SIAM J. OPTIM.
Vol. 31, No. 1, pp. 518–544
© 2021 Society for Industrial and Applied Mathematics

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

TRUST-REGION NEWTON-CG WITH STRONG SECOND-ORDER COMPLEXITY GUARANTEES FOR NONCONVEX OPTIMIZATION*

FRANK E. CURTIS†, DANIEL P. ROBINSON†, CLÉMENT W. ROYER‡, AND STEPHEN J. WRIGHT§

Abstract. Worst-case complexity guarantees for nonconvex optimization algorithms have been a topic of growing interest. Multiple frameworks that achieve the best known complexity bounds among a broad class of first- and second-order strategies have been proposed. These methods have often been designed primarily with complexity guarantees in mind and, as a result, represent a departure from the algorithms that have proved to be the most effective in practice. In this paper, we consider trust-region Newton methods, one of the most popular classes of algorithms for solving nonconvex optimization problems. By introducing slight modifications to the original scheme, we obtain two methods—one based on exact subproblem solves and one exploiting inexact subproblem solves as in the popular "trust-region Newton-conjugate gradient" (trust-region Newton-CG) method—with iteration and operation complexity bounds that match the best known bounds for the aforementioned class of first- and second-order methods. The resulting trust-region Newton-CG method also retains the attractive practical behavior of classical trust-region Newton-CG, which we demonstrate with numerical comparisons on a standard benchmark test set.

Key words. smooth nonconvex optimization, trust-region methods, Newton's method, conjugate gradient method, Lanczos method, worst-case complexity, negative curvature

AMS subject classifications. 49M05, 49M15, 65K05, 90C60

DOI. 10.1137/19M130563X

1. Introduction.

Consider the unconstrained optimization problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice Lipschitz continuously differentiable and possibly nonconvex. We propose and analyze the complexity of two trust-region algorithms for solving problem (1.1). Our main interest is in an algorithm that, for each subproblem, uses the conjugate gradient (CG) method to minimize an exact second-order Taylor series approximation of f subject to a trust-region constraint, as in so-called *trust-region Newton-CG* methods. Our complexity analysis for both methods is based on approximate satisfaction of second-order necessary conditions for stationarity, that is,

$$\nabla f(x) = 0 \ \text{ and } \ \nabla^2 f(x) \ \text{ positive semidefinite.} \tag{1.2}$$

*Received by the editors December 10, 2019; accepted for publication (in revised form) November 19, 2020; published electronically February 8, 2021.
https://doi.org/10.1137/19M130563X
Funding: The work of the first author was supported by DOE award DE-SC0010615 and NSF awards CCF-1740796 and CCF-1618717. The work of the second author was supported by NSF award DMS-2012243. The work of the third and fourth authors was partially supported by award N660011824020 from the DARPA Lagrange Program. The fourth author was also supported by NSF awards 1628384, 1634597, 2023239, and 1740707 and subcontract 8F-30039 from the Argonne National Laboratory.

†Department of Industrial and Systems Engineering, Lehigh University, 200 W. Packer Ave., Bethlehem, PA 18015-1582 USA (frank.e.curtis@lehigh.edu, daniel.p.robinson@lehigh.edu).
‡LAMSADE, CNRS, Université Paris-Dauphine, Université PSL, 75016 Paris, France (clement.royer@dauphine.psl.eu).
§Computer Sciences Department, University of Wisconsin, 1210 W. Dayton St., Madison, WI 53706 USA (swright@cs.wisc.edu).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Specifically, given a pair of (small) real positive tolerances (ϵ_g, ϵ_H), our algorithms terminate when they find a point x^ϵ such that

$$\|\nabla f(x^\epsilon)\| \le \epsilon_g \quad \text{and} \quad \lambda_{\min}(\nabla^2 f(x^\epsilon)) \ge -\epsilon_H, \tag{1.3}$$

where $\lambda_{\min}(\cdot)$ denotes the minimum eigenvalue of its symmetric matrix argument. Such a point is said to be (ϵ_g, ϵ_H)-*stationary*. By contrast, any point satisfying the approximate first-order condition $\|\nabla f(x)\| \le \epsilon_g$ is called an ϵ_g-*stationary* point.

Recent interest in complexity bounds for nonconvex optimization stems in part from applications in machine learning, where for certain interesting classes of problems all local minima are global minima. We have a particular interest in trust-region Newton-CG algorithms since they have proved to be extremely effective in practice for a wide range of large-scale applications. We show in this paper that by making fairly minor modifications to such an algorithm, we can equip it with strong theoretical complexity properties without significantly degrading important performance measures such as the number of iterations, function evaluations, and gradient evaluations required until an (ϵ_g, ϵ_H)-stationary point is reached. This is in contrast to other recently proposed schemes that achieve good complexity properties but have not demonstrated such good performance in practice against a state-of-the-art trust-region Newton-CG algorithm; see, e.g., [1, 6].

We prove results concerning both *iteration complexity* and *operation complexity*. The former refers to a bound on the number of "outer" iterations required to identify a point that satisfies (1.3). For the latter, we identify a *unit operation* and find a bound on the number of such operations required to find a point satisfying (1.3). As in earlier works on Newton-CG methods [30, 29], the unit operation is either a gradient evaluation or a Hessian-vector multiplication. In both types of complexity—iteration and operation—we focus on the dependence of bounds on the tolerances ϵ_g and ϵ_H.

Our chief contribution is to show that a trust-region Newton-CG method can be modified to have state-of-the-art operation complexity properties for locating an $(\epsilon_g, \epsilon_g^{1/2})$-stationary point, matching recent results for modified line search Newton methods, cubic regularization methods, and other approaches based on applying accelerated gradient to nonconvex functions (see section 1.3). The setting $\epsilon_H = \epsilon_g^{1/2}$ is known to yield the lowest operation complexity bounds for several classes of second-order algorithms.

1.1. Outline. We specify assumptions and notation used throughout the paper in section 1.2 and discuss relevant literature briefly in section 1.3. Section 2 describes a trust-region Newton method in which we assume that the subproblem is solved exactly at each iteration and in which the minimum eigenvalue of the Hessian is calculated as necessary to verify the conditions (1.3). We prove the iteration complexity of this method, setting the stage for our investigation of a method using inexact subproblem solves. In section 3, we describe an inexact implementation of the solution of the trust-region subproblem by a CG method and find bounds on the number of matrix-vector multiplications required for this method. We also discuss the use of iterative methods to obtain approximations to the minimum eigenvalue of the Hessian. Section 4 describes a trust-region Newton-CG method that incorporates the inexact solvers of section 3 and analyzes its iteration and operation complexity properties. We describe implementation challenges in section 5, then detail our computational experiments in section 6. Finally, we make some concluding observations in section 7.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

1.2. Assumptions and notation. We write $\mathbb{R}$ for the set of real numbers (that is, scalars), $\mathbb{R}^n$ for the set of n-dimensional real vectors, $\mathbb{R}^{m\times n}$ for the set of m-by-n-dimensional real matrices, $\mathbb{S}^n \subset \mathbb{R}^{n\times n}$ for the set of n-by-n-dimensional real symmetric matrices, and $\mathbb{N}$ for the set of nonnegative integers. For $v \in \mathbb{R}^n$, we use $\|v\|$ to denote the ℓ_2-norm of v. Given scalars $(a, b) \in \mathbb{R} \times \mathbb{R}$, we write $a \perp b$ to mean $ab = 0$.

In reference to problem (1.1), we use $g := \nabla f : \mathbb{R}^n \to \mathbb{R}$ and $H := \nabla^2 f : \mathbb{R}^n \to \mathbb{S}^n$ to denote the gradient and Hessian functions of f, respectively. For each iteration $k \in \mathbb{N} = \{0, 1, 2, \dots\}$ of an algorithm for solving (1.1), we let x_k denote the kth solution estimate (that is, iterate) computed. For brevity, we append $k \in \mathbb{N}$ as a subscript to a function to denote its value at the kth iterate, e.g., $f_k := f(x_k)$, $g_k := g(x_k)$, and $H_k := H(x_k)$. The subscript $j \in \mathbb{N}$ is similarly used for the iterates of the subroutines used for computing search directions for an algorithm for solving (1.1). Given $H_k \in \mathbb{S}^n$, we let $\lambda_k := \lambda_{\min}(H_k)$ denote the minimum eigenvalue of H_k with respect to $\mathbb{R}$.

Given functions $\phi : \mathbb{R} \to \mathbb{R}$ and $\varphi : \mathbb{R} \to [0, \infty)$, we write $\phi(\cdot) = \mathcal{O}(\varphi(\cdot))$ to indicate that $|\phi(\cdot)| \le C\varphi(\cdot)$ for some $C \in (0, \infty)$. Similarly, we write $\phi(\cdot) = \tilde{\mathcal{O}}(\varphi(\cdot))$ to indicate that $|\phi(\cdot)| \le C\varphi(\cdot)|\log^c(\cdot)|$ for some $C \in (0, \infty)$ and $c \in (0, \infty)$. In this manner, one finds that $\mathcal{O}(\varphi(\cdot)\log^c(\cdot)) \equiv \tilde{\mathcal{O}}(\varphi(\cdot))$ for any $c \in (0, \infty)$.

The following assumption on the objective function in (1.1) is made throughout.

Assumption 1. The objective function value sequence $\{f_k\}$ is bounded below by $f_{\text{low}} \in \mathbb{R}$. The sequence of line segments $\{[x_k, x_k + s_k]\}$ lies in an open set over which f is twice continuously differentiable and the gradient and Hessian functions are Lipschitz continuous with constants $L_g \in (0, \infty)$ and $L_H \in (0, \infty)$, respectively.

The following bounds are implied by Assumption 1 (see, e.g., [26]):

$$f(x_k + s_k) - f_k - g_k^T s_k - \frac{1}{2} s_k^T H_k s_k \le \frac{L_H}{6}\|s_k\|^3 \qquad \text{for all } k \in \mathbb{N}, \tag{1.4a}$$

$$\|g(x_k + s_k) - g_k - H_k s_k\| \le \frac{L_H}{2}\|s_k\|^2 \qquad \text{for all } k \in \mathbb{N}, \tag{1.4b}$$

$$\text{and } \|H_k\| \le L_g \qquad \text{for all } k \in \mathbb{N}. \tag{1.4c}$$

1.3. Literature review. Complexity results for smooth nonconvex optimization algorithms abound in recent literature. We discuss these briefly and give some pointers below, with a focus on methods with best known complexity.

Cubic regularization [27, Theorem 1] has iteration complexity $\mathcal{O}(\epsilon_g^{-3/2})$ to find an $(\epsilon_g, \epsilon_g^{1/2})$-stationary point; see also [7, 9]. Algorithms that find such a point with *operation* complexity $\tilde{\mathcal{O}}(\epsilon_g^{-7/4})$, with high probability, were proposed in [1, 6]. (The "high probability" is due to the use of randomized iterative methods for calculating a minimum eigenvalue and/or solving a linear system.) A method that *deterministically* finds an ϵ_g-stationary point in $\tilde{\mathcal{O}}(\epsilon_g^{-7/4})$ gradient evaluations was described in [5].

Line search methods that make use of Newton-like steps, the CG method for inexactly solving linear systems, and randomized Lanczos for calculating negative curvature directions are described in [30, 29]. These methods also have operation complexity $\tilde{\mathcal{O}}(\epsilon_g^{-7/4})$ to find a $(\epsilon_g, \epsilon_g^{1/2})$-stationary point with high probability. The method in [29] finds an ϵ_g-stationary point deterministically in $\tilde{\mathcal{O}}(\epsilon_g^{-7/4})$ operations, showing that the CG method on nonconvex quadratics shares properties with accelerated gradient on nonconvex problems as described in [5].

Trust-region methods. An early result of [19] shows that standard trust-region methods require $\mathcal{O}(\epsilon_g^{-2})$ iterations to find an ϵ_g-stationary point; this complexity was

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

shown to be sharp in [8]. A trust-region Newton method with iteration complexity of $\mathcal{O}(\max\{\epsilon_g^{-3/2}, \epsilon_H^{-3}\})$ for finding an (ϵ_g, ϵ_H)-stationary point is described in [11]. This complexity matches that of cubic regularization methods [27, 7, 9].

Another method that uses trust regions in conjunction with a cubic model to find an (ϵ_g, ϵ_H)-stationary point with guaranteed complexity appears in [23]. This is not a trust-region method in the conventional sense because it fixes the trust-region radius at a constant value. Other methods that combine trust-region and cubic-regularization techniques in search of good complexity bounds are described in [12, 14, 15, 2, 3].

Solving the trust-region subproblem. Efficient solution of the trust-region subproblem is a core aspect of both the theory and practice of trust-region methods. In the context of this paper, such results are vital in turning an iteration complexity bound into an operation complexity bound. The fact that the trust-region subproblem (with a potentially nonconvex objective) can be solved efficiently remains surprising to many. This is especially true since it has some complicating features, particularly the "hard case" in which, in iteration $k \in \mathbb{N}$, the gradient g_k is orthogonal to the eigenspace of the Hessian H_k corresponding to its minimum eigenvalue.

Approaches for solving trust-region subproblems based on matrix factorizations are described in [24]; see also [28, Chapter 4]. For large-scale problems, iterative techniques based on the CG algorithm [31, 32] and the Lanczos method [16, 25] have been described in the literature. Convergence rates for the method of [16] are presented in [34], though results are weaker in the hard case.

Global convergence rates in terms of the objective function values for the trust-region subproblem are a recent focus; see, for example, [20], wherein the authors use a semidefinite programming relaxation, and [33], wherein the authors apply an accelerated gradient method to a convex reformulation of the trust-region subproblem (which requires an estimate of the minimum eigenvalue of the Hessian). Both solve the trust-region subproblem to within ϵ of the optimal subproblem objective value in $\tilde{\mathcal{O}}(\epsilon^{-1/2})$ time.

A recent method based on Krylov subspaces is presented in [4]. This method circumvents the hard case by its use of randomization. Subsequent work in [18] derives a convergence rate for the norm of the residual vectors in the Krylov-subspace approach.

The hard case does not present a serious challenge to the main algorithm that we propose (Algorithm 4.1). When it occurs, either the CG procedure (Algorithm 3.1) returns an acceptable trial step, or else the minimum-eigenvalue procedure (Algorithm 3.2) will be invoked to find a negative curvature step.

2. An exact trust-region Newton method.

In this section, we propose a trust-region Newton method that uses, during each iteration, the *exact* solution of a (regularized) trust-region subproblem. The algorithm is described in section 2.1, and its complexity guarantees are analyzed in section 2.2. Our analysis of this method sets the stage for our subsequent method that uses inexact subproblem solutions.

2.1. The algorithm.

Our trust-region Newton method with exact subproblem solves, which is inspired in part by the line search method proposed in [30], is written as Algorithm 2.1. Unlike a traditional trust-region method, the second-order stationarity tolerance $\epsilon_H \in (0, \infty)$ is used to quantify a regularization of the quadratic model $m_k : \mathbb{R}^n \to \mathbb{R}$ of f at x_k used in the subproblem, which is given by

$$m_k(x) := f_k + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k(x - x_k); \tag{2.1}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 2.1. Trust-Region Newton Method (exact version)

Require: Tolerances $\epsilon_g \in (0,\infty)$ and $\epsilon_H \in (0,\infty)$; trust-region adjustment parameters $\gamma_1 \in (0,1)$, $\gamma_2 \in [1,\infty)$, and $\psi \in (1/\gamma_2, 1]$; initial iterate $x_0 \in \mathbb{R}^n$; initial trust-region radius $\delta_0 \in (0,\infty)$; maximum trust-region radius $\delta_{\max} \in [\delta_0,\infty)$; and step acceptance parameter $\eta \in (0,1)$.

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Evaluate g_k and H_k.
3: Initialize $\lambda_k \leftarrow \infty$.
4: **if** $\|g_k\| \leq \epsilon_g$ **then**
5: Compute $\lambda_k \leftarrow \lambda_{\min}(H_k)$.
6: **if** $\lambda_k \geq -\epsilon_H$ **then**
7: **return** x_k as an (ϵ_g, ϵ_H)-stationary point for problem (1.1).
8: **end if**
9: **end if**
10: Compute a trial step s_k as a solution to the regularized trust-region subproblem

$$\min_{s\in\mathbb{R}^n} \; m_k(x_k+s) + \frac{1}{2}\epsilon_H\|s\|^2 \;\text{ s.t. }\; \|s\| \leq \delta_k. \tag{2.2}$$

11: Compute the ratio of actual-to-predicted reduction in f, defined as

$$\rho_k \leftarrow \frac{f_k - f(x_k+s_k)}{m_k(x_k) - m_k(x_k+s_k)}. \tag{2.3}$$

12: **if** $\rho_k \geq \eta$ **then**
13: Set $x_{k+1} \leftarrow x_k + s_k$.
14: **if** $\|s_k\| \geq \psi\delta_k$ **then**
15: Set $\delta_{k+1} \leftarrow \min\{\gamma_2\delta_k, \delta_{\max}\}$.
16: **else**
17: Set $\delta_{k+1} \leftarrow \delta_k$.
18: **end if**
19: **else**
20: Set $x_{k+1} \leftarrow x_k$ and $\delta_{k+1} \leftarrow \gamma_1\|s_k\|$.
21: **end if**
22: **end for**

see (2.2). Our choice of regularization makes for a relatively straightforward complexity analysis because it causes the resulting trial step s_k to satisfy certain desirable objective function decrease properties. Of course, a possible downside is that the practical behavior of the method may be affected by the choice of the stationarity tolerance ϵ_H, which is not the case for a traditional trust-region framework. In any case, the remainder of Algorithm 2.1 is identical to a traditional trust-region Newton method.

Before presenting our analysis of Algorithm 2.1, we remark that the sequence $\{\lambda_k\}$ of minimum eigenvalues of the Hessians $\{H_k\}$ does not influence the iterate sequence $\{x_k\}$. The only use of these values is in the termination test in line 6 to determine when an (ϵ_g, ϵ_H)-stationary point has been found.

2.2. Iteration complexity. We show that Algorithm 2.1 reaches an (ϵ_g, ϵ_H)-stationary point in a number of iterations that is bounded by a function of ϵ_g and ϵ_H. To this end, let us define the set of iteration numbers

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

$$\mathcal{K} := \{k \in \mathbb{N} : \text{iteration } k \text{ is completed without algorithm termination}\}$$

along with the subsets

$$\mathcal{I} := \{k \in \mathcal{K} : \|s_k\| < \delta_k\} \quad \text{and} \quad \mathcal{B} := \{k \in \mathcal{K} : \|s_k\| = \delta_k\}$$

and

$$\mathcal{S} := \{k \in \mathcal{K} : \rho_k \geq \eta\} \quad \text{and} \quad \mathcal{U} := \{k \in \mathcal{K} : \rho_k < \eta\}.$$

The pairs $(\mathcal{I}, \mathcal{B})$ and $(\mathcal{S}, \mathcal{U})$ are each partitions of $\mathcal{K}$. The iterations with $k \in \mathcal{I}$ are those with s_k in the *interior* of the trust region, and those with $k \in \mathcal{B}$ are those with s_k on the *boundary* of the trust region. The iterations with $k \in \mathcal{S}$ are called the *successful* iterations, and those with $k \in \mathcal{U}$ are called the *unsuccessful* iterations. Due to the termination conditions in line 6, it follows for Algorithm 2.1 that

$$\mathcal{K} = \{k \in \mathbb{N} : \text{iteration } k \text{ is reached and either } \|g_k\| > \epsilon_g \text{ or } \lambda_k < -\epsilon_H\}. \tag{2.4}$$

It follows that, for a run of Algorithm 2.1, the cardinalities of all of the index sets $\mathcal{K}$, $\mathcal{I}$, $\mathcal{B}$, $\mathcal{S}$, and $\mathcal{U}$ are functions of the tolerance parameters ϵ_g and ϵ_H.

Since s_k is computed as the global solution of the trust-region subproblem (2.2), it is well known [24, 28] that there exists a scalar Lagrange multiplier μ_k such that

$$g_k + (H_k + \epsilon_H I + \mu_k I)s_k = 0, \tag{2.5a}$$

$$H_k + \epsilon_H I + \mu_k I \succeq 0, \tag{2.5b}$$

$$\text{and} \quad 0 \leq \mu_k \perp (\delta_k - \|s_k\|) \geq 0. \tag{2.5c}$$

Our first result is a lower bound on the model reduction achieved by a trial step.

LEMMA 2.1. *For all $k \in \mathcal{K}$, the model reduction satisfies*

$$m_k(x_k) - m_k(x_k + s_k) \geq \frac{1}{2}\epsilon_H \|s_k\|^2. \tag{2.6}$$

Proof. The definition of m_k in (2.1) and the optimality conditions in (2.5) give

$$\begin{aligned} m_k(x_k) - m_k(x_k + s_k) &= -g_k^T s_k - \frac{1}{2} s_k^T H_k s_k \\ &= s_k^T (H_k + \epsilon_H I + \mu_k I) s_k - \frac{1}{2} s_k^T H_k s_k \\ &= \frac{1}{2} s_k^T (H_k + \epsilon_H I + \mu_k I) s_k + \frac{1}{2}\epsilon_H \|s_k\|^2 + \frac{1}{2}\mu_k \|s_k\|^2 \\ &\geq \frac{1}{2}\epsilon_H \|s_k\|^2, \end{aligned}$$

as desired. ∎

Next, we show that all sufficiently small trial steps lead to successful iterations.

LEMMA 2.2. *For all $k \in \mathcal{K}$, if $k \in \mathcal{U}$, then $\delta_k > 3(1-\eta)\epsilon_H / L_H$. Hence, by the trust-region radius update procedure, it follows that*

$$\delta_k \geq \delta_{\min} := \min\left\{\delta_0, \left(\frac{3\gamma_1(1-\eta)}{L_H}\right)\epsilon_H\right\} \in (0, \infty) \quad \textit{for all} \quad k \in \mathcal{K}.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Proof. We begin by proving the first statement of the lemma. To that end, suppose that $k \in \mathcal{U}$ (meaning that $\rho_k < \eta$), which from the definition of ρ_k means that

$$\eta \left(m_k(x_k + s_k) - m_k(x_k)\right) < f(x_k + s_k) - f_k. \tag{2.7}$$

Combining (2.7) with (1.4a), (2.2), Lemma 2.1, and (2.1) leads to

$$\begin{aligned}
&\eta \left(m_k(x_k + s_k) - m_k(x_k)\right) < g_k^T s_k + \frac{1}{2} s_k^T H_k s_k + \frac{L_H}{6}\|s_k\|^3 \\
&\implies (\eta - 1)\left(m_k(x_k + s_k) - m_k(x_k)\right) < \frac{L_H}{6}\|s_k\|^3 \\
&\implies \frac{1-\eta}{2}\epsilon_H \|s_k\|^2 < \frac{L_H}{6}\|s_k\|^3 \\
&\implies \frac{3(1-\eta)}{L_H}\epsilon_H < \|s_k\|.
\end{aligned}$$

We have shown that $k \in \mathcal{U}$ implies $\delta_k \geq \|s_k\| > 3(1-\eta)\epsilon_H / L_H$, as desired. Using a contraposition argument, we also have that $\|s_k\| \leq 3(1-\eta)\epsilon_H / L_H$ implies $k \in \mathcal{S}$. Combining this with the trust-region radius update procedure and accounting for the initial radius δ_0 completes the proof. ∎

We now establish that each successful step guarantees that a certain amount of decrease in the objective function value is achieved.

LEMMA 2.3. *The following hold for all successful iterations:*

(i) *If $k \in \mathcal{B} \cap \mathcal{S}$, then*

$$f_k - f_{k+1} \geq \frac{\eta}{2}\epsilon_H \delta_k^2.$$

(ii) *If $k \in \mathcal{I} \cap \mathcal{S}$, then*

$$f_k - f_{k+1} \geq \frac{\eta}{2(1 + 2L_H)} \min\left\{\|g_{k+1}\|^2 \epsilon_H^{-1}, \epsilon_H^3\right\}.$$

Proof. Part (i) follows from Lemma 2.1 and the definition of $\mathcal{B}$, which imply that

$$f_k - f_{k+1} \geq \eta \left(m_k(x_k) - m_k(x_k + s_k)\right) \geq \frac{\eta}{2}\epsilon_H \|s_k\|^2 = \frac{\eta}{2}\epsilon_H \delta_k^2.$$

For part (ii), from $k \in \mathcal{I}$ we know that $\|s_k\| < \delta_k$. This fact along with (2.5c) and (2.5a) imply that $\mu_k = 0$ and $g_k + (H_k + \epsilon_H I)s_k = 0$. Now, with (1.4b), we have

$$\begin{aligned}
\|g_{k+1}\| &= \|g_{k+1} - g_k - (H_k + \epsilon_H I)s_k\| \\
&\leq \|g_{k+1} - g_k - H_k s_k\| + \epsilon_H \|s_k\| \leq \frac{L_H}{2}\|s_k\|^2 + \epsilon_H \|s_k\|,
\end{aligned}$$

which after rearrangement yields

$$\frac{L_H}{2}\|s_k\|^2 + \epsilon_H \|s_k\| - \|g_{k+1}\| \geq 0.$$

Treating the left-hand side as a quadratic scalar function of $\|s_k\|$ implies that

$$\|s_k\| \geq \frac{-\epsilon_H + \sqrt{\epsilon_H^2 + 2L_H\|g_{k+1}\|}}{L_H} = \frac{-1 + \sqrt{1 + 2L_H\|g_{k+1}\|\epsilon_H^{-2}}}{L_H}\,\epsilon_H.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

To put this lower bound into a slightly more useful form, we use [30, Lemma 17 in Appendix A], which states that for scalars $(a, b, t) \in (0, \infty) \times (0, \infty) \times [0, \infty)$, we have

$$-a + \sqrt{a^2 + bt} \geq (-a + \sqrt{a^2 + b}) \min\{t, 1\}. \tag{2.8}$$

By setting $a = 1$, $b = 2L_H$, and $t = \|g_{k+1}\|\epsilon_H^{-2}$, we obtain

$$\begin{aligned}
\|s_k\| &\geq \left(\frac{-1 + \sqrt{1 + 2L_H}}{L_H}\right) \min\left\{\|g_{k+1}\|\epsilon_H^{-2}, 1\right\} \epsilon_H \\
&= \left(\frac{2L_H}{L_H(1 + \sqrt{1 + 2L_H})}\right) \min\left\{\|g_{k+1}\|\epsilon_H^{-1}, \epsilon_H\right\} \\
&\geq \left(\frac{1}{\sqrt{1 + 2L_H}}\right) \min\left\{\|g_{k+1}\|\epsilon_H^{-1}, \epsilon_H\right\}.
\end{aligned}$$

Using this inequality in conjunction with $k \in \mathcal{S}$ and Lemma 2.1 proves that

$$\begin{aligned}
f_k - f(x_k + s_k) &\geq \eta\left(m_k(x_k) - m_k(x_k + s_k)\right) \geq \frac{\eta}{2}\epsilon_H\|s_k\|^2 \\
&\geq \frac{\eta}{2(1 + 2L_H)} \min\left\{\|g_{k+1}\|^2\epsilon_H^{-1}, \epsilon_H^3\right\},
\end{aligned}$$

which completes the proof for part (ii). ∎

We now bound the number of successful iterations before termination.

LEMMA 2.4. *The number of successful iterations performed by Algorithm* 2.1 *before an* (ϵ_g, ϵ_H)*-stationary point is reached satisfies*

$$|\mathcal{S}| \leq \left\lfloor \mathcal{C}_{\mathcal{S}} \max\{\epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\} \right\rfloor + 1, \tag{2.9}$$

where

$$\mathcal{C}_{\mathcal{S}} := \frac{4(f_0 - f_{\text{low}})}{\eta} \max\left\{\frac{1}{\delta_0^2}, \frac{L_H^2}{9\gamma_1^2(1 - \eta)^2}, 1 + 2L_H\right\}. \tag{2.10}$$

Proof. The successful iterations may be written as $\mathcal{S} = \mathcal{S}_L \cup \mathcal{S}_{GG} \cup \mathcal{S}_{GL}$, where

$$\begin{aligned}
\mathcal{S}_L &:= \{k \in \mathcal{S} : \|g_k\| \leq \epsilon_g\}, \\
\mathcal{S}_{GG} &:= \{k \in \mathcal{S} : \|g_k\| > \epsilon_g \text{ and } \|g_{k+1}\| > \epsilon_g\}, \\
\text{and } \mathcal{S}_{GL} &:= \{k \in \mathcal{S} : \|g_k\| > \epsilon_g \text{ and } \|g_{k+1}\| \leq \epsilon_g\}.
\end{aligned}$$

We first bound $|\mathcal{S}_L \cup \mathcal{S}_{GG}|$, for which we will make use of the constant

$$c := \frac{\eta}{2} \min\left\{\delta_0^2, \frac{9\gamma_1^2(1 - \eta)^2}{L_H^2}, \frac{1}{1 + 2L_H}\right\}. \tag{2.11}$$

For $k \in \mathcal{S}_L$, the fact that the algorithm has not yet terminated implies (see (2.4)) that $\lambda_k < -\epsilon_H$. By (2.5), it follows that $\mu_k > 0$ and $\|s_k\| = \delta_k$, and thus $k \in \mathcal{B}$. Thus, for $k \in \mathcal{S}_L$, Lemma 2.3(i), Lemma 2.2, and (2.11) imply that

$$f_k - f_{k+1} \geq \frac{\eta}{2}\epsilon_H\delta_k^2 \geq \frac{\eta}{2} \min\left\{\delta_0^2\epsilon_H, \frac{9\gamma_1^2(1 - \eta)^2}{L_H^2}\epsilon_H^3\right\} \geq c \min\left\{\epsilon_H, \epsilon_H^3\right\}. \tag{2.12}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Now consider $k \in \mathcal{S}_{GG}$. Since in this case either of the cases in Lemma 2.3 may apply, one can only conclude that, for each $k \in \mathcal{S}_{GG}$, the following bound holds:

$$f_k - f_{k+1} \geq \frac{\eta}{2} \min \left\{ \delta_k^2 \epsilon_H, \left(\frac{\|g_{k+1}\|^2}{1 + 2L_H} \right) \epsilon_H^{-1}, \left(\frac{1}{1 + 2L_H} \right) \epsilon_H^3 \right\}.$$

Combining this with the definition of $\mathcal{S}_{GG}$, the lower bound on δ_k in Lemma 2.2, and the definition of c in (2.11), it follows that

$$f_k - f_{k+1} \geq c \min \left\{ \epsilon_H, \epsilon_g^2 \epsilon_H^{-1}, \epsilon_H^3 \right\}. \tag{2.13}$$

To bound $|\mathcal{S}_L \cup \mathcal{S}_{GG}|$, we sum the objective function decreases obtained over all such iterations, which with Assumption 1 and the monotonicity of $\{f_k\}$ gives

$$f_0 - f_{\text{low}} \geq \sum_{k \in \mathcal{K}} (f_k - f_{k+1}) \geq \sum_{k \in \mathcal{S}_L} (f_k - f_{k+1}) + \sum_{k \in \mathcal{S}_{GG}} (f_k - f_{k+1}).$$

Combining this inequality with (2.12) and (2.13) shows that

$$\begin{aligned} f_0 - f_{\text{low}} &\geq \sum_{k \in \mathcal{S}_L} c \min\{\epsilon_H, \epsilon_H^3\} + \sum_{k \in \mathcal{S}_{GG}} c \min \left\{ \epsilon_H, \epsilon_g^2 \epsilon_H^{-1}, \epsilon_H^3 \right\} \\ &\geq c(|\mathcal{S}_L| + |\mathcal{S}_{GG}|) \min \left\{ \epsilon_H, \epsilon_g^2 \epsilon_H^{-1}, \epsilon_H^3 \right\}, \end{aligned}$$

from which it follows that

$$|\mathcal{S}_L| + |\mathcal{S}_{GG}| \leq \left(\frac{f_0 - f_{\text{low}}}{c} \right) \max \left\{ \epsilon_H^{-1}, \epsilon_g^{-2} \epsilon_H, \epsilon_H^{-3} \right\}. \tag{2.14}$$

Next, let us consider the set $\mathcal{S}_{GL}$. Since $k \in \mathcal{S}_{GL}$ means $\|g_{k+1}\| \leq \epsilon_g$, the index corresponding to the next successful iteration (if one exists) must be an element of the index set $\mathcal{S}_L$. This implies that $|\mathcal{S}_{GL}| \leq |\mathcal{S}_L| + 1$, where the 1 accounts for the possibility that the last successful iteration (prior to termination) has an index in $\mathcal{S}_{GL}$. Combining this bound with (2.14) yields

$$|\mathcal{S}| = |\mathcal{S}_L| + |\mathcal{S}_{GG}| + |\mathcal{S}_{GL}| \leq \frac{2(f_0 - f_{\text{low}})}{c} \max \left\{ \epsilon_H^{-1}, \epsilon_g^{-2} \epsilon_H, \epsilon_H^{-3} \right\} + 1,$$

which completes the proof when we substitute for c from (2.11). ∎

We now bound the number of unsuccessful iterations.

LEMMA 2.5. *The number of unsuccessful iterations that occur before an (ϵ_g, ϵ_H)-stationary point is reached either is zero or else satisfies*

$$|\mathcal{U}| \leq \left\lfloor 1 + \log_{\gamma_1} \left(\frac{3(1-\eta)}{L_H \delta_{\max}} \right) + \log_{\gamma_1} (\epsilon_H) \right\rfloor |\mathcal{S}|. \tag{2.15}$$

Proof. If the number of successful iterations is zero, then the initial point must be (ϵ_g, ϵ_H)-stationary and there are no unsuccessful iterations. Hence, let us proceed under the assumption that $|\mathcal{S}| \geq 1$. Let us denote the successful iteration indices as $\{k_1, \ldots, k_{|\mathcal{S}|}\} := \mathcal{S}$. If the number of unsuccessful iterations is zero, then there is nothing left to prove, so we may proceed under the assumption that there is at least one unsuccessful iteration. Thus, we may consider arbitrary $i \in \{1, \ldots, |\mathcal{S}| - 1\}$ such that $k_{i+1} - k_i - 1 \geq 1$, that is, there is at least one unsuccessful iteration

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

between iterations k_i and k_{i+1}. We seek a bound on $k_{i+1} - k_i - 1$. From the update formulas for the trust-region radius, one finds for all unsuccessful iteration indices $l \in \{k_i + 1, \ldots, k_{i+1} - 1\}$ that $\delta_l = \gamma_1 \|s_{l-1}\| \leq \gamma_1 \delta_{l-1}$, so

$$\delta_l \leq \min\{\gamma_2 \delta_{k_i}, \delta_{\max}\} \gamma_1^{l-k_i-1} \leq \delta_{\max} \gamma_1^{l-k_i-1}. \tag{2.16}$$

Moreover, for any unsuccessful iteration index $l \in \{k_i + 1, \ldots, k_{i+1} - 1\}$ we have from Lemma 2.2 that $\delta_l > 3(1-\eta)\epsilon_H / L_H$. Thus, for such l, (2.16) implies that

$$\frac{3(1-\eta)}{L_H} \epsilon_H < \delta_{\max} \gamma_1^{l-k_i-1} \implies l - k_i - 1 \leq \log_{\gamma_1} \left(\frac{3(1-\eta)\epsilon_H}{L_H \delta_{\max}} \right).$$

Consequently, using the specific choice $l = k_{i+1} - 1$, one finds that

$$k_{i+1} - k_i - 1 \leq 1 + \log_{\gamma_1} \left(\frac{3(1-\eta)}{L_H \delta_{\max}} \right) + \log_{\gamma_1}(\epsilon_H),$$

and because the left-hand side is an integer, we have

$$k_{i+1} - k_i - 1 \leq \left\lfloor 1 + \log_{\gamma_1} \left(\frac{3(1-\eta)}{L_H \delta_{\max}} \right) + \log_{\gamma_1}(\epsilon_H) \right\rfloor. \tag{2.17}$$

Since i was chosen arbitrarily such that $k_{i+1} - k_i - 1 \geq 1$, the right-hand side in (2.17) is at least 1 if there are *any* unsuccessful iterations between iteration k_1 and $k_{|\mathcal{S}|}$.

Consider now the first successful iteration k_1. We seek a bound on the number of unsuccessful iterations prior to iteration k_1. If there are no such unsuccessful iterations, then there is nothing left to prove; hence, we may assume $k_1 \geq 1$. We have that $\delta_{k_1} \leq \gamma_1^{k_1} \delta_0 \leq \gamma_1^{k_1} \delta_{\max}$, and thus from Lemma 2.2 it follows that

$$\min \left\{ \delta_0, \frac{3\gamma_1(1-\eta)}{L_H} \epsilon_H \right\} \leq \delta_{k_1} \leq \gamma_1^{k_1} \delta_0 \leq \gamma_1^{k_1} \delta_{\max}.$$

From the first two of these inequalities and the facts that $\gamma_1 \in (0,1)$ and $k_1 \geq 1$, the "min" on the left-hand side is *not* achieved by δ_0, so we have

$$\frac{3\gamma_1(1-\eta)}{L_H} \epsilon_H \leq \gamma_1^{k_1} \delta_{\max},$$

which, when we take into account that k_1 is an integer, leads to

$$k_1 \leq \left\lfloor 1 + \log_{\gamma_1} \left(\frac{3(1-\eta)}{L_H \delta_{\max}} \right) + \log_{\gamma_1}(\epsilon_H) \right\rfloor. \tag{2.18}$$

Under our assumption that $k_1 \geq 1$, the right-hand side of (2.18) is at least 1.

Since the iteration immediately prior to termination is $k_{|\mathcal{S}|}$ (except in the trivial case in which termination occurs at the initial point), we have

$$|\mathcal{U}| = k_1 + \sum_{i=1}^{|\mathcal{S}|-1} (k_{i+1} - k_i - 1). \tag{2.19}$$

If $\mathcal{U} \neq \emptyset$, we have that $k_1 \geq 1$ and/or at least one of the terms in the summation is at least 1. We can in this case bound *every* term on the right-hand side of (2.19) by the right-hand sides of (2.17) and (2.18) to deduce the result. ∎

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

The main result for iteration complexity of Algorithm 2.1 may now be proved.

THEOREM 2.6. *Under Assumption* 1*, the number of successful iterations* (*and objective gradient and Hessian evaluations*) *performed by Algorithm* 2.1 *before an* (ϵ_g, ϵ_H)*-stationary point is obtained satisfies*

$$|\mathcal{S}| = \mathcal{O}\left(\max\left\{\epsilon_H^{-3}, \epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H\right\}\right), \tag{2.20}$$

and the total number of iterations (*and objective function evaluations*) *performed before such a point is obtained satisfies*

$$|\mathcal{K}| = \mathcal{O}\left(\log_{1/\gamma_1}\left(\epsilon_H^{-1}\right)\max\left\{\epsilon_H^{-3}, \epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H\right\}\right). \tag{2.21}$$

Proof. Formula (2.20) follows from Lemma 2.4. Formula (2.21) follows from Lemma 2.4, Lemma 2.5, and the fact that $\log_{\gamma_1}(\epsilon_H) = \log_{1/\gamma_1}(\epsilon_H^{-1})$. ∎

If one chooses $\epsilon_H = \epsilon_g^{1/2}$ in (2.20) and (2.21) as well as any positive scalar $\bar{\epsilon}_g \in \mathbb{R}$, then Theorem 2.6 implies that, for all $\epsilon_g \in (0, \bar{\epsilon}_g]$, one has

$$|\mathcal{S}| = \mathcal{O}\left(\epsilon_g^{-3/2}\right) \quad \text{and} \quad |\mathcal{K}| = \mathcal{O}\left(\epsilon_g^{-3/2}\log_{1/\gamma_1}\left(\epsilon_g^{-1/2}\right)\right) = \tilde{\mathcal{O}}\left(\epsilon_g^{-3/2}\right)$$

for the numbers of successful and total iterations, respectively. These correspond to the results obtained for the line search method in [30, Theorem 5, Theorem 6]).

3. Iterative methods for solving the subproblems inexactly. This section describes the algorithms needed to develop an *inexact* trust-region Newton method, which will be presented and analyzed in section 4. A truncated CG method for computing directions of descent is discussed in section 3.1 and an iterative algorithm for computing directions of negative curvature is described in section 3.2.

3.1. A truncated CG method. We propose Algorithm 3.1 as an appropriate iterative method for approximately solving the trust-region subproblem

$$\min_{s\in\mathbb{R}^n} \; g^T s + \frac{1}{2}s^T(H + 2\epsilon I)s \;\; \text{s.t.} \;\; \|s\| \le \delta, \tag{3.1}$$

where $g \in \mathbb{R}^n$ is assumed to be a nonzero vector, $H \in \mathbb{S}^n$ is possibly indefinite, $\epsilon \in (0, \infty)$ plays the role of a regularization parameter, and $\delta \in (0, \infty)$. Algorithm 3.1 is based on the CG method and builds on the Steihaug–Toint approach [31, 32]. (The factor of 2 in the regularization term in (3.1) is intentional. For consistency in the termination condition, the inexact trust-region Newton method in section 4 employs a larger regularization term than the exact method analyzed in section 2.)

For the most part, Algorithm 3.1 is identical to traditional truncated CG. For example, termination occurs in line 18 when the next CG iterate y_{j+1} would lie outside the trust region, and we return s as the largest feasible step on the line segment connecting y_j to y_{j+1}. In this situation, we also set `outCG` ← BND-NORM to indicate that s lies on the boundary of the trust-region constraint.

However, there are three key differences between Algorithm 3.1 and truncated CG. First, the residual termination criterion in line 21 enforces the condition

$$\|(H + 2\epsilon I)s + g\| \;\le\; \frac{\zeta}{2}\min\{\|g\|, \epsilon\|s\|\}, \tag{3.2}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

which is stronger than the condition traditionally used in truncated CG (which typically has $\frac{\zeta}{2}\|g\|$ for the right-hand side) and incorporates a criterion typical of Newton-type methods with optimal complexity [10, 30] (which use $\epsilon\|s\|$ for the right-hand side). If this criterion is satisfied, then we return the current CG iterate as the step (that is, we set $s \leftarrow y_{j+1}$) and indicate that s lies in the interior of the trust region and satisfies the residual condition (3.2) by setting `outCG` $\leftarrow$ INT-RES.

Second, traditional truncated CG terminates if a direction of nonpositive curvature is encountered. Line 10 of Algorithm 3.1 triggers termination if a direction with curvature less than or equal to ϵ is found for $H + 2\epsilon I$, since this condition implies that the curvature of H along the same direction is less than or equal to $-\epsilon$. In this case, we return a step s obtained by moving along the direction of negative curvature to the boundary of the trust-region constraint and return `outCG` $\leftarrow$ BND-NEG to indicate that s lies on the boundary because a direction of negative curvature was computed.

Third, unlike traditional truncated CG, which (in exact arithmetic) requires up to a maximum of $k_{\max} = n$ iterations, line 4 allows for an alternative iteration limit to be imposed. Regardless of which limit is used, if $k_{\max}$ iterations are performed, Algorithm 3.1 returns s as the current CG iterate and sets `outCG` $\leftarrow$ INT-MAX. This flag indicates that the maximum number of iterations has been reached while remaining in the interior of the trust region.

The lemma below motivates the alternative choice for $k_{\max}$ in line 4.

LEMMA 3.1. *Suppose* $\epsilon I \prec H + 2\epsilon I \preceq (M + 2\epsilon)I$ *for* $M \in [\|H\|, \infty)$, *and define*

$$\kappa(M,\epsilon) := (M + 2\epsilon)/\epsilon \quad \textit{and} \quad J(M,\epsilon,\zeta) := \frac{1}{2}\sqrt{\kappa(M,\epsilon)}\ln\left(4\kappa(M,\epsilon)^{3/2}/\zeta\right), \tag{3.3}$$

where $\zeta \in (0,1)$ *is input to Algorithm* 3.1. *If lines* 3–7 *were simply to set* $k_{\max} \leftarrow \infty$, *then Algorithm* 3.1 *would terminate at either line* 18 *or line* 22 *after a number of iterations (equivalently, matrix-vector products) equal to at most*

$$\min\{n, J(M,\epsilon,\zeta)\} \;=\; \min\left\{n, \tilde{\mathcal{O}}\left(\epsilon^{-1/2}\right)\right\}. \tag{3.4}$$

Proof. Since $H + 2\epsilon I \succ \epsilon I$ by assumption, a direction of curvature less than ϵ for $H + 2\epsilon I$ does not exist, meaning that termination in line 12 cannot occur. It follows from the fact that $\epsilon I \prec H + 2\epsilon I \preceq (M + 2\epsilon)I$ and [30, proof of Lemma 11] that CG would reach an iterate satisfying (3.2)—so that termination in line 22 would occur—in at most the number of iterations given by (3.4). Of course, if termination occurs earlier in line 18, the bound (3.4) still holds. ∎

When employing the trust-region method of section 4 for minimizing f, Algorithm 3.1 is invoked without knowing whether or not $H + 2\epsilon I \succ \epsilon I$. Nevertheless, Lemma 3.1 allows us to make the following crucial observation.

LEMMA 3.2. *If the iteration limit in Algorithm* 3.1 *is exceeded (that is, termination occurs at line* 28*), then* $H \not\succ -\epsilon I$.

Proof. If $k_{\max}$ is set to n in line 4 or line 6, then it would follow from standard CG theory that Algorithm 3.1 cannot reach line 28, because either $r_{j+1} = 0$ for some $j < n$ (thus termination would have occurred at line 21) or else one of the other termination conditions would have been activated before this point. Hence, $k_{\max}$ must have been set in line 4 to some value less than n. In this case, it follows from Lemma 3.1, the choice of M, and the choice of $k_{\max}$ in line 4 that $H + 2\epsilon I \not\succ \epsilon I$. ∎

When Algorithm 3.1 returns because the inequality in line 10 holds, it is possible that the objective function in (3.1) evaluated at the returned vector s is larger than

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 3.1. Truncated CG Method for the Trust-Region Subproblem

1: **Input:** Nonzero $g \in \mathbb{R}^n$; $H \in \mathbb{S}^n$; regularization parameter $\epsilon \in (0, \infty)$; trust-region radius $\delta \in (0, \infty)$; accuracy parameter $\zeta \in (0, 1)$; flag `capCG` $\in \{\text{TRUE}, \text{FALSE}\}$; and (if `capCG` = TRUE) upper bound $M \in [\|H\|, \infty)$.
2: **Output:** trial step s and flag `outCG` indicating termination type.

3: **if** `capCG` = TRUE **then**
4: Set $k_{\max} \leftarrow \min\left\{n, \frac{1}{2}\sqrt{\kappa}\ln\left(4\kappa^{3/2}/\zeta\right)\right\}$ where $\kappa \leftarrow (M + 2\epsilon)/2$.
5: **else**
6: Set $k_{\max} \leftarrow n$.
7: **end if**
8: Set $y_0 \leftarrow 0$, $r_0 \leftarrow g$, $p_0 \leftarrow -g$, and $j \leftarrow 0$.
9: **while** $j < k_{\max}$ **do**
10: **if** $p_j^T(H + 2\epsilon I)p_j \leq \epsilon\|p_j\|^2$ **then**
11: Compute $\sigma \geq 0$ such that $\|y_j + \sigma\, p_j\| = \delta$.
12: **return** $s \leftarrow y_j + \sigma\, p_j$ and `outCG` $\leftarrow$ BND-NEG.
13: **end if**
14: Set $\alpha_j \leftarrow \|r_j\|^2/(p_j^T(H + 2\epsilon I)p_j)$.
15: Set $y_{j+1} \leftarrow y_j + \alpha_j p_j$.
16: **if** $\|y_{j+1}\| \geq \delta$ **then**
17: Compute $\sigma \geq 0$ such that $\|y_j + \sigma\, p_j\| = \delta$.
18: **return** $s \leftarrow y_j + \sigma\, p_j$ and `outCG` $\leftarrow$ BND-NORM.
19: **end if**
20: Set $r_{j+1} \leftarrow r_j + \alpha_j(H + 2\epsilon I)p_j$.
21: **if** $\|r_{j+1}\| \leq \frac{\zeta}{2}\min\{\|g\|, \epsilon\|y_{j+1}\|\}$ **then**
22: **return** $s \leftarrow y_{j+1}$ and `outCG` $\leftarrow$ INT-RES.
23: **end if**
24: Set $\beta_{j+1} \leftarrow (r_{j+1}^T r_{j+1})/(r_j^T r_j)$.
25: Set $p_{j+1} \leftarrow -r_{j+1} + \beta_{j+1}p_j$.
26: Set $j \leftarrow j + 1$.
27: **end while**
28: **return** $s \leftarrow y_{k_{\max}}$ and `outCG` $\leftarrow$ INT-MAX.

its value at $s = 0$, a situation that is typically not possible when CG is used as a subproblem solver in trust-region methods. This is because, although the inequality in line 10 implies that p_j is a direction of negative curvature for H, p_j is not necessarily a direction of negative curvature for the matrix $H + 2\epsilon I$ that defines the quadratic model in (3.1). Since, in this case, s is obtained by moving to the boundary of the trust-region along the direction p_j (similar to the behavior of Steihaug's CG method in [31] and needed for our complexity result), we require the following result, which establishes that any step computed by Algorithm 3.1 possesses a decrease property with respect to the *nonregularized* version of the quadratic model.

LEMMA 3.3. *The step s returned by Algorithm* 3.1 *satisfies*

$$g^T s + \frac{1}{2}s^T H s \leq -\frac{1}{2}\epsilon\|s\|^2.$$

Proof. Basic CG theory ensures that for any j up to termination, the sequence $\{g^T y_j + \frac{1}{2}y_j^T(H + 2\epsilon I)y_j\}$ is monotonically decreasing. Since $y_0 = 0$, we thus have

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

$$g^T y_i + \frac{1}{2} y_i^T (H + 2\epsilon I) y_i \leq 0 \quad \text{for all} \quad i \in \{0, 1, \ldots, j\}. \tag{3.5}$$

Suppose outCG $\in$ {BND-NORM, INT-RES, INT-MAX}. From (3.5) and the fact (by [31, Theorem 2.1]) that $g^T s + \frac{1}{2} s^T (H + 2\epsilon I) s \leq g^T y_j + \frac{1}{2} y_j^T (H + 2\epsilon I) y_j$ when outCG $\leftarrow$ BND-NORM, we have

$$g^T s + \frac{1}{2} s^T (H + 2\epsilon I) s \;\leq\; 0 \;\Leftrightarrow\; g^T s + \frac{1}{2} s^T H s \;\leq\; -\epsilon \|s\|^2,$$

which implies the desired result.

Second, suppose that outCG $\leftarrow$ BND-NEG, meaning that Algorithm 3.1 terminates because iteration j yields $p_j^T (H + 2\epsilon I) p_j \leq \epsilon \|p_j\|^2$. If $j = 0$, then the fact that $p_0 = -g$ allows us to conclude that $s = \delta(p_0/\|p_0\|) = -\delta(g/\|g\|)$, $\|s\| = \delta$, and

$$\frac{1}{2} s^T (H + 2\epsilon I) s = \frac{1}{2} \delta^2 \left(p_0^T (H + 2\epsilon I) p_0 \right) / \|p_0\|^2 \leq \frac{1}{2} \epsilon \delta^2 = \frac{1}{2} \epsilon \|s\|^2,$$

from which it follows that

$$g^T s + \frac{1}{2} s^T H s = -\delta \|g\| + \frac{1}{2} s^T (H + 2\epsilon I) s - \epsilon \|s\|^2 \leq -\frac{1}{2} \epsilon \|s\|^2,$$

as desired. On the other hand, if $j \geq 1$, then the fact that outCG $\leftarrow$ BND-NEG means that $s \leftarrow y_j + \sigma p_j$ with $\sigma \geq 0$ such that $\|s\| = \delta$. The CG process yields the following:

$$y_i = \sum_{\ell=0}^{i-1} \alpha_\ell p_\ell \in \operatorname{span}\{p_0, \ldots, p_{i-1}\} \quad \text{for all } i \in \{1, 2, \ldots, j\}, \tag{3.6a}$$

$$p_i^T (H + 2\epsilon I) p_\ell = 0 \quad \text{for all } \{i, \ell\} \subseteq \{0, 1, \ldots, j\} \text{ with } i \neq \ell, \tag{3.6b}$$

$$r_i^T p_j = -\|r_j\|^2 \quad \text{for all } i \in \{0, 1, \ldots, j\}, \tag{3.6c}$$

$$\text{and} \quad y_i^T p_i \geq 0 \quad \text{for all } i \in \{0, 1, \ldots, j\}. \tag{3.6d}$$

(Referring to Algorithm 3.1, the property (3.6a) follows from line 15; (3.6b) is the well known conjugacy property; and (3.6c) is obtained by successively substituting for $p_j, p_{j-1}, \ldots, p_{i+1}$ from line 25, using the property that $r_i^T r_l = 0$ for $l \neq i$, and using the definition of β_j from line 24. For (3.6d), see [31, equation (2.13)].) Together, (3.6) and $s = y_j + \sigma p_j$ imply

$$g^T p_j = r_0^T p_j = -\|r_j\|^2 \leq 0, \tag{3.7a}$$

$$s^T (H + 2\epsilon I) s = y_j^T (H + 2\epsilon I) y_j + \sigma^2 p_j^T (H + 2\epsilon I) p_j, \tag{3.7b}$$

$$\text{and} \quad \|s\|^2 = \|y_j\|^2 + 2\sigma y_j^T p_j + \sigma^2 \|p_j\|^2 \geq \sigma^2 \|p_j\|^2. \tag{3.7c}$$

Combining (3.5), (3.7), $\sigma \geq 0$, and $p_j^T (H + 2\epsilon I) p_j \leq \epsilon \|p_j\|^2$ shows that

$$\begin{aligned} g^T s + \frac{1}{2} s^T H s &= g^T s + \frac{1}{2} s^T (H + 2\epsilon I) s - \epsilon \|s\|^2 \\ &= g^T y_j + \frac{1}{2} y_j^T (H + 2\epsilon I) y_j + \sigma g^T p_j + \frac{1}{2} \sigma^2 p_j^T (H + 2\epsilon I) p_j - \epsilon \|s\|^2 \\ &\leq \frac{1}{2} \sigma^2 p_j^T (H + 2\epsilon I) p_j - \epsilon \|s\|^2 \leq \frac{1}{2} \sigma^2 \epsilon \|p_j\|^2 - \epsilon \|s\|^2 \leq -\frac{1}{2} \epsilon \|s\|^2, \end{aligned}$$

which completes the proof. ∎

Lemma 3.3 shows that if $\epsilon = \epsilon_H$, then the bound on the model decrease obtained by the truncated CG step s is the same as the bound guaranteed by the global solution computed for Algorithm 2.1 (see Lemma 2.1). However, we note that this decrease is obtained by using a larger regularization term.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 3.2. Minimum Eigenvalue Oracle (MEO)

Input: $g \in \mathbb{R}^n$; $H \in \mathbb{S}^n$; regularization parameter $\epsilon \in (0, \infty)$; trust-region radius $\delta \in (0, \infty)$; failure probability tolerance $\xi \in (0, 1)$; and $M \in [\|H\|, \infty)$.
Output: Either (i) a vector $s = \pm\delta v$ satisfying

$$g^T s \leq 0, \quad s^T H s \leq -\frac{1}{2}\epsilon\|s\|^2, \quad \text{and} \quad \|s\| = \delta, \tag{3.8}$$

where v has been computed to satisfy $\|v\| = 1$ and $v^T H v \leq -\epsilon/2$, or (ii) an indication that $H \succeq -\epsilon I$ holds. The probability that the indication in case (ii) is made yet $H \prec -\epsilon I$ is at most ξ. (The bound M may be needed for algorithm termination; see Assumption 2 on page 532.)

3.2. A minimum eigenvalue oracle. The truncated CG algorithm presented in section 3.1 is only one of the tools we need for our proposed inexact trust-region Newton method. Two complicating cases require an additional tool.

The first case is when `outCG` = INT-MAX is returned by Algorithm 3.1. In this case, it must hold that the maximum allowed number of iterations satisfies $k_{\max} < n$ and, as a consequence of Lemma 3.2, that $H \not\succ -\epsilon I$. Thus, there exists a direction of sufficient negative curvature for H, and we need a means of computing one. The second case is when Algorithm 3.1 terminates with `outCG` = INT-RES. In this case, we only know that the curvature is not sufficiently negative along the directions computed by the algorithm. However, it may still be true that $H \not\succ -\epsilon I$.

These two cases motivate the need for a *minimum eigenvalue oracle* that estimates the minimum eigenvalue of H or else returns an indication that (with some desired probability) no sufficiently negative eigenvalue exists. The oracle that we employ is given by Algorithm 3.2.

4. An inexact trust-region Newton method. In this section, we propose a trust-region Newton method that may use, during each iteration, an *inexact* solution to the trust-region subproblem computed using the iterative procedures described in section 3. The proposed algorithm is described in section 4.1, and a second-order complexity analysis is presented in section 4.2.

4.1. The algorithm. Algorithm 4.1 can be viewed as an inexact version of Algorithm 2.1. We aim at remaining close to the traditional Newton-CG approaches in [31, 32] by having Algorithm 4.1 compute, when appropriate, a truncated CG step in line 4. Once such a step is computed (or set to zero since the current iterate is first-order stationary), Algorithm 4.1 deviates from traditional Newton-CG in the "else" branch (line 10), which accounts for the two situations described in section 3.2, where an additional check for a negative curvature direction is needed. (There is one minor difference: when `outCG` = INT-RES, the MEO need be called only when $\|g_k\| \leq \epsilon_g$.)

4.2. Complexity. As in [29], we make the following assumption on the MEO in order to obtain complexity results for Algorithm 4.1.

Assumption 2. When Algorithm 3.2 is called by Algorithm 4.1, the number of Hessian-vector products required is no more than

$$N_{\text{meo}} = N_{\text{meo}}(\epsilon_H) := \min\left\{n, 1 + \left\lceil \mathcal{C}_{\text{meo}} \epsilon_H^{-1/2} \right\rceil\right\}, \tag{4.1}$$

where the quantity $\mathcal{C}_{\text{meo}}$ depends at most logarithmically on ξ.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Algorithm 4.1. Trust-Region Newton-CG Method (inexact version)

Require: Tolerances $\epsilon_g \in (0,\infty)$ and $\epsilon_H \in (0,\infty)$; trust-region adjustment parameters $\gamma_1 \in (0,1)$, $\gamma_2 \in [1,\infty)$, and $\psi \in (1/\gamma_2, 1]$; initial iterate $x_0 \in \mathbb{R}^n$; initial trust-region radius $\delta_0 \in (0,\infty)$; maximum trust-region radius $\delta_{\max} \in [\delta_0,\infty)$; step acceptance parameter $\eta \in (0,1)$; truncated CG accuracy parameter $\zeta \in (0,1)$; MEO failure probability tolerance $\xi \in [0,1)$; flag `capCG` $\in \{\text{TRUE}, \text{FALSE}\}$; and upper bound $M \in [L_g,\infty)$.

for $k = 0, 1, 2, \ldots$ **do**
  Evaluate g_k and H_k.
  if $g_k \neq 0$ **then**
    Call Algorithm 3.1 with input $g = g_k$, $H = H_k$, $\epsilon = \epsilon_H$, $\delta = \delta_k$, ζ, `capCG`, and (if `capCG` = TRUE) M to compute s_k^{CG} and output flag `outCG`.
  else
    Set $s_k^{\mathrm{CG}} \leftarrow 0$ and `outCG` $\leftarrow$ INT-RES.
  end if
  if `outCG` $\in \{$BND-NEG, BND-NORM$\}$ or ($\|g_k\| > \epsilon_g$ and `outCG` = INT-RES) **then**
    Set $s_k \leftarrow s_k^{\mathrm{CG}}$.
  else {that is, `outCG` = INT-MAX or ($\|g_k\| \leq \epsilon_g$ and `outCG` = INT-RES)}
    Call Algorithm 3.2 with inputs $g = g_k$, $H = H_k$, $\epsilon = \epsilon_H$, $\delta = \delta_k$, ξ, and M, obtaining either s_k satisfying (3.8) or an indication that $H_k \succeq -\epsilon_H I$.
    if Algorithm 3.2 predicts that $H_k \succeq -\epsilon_H I$ **then**
      return x_k.
    end if
  end if
  Compute the ratio of actual to predicted decrease in f defined as

$$\rho_k \leftarrow \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}.$$

  if $\rho_k \geq \eta$ **then**
    Set $x_{k+1} \leftarrow x_k + s_k$.
    if $\|s_k\| \geq \psi\delta_k$ **then**
      Set $\delta_{k+1} \leftarrow \min\{\gamma_2\delta_k, \delta_{\max}\}$.
    else
      Set $\delta_{k+1} \leftarrow \delta_k$.
    end if
  else
    Set $x_{k+1} \leftarrow x_k$ and $\delta_{k+1} \leftarrow \gamma_1\|s_k\|$.
  end if
end for

The following instances of Algorithm 3.2 satisfy Assumption 2.

- The *Lanczos algorithm* applied to H starting with a random vector uniformly distributed on the unit sphere. For any $\xi \in (0,1)$, this satisfies the conditions in Assumption 2 with $\mathcal{C}_{\mathrm{meo}} = \ln(2.75n/\xi^2)\sqrt{M}/2$; see [29, Lemma 2].
- The *CG algorithm* applied to $\left(H + \frac{\epsilon_H}{2} I\right) s = b$, where b is a random vector uniformly distributed on the unit sphere. For any $\xi \in (0,1)$, this offers Assumption 2 with the same value of $\mathcal{C}_{\mathrm{meo}}$ as in the Lanczos-based approach; see [29, Theorem 1].

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Since for each instance the conditions of Assumption 2 hold with $\mathcal{C}_{\text{meo}}$ equal to the given value, it follows that throughout a run of Algorithm 4.1, the conditions hold with $\mathcal{C}_{\text{meo}} = \ln(2.75n/\xi^2)\sqrt{L_g}/2$. Algorithm 3.2 could also be implemented by means of an exact (minimum) eigenvalue calculation of the Hessian. In that case, up to n Hessian-vector products may be required to evaluate the full Hessian.

In the following analysis, we use similar notation as in section 2, although the analysis here is notably different due to the randomness of the MEO. For consistency, we use the same definitions of the index sets $\mathcal{K}$, $\mathcal{I}$, $\mathcal{B}$, $\mathcal{S}$, and $\mathcal{U}$ that appear in the beginning of section 2.2; in particular, we define $\mathcal{K}$ as the index set of iterations completed prior to termination. However, note that for Algorithm 4.1 these sets are random variables, in the sense that for the same objective function and algorithm inputs, they may have different realizations due to the randomness in Algorithm 3.2. Thus, when we refer, for example, to $k \in \mathcal{K}$, we are referring to $k \in \mathcal{K}$ *for a given realization of a run of Algorithm* 4.1. We also prove bounds on quantities that are shown to hold for *all* realizations of a run of the algorithm (for a given objective function and algorithm inputs). To emphasize that these bounds hold for all realizations, their constants are written with a bar over the letter in the definition.

Our first result provides a lower bound on the reduction in the quadratic model of the objective function achieved by each trial step.

LEMMA 4.1. *Consider any realization of a run of Algorithm* 4.1. *For all* $k \in \mathcal{K}$,

$$m_k(x_k) - m_k(x_k + s_k) \geq \frac{1}{4}\epsilon_H \|s_k\|^2.$$

Proof. If $s_k = s_k^{\text{CG}}$, where s_k^{CG} is computed from Algorithm 3.1, then it follows by Lemma 3.3 that the desired bound holds. Now suppose that s_k is computed from Algorithm 3.2 in line 11. Since $k \in \mathcal{K}$, Algorithm 4.1 does not terminate in iteration k, and it follows from (3.8) that

$$m_k(x_k) - m_k(x_k + s_k) = -g_k^T s_k - \frac{1}{2}s_k^T H_k s_k \geq -\frac{1}{2}s_k^T H_k s_k \geq \frac{1}{4}\epsilon_H \|s_k\|^2,$$

as desired. ∎

We can now show that a sufficiently small trust-region radius leads to a successful iteration and provide a lower bound on the sequence of trust-region radii.

LEMMA 4.2. *Consider any realization of a run of Algorithm* 4.1. *For all* $k \in \mathcal{K}$, *if* $k \in \mathcal{U}$, *then* $\delta_k > 3(1-\eta)\epsilon_H/(2L_H)$. *Hence, by the trust-region radius update procedure, it follows that for any realization of a run of Algorithm* 4.1 *that*

$$\delta_k \geq \bar{\delta}_{\min} := \min\left\{\delta_0, \left(\frac{3\gamma_1(1-\eta)}{2L_H}\right)\epsilon_H\right\} \in (0,\infty) \quad \textit{for all} \quad k \in \mathcal{K}. \tag{4.2}$$

Proof. For any realization of a run of the algorithm, we can follow the proof of Lemma 2.2, using Lemma 4.1 in lieu of Lemma 2.1. Hence, the lower bound in (4.2) holds, where $\bar{\delta}_{\min}$ is independent of any particular realization of a run. ∎

We now establish a bound on the objective reduction for a successful step.

LEMMA 4.3. *Consider any realization of a run of Algorithm* 4.1. *The following hold for all successful iterations:*

(i) *If* $k \in \mathcal{B} \cap \mathcal{S}$, *then*

$$f_k - f_{k+1} \geq \frac{\eta}{4}\epsilon_H \delta_k^2.$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

(ii) *If $k \in \mathcal{I} \cap \mathcal{S}$, then $\|g_k\| > \epsilon_g$,* `outCG` = INT-RES, *and*

$$f_k - f_{k+1} \geq \frac{\eta}{4(7+2L_H)} \min\left\{\|g_{k+1}\|^2 \epsilon_H^{-1}, \epsilon_H^3\right\}.$$

Proof. For part (i), we combine $k \in \mathcal{B} \cap \mathcal{S}$ with Lemma 4.1 to obtain, as desired,

$$f_k - f_{k+1} \geq \eta\left(m_k(x_k) - m_k(x_k + s_k)\right) \geq \frac{\eta}{4}\epsilon_H \|s_k\|^2 = \frac{\eta}{4}\epsilon_H \delta_k^2.$$

Now consider part (ii). Note that since $k \in \mathcal{I}$, s_k cannot have been computed from a call to Algorithm 3.2 in line 11, since such steps always have $\|s_k\| = \delta_k$. Thus, $s_k = s_k^{\mathrm{CG}}$. Moreover, from line 8 and the fact that $k \in \mathcal{I}$, we have that $\|g_k\| > \epsilon_g$ and `outCG` = INT-RES, as desired. In turn, the fact that `outCG` = INT-RES implies that (3.2) holds with $H = H_k$, $g = g_k$, $s = s_k$, and $\epsilon = \epsilon_H$ so that

$$r_k := (H_k + 2\epsilon_H)s_k + g_k \ \text{ has } \ \|r_k\| \leq \frac{\zeta}{2}\epsilon_H \|s_k\|. \tag{4.3}$$

Combining this bound with (1.4b) and $\zeta \in (0,1)$, we have

$$\begin{aligned}\|g_{k+1}\| &= \|g_{k+1} - g_k - (H_k + 2\epsilon_H)s_k + r_k\| \\ &\leq \|g_{k+1} - g_k - H_k s_k\| + 2\epsilon_H\|s_k\| + \|r_k\| \\ &\leq \frac{L_H}{2}\|s_k\|^2 + \left(\frac{4+\zeta}{2}\right)\epsilon_H\|s_k\| \leq \frac{L_H}{2}\|s_k\|^2 + \frac{5}{2}\epsilon_H\|s_k\|,\end{aligned}$$

which can be rearranged to yield

$$\frac{L_H}{2}\|s_k\|^2 + \frac{5}{2}\epsilon_H\|s_k\| - \|g_{k+1}\| \geq 0.$$

Reasoning as in the proof of Lemma 2.3, with $\frac{5}{2}\epsilon_H$ replacing ϵ_H, we obtain

$$\|s_k\| \geq \frac{-\frac{5}{2}\epsilon_H + \sqrt{\left(\frac{5}{2}\right)^2\epsilon_H^2 + 2L_H\|g_{k+1}\|}}{L_H} = \left(\frac{-5 + \sqrt{25 + 8L_H\|g_{k+1}\|\epsilon_H^{-2}}}{2L_H}\right)\epsilon_H.$$

By setting $(a,b,t) = (5, 8L_H, \|g_{k+1}\|\epsilon_H^{-2})$ in the inequality (2.8), we have

$$\begin{aligned}\|s_k\| &\geq \left(\frac{-5+\sqrt{25+8L_H}}{2L_H}\right)\min\left\{\|g_{k+1}\|\epsilon_H^{-2}, 1\right\}\epsilon_H \\ &= \left(\frac{8L_H}{2L_H(5+\sqrt{25+8L_H})}\right)\min\left\{\|g_{k+1}\|\epsilon_H^{-1}, \epsilon_H\right\} \\ &= \left(\frac{4}{5+\sqrt{25+8L_H}}\right)\min\left\{\|g_{k+1}\|\epsilon_H^{-1}, \epsilon_H\right\} \\ &\geq \left(\frac{2}{\sqrt{25+8L_H}}\right)\min\left\{\|g_{k+1}\|\epsilon_H^{-1}, \epsilon_H\right\} \geq \frac{1}{\sqrt{7+2L_H}}\min\left\{\|g_{k+1}\|\epsilon_H^{-1}, \epsilon_H\right\},\end{aligned}$$

which may be combined with $k \in \mathcal{I} \cap \mathcal{S}$ and Lemma 4.1 to obtain

$$f_k - f_{k+1} \geq \eta(m_k(x_k) - m_k(x_{k+1})) \geq \frac{1}{4}\eta\epsilon_H\|s_k\|^2 \geq \frac{\eta}{4(7+2L_H)}\min\left\{\|g_{k+1}\|^2\epsilon_H^{-1}, \epsilon_H^3\right\},$$

which completes the proof. ∎

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

The next result is analogous to Lemma 2.4 and takes randomness in the MEO into account.

LEMMA 4.4. *For any realization of a run of Algorithm* 4.1*, the number of successful iterations performed before termination occurs satisfies*

$$|\mathcal{S}| \leq \bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H) := \left\lfloor \bar{\mathcal{C}}_{\mathcal{S}} \max\left\{\epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\right\} \right\rfloor + 1, \tag{4.4}$$

where

$$\bar{\mathcal{C}}_{\mathcal{S}} := \frac{8(f_0 - f_{\text{low}})}{\eta} \max\left\{ \frac{1}{\delta_0^2}, \frac{4L_H^2}{9\gamma_1^2(1-\eta)^2}, 7 + 2L_H) \right\}. \tag{4.5}$$

Proof. For a given realization of a run of the algorithm, we can follow the reasoning of the proof for Lemma 2.4. In what follows, $\mathcal{S}_L$, $\mathcal{S}_{GG}$, and $\mathcal{S}_{GL}$ are defined as in the proof of Lemma 2.4. (As is the case for $\mathcal{K}$, we note that these index sets are now realizations of random index sets.)

Consider first $k \in \mathcal{S}_L$, and let us define the constant

$$c_1 := \frac{\eta}{4} \min\left\{ \delta_0^2, \frac{9\gamma_1^2(1-\eta)^2}{4L_H^2}, \frac{1}{7+2L_H} \right\}. \tag{4.6}$$

We can use Lemma 4.3 to conclude that $k \in \mathcal{K} \cap \mathcal{B}$, so that $\|s_k\| = \delta_k$. By combining Lemma 4.3(i), Lemma 4.2, and (4.6), we have for $k \in \mathcal{S}_L$ that

$$f_k - f_{k+1} \geq \frac{\eta}{4}\epsilon_H \delta_k^2 \geq \frac{\eta}{4} \min\left\{ \delta_0^2 \epsilon_H, \frac{9\gamma_1^2(1-\eta)^2}{4L_H^2}\epsilon_H^3 \right\} \geq c_1 \min\left\{\epsilon_H, \epsilon_H^3\right\}. \tag{4.7}$$

For $k \in \mathcal{S}_{GG}$, we have from Lemma 4.3 (either (i) or (ii)) and Lemma 4.2 that

$$\begin{aligned} f_k - f_{k+1} &\geq \frac{\eta}{4} \min\left\{ \delta_0^2\epsilon_H, \frac{9\gamma_1^2(1-\eta)^2}{4L_H^2}\epsilon_H^3, \frac{1}{7+2L_H}\epsilon_g^2\epsilon_H^{-1}, \frac{1}{7+2L_H}\epsilon_H^3 \right\} \\ &\geq c_1 \min\left\{\epsilon_H, \epsilon_H^3, \epsilon_g^2\epsilon_H^{-1}\right\}. \end{aligned} \tag{4.8}$$

By following the reasoning that led to (2.14), we obtain from (4.7) and (4.8) that

$$|\mathcal{S}_L| + |\mathcal{S}_{GG}| \leq \left(\frac{f_0 - f_{\text{low}}}{c_1} \right) \max\left\{\epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\right\}.$$

As in the proof of Lemma 2.4, we have that $|\mathcal{S}_{GL}| \leq |\mathcal{S}_L| + 1$, so that

$$|\mathcal{S}| = |\mathcal{S}_L| + |\mathcal{S}_{GG}| + |\mathcal{S}_{GL}| \leq \frac{2(f_0 - f_{\text{low}})}{c_1} \max\left\{\epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\right\} + 1.$$

The desired bound follows by substituting the definition (4.6) into this bound. To complete the proof, we note that the right-hand side of (4.4) is identical for any realization of the algorithm run with the same inputs. ∎

We now provide a bound on the maximum number of unsuccessful iterations.

LEMMA 4.5. *For any realization of a run of Algorithm* 4.1*, the number of unsuccessful iterations performed before termination occurs either is zero or else satisfies*

$$|\mathcal{U}| \leq \left\lfloor 1 + \log_{\gamma_1}\left(\frac{3(1-\eta)}{2L_H\delta_{\max}} \right) + \log_{\gamma_1}(\epsilon_H) \right\rfloor (|\mathcal{S}| + 1). \tag{4.9}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Proof. For a given realization of a run of the algorithm, the bound follows from the argument in the proof of Lemma 2.5 with two changes. First, Lemma 4.2 is used in place of Lemma 2.2. Second, we do not know that the iteration immediately prior to termination must be a successful iteration for Algorithm 4.1, as was the case for Algorithm 2.1. However, using the argument in the proof of Lemma 2.5 along with Lemma 4.2 shows that if a sequence of consecutive unsuccessful iterations is taken after the final successful iteration, there can be no more than

$$\left\lfloor 1 + \log_{\gamma_1}\left(\frac{3(1-\eta)}{2L_H\delta_{\max}}\right) + \log_{\gamma_1}(\epsilon_H)\right\rfloor$$

such iterations in this sequence, since otherwise an additional successful iteration would be performed. Taking this fact into account leads to the extra 1 on the right-hand side of (4.9) as compared to (2.15). ∎

We assume in our remaining complexity results that the following common-sense rule is used in the implementation of Algorithm 4.1.

IMPLEMENTATION STRATEGY 4.1. *For any realization of a run of Algorithm* 4.1*, suppose $k \in \mathcal{K}$ is an index of an iteration such that* (i) *Algorithm* 3.2 *is called in line* 11 *and returns a negative curvature direction s_k for H_k and* (ii) *the step s_k is subsequently rejected (that is, $k \in \mathcal{U}$). Then, the negative curvature direction (call it $v = v_k$) used to compute s_k is stored and used until the next successful iteration. Until then, every call to Algorithm* 3.2 *is replaced by an access to v_k, scaled appropriately to compute s_k with norm δ_k.*

This strategy implies that Algorithm 4.1 cannot terminate following a sequence of unsuccessful iterations if any one of them yields a direction of sufficiently negative curvature. In practice, this means that Algorithm 4.1 calls Algorithm 3.2 at most once between successful iterations. In the next iteration complexity result, this assumption is used to obtain the probabilistic result for returning an (ϵ_g, ϵ_H)-stationarity point.

THEOREM 4.6. *Under Assumption* 1*, for any realization of a run, the number of successful iterations* (*and objective gradient evaluations*) *performed by Algorithm* 4.1 *before termination occurs satisfies* (*with $\bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H)$ defined in* (4.4))

$$|\mathcal{S}| \le \bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H) = \mathcal{O}\left(\max\left\{\epsilon_H^{-3}, \epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H\right\}\right), \tag{4.10}$$

and the total number of iterations (and objective function evaluations) performed before termination occurs satisfies

$$\begin{aligned} |\mathcal{K}| &\le \left\lfloor 1 + \log_{\gamma_1}\left(\frac{3(1-\eta)}{2L_H\delta_{\max}}\right) + \log_{\gamma_1}(\epsilon_H)\right\rfloor \left(\bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H) + 1\right) \\ &= \mathcal{O}\left(\log_{1/\gamma_1}(\epsilon_H^{-1}) \max\left\{\epsilon_H^{-3}, \epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H\right\}\right). \end{aligned} \tag{4.11}$$

If `capCG` = FALSE*, then $\|g_k\| \le \epsilon_g$ holds at termination. In any case, given Implementation Strategy* 4.1*, the vector x_k returned by Algorithm* 4.1 *is an (ϵ_g, ϵ_H)-stationary point with probability at least $(1-\xi)^{\bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H)}$.*

Proof. The results in (4.10) and (4.11) follow from Lemma 4.4 and Lemma 4.5. If `capCG` = FALSE, then the flag output by Algorithm 3.1 has `outCG` ≠ INT-MAX. Combining this fact with line 8 of Algorithm 4.1 allows us to conclude that $\|g_k\| \le \epsilon_g$ when termination occurs in this case.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Suppose that Implementation Strategy 4.1 is used. Then, the vector x_k returned by Algorithm 4.1 is not an (ϵ_g, ϵ_H)-stationary point only if the MEO (Algorithm 3.2) makes an inaccurate indication of near-positive-definiteness, which, each time it is called, can occur with probability at most ξ. Our goal now is to prove that the vector x_k returned by Algorithm 4.1 is an (ϵ_g, ϵ_H)-stationary point with probability at least $(1-\xi)^{\bar{K}_{\mathcal{S}}(\epsilon_g,\epsilon_H)}$. To that end, for all $k \in \mathbb{N}$, let $\tilde{P}_k$ be the probability that the algorithm reaches iteration k and x_k is not (ϵ_g, ϵ_H)-stationary. Similarly, for all $k \in \mathbb{N}$, let P_k be the probability that the algorithm reaches iteration k and x_k is not (ϵ_g, ϵ_H)-stationary, yet the algorithm terminates in iteration k due to an inaccurate indication from the MEO. It follows from the aforementioned property of the MEO in this setting that $P_k \leq \xi \tilde{P}_k$ for all $k \in \mathbb{N}$. Thus, since it is trivially true that

$$\tilde{P}_k + \sum_{i=0}^{k-1} P_i \leq 1 \quad \text{for all} \quad k \in \mathbb{N},$$

it follows that

$$P_k \leq \xi \tilde{P}_k \leq \xi \left(1 - \sum_{i=0}^{k-1} P_i\right) \quad \text{for all} \quad k \in \mathbb{N}. \tag{4.12}$$

We now define M_k to be the number of calls to the MEO that have occurred up to and including iteration k for any $k \in \mathbb{N}$. Let us now prove by induction that $\sum_{i=0}^{k} P_i \leq 1 - (1-\xi)^{M_k}$ for all $k \in \mathbb{N}$. For $k = 0$, the claim holds trivially both when $M_0 = 0$ (in which case $P_0 = 0$) and when $M_0 = 1$ (in which case $P_0 \leq \xi$). Now suppose that the claim is true for some $k \in \mathbb{N}$; we aim to prove that it remains true for $k+1$. If the algorithm reaches iteration $k+1$ in which x_{k+1} is not (ϵ_g, ϵ_H)-stationary and the MEO *not* called in iteration $k+1$, then $M_{k+1} = M_k$ and $P_{k+1} = 0$, so by the induction hypothesis it follows that

$$\sum_{i=0}^{k+1} P_i = \sum_{i=0}^{k} P_i \leq 1 - (1-\xi)^{M_k} = 1 - (1-\xi)^{M_{k+1}},$$

as desired. On the other hand, if the algorithm reaches iteration $k+1$, the iterate x_{k+1} is not (ϵ_g, ϵ_H)-stationary, and the MEO *is* called in iteration $k+1$, then $M_{k+1} = M_k + 1$, and along with (4.12) and the induction hypothesis it follows that

$$\begin{aligned}
\sum_{i=0}^{k+1} P_i &= \sum_{i=0}^{k} P_i + P_{k+1} \\
&\leq \sum_{i=0}^{k} P_i + \xi \left(1 - \sum_{i=0}^{k} P_i\right) \\
&= \xi + (1-\xi) \sum_{i=0}^{k} P_i \\
&\leq \xi + (1-\xi)\left(1 - (1-\xi)^{M_k}\right) \\
&= 1 - (1-\xi)^{M_k+1} = 1 - (1-\xi)^{M_{k+1}},
\end{aligned}$$

as desired, again. Since $|\mathcal{S}|$ is an upper bound on the number of successful steps, which in turn bounds the number of calls to MEO when Implementation Strategy 4.1

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

is used, we have $M_k \leq |\mathcal{S}| \leq \bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H)$ for all k. We conclude that the probability that the algorithm terminates incorrectly due to an incorrect indication from MEO on *any* iteration is bounded above by $1-(1-\xi)^{\bar{K}_{\mathcal{S}}(\epsilon_g,\epsilon_H)}$. Consequently, the probability that the algorithm outputs an (ϵ_g, ϵ_H)-stationary point is at least $(1-\xi)^{\bar{K}_{\mathcal{S}}(\epsilon_g,\epsilon_H)}$, as claimed. ∎

Finally, we state a complexity result for the number of Hessian-vector products. For simplicity, we focus on the case of a small tolerance ϵ_g.

THEOREM 4.7. *Let Assumption* 1 *and Assumption* 2 *hold, and suppose that Implementation Strategy* 4.1 *is used. Suppose that* $\epsilon_H = \epsilon_g^{1/2}$ *with* $\epsilon_g \in (0,1)$. *Then, for any realization of a run of Algorithm* 4.1*, the total number of Hessian-vector products performed satisfies the following:*

(i) *If* `capCG` = FALSE*, then the number of Hessian-vector products is bounded by*

$$n|\mathcal{K}| + N_{\text{meo}}(\epsilon_H)|\bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H)| = n\tilde{\mathcal{O}}(\epsilon_g^{-3/2}).$$

(ii) *If* `capCG` = TRUE*, then the number of Hessian-vector products is bounded by*

$$\min\{n, J(L_g, \epsilon_H, \zeta)\}|\mathcal{K}| + N_{\text{meo}}(\epsilon_H)|\bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H)| = \min\left\{n, \epsilon_g^{-1/4}\right\}\tilde{\mathcal{O}}\left(\epsilon_g^{-3/2}\right).$$

Proof. First, suppose `capCG` = FALSE. Then, for any $k \in \mathcal{K}$ in any realization of a run of the algorithm, the maximum number of Hessian-vector products computed by the truncated CG algorithm is n. In addition, over any realization of a run, the maximum number of Hessian-vector products computed by the MEO is $N_{\text{meo}}(\epsilon_H)$ (see (4.1)) each of the (at most) $|\bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H)|$ times it is called. Since the number of Hessian-vector products performed by Algorithm 4.1 is the sum of these two, we have proved the left-hand side of part (i). For the estimate $n\tilde{\mathcal{O}}(\epsilon_g^{-3/2})$, we use $\epsilon_H = \epsilon_g^{1/2}$, the bound on $\mathcal{K}$ from Theorem 4.6, the estimate of N_{meo} from Assumption 2, and the fact that $\max\{\epsilon_H^{-3}, \epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H\} = \max\{\epsilon_g^{-3/2}, \epsilon_g^{-1/2}\} = \epsilon_g^{-3/2}$ when $\epsilon_g \in (0,1)$.

For part (ii), we use the same estimates as well as the estimate of $J(L_g, \epsilon_H, \zeta)$ from Lemma 3.1 and (1.4c), noting that both $|\mathcal{K}|$ and $\bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H)$ are $\tilde{\mathcal{O}}(\epsilon_g^{-3/2})$ while $J(L_g, \epsilon_H, \zeta)$ and N_{meo} are both $\min\{n, \tilde{\mathcal{O}}(\epsilon_g^{-1/4})\}$. ∎

Note that for $n \gg \epsilon_g^{-1/4}$, the bound in part (ii) of this theorem is $\tilde{\mathcal{O}}(\epsilon_g^{-7/4})$, which is a familiar quantity in the literature on the operation complexity required to find an $(\epsilon_g, \epsilon_g^{1/2})$-stationary point [1, 6, 30].

Theorem 4.7 illustrates the benefits of using a capped truncated CG routine in terms of attaining good computational complexity guarantees. As a final remark, we expect the "cap" of Algorithm 3.1 to be triggered only in rare cases, due to the conservative nature of the CG convergence bounds that gave rise to this cap.

5. Practical considerations. Having presented an analysis of the theoretical complexity of our inexact trust-region Newton-CG approach, we consider several practical issues that arise in developing a computational implementation of this method.

The randomness inherent in Algorithm 3.2 is central to the complexity analysis of Algorithm 4.1 that was presented in section 4. Since the randomness carries with it a small probability of failure of Algorithm 3.2, two unsavory situations can occur that lead to "failure modes" for Algorithm 4.1. First, suppose that Algorithm 3.2 is called in line 11 because $\|g_k\| \leq \epsilon_g$ and `outCG` = INT-RES. In this case, if `capCG` = TRUE and Algorithm 3.2 predicts that $\lambda_{\min}(H_k) \geq -\epsilon_H$, then with probability up to ξ this indication is incorrect, and a direction of sufficiently negative curvature

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

actually exists but was not found. Second, suppose that Algorithm 3.2 is called because outCG = INT-MAX. Here, it is again possible with probability up to ξ that the indication $\lambda_{\min}(H_k) > -\epsilon_H$ will be made, even though we know from Lemma 3.2 that $\lambda_{\min}(H_k) \leq -\epsilon_H$. This second case can occur even when $\|g_k\| > \epsilon_g$, meaning that termination can occur at a point that is not even ϵ_g-stationary. Note, however, that in a given iteration the probability of these two situations is bounded by ξ, which appears only logarithmically in the constant $\mathcal{C}_{\text{meo}}$ of Assumption 2, and thus can be chosen to be extremely small. We can avoid this second case by replacing Algorithm 3.1 with the alternative truncated CG method of [29, Algorithm 1], which computes and returns a negative curvature direction whenever it detects that one exists. This "implicitly capped" method is more complicated to describe than the "explicitly capped" version of CG that we consider here, so in this paper we opted for simplicity of description at the expense of a (small) probability of failure in the subsequent call to Algorithm 3.2.

Our experiments with the inexact algorithms reported in section 6 use the randomized Lanczos procedure of [21, 22]. In practice, Algorithm 3.2 is rarely invoked by Algorithm 4.1—in the vast majority of test problems, it is invoked only once on the last iteration of Algorithm 4.1 as a final check that the Hessian is approximately positive semidefinite. Rather than explicitly capping the number of iterations to the value described in Assumption 2 and the comments that follow it, we use an adaptive criterion to decide when a close approximation to the minimum eigenvalue has been detected. Specifically, we stop at iteration l if $\lambda_{l-t} - \lambda_l \leq 10^{-5}$, where $t = \min\{l, n, 10\}$ and λ_l is the estimate of the minimum eigenvalue at the lth iteration of randomized Lanczos. It makes sense to use such a tight tolerance, since Algorithm 3.2 is rarely invoked by Algorithm 4.1, so that the cost of doing a careful check for approximate semidefiniteness is worthwhile.

In our implementations of the truncated CG procedure (Algorithm 3.1), we use the quantity $\bar{n} := \min\{n+2, 1.2n\}$ in place of n in determining the maximum number of iterations $k_{\max}$. This relaxation can be beneficial in practice since loss of conjugacy due to numerical rounding can result in a zero residual *not* being attained by CG after n steps. Typically, a small number of additional iterations beyond n suffices to obtain a more accurate solution. Another key parameter in this algorithm, the value ζ used as a convergence threshold for the residual, is set to .25 in our tests. (We discuss the settings of the parameters in Algorithm 4.1 in the next section.)

6. Computational experiments. We implemented several variants of trust-region Newton methods in MATLAB, as follows.

- *TR-Newton*. An implementation of Algorithm 2.1 with the trust-region subproblem solved using a Moré–Sorensen approach [24].
- *TR-Newton (no reg.)*. The same as *TR-Newton*, except that the regularization term involving ϵ_H is omitted from the subproblem objective (2.2). This variant demonstrates the effect of this regularization term on the practical performance of *TR-Newton*.
- *TR-Newton-CG-explicit*. An implementation of Algorithm 4.1 with an explicit cap on the number of CG iterations (that is, capCG = TRUE).
- *TR-Newton-CG-explicit (no reg.)*. The same as *TR-Newton-CG-explicit*, except that the regularization term involving ϵ_H is omitted from the subproblem objective, again to illustrate the impact of this regularization.
- *TR-Newton-CG*. An implementation of Algorithm 4.1 without an explicit cap (that is, capCG = FALSE).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

- *TR-Newton-CG (no reg.).* The same as *TR-Newton-CG*, except that the regularization term involving ϵ_H is omitted from the subproblem objective. This method is the most similar to traditional trust-region Newton-CG with Steihaug–Toint stopping rules for the CG routine. The only differences with the latter approach is that, for consistency with the other methods, we use the stopping test on line 21 of Algorithm 3.1 and still use the MEO (Algorithm 3.2) to ensure convergence to an (ϵ_g, ϵ_H)-stationary point (with high probability).
- *TRACE.* The trust-region algorithm with guaranteed optimal complexity proposed and analyzed in [11].

For the six instances of our algorithms, we used a regularization of $2\epsilon_H$ while computing the Newton-type step. We set $\gamma_1 = \gamma_2^{-1} = 0.5$, $\psi = 0.75$, $\eta = 0.1$, $\zeta = 0.25$, $\delta_{\max} = 10^{20}$, and $\delta_0 = 10$.

Our experiments show that the empirical performance of these methods is similar in terms of the number of iterations, function evaluations, and gradient evaluations required to locate an (ϵ_g, ϵ_H)-stationary point. In particular, the performance of the Newton-CG variants in terms of iterations, function and gradient evaluations is almost unaffected by the presence of a cap. A second observation is that the regularization term in the trust-region subproblem objective in Algorithm 4.1, which is required to ensure optimal iteration and operation complexity properties for this method, has a noticeable effect on practical performance, mostly in terms of Hessian-vector products. We discuss this effect further below.

We tested the algorithms using problems from the CUTEst test collection [17]. Many problems in this benchmark come with different size options. If the default size (according to the sizes that come with the distribution, downloaded July 1, 2020) was in the range [100,1000], then we used the default size. Otherwise, we choose the size closest to the range [100,1000] from the default value. This resulted in a test set of 233 problems. In order to focus on the results of experiments for the larger problems in the set, the following discussion and presentation of results considers only the problems with $n \geq 100$; this is a test set of 109 problems.

Figure 6.1 shows performance profiles for various metrics [13]. The horizontal axis is capped at $\tau = 10$ in order to distinguish the performance of the methods more clearly. We considered two termination tolerances. In the first set of experiments, corresponding to the left column of plots in Figure 6.1, termination was declared when the algorithm encountered a $(10^{-5}, 10^{-5/2})$-stationary point. In the second set of experiments (the right column of plots in Figure 6.1) we terminate at $(10^{-5}, 10^{-5})$-stationary points. In both sets of runs, we imposed an iteration limit of 10^4. For the trust-region Newton-CG methods, we also imposed an overall Hessian-vector product limit of $10^4 n$: A run was declared to be unsuccessful if this limit is reached without a stationary point of the specified precision being found. Although not evident from the performance profiles due to the cap on τ, all algorithms solved at least 101 test problems out of 109 for both stationarity tolerances, a reliability of about 93%.

Figure 6.1 shows the performance of all algorithms to be similar in terms of required iterations and gradient evaluations. We do, however, see significant differences in the number of Hessian-vector products required for the four TR-Newton-CG methods. The variants with no regularization term in the subproblems outperform the others in this respect; recall that these variants do not possess optimal complexity guarantees. The practical significance of this difference in performance depends on the cost of computing a gradient relative to the cost of a Hessian-vector product. If gradient evaluations are significantly more expensive, our results suggest no

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

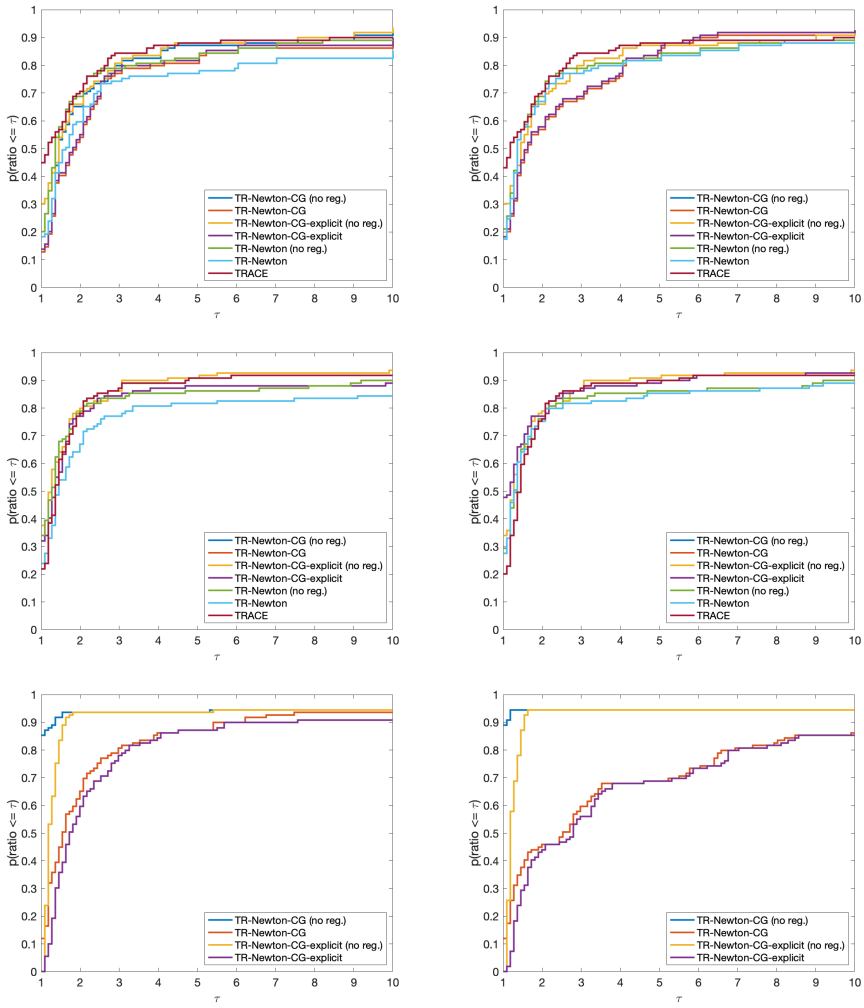

FIG. 6.1. *Performance profiles for iterations (top), gradient evaluations (middle), and Hessian-vector products (bottom). A termination tolerance of* $(\epsilon_g, \epsilon_H) = (10^{-5}, 10^{-5/2})$ *is used for the left column, and a termination tolerance of* $(\epsilon_g, \epsilon_H) = (10^{-5}, 10^{-5})$ *is used for the right column.*

substantial difference in computation time between the four Newton-CG methods. On the other hand, if Hessian-vector products are expensive relative to gradients, there may be a significant increase in run time as a result of including the regularization term. We remark that the vast majority of the Hessian-vector products in the Newton-CG variants were computed in Algorithm 3.1: on all problems but three, Algorithm 3.2 was only called at the last iteration to assess termination.

7. Conclusion. We have established that, with a few critical modifications, the popular trust-region Newton-CG method can be equipped with second-order complexity guarantees that match the best known bounds for second-order methods for solving smooth nonconvex optimization problems. We derived iteration complexity results for both exact and inexact variants of the approach, and for the inexact variant we leveraged iterative linear algebra techniques to obtain strong operation

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

complexity guarantees (in terms of gradient computations and Hessian-vector products) that again match the best known methods in the literature. Finally, we showed that the practical effects of including these modifications can be relatively minor.

Our results could be modified to obtain alternative complexity results for approximate ϵ_g-stationary points. For instance, we could modify Algorithm 3.1 by monitoring the decrease rate of the residual norm in a way that the number of CG iterations is subject to an *implicit cap* [29], in place of the explicit cap used here (when `capCG` = TRUE). With appropriate modifications in Algorithm 4.1, and under the assumptions of Theorem 4.7, one could establish a deterministic operation complexity bound of $\tilde{\mathcal{O}}(\epsilon_g^{-7/4})$ for reaching an ϵ_g-stationary point. However, the resulting method is significantly more delicate to implement and must be paired with Algorithm 3.2 to be endowed with a second-order complexity analysis.

Acknowledgments. We thank Yue Xie for providing most of the proof of Theorem 4.6. We are grateful to the editor and two anonymous referees, whose comments led to significant improvements to the paper.

REFERENCES

[1] N. Agarwal, Z. Allen-Zhu, B. Bullins, E. Hazan, and T. Ma, *Finding approximate local minima faster than gradient descent*, in Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017), 2017.

[2] E. Bergou, Y. Diouane, and S. Gratton, *On the use of the energy norm in trust-region and adaptive cubic regularization subproblems*, Comput. Optim. Appl., 68 (2017), pp. 533–554.

[3] E. Bergou, Y. Diouane, and S. Gratton, *A line-search algorithm inspired by the adaptive cubic regularization framework and complexity analysis*, J. Optim. Theory Appl., 178 (2018), pp. 885–913.

[4] Y. Carmon and J. C. Duchi, *Analysis of Krylov subspace solutions of regularized nonconvex quadratic problems*, in Advances in Neural Information Processing Systems 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., 2018, pp. 10726–10736.

[5] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, *"Convex until proven guilty": Dimension-free acceleration of gradient descent on non-convex functions*, in Proceedings of the International Conference on Machine Learning, August 2017, Sydney, Australia, 2017, pp. 654–663.

[6] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, *Accelerated methods for non-convex optimization*, SIAM J. Optim., 28 (2018), pp. 1751–1772.

[7] C. Cartis, N. I. M. Gould, and P. L. Toint, *Adaptive cubic regularisation methods for unconstrained optimization. Part* II*: worst-case function- and derivative-evaluation complexity*, Math. Program., 130 (2011), pp. 295–319.

[8] C. Cartis, N. I. M. Gould, and P. L. Toint, *Optimal Newton-Type Methods for Nonconvex Optimization*, Tech. Report naXys-17-2011, Department of Mathematics, University of Namur, Namur, Belgium, 2011.

[9] C. Cartis, N. I. M. Gould, and P. L. Toint, *Complexity bounds for second-order optimality in unconstrained optimization*, J. Complexity, 28 (2012), pp. 93–108.

[10] C. Cartis, N. I. M. Gould, and P. L. Toint, *Worst-case evaluation complexity and optimality of second-order methods for nonconvex smooth optimization*, in Proceedings of the International Congress of Mathematicians (ICM 2018), vol. 3, 2019, pp. 3697–3738.

[11] F. E. Curtis, D. P. Robinson, and M. Samadi, *A trust region algorithm with a worst-case iteration complexity of* $O\left(\epsilon^{-3/2}\right)$ *for nonconvex optimization*, Math. Program., 162 (2017), pp. 1–32.

[12] F. E. Curtis, D. P. Robinson, and M. Samadi, *An inexact regularized Newton framework with a worst-case iteration complexity of* $\mathcal{O}(\epsilon^{-3/2})$ *for nonconvex optimization*, IMA J. Numer. Anal., 39 (2019), pp. 1296–1327.

[13] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.

[14] J.-P. Dussault, *Arc$_q$: A new adaptive regularization by cubics*, Optim. Methods Softw., 33 (2018), pp. 322–335.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[15] J.-P. Dussault and D. Orban, *Scalable Adaptive Cubic Regularization Methods*, Tech. Report G-2015-109, GERAD, 2015.
[16] N. I. M. Gould, S. Lucidi, M. Roma, and P. L. Toint, *Solving the trust-region subproblem using the Lanczos method*, SIAM J. Optim., 9 (1999), pp. 504–525.
[17] N. I. M. Gould, D. Orban, and P. L. Toint, *CUTEst: A constrained and unconstrained testing environment with safe threads*, Comput. Optim. Appl., 60 (2015), pp. 545–557.
[18] N. I. M. Gould and V. Simoncini, *Error Estimates for Iterative Algorithms for Minimizing Regularized Quadratic Subproblems*, Tech. Report RAL-TR-2019-004, Rutherford Appleton Laboratory, 2019.
[19] S. Gratton, A. Sartenaer, and P. L. Toint, *Recursive trust-region methods for multiscale nonlinear optimization*, SIAM J. Optim., 19 (2008), pp. 414–444.
[20] E. Hazan and T. Koren, *A linear-time algorithm for trust region problems*, Math. Program., 158 (2016), pp. 363–381.
[21] J. Kuczyński and H. Woźniakowski, *Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1094–1122.
[22] J. Kuczyński and H. Woźniakowski, *Probabilistic bounds on the extremal eigenvalues and condition number by the Lanczos algorithm*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 672–691.
[23] J. M. Martínez and M. Raydan, *Cubic-regularization counterpart of a variable-norm trust-region method for unconstrained minimization*, J. Global Optim., 68 (2017), pp. 367–385.
[24] J. J. Moré and D. C. Sorensen, *Computing a trust region step*, SIAM J. Sci. Comput., 4 (1983), pp. 553–572.
[25] S. G. Nash, *Newton-type minimization via the Lanczos method*, SIAM J. Numer. Anal., 21 (1984), pp. 770–788.
[26] Y. Nesterov, *Introductory Lectures on Convex Programming. Volume* I*: Basic Course*, manuscript, 1998, p. 5.
[27] Y. Nesterov and B. T. Polyak, *Cubic regularization of Newton method and its global performance*, Math. Program., 108 (2006), pp. 177–205.
[28] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer Ser. Oper. Res. Financ. Eng., Springer-Verlag, New York, 2006.
[29] C. W. Royer, M. O'Neill, and S. J. Wright, *A Newton-CG algorithm with complexity guarantees for smooth unconstrained optimization*, Math. Program., 180 (2020), pp. 451–488. https://doi.org/10.1007/s10107-019-01362-7.
[30] C. W. Royer and S. J. Wright, *Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization*, SIAM J. Optim., 28 (2018), pp. 1448–1477.
[31] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.
[32] P. L. Toint, *Towards an efficient sparsity exploiting Newton method for minimization*, in Sparse Matrices and Their Uses, I. S. Duff, ed., Academic Press, London, 1981, pp. 57–88.
[33] J. Wang and Y. Xia, *A linear-time algorithm for the trust region subproblem based on hidden convexity*, Optim. Lett., 11 (2017), pp. 1639–1646.
[34] L.-H. Zhang, C. Shen, and R.-C. Li, *On the generalized Lanczos trust-region method*, SIAM J. Optim., 27 (2017), pp. 2110–2142.

Downloaded 04/12/21 to 76.116.8.166. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.