A Newton-CG Algorithm with Complexity Guarantees for Smooth Unconstrained Optimization

Clément W. Royer · Michael O’Neill · Stephen J. Wright

Abstract We consider minimization of a smooth nonconvex objective function using an iterative algorithm based on Newton’s method and linear conjugate gradient, with explicit detection and use of negative curvature directions for the Hessian of the objective function. The algorithm tracks Newton-conjugate gradient procedures developed in the 1980s closely, but includes enhancements that allow worst-case complexity results to be proved for convergence to points that satisfy approximate first-order and second-order optimality conditions. The complexity results match the best known results in the literature for second-order methods.

Keywords smooth nonconvex optimization · Newton’s method · conjugate gradient method · optimality conditions · worst-case complexity.

Mathematics Subject Classification (2000) 49M05, 49M15, 65F10, 65F15, 90C06, 90C60.

Received: date / Accepted: date

Research 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.

C. W. Royer
Wisconsin Institute of Discovery, University of Wisconsin, 330 N. Orchard St., Madison, WI 53715.
E-mail: croyer2@wisc.edu

M. O’Neill
Computer Sciences Department, University of Wisconsin, 1210 W. Dayton St., Madison, WI 53706.
E-mail: moneill@cs.wisc.edu

S. J. Wright
Computer Sciences Department, University of Wisconsin, 1210 W. Dayton St., Madison, WI 53706.
E-mail: swright@cs.wisc.edu.
1 Introduction

We consider the unconstrained optimization problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice Lipschitz continuously differentiable function that is nonconvex in general. We further assume that $f$ is bounded below for all $x$, by some constant $f_{\text{low}}$. Although the Hessian $\nabla^2 f(x)$ is well defined for such functions, we assume that $n$ is large enough that full evaluation of this matrix is undesirable from a computational viewpoint, though we assume that Hessian-vector products of the form $\nabla^2 f(x)v$ can be computed with reasonable efficiency, for arbitrary vectors $v$.

Unconstrained minimization of nonconvex smooth functions of many variables is a much-studied paradigm in optimization. Approaches such as limited-memory BFGS and nonlinear conjugate gradient are widely used to tackle (1), particularly in the case of large dimension $n$. Another popular approach, known as “Newton-CG,” applies the (linear) conjugate gradient (CG) method to the second-order Taylor-series approximation of $f$ around the current iterate $x_k$. Each iteration of CG requires computation of one Hessian-vector product of the form $\nabla^2 f(x_k)v$. A trust-region variant of Newton-CG, due to Steihaug [27], terminates the CG iterations when sufficient accuracy is achieved in minimizing the quadratic approximation, when a CG step leads outside the trust region, or when negative curvature is encountered in $\nabla^2 f(x_k)$. A line-search variant presented in [25] applies CG until some convergence criterion is satisfied, or until negative curvature is encountered, in which case the search direction reverts to negative gradient.

Theoretical guarantees for Newton-CG algorithms have been provided, e.g. in [12,15–17,25]. Convergence analysis for such methods typically shows that accumulation points are stationary, that is, they satisfy the first-order optimality condition $\nabla f(x) = 0$. Local linear or superlinear convergence to a point satisfying second-order sufficient conditions is sometimes also proved for Newton-CG methods. Bounds on global complexity of methods of this type have not been considered in the optimization literature. Recently, with the upsurge of interest in complexity, several new algorithms have been proposed that have good global complexity guarantees. We review some such contributions in Section 2. In most cases, these new methods depart significantly from those seen in the traditional optimization literature, and there are questions surrounding their practical appeal.

Our aim in this paper is to develop a method that hews closely to the Newton-CG approach, but which comes equipped with certain safeguards and enhancements that allow worst-case complexity results to be proved. At each iteration, we use CG to solve a slightly damped version of the Newton equations, monitoring the CG iterations for evidence of indefiniteness in the Hessian. If the CG process terminates with an approximation to the Newton step, we perform a backtracking line search along this direction. Otherwise, we step
along a negative curvature direction for the Hessian, obtained either from the CG procedure on the Newton equations, or via some auxiliary computation (possibly another CG process). In either case, we can show that significant decrease can be attained in \( f \) at each iteration, at reasonable computational cost (in terms of the number of gradient evaluations or Hessian-vector products), allowing worst-case complexity results to be proved.

The remainder of the paper is organized as follows. We position our work within the existing literature in Section 2. Our algorithm is described in Section 3. Section 4 contains the complexity analysis, showing both a deterministic upper bound on the computation required to attain approximate first-order conditions (Section 4.2) and a high-probability upper bound on the computation required to satisfy approximate second-order necessary conditions (Section 4.3). Section 5 contains some conclusions and discussion. Several technical results and proofs related to CG are gathered in the Appendix.

**Assumptions, Background, Notation.** Our algorithm seeks a point that approximately satisfies second-order necessary conditions for optimality, that is,

\[
\|\nabla f(x)\| \leq \epsilon_g, \quad \lambda_{\min}(\nabla^2 f(x)) \geq -\epsilon_H, \tag{2}
\]

for specified small positive tolerances \( \epsilon_g \) and \( \epsilon_H \). (Here and subsequently, \( \| \cdot \| \) denotes the Euclidean norm, or its induced norms on matrices.) We make the following standard assumptions throughout.

**Assumption 1** The level set \( L_f(x_0) = \{ x | f(x) \leq f(x_0) \} \) is compact.

**Assumption 2** The function \( f \) is twice Lipschitz continuously differentiable on an open neighborhood of \( L_f(x_0) \), and we denote by \( L_g \) and \( L_H \) the respective Lipschitz constants for \( \nabla f \) and \( \nabla^2 f \) on this set.

Under these two assumptions, there exist scalars \( f_{\text{low}}, U_g > 0, \) and \( U_H > 0 \) such that the following are satisfied for \( x \in L_f(x_0) \):

\[
f(x) \geq f_{\text{low}}, \quad \|\nabla f(x)\| \leq U_g, \quad \|\nabla^2 f(x)\| \leq U_H. \tag{3}
\]

A theoretically valid possibility is to set \( U_H = L_g \). However, since \( U_H \) is an explicit parameter in our method, we use separate notation for the upper bound on the Hessian norm.

For any \( x \) and \( d \) such that Assumption 2 is satisfied at \( x \) and \( x + d \), we have

\[
f(x + d) \leq f(x) + \nabla f(x)^T d + \frac{1}{2} d^T \nabla^2 f(x) d + \frac{L_H}{6} \|d\|^3. \tag{4}
\]

Notationally, we use order notation \( \mathcal{O} \) in the usual sense, whereas \( \tilde{\mathcal{O}}(\cdot) \) represents \( \mathcal{O} \) with logarithmic terms ignored. We use such notation in bounding iteration count and computational effort, and focus on the dependencies of such complexities on \( \epsilon_g \) and \( \epsilon_H \), not explicitly stating dependence on dimension \( n \) or the Lipschitz constants defined above.
2 Complexity in Nonconvex Optimization

In recent years, many algorithms have been proposed for finding a point that satisfies conditions (2), with iteration complexity and computational complexity bounded in terms of $\epsilon_g$ and $\epsilon_H$. We review several works most relevant to this paper here, and relate their results to our contributions. For purposes of computational complexity, we define the unit of computation to be one Hessian-vector product or one gradient evaluation, implicitly using the observation from computational / algorithmic differentiation [18] that these two operations differ in cost only by a modest factor, independent of the dimension $n$.

Classical second-order convergent trust-region schemes [12] can be shown to satisfy (2) after at most $O(\max\{\epsilon_g^{-2}\epsilon_H^{-1}, \epsilon_H^{-3}\})$ iterations [10]. For the class of second-order algorithms (that is, algorithms which rely on second-order derivatives and Newton-type steps) the best known iteration bound is $O(\max\{\epsilon_g^{-3/2}, \epsilon_H^{-3}\})$. This bound was first established for a form of cubic regularization of Newton’s method [24]. Following this paper, numerous other algorithms have also been proposed which match this bound, see for example [3,8,13,14,23].

A recent trend in complexity analysis of these methods also accounts for the computational cost of each iteration, thus yielding a bound on the computational complexity. Two independent proposals, respectively based on adapting accelerated gradient to the nonconvex setting [5] and approximately solving the cubic subproblem [1], require $\tilde{O}(\epsilon^{-7/4})$ operations (with high probability, but we show dependency only on $\epsilon$) to find a point $x$ that satisfies

$$\|\nabla f(x)\| \leq \epsilon \quad \text{and} \quad \lambda_{\min}(\nabla^2 f(x)) \geq -\sqrt{U_H}\epsilon. \quad (5)$$

The difference of a factor of $\epsilon^{-1/4}$ with the results presented above arises from the cost of computing a negative curvature direction of $\nabla^2 f(x_k)$ and/or the cost of solving a linear system. The probabilistic nature of the bound is generally due to the introduction of randomness in the curvature estimation process (see [2,28] for two recent illustrations). A complexity bound of the same type was also established for a variant of accelerated gradient free of negative curvature computation, that regularly adds a random perturbation to the iterate when the gradient norm is small [19].

In an interesting followup to [5], an algorithm based on accelerated gradient with a nonconvexity monitor was proposed [6]. It requires at most $O(\epsilon^{-7/4})$ iterations to satisfy (5) with high probability. However, if one is concerned only with satisfying the gradient condition $\|\nabla f(x)\| \leq \epsilon$, the $O(\epsilon^{-7/4})$ bound holds deterministically. Note that this bound represents an improvement over the $O(\epsilon^{-2})$ of steepest descent and classical Newton’s method [7]. The improvement is due to a modification of the accelerated gradient paradigm that allows for deterministic detection and exploitation of negative curvature directions in regions of sufficient nonconvexity.

In a previous work [26], two authors of the current paper proposed a Newton-based algorithm in a line-search framework which has an iteration
complexity of $\mathcal{O}(\max\{\epsilon_g^{-3/2}, \epsilon_H^{-3}\})$ when the subproblems are solved exactly, and a computational complexity of $\mathcal{O}(\epsilon^{-7/4})$ Hessian-vector products and/or gradient evaluations, when the subproblems are solved inexactly using CG and the randomized Lanczos algorithm. Compared to the accelerated gradient methods, this approach aligns more closely with traditional optimization practice, as described in Section 1.

Building on [26], the current paper describes a similar line-search framework with inexact Newton steps, but uses a modified version of CG to solve the system of Newton equations, without first checking for positive definiteness of the coefficient matrix. The modification is based on the idea of a convexity monitor from the accelerated gradient algorithms mentioned above, but adapted to CG. An implicit cap is applied on the number of CG iterations that are used to solve the damped Newton system. We show that once this cap is reached, either the damped Newton system has been solved to sufficient accuracy or else a direction of “sufficiently negative curvature” has been identified for the Hessian. (A single extra CG iteration may be needed to identify the negative curvature direction, in much the same manner as in [6] for accelerated gradient.) In contrast to the previous work [26], no estimate of the smallest eigenvalue of the Hessian is required prior to computing a Newton step. In addition to removing potentially unnecessary computation, this approach allows a deterministic result for first-order optimality to be proved, as in [6].

Our adaptation of the convexity monitoring strategy from accelerated gradient to CG is not surprising, since these two methods have similar theoretical properties when applied to strongly convex quadratics. Recent findings have shown that both methods can be analyzed via the same potential function [20] and viewed as instances of a underlying “idealized algorithm” [21].

We are deliberate in our use of CG rather than accelerated gradient as the method of choice for minimizing the quadratic objective that arises in the damped Newton step. When applied to strongly convex quadratics, both approaches have similar asymptotic linear convergence rates that depend only on the extreme eigenvalues of the Hessian, but CG has several advantages. It has a rich convergence theory that depends on the full spectrum of eigenvalues. It is fully adaptive, requiring no prior estimates of the extreme eigenvalues. Moreover, its practical performance on convex quadratics is superior. (See, for example, [25, Chapter 5] for a description of these properties.) Further, as we prove in this paper (Section 3.1), CG can be adapted to detect nonconvexity efficiently in a quadratic function, without altering its core properties. We show further (see Section 3.2 and Appendix B) that by applying CG to a linear system with a random right-hand side, we can find a direction of negative curvature in an indefinite matrix efficiently, with the same iteration complexity as the randomized Lanczos process of [22] used elsewhere. CG has potentially greater appeal in this context, however, as it dispenses with the need to store or regenerate iterates.
The practical benefits of CG in large-scale optimization have long been appreciated. We establish here that with suitable enhancements, methods based on CG can also be equipped with good complexity properties as well.

3 Damped-Newton / Capped-CG Method with Negative Curvature Steps

We detail our algorithm in this section, starting with its two basic elements. The first element is a conjugate gradient procedure that is used to solve the damped Newton system, which includes enhancements to detect indefiniteness in the Hessian and to guarantee that not too many iterations are performed. The second element is a “minimum eigenvalue oracle” which finds an approximation of the smallest eigenvalue of a symmetric matrix, together with a vector that corresponds to this approximation.

3.1 Capped Conjugate Gradient

Conjugate Gradient (CG) is a widely used technique for solving linear equations with symmetric positive definite coefficient matrices or, equivalently, minimizing strongly convex quadratic functions. We devise a modified CG algorithm and apply it to a system of the form $\bar{H}y = -g$, where $\bar{H} = H + 2\epsilon I$ is a damped version of the symmetric matrix $H$, which is our notational proxy for the Hessian $\nabla^2 f(x_k)$.

Algorithm 1 presents our Capped CG procedure. The main loop consists of classical CG iterations. When $\bar{H} \succeq \epsilon I$, Algorithm 1 will generate the same iterates as a classical conjugate gradient method, and terminate at an inexact solution of $\bar{H}y = -g$. When $\bar{H} \not\succeq \epsilon I$, the features added to Algorithm 1 cause a direction $d$ to be identified along which $d^T \bar{H}d < \epsilon \|d\|^2$ or, equivalently, $d^T Hd < -\epsilon \|d\|^2$ — a direction of “sufficiently negative curvature” for $H$. Directions $d$ of this type are encountered most obviously when they arise as iterates $y_j$ or search directions $p_j$ in the CG iterations. But evidence of the situation $\bar{H} \not\succeq \epsilon I$ can arise more subtly, when the residual norms $\|r_j\|$ decrease more slowly than we would expect if the eigenvalues of $\bar{H}$ were bounded below by $\epsilon$. Accordingly, Algorithm 1 checks residual norms for slow decrease, and if such behavior is detected, it uses a technique based on one used for accelerated gradient methods in [6] to recover a direction $d$ such that $d^T \bar{H}d < \epsilon \|d\|^2$.

The following lemma justifies our use of the term “capped” in connection with Algorithm 1. Regardless of whether the condition $\bar{H} \succeq \epsilon I$ is satisfied or not, the number of iterations will not exceed a certain number $J(M, \epsilon, \zeta)$ that we subsequently show to be $\tilde{O}(\epsilon^{-1/2})$. (We write $J$ for $J(M, \epsilon, \zeta)$ in some of the subsequent discussion, to avoid clutter.)

**Lemma 1** The number of iterations of Algorithm 1 is bounded by

$$\min\{n, J(M, \epsilon, \zeta)\},$$
Algorithm 1 Capped Conjugate Gradient

Inputs: Symmetric matrix $H \in \mathbb{R}^{n \times n}$; vector $g \neq 0$; damping parameter $\epsilon \in (0, 1)$; positive scalar $M \geq \|H\|$; desired relative accuracy $\zeta \in (0, \min\{1, M\})$;

Set

$$H := H + 2\epsilon, \quad \kappa := \frac{M + 2\epsilon}{\epsilon}, \quad \hat{\zeta} := \frac{\zeta}{3\kappa}, \quad T := \frac{4\kappa^4}{(1 - \sqrt{1 - T^2})^2}, \quad \tau := \frac{1}{\sqrt{\kappa} + 1}.$$ 

$y_0 \leftarrow 0, r_0 \leftarrow g, p_0 \leftarrow -g, j \leftarrow 0$;

if $p_0^T H p_0 < \epsilon \|p_0\|^2$ then

Set $d = p_0$ and terminate with d_type=NC;

end if

while TRUE do

$$\alpha_j \leftarrow r_j^T r_j / p_j^T H p_j; \quad \{\text{Begin Standard CG Operations}\}$$

$y_{j+1} \leftarrow y_j + \alpha_j p_j$;

$r_{j+1} \leftarrow r_j + \alpha_j H p_j$;

$\beta_{j+1} \leftarrow (r_{j+1}^T r_{j+1}) / (r_j^T r_j)$;

$p_{j+1} \leftarrow -r_{j+1} + \beta_{j+1} p_j; \quad \{\text{End Standard CG Operations}\}$

$j \leftarrow j + 1$;

if $y_j^T H y_j < \epsilon \|y_j\|^2$ then

Set $d \leftarrow y_j$ and terminate with d_type=NC;

else if $\|r_j\| \leq \hat{\zeta} \|r_0\|$ then

Set $d \leftarrow y_j$ and terminate with d_type=SOL;

else if $p_j^T H p_j < \epsilon \|p_j\|^2$ then

Set $d \leftarrow p_j$ and terminate with d_type=NC;

else if $\|r_j\| > \sqrt{T}(1 - \tau)^{1/2} \|r_0\|$ then

Compute $\alpha_j, y_{j+1}$ as in the main loop above;

Find $i \in \{0, \ldots, j - 1\}$ such that

$$\frac{(y_{j+1} - y_i)^T H (y_{j+1} - y_i)}{\|y_{j+1} - y_i\|^2} < \epsilon; \quad (6)$$

Set $d \leftarrow y_{j+1} - y_i$ and terminate with d_type=NC;

end if

end while

where $J = J(M, \epsilon, \zeta)$ is the smallest integer such that $\sqrt{T}(1 - \tau)^{J/2} \leq \hat{\zeta}$. The number of matrix-vector multiplications required is bounded by $2 \min\{n, J(M, \epsilon, \zeta)\} + 1$, unless all iterates $y_i, i = 1, 2, \ldots$ that could define the $d$ returned after (6) are stored, in which case the bound is $\min\{n, J(M, \epsilon, \zeta)\} + 1$.

Proof. If the full $n$ iterations are performed, without any of the termination conditions being tripped, the standard properties of CG (see Appendix A) ensure that the final residual $r_n$ is zero, so that the condition $\|r_n\| \leq \hat{\zeta} \|r_0\|$ is satisfied, and termination occurs.

Supposing that $J < n$, we note from the definition of $J$ that $\sqrt{T}(1 - \tau)^{J/2} \|r_0\| \leq \hat{\zeta} \|r_0\|$. Thus at iteration $J$, at least one of the following two termination conditions must be satisfied: $\|r_J\| \leq \hat{\zeta} \|r_0\|$ or $\|r_J\| > \sqrt{T}(1 - \tau)^{J/2} \|r_0\|$. Thus, termination must occur at (or before) iteration $J$.

For the final statement, note that the main workload at each iteration is computation of a single matrix-vector product $H p_j$, for $j = 0, 1, \ldots, \min\{n, J(M, \epsilon, \zeta)\}$. If we do not store additional information, we may need to regenerate the in-
formation needed to compute the direction satisfying (6) by re-running the iterations of CG, up to the second-last iteration. This fact accounts for the additional cost of \( \min \{ n, J(M, \epsilon, \zeta) \} \) in the no-storage case.

We can estimate \( J \) by taking logs in its definition, as follows:

\[
J \leq \frac{2 \ln(\hat{\zeta}/\sqrt{T})}{\ln(1 - \tau)} = \frac{\ln(\hat{\zeta}^2/T)}{\ln(1 + \frac{1}{\sqrt{\kappa}})} \leq \left( \sqrt{\kappa} + \frac{1}{2} \right) \ln \left( \frac{T}{\hat{\zeta}^2} \right),
\]

where we used \( \ln(1 + \frac{1}{\tau}) \geq \frac{1}{\tau + 1/\tau} \) to obtain the latest inequality. By replacing \( T, \tau, \hat{\zeta}, \kappa \) by their definitions in Algorithm 1, and using \( \frac{1}{1 - \sqrt{1 - \tau}} = \frac{1 + \sqrt{1 - \tau}}{\tau} \leq \frac{2}{\tau} \),
we obtain

\[
J(M, \epsilon, \zeta) \leq \min \left\{ n, \left[ \sqrt{\frac{M + 2\epsilon}{\epsilon}} + \frac{1}{2} \right] \ln \left( \frac{144 \left( \frac{M + 2\epsilon}{\epsilon} + 1 \right)^2 \left( \frac{M + 2\epsilon}{\epsilon} \right)^6}{\hat{\zeta}^2} \right) \right\} = \min \left\{ n, \tilde{O}(\epsilon^{-1/2}) \right\}. (7)
\]

3.2 Minimum Eigenvalue Oracle.

A minimum eigenvalue oracle is needed in the main algorithm to either return a direction of "sufficient negative curvature" in a given symmetric matrix, or else return a certificate that the matrix is almost positive definite. This oracle is stated as Procedure 2.

**Procedure 2 Minimum Eigenvalue Oracle**

*Inputs:* Symmetric matrix \( H \in \mathbb{R}^{n \times n} \), scalar \( M \) such that \( \lambda_{\text{max}}(H) \leq M \), and \( \epsilon > 0 \);
Set parameter \( \delta \in [0, 1) \);

*Outputs:* Estimate \( \lambda \) of \( \lambda_{\text{min}}(H) \) such that \( \lambda \leq -\epsilon/2 \), and vector \( v \) with \( \|v\| = 1 \) such that \( v^T Hv = \lambda \) OR certificate that \( \lambda_{\text{min}}(H) \geq -\epsilon \). (If the certificate is output, it is false with probability \( \delta \)).

To implement this oracle, we can use any procedure that finds the smallest eigenvalue of \( H \) to an absolute precision of \( \epsilon/2 \). Several possibilities for implementing Procedure 2 have been proposed in the literature, with various guarantees. An exact computation of the minimum eigenvalue and eigenvector (through a full Hessian evaluation and factorization) would be a valid choice for Procedure 2, but is unsuited to our setting in which Hessian-vector products and vector operations are the basic computations. For the same reason, we mention only in passing strategies that require only gradient evaluations \([2, 28]\), though they may offer similar guarantees to those discussed below.

We focus on two approaches for implementing Procedure 2. The first is the Lanczos method, which finds the smallest eigenvalue of the restriction of a given symmetric matrix to a Krylov subspace based on some initial vector.
When the starting vector is chosen randomly, the Krylov subspace increases in dimension by 1 at each Lanczos iteration, with high probability (see Appendix B and [22]). To the best of our knowledge, [5] was the first paper to use randomized Lanczos to detect negative curvature. The key result is as follows.

**Lemma 2** Consider using the Lanczos method to find the largest eigenvalue of $MI - H$ starting with a random vector uniformly generated on the unit sphere. This approach finds the largest eigenvalue of $MI - H$ (and thus the smallest eigenvalue of $H$) to an absolute precision of $\epsilon/2$, together with a corresponding direction $v$, in at most

$$\min \left\{ n, \frac{\ln(n/\delta^2)}{2} \sqrt{M/\epsilon} \right\} \text{ iterations},$$

with probability at least $1 - \delta$.

The proof of this result follows [26, Lemma 9] by setting $\epsilon = \epsilon/2$. Procedure 2 can be implemented by outputting the approximate eigenvalue $\lambda$ for $H$, determined by the randomized Lanczos process, along with the corresponding direction $v$, provided that $\lambda \leq -\epsilon/2$. When $\lambda > -\epsilon/2$, Procedure 2 returns the certificate that $\lambda_{\min}(H) \geq -\epsilon$.

The second approach is to apply the classical CG algorithm to solve a linear system in which the coefficient matrix is a shifted version of the matrix $H$ and the right-hand side is random. This procedure has essentially identical performance to Lanczos in terms of the number of iterations required to detect the required direction of sufficiently negative curvature, as the following theorem shows.

**Theorem 1** Suppose that Procedure 2 consists in applying the standard CG algorithm (see Appendix A) to the linear system

$$(H + \frac{1}{2}\epsilon I) d = b,$$

where $b$ is chosen randomly from a uniform distribution over the unit sphere. If $\lambda_{\min}(H) < -\epsilon$, then with probability at least $1 - \delta$, CG will yield a direction $v$ satisfying the conditions of Procedure 2 in a number of iterations bounded above by (8). Conversely, if CG runs for this number of iterations without encountering a direction of negative curvature for $H + \frac{1}{2}\epsilon I$, then $\lambda_{\min}(H) \geq -\epsilon$ with probability at least $1 - \delta$.

We prove this result, and give some additional details of the CG implementation, in Appendices A and B.

### 3.3 Damped Newton-CG

Algorithm 3 presents our method for finding a point that satisfies (2). It uses two kinds of search directions. Negative curvature directions (that are also first-order descent steps) are used when they are either encountered in the
course of applying the Capped CG method (Algorithm 1) to the damped Newton equations, or found explicitly by application of Procedure 2. The second type of step is an inexact damped Newton step, which is the other possible outcome of Algorithm 1. For both types of steps, a backtracking line search is used to identify a new iterate that satisfies a sufficient decrease condition, that depends on the cubic norm of the step. Such a criterion is instrumental in establishing optimal complexity guarantees in second-order methods [3, 13, 14, 26].

**Algorithm 3 Damped Newton-CG**

*Inputs:* Tolerances $\epsilon_g > 0$, $\epsilon_H > 0$; backtracking parameter $\theta \in (0, 1)$; starting point $x_0$; upper bound on Hessian norm $U_H > 0$; accuracy parameter $\zeta \in (0, \min\{1, U_H\})$;

for $k = 0, 1, 2, \ldots$ do

if $\|\nabla f(x_k)\| > \epsilon_g$ then

Call Algorithm 1 with $H = \nabla^2 f(x_k)$, $M = U_H$, $\epsilon = \epsilon_H$, $g = \nabla f(x_k)$, and accuracy parameter $\zeta$ to obtain outputs $d$ and $d_{\text{type}}$;

if $d_{\text{type}} = \text{NC}$ then

$d_k \leftarrow -\text{sgn}(d^T g) \frac{|d^T \nabla^2 f(x_k) d|}{\|d\|^3} \frac{d}{\|d\|}$;

else

$d_k \leftarrow d$;

end if

Go to Line Search;

else

Call Procedure 2 with $H = \nabla^2 f(x_k)$, $M = U_H$, and $\epsilon = \epsilon_H$;

if Procedure 2 certifies that $\lambda_{\text{min}}(\nabla^2 f(x_k)) \geq -\epsilon_H$ then

Terminate;

else

$d_k \leftarrow -\text{sgn}(v^T g) \frac{|v^T \nabla^2 f(x_k) v|}{\|v\|^3} v$ (where $v$ is the output from Procedure 2) and go to Line Search;

end if

end if

**Line Search:** Compute a step length $\alpha_k = \theta j_k$, where $j_k$ is the smallest nonnegative integer such that

$$f(x_k + \alpha_k d_k) < f(x_k) - \frac{\eta}{6} \alpha_k^3 \|d_k\|^3; \quad (9)$$

$x_{k+1} \leftarrow x_k + \alpha_k d_k$;

end for

In its deployment of two types of search directions, our method is similar to Steihaug’s trust-region Newton-CG method [27], which applies CG (starting from a zero initial guess) to solve the Newton equations but, if it encounters a negative curvature direction during CG, steps along that direction to the trust-region boundary. It differs from the line-search Newton-CG method described in [25, Section 7.1] in that it makes use of negative curvature directions when they are encountered, rather than discarding them in favor of a steepest-descent direction. Algorithm 3 improves over both approaches in having a global complexity theory for convergence to both approximate first-order points, and points satisfying the approximate second-order conditions (2).
Our method could also be equipped with a local phase, to obtain fast local convergence properties near points that satisfy a second-order sufficient condition (as in [26]). However, we omit this feature from our framework, since our focus here is on global convergence rates.

4 Convergence and Complexity Analysis

In this section, we present a global worst-case complexity analysis of Algorithm 3. Elements of the analysis follow those in the earlier paper [26]. The most technical part appears in Section 4.1 below, where we show that the Capped CG procedure returns (deterministically) either an inexact Newton step or a negative curvature direction, both of which can be used as the basis of a successful backtracking line search. These properties are used in Section 4.2 to prove complexity results for convergence to a point satisfying the approximate first-order condition $\|\nabla f(x)\| \leq \epsilon_g$. Section 4.3 proves complexity results for finding approximate second-order points (2), leveraging properties of the minimum eigenvalue oracle, Procedure 2.

4.1 Properties of Capped CG

We now explore the properties of the directions $d$ that are output by our Capped CG procedure. The main result deals with the case in which Algorithm 1 terminates due to insufficiently rapid decrease in $\|r_j\|$, showing that the strategy for identifying a direction of sufficient negative curvature for $H$ is effective.

**Theorem 2** Suppose that the loop of Algorithm 1 terminates with $j = \hat{J}$, where

$$\hat{J} \in \{1, \ldots, \min\{n, J(M, \epsilon, \zeta)\}\},$$

(where $J(M, \epsilon, \zeta)$ is defined in Lemma 1 and (7)) with

$$\|r_j\| > \max\{\zeta, \sqrt{T(1 - \tau)^{3/2}}\|r_0\|, \quad (10)$$

Suppose further that $y_j^T \bar{H} y_j \geq \epsilon \|y_j\|^2$, and $p_j^T \bar{H} p_j \geq \epsilon \|p_j\|^2$ so that $y_{j+1}$ is computed. Then we have

$$\frac{(y_{j+1} - y_i)^T \bar{H}(y_{j+1} - y_i)}{\|y_{j+1} - y_i\|^2} < \epsilon, \quad \text{for some } i \in \{0, \ldots, \hat{J} - 1\}. \quad (11)$$

The proof of Theorem 2 is quite technical, and can be found in Appendix C. It relies on an argument previously used to analyze a strategy based on accelerated gradient [6, Appendix A.1], itself inspired by a result of Bubeck [4], but it needs some additional steps that relate specifically to CG. The part of our proof that corresponds to [6, Appendix A.1] is simplified in some respects, thanks to the use of CG and the fact that it is being applied to a quadratic.

Having shown that Algorithm 1 is well-defined, we summarize the properties of its outputs.
Lemma 3 Let Assumptions 1 and 2 hold, and suppose that Algorithm 1 is invoked at an iterate $x_k$ of Algorithm 3 (so that $\|\nabla f(x_k)\| > \epsilon_g > 0$). Let $d_k$ be the vector obtained in Algorithm 3 from the output $d$ of Algorithm 1. Then, one of the two following statements holds:

1. $d_{\text{type}}=\text{SOL}$, and the direction $d_k$ satisfies

$$d_k^T(\nabla^2 f(x_k) + 2\epsilon_H I)d_k \geq \epsilon_H \|d_k\|^2,$$

$$\|d_k\| \leq \epsilon_H^{-1}\sqrt{1 + \frac{\zeta^2}{4}\|\nabla f(x_k)\|},$$

$$\|\hat{r}_k\| \leq \frac{\hat{\zeta}}{1 - \hat{\zeta}}(U_H + 2\epsilon_H)\|d_k\| \leq \frac{1}{2}\epsilon_H\zeta\|d_k\|,$$

where

$$\hat{r}_k := (\nabla^2 f(x_k) + 2\epsilon_H I)d_k + \nabla f(x_k);$$

2. $d_{\text{type}}=\text{NC}$, and the direction $d_k$ satisfies $d_k^T \nabla f(x_k) \leq 0$ as well as

$$\frac{d_k^T \nabla^2 f(x_k)d_k}{\|d_k\|^2} \leq -\epsilon_H,$$

$$\|d_k\| \geq \epsilon_H.$$

Proof. For simplicity in the proof, we let $H = \nabla^2 f(x_k)$ and $g = \nabla f(x_k)$. Suppose first that $d_{\text{type}}=\text{SOL}$. In that case, we have from the termination conditions in Algorithm 1 and (13) that

$$d_k^T(H + 2\epsilon_H I)d_k \geq \epsilon_H \|d_k\|^2,$$

$$\|\hat{r}_k\| \leq \hat{\zeta}\|g\|.$$

We immediately recognize (12a) in (15a). We now prove (12b). Observe first that (15a) yields

$$\epsilon_H \|d_k\|^2 \leq d_k^T(H + 2\epsilon_H I)d_k \leq \|d_k\|\|(H + 2\epsilon_H I)d_k\|,$$

so from (13) we have

$$\|d_k\| \leq \epsilon_H^{-1}\|(H + 2\epsilon_H I)d_k\| = \epsilon_H^{-1}\|g\| + \|\hat{r}_k\| = \epsilon_H^{-1}\sqrt{\|g\|^2 + \|\hat{r}_k\|^2} \leq \epsilon_H^{-1}\sqrt{1 + \frac{\zeta^2}{4}\|g\|},$$

where we used (15b) to obtain the final bound, together with the equality $\|g + \hat{r}_k\|^2 = \|g\|^2 + \|\hat{r}_k\|^2$, which follows from $g^T\hat{r}_k = -r_0^T\hat{r}_k = 0$, by orthogonality of the residuals in CG (see Lemma 7, Property 2). Since $\hat{\zeta} \leq \zeta/2$ by construction, (12b) holds.

The bound (12c) follows from (15b) and the logic below:

$$\|\hat{r}_k\| \leq \hat{\zeta}\|g\| \leq \hat{\zeta}(\|(H + 2\epsilon_H I)d_k\| + \|\hat{r}_k\|) \leq \hat{\zeta}(\|U_H + 2\epsilon_H\|\|d_k\| + \|\hat{r}_k\|)$$

$$\Rightarrow \|\hat{r}_k\| \leq \frac{\hat{\zeta}}{1 - \hat{\zeta}}(U_H + 2\epsilon_H)\|d_k\|. $$
We finally use
\[ \hat{\zeta} = \frac{\zeta}{3} \frac{\epsilon_H}{U_H + 2\epsilon_H} \leq \frac{\zeta}{3} \frac{1}{2} < \frac{1}{6}, \]
to arrive at
\[ \frac{\hat{\zeta}}{1 - \hat{\zeta}} (U_H + 2\epsilon_H) \leq \frac{6}{5} \hat{\zeta} (U_H + 2\epsilon_H) = \frac{6}{5} \frac{\zeta \epsilon_H}{3} < \frac{1}{2} \zeta \epsilon_H, \]
yielding (12c).

In the case of d\_type=NC, we recall that Algorithm 3 defines
\[ d_k = -\text{sgn}(d^\top g) \frac{|d^\top H d|}{\|d\|^2} \|d\| \] (16)
where \( d \) denotes the direction obtained by Algorithm 1. It follows immediately that \( d_k^\top g \leq 0 \). Since \( d_k \) and \( d \) are colinear, we also have that
\[ \frac{d_k^\top (H + 2\epsilon_H I) d_k}{\|d_k\|^2} = \frac{d^\top (H + 2\epsilon_H I) d}{\|d\|^2} \leq \epsilon_H \iff \frac{d_k^\top H d_k}{\|d_k\|^2} = \frac{d^\top H d}{\|d\|^2} \leq -\epsilon_H, \]
establishing (14a). By (16), we then have
\[ \|d_k\| = \frac{|d^\top H d|}{\|d\|^2} \geq \epsilon_H, \]
proving (14b).

### 4.2 First-Order Complexity Analysis

We now find a bound on the number of iterations and the amount of computation required to identify an iterate \( x_k \) for which \( \|\nabla f(x_k)\| \leq \epsilon^g \). We consider in turn the two types of steps (approximate damped Newton and negative curvature), finding a lower bound on the descent in \( f \) achieved on the current iteration in each case. We then prove an upper bound on the number of iterations required to satisfy these approximate first-order conditions (Theorem 3) and an upper bound on the number of gradient evaluations and Hessian-vector multiplications required (Theorem 4).

We start with a lemma concerning the approximate damped Newton steps.

**Lemma 4** Suppose that Assumptions 1 and 2 hold. Suppose that at iteration \( k \) of Algorithm 3, we have \( \|\nabla f(x_k)\| > \epsilon^g \), so that Algorithm 1 is called. When Algorithm 1 outputs a direction \( d_k \) with \( d\_type=SOL \), then the backtracking line search requires at most \( j_k \leq j_{sol} + 1 \) iterations, where
\[ j_{sol} = \left\lfloor \frac{1}{2} \log_\theta \left( \frac{3(1 - \hat{\zeta}) \epsilon_H^2}{L_H + \eta 1.5 U_g} \right) \right\rfloor^+, \] (17)
and the resulting step \( x_{k+1} = x_k + \alpha_k d_k \) satisfies
\[
f(x_k) - f(x_{k+1}) \geq c_{sol} \min \left( \| \nabla f(x_{k+1}) \|^3 \epsilon_H^{-3}, \epsilon_H^3 \right),
\]
where
\[
c_{sol} = \frac{\eta}{6} \min \left\{ \left[ \frac{4}{\sqrt{(4+\zeta)^2 + 8L_H + 4 + \zeta}} \right]^3, \left[ \frac{3\theta^2 (1 - \zeta)}{L_H + \eta} \right]^3 \right\}.
\]

Proof. The proof tracks exactly that of [26, Lemma 13]. The only significant difference is that equation (65) of [26], which is instrumental to the proof and requires a probabilistic assumption on \( \lambda_{\min}(\nabla^2 f(x_k)) \), is now ensured deterministically by (12b) from Lemma 3. As a result, both the proof and the result become deterministic. \( \square \)

In the case of \( \| \nabla f(x_{k+1}) \| \leq \epsilon_g \), the estimate (18) may not guarantee a “significant” decrease in \( f \) at this iteration. However, in this case, the approximate first-order condition \( \| \nabla f(x) \| \leq \epsilon_g \) holds at the next iteration, so that Algorithm 3 will invoke Procedure 2 at iteration \( k + 1 \), leading either to termination with satisfaction of the conditions (2) or to a step that reduces \( f \) by a multiple of \( \epsilon_H^3 \), as we show in Theorem 4 below.

We now address the case in which Algorithm 1 returns a negative curvature direction to Algorithm 3 at iteration \( k \). The backtracking line search guarantees that a sufficient decrease will be achieved at such an iteration. Although the Lipschitz constant \( L_H \) appears in our result, our algorithm does not require this constant to be known or estimated. This is in contrast to [6], in which an estimate of \( L_H \) is needed to obtain similar guarantees.

Lemma 5. Suppose that Assumptions 1 and 2 hold. Suppose that at iteration \( k \) of Algorithm 3, we have \( \| \nabla f(x_k) \| \geq \epsilon_g \), so that Algorithm 1 is called. When Algorithm 1 outputs \( d_{\text{type}} = \text{NC} \), the direction \( d_k \) (computed from \( d \) in Algorithm 3) has the following properties: The backtracking line search terminates with step length \( \alpha_k = \theta^{j_k} \) with \( j_k \leq j_{nc} + 1 \), where
\[
j_{nc} := \left\lceil \log \theta \left( \frac{3}{L_H + \eta} \right) \right\rceil,
\]
and the resulting step \( x_{k+1} = x_k + \alpha_k d_k \) satisfies
\[
f(x_k) - f(x_k + \alpha_k d_k) \geq c_{nc} \epsilon_H^3,
\]
with
\[
c_{nc} := \frac{\eta}{6} \min \left\{ 1, \frac{27\theta^3}{(L_H + \eta)^3} \right\}.
\]

Proof. By Lemma 3, we have by combining the properties in (14) that
\[
d_k^T \nabla^2 f(x_k) d_k \leq -\| d_k \|^3 \leq -\epsilon_H \| d_k \|^3 \leq \epsilon_H^3 \| d_k \|^3 \]
(21)
The result can thus obtained exactly as in [26, Lemma 1]. \( \square \)
Theorem 3 Let Assumptions 1 and 2 hold. Then, defining
\[ K_1 := \left\lceil \frac{f(x_0) - f_{\text{low}}}{\min\{c_{\text{sol}}, c_{\text{nc}}\}} \max\{\epsilon_g^{-3}, \epsilon_H^{-3}\} \right\rceil, \]
some iterate \( x_k, k = 0, 1, \ldots, K_1 + 1 \) generated by Algorithm 3 will satisfy
\[ \|\nabla f(x_k)\| \leq \epsilon_g. \] (22)

Proof. Suppose for contradiction that \( \|\nabla f(x_k)\| > \epsilon_g \) for all \( k = 0, 1, \ldots, K_1 + 1 \), so that
\[ \|\nabla f(x_{l+1})\| > \epsilon_g, \quad l = 0, 1, \ldots, K_1. \] (23)
Algorithm 1 will be invoked at each of the first \( K_1 + 1 \) iterates of Algorithm 3. For each iteration \( l = 0, 1, \ldots, K_1 \) for which Algorithm 1 returns \( \text{d\_type=SOL} \), we have from Lemma 4 and (23) that
\[ f(x_l) - f(x_{l+1}) \geq c_{\text{sol}} \min\{\|\nabla f(x_{l+1})\|\epsilon_H^{-3}, \epsilon_H^{-3}\} \geq c_{\text{sol}} \min\{\epsilon_g^{-3}, \epsilon_H^{-3}\}. \] (24)
For each iteration \( l = 0, 1, \ldots, K_1 \) for which Algorithm 1 returns \( \text{d\_type=NC} \), we have by Lemma 5 that
\[ f(x_l) - f(x_{l+1}) \geq c_{\text{nc}} \epsilon_H^3. \] (25)
By combining these results, we obtain
\[ f(x_0) - f(x_{K_1+1}) \geq \sum_{l=0}^{K_1} (f(x_l) - f(x_{l+1})) \geq \sum_{l=0}^{K_1} \min\{c_{\text{sol}}, c_{\text{nc}}\} \min\{\epsilon_g^{-3}, \epsilon_H^{-3}\} = (K_1 + 1) \min\{c_{\text{sol}}, c_{\text{nc}}\} \min\{\epsilon_g^{-3}, \epsilon_H^{-3}\} > f(x_0) - f_{\text{low}}. \]
where we used the definition of \( K_1 \) for the final inequality. This inequality contradicts the definition of \( f_{\text{low}} \) in (3), so our claim is proved. \( \square \)

If we choose \( \epsilon_H \) in the range \([\epsilon_g^{1/3}, \epsilon_g^{2/3}]\), this bound improves over the classical \( O(\epsilon_g^{-2}) \) rate of gradient-based methods. The choice \( \epsilon_H = \epsilon_g^{1/2} \) yields the rate \( O(\epsilon_g^{-3/2}) \), which is known to be optimal among second-order methods [9].

Recalling that the workload of Algorithm 1 in terms of Hessian-vector products depends on the index \( J \) defined by (7), we obtain the following corollary.

Corollary 1 Suppose that the assumptions of Theorem 3 are satisfied, and let \( K_1 \) be as defined in that theorem and \( J(U_H, \epsilon_H, \zeta) \) be as defined in (7). Then the number of Hessian-vector products and/or gradient evaluations required by Algorithm 3 to output an iterate satisfying (22) is at most
\[ (2 \min(n, J(U_H, \epsilon_H, \zeta)) + 2)(K_1 + 1) = \tilde{O}\left(\max(\epsilon_g^{-3}, \epsilon_H^{-3/2})\right). \]
Proof. The number of Hessian-vector multiplications in the main loop of Algorithm 1 is bounded by \( J(U_H, \epsilon_H, \zeta) + 1 \). An additional \( J(U_H, \epsilon_H, \zeta) \) Hessian-vector products may be needed to find a direction satisfying (6). Each iteration also requires a single evaluation of the gradient \( \nabla f \), giving a bound of \( 2J(U_H, \epsilon_H, \zeta) + 2 \) on the workload per iteration of Algorithm 3. We multiply this quantity by the iteration bound from Theorem 3 to obtain the result.

By setting \( \epsilon_H = \epsilon_g^{1/2} \), we obtain from this corollary a computational bound of \( \tilde{O}(\epsilon_g^{-7/4}) \), which matches the deterministic first-order guarantee obtained in [6], and also improves over the \( \tilde{O}(\epsilon_g^{-2}) \) computational complexity of gradient-based methods.

4.3 Second-Order Complexity Results

We now find bounds on iteration and computational complexity of finding a point that satisfies (2). In this section, as well as using results from Sections 4.1 and 4.2, we also need to use the properties of the minimum eigenvalue oracle, Procedure 2. To this end, we make the following generic assumption.

Assumption 3 For every iteration \( k \) at which Algorithm 3 calls Procedure 2, and for a specified failure probability \( \delta \) with \( 0 \leq \delta < 1 \), Procedure 2 either certifies that \( \nabla^2 f(x_k) \succeq -\epsilon_H I \) or finds a vector of curvature smaller than \( -\epsilon_H / 2 \) in at most

\[
N_{\text{meo}} := \min \left\{ n, C_{\text{meo}} \epsilon_H^{-1/2} \right\}
\]

Hessian-vector products, where \( C_{\text{meo}} \) may depend on the probability threshold \( \delta \) but not on \( \epsilon_H \).

Assumption 3 encompasses all the strategies we mentioned in Section 3.2. For both the Lanczos method with a random starting vector and the conjugate gradient algorithm with a random right-hand side, (26) holds with \( C_{\text{meo}} = \ln(n/\delta^2) \sqrt{U_H} / 2 \).

The next lemma guarantees termination of the backtracking line search for a negative curvature direction, regardless of whether it is produced by Algorithm 1 or Procedure 2. As for Lemma 4, the result is deterministic.

Lemma 6 Suppose that Assumptions 1 and 2 hold. Suppose that at iteration \( k \) of Algorithm 3, the search direction \( d_k \) is of negative curvature type, obtained either directly from Procedure 2 or as the output of Algorithm 1 and \( d_{\text{type}}=\text{NC} \). Then the backtracking line search terminates with step length \( \alpha_k = \theta^{j_k} \) with \( j_k \leq j_{\text{nc}} + 1 \), where \( j_{\text{nc}} \) is defined as in Lemma 5 and the decrease in the function value resulting from the chosen step length satisfies

\[
f(x_k) - f(x_k + \alpha_k d_k) \geq \frac{c_{\text{nc}}}{8} \epsilon_H^3,
\]

with \( c_{\text{nc}} \) is given in Lemma 5.
Proof. Lemma 5 shows that the claim holds (with a factor of 8 to spare) when the direction of negative curvature is obtained from Algorithm 1. When the direction is obtained from Procedure 2, we have
\[ d_k^T \nabla^2 f(x_k) d_k = -\|d_k\|^3 \leq -\frac{1}{2}\epsilon_H \|d_k\|^2 < 0, \] (28)
from which it follows that \( \|d_k\| \geq \frac{1}{\epsilon_H} \). The result can now be obtained by following the proof of Lemma 5, with \( \frac{1}{2}\epsilon_H \) replacing \( \epsilon_H \).

Theorem 4 Suppose that Assumptions 1, 2, and 3 hold, and define
\[ \tilde{K}_2 := \left[ \frac{3(f(x_0) - f_{low})}{\min\{c_{sol}, c_{nc}/8\}} \max\{c_g^{-3}\epsilon_H^3, \epsilon_H^{-3}\} \right] + 1, \] (29)
where constants \( c_{sol} \) and \( c_{nc} \) are defined in Lemmas 4 and 5, respectively. Then with probability at least \( 1 - \tilde{K}_2\delta \), Algorithm 3 terminates at a point satisfying (2) in at most \( \tilde{K}_2\delta \) iterations. (With probability at most \( \tilde{K}_2\delta \), it terminates incorrectly within \( \tilde{K}_2 \) iterations at a point for which \( \|\nabla f(x_l)\| \leq \epsilon_g \) but \( \lambda_{\text{min}}(\nabla^2 f(x)) < -\epsilon_H \).)

Proof. Algorithm 3 terminates incorrectly with probability \( \delta \) at any iteration at which Procedure 2 is called. The overall probability of this happening during the first \( \tilde{K}_2\delta \) iterations is at most \( \tilde{K}_2\delta \). Therefore, with probability at least \( 1 - \tilde{K}_2\delta \), no incorrect termination occurs in the first \( \tilde{K}_2 \) iterations.

Suppose now for contradiction that Algorithm 3 runs for \( \tilde{K}_2 \) iterations without terminating (correctly). That is, for all \( l = 0, 1, \ldots, \tilde{K}_2 \), we have either \( \|\nabla f(x_l)\| > \epsilon_g \) or \( \lambda_{\text{min}}(\nabla^2 f(x_l)) < -\epsilon_H \). We perform the following partition of the set of iteration indices:
\[ K_1 \cup K_2 \cup K_3 = \{0, 1, \ldots, \tilde{K}_2 - 1\}, \] (30)
where \( K_1, K_2, \) and \( K_3 \) are defined as follows.

Case 1: \( K_1 := \{l = 0, 1, \ldots, \tilde{K}_2 - 1 : \|\nabla f(x_l)\| \leq \epsilon_g\} \). At each iteration \( l \in K_1 \), Algorithm 3 calls the minimum eigenvalue oracle, which does not certify that \( \lambda_{\text{min}}(\nabla^2 f(x_l)) \geq -\epsilon_H \) (since the algorithm continues to iterate) but rather returns a direction of sufficient negative curvature, which by Lemma 6 leads to an improvement in \( f \) that is bounded as follows:
\[ f(x_l) - f(x_{l+1}) \geq \frac{c_{nc}}{8}\epsilon_H^3. \] (31)

Case 2: \( K_2 := \{l = 0, 1, \ldots, \tilde{K}_2 - 1 : \|\nabla f(x_l)\| > \epsilon_g \) and \( \|\nabla f(x_{l+1})\| > \epsilon_g\} \). Algorithm 3 calls Algorithm 1 at each iteration \( l \in K_2 \), returning either an approximate damped Newton or a negative curvature direction. By combining Lemmas 4 and 5, we obtain a decrease in \( f \) satisfying
\[ f(x_l) - f(x_{l+1}) \geq \min\{c_{sol}, c_{nc}\} \min\{\|\nabla f(x_{l+1})\|^{-3}\epsilon_H^{-3}, \epsilon_H^{-3}\} \]
\[ \geq \min\{c_{sol}, c_{nc}/8\} \min\{\epsilon_g^{-3}\epsilon_H^{-3}, \epsilon_H^{-3}\}. \] (32)
Case 3: \( \mathcal{K}_3 := \{ l = 0, 1, \ldots, \bar{K}_2 - 1 : \| \nabla f(x_l) \| > \epsilon_g \geq \| \nabla f(x_{l+1}) \| \} \). Because \( \| \nabla f(x_{l+1}) \| \) may be small in this case, we can no longer bound the decrease in \( f \) by an expression such as (32). We can however guarantee at least that \( f(x_l) - f(x_{l+1}) \geq 0 \). Moreover, provided that \( l < \bar{K}_2 - 1 \), we have from \( \| \nabla f(x_{l+1}) \| \leq \epsilon_g \) that the next iterate \( l + 1 \) is in \( \mathcal{K}_1 \). Thus, a significant decrease in \( f \) will be attained at the next iteration, and we have

\[
|\mathcal{K}_3| \leq |\mathcal{K}_1| + 1. \tag{33}
\]

We now consider the total decrease in \( f \) over the span of \( \bar{K}_2 \) iterations, which is bounded by \( f(x_0) - f_{\text{low}} \) as follows:

\[
f(x_0) - f_{\text{low}} \geq \sum_{l=0}^{\bar{K}_2-1} (f(x_l) - f(x_{l+1})) \\
\geq \sum_{l \in \mathcal{K}_1} (f(x_l) - f(x_{l+1})) + \sum_{l \in \mathcal{K}_2} (f(x_l) - f(x_{l+1})) \tag{34}
\]

where both sums in the final expression are nonnegative. Using first the bound (31) for the sum over \( \mathcal{K}_1 \), we obtain

\[
f(x_0) - f_{\text{low}} \geq |\mathcal{K}_1| \frac{c_{nc}}{8} \epsilon_H^3 \Leftrightarrow |\mathcal{K}_1| \leq \frac{f(x_0) - f_{\text{low}}}{c_{nc}/8} \epsilon_H^{-3}. \tag{35}
\]

Applying (32) to the sum over \( \mathcal{K}_2 \) leads to

\[
|\mathcal{K}_2| \leq \frac{f(x_0) - f_{\text{low}}}{\min\{c_{\text{sol}}, c_{nc}/8\}} \max\{\epsilon_g^{-3} \epsilon_H^{-3}, \epsilon_H^{-3}\}. \tag{36}
\]

Using these bounds together with (33), we have

\[
\bar{K}_2 = |\mathcal{K}_1| + |\mathcal{K}_2| + |\mathcal{K}_3| \\
\leq 2|\mathcal{K}_1| + |\mathcal{K}_2| + 1 \\
\leq 3 \max\{|\mathcal{K}_1|, |\mathcal{K}_2|\} + 1 \\
\leq \frac{3(f(x_0) - f_{\text{low}})}{\min\{c_{\text{sol}}, c_{nc}/8\}} \max\{\epsilon_g^{-3} \epsilon_H^{-3}, \epsilon_H^{-3}\} \\
\leq \bar{K}_2 - 1,
\]

giving the required contradiction. \( \square \)

Finally, as in Section 4.3, we provide a bound on the number of Hessian-vector products and gradient evaluations necessary to achieve (2).

**Corollary 2** Suppose that assumptions of Theorem 4 hold, and let \( \bar{K}_2 \) be defined as in (29). Then with probability at least \( 1 - \bar{K}_2 \delta \), Algorithm 3 terminates at a point satisfying (2) after at most

\[
(1 + \max\{2J(U_H, \epsilon_H, \zeta) + 1, N_{\text{meo}}\}) \bar{K}_2 = \tilde{O}(\epsilon_g^{-3} \epsilon_H^{5/2}, \epsilon_H^{-7/2})
\]
Hessian-vector products and/or gradient evaluations. (With probability at most $K_2\delta$, it terminates incorrectly within this number of Hessian-vector products and/or gradient evaluations at a point for which $\|\nabla f(x_k)\| \leq \epsilon_g$ but $\lambda_{\min}(\nabla^2 f(x)) < -\epsilon_H$.)

Proof. The proof follows by combining Theorem 4 (which bounds the number of iterations) with Lemma 1 and Assumption 3 (which bound the workload per iteration).

By setting $\epsilon_H = \epsilon_g^{1/2}$ and assuming that $n$ is sufficiently large, we recover (with high probability) the familiar complexity bound of order $\tilde{O}(\epsilon^{-7/4})$, matching the bound of accelerated gradient-type methods such as [1,5,19].

5 Discussion

We have presented a Newton-CG approach for smooth nonconvex unconstrained minimization that is close to traditional variants of this method, but which incorporates additional checks and safeguards that enable convergence to a point satisfying approximate second-order conditions (2) with guaranteed complexity. This was achieved by exploiting the properties of linear conjugate gradient in two ways. First, we used CG to compute Newton-type steps when possible, while monitoring convexity during the CG iterations to detect negative curvature directions when those exist. Second, by exploiting the close relationship between the Lanczos and CG algorithms, we show that both methods can be used to detect negative curvature of a given symmetric matrix with high probability. Both techniques are endowed with their own complexity guarantees, and can be combined in a Newton-CG framework with the best known bounds for second-order algorithms on nonconvex optimization [11].

Nonconvexity detection can be introduced into CG in other ways than in Algorithm 1. For instance, we can drop the implicit cap on the number of CG iterations that is due to monitoring of the condition $\|r_j\| > \sqrt{T(1-\tau)^{3/2}}\|r_0\|$ and use of the negative curvature direction generation procedure (6) from Algorithm 1, and instead imposing an explicit cap (smaller by a factor of approximately 4 than $J(M,\epsilon,\zeta)$) on the number of CG iterations. In this version, if the explicit cap is reached without detection of a direction of sufficient negative curvature for $\bar{H}$, then Procedure 2 is invoked to find one. This strategy comes equipped with essentially the same high-probability complexity results as Theorem 4 and Corollary 2, but it lacks the deterministic approximate-first-order complexity guarantee of Theorem 3. On the other hand, it is more elementary, both in the specification of the Capped CG procedure and the analysis.

A common feature to the Capped CG procedures described in Algorithm 1 and in the above paragraph, which also emerges in most Newton-type methods with good complexity guarantees [11], is the need for high accuracy in the step computation. That is, although the solution of the damped Newton system is not exact, only a small residual is allowed at the approximate solution. Looser
restrictions on the accuracy of the subproblems are typically used in practical algorithms, but our tighter bounds appear to be necessary for the complexity analysis. Further investigation of the differences between our procedure in this paper and practical Newton-CG procedures is a subject of ongoing research.

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. Z. Allen-Zhu and Y. Li, NEON2: Finding local minima via first-order oracles. arXiv:1711.06673, 2017.
3. E. G. Birgin and J. M. Martínez, The use of quadratic regularization with a cubic descent condition for unconstrained optimization, SIAM J. Optim., 27 (2017), pp. 1049–1074.
4. S. Bubeck, Convex optimization: Algorithms and complexity, Foundations and Trends® in Machine Learning, 8 (2015), pp. 231–357.
5. Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, Accelerated methods for non-convex optimization. arXiv:1611.00756v2, 2017.
6. ———, “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.
7. C. Cartis, N. I. M. Gould, and P. L. Toint, On the complexity of steepest descent, Newton’s and regularized Newton’s methods for nonconvex unconstrained optimization, SIAM J. Optim., 20 (2010), pp. 2833–2852.
8. ———, Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results, Math. Program., 127 (2011), pp. 245–295.
9. ———, Optimal Newton-type methods for nonconvex optimization, Tech. Rep. naXys-17-2011, Dept of Mathematics, FUNDP, Namur (B), 2011.
10. ———, Complexity bounds for second-order optimality in unconstrained optimization, Journal of Complexity, 28 (2012), pp. 93–108.
11. ———, Worst-case evaluation complexity and optimality of second-order methods for nonconvex smooth optimization. arXiv:1709.07180, 2017.
12. A. R. Conn, N. I. M. Gould, and P. L. Toint, Trust-Region Methods. MPS-SIAM Series on Optimization, Society for Industrial and Applied Mathematics, Philadelphia, 2000.
13. F. E. Curtis, D. P. Robinson, and M. Samadi, An inexact regularized Newton framework with a worst-case iteration complexity of $O(\epsilon^{-3/2})$ for nonconvex optimization, Tech. Rep. 17T-011, COR@L Laboratory, Department of ISE, Lehigh University, 2017.
14. ———, A trust region algorithm with a worst-case iteration complexity of $O(\epsilon^{-3/2})$ for nonconvex optimization, Math. Program., 162 (2017), pp. 1–32.
15. R. S. Dembo and T. Steihaug, Truncated-Newton algorithms for large-scale unconstrained optimization, Math. Program., 26 (1983), pp. 190–212.
16. G. Fasano and S. Lucidi, A nonmonotone truncated Newton-Krylov method exploiting negative curvature directions, for large-scale unconstrained optimization, Optim. Lett., 3 (2009), pp. 521–535.
17. N. I. M. Gould, S. Lucidi, M. Roma, and P. L. Toint, Exploiting negative curvature directions in linesearch methods for unconstrained optimization, Optim. Methods Softw., 14 (2000), pp. 75–98.
18. A. Griewank and A. Walther, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, second ed., 2008.
19. C. Jin, P. Netrapalli, and M. I. Jordan, Accelerated gradient descent escapes saddle points faster than gradient descent. arXiv:1711.10456, 2017.
A Newton-CG Algorithm with Complexity Guarantees

20. S. Karimi and S. A. Vavasis, A unified convergence bound for conjugate gradient and accelerated gradient. arXiv:1605.00320, 2016.

21. A single potential governing convergence of conjugate gradient, accelerated gradient and geometric descent. arXiv:1712.09498, 2017.

22. 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.

23. J. M. Martínez and M. Raydan, Cubic-regularization counterpart of a variable-norm trust-region method for unconstrained minimization, Journal of Global Optimization, 68 (2017), pp. 367–385.

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

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

26. C. W. Royer and S. J. Wright, Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization, SIAM J. Optim., (2018 (to appear)).

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

28. Y. Xu, R. Jin, and T. Yang, NEON+: Accelerated gradient methods for extracting negative curvature for non-convex optimization. arXiv:1712.01033, 2017.

A Linear Conjugate Gradient: Relevant Properties

In this appendix, we provide useful results for the classical CG algorithm, that originate from its definition and thus also apply to the “standard CG” operations within Algorithm 4. To this end, and for the sake of discussion below in Appendix B, we sketch the standard CG method in Algorithm 4, reusing the notation of Algorithm 1.

Algorithm 4 Conjugate Gradient

Inputs: Symmetric matrix $\bar{H}$, vector $g$;

$r_0 \leftarrow g$, $p_0 \leftarrow -r_0$, $y_0 \leftarrow 0$, $j \leftarrow 0$;

while $p_j^\top \bar{H} p_j > 0$ and $r_j \neq 0$ do

$\alpha_j \leftarrow r_j^\top r_j / p_j^\top \bar{H} p_j$;

$y_{j+1} \leftarrow y_j + \alpha_j p_j$;

$r_{j+1} \leftarrow r_j + \alpha_j \bar{H} p_j$;

$\beta_{j+1} \leftarrow (r_{j+1}^\top r_{j+1}) / (r_j^\top r_j)$;

$p_{j+1} \leftarrow -r_{j+1} + \beta_{j+1} p_j$;

$j \leftarrow j + 1$;

end while

Here and below, we refer often to the following quadratic function:

$$q(y) := \frac{1}{2} y^\top \bar{H} y + g^\top y,$$

where $\bar{H}$ and $g$ are the matrix and vector parameters of Algorithms 1 or 4. When $\bar{H}$ is positive definite, the minimizer of $q$ is identical to the unique solution of $\bar{H} y = -g$. CG can be viewed either as an algorithm to solve $\bar{H} y = -g$ or as an algorithm to minimize $q$.

The next lemma details several properties of the conjugate gradient method to be used in the upcoming proofs.

Lemma 7 Suppose that $j$ iterations of the CG loop are performed in Algorithm 1 or 4. Then, we have

$$\frac{p_i^\top \bar{H} p_i}{\|p_i\|^2} > 0 \quad \text{for all } i = 0, 1, \ldots, j - 1.$$
Moreover, the following properties hold:

1. $y_i \in \text{span}\{p_0, \ldots, p_{i-1}\}$, $i = 1, 2, \ldots, j$;
2. $r_i \in \text{span}\{p_0, \ldots, p_i\}$ for all $i = 1, 2, \ldots, j$, and
   \[ r_i^Tv = 0, \quad \text{for all } v \in \text{span}\{p_0, \ldots, p_{i-1}\} \text{ and all } i = 1, 2, \ldots, j, \]
   (in particular $r_i^Tr_i = 0$ if $0 \leq i < l \leq j$, and if $j = n$, then $r_n = 0$);
3. $\|r_i\| \leq \|p_i\|$, $i = 0, 1, \ldots, j$;
4. $r_i^tp_i = -\|r_i\|^2$, $i = 0, 1, \ldots, j$;
5. $p_i^T \bar{H}p_k = 0$ for all $i, k = 0, 1, \ldots, j$ with $k \neq i$;
6. $p_i = -\sum_{k=0}^{i+1} \frac{\|r_i\|^2}{\|r_{k+1}\|^2} r_k$, $i = 0, 1, \ldots, j$;
7. $q(y_{i+1}) = q(y_i) - \frac{\|r_i\|^4}{2p_i^T \bar{H}p_i}$, $i = 0, 1, \ldots, j$;
8. $r_i^T \bar{H}r_i \geq p_i^T \bar{H}p_i$, $i = 0, 1, \ldots, j$.

Proof. Since CG has not terminated prior to iteration $j$, (38) clearly holds. All properties then follow from the definition of the CG process, and most are proved in standard texts (see, for example, [25, Chapter 5]). Property 8 is less commonly used, so we provide a proof here.

The case $i = 0$ is immediate since $r_0 = -p_0$ and there is equality. When $i \geq 1$, we have:

$$p_i = -r_i + \frac{\|r_i\|^2}{\|p_{i-1}\|^2} p_{i-1} \iff r_i = -p_i + \frac{\|r_i\|^2}{\|p_{i-1}\|^2} p_{i-1}.$$  

(Note that if iteration $i$ is reached, we cannot have $\|r_{i+1}\| = 0$.) It follows that

$$r_i^T \bar{H}r_i = p_i^T \bar{H}p_i - 2 \frac{\|r_i\|^2}{\|p_{i-1}\|^2} p_i^T \bar{H}p_{i-1} + \frac{\|r_i\|^4}{\|p_{i-1}\|^4} p_i^T \bar{H}p_{i-1}$$

$$= p_i^T \bar{H}p_i + \frac{\|r_i\|^4}{\|p_{i-1}\|^4} p_i^T \bar{H}p_{i-1},$$

as $p_i^T \bar{H}p_{i-1} = 0$ by Property 5 above. Since iteration $i$ has been reached, $p_{i-1}$ is a direction of curvature at least $\epsilon$, and we obtain $r_i^T \bar{H}r_i \geq p_i^T \bar{H}p_i$, as required. \(\Box\)

We next address an important technical point about Algorithm 1: the test (6) to identify a direction of negative curvature for $H$ after an insufficiently rapid rate of reduction in the residual norm $\|r_i\|$ has been observed. As written, the formula (6) suggests both that previous iterations $y_i, i = 1, 2, \ldots, j-1$ must be stored (or regenerated) and that additional matrix-vector multiplications (specifically, $\bar{H}(y_{j+1} - y_i)$, $i = 0, 1, \ldots$) must be performed. We show here that in fact (6) can be evaluated at essentially no cost, provided we store two extra scalars at each iteration of CG: the quantities $\alpha_k$ and $\|r_k\|^2$, for $k = 0, 1, \ldots, j$.

**Lemma 8** Suppose that Algorithm 1 computes iterates up to iteration $j + 1$. Then, for any $i \in \{0, \ldots, j\}$, we can compute (6) as

$$\frac{(y_{j+1} - y_i)^T \bar{H}(y_{j+1} - y_i)}{\|y_{j+1} - y_i\|^2} = \frac{\sum_{k=0}^{j} \alpha_k \|r_k\|^2}{\sum_{\ell=0}^{j} \left[ \sum_{k=\max\{\ell, i\}}^{j} \alpha_k \|r_k\|^2 \right]^2 / \|r_{\ell}\|^2}.$$  

Proof. By definition, $y_{j+1} - y_i = \sum_{k=0}^{j} \alpha_k p_k$. By conjugacy of the $p_k$ vectors, we have

$$\frac{(y_{j+1} - y_i)^T \bar{H}(y_{j+1} - y_i)}{\|y_{j+1} - y_i\|^2} = \sum_{k=0}^{j} \alpha_k^2 p_k^T \bar{H}p_k = \sum_{k=0}^{j} \alpha_k \|r_k\|^2,$$  

(39)
where we used the definition of $\alpha_k$ to obtain the last equality. Now we turn our attention to the denominator. Using Property 6 of Lemma 7, we have that
\[
y_{j+1} - y_i = \sum_{k=i}^{j} \alpha_k p_k = \sum_{k=i}^{j} \alpha_k \left( -\sum_{\ell=0}^{k} \frac{\|r_\ell\|^2}{\|p_\ell\|^2} r_\ell \right),
\]
By rearranging the terms in the sum, we obtain
\[
y_{j+1} - y_i = -\sum_{k=i}^{j} \sum_{\ell=0}^{k} \alpha_k \frac{\|r_\ell\|^2}{\|p_\ell\|^2} p_\ell = -\sum_{\ell=0}^{j} \left[ \sum_{k=\max(\ell,i)}^{j} \alpha_k \|r_\ell\|^2 \right] \frac{p_\ell}{\|p_\ell\|^2}.
\]
Using the fact that the residuals $\{r_\ell\}_{\ell=0}^{j}$ form an orthogonal set (by Property 2 of Lemma 7), we have that
\[
\|y_{j+1} - y_i\|^2 = \sum_{\ell=0}^{j} \frac{1}{\|p_\ell\|^2} \left[ \sum_{k=\max(\ell,i)}^{j} \alpha_k \|r_\ell\|^2 \right]^2.
\]
Combining this with (39) gives the desired result. \( \Box \)

**B Relating Lanczos and Conjugate Gradient Procedures for Detecting Negative Curvature**

In this appendix, we analyze the CG method (Algorithm 4) applied to a (possibly nonconvex) quadratic function with a random linear term. We show that the number of iterations required by CG to detect nonpositive curvature in an indefinite matrix is the same as the number required by Lanczos, when the two approaches are initialized in a consistent way, thereby proving Theorem 1.

Consider first the Lanczos algorithm applied to a symmetric, $n$-by-$n$ matrix $\bar{H}$ and a (not necessarily random) starting vector $\bar{b} \in \mathbb{R}^n$ with $\|\bar{b}\| = 1$. After $t+1$ iterations, Lanczos returns
\[
z_{t+1} := \arg \min_z \frac{1}{2} z^\top \bar{H} z \quad \text{subject to } \|z\|_2 = 1, z \in \mathcal{K}_i(b, \bar{H}), \text{ for } t = 0, 1, \ldots, (40)
\]
where the $t$th Krylov subspace $\mathcal{K}_i(b, \bar{H})$ is defined by
\[
\mathcal{K}_i(b, \bar{H}) = \text{span}\{b, \bar{H}b, \ldots, \bar{H}^i b\}. \quad (41)
\]

Consider now Algorithm 4 applied to $\bar{H}$ and $g = -b$. By Lemma 7 Property 2, we can see that if Algorithm 4 does not terminate with $j \leq t$, for some given index $t$, then $y_{t+1}$, $r_{t+1}$, and $p_{t+1}$ are computed, and we have
\[
\text{span}\{p_0, \ldots, p_i\} = \text{span}\{r_0, \ldots, r_i\} = \mathcal{K}_i(b, \bar{H}), \quad \text{for } i = 0, 1, \ldots, t, \quad (42)
\]
because $\{r_\ell\}_{\ell=0}^{i+1}$ is a set of $i+1$ orthogonal vectors in $\mathcal{K}_i(b, \bar{H})$. Thus $\{p_0, \ldots, p_i\}$, $\{r_0, \ldots, r_i\}$, and $\{b, \bar{H}b, \ldots, \bar{H}^i b\}$ are all bases for $\mathcal{K}_i(b, \bar{H})$, $i = 0, 1, \ldots, t$. As long as they are computed, the iterates of Algorithm 4 satisfy
\[
y_{t+1} := \arg \min_y \frac{1}{2} y^\top H y - b^\top y \quad \text{subject to } y \in \mathcal{K}_i(b, \bar{H}), \text{ for } t = 0, 1, \ldots, (43)
\]
Based on this relationship between Lanczos and CG via the Krylov subspace, we have the following result about the number of iterations required by CG to detect non-positive-definiteness.
Theorem 5 Consider applying Algorithm 4 to the quadratic function (37), with \( q = -b \) for some \( b \) with \( \|b\| = 1 \). Let \( J \) be the smallest value of \( t \geq 0 \) such that \( K_t(b, \bar{H}) \) contains a direction of nonpositive curvature, so that \( J \) is also the smallest index \( t \geq 0 \) such that \( z_{t+1}^\top H z_{t+1} \leq 0 \), where \( \{z_t\} \) are the Lanczos iterates from (40). Then Algorithm 4 terminates with \( j = J \), with \( \bar{p}_j^\top \bar{H} p_j \leq 0 \).

Proof. We consider first the case of \( J = 0 \). Then \( z_1 = b/\|b\| \) and \( b^\top \bar{H} b \leq 0 \), so since \( p_0 = -r_0 = b \), we have \( p_0^\top \bar{H} p_0 = 0 \), so the result holds in this case. We assume that \( J \geq 1 \) for the remainder of the proof.

Suppose first that Algorithm 4 terminates with \( j = t \), for some \( t \) satisfying \( 1 \leq t \leq J \), because of a zero residual — \( r_t = 0 \) — without having encountered nonpositive curvature. In that case, we can show that \( \bar{H} b \in \text{span}\{b, \ldots, \bar{H}^{t-1} b\} \).

We invoke (42) with \( t \) replaced by \( t-1 \) since Algorithm 4 has not terminated at iteration \( t-1 \). By the recursive definition of \( r_{t-1} \) within Algorithm 4, there exist coefficients \( \tau_i \) and \( \sigma_i \) such that

\[
r_{t-1} = -b + \sum_{i=1}^{t-1} \tau_i \bar{H}^i b, \quad p_{t-1} = \sum_{i=0}^{t-1} \sigma_i \bar{H}^i b.
\]

Since \( r_t = 0 \), we have again from Algorithm 4 that

\[
0 = r_{t-1} + \alpha_{t-1} \bar{H} p_{t-1} = -b + \sum_{i=1}^{t-1} (\tau_i + \alpha_{t-1} \sigma_{t-1}) \bar{H}^i b + \alpha_{t-1} \sigma_{t-1} \bar{H}^t b.
\]

The coefficient \( \alpha_{t-1} \sigma_{t-1} \) must be nonzero, because otherwise this expression would represent a nontrivial linear combination of the basis elements \( \{b, \bar{H} b, \ldots, \bar{H}^{t-1} b\} \) of \( K_{t-1}(b, \bar{H}) \) that is equal to zero. It follows from this observation and (44) that \( \bar{H} b \in \text{span}\{b, \bar{H} b, \ldots, \bar{H}^{t-1} b\} = K_{t-1}(b, \bar{H}) \), as required.

Consequently,

\[
K_i(b, \bar{H}) = \text{span}\{b, \bar{H} b, \ldots, \bar{H}^i b\} = \text{span}\{b, \ldots, \bar{H}^{t-1} b\} = K_{t-1}(b, \bar{H}).
\]

By using a recursive argument on the definition of \( K_i(b, \bar{H}) \) for \( i = t, \ldots, J \), we arrive at \( K_{t-1}(b, \bar{H}) = K_j(b, \bar{H}) \). Thus there is a value of \( t \) smaller than \( J \) such that \( K_t(b, \bar{H}) \) contains a direction of nonpositive curvature, contradicting our definition of \( J \). Thus we cannot have termination of Algorithm 4 with \( j \leq J \) unless \( p_j^\top \bar{H} p_j \leq 0 \).

Suppose next that CG terminates with \( j = t \) for some \( t > J \). It follows that \( p_{J}^\top \bar{H} p_{j} > 0 \) for all \( j = 0, 1, \ldots, J \). By definition of \( J \), there is a nonzero vector \( z \in K_j(b, \bar{H}) \) such that \( z^\top \bar{H} z \leq 0 \). On the other hand, we have \( K_j(b, \bar{H}) = \text{span}\{p_0, p_1, \ldots, p_J\} \) by (42), thus, we can write \( z = \sum_{j=0}^{J} \gamma_j p_j \), for some scalars \( \gamma_j, j = 0, 1, \ldots, J \). By Property 5 of Lemma 7, we then have

\[
0 \geq z^\top \bar{H} z = \sum_{j=0}^{J} \gamma_j^2 p_j^\top \bar{H} p_j.
\]

Since \( p_j^\top \bar{H} p_j > 0 \) for every \( j = 0, 1, \ldots, J \), and not all \( \gamma_j \) can be zero (because \( z \neq 0 \)), the final summation is strictly positive, a contradiction.

Suppose now that CG terminates at some \( j < J \). Then \( p_j^\top \bar{H} p_j \leq 0 \), and since \( p_j \in K_j(b, \bar{H}) \), it follows that \( K_j(b, \bar{H}) \) contains a direction of nonpositive curvature, contradicting the definition of \( J \).

We conclude that Algorithm 4 must terminate with \( j = J \) and \( p_j^\top \bar{H} p_j \leq 0 \), as claimed. \( \square \)

Theorem 5 is a generic result that does not require \( b \) to be chosen randomly. It does not guarantee that Lanczos will detect nonpositive curvature in \( H \) whenever present, because \( b \) could be orthogonal to the subspace corresponding to the nonpositive curvature, so the Lanczos subspace never intersects with the subspace of negative curvature. When \( b \) is chosen randomly from a uniform distribution over the unit ball, however, we can certify the
performance of Lanczos, as we have shown in Lemma 2 and [22]. We can exploit Theorem 5 to obtain the same performance for CG, as stated in Theorem 1. We restate this result as a corollary, and prove it now.

**Corollary 3** Let \( b \) be distributed uniformly on the unit ball and \( H \) be a symmetric \( n \times n \) matrix, with \( \|H\| \leq M \). Given \( \delta \in (0,1) \), define
\[
\hat{J} := \min \left\{ n, \left\lceil \frac{\ln(n/\delta^2)}{2} \sqrt{\frac{M}{\epsilon}} \right\rceil \right\}.
\]

Consider applying Algorithm 4 with \( H := H + \frac{1}{\epsilon}I \) and \( g = -b \). Then, the following properties hold:

(a) If \( \lambda_{\min}(H) < -\epsilon \), then with probability at least \( 1 - \delta \), there is some index \( j \leq \hat{J} \) such that Algorithm 4 terminates with a direction \( p_j \) such that \( p_j^\top H p_j \leq -\epsilon/2 \).

(b) If Algorithm 4 runs for \( \hat{J} \) iterations without terminating, then with probability at least \( 1 - \delta \), we have that \( \lambda_{\min}(H) \geq -\epsilon \).

**Proof.** We exploit the invariance of the Krylov subspaces to linear shifts, that is,
\[
K_j(b, H) = K_j(b, H) = K_j(b, M I - H), \quad \text{for all } j = 0, 1, \ldots
\]

This fact allows us to make inferences about the behavior of Algorithm 4 applied to \( \hat{H} \) from the behavior of the Lanczos method applied to \( M I - H \), which has been described in Lemma 2.

Suppose first that \( \lambda_{\min}(H) < -\epsilon \). By Lemma 2, we know that with probability at least \( 1 - \delta \), the Lanczos procedure returns a vector \( v \) such that \( \|v\| = 1 \) and \( v^\top H v \leq -(\epsilon/2) \) after at most \( J \) iterations. Thus, for some \( j \leq \hat{J} \), we have \( v \in K_j(b, H) = K_j(b, H) \), and moreover \( v^\top H v \leq 0 \) by definition of \( H \), so the Krylov subspace \( K_j(b, H) \) contains directions of nonpositive curvature, for some \( j \leq \hat{J} \). It then follows from Theorem 5 that \( p_j^\top H p_j \leq 0 \) for some \( j \leq \hat{J} \). To summarize: If \( \lambda_{\min}(H) < -\epsilon \), then with probability \( 1 - \delta \), Algorithm 4 applied to \( \hat{H} \) and \( g = -b \) will terminate with some \( p_j \) such that \( p_j^\top H p_j \leq 0 \) for some \( j \) with \( j \leq \hat{J} \). The proof of (a) is complete.

Suppose now that Algorithm 4 applied to \( \hat{H} \) and \( g = -b \) runs for \( \hat{J} \) iterations without terminating, that is \( p_j^\top H p_j > 0 \) for \( j = 0, 1, \ldots, \hat{J} \). It follows from the logic of Theorem 5 that \( K_j(b, H) \) contains no directions of nonpositive curvature for \( H \). Equivalently, there is no direction of curvature less than \( -\epsilon/2 \) for \( H \) in \( K_j(b, H) \). By Lemma 2, this certify with probability at least \( 1 - \delta \) that \( \lambda_{\min}(H) \geq -\epsilon \), establishing (b). \( \Box \)

### C Proof of Theorem 2

**Proof.** The proof proceeds by contradiction: If we assume that all conditions specified in the statement of the theorem hold, and in addition that
\[
\frac{(y_{j+1} - y_i)^\top \hat{H}(y_{j+1} - y_i)}{\|y_{j+1} - y_i\|^2} \geq \epsilon, \quad \text{for all } i = 0, 1, \ldots, \hat{J} - 1,
\]
then we must have
\[
\|r_j\| \leq \sqrt{T}(1 - \tau)^{J/2} \|\tau_0\|,
\]
contradicting (10). Note that
\[
\frac{(y_{j+1} - y_j)^\top \hat{H}(y_{j+1} - y_j)}{\|y_{j+1} - y_j\|^2} = \frac{\alpha_j p_j^\top \hat{H}(\alpha_j p_j)}{\|\alpha_j p_j\|^2} = \frac{p_j^\top \hat{H} p_j}{\|p_j\|^2} \geq \epsilon
\]

by assumption, therefore we can consider (47) to hold for \( i = 0, 1, \ldots, J \). Moreover, recalling the definition (37) of the quadratic function \( q \), we have for any \( i = 0, 1, \ldots, J \) that

\[
q(y_{j+1}) = q(y_j) + g_j^\top (y_{j+1} - y_j) + \frac{1}{2} (y_{j+1} - y_j)^\top H(y_{j+1} - y_j).
\]

Thus, (47) can be reformulated as

\[
q(y_{j+1}) \geq q(y_j) + g_j^\top (y_{j+1} - y_j) + \frac{\epsilon}{2} \|y_{j+1} - y_j\|^2, \quad \text{for all } i = 0, 1, \ldots, J,
\]

where we used \( \nabla q(y_i) = r_i \) and the definitions (47), (49) of strong convexity along the directions \( y_{j+1} - y_i \). In the remainder of the proof, and similarly to [6, Proof of Proposition 1], we will show that (50) leads to the contradiction (48), thus proving that (11) holds.

We define the sequence of functions \( \Phi_j, j = 0, 1, \ldots, J \) as follows:

\[
\Phi_0(z) := q(y_0) + \frac{\epsilon}{2} \|z - y_0\|^2,
\]

and for \( j = 0, \ldots, J - 1 \):

\[
\Phi_{j+1}(z) := (1 - \tau) \Phi_j(z) + \tau \left( q(y_j) + r_j^\top (z - y_j) + \frac{\epsilon}{2} \|z - y_j\|^2 \right).
\]

Since each \( \Phi_j \) is a quadratic function with Hessian \( \epsilon I \), it can be written as follows:

\[
\Phi_j(z) = \Phi_j^* + \frac{\epsilon}{2} \|z - v_j\|^2,
\]

where \( v_j \) is the unique minimizer of \( \Phi_j \), and \( \Phi_j^* = \Phi_j(v_j) \) is the minimum value of \( \Phi_j \). (Note that \( v_0 = y_0 = 0 \) and \( \Phi_0^* = q(y_0) = 0 \).

Defining \( \psi(y) := q(y_0) - q(y) + \frac{\epsilon}{2} \|y - y_0\|^2 = \Phi_0(y) - q(y) \),

we give a short inductive argument to establish that

\[
\Phi_j(y_{j+1}) \leq q(y_{j+1}) + (1 - \tau)^j \psi(y_{j+1}), \quad j = 0, 1, \ldots, J.
\]

For \( j = 0 \), (54) holds because \( \Phi_0(y) = q(y) + \psi(y) \) by definition. Assuming that (54) holds for some index \( j \geq 0 \), we find by first applying (51) (with \( z = y_{j+1} \)) and then (50) (with \( i = j \)) that

\[
\Phi_{j+1}(y_{j+1}) = (1 - \tau) \Phi_j(y_{j+1}) + \tau \left( q(y_j) + r_j^\top (y_{j+1} - y_j) + \frac{\epsilon}{2} \|y_{j+1} - y_j\|^2 \right)
\]

\[
\leq (1 - \tau) \Phi_j(y_{j+1}) + \tau q(y_{j+1}).
\]

Thus, we have

\[
\Phi_{j+1}(y_{j+1}) \leq (1 - \tau) \Phi_j(y_{j+1}) + \tau q(y_{j+1})
\]

\[
\leq (1 - \tau) q(y_{j+1}) + (1 - \tau)^{j+1} \psi(y_{j+1}) + \tau q(y_{j+1}) \quad \text{from (54)}
\]

\[
= q(y_{j+1}) + (1 - \tau)^{j+1} \psi(y_{j+1}),
\]

which proves (54) for \( j + 1 \), and thus completes the inductive argument.

We next prove another technical fact about the relationship between \( q(y_j) \) and \( \Phi_j^* \), namely,

\[
q(y_j) \leq \Phi_j^*, \quad j = 0, 1, \ldots, J.
\]

We establish this result by an inductive argument that is quite lengthy and technical; we note the termination of this phase of the proof clearly below.
This result trivially holds (with equality) at $j = 0$. Supposing that it holds for some $j = 0, 1, \ldots, J - 1$, we will prove that it also holds for $j + 1$.

By using Properties 7 and 8 of Lemma 7, and also $\|H\| \leq (M + 2\epsilon)$, we have

$$q(y_{j+1}) = q(y_j) - \frac{\|r_j\|^4}{2p_j H p_j} \leq q(y_j) - \frac{\|r_j\|^4}{2r_j \tau H r_j} \leq q(y_j) - \frac{\|r_j\|^2}{2(M + 2\epsilon)}.$$ 

It follows from induction hypothesis $q(y_j) \leq \Phi^*_j$ that

$$q(y_{j+1}) \leq q(y_j) - \frac{\|r_j\|^2}{2(M + 2\epsilon)} = (1 - \tau)q(y_j) + \tau q(y_j) - \frac{\|r_j\|^2}{2(M + 2\epsilon)} \leq (1 - \tau)\Phi^*_j + \tau q(y_j) - \frac{\|r_j\|^2}{2(M + 2\epsilon)}. \quad (56)$$

By taking the derivative in (51), and using (52), we obtain

$$\nabla \Phi_{j+1}(z) = (1 - \tau)\nabla \Phi_{j}(z) + \tau [r_j + \epsilon(z - y_j)] \Rightarrow \epsilon(z - v_{j+1}) = \epsilon[(1 - \tau)(z - v_j) + \tau (r_j + \epsilon(z - y_j))].$$

By rearranging the above relation (and noting that the $z$ terms cancel out), we obtain:

$$v_{j+1} = (1 - \tau)v_j - \frac{\tau}{\epsilon} r_j + \tau y_j. \quad (57)$$

It follows from this expression together with Properties 1 and 2 of Lemma 7 that

$$v_j \in \text{span} \{p_0, p_1, \ldots, p_{j-1}\}, \quad j = 1, 2, \ldots, J. \quad (58)$$

(The result holds for $j = 1$, from (57) we have $v_1 \in \text{span} \{v_0, r_0, y_0\} = \text{span} \{v_0\} = \text{span} \{p_0\}$, and an induction based on (57) can be used to establish (58) for the other values of $j$.) By combining the expressions (52) for $\Phi_j$, $\Phi_{j+1}$ with the recurrence formula (51) for $\Phi_{j+1}$, we obtain

$$\Phi^*_{j+1} + \frac{\epsilon}{2}\|y_j - v_{j+1}\|^2 = \Phi_{j+1}(y_j) = (1 - \tau)\Phi_j(y_j) + \tau q(y_j) = (1 - \tau) \left( \Phi^*_j + \frac{\epsilon}{2}\|y_j - v_j\|^2 \right) + \tau q(y_j)$$

and therefore

$$\Phi^*_{j+1} = (1 - \tau) \left( \Phi^*_j + \frac{\epsilon}{2}\|y_j - v_j\|^2 \right) + \tau q(y_j) - \frac{\epsilon}{2}\|y_j - v_{j+1}\|^2. \quad (59)$$

On the other hand, we have by (57) that

$$\|y_j - v_{j+1}\|^2 = \left\|(1 - \tau)(y_j - v_j) + \frac{\tau}{\epsilon} r_j \right\|^2 = (1 - \tau)^2\|y_j - v_j\|^2 + \frac{\tau^2}{\epsilon^2}\|r_j\|^2 + \frac{\tau}{\epsilon}(1 - \tau)r_j^\top(y_j - v_j)$$

$$= (1 - \tau)^2\|y_j - v_j\|^2 + \frac{\tau^2}{\epsilon^2}\|r_j\|^2,$$ 

where the last relation comes from $r_j \perp \text{span} \{p_0, \ldots, p_{j-1}\}$ (Property 2 of Lemma 7) and (58) in the case $j \geq 1$, and immediately in the case $j = 0$, since $y_0 = v_0 = 0$. By combining
(59) and (60), we arrive at:
\[
\Phi^*_{j+1} = (1 - \tau) \left( \Phi^*_j + \frac{\epsilon}{2} \|y_j - v_j\|^2 \right) + \tau q(y_j) - \frac{\epsilon}{2} \|y_j - v_{j+1}\|^2
\]
\[
= (1 - \tau) \left( \Phi^*_j + \frac{\epsilon}{2} \|y_j - v_j\|^2 \right) + \tau q(y_j) - \frac{\epsilon}{2} (1 - \tau)^2 \|y_j - v_j\|^2 - \frac{\tau^2}{2 \epsilon} \|r_j\|^2
\]
\[
= (1 - \tau) \Phi^*_j + \frac{\epsilon}{2} \left[ 1 - \tau - (1 - \tau)^2 \right] \|y_j - v_j\|^2 + \tau q(y_j) - \frac{\tau^2}{2 \epsilon} \|r_j\|^2
\]
\[
\geq (1 - \tau) \Phi^*_j + \tau q(y_j) - \frac{\tau^2}{2 \epsilon} \|r_j\|^2
\]
\[
\geq q(y_{j+1}) + \frac{1}{2(M + 2 \epsilon)} \|r_j\|^2 - \frac{\tau^2}{2 \epsilon} \|r_j\|^2.
\] (61)

where the last inequality comes from (56). By using the definitions of \( \tau \) and \( \kappa \) in Algorithm 1, we have
\[
\frac{\tau^2}{2 \epsilon} = \frac{1}{2 \epsilon (\sqrt{\kappa} + 1)^2} \leq \frac{1}{2 \epsilon \kappa} = \frac{1}{2(M + 2 \epsilon)}.
\]
It therefore follows from (61) that \( q(y_{j+1}) \leq \Phi^*_{j+1} \). At this point, we have shown that when \( q(y_j) \leq \Phi^*_j \) for \( j = 0, 1, \ldots, J - 1 \), it follows that \( q(y_{j+1}) \leq \Phi^*_{j+1} \), establishing the inductive step. As a result, our proof of (55) is complete.

By substituting \( j = J \) into (55), we obtain \( q(y_J) \leq \Phi^*_J \), which in combination with (54) with \( j = J \), and the definition (52), yields
\[
q(y_J) - q(y_{J+1}) \leq \Phi^*_J - q(y_{J+1}) \leq \Phi^*_J (y_{J+1}) - q(y_{J+1}) \leq (1 - \tau)^J \psi(y_{J+1}).
\] (62)

By substitution from (53), we obtain
\[
q(y_J) - q(y_{J+1}) \leq (1 - \tau)^J \left( q(y_0) - q(y_{J+1}) + \frac{\epsilon}{2} \|y_0 - y_{J+1}\|^2 \right).
\] (63)

We now depart from the analysis of [6], and complete the proof of this result by expressing (63) in terms of residual norms. On the left-hand side, we have
\[
q(y_J) - q(y_{J+1}) = r^\top_{J+1} (y_J - y_{J+1}) + \frac{1}{2} (y_J - y_{J+1})^\top H (y_J - y_{J+1})
\]
\[
= \frac{1}{2} (y_J - y_{J+1})^\top H (y_J - y_{J+1})
\]
because \( r^\top_{J+1} (y_J - y_{J+1}) = r^\top_{J+1} (\alpha_J p_J) = 0 \) by Lemma 7, Property 2. We thus have from (49) that
\[
q(y_J) - q(y_{J+1}) \geq \frac{\epsilon}{2} \|y_J - y_{J+1}\|^2
\]
\[
\geq \frac{\epsilon}{2(M + 2 \epsilon)^2} \|H (y_J - y_{J+1})\|^2 \quad \text{(since } H \preceq (M + 2 \epsilon) I \text{)}
\]
\[
= \frac{\epsilon}{2(M + 2 \epsilon)^2} \|r_J - r_{J+1}\|^2 \quad \text{(since } r_J = g + H y_J \text{)}
\]
\[
= \frac{\epsilon}{2(M + 2 \epsilon)^2} (\|r_J\|^2 + \|r_{J+1}\|^2) \quad \text{(by Lemma 7, Property 2)}
\]
\[
\geq \frac{\epsilon}{2(M + 2 \epsilon)^2} \|r_J\|^2,
\] (64)
On the right-hand side of (63), because of the strong convexity condition (50) at $i = 0$, we have
\[
q(y_0) - q(y_{j+1}) + \frac{\epsilon}{2} \|y_0 - y_{j+1}\|^2 \leq -\nabla q(y_0)\top (y_{j+1} - y_0) \\
= -r_0\top (y_{j+1} - y_0) \leq \|r_0\|\|y_{j+1} - y_0\|.
\]
Moreover, we have
\[
\|y_{j+1} - y_0\| = \left\| \sum_{i=0}^{J} \alpha_ip_i \right\| \leq \sum_{i=0}^{J} \alpha_i \|p_i\| = \sum_{i=0}^{J} \frac{\|r_i\|^2}{\|p_i Hp_i\|},
\]
where the last relation follows from the definition of $\alpha_i$. By combining these last two bounds, and using Property 3 of Lemma 7, we obtain
\[
q(y_0) - q(y_{j+1}) + \frac{\epsilon}{2} \|y_0 - y_{j+1}\|^2 \leq \|r_0\| \sum_{i=0}^{J} \|r_i\| \frac{\|p_i\|^2}{\|p_i Hp_i\|} \leq \|r_0\| \frac{1}{\epsilon} \sum_{i=0}^{J} \|r_i\|,
\]
where the last inequality holds $p_j\top Hp_j \geq \epsilon\|p_j\|^2$ for $j = 0, 1, \ldots, J$ by assumption.

To bound the sum in (65), we recall that since Algorithm 1 did not terminate until iteration $J$, all residuals up to iteration $J - 1$ must have decreased at the expected convergence rate. That is, for every $i = 0, 1, \ldots, J - 1$, we have
\[
\|r_i\| \leq \sqrt{T}(1 - \tau)^{i/2}\|r_0\| \leq (1 - \tau)^{(i-1)/2}\|r_j\|,
\]
where we used $\|r_j\| \geq \sqrt{T}(1 - \tau)^{j/2}\|r_0\|$ (from (10)) for the last inequality. Observing that this bound also holds (trivially) for $i = J$, we obtain by substituting in (65) that
\[
q(y_0) - q(y_{j+1}) + \frac{\epsilon}{2} \|y_0 - y_{j+1}\|^2 \leq \|r_0\| \frac{1}{\epsilon} \sum_{i=0}^{J} (1 - \tau)^{(i-1)/2}\|r_j\| \\
\leq \|r_0\| \frac{(1 - \tau)^{-J/2}}{\epsilon} \|r_j\| \sum_{i=0}^{J} (\sqrt{1 - \tau})^i \\
\leq \|r_0\| \frac{(1 - \tau)^{-J/2}}{\epsilon} \|r_j\| \frac{1}{1 - \sqrt{1 - \tau}}.
\]
Applying successively (64), (63) and (66) finally yields:
\[
\frac{\epsilon}{2(M + 2\epsilon)} \|r_j\|^2 \leq q(y_j) - q(y_{j+1}) \\
\leq (1 - \tau)^J \left( q(y_0) - q(y_{j+1}) + \frac{\epsilon}{2} \|y_0 - y_{j+1}\|^2 \right) \\
\leq (1 - \tau)^J \|r_0\|\|r_j\| \frac{(1 - \tau)^{-J/2}}{\epsilon} \frac{1}{1 - \sqrt{1 - \tau}}.
\]
After rearranging this inequality and dividing by $\|r_j\| > 0$, we obtain
\[
\|r_j\| \leq \frac{2(M + 2\epsilon)}{\epsilon} \frac{(1 - \tau)^{J/2}}{1 - \sqrt{1 - \tau}} \|r_0\| = \sqrt{T}(1 - \tau)^{J/2}\|r_0\|.
\]
We have thus established (48) which, as we noted earlier, contradicts (10). Thus (47) cannot be true, so we have established (11), as required.
