Industrial and
Systems Engineering

Trust-Region Newton-CG with Strong Second-Order Complexity Guarantees for Nonconvex Optimization

Frank E. Curtis[1], Daniel P. Robinson[1], Clément W. Royer[2], and Stephen J. Wright[3]

[1]Department of Industrial and Systems Engineering, Lehigh University

[2]LAMSADE, CNRS, Université Paris-Dauphine, Université PSL

[3]Computer Sciences Department, University of Wisconsin

ISE Technical Report 19T-028

LEHIGH UNIVERSITY

Trust-Region Newton-CG with Strong Second-Order Complexity Guarantees for Nonconvex Optimization[*]

Frank E. Curtis[†] Daniel P. Robinson[†] Clément W. Royer[‡] Stephen J. Wright[§]

December 9, 2019

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" (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 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

1 Introduction

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \; f(x), \tag{1}$$

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). 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*

[*]Version of December 9, 2019.

[†]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). Work of the first author was supported by DOE Award DE-SC0010615 and NSF Awards CCF-1740796 and CCF-1618717.

[‡]LAMSADE, CNRS, Université Paris-Dauphine, Université PSL, 75016 PARIS, FRANCE. (clement.royer@dauphine.psl.eu). Work of this author was partially supported by Award N660011824020 from the DARPA Lagrange Program.

[§]Computer Sciences Department, University of Wisconsin, 1210 W. Dayton St., Madison, WI 53706, USA. (swright@cs.wisc.edu). Work of this author was supported by NSF Awards IIS-1447449, 1628384, 1634597, and 1740707; AFOSR Award FA9550-13-1-0138; Subcontracts 3F-30222 and 8F-30039 from Argonne National Laboratory; and Award N660011824020 from the DARPA Lagrange Program.

Newton-CG methods. Our complexity analysis for both methods is based on approximate satisfaction of second-order necessary conditions for stationarity, that is,

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

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

$$\|\nabla f(x^\epsilon)\| \le \epsilon_g \;\text{ and }\; \lambda_{\min}(\nabla^2 f(x^\epsilon)) \ge -\epsilon_H, \tag{3}$$

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 affecting practical performance. This is in contrast to other recently proposed schemes that achieve good complexity properties, but have not demonstrated such good performance in practice [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 (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 (3). As in earlier works on Newton-CG methods [28, 27], 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 regulariztion 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 (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 conjugate gradient 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 some computational experiments in Section 5, and make some concluding observations in Section 6.

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), 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 $k \in \mathbb{N}$ of an algorithm for solving (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). 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 $|\phi(\cdot)| \leq C\varphi(\cdot)$ for some $C \in (0, \infty)$. Similarly, we write $\phi(\cdot) = \tilde{\mathcal{O}}(\varphi(\cdot))$ to indicate that $|\phi(\cdot)| \leq C\varphi(\cdot)|\log^c(\cdot)|$ 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) 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., [24]):

$$f(x_k + s_k) - f_k - g_k^T s_k - \tfrac{1}{2} s_k^T H_k s_k \leq \tfrac{L_H}{6}\|s_k\|^3 \quad \text{for all } k \in \mathbb{N}, \tag{4a}$$
$$\|g(x_k + s_k) - g_k - H_k s_k\| \leq \tfrac{L_H}{2}\|s_k\|^2 \quad \text{for all } k \in \mathbb{N}, \tag{4b}$$
$$\text{and} \quad \|H_k\| \leq L_g \quad \text{for all } k \in \mathbb{N}. \tag{4c}$$

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 [25, 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 conjugate gradient method for inexactly solving linear systems, and randomized Lanczos for calculating negative curvature directions are described in [28, 27]. 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 [27] finds an ϵ_g-stationary point deterministically in $\tilde{\mathcal{O}}(\epsilon_g^{-7/4})$ operations, showing that the conjugate gradient 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 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 [25, 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 [21]. 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 [22]; see also [26, Chapter 4]. For large-scale problems, iterative techniques based on the conjugate gradient (CG) algorithm [29, 30] and the Lanczos method [16, 23] have been described in the literature. Convergence rates for the method of [16] are presented in [32], 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 an SDP relaxation, and [31], 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). When it occurs, either the conjugate gradient procedure (Algorithm 2) returns an acceptable trial step, or else the minimum-eigenvalue procedure (Algorithm 3) 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 [28], is written as Algorithm 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

$$m_k(x) := f_k + g_k^T(x - x_k) + \tfrac{1}{2}(x - x_k)^T H_k (x - x_k); \tag{5}$$

see (6). 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. However, we claim that this effect is not substantial in practice, a claim for which our numerical experiments in Section 5 provide evidence. In any case, the remainder of Algorithm 1 is identical to a traditional trust-region Newton method.

Before presenting our analysis of Algorithm 1, we remark that $\{\lambda_k\}$ 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 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

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

Algorithm 1 Trust-Region Newton Method (exact version)

Require: Tolerances $\epsilon_g \in (0, \infty)$ and $\epsilon_H \in (0, \infty)$; parameters $\gamma_1 \in (0, 1)$ and $\gamma_2 \in [1, \infty)$; 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** $\|g_k\| \le \epsilon_g$ **then**
5: Compute $\lambda_k \leftarrow \lambda_{\min}(H_k)$.
6: **if** $\lambda_k \ge -\epsilon_H$ **then**
7: **return** x_k as an (ϵ_g, ϵ_H)-stationary point for problem (1).
8: **end if**
9: **end if**
10: Compute a trial step s_k as a solution to the regularized trust-region subproblem

$$\min_{s \in \mathbb{R}^n} \; m_k(x_k + s) + \tfrac{1}{2}\epsilon_H \|s\|^2 \;\; \text{s.t.} \;\; \|s\| \le \delta_k. \tag{6}$$

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

$$\rho_k \leftarrow \frac{f_k - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}. \tag{7}$$

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

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

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

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 1 that

$$\mathcal{K} = \{k \in \mathbb{N} : \text{iteration } k \text{ is reached and either } \|g_k\| > \epsilon_g \text{ or } \lambda_k < -\epsilon_H\}. \tag{8}$$

It follows that, for a run of Algorithm 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 (6), it is well known [22, 26] that there exists a scalar Lagrange multiplier μ_k such that

$$g_k + (H_k + \epsilon_H I + \mu_k I)s_k = 0, \tag{9a}$$
$$H_k + \epsilon_H I + \mu_k I \succeq 0, \tag{9b}$$
$$\text{and} \;\; 0 \le \mu_k \perp (\delta_k - \|s_k\|) \ge 0. \tag{9c}$$

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

Lemma 1. *For all $k \in \mathcal{K}$, the model reduction satisfies*

$$m_k(x_k) - m_k(x_k + s_k) \geq \tfrac{1}{2}\epsilon_H \|s_k\|^2. \tag{10}$$

Proof. The definition of m_k in (5) and the optimality conditions in (9) give

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

as desired. □

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

Lemma 2. *For all $k \in \mathcal{K}$, if $\delta_k \leq 3(1-\eta)\epsilon_H / L_H$, then $k \in \mathcal{S}$. Hence, by the trust-region radius update procedure, it follows that*

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

Proof. To prove the first statement of the lemma, we prove the equivalent contrapositive statement. To that end, suppose that $k \in \mathcal{U}$ (meaning that $\rho_k < \eta$), which from the definition of ρ_k means that

$$\eta\,(m_k(x_k + s_k) - m_k(x_k)) < f(x_k + s_k) - f_k. \tag{11}$$

Combining (11) with (4a), (6), Lemma 1, and (5) leads to

$$\begin{aligned} & \eta\,(m_k(x_k + s_k) - m_k(x_k)) < g_k^T s_k + \tfrac{1}{2} s_k^T H_k s_k + \tfrac{L_H}{6}\|s_k\|^3 \\ \Longrightarrow\quad & (\eta - 1)\,(m_k(x_k + s_k) - m_k(x_k)) < \tfrac{L_H}{6}\|s_k\|^3 \\ \Longrightarrow\quad & (1 - \eta)\,(m_k(x_k) - m_k(x_k + s_k)) < \tfrac{L_H}{6}\|s_k\|^2 \delta_k \\ \Longrightarrow\quad & \tfrac{1-\eta}{2}\epsilon_H \|s_k\|^2 < \tfrac{L_H}{6}\|s_k\|^2 \delta_k \\ \Longrightarrow\quad & \tfrac{3(1-\eta)}{L_H}\epsilon_H < \delta_k. \end{aligned}$$

We have shown that $k \in \mathcal{U}$ implies $\delta_k > 3(1-\eta)\epsilon_H/L_H$, meaning $\delta_k \leq 3(1-\eta)\epsilon_H/L_H$ implies $k \in \mathcal{S}$, as desired. 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 3. *The following hold for all successful iterations:*

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

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

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

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

Proof. Part (i) follows from Lemma 1 and the definition of $\mathcal{B}$, which imply that

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

For part (ii), from $k \in \mathcal{I}$ we know that $\|s_k\| < \delta_k$. This fact along with (9c) and (9a) imply that $\mu_k = 0$ and $g_k + (H_k + \epsilon_H I)s_k = 0$. Now, with (4b), we have

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

which after rearrangement yields

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

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

$$\|s_k\| \ge \tfrac{-\epsilon_H + \sqrt{\epsilon_H^2 + 2L_H\|g_{k+1}\|}}{L_H} = \left(\tfrac{-1+\sqrt{1+2L_H\|g_{k+1}\|\epsilon_H^{-2}}}{L_H}\right)\epsilon_H.$$

This inequality may be combined with [28, Lemma 17 in Appendix A]—using the values $a = 1$, $b = 2L_H$, and $t = \|g_{k+1}\|\epsilon_H^{-2}$ for that lemma—to obtain

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

Using this inequality in conjunction with $k \in \mathcal{S}$ and Lemma 1 proves that

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

which completes the proof for part (ii). □

We now bound the number of successful iterations before termination.

Lemma 4. *The number of successful iterations performed by Algorithm 1 before an (ϵ_g, ϵ_H)-stationary point is reached satisfies*

$$|\mathcal{S}| \le \left\lfloor \mathcal{C}_{\mathcal{S}} \max\{\epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\}\right\rfloor + 1, \tag{12}$$

where

$$\mathcal{C}_{\mathcal{S}} := \tfrac{4(f_0 - f_{\text{low}})}{\eta}\max\left\{\tfrac{1}{\delta_0^2}, \tfrac{L_H^2}{9\gamma_1^2(1-\eta)^2}, 1+2L_H\right\}. \tag{13}$$

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

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

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

$$c := \tfrac{\eta}{2}\min\left\{\delta_0^2, \tfrac{9\gamma_1^2(1-\eta)^2}{L_H^2}, \tfrac{1}{1+2L_H}\right\}. \tag{14}$$

For $k \in \mathcal{S}_L$, the fact that the algorithm has not yet terminated implies (see (8)) that $\lambda_k < -\epsilon_H$. By (9), 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 3(i), Lemma 2, and (14) imply that

$$f_k - f_{k+1} \ge \tfrac{\eta}{2}\epsilon_H\delta_k^2 \ge \tfrac{\eta}{2}\min\left\{\delta_0^2\epsilon_H, \tfrac{9\gamma_1^2(1-\eta)^2}{L_H^2}\epsilon_H^3\right\} \ge c\min\left\{\epsilon_H, \epsilon_H^3\right\}. \tag{15}$$

Now consider $k \in \mathcal{S}_{GG}$. Since in this case either of the cases in Lemma 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, and the definition of c in (14), it follows that

$$f_k - f_{k+1} \geq c \min \left\{ \epsilon_H, \epsilon_g^2 \epsilon_H^{-1}, \epsilon_H^3 \right\}. \tag{16}$$

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 (15) and (16) shows that

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

from which it follows that

$$|\mathcal{S}_L| + |\mathcal{S}_{GG}| \leq \left(\frac{f_0 - f_{\text{low}}}{c} \right) \max\{\epsilon_H^{-1}, \epsilon_g^{-2} \epsilon_H, \epsilon_H^{-3}\}. \tag{17}$$

Next, let us consider the set $\mathcal{S}_{GL}$. Since $k \in \mathcal{S}_{GL}$ means $\|g_{k+1}\| \leq \epsilon_H$, 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 (17) yields

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

which completes the proof when we substitute for c from (14). □

We now bound the number of unsuccessful iterations. The approach used here is based on the technique of [9], wherein a similar result is obtained.

Lemma 5. *The number of unsuccessful iterations performed by Algorithm 1 before an (ϵ_g, ϵ_H)-stationary point is reached satisfies*

$$|\mathcal{U}| \leq \left\lfloor 1 + \log_{\gamma_1} \left(\frac{3(1-\eta)}{L_H \delta_{\max}} \right) + \log_{\gamma_1} (\epsilon_H) \right\rfloor |\mathcal{S}|. \tag{18}$$

Proof. Let us denote the successful iteration indices as $\{k_1, \dots, k_{|\mathcal{S}|}\} := \mathcal{S}$. For any $i \in \{1, \dots, |\mathcal{S}| - 1\}$, our goal is to bound $k_{i+1} - k_i - 1$. From the update formulas for the trust-region radius, one finds for all $l \in \{k_i + 1, \dots, k_{i+1} - 1\}$ that

$$\delta_l \leq \min(\gamma_2 \delta_{k_i}, \delta_{\max}) \gamma_1^{l - k_i - 1} \leq \delta_{\max} \gamma_1^{l - k_i - 1}. \tag{19}$$

Since $k_{i+1} \in \mathcal{S}$, it follows that $x_{k_{i+1}}$ is not an (ϵ_g, ϵ_H)-stationary point, or else the algorithm would have terminated in iteration $k_i + 1$. Hence, a step is computed corresponding to all iteration indices $\{k_i + 1, \dots, k_{i+1}\}$. Moreover, for any index $l \in \{k_i + 1, \dots, k_{i+1} - 1\}$ the iteration is unsuccessful, which according to Lemma 2 means that, for such l, one has $\delta_l > 3(1-\eta)\epsilon_H / L_H$. Thus, for such l, (19) implies

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

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

$$k_{i+1} - k_i - 1 \le 1 + \log_{\gamma_1}\left(\tfrac{3(1-\eta)}{L_H \delta_{\max}}\right) + \log_{\gamma_1}(\epsilon_H). \tag{20}$$

For the first successful iteration, $\delta_{k_1} = \gamma_1^{k_1}\delta_0 \le \gamma_1^{k_1}\delta_{\max}$, so by Lemma 2 one finds

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

which implies that

$$k_1 \le 1 + \log_{\gamma_1}\left(\tfrac{3(1-\eta)}{L_H \delta_{\max}}\right) + \log_{\gamma_1}(\epsilon_H). \tag{21}$$

Since the iteration immediately prior to termination is $k_{|\mathcal{S}|}$ (except in the trivial case in which termination occurs at iteration 0), we have

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

The result follows by substituting (20) and (21) into this expression. □

The main result for iteration complexity of Algorithm 1 may now be proved.

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

$$|\mathcal{S}| = \mathcal{O}\left(\max\{\epsilon_H^{-3}, \epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H\}\right) \tag{22}$$

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

$$|\mathcal{K}| = \mathcal{O}\left(\log_{1/\gamma_1}(\epsilon_H^{-1})\max\{\epsilon_H^{-3}, \epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H\}\right). \tag{23}$$

Proof. Formula (22) follows from Lemma 4. Formula (23) follows from Lemma 4, Lemma 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 (22) and (23) as well as any positive scalar $\bar{\epsilon}_g \in \mathbb{R}$, then Theorem 1 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 [28, 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 2 as an appropriate iterative method for approximately solving the trust-region subproblem

$$\min_{s \in \mathbb{R}^n} \; g^T s + \tfrac{1}{2}s^T(H + 2\epsilon I)s \;\; \text{s.t.} \;\; \|s\| \le \delta, \tag{24}$$

where $g \in \mathbb{R}^n$ is assumed to be a non-zero 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 2 is based on the CG method and builds on the Steihaug-Toint approach [29, 30]. (The factor of 2 in the regularization term in (24) 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 2 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` $\leftarrow$ BND-NORM to indicate that s lies on the boundary of the trust-region constraint.

However, there are three key differences between Algorithm 2 and truncated CG. First, the residual termination criterion in line 21 enforces the condition

$$\|(H+2\epsilon I)s+g\| \;\le\; \tfrac{\zeta}{2}\min\{\|g\|,\epsilon\|s\|\}, \tag{25}$$

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, 28] (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 (25) by setting `outCG` $\leftarrow$ INT-RES.

Second, traditional truncated CG terminates if a direction of nonpositive curvature is encountered. Line 10 of Algorithm 2 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 $-\epsilon$. 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 2 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 6. *Suppose $\epsilon I \prec H+2\epsilon I \preceq (M+2\epsilon)I$ for $M \in [\|H\|,\infty)$ and define*

$$\kappa(M,\epsilon) := (M+2\epsilon)/\epsilon \quad \text{and} \quad J(M,\epsilon,\zeta) := \tfrac{1}{2}\sqrt{\kappa(M,\epsilon)}\,\ln\left(4\kappa(M,\epsilon)^{3/2}/\zeta\right), \tag{26}$$

where $\zeta \in (0,1)$ is input to Algorithm 2. If lines 3–7 were simply to set $k_{\max} \leftarrow \infty$, then Algorithm 2 would terminate at either line 18 or line 22 after a number of iterations (equivalently, matrix-vector products) equal to at most

$$\min\{n, J(M,\epsilon,\zeta)\} \;=\; \min\left\{n, \tilde{\mathcal{O}}(\epsilon^{-1/2})\right\}. \tag{27}$$

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 [28, proof of Lemma 11] that CG would reach an iterate satisfying (25)—so that termination in line 22 would occur—in at most the number of iterations given by (27). Of course, if termination occurs earlier in line 18, the bound (27) still holds. □

When employing the trust-region method of Section 4 for minimizing f, Algorithm 2 is invoked without knowing whether or not $H+2\epsilon I \succ \epsilon I$. Nevertheless, Lemma 6 allows us to make the following crucial observation.

Lemma 7. *If the iteration limit in Algorithm 2 is exceeded (that is, termination occurs at line 29), then $H \not\succeq -\epsilon I$.*

Algorithm 2 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** $p_j^T(H+2\epsilon I)p_j \le \epsilon\|p_j\|^2$ **then**
11:         Compute $\sigma \ge 0$ 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** $\|y_{j+1}\| \ge \delta$ **then**
17:         Compute $\sigma \ge 0$ 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** $\|r_{j+1}\| \le \frac{\zeta}{2}\min\{\|g\|, \epsilon\|y_{j+1}\|\}$ **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 $y_{j+1} \leftarrow y_j$.
27:     Set $j \leftarrow j+1$.
28: **end while**
29: **return** $s \leftarrow y_{k_{\max}}$ and `outCG` $\leftarrow$ INT-MAX.

Proof. If $k_{\max}$ is set to n in line 4 or line 6, then it would follow from standard CG theory that Algorithm 2 cannot reach line 29, 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 6, the choice of M, and the choice of $k_{\max}$ in line 4 that $H + 2\epsilon I \not\succeq \epsilon$. □

We now establish that any step computed by Algorithm 2 possesses favorable properties with respect to the *non-regularized* version of the quadratic model.

Lemma 8. *If Algorithm 2 returns* `outCG` $\in \{\text{BND-NEG}, \text{BND-NORM}, \text{INT-RES}\}$ *(that is,* `outCG` $\ne$ INT-MAX*), then the step s returned by the algorithm satisfies*

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

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

$$g^T y_i + \tfrac{1}{2}y_i^T(H+2\epsilon I)y_i \le 0 \quad \text{for all} \quad i \in \{0,1,\dots,j\}. \tag{28}$$

Suppose $\texttt{outCG} \in \{\text{BND-NORM}, \text{INT-RES}\}$. From (28) and the fact that $g^T s + \frac{1}{2}s^T(H+2\epsilon I)s < g^T y_j + \frac{1}{2}y_j^T(H+2\epsilon I)y_j$ when $\texttt{outCG} \leftarrow \text{BND-NORM}$, we have

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

which implies the desired result.

Second, suppose that $\texttt{outCG} \leftarrow \text{BND-NEG}$, meaning that Algorithm 2 terminates because iteration j yields $p_j^T(H+2\epsilon I)p_j \le \epsilon\|p_j\|^2$. 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

$$\tfrac{1}{2}s^T(H+2\epsilon I)s = \tfrac{1}{2}\delta^2(p_0^T(H+2\epsilon I)p_0)/\|p_0\|^2 \le \tfrac{1}{2}\epsilon\delta^2 = \tfrac{1}{2}\epsilon\|s\|^2,$$

from which it follows that

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

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

$$y_i = \sum_{\ell=0}^{i-1} \alpha_\ell p_\ell \in \operatorname{span}\{p_0,\dots,p_{i-1}\} \quad \text{for all } i \in \{1,2,\dots,j\}, \tag{29a}$$

$$p_i^T(H+2\epsilon I)p_\ell = 0 \quad \text{for all } \{i,\ell\} \subseteq \{0,1,\dots,j\} \text{ with } i \ne \ell, \tag{29b}$$

$$r_i^T p_j = -\|r_i\|^2 \quad \text{for all } i \in \{0,1,\dots,j\}, \tag{29c}$$

$$\text{and} \quad y_i^T p_i \ge 0 \quad \text{for all } i \in \{0,1,\dots,j\}. \tag{29d}$$

(The relationships (29a)–(29c) are standard in the literature; see, for example, [26, Chapter 5]. For (29d), see [29, eq. (2.13)].) Together, (29) and $s = y_j + \sigma p_j$ imply

$$g^T p_j = r_0^T p_j \le 0 \tag{30a}$$

$$s^T(H+2\epsilon I)s = y_j^T(H+2\epsilon I)y_j + \sigma^2 p_j^T(H+2\epsilon I)p_j \tag{30b}$$

$$\text{and} \quad \|s\|^2 = \|y_j\|^2 + 2\sigma y_j^T p_j + \sigma^2\|p_j\|^2 \ge \sigma^2\|p_j\|^2. \tag{30c}$$

Combining (28), (30), $\sigma \ge 0$, and $p_j^T(H+2\epsilon I)p_j \le \epsilon\|p_j\|^2$ shows that

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

which completes the proof. □

Lemma 8 shows that if $\epsilon = \epsilon_H$ and $\texttt{outCG} \ne \text{INT-MAX}$, 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 1 (see Lemma 1). However, we note that this decrease is obtained by using a larger regularization term.

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 $\texttt{outCG} = \text{INT-MAX}$ is returned by Algorithm 2. In this case, it must hold that the maximum allowed number of iterations satisfies $k_{\max} < n$ and, as a consequence of Lemma 7, that $H \not\succeq -\epsilon I$. 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 2 terminates with $\texttt{outCG} = \text{INT-RES}$. 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 $H \not\succeq -\epsilon I$.

These two cases motivate the need for a *minimum eigenvalue oracle* that estimates the minimum eigenvalue of H, or else returns a prediction that (with some desired probability) no sufficiently negative eigenvalue exists. The oracle that we employ is given by Algorithm 3.

Algorithm 3 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

$$g^T s \le 0, \quad s^T H s \le -\tfrac{1}{2}\epsilon\|s\|^2, \quad \text{and} \quad \|s\| = \delta, \tag{31}$$

where v has been computed to satisfy $\|v\| = 1$ and $v^T H v \le -\epsilon/2$, or (ii) a prediction that $H \succeq -\epsilon I$ holds. In the event that (ii) occurs, the probability that the prediction is wrong (that is, that $\lambda_{\min}(H) < -\epsilon$) must be no greater than ξ. (The bound M may be needed for algorithm termination; see Assumption 2 on page 13.)

---

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 can be viewed as an inexact version of Algorithm 1. We aim at remaining close to the traditional Newton-CG approaches in [29, 30] by having Algorithm 4 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 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 $\|g_k\| \le \epsilon_g$.)

4.2 Complexity

As in [27], we make the following assumption on the MEO in order to obtain complexity results for Algorithm 4.

Assumption 2. *When Algorithm 3 is called by Algorithm 4, the number of Hessian-vector products required is no more than*

$$N_{\text{meo}} = N_{\text{meo}}(\epsilon_H) := \min\left\{n, 1 + \left\lceil \mathcal{C}_{\text{meo}}\epsilon_H^{-1/2} \right\rceil\right\} \tag{32}$$

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

The following instances of Algorithm 3 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 $\mathcal{C}_{\text{meo}} = \ln(2.75n/\xi^2)\sqrt{M}/2$; see [27, Lemma 2].

Algorithm 4 Trust-Region Newton-CG Method (inexact version)

Require: Tolerances $\epsilon_g \in (0,\infty)$ and $\epsilon_H \in (0,\infty)$; parameters $\gamma_1 \in (0,1)$ and $\gamma_2 \in [1,\infty)$; 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 $M \in [L_g,\infty)$.

1: **for** $k = 0,1,2,\ldots$ **do**
2: Evaluate g_k and H_k.
3: **if** $g_k \neq 0$ **then**
4: Call Algorithm 2 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`.
5: **else**
6: Set $s_k^{\mathrm{CG}} \leftarrow 0$ and `outCG` $\leftarrow$ INT-RES.
7: **end if**
8: **if** `outCG` $\in \{$BND-NEG, BND-NORM$\}$ or ($\|g_k\| > \epsilon_g$ and `outCG` = INT-RES) **then**
9: Set $s_k \leftarrow s_k^{\mathrm{CG}}$.
10: **else** {that is, `outCG` = INT-MAX or ($\|g_k\| \le \epsilon_g$ and `outCG` = INT-RES)}
11: Call Algorithm 3 with inputs $g = g_k$, $H = H_k$, $\epsilon = \epsilon_H$, $\delta = \delta_k$, ξ, and M, obtaining either s_k satisfying (31) or a prediction that $H_k \succeq -\epsilon_H I$.
12: **if** Algorithm 3 predicts that $H_k \succeq -\epsilon_H I$ **then**
13: **return** x_k.
14: **end if**
15: **end if**
16: Compute the ratio of actual to predicted decrease in f defined as

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

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

- The *conjugate gradient 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 [27, Theorem 1].

Since for each instance the conditions of Assumption 2 hold with $\mathcal{C}_{\mathrm{meo}}$ equal to the given value, it follows that throughout a run of Algorithm 4, the conditions hold with $\mathcal{C}_{\mathrm{meo}} = \ln(2.75n/\xi^2)\sqrt{L_g}/2$. Algorithm 3 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.

To establish complexity results for Algorithm 4 we must take the randomness inherent in Algorithm 3 into account. Two unsavory situations can occur. First, suppose that Algorithm 3 is called in line 11 because $\|g_k\| \le \epsilon_g$ and `outCG` = INT-RES. In this case, if `capCG` = TRUE and Algorithm 3 predicts that $\lambda_{\min}(H_k) \ge -\epsilon_H$, then with probability up to ξ this prediction is incorrect and a direction of sufficiently negative curvature actually exists but was not found. Second, suppose that Algorithm 3 is called because `outCG` = INT-MAX. Here, it is again possible with probability up to ξ that the prediction $\lambda_{\min}(H_k) > -\epsilon_H$ will be made, even though we know from Lemma 7 that $\lambda_{\min}(H_k) \le -\epsilon_H$. This second case can occur 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 $\mathcal{C}_{\rm meo}$ of Assumption 2, and thus can be chosen to be very small.

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 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. 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*. 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 9. *Consider any realization of a run of Algorithm 4. For all $k \in \mathcal{K}$,*

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

Proof. If $s_k = s_k^{\rm CG}$, where $s_k^{\rm CG}$ is computed from Algorithm 2, then, in view of line 8, this means that `outCG` $\neq$ INT-MAX. Hence, it follows as in Lemma 8 that the desired bound holds. Now suppose that s_k is computed from Algorithm 3 in line 11. Since $k \in \mathcal{K}$, Algorithm 4 does not terminate in iteration k, and it follows from (31) that

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

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 10. *Consider any realization of a run of Algorithm 4. For all $k \in \mathcal{K}$, if $\delta_k \leq 3(1-\eta)\epsilon_H/(2L_H)$, then $k \in \mathcal{S}$. Hence, by the trust-region radius update procedure, it follows that for any realization of a run of Algorithm 4 that*

$$\delta_k \geq \bar{\delta}_{\min} := \min\left\{\delta_0, \left(\tfrac{3\gamma_1(1-\eta)}{2L_H}\right)\epsilon_H\right\} \in (0,\infty) \quad \text{for all} \quad k \in \mathcal{K}. \tag{33}$$

Proof. For any realization of a run of the algorithm, we can follow the proof of Lemma 2, using Lemma 9 in lieu of Lemma 1. Hence, the lower bound in (33) 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 11. *Consider any realization of a run of Algorithm 4. The following hold for all successful iterations:*

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

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

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

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

Proof. For part (i), we combine $k \in \mathcal{B} \cap \mathcal{S}$ with Lemma 9 to obtain, as desired,

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

Now consider part (ii). Note that since $k \in \mathcal{I}$, s_k cannot have been computed from a call to Algorithm 3 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 $\texttt{outCG} = \text{INT-RES}$, as desired. In turn, the fact that $\texttt{outCG} = \text{INT-RES}$ implies that (25) holds with $H = H_k$, $g = g_k$, $s = s_k$, and $\epsilon = \epsilon_H$ so that

$$r_k := (H_k + 2\epsilon_H)s_k + g_k \quad \text{has} \quad \|r_k\| \leq \tfrac{\zeta}{2}\epsilon_H\|s_k\|. \tag{34}$$

Combining this bound with (4b) and $\zeta \in (0,1)$, we have

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

which can be rearranged to yield

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

Reasoning as in the proof of Lemma 3, with $\frac{5}{2}\epsilon_H$ replacing ϵ_H, we obtain

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

By applying [28, Lemma 17 in Appendix A]—using $(a,b,t) = (5, 8L_H, \|g_{k+1}\|\epsilon_H^{-2})$ for the values in that lemma—it follows that

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

which may be combined with $k \in \mathcal{I} \cap \mathcal{S}$ and Lemma 9 to obtain

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

which completes the proof. □

The next result is analogous to Lemma 4 and takes randomness in the MEO into account.

Lemma 12. *For any realization of a run of Algorithm 4, the number of successful iterations performed before termination occurs satisfies*

$$|\mathcal{S}| \leq \bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H) := \left\lfloor \bar{\mathcal{C}}_{\mathcal{S}} \max\{\epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\}\right\rfloor + 1, \tag{35}$$

where

$$\bar{\mathcal{C}}_{\mathcal{S}} := \tfrac{8(f_0 - f_{\text{low}})}{\eta}\max\left\{\tfrac{1}{\delta_0^2}, \tfrac{4L_H^2}{9\gamma_1^2(1-\eta)^2}, 7 + 2L_H)\right\}. \tag{36}$$

Proof. For a given realization of a run of the algorithm, we can follow the reasoning of the proof for Lemma 4. In what follows, $\mathcal{S}_L$, $\mathcal{S}_{GG}$, and $\mathcal{S}_{GL}$ are defined as in the proof of Lemma 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

$$c_1 := \tfrac{\eta}{4}\min\left\{\delta_0^2, \tfrac{9\gamma_1^2(1-\eta)^2}{4L_H^2}, \tfrac{1}{7+2L_H}\right\}. \tag{37}$$

We can use Lemma 11 to conclude that $k \in \mathcal{K} \cap \mathcal{B}$, so that $\|s_k\| = \delta_k$. By combining Lemma 11(i), Lemma 10, and (37), we have for $k \in \mathcal{S}_L$ that

$$f_k - f_{k+1} \geq \frac{\eta}{4}\epsilon_H \delta_k^2 \geq \frac{\eta}{4} \min\left\{\delta_0^2 \epsilon_H, \frac{9\gamma_1^2(1-\eta)^2}{4L_H^2}\epsilon_H^3\right\} \geq c_1 \min\left\{\epsilon_H, \epsilon_H^2\right\}. \tag{38}$$

For $k \in \mathcal{S}_{GG}$, we have from Lemma 11 (either (i) or (ii)) and Lemma 10 that

$$\begin{aligned} f_k - f_{k+1} &\geq \frac{\eta}{4} \min\left\{\delta_0^2 \epsilon_H, \frac{9\gamma_1^2(1-\eta)^2}{4L_H^2}\epsilon_H^3, \frac{1}{7+2L_H}\epsilon_g^2\epsilon_H^{-1}, \frac{1}{7+2L_H}\epsilon_H^3\right\} \\ &\geq c_1 \min\left\{\epsilon_H, \epsilon_H^3, \epsilon_g^2\epsilon_H^{-1}\right\}. \end{aligned} \tag{39}$$

By following the reasoning that led to (17), we obtain from (38) and (39) that

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

As in the proof of Lemma 4, we have that $|\mathcal{S}_{GL}| \leq |\mathcal{S}_L| + 1$, so that

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

The desired bound follows by substituting the definition (37) into this bound. To complete the proof, we note that the right-hand side of (35) 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 13. *For any realization of a run of Algorithm 4, the number of unsuccessful iterations performed before termination occurs satisfies*

$$|\mathcal{U}| \leq \left\lfloor 1 + \log_{\gamma_1}\left(\frac{3(1-\eta)}{2L_H\delta_{\max}}\right) + \log_{\gamma_1}(\epsilon_H)\right\rfloor (|\mathcal{S}| + 1). \tag{40}$$

Proof. For a given realization of a run of the algorithm, the bound follows from the argument in the proof of Lemma 5 with two changes. First, Lemma 10 is used in place of Lemma 2. Second, we do not know that the iteration immediately prior to termination must be a successful iteration for Algorithm 4, as was the case for Algorithm 1. However, using the argument in the proof of Lemma 5 along with Lemma 10 shows that a sequence of no more than

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

consecutive unsuccessful iterations can occur after the final successful iteration, for otherwise an additional successful iteration would be obtained. Taking this fact into account leads to the extra 1 on the right-hand side of (40) as compared to (18). □

Before stating our complete iteration complexity result, we make the following assumption about how Algorithm 4 is implemented.

Assumption 3. *For any realization of a run of Algorithm 4, suppose $k \in \mathcal{K}$ is an index of an iteration such that (i) Algorithm 3 is called in line 11 and returns a negative curvature direction s_k for H_k and (ii) the step s_k is 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 is replaced by an access to v_k, scaled appropriately to compute s_k with norm δ_k.*

Assumption 3 implies that Algorithm 4 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 calls Algorithm 3 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 2. *Under Assumption 1, for any realization of a run, the number of successful iterations (and objective gradient evaluations) performed by Algorithm 4 before termination occurs satisfies (with $\bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H)$ defined in (35))*

$$|\mathcal{S}| \le \bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H) = \mathcal{O}\left(\max\{\epsilon_H^{-3}, \epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H\}\right) \tag{41}$$

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

$$\begin{aligned} |\mathcal{K}| &\le \left\lfloor 1 + \log_{\gamma_1}\left(\frac{3(1-\eta)}{2L_H\delta_{\max}}\right) + \log_{\gamma_1}(\epsilon_H)\right\rfloor (\bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H) + 1) \\ &= \mathcal{O}\left(\log_{1/\gamma_1}(\epsilon_H^{-1}) \max\{\epsilon_H^{-3}, \epsilon_H^{-1}, \epsilon_g^{-2}\epsilon_H\}\right). \end{aligned} \tag{42}$$

If `capCG` $=$ FALSE*, then $\|g_k\| \le \epsilon_g$ holds at termination. In any case, if Assumption 3 holds, then the vector x_k returned by Algorithm 4 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 (41) and (42) follow from Lemma 12 and Lemma 13. If `capCG` $=$ FALSE, then the flag output by Algorithm 2 has `outCG` $\neq$ INT-MAX. Combining this fact with line 8 of Algorithm 4 allows us to conclude that $\|g_k\| \le \epsilon_g$ when termination occurs in this case.

Finally, let us consider the case in which Assumption 3 holds. In this setting, the vector x_k returned by Algorithm 4 is not an (ϵ_g, ϵ_H)-stationary point only if the MEO (Algorithm 3) makes an inaccurate prediction, which, each time it is called, can occur with probability at most ξ. When a call to the MEO is made in iteration k, this necessarily means that all previous calls of the MEO between iterations 0 and $k-1$ have lead to accurate predictions. The MEO is called at most $|\mathcal{S}|$ times under Assumption 3, thus the total probability of *any of these calls* making an inaccurate prediction is at most $\sum_{i=0}^{|\mathcal{S}|-1}(1-\xi)^i\xi = 1-(1-\xi)^{|\mathcal{S}|}$. Since $|\mathcal{S}| \le \bar{K}_{\mathcal{S}}(\epsilon_g, \epsilon_H)$, the probability that all MEO calls are accurate—and hence the probability that x_k is an (ϵ_g, ϵ_H)-stationary point—is at least $(1-\xi)^{\bar{K}_{\mathcal{S}}(\epsilon_g,\epsilon_H)}$. ∎

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 3. *Let Assumption 1, Assumption 2, and Assumption 3 hold, and suppose that $\epsilon_H = \epsilon_g^{1/2}$ with $\epsilon_g \in (0,1)$. Then, for any realization of a run of Algorithm 4, the total number of Hessian-vector products performed satisfies:*

(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*

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

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 (32)) 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 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 2, 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 6 and (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, 28].

Theorem 3 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 2 to be triggered only in rare cases, due to the conservative nature of the CG convergence bounds that gave rise to this cap.

5 Computational experiments

We implemented several variants of trust-region Newton methods in Matlab, as follows.

- *TRACE*. The trust-region algorithm with guaranteed optimal complexity proposed and analyzed in [11].
- *TR-Newton*. An implementation of Algorithm 1 with the trust-region subproblem solved using a Moré-Sorensen approach [22].
- *TR-Newton (no reg.)*. The same as *TR-Newton*, except that the regularization term $\frac{1}{2}\epsilon_H\|s\|^2$ is removed from the subproblem objective in (6). 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 with an explicit cap on the number of CG iterations (that is, capCG = TRUE).
- *TR-Newton-CG-implicit*. An implementation of an algorithm similar to Algorithm 4, except that by monitoring the rate of decrease of the residual norm during CG, the number of CG iterations is subject to an *implicit cap*; see [27] for further details and discussion, and also our comments in Section 6.
- *TR-Newton-CG-implicit (no reg.)*. The same as *TR-Newton-CG-implicit*, 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 are that, for consistency with the other methods, we use an implicit cap on the number of CG iterations (as in *TR-Newton-CG-implicit*), and still use the MEO (Algorithm 3) to promote convergence to an (ϵ_g, ϵ_H)-stationary point.

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. A second observation is that the regularization term in the trust-region subproblem objective in Algorithm 4, which is required to ensure optimal iteration and operation complexity properties for this method, has a small but noticeable effect on practical performance. This effect could be viewed as the price of equipping the approach with optimal complexity guarantees.

We tested the algorithms using problems from the CUTEst test collection [17]. We report on unconstrained problems whose default sizes involved at least 100 variables, resulting in a test set of 41 problems. (In fact, we ran experiments using *all* unconstrained problems from the CUTEst collection and found the results to be qualitatively the same as those presented below.)

Figure 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 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 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 38 out of 41 test problems for both stationarity tolerances, a reliability of about 93%.

Figure 1 show 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 three TR-Newton-CG methods. The variant with no regularization term in the subproblems outperforms the others. This is the variant without 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 products. If gradient evaluations are significantly more expensive, our results suggest no substantial

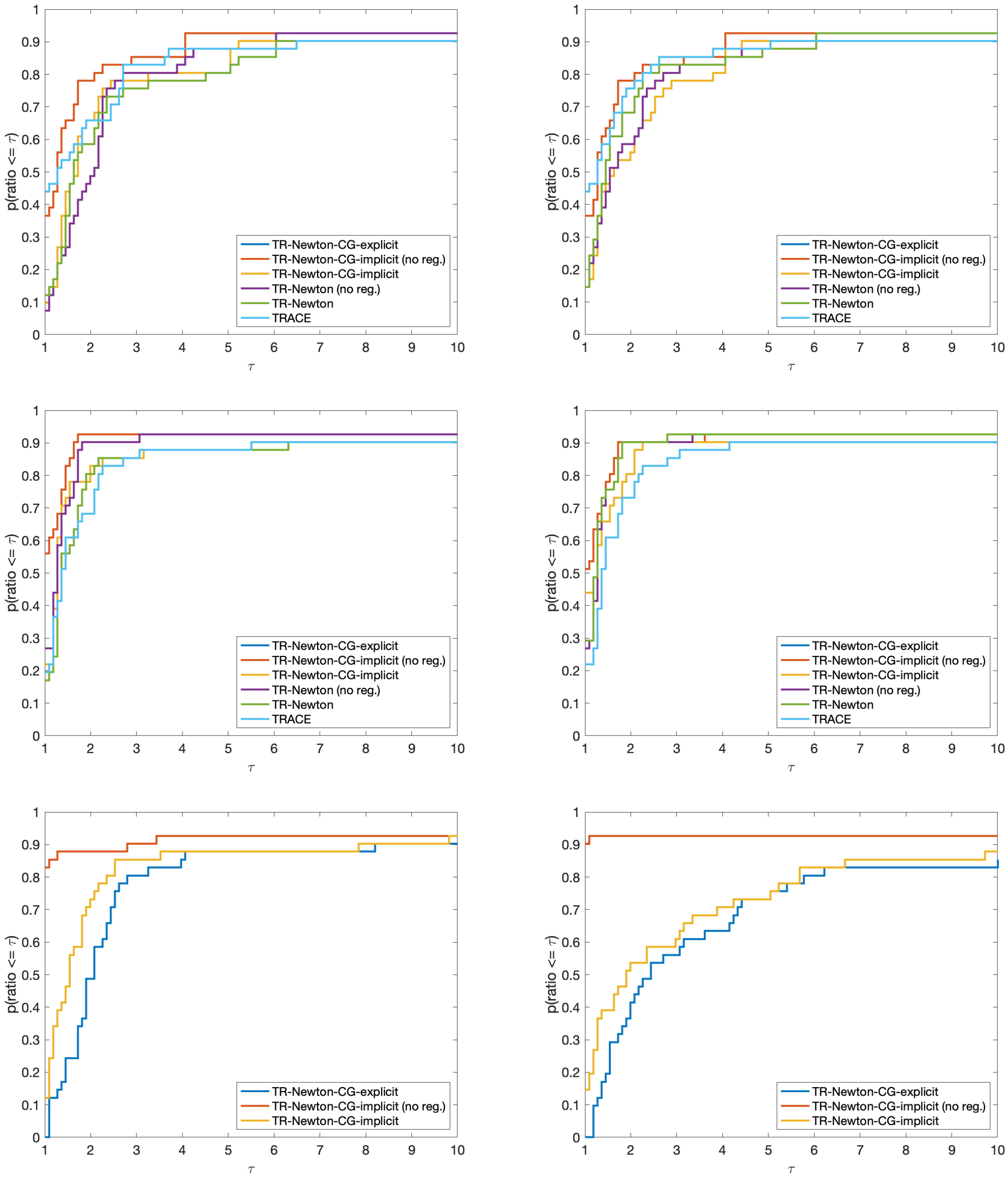

Figure 1: Performance profiles for iterations (top), gradient evaluations (middle), and Hessian-vector products (bottom). A termination tolerance of $(\epsilon_g, \epsilon_H) = (10^{-5}, 10^{-5/2})$ is used in the left column, with $(10^{-5}, 10^{-5})$ in the right column.

difference in computation time between the three 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.

6 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 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 is relatively minor.

Our results could be modified to obtain alternative complexity results for approximate ϵ_g-stationary points. For instance, we could modify Algorithm 2 by monitoring the decrease rate of the residual norm in a way that the number of CG iterations is subject to an *implicit cap* [27], in place of the explicit cap used here (when `capCG` = TRUE). With appropriate modifications in Algorithm 4, and under the assumptions of Theorem 3, one could establish a deterministic operation complexity bound of $\tilde{\mathcal{O}}(\epsilon_g^{-7/4})$ for reaching an ϵ_g-stationary point. As seen in our computational experiments, this version yielded comparable empirical performance with the explicitly capped method used in Algorithm 4.

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), PMLR, 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] ——, *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 Volume 70: International Conference on Machine Learning, 6-11 August 2017, International Convention Centre, Sydney, Australia, PMLR, 2017, pp. 654–663.

[6] ——, *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] ——, *Optimal Newton-type methods for nonconvex optimization*, Tech. Rep. naXys-17-2011, Dept of Mathematics, FUNDP, Namur (B), 2011.

[9] ——, *Complexity bounds for second-order optimality in unconstrained optimization*, J. Complexity, 28 (2012), pp. 93–108.

[10] ——, *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] ——, *An inexact regularized Newton framework with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization*, IMA J. Numer. Anal., (2018 (available online)).

[13] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 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.

[15] J.-P. Dussault and D. Orban, *Scalable adaptive cubic regularization methods*, Tech. Rep. 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. Rep. 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. 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.

[22] J. J. Moré and D. C. Sorensen, *Computing a trust region step*, SIAM J. Sci. Comput., 4 (1983), pp. 553–572.

[23] S. G. Nash, *Newton-type minimization via the Lanczos method*, SIAM J. Numer. Anal., 21 (1984), pp. 770–788.

[24] Y. Nesterov, *Introductory lectures on convex programming volume i: Basic course*, Lecture notes, 3 (1998), p. 5.

[25] Y. Nesterov and B. T. Polyak, *Cubic regularization of Newton method and its global performance*, Math. Program., 108 (2006), pp. 177–205.

[26] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer-Verlag, New York, second ed., 2006.

[27] C. W. Royer, M. O'Neill, and S. J. Wright, *A Newton-CG algorithm with complexity guarantees for smooth unconstrained optimization*, Math. Program., (2019 (available online)). https://doi.org/10.1007/s10107-019-01362-7.

[28] 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.

[29] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.

[30] 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 and New York, 1981, pp. 57–88.

[31] 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.

[32] L.-H. Zhang, C. Shen, and R.-C. Li, *On the generalized Lanczos trust-region method*, SIAM J. Optim., 27 (2017), pp. 2110–2142.