter identification of multibody systems based on second order sensitivity analysis. Int. J. Non-Linear Mech. **47**, 1105–1110 (2012)
11. Dolan, E., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–2013 (2002)
12. Dua, D., Graff, C.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine (2019). http://archive.ics.uci.edu/ml
16. Fan, J., Yuan, Y.: A new trust region algorithm with trust region radius converging to zero. In: Li, D. (ed.) Proceeding of the 5th International Conference on Optimization: Techiniques and Applications, pp. 786-794. Hong Kong (2001)
13. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annu. Eugenics **7**(Part II), 179–188 (1936)
14. Fletcher, R.: An efficient, global convergent algorithm for unconstrained and linearly constrained optimization problems. Technical Report TP 431, AERE, Harwell Laboratory, Oxfordshire, England (1970)
15. Fletcher, R.: Practical Methods of Optimization, Volume 1: Unconstrained Optimization. Wiley, Chichester, England (1980)
17. Gorman, R.P., Sejnowski, T.J.: Analysis of hidden units in a layered network trained to classify sonar targets. Neural Netw. **1**, 75–89 (1988)
18. Grapiglia, G.N., Yuan, J., Yuan, Y.: On the convergence and worst-case complexity of trust-region and regularization methods for unconstrained optimization. Math. Program. **152**, 491–520 (2015)
19. Grapiglia, G.N., Yuan, J., Yuan, Y.: Nonlinear stepsize control algorithms: complexity bounds for first-and second-order optimality. J. Optim. Theory Appl. **171**, 980–997 (2016)
20. Gratton, S., Sartenaer, A., Toint, Ph.L.: Recursive trust-region methods for multiscale nonlinear optimization. SIAM J. Optim. **19**, 414–444 (2008)
21. Griewank, A., Walther, A.: Evaluationg Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM, Philadelphia (2008)
22. Hebden, M.D.: An algorithm for minimization using exact second order derivatives. Technical Report TP 515, AERE, Harwell Laboratory, Oxfordshire, England (1973)
23. Heusinger, A., Kanzow, C.: Optimization reformulations of the generalized Nash equilibrium problem using Nikaido-Isoda-type functions. Comput. Optim. Appl. **43**, 353–377 (2009)
24. Koziel, S., Mosler, F., Reitzinger, S., Thoma, P.: Robust microwave design optimization using adjoint sensitivity and trust regions. Int. J. RF Microwave Comput. Aided Eng. **22**, 10–19 (2012)
25. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**, 17–41 (1981)
26. Powell, M.J.D.: A new algorithm for unconstrained optimization. In: Rosen, J.B., Mangasarian, O.L., Ritter, K. (eds.) Nonlinear Programming, pp. 31–66. Academic Press, New York (1970)
27. Powell, M.J.D.: Convergence properties of a class of minimization algorithms. In: O.L. Mangasarian, R.R. Meyer and S.M. Robinson (eds.) Nonlinear Programming, pp. 1–27 (1975)

28. Rossi, R.A., Ahmed, N.K.: The network data repository with interative graph analytics and visualization (2015). http://networkrepository.com
29. Sigillito, V.G., Wing, S.P., Hutton, L.V., Baker, K.B.: Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Tech. Digest **10**, 262–266 (1989)
30. Steihaug, T.: The conjugate gradient and trust regions in large scale optimization. SIAM J. Numer. Anal. **20**, 626–637 (1983)
31. Street, W.N., Wolberg, W.H., Mangasarian, O.L.: Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA (1993)
32. Toint, Ph.L.: Towards an efficient sparsity exploiting Newton method for minimization. In: Duff, I.S. (ed.) Sparse Matrices and Their Uses, pp. 57–88. Academic Press, London (1981)
33. Walmag, J.M.B., Dellez, E.J.M.: A trust-region method applied to parameter identification of a simple prey-predator model. Appl. Math. Model. **29**, 289–307 (2005)
34. Wu, X., Ward, R., Bottou, L.: WNGrad: learn the learning rate in gradient descent. ArXiv:1803.02865, November 2020
35. Yuan, Y.: Recent advances in trust region algorithms. Math. Program. Ser. B **151**, 249–281 (2015)
36. Zhang, H., Li, X., Song, H., Liu, S.: An adaptive subspace trust-region method for frequency-domain seismic full waveform inversion. Comput. Geosci. **78**, 1–14 (2015)