SIAM J. OPTIM.
Vol. 33, No. 3, pp. 1734–1766

A NEWTON-CG BASED AUGMENTED LAGRANGIAN METHOD FOR FINDING A SECOND-ORDER STATIONARY POINT OF NONCONVEX EQUALITY CONSTRAINED OPTIMIZATION WITH COMPLEXITY GUARANTEES[*]

CHUAN HE[†], ZHAOSONG LU[†], AND TING KEI PONG[‡]

Abstract. In this paper we consider finding a second-order stationary point (SOSP) of nonconvex equality constrained optimization when a nearly feasible point is known. In particular, we first propose a new Newton-conjugate gradient (Newton-CG) method for finding an approximate SOSP of unconstrained optimization and show that it enjoys a substantially better complexity than the Newton-CG method in [C. W. Royer, M. O'Neill, and S. J. Wright, *Math. Program.*, 180 (2020), pp. 451–488]. We then propose a Newton-CG based augmented Lagrangian (AL) method for finding an approximate SOSP of nonconvex equality constrained optimization, in which the proposed Newton-CG method is used as a subproblem solver. We show that under a generalized linear independence constraint qualification (GLICQ), our AL method enjoys a total inner iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-7/2})$ and an operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-7/2}\min\{n,\epsilon^{-3/4}\})$ for finding an $(\epsilon,\sqrt{\epsilon})$-SOSP of nonconvex equality constrained optimization with high probability, which are significantly better than the ones achieved by the proximal AL method in [Y. Xie and S. J. Wright, *J. Sci. Comput.*, 86 (2021), pp. 1–30]. In addition, we show that it has a total inner iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2})$ and an operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2}\min\{n,\epsilon^{-5/4}\})$ when the GLICQ does not hold. To the best of our knowledge, all the complexity results obtained in this paper are new for finding an approximate SOSP of nonconvex equality constrained optimization with high probability. Preliminary numerical results also demonstrate the superiority of our proposed methods over the other competing algorithms.

Key words. nonconvex equality constrained optimization, second-order stationary point, augmented Lagrangian method, Newton conjugate gradient method, iteration complexity, operation complexity

MSC codes. 49M15, 68Q25, 90C06, 90C26, 90C30, 90C60

DOI. 10.1137/22M1489824

1. Introduction.

In this paper we consider the nonconvex equality constrained optimization problem

$$\min_{x\in\mathbb{R}^n}\ f(x)\quad \text{s.t. } c(x)=0, \tag{1.1}$$

where $f:\mathbb{R}^n\to\mathbb{R}$ and $c:\mathbb{R}^n\to\mathbb{R}^m$ are twice continuously differentiable, and we assume that problem (1.1) has at least one optimal solution. Since (1.1) is a nonconvex optimization problem, it may have many local but nonglobal minimizers, and finding its global minimizer is generally NP-hard. A first-order stationary point (FOSP) of it is usually found in practice instead. Nevertheless, a mere FOSP may sometimes not suit our needs and a *second-order stationary point* (SOSP) needs to be sought.

[*]Received by the editors April 11, 2022; accepted for publication (in revised form) January 8, 2023; published electronically August 2, 2023.
https://doi.org/10.1137/22M1489824

Funding: The work of the second author was partially supported by NSF award IIS-2211491. The work of the third author was partially supported by a Research Scheme of the Research Grants Council of Hong Kong SAR, China (project T22-504/21R).

[†]Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, MN 55455 USA (he000233@umn.edu, zhaosong@umn.edu).

[‡]Department of Applied Mathematics, The Hong Kong Polytechnic University, Hong Kong, People's Republic of China (tk.pong@polyu.edu.hk).

For example, in the context of linear semidefinite programming (SDP), a powerful approach to solving it is by solving an equivalent nonconvex equality constrained optimization problem [17, 18]. It was shown in [18, 15] that under some mild conditions an SOSP of the latter problem can yield an optimal solution of the linear SDP, while a mere FOSP generally cannot. It is therefore important to find an SOSP of problem (1.1).

In recent years, numerous methods with complexity guarantees have been developed for finding an approximate SOSP of several types of nonconvex optimization. For example, cubic regularized Newton methods [52, 25, 1, 22], accelerated gradient methods [23, 24], trust-region methods [34, 35, 50], the quadratic regularization method [12], the second-order line-search method [57], and the Newton-conjugate gradient (Newton-CG) method [56] were developed for nonconvex unconstrained optimization. In addition, the interior-point method [8] and the log-barrier method [54] were proposed for nonconvex optimization with sign constraints. The interior-point method [8] was also generalized in [38] to solve nonconvex optimization with sign constraints and additional linear equality constraints. Furthermore, a projected gradient descent method with random perturbations was proposed in [47] for nonconvex optimization with linear inequality constraints. Iteration complexity was established for these methods for finding an approximate SOSP. In addition, operation complexity measured by the number of fundamental operations such as gradient evaluations and matrix-vector products was also studied in [1, 23, 34, 41, 24, 57, 22, 56].

Several methods, including trust-region methods [21, 33], the sequential quadratic programming method [14], the two-phase method [9, 30, 32], and augmented Lagrangian (AL) type methods [4, 10, 58, 60], were proposed for finding an SOSP of problem (1.1). However, only a few of them have *complexity guarantees* for finding an approximate SOSP of (1.1). In particular, the inexact AL method [58] has a worst-case complexity in terms of the number of calls to a second-order oracle. Yet its operation complexity, measured by the number of fundamental operations such as gradient evaluations and Hessian-vector products, is unknown. To the best of our knowledge, the proximal AL method in [60] appears to be the only existing method that enjoys a worst-case complexity for finding an approximate SOSP of (1.1) in terms of fundamental operations. In this method, given an iterate x^k and a multiplier estimate λ^k at the kth iteration, the next iterate x^{k+1} is obtained by finding an approximate stochastic SOSP of the proximal AL subproblem:

$$\min_{x\in\mathbb{R}^n} \mathcal{L}(x,\lambda^k;\rho)+\beta\|x-x^k\|^2/2$$

for some suitable positive ρ and β using a Newton-CG method proposed in [56], where $\mathcal{L}$ is the AL function of (1.1) defined as

$$\mathcal{L}(x,\lambda;\rho):=f(x)+\lambda^T c(x)+\rho\|c(x)\|^2/2.$$

Then the multiplier estimate is updated using the classical scheme, i.e., $\lambda^{k+1}=\lambda^k+\rho c(x^{k+1})$ (e.g., see [39, 55]). The authors of [60] studied the worst-case complexity of their proximal AL method including (i) *total inner iteration complexity*, which measures the total number of iterations of the Newton-CG method [56] performed in their method; (ii) *operation complexity*, which measures the total number of gradient evaluations and matrix-vector products involving the Hessian of the AL function that are evaluated in their method. Under some suitable assumptions, including that a generalized linear independence constraint qualification (GLICQ) holds at all iterates, it was established in [60] that their proximal AL method enjoys a total inner iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2})$ and an operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2}\min\{n,\epsilon^{-3/4}\})$ for finding an $(\epsilon,\sqrt{\epsilon})$-SOSP of problem (1.1) with high probability.[1] Yet, there is a big gap between these complexities and the iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-3/2})$ and the operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-3/2}\min\{n,\epsilon^{-1/4}\})$ that are achieved by the methods in [1, 24, 57, 56] for finding an $(\epsilon,\sqrt{\epsilon})$-SOSP of nonconvex unconstrained optimization with high probability, which is a special case of (1.1) with $c\equiv 0$. Also, there is a lack of complexity guarantees for this proximal AL method when the GLICQ does not hold. It shall be mentioned that Newton-CG based AL methods were also developed for efficiently solving various convex optimization problems (e.g., see [61, 62]), though their complexities remain unknown.

In this paper we propose a Newton-CG based AL method for finding an approximate SOSP of problem (1.1) with high probability, and study its worst-case complexity with and without the assumption of a GLICQ. In particular, we show that this method enjoys a total inner iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-7/2})$ and an operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-7/2}\min\{n,\epsilon^{-3/4}\})$ for finding a stochastic $(\epsilon,\sqrt{\epsilon})$-SOSP of (1.1) under the GLICQ, which are significantly better than the aforementioned ones achieved by the proximal AL method in [60]. Besides, when the GLICQ does not hold, we show that it has a total inner iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2})$ and an operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2}\min\{n,\epsilon^{-5/4}\})$ for finding a stochastic $(\epsilon,\sqrt{\epsilon})$-SOSP of (1.1), which fills the research gap in this topic. Specifically, our AL method (Algorithm 4.1) proceeds in the following manner. Instead of directly solving problem (1.1), it solves a perturbed problem of (1.1) with c replaced by its perturbed counterpart $\tilde{c}$ constructed by using a nearly feasible point of (1.1) (see (4.4) for details). At the kth iteration, an approximate stochastic SOSP x^{k+1} of the AL subproblem of this perturbed problem is found by our newly proposed Newton-CG method (Algorithm 3.1) for a penalty parameter ρ_k and a truncated Lagrangian multiplier λ^k, which results from projecting onto a Euclidean ball the standard multiplier estimate $\tilde{\lambda}^k$ obtained by the classical scheme $\tilde{\lambda}^k=\lambda^{k-1}+\rho_k\tilde{c}(x^k)$.[2] The penalty parameter ρ_{k+1} is then updated by the following practical scheme (e.g., see [7, section 4.2]):

$$\rho_{k+1}=\begin{cases} r\rho_k & \text{if } \|\tilde{c}(x^{k+1})\|>\alpha\|\tilde{c}(x^k)\|,\\ \rho_k & \text{otherwise}\end{cases}$$

for some $r>1$ and $\alpha\in(0,1)$. It shall be mentioned that in contrast with the classical AL method, our method has two distinct features: (i) the values of the AL function along the iterates are bounded from above; (ii) the multiplier estimates associated with the AL subproblems are bounded. In addition, to solve the AL subproblems with better complexity guarantees, we propose a variant of the Newton-CG method in [56] for finding an approximate stochastic SOSP of unconstrained optimization, whose complexity has significantly less dependence on the Lipschitz constant of the Hessian of the objective than that of the Newton-CG method in [56], while improving or retaining the same order of dependence on tolerance parameter. Given that such

[1]In fact, a total inner iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-7})$ and an operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-7}\min\{n,\epsilon^{-1}\})$ were established in [60] for finding an (ϵ,ϵ)-SOSP of problem (1.1) with high probability; see [60, Theorem 4(ii), Corollary 3(ii), and Theorem 5]. Nonetheless, they can be modified to obtain the aforementioned complexity for finding an $(\epsilon,\sqrt{\epsilon})$-SOSP of (1.1) with high probability.

[2]The λ^k obtained by projecting $\tilde{\lambda}^k$ onto a compact set is also called a safeguarded Lagrangian multiplier in the relevant literature [11, 42, 13], which has been shown to enjoy many practical and theoretical advantages (see [11] for discussions).

a Lipschitz constant is typically large for the AL subproblems, from a theoretical complexity perspective, our Newton-CG method (Algorithm 3.1) is a much more favorable subproblem solver than the Newton-CG method in [56] that is used in the proximal AL method in [60].

The main contributions of this paper are summarized as follows:

- We propose a new Newton-CG method for finding an approximate SOSP of unconstrained optimization and show that it enjoys an iteration and operation complexity with a *quadratic* dependence on the Lipschitz constant of the Hessian of the objective that improves the *cubic* dependence achieved by the Newton-CG method in [56], while improving or retaining the same order of dependence on tolerance parameter. In addition, our complexity results are established under the assumption that the Hessian of the objective is Lipschitz continuous in a convex neighborhood of a level set of the objective. This assumption is weaker than the one commonly imposed for the Newton-CG method in [56] and some other methods (e.g., [12, 35]) that the Hessian of the objective is Lipschitz continuous in a convex set containing this neighborhood and also *all the trial points* arising in the line- search or trust-region steps of the methods (see section 3 for more detailed discussion).
- We propose a Newton-CG based AL method for finding an approximate SOSP of nonconvex equality constrained optimization (1.1) with high probability, and study its worst-case complexity with and without the assumption of a GLICQ. Prior to our work, there was no complexity study on finding an approximate SOSP of problem (1.1) without imposing a GLICQ. Besides, under the GLICQ and some other suitable assumptions, we show that our method enjoys a total inner iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-7/2})$ and an operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-7/2}\min\{n,\epsilon^{-3/4}\})$ for finding an $(\epsilon,\sqrt{\epsilon})$-SOSP of (1.1) with high probability, which are significantly better than the respective complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2})$ and $\widetilde{\mathcal{O}}(\epsilon^{-11/2}\min\{n,\epsilon^{-3/4}\})$ achieved by the proximal AL method in [60]. To the best of our knowledge, all the complexity results obtained in this paper are new for finding an approximate SOSP of nonconvex equality constrained optimization with high probability.

For ease of comparison, we summarize in Table 1 the total inner iteration and operation complexity of our AL method and the proximal AL method in [60] for finding a stochastic $(\epsilon,\sqrt{\epsilon})$-SOSP of problem (1.1) with or without assuming GLICQ.

It should be mentioned that there are many works other than [60] studying complexity of AL methods for nonconvex constrained optimization. However, they aim to find an approximate FOSP rather than SOSP of the problem (e.g., see [40, 37, 13, 51, 45]). Since our main focus is on the complexity of finding an approximate SOSP by AL methods, we do not include them in Table 1 for comparison.

The rest of this paper is organized as follows. In section 2, we introduce some notation and optimality conditions. In section 3, we propose a Newton-CG method

TABLE 1
Total inner iteration and operation complexity of finding a stochastic $(\epsilon,\sqrt{\epsilon})$*-SOSP of* (1.1).

Method	GLICQ	Total inner iteration complexity	Operation complexity
Proximal AL method [60]	✓	$\widetilde{\mathcal{O}}(\epsilon^{-11/2})$	$\widetilde{\mathcal{O}}(\epsilon^{-11/2}\min\{n,\epsilon^{-3/4}\})$
Proximal AL method [60]	✗	unknown	unknown
Our AL method	✓	$\widetilde{\mathcal{O}}(\epsilon^{-7/2})$	$\widetilde{\mathcal{O}}(\epsilon^{-7/2}\min\{n,\epsilon^{-3/4}\})$
Our AL method	✗	$\widetilde{\mathcal{O}}(\epsilon^{-11/2})$	$\widetilde{\mathcal{O}}(\epsilon^{-11/2}\min\{n,\epsilon^{-5/4}\})$

for unconstrained optimization and study its worst-case complexity. In section 4, we propose a Newton-CG based AL method for (1.1) and study its worst-case complexity. We present numerical results and the proof of the main results in sections 5 and 6, respectively. In section 7, we discuss some future research directions.

2. Notation and preliminaries.

Throughout this paper, we let $\mathbb{R}^n$ denote the n-dimensional Euclidean space. We use $\|\cdot\|$ to denote the Euclidean norm of a vector or the spectral norm of a matrix. For a real symmetric matrix H, we use $\lambda_{\min}(H)$ to denote its minimum eigenvalue. The Euclidean ball centered at the origin with radius $R \geq 0$ is denoted by $\mathcal{B}_R := \{x : \|x\| \leq R\}$, and we use $\Pi_{\mathcal{B}_R}(v)$ to denote the Euclidean projection of a vector v onto $\mathcal{B}_R$. For a given finite set $\mathcal{A}$, we let $|\mathcal{A}|$ denote its cardinality. For any $s \in \mathbb{R}$, we let $\mathrm{sgn}(s)$ be 1 if $s \geq 0$ and let it be -1 otherwise. In addition, $\widetilde{\mathcal{O}}(\cdot)$ represents $\mathcal{O}(\cdot)$ with logarithmic terms omitted.

Suppose that x^* is a local minimizer of problem (1.1) and the linear independence constraint qualification holds at x^*, i.e., $\nabla c(x^*) := [\nabla c_1(x^*)\ \nabla c_2(x^*)\ \cdots\ \nabla c_m(x^*)]$ has full column rank. Then there exists a Lagrangian multiplier $\lambda^* \in \mathbb{R}^m$ such that

$$\nabla f(x^*) + \nabla c(x^*)\lambda^* = 0, \tag{2.1}$$

$$d^T\left(\nabla^2 f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla^2 c_i(x^*)\right) d \geq 0 \quad \forall d \in \mathcal{C}(x^*), \tag{2.2}$$

where $\mathcal{C}(\cdot)$ is defined as

$$\mathcal{C}(x) := \{d \in \mathbb{R}^n : \nabla c(x)^T d = 0\}. \tag{2.3}$$

The relations (2.1) and (2.2) are respectively known as the first- and second-order optimality conditions for (1.1) in the literature (e.g., see [53]). Note that it is in general impossible to find a point that exactly satisfies (2.1) and (2.2). Thus, we are instead interested in finding a point that satisfies their approximate counterparts. In particular, we introduce the following definitions of an approximate first-order stationary point (FOSP) and second-order stationary point (SOSP), which are similar to those considered in [4, 10, 60]. The rationality of them can be justified by the study of the sequential optimality conditions for constrained optimization [3, 4].

DEFINITION 2.1 (ϵ_1-first-order stationary point). *Let $\epsilon_1 > 0$. We say that $x \in \mathbb{R}^n$ is an ϵ_1-first-order stationary point* (ϵ_1-FOSP) *of problem* (1.1) *if it, together with some $\lambda \in \mathbb{R}^m$, satisfies*

$$\|\nabla f(x) + \nabla c(x)\lambda\| \leq \epsilon_1, \quad \|c(x)\| \leq \epsilon_1. \tag{2.4}$$

DEFINITION 2.2 ((ϵ_1,ϵ_2)-second-order stationary point). *Let $\epsilon_1, \epsilon_2 > 0$. We say that $x \in \mathbb{R}^n$ is an (ϵ_1,ϵ_2)-second-order stationary point* ((ϵ_1,ϵ_2)-SOSP) *of problem* (1.1) *if it, together with some $\lambda \in \mathbb{R}^m$, satisfies* (2.4) *and additionally*

$$d^T\left(\nabla^2 f(x) + \sum_{i=1}^m \lambda_i \nabla^2 c_i(x)\right) d \geq -\epsilon_2\|d\|^2 \quad \forall d \in \mathcal{C}(x), \tag{2.5}$$

where $\mathcal{C}(\cdot)$ is defined as in (2.3).

3. A Newton-CG method for unconstrained optimization.

In this section we propose a variant of the Newton-CG method [56, Algorithm 3] for finding an approximate SOSP of a class of unconstrained optimization problems, which will be used as a subproblem solver for the AL method proposed in the next section. In particular, we consider an unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} \; F(x), \tag{3.1}$$

where the function F satisfies the following assumptions.

Assumption 3.1.
(a) The level set $\mathscr{L}_F(u^0) := \{x : F(x) \le F(u^0)\}$ is compact for some $u^0 \in \mathbb{R}^n$.
(b) The function F is twice Lipschitz continuously differentiable in a convex open neighborhood, denoted by Ω, of $\mathscr{L}_F(u^0)$, that is, there exists $L_H^F > 0$ such that

$$\|\nabla^2 F(x) - \nabla^2 F(y)\| \le L_H^F \|x-y\| \quad \forall x, y \in \Omega. \tag{3.2}$$

By Assumption 3.1, there exist $F_{\text{low}} \in \mathbb{R}$, $U_g^F > 0$, and $U_H^F > 0$ such that

$$F(x) \ge F_{\text{low}}, \quad \|\nabla F(x)\| \le U_g^F, \quad \|\nabla^2 F(x)\| \le U_H^F \quad \forall x \in \mathscr{L}_F(u^0). \tag{3.3}$$

Recently, a Newton-CG method [56, Algorithm 3] was developed to find an approximate stochastic SOSP of problem (3.1), which is not only easy to implement but also enjoys a nice feature that the main computation consists only of gradient evaluations and Hessian-vector products associated with the function F. Under the assumption that $\nabla^2 F$ is Lipschitz continuous in a convex open set containing $\mathscr{L}_F(u^0)$ and also *all the trial points* arising in the line-search steps of this method (see [56, Assumption 2]), it was established in [56, Theorem 4 and Corollary 2] that the iteration and operation complexity of this method for finding a stochastic (ϵ_g, ϵ_H)-SOSP of (3.1) (namely, a point x satisfying $\|\nabla F(x)\| \le \epsilon_g$ deterministically and $\lambda_{\min}(\nabla^2 F(x)) \ge -\epsilon_H$ with high probability) are

$$\mathcal{O}((L_H^F)^3 \max\{\epsilon_g^{-3}\epsilon_H^3, \epsilon_H^{-3}\}) \text{ and } \widetilde{\mathcal{O}}((L_H^F)^3 \max\{\epsilon_g^{-3}\epsilon_H^3, \epsilon_H^{-3}\} \min\{n, (U_H^F/\epsilon_H)^{1/2}\}), \tag{3.4}$$

respectively, where $\epsilon_g, \epsilon_H \in (0,1)$ are prescribed tolerances. *Yet, this assumption can be hard to check* because these trial points are *unknown* before the method terminates and, moreover, the distance between the origin and these points depends on the tolerance ϵ_H in $\mathcal{O}(\epsilon_H^{-1})$ (see [56, Lemma 3]). In addition, as seen from (3.4), iteration and operation complexity of the Newton-CG method in [56] depend *cubically* on L_H^F. Notice that L_H^F can sometimes be very large. For example, the AL subproblems arising in Algorithm 4.1 have $L_H^F = \mathcal{O}(\epsilon_1^{-2})$ or $\mathcal{O}(\epsilon_1^{-1})$, where $\epsilon_1 \in (0,1)$ is a prescribed tolerance for problem (1.1) (see section 4). The cubic dependence on L_H^F makes such a Newton-CG method not appealing as an AL subproblem solver from a theoretical complexity perspective.

In the rest of this section, we propose a variant of the Newton-CG method [56, Algorithm 3] and show that under Assumption 3.1, it enjoys an iteration and an operation complexity of

$$\mathcal{O}((L_H^F)^2 \max\{\epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\}) \text{ and } \widetilde{\mathcal{O}}((L_H^F)^2 \max\{\epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\} \min\{n, (U_H^F/\epsilon_H)^{1/2}\}), \tag{3.5}$$

respectively, for finding a stochastic (ϵ_g, ϵ_H)-SOSP of problem (3.1). These complexities are substantially superior to those in (3.4) achieved by the Newton-CG method in [56]. Indeed, the complexities in (3.5) depend quadratically on L_H^F, while those in (3.4) depend cubically on L_H^F. In addition, it can be verified that they improve or retain the order of dependence on ϵ_g and ϵ_H given in (3.4).

3.1. Main components of a Newton-CG method. In this subsection we briefly discuss two main components of the Newton-CG method in [56], which will be used to propose a variant of this method for finding an approximate stochastic SOSP of problem (3.1) in the next subsection.

The first main component of the Newton-CG method in [56] is a *capped CG method* [56, Algorithm 1], which is a modified CG method, for solving a possibly indefinite linear system

$$(H+2\varepsilon I)d=-g, \tag{3.6}$$

where $0\neq g\in\mathbb{R}^n$, $\varepsilon>0$, and $H\in\mathbb{R}^{n\times n}$ is a symmetric matrix. This capped CG method terminates within a finite number of iterations. It outputs either an approximate solution d to (3.6) such that $\|(H+2\varepsilon I)d+g\|\leq\widehat{\zeta}\|g\|$ and $d^THd\geq-\varepsilon\|d\|^2$ for some $\widehat{\zeta}\in(0,1)$ or a sufficiently negative curvature direction d of H with $d^THd<-\varepsilon\|d\|^2$. The second main component of the Newton-CG method in [56] is a minimum eigenvalue oracle that either produces a sufficiently negative curvature direction v of H with $\|v\|=1$ and $v^THv\leq-\varepsilon/2$ or certifies that $\lambda_{\min}(H)\geq-\varepsilon$ holds with high probability. For ease of reference, we present these two components in Algorithms A.1 and B.1 in Appendices A and B, respectively.

3.2. A Newton-CG method for problem (3.1). In this subsection we propose a Newton-CG method in Algorithm 3.1, which is a variant of the Newton-CG method [56, Algorithm 3], for finding an approximate stochastic SOSP of problem (3.1).

Our Newton-CG method (Algorithm 3.1) follows the same framework as [56, Algorithm 3]. In particular, at each iteration, if the gradient of F at the current iterate is not desirably small, then the capped CG method (Algorithm A.1) is called to solve a damped Newton system for obtaining a descent direction and a subsequent line search along this direction results in a sufficient reduction on F. Otherwise, the current iterate is already an approximate FOSP of (3.1), and the minimum eigenvalue oracle (Algorithm B.1) is then called, which either produces a sufficiently negative curvature direction for F and a subsequent line search along this direction results in a sufficient reduction on F, or certifies that the current iterate is an approximate SOSP of (3.1) with high probability and terminates the algorithm. More details about this framework can be found in [56].

Despite sharing the same framework, our Newton-CG method and [56, Algorithm 3] use different line-search criteria. Indeed, our Newton-CG method uses a hybrid line-search criterion adopted from [59], which is a combination of the quadratic descent criterion (3.10) and the cubic descent criterion (3.11). Specifically, it uses the quadratic descent criterion (3.10) when the search direction is of type "SOL." On the other hand, it uses the cubic descent criterion (3.11) when the search direction is of type "NC."[3] In contrast, the Newton-CG method in [56] always uses a cubic descent criterion regardless of the type of search directions. As observed from Theorem 3.2, our Newton-CG method achieves an iteration and an operation complexity given in (3.5), which are superior to those in (3.4) achieved by [56, Algorithm 3] in terms of the order dependence on L_H^F, while improving or retaining the order of dependence on ϵ_g and ϵ_H as given in (3.4). Consequently, from a theoretical complexity perspective, our Newton-CG method is more appealing than [56, Algorithm 3] as an AL subproblem solver for the AL method proposed in section 4.

The following theorem states the iteration and operation complexity of Algorithm 3.1; the proof is deferred to section 6.1.

[3]SOL and NC stand for "approximate solution" and "negative curvature," respectively.

Algorithm 3.1 A Newton-CG method for problem (3.1).

```
Input: Tolerances $\epsilon_g, \epsilon_H \in (0,1)$, backtracking ratio $\theta \in (0,1)$, starting point $u^0$, CG-accuracy
parameter $\zeta \in (0,1)$, line-search parameter $\eta \in (0,1)$, probability parameter $\delta \in (0,1)$.
Set $x^0 = u^0$;
for $t = 0,1,2,\ldots$ do
  if $\|\nabla F(x^t)\| > \epsilon_g$ then
    Call Algorithm A.1 with $H = \nabla^2 F(x^t)$, $\varepsilon = \epsilon_H$, $g = \nabla F(x^t)$, accuracy parameter $\zeta$, and
    $U = 0$ to obtain outputs $d$, d_type;
    if d_type=NC then
(3.7)          $d^t \leftarrow \operatorname{sgn}(d^T \nabla F(x^t)) \frac{|d^T \nabla^2 F(x^t) d|}{\|d\|^3} d;$
    else {d_type=SOL}
(3.8)          $d^t \leftarrow d;$
    end if
    Go to Line Search;
  else
    Call Algorithm B.1 with $H = \nabla^2 F(x^t)$, $\varepsilon = \epsilon_H$, and probability parameter $\delta$;
    if Algorithm B.1 certifies that $\lambda_{\min}(\nabla^2 F(x^t)) \geq \quad \epsilon_H$ then
      Output $x^t$ and terminate;
    else {Sufficiently negative curvature direction $v$ returned by Algorithm B.1}
      Set d_type=NC and
(3.9)          $d^t \leftarrow \operatorname{sgn}(v^T \nabla F(x^t)) |v^T \nabla^2 F(x^t) v| v;$
      Go to Line Search;
    end if
  end if
  Line Search:
  if d_type=SOL then
    Find $\alpha_t = \theta^{j_t}$, where $j_t$ is the smallest nonnegative integer $j$ such that
(3.10)          $F(x^t + \theta^j d^t) < F(x^t) \quad \eta \epsilon_H \theta^{2j} \|d^t\|^2;$
  else {d_type=NC}
    Find $\alpha_t = \theta^{j_t}$, where $j_t$ is the smallest nonnegative integer $j$ such that
(3.11)          $F(x^t + \theta^j d^t) < F(x^t) \quad \eta \theta^{2j} \|d^t\|^3/2;$
  end if
  $x^{t+1} = x^t + \alpha_t d^t$;
end for
```

THEOREM 3.2. *Suppose that Assumption* 3.1 *holds. Let*

(3.12)
$$T_1 := \left\lceil \frac{F_{\text{hi}} - F_{\text{low}}}{\min\{c_{\text{sol}}, c_{\text{nc}}\}} \max\{\epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\} \right\rceil + \left\lceil \frac{F_{\text{hi}} - F_{\text{low}}}{c_{\text{nc}}} \epsilon_H^{-3} \right\rceil + 1,\ T_2 := \left\lceil \frac{F_{\text{hi}} - F_{\text{low}}}{c_{\text{nc}}} \epsilon_H^{-3} \right\rceil + 1,$$

where $F_{\text{hi}} = F(u^0)$, F_{low} *is given in* (3.3), *and*

$$c_{sol} := \eta \min\left\{ \left[\frac{4}{4 + \zeta + \sqrt{(4+\zeta)^2 + 8L_H^F}} \right]^2, \left[\frac{\min\{6(1-\eta), 2\}\theta}{L_H^F} \right]^2 \right\}, \tag{3.13}$$

$$c_{nc} := \frac{\eta}{16} \min\left\{ 1, \left[\frac{\min\{3(1-\eta), 1\}\theta}{L_H^F} \right]^2 \right\}. \tag{3.14}$$

Then the following statements hold:

(i) *The total number of calls of Algorithm* B.1 *in Algorithm* 3.1 *is at most* T_2.

(ii) *The total number of calls of Algorithm* A.1 *in Algorithm* 3.1 *is at most* T_1.

(iii) (iteration complexity) *Algorithm* 3.1 *terminates in at most* $T_1 + T_2$ *iterations with*

$$T_1 + T_2 = \mathcal{O}((F_{\text{hi}} - F_{\text{low}})(L_H^F)^2 \max\{\epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\}). \tag{3.15}$$

Also, its output x^t *satisfies* $\|\nabla F(x^t)\| \le \epsilon_g$ *deterministically and* $\lambda_{\min}(\nabla^2 F(x^t)) \ge -\epsilon_H$ *with probability at least* $1-\delta$ *for some* $0 \le t \le T_1 + T_2$.

(iv) (operation complexity) *Algorithm* 3.1 *requires at most*

$$\widetilde{\mathcal{O}}((F_{\text{hi}} - F_{\text{low}})(L_H^F)^2 \max\{\epsilon_g^{-2}\epsilon_H, \epsilon_H^{-3}\} \min\{n, (U_H^F/\epsilon_H)^{1/2}\})$$

matrix-vector products, where U_H^F *is given in* (3.3).

4. A Newton-CG based AL method for problem (1.1). In this section we propose a Newton-CG based AL method for finding a stochastic (ϵ_1, ϵ_2)-SOSP of problem (1.1) for any prescribed tolerances $\epsilon_1, \epsilon_2 \in (0,1)$. Before proceeding, we make some additional assumptions on problem (1.1).

Assumption 4.1.

(a) An $\epsilon_1/2$-approximately feasible point z_{ϵ_1} of problem (1.1), namely satisfying $\|c(z_{\epsilon_1})\| \le \epsilon_1/2$, is known.

(b) There exist constants f_{hi}, f_{low} and $\gamma > 0$, independent of ϵ_1 and ϵ_2, such that

$$f(z_{\epsilon_1}) \le f_{\text{hi}}, \tag{4.1}$$

$$f(x) + \gamma\|c(x)\|^2/2 \ge f_{\text{low}} \quad \forall x \in \mathbb{R}^n, \tag{4.2}$$

where z_{ϵ_1} is given in (a).

(c) There exist some $\delta_f, \delta_c > 0$ such that the set

$$\mathcal{S}(\delta_f, \delta_c) := \{x : f(x) \le f_{\text{hi}} + \delta_f,\ \|c(x)\| \le 1 + \delta_c\} \tag{4.3}$$

is compact with f_{hi} given above. Also, $\nabla^2 f$ and $\nabla^2 c_i$, $i = 1, 2, \ldots, m$, are Lipschitz continuous in a convex open neighborhood, denoted by $\Omega(\delta_f, \delta_c)$, of $\mathcal{S}(\delta_f, \delta_c)$.

We now make some remarks about Assumption 4.1.

Remark 4.2.

(i) An assumption very similar to Assumption 4.1(a) was considered in [31, 37, 49, 60]. By imposing Assumption 4.1(a), we restrict our study on problem (1.1) for which an $\epsilon_1/2$-approximately feasible point z_{ϵ_1} can be found by an inexpensive procedure. One example of such problem instances arises when there exists v^0 such that $\{x : \|c(x)\| \le \|c(v^0)\|\}$ is compact, $\nabla^2 c_i$, $1 \le i \le m$, is Lipschitz continuous on a convex neighborhood of this set, and the LICQ holds on this set. Indeed, for this instance, a point z_{ϵ_1} satisfying $\|c(z_{\epsilon_1})\| \le \epsilon_1/2$ can be computed by applying our Newton-CG method (Algorithm 3.1) to the problem $\min_{x\in\mathbb{R}^n} \|c(x)\|^2$. As seen from Theorem 3.2, the resulting iteration and operation complexity of Algorithm 3.1 for finding such z_{ϵ_1} are, respectively, $\mathcal{O}(\epsilon_1^{-3/2})$ and $\widetilde{\mathcal{O}}(\epsilon_1^{-3/2}\min\{n, \epsilon_1^{-1/4}\})$, which are negligible compared with those of our AL method (see Theorems 4.10 and 4.14 below). As another example, when the standard error bound condition $\|c(x)\|^2 = \mathcal{O}(\|\nabla(\|c(x)\|^2)\|^\nu)$ holds on a level set of $\|c(x)\|$ for some $\nu > 0$, one can find the above z_{ϵ_1} by applying a gradient method to the problem $\min_{x\in\mathbb{R}^n} \|c(x)\|^2$ (e.g., see [46, 58]). In addition, the Newton-CG based AL method (Algorithm 4.1) proposed below is a second-order method with the aim to find an SOSP. It is more expensive than a first-order method in general. To make best use of such an AL method in practice, it is natural to run a first-order method in advance to obtain an $\epsilon_1/2$-FOSP z_{ϵ_1} and then run the AL method using z_{ϵ_1} as an $\epsilon_1/2$-approximately feasible point. Therefore, Assumption 4.1(a) is met in practice, provided that an $\epsilon_1/2$-FOSP of (1.1) can be found by a first-order method.

(ii) Assumption 4.1(b) is mild. In particular, the assumption in (4.1) holds if $f(x) \le f_{\rm hi}$ holds for all x with $\|c(x)\| \le 1$, which is imposed in [60, Assumption 3]. It also holds if problem (1.1) has a known feasible point, which is often imposed for designing AL methods for nonconvex constrained optimization (e.g., see [49, 31, 48, 37]). In addition, the assumption in (4.2) implies that the quadratic penalty function is bounded below when the associated penalty parameter is sufficiently large, which is typically used in the study of quadratic penalty and AL methods for solving problem (1.1) (e.g., see [40, 37, 60, 43]). Clearly, when $\inf_{x\in\mathbb{R}^n} f(x) > -\infty$, one can see that (4.2) holds for any $\gamma > 0$. In general, one possible approach to identifying γ is to apply the techniques on infeasibility detection developed in the literature (e.g., [20, 19, 6]) to check the infeasibility of the level set $\{x : f(x) + \gamma\|c(x)\|^2/2 \le \tilde{f}_{\rm low}\}$ for some sufficiently small $\tilde{f}_{\rm low}$. Note that this level set being infeasible for some $\tilde{f}_{\rm low}$ implies that (4.2) holds for the given γ and $f_{\rm low} = \tilde{f}_{\rm low}$.

(iii) Assumption 4.1(c) is not too restrictive. Indeed, the set $\mathcal{S}(\delta_f, \delta_c)$ is compact if f or $f(\cdot) + \gamma\|c(\cdot)\|^2/2$ is level-bounded. The latter level-boundedness assumption is commonly imposed for studying AL methods (e.g., see [37, 60]), which is stronger than our assumption.

We next propose a Newton-CG based AL method in Algorithm 4.1 for finding a stochastic (ϵ_1, ϵ_2)-SOSP of problem (1.1) under Assumption 4.1. Instead of solving (1.1) directly, this method solves the perturbed problem

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{s.t.} \quad \tilde{c}(x) := c(x) - c(z_{\epsilon_1}) = 0, \tag{4.4}$$

where z_{ϵ_1} is given in Assumption 4.1(a). Specifically, at the kth iteration, this method applies the Newton-CG method (Algorithm 3.1) to find an approximate stochastic SOSP x^{k+1} of the AL subproblem associated with (4.4):

$$\min_{x\in\mathbb{R}^n} \left\{ \widetilde{\mathcal{L}}(x, \lambda^k, \rho_k) := f(x) + (\lambda^k)^T \tilde{c}(x) + \rho_k\|\tilde{c}(x)\|^2/2 \right\} \tag{4.5}$$

such that $\widetilde{\mathcal{L}}(x^{k+1}, \lambda^k; \rho_k)$ is below a threshold (see (4.6) and (4.7)), where λ^k is a truncated Lagrangian multiplier, i.e., the one that results from projecting the standard multiplier estimate $\tilde{\lambda}^k$ onto a Euclidean ball (see step 6 of Algorithm 4.1). The standard multiplier estimate $\tilde{\lambda}^{k+1}$ is then updated by the classical scheme described in step 4 of Algorithm 4.1. Finally, the penalty parameter ρ_{k+1} is adaptively updated based on the improvement on constraint violation (see step 7 of Algorithm 4.1). Such a practical update scheme is often adopted in the literature (e.g., see [7, 2, 31]).

We would like to point out that the truncated Lagrangian multiplier sequence $\{\lambda^k\}$ is used in the AL subproblems of Algorithm 4.1 and is bounded, while the standard Lagrangian multiplier sequence $\{\tilde{\lambda}^k\}$ is used in those of the classical AL methods

Algorithm 4.1 A Newton-CG based AL method for problem (1.1).

```
Let $\gamma$ be given in Assumption 4.1.
Input: $\epsilon_1, \epsilon_2 \in (0,1)$, $\Lambda > 0$, $x^0 \in \mathbb{R}^n$, $\lambda^0 \in \mathcal{B}_\Lambda$, $\rho_0 > 2\gamma$, $\alpha \in (0,1)$, $r > 1$, $\delta \in (0,1)$, and $z_{\epsilon_1}$ given in Assumption 4.1.
1: Set $k = 0$.
2: Set $\tau_k^g = \max\{\epsilon_1, r^{k \log \epsilon_1 / \log 2}\}$ and $\tau_k^H = \max\{\epsilon_2, r^{k \log \epsilon_2 / \log 2}\}$.
3: Call Algorithm 3.1 with $\epsilon_g = \tau_k^g$, $\epsilon_H = \tau_k^H$ and $u^0 = x^k_{\rm init}$ to find an approximate solution $x^{k+1}$ to $\min_{x\in\mathbb{R}^n} \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k)$ such that
   (4.6)  $\widetilde{\mathcal{L}}(x^{k+1}, \lambda^k; \rho_k) \le f(z_{\epsilon_1}), \ \|\nabla_x \widetilde{\mathcal{L}}(x^{k+1}, \lambda^k; \rho_k)\| \le \tau_k^g,$
   (4.7)  $\lambda_{\min}(\nabla^2_{xx} \widetilde{\mathcal{L}}(x^{k+1}, \lambda^k; \rho_k)) \ge -\tau_k^H$ with probability at least $1 - \delta$,
   where
   (4.8)  $x^k_{\rm init} = \begin{cases} z_{\epsilon_1} & \text{if } \widetilde{\mathcal{L}}(x^k, \lambda^k; \rho_k) > f(z_{\epsilon_1}), \\ x^k & \text{otherwise}, \end{cases}$   for $k \ge 0$.
4: Set $\tilde{\lambda}^{k+1} = \lambda^k + \rho_k \tilde{c}(x^{k+1})$.
5: If $\tau_k^g \le \epsilon_1$, $\tau_k^H \le \epsilon_2$ and $\|c(x^{k+1})\| \le \epsilon_1$, then output $(x^{k+1}, \tilde{\lambda}^{k+1})$ and terminate.
6: Set $\lambda^{k+1} = \Pi_{\mathcal{B}_\Lambda}(\tilde{\lambda}^{k+1})$.
7: If $k = 0$ or $\|\tilde{c}(x^{k+1})\| > \alpha \|\tilde{c}(x^k)\|$, set $\rho_{k+1} = r\rho_k$. Otherwise, set $\rho_{k+1} = \rho_k$.
8: Set $k \leftarrow k + 1$, and go to step 2.
```

and can be unbounded. Therefore, Algorithm 4.1 can be viewed as a safeguarded AL method. Truncated Lagrangian multipliers have been used in the literature for designing some AL methods [2, 11, 42, 13], and will play a crucial role in the subsequent complexity analysis of Algorithm 4.1.

Remark 4.3.

(i) Notice that the starting point $x^0_{\rm init}$ of Algorithm 4.1 can be different from z_{ϵ_1} and it may be rather infeasible, though z_{ϵ_1} is a nearly feasible point of (1.1). Besides, z_{ϵ_1} is used to ensure convergence of Algorithm 4.1. Specifically, if the algorithm runs into a "poorly infeasible point" x^k, namely satisfying $\widetilde{\mathcal{L}}(x^k, \lambda^k; \rho_k) > f(z_{\epsilon_1})$, it will be superseded by z_{ϵ_1} (see (4.8)), which prevents the iterates $\{x^k\}$ from converging to an infeasible point. However, x^k may be rather infeasible when k is not large. Thus, Algorithm 4.1 substantially differs from a funneling or two-phase type algorithm, in which a nearly feasible point is found in Phase 1, and then approximate stationarity is sought, while near feasibility is maintained throughout Phase 2 (e.g., see [9, 16, 26, 27, 28, 29, 30, 36]).

(ii) The choice of ρ_0 in Algorithm 4.1 is mainly for the simplicity of complexity analysis. However, it may be overly large and lead to highly ill-conditioned AL subproblems in practice. To make Algorithm 4.1 practically more efficient, one can possibly modify it by choosing a relatively small initial penalty parameter, then solving the subsequent AL subproblems by a first-order method until an ϵ_1-FOSP $\hat{x}$ of (1.1) along with a Lagrangian multiplier $\hat{\lambda}$ is found, and finally performing the steps described in Algorithm 4.1 but with $x^0 = \hat{x}$ and $\lambda^0 = \Pi_{\mathcal{B}_\Lambda}(\hat{\lambda})$.

Before analyzing the complexity of Algorithm 4.1, we first argue that it is well-defined if ρ_0 is suitably chosen. Specifically, we will show that when ρ_0 is sufficiently large, one can apply the Newton-CG method (Algorithm 3.1) to the AL subproblem $\min_{x\in\mathbb{R}^n} \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k)$ with x^k_{init} as the initial point to find an x^{k+1} satisfying (4.6) and (4.7). To this end, we start by noting from (4.1), (4.4), (4.5), and (4.8) that

$$\widetilde{\mathcal{L}}(x^k_{\mathrm{init}}, \lambda^k; \rho_k) \le \max\{\widetilde{\mathcal{L}}(z_{\epsilon_1}, \lambda^k; \rho_k), f(z_{\epsilon_1})\} = f(z_{\epsilon_1}) \le f_{\mathrm{hi}}. \tag{4.9}$$

Based on the above observation, we show in the next lemma that when ρ_0 is sufficiently large, $\widetilde{\mathcal{L}}(\cdot, \lambda^k; \rho_k)$ is bounded below and its certain level set is bounded; the proof is deferred to section 6.2.

Lemma 4.4. *Suppose that Assumption* 4.1 *holds. Let* (λ^k, ρ_k) *be generated at the kth iteration of Algorithm* 4.1 *for some* $k \ge 0$, *let* $\mathcal{S}(\delta_f, \delta_c)$ *and* x^k_{init} *be defined in* (4.3) *and* (4.8), *respectively, and let* f_{hi}, f_{low}, δ_f, *and* δ_c *be given in Assumption* 4.1. *Suppose that* ρ_0 *is sufficiently large such that* $\delta_{f,1} \le \delta_f$ *and* $\delta_{c,1} \le \delta_c$, *where*

$$\delta_{f,1} := \Lambda^2/(2\rho_0) \quad \text{and} \quad \delta_{c,1} := \sqrt{\frac{2(f_{\mathrm{hi}} - f_{\mathrm{low}} + \gamma)}{\rho_0 - 2\gamma} + \frac{\Lambda^2}{(\rho_0 - 2\gamma)^2}} + \frac{\Lambda}{\rho_0 - 2\gamma}. \tag{4.10}$$

Then the following statements hold.

(i) $\{x : \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k) \le \widetilde{\mathcal{L}}(x^k_{\mathrm{init}}, \lambda^k; \rho_k)\} \subseteq \mathcal{S}(\delta_f, \delta_c)$.

(ii) $\inf_{x\in\mathbb{R}^n} \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k) \ge f_{\mathrm{low}} - \gamma - \Lambda\delta_c$.

Using Lemma 4.4, we can verify that the Newton-CG method (Algorithm 3.1), starting with $u^0 = x^k_{\mathrm{init}}$, is capable of finding an approximate solution x^{k+1} of the AL subproblem $\min_{x\in\mathbb{R}^n} \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k)$ satisfying (4.6) and (4.7). Indeed, let $F(\cdot) = \widetilde{\mathcal{L}}(\cdot, \lambda^k; \rho_k)$ and $u^0 = x^k_{\mathrm{init}}$. By these and Lemma 4.4, one can see that $\{x : F(x) \le F(u^0)\} \subseteq \mathcal{S}(\delta_f, \delta_c)$. It then follows from this and Assumption 4.1(c) that the level set $\{x : F(x) \le F(u^0)\}$ is compact and $\nabla^2 F$ is Lipschitz continuous on a convex open neighborhood of $\{x : F(x) \le F(u^0)\}$. Thus, such F and u^0 satisfy Assumption 3.1. Based on this and the discussion in section 3, one can conclude that Algorithm 3.1, starting with $u^0 = x^k_{\mathrm{init}}$, is applicable to the AL subproblem $\min_{x\in\mathbb{R}^n} \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k)$. Moreover, it follows from Theorem 3.2 that this Algorithm 3.1 with $(\epsilon_g, \epsilon_H) = (\tau^g_k, \tau^H_k)$ can produce a point x^{k+1} satisfying (4.7) and also the second relation in (4.6). In addition, since this algorithm is descent and its starting point is x^k_{init}, its output x^{k+1} must satisfy $\widetilde{\mathcal{L}}(x^{k+1}, \lambda^k; \rho_k) \le \widetilde{\mathcal{L}}(x^k_{\mathrm{init}}, \lambda^k; \rho_k)$, which along with (4.9) implies that $\widetilde{\mathcal{L}}(x^{k+1}, \lambda^k; \rho_k) \le f(z_{\epsilon_1})$ and thus x^{k+1} also satisfies the first relation in (4.6).

This discussion leads to the following conclusion concerning the *well-definedness of Algorithm* 4.1.

Theorem 4.5. *Under the same settings as in Lemma* 4.4, *the Newton-CG method (Algorithm* 3.1*) applied to the AL subproblem* $\min_{x\in\mathbb{R}^n} \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k)$ *with* $u^0 = x^k_{\mathrm{init}}$ *finds a point* x^{k+1} *satisfying* (4.6) *and* (4.7).

The following theorem characterizes the *output of Algorithm* 4.1. Its proof is deferred to section 6.2.

Theorem 4.6. *Suppose that Assumption* 4.1 *holds and that* ρ_0 *is sufficiently large such that* $\delta_{f,1} \le \delta_f$ *and* $\delta_{c,1} \le \delta_c$, *where* $\delta_{f,1}$ *and* $\delta_{c,1}$ *are defined as in* (4.10). *If Algorithm* 4.1 *terminates at some iteration k, then* x^{k+1} *is a deterministic* ϵ_1*-FOSP of problem* (1.1), *and moreover, it is an* (ϵ_1, ϵ_2)*-SOSP of* (1.1) *with probability at least* $1 - \delta$.

Remark 4.7. As seen from this theorem, the output of Algorithm 4.1 is a stochastic (ϵ_1, ϵ_2)-SOSP of problem (1.1). Nevertheless, one can easily modify Algorithm 4.1 to seek some other approximate solutions. For example, if one is only interested in finding an ϵ_1-FOSP of (1.1), one can remove the condition (4.7) from Algorithm 4.1. In addition, if one aims to find a deterministic (ϵ_1, ϵ_2)-SOSP of (1.1), one can replace the condition (4.7) and Algorithm 3.1 by $\lambda_{\min}(\nabla^2_{xx}\widetilde{\mathcal{L}}(x^{k+1}, \lambda^k; \rho_k)) \geq -\tau_k^H$ and a deterministic counterpart, respectively. The purpose of imposing high probability in the condition (4.7) is to enable us to derive operation complexity of Algorithm 4.1 measured by the number of matrix-vector products.

In the rest of this section, we study the worst-case complexity of Algorithm 4.1. Since our method has two nested loops, in particular, outer loops executed by the AL method and inner loops executed by the Newton-CG method for solving the AL subproblems, we consider the following measures of complexity for Algorithm 4.1:

- *outer iteration complexity*, which measures the number of outer iterations of Algorithm 4.1;
- *total inner iteration complexity*, which measures the total number of iterations of the Newton-CG method that are performed in Algorithm 4.1;
- *operation complexity*, which measures the total number of matrix-vector products involving the Hessian of the augmented Lagrangian function that are evaluated in Algorithm 4.1.

4.1. Outer iteration complexity of Algorithm 4.1. In this subsection we establish outer iteration complexity of Algorithm 4.1. For notational convenience, we rewrite (τ_k^g, τ_k^H) arising in Algorithm 4.1 as

$$(\tau_k^g, \tau_k^H) = (\max\{\epsilon_1, \omega_1^k\}, \max\{\epsilon_2, \omega_2^k\}) \text{ with } (\omega_1, \omega_2) := (r^{\log\epsilon_1/\log 2}, r^{\log\epsilon_2/\log 2}), \tag{4.11}$$

where ϵ_1, ϵ_2, and r are the input parameters of Algorithm 4.1. Since $r > 1$ and $\epsilon_1, \epsilon_2 \in (0,1)$, it is not hard to verify that $\omega_1, \omega_2 \in (0,1)$. Also, we introduce the following quantity that will be used frequently later:

$$K_{\epsilon_1} := \left\lceil \min\{k \geq 0 : \omega_1^k \leq \epsilon_1\} \right\rceil = \lceil \log\epsilon_1 / \log\omega_1 \rceil . \tag{4.12}$$

In view of (4.11), (4.12), and the fact that

$$\log\epsilon_1/\log\omega_1 = \log\epsilon_2/\log\omega_2 = \log 2/\log r, \tag{4.13}$$

we see that $(\tau_k^g, \tau_k^H) = (\epsilon_1, \epsilon_2)$ for all $k \geq K_{\epsilon_1}$. This along with the termination criterion of Algorithm 4.1 implies that it runs for at least K_{ϵ_1} iterations and terminates once $\|c(x^{k+1})\| \leq \epsilon_1$ for some $k \geq K_{\epsilon_1}$. As a result, to establish outer iteration complexity of Algorithm 4.1, it suffices to bound such k. The resulting outer iteration complexity of Algorithm 4.1 is presented below; the proof is deferred to section 6.2.

THEOREM 4.8. *Suppose that Assumption* 4.1 *holds and that* ρ_0 *is sufficiently large such that* $\delta_{f,1} \leq \delta_f$ *and* $\delta_{c,1} \leq \delta_c$*, where* $\delta_{f,1}$ *and* $\delta_{c,1}$ *are defined in* (4.10)*. Let*

$$\rho_{\epsilon_1} := \max\left\{8(f_{hi} - f_{low} + \gamma)\epsilon_1^{-2} + 4\Lambda\epsilon_1^{-1} + 2\gamma, 2\rho_0\right\}, \tag{4.14}$$

$$\overline{K}_{\epsilon_1} := \inf\{k \geq K_{\epsilon_1} : \|c(x^{k+1})\| \leq \epsilon_1\}, \tag{4.15}$$

where K_{ϵ_1} *is defined in* (4.12)*, and* γ*,* f_{hi}*, and* f_{low} *are given in Assumption* 4.1*. Then* $\overline{K}_{\epsilon_1}$ *is finite, and Algorithm* 4.1 *terminates at iteration* $\overline{K}_{\epsilon_1}$ *with*

$$\overline{K}_{\epsilon_1} \leq \left(\frac{\log(\rho_{\epsilon_1}\rho_0^{-1})}{\log r} + 1\right)\left(\left|\frac{\log(\epsilon_1(2\delta_{c,1})^{-1})}{\log\alpha}\right| + 2\right) + 1. \tag{4.16}$$

Moreover, $\rho_k \leq r\rho_{\epsilon_1}$ *holds for* $0 \leq k \leq \overline{K}_{\epsilon_1}$

Remark 4.9 (upper bounds for $\overline{K}_{\epsilon_1}$ and $\{\rho_k\}$). As observed from Theorem 4.8, the number of outer iterations of Algorithm 4.1 for finding a stochastic (ϵ_1,ϵ_2)-SOSP of problem (1.1) is $\overline{K}_{\epsilon_1}+1$, which is at most $\mathcal{O}(|\log\epsilon_1|^2)$. In addition, the penalty parameters $\{\rho_k\}$ generated in this algorithm are at most $\mathcal{O}(\epsilon_1^{-2})$.

4.2. Total inner iteration and operation complexity of Algorithm 4.1. We present the total inner iteration and operation complexity of Algorithm 4.1 for finding a stochastic (ϵ_1,ϵ_2)-SOSP of (1.1); the proof is deferred to section 6.2.

THEOREM 4.10. *Suppose that Assumption* 4.1 *holds and that* ρ_0 *is sufficiently large such that* $\delta_{f,1}\le\delta_f$ *and* $\delta_{c,1}\le\delta_c$*, where* $\delta_{f,1}$ *and* $\delta_{c,1}$ *are defined in* (4.10)*. Then the following statements hold:*

(i) *The total number of iterations of Algorithm* 3.1 *performed in Algorithm* 4.1 *is at most* $\widetilde{\mathcal{O}}(\epsilon_1^{-4}\max\{\epsilon_1^{-2}\epsilon_2,\epsilon_2^{-3}\})$*. If c is further assumed to be affine, then it is at most* $\widetilde{\mathcal{O}}(\max\{\epsilon_1^{-2}\epsilon_2,\epsilon_2^{-3}\})$.

(ii) *The total number of matrix-vector products performed by Algorithm* 3.1 *in Algorithm* 4.1 *is at most* $\widetilde{\mathcal{O}}(\epsilon_1^{-4}\max\{\epsilon_1^{-2}\epsilon_2,\epsilon_2^{-3}\}\min\{n,\epsilon_1^{-1}\epsilon_2^{-1/2}\})$*. If c is further assumed to be affine, then it is at most* $\widetilde{\mathcal{O}}(\max\{\epsilon_1^{-2}\epsilon_2,\epsilon_2^{-3}\}\min\{n,\epsilon_1^{-1}\epsilon_2^{-1/2}\})$.

Remark 4.11.

(i) Note that the above complexity results of Algorithm 4.1 are established without assuming any constraint qualification (CQ). In contrast, similar complexity results are obtained in [60] for a proximal AL method under a generalized LICQ condition. To the best of our knowledge, our work provides the first study on complexity for finding a stochastic SOSP of (1.1) without CQ.

(ii) Letting $(\epsilon_1,\epsilon_2)=(\epsilon,\sqrt{\epsilon})$ for some $\epsilon\in(0,1)$, we see that Algorithm 4.1 achieves a total inner iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2})$ and an operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2}\min\{n,\epsilon^{-5/4}\})$ for finding a stochastic $(\epsilon,\sqrt{\epsilon})$-SOSP of problem (1.1) without CQ.

4.3. Enhanced complexity of Algorithm 4.1 under constraint qualification. In this subsection we study complexity of Algorithm 4.1 under one additional assumption that a generalized linear independence constraint qualification (GLICQ) holds for problem (1.1), which is introduced below. In particular, under GLICQ we will obtain an enhanced total inner iteration and operation complexity for Algorithm 4.1, which are significantly better than the ones in Theorem 4.10 when problem (1.1) has nonlinear constraints. Moreover, when $(\epsilon_1,\epsilon_2)=(\epsilon,\sqrt{\epsilon})$ for some $\epsilon\in(0,1)$, our enhanced complexity bounds are also better than those obtained in [60] for a proximal AL method. We now introduce the GLICQ assumption for problem (1.1).

Assumption 4.12 (GLICQ). $\nabla c(x)$ has full column rank for all $x\in\mathcal{S}(\delta_f,\delta_c)$, where $\mathcal{S}(\delta_f,\delta_c)$ is as in (4.3).

Remark 4.13. A related yet different GLICQ is imposed in [60, Assumption 2(ii)] for problem (1.1), which assumes that $\nabla c(x)$ has full column rank for all x in a level set of $f(\cdot)+\gamma\|c(\cdot)\|^2/2$. It is not hard to verify that this assumption is generally stronger than the above GLICQ assumption.

The following theorem shows that under Assumption 4.12, the total inner iteration and operation complexity results presented in Theorem 4.10 can be significantly improved; the proof is deferred to section 6.2.

THEOREM 4.14. *Suppose that Assumptions* 4.1 *and* 4.12 *hold and that* ρ_0 *is sufficiently large such that* $\delta_{f,1} \le \delta_f$ *and* $\delta_{c,1} \le \delta_c$*, where* $\delta_{f,1}$ *and* $\delta_{c,1}$ *are defined in* (4.10)*. Then the following statements hold:*

(i) *The total number of iterations of Algorithm* 3.1 *performed in Algorithm* 4.1 *is at most* $\widetilde{\mathcal{O}}(\epsilon_1^{-2}\max\{\epsilon_1^{-2}\epsilon_2, \epsilon_2^{-3}\})$*. If c is further assumed to be affine, then it is at most* $\widetilde{\mathcal{O}}(\max\{\epsilon_1^{-2}\epsilon_2, \epsilon_2^{-3}\})$*.*

(ii) *The total number of matrix-vector products performed by Algorithm* 3.1 *in Algorithm* 4.1 *is at most* $\widetilde{\mathcal{O}}(\epsilon_1^{-2}\max\{\epsilon_1^{-2}\epsilon_2, \epsilon_2^{-3}\}\min\{n, \epsilon_1^{-1/2}\epsilon_2^{-1/2}\})$*. If c is further assumed to be affine, then it is at most* $\widetilde{\mathcal{O}}(\max\{\epsilon_1^{-2}\epsilon_2, \epsilon_2^{-3}\}\min\{n, \epsilon_1^{-1/2}\epsilon_2^{-1/2}\})$*.*

Remark 4.15.

(i) As seen from Theorem 4.14, when problem (1.1) has nonlinear constraints, under GLICQ and some other suitable assumptions, Algorithm 4.1 achieves significantly better complexity bounds than the ones in Theorem 4.10 without constraint qualification.

(ii) Letting $(\epsilon_1, \epsilon_2) = (\epsilon, \sqrt{\epsilon})$ for some $\epsilon \in (0,1)$, we see that when problem (1.1) has nonlinear constraints, under GLICQ and some other suitable assumptions, Algorithm 4.1 achieves a total inner iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-7/2})$ and an operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-7/2}\min\{n, \epsilon^{-3/4}\})$. They are vastly better than the total inner iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2})$ and the operation complexity of $\widetilde{\mathcal{O}}(\epsilon^{-11/2}\min\{n, \epsilon^{-3/4}\})$ that are achieved by a proximal AL method in [60] for finding a stochastic $(\epsilon, \sqrt{\epsilon})$-SOSP of (1.1) yet under a generally stronger GLICQ.

5. Numerical results.

We conduct some preliminary experiments to test the performance of our proposed methods (Algorithms 3.1 and 4.1) and compare them with the Newton-CG method in [56] and the proximal AL method in [60], respectively. All the algorithms are coded in MATLAB, and all the computations are performed on a desktop with a 3.79 GHz AMD 3900XT 12-Core processor and 32 GB of RAM.

5.1. Regularized robust regression.

In this subsection we consider the regularized robust regression problem

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} \phi(a_i^T x - b_i) + \mu\|x\|_4^4, \tag{5.1}$$

where $\phi(t) = t^2/(1+t^2)$, $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ for any $p \ge 1$, and $\mu > 0$.

For each triple (n, m, μ), we randomly generate 10 instances of problem (5.1). In particular, we first randomly generate a_i, $1 \le i \le m$, with all the entries independently chosen from the standard normal distribution. We then randomly generate $\bar{b}_i$ according to the standard normal distribution and set $b_i = 2m\bar{b}_i$ for $i = 1, \ldots, m$.

Our aim is to find a $(10^{-5}, 10^{-5/2})$-SOSP of (5.1) for the above instances by Algorithm 3.1 and the Newton-CG method in [56] and compare their performance. For a fair comparison, we use a minimum eigenvalue oracle that returns a deterministic output for them so that they both certainly output an approximate second-order stationary point. Specifically, we use the MATLAB subroutine `[v,λ] = eigs(H,1,'smallestreal')` as the minimum eigenvalue oracle to find the minimum eigenvalue λ and its associated unit eigenvector v of a real symmetric matrix H. Also, for both methods, we choose the all-ones vector as the initial point, and set $\theta = 0.8$, $\zeta = 0.5$, and $\eta = 0.2$.

Table 2
Numerical results for problem (5.1).

			Objective value		Iterations		CPU time (seconds)	
n	m	μ	Alg. 3.1	Newton-CG	Alg. 3.1	Newton-CG	Alg. 3.1	Newton-CG
100	10	1	5.9	5.9	85.7	116.3	1.4	1.6
100	50	1	45.9	45.9	82.6	158.2	1.0	2.7
100	90	1	84.8	84.8	102.2	224.7	2.0	4.2
500	50	5	42.2	42.5	173.1	344.7	44.2	72.2
500	250	5	243.0	242.9	145.5	362.4	41.9	95.0
500	450	5	442.2	442.2	163.7	425.2	47.6	138.3
1000	100	10	90.1	90.4	162.5	361.0	110.8	259.0
1000	500	10	491.1	491.2	158.3	475.4	129.1	558.4
1000	900	10	891.1	891.1	193.5	300.7	187.0	298.5

The computational results of Algorithm 3.1 and the Newton-CG method in [56] for the instances randomly generated above are presented in Table 2. In detail, the values of n, m, and μ are listed in the first three columns, respectively. For each triple (n, m, μ), the average CPU time (in seconds), the average number of iterations, and the average final objective value over 10 random instances are given in the remaining the columns. One can observe that both methods output an approximate solution with a similar objective value, while our Algorithm 3.1 substantially outperforms the Newton-CG method in [56] in terms of CPU time. This is consistent with our theoretical finding that Algorithm 3.1 achieves a better iteration complexity than the Newton-CG method in [56] in terms of dependence on the Lipschitz constant of the Hessian for finding an approximate SOSP.

5.2. Spherically constrained regularized robust regression. In this subsection we consider the spherically constrained regularized robust regression problem

$$(5.2) \qquad \min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} \phi(a_i^T x - b_i) + \mu\|x\|_4^4 \quad \text{s.t.} \quad \|x\|_2^2 = 1,$$

where $\phi(t) = t^2/(1+t^2)$, $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ for any $p \geq 1$, and $\mu > 0$ is a tuning parameter. For each triple (n, m, μ), we randomly generate 10 instances of problem (5.2) in the same manner as described in subsection 5.1.

Our aim is to find a $(10^{-4}, 10^{-2})$-SOSP of (5.2) for the above instances by Algorithm 4.1 and the proximal AL method [60, Algorithm 3] and compare their performance. For a fair comparison, we use a minimum eigenvalue oracle that returns a deterministic output for them so that they both certainly output an approximate SOSP. Specifically, we use the MATLAB subroutine `[v,λ] = eigs(H,1,'smallestreal')` as the minimum eigenvalue oracle to find the minimum eigenvalue λ and its associated unit eigenvector v of a real symmetric matrix H. In addition, for both methods, we choose the initial point as $z^0 = (1/\sqrt{n}, \ldots, 1/\sqrt{n})^T$, the initial Lagrangian multiplier as $\lambda^0 = 0$, and the other parameters as

- $\Lambda = 100$, $\rho_0 = 10$, $\alpha = 0.25$, and $r = 10$ for Algorithm 4.1;
- $\eta = 1$, $q = 10$, and $T_0 = 2$ for the proximal AL method [60].

The computational results of Algorithm 4.1 and the proximal AL method in [60] (abbreviated as Prox-AL) for solving problem (5.2) for the instances randomly generated above are presented in Table 3. In detail, the values of n, m, and μ are listed in the first three columns, respectively. For each triple (n, m, μ), the average CPU time (in seconds), the average total number of inner iterations, the average final

TABLE 3
Numerical results for problem (5.2).

			Objective value		Feasibility violation ($\times 10^{-4}$)		Total inner iterations		CPU time (seconds)	
n	m	μ	Alg. 4.1	Prox-AL	Alg. 4.1	Prox-AL	Alg. 4.1	Prox-AL	Alg. 4.1	Prox-AL
100	10	1	7.1	7.1	0.18	0.27	40.9	97.3	0.73	2.2
100	50	1	46.6	46.6	0.21	0.30	37.0	86.3	0.78	1.7
100	90	1	87.0	87.0	0.12	0.40	39.5	68.6	1.1	1.9
500	50	5	44.4	44.4	0.40	0.68	59.0	343.4	11.4	134.9
500	250	5	244.3	244.3	0.37	0.47	59.0	543.3	11.7	178.2
500	450	5	444.0	444.0	0.27	0.53	66.7	634.1	17.1	158.2
1000	100	10	92.8	92.8	0.28	0.42	95.0	2054.6	46.3	1516.8
1000	500	10	491.9	491.9	0.22	0.72	68.3	756.2	39.5	558.6
1000	900	10	893.4	893.4	0.19	0.37	81.8	1281.4	57.7	1099.6

objective value, and the average final feasibility violation over 10 random instances are given in the remaining columns. One can observe that both methods output an approximate solution of similar quality in terms of objective value and feasibility violation, while our Algorithm 4.1 vastly outperforms the proximal AL method in [60] in terms of CPU time. This corroborates our theoretical finding that Algorithm 4.1 achieves a significantly better operation complexity than the proximal AL method in [60] for finding an approximate SOSP.

6. Proof of the main results.

We provide proofs of our main results in sections 3 and 4, including Theorem 3.2, Lemma 4.4, and Theorems 4.6, 4.8, 4.10, and 4.14.

6.1. Proof of the main results in section 3.

In this subsection we first establish several technical lemmas and then use them to prove Theorem 3.2.

One can observe from Assumption 3.1(b) that for all x and $y \in \Omega$,

(6.1)
$$\|\nabla F(y) - \nabla F(x) - \nabla^2 F(x)(y-x)\| \le L_H^F \|y-x\|^2/2,$$

(6.2)
$$F(y) \le F(x) + \nabla F(x)^T(y-x) + (y-x)^T \nabla^2 F(x)(y-x)/2 + L_H^F\|y-x\|^3/6.$$

The next lemma provides useful properties of the output of Algorithm A.1; the proof is similar to the ones in [56, Lemma 3] and [54, Lemma 7] and is thus omitted here.

LEMMA 6.1. *Suppose that Assumption* 3.1 *holds and the direction d^t results from the output d of Algorithm* A.1 *with a type specified in* d_type *at some iteration t of Algorithm* 3.1. *Then the following statements hold:*

(i) *If* d_type=SOL*, then d^t satisfies*

$$\epsilon_H \|d^t\|^2 \le (d^t)^T \left(\nabla^2 F(x^t) + 2\epsilon_H I\right) d^t, \tag{6.3}$$
$$\|d^t\| \le 1.1\epsilon_H^{-1}\|\nabla F(x^t)\|, \tag{6.4}$$
$$(d^t)^T \nabla F(x^t) = -(d^t)^T\left(\nabla^2 F(x^t) + 2\epsilon_H I\right) d^t, \tag{6.5}$$
$$\|(\nabla^2 F(x^t) + 2\epsilon_H I)d^t + \nabla F(x^t)\| \le \epsilon_H \zeta \|d^t\|/2. \tag{6.6}$$

(ii) *If* d_type=NC*, then d^t satisfies $(d^t)^T\nabla F(x^t) \le 0$ and*

$$(d^t)^T \nabla^2 F(x^t) d^t/\|d^t\|^2 = -\|d^t\| \le -\epsilon_H. \tag{6.7}$$

The next lemma shows that when the search direction d^t in Algorithm 3.1 is of type "SOL," the line-search step results in a sufficient reduction on F.

LEMMA 6.2. *Suppose that Assumption* 3.1 *holds and the direction d^t results from the output d of Algorithm* A.1 *with* d_type=SOL *at some iteration t of Algorithm* 3.1. *Let U_g^F and $c_{\rm sol}$ be given in* (3.3) *and* (3.13)*, respectively. Then the following statements hold:*

(i) *The step length α_t is well-defined, and moreover,*

$$\alpha_t \geq \min\left\{1, \sqrt{\frac{\min\{6(1-\eta),2\}}{1.1L_H^F U_g^F}}\theta\epsilon_H\right\}. \tag{6.8}$$

(ii) *The next iterate $x^{t+1}=x^t+\alpha_t d^t$ satisfies*

$$F(x^t)-F(x^{t+1}) \geq c_{\rm sol}\min\{\|\nabla F(x^{t+1})\|^2\epsilon_H^{-1}, \epsilon_H^3\}. \tag{6.9}$$

Proof. One can observe that F is descent along the iterates (whenever well-defined) generated by Algorithm 3.1, which together with $x^0=u^0$ implies that $F(x^t)\leq F(u^0)$ and hence $\|\nabla F(x^t)\|\leq U_g^F$ due to (3.3). In addition, since d^t results from the output d of Algorithm A.1 with d_type=SOL, one can see that $\|\nabla F(x^t)\|>\epsilon_g$ and (6.3)–(6.6) hold for d^t. Moreover, by $\|\nabla F(x^t)\|>\epsilon_g$ and (6.6), one can conclude that $d^t\neq 0$.

We first prove statement (i). If (3.10) holds for $j=0$, then $\alpha_t=1$, which clearly implies that (6.8) holds. We now suppose that (3.10) fails for $j=0$. Claim that for all $j\geq 0$ that violate (3.10), it holds that

$$\theta^{2j} \geq \min\{6(1-\eta),2\}\epsilon_H(L_H^F)^{-1}\|d^t\|^{-1}. \tag{6.10}$$

Indeed, suppose that (3.10) is violated by some $j\geq 0$. We now show that (6.10) holds for such j by considering two separate cases below.

Case 1. $F(x^t+\theta^j d^t)>F(x^t)$. Let $\phi(\alpha)=F(x^t+\alpha d^t)$. Then $\phi(\theta^j)>\phi(0)$. Also, since $d^t\neq 0$, by (6.3) and (6.5), one has $\phi'(0)=\nabla F(x^t)^T d^t=-(d^t)^T(\nabla^2 F(x^t)+2\epsilon_H I)d^t\leq -\epsilon_H\|d^t\|^2<0$. Using these, we can observe that there exists a local minimizer $\alpha^*\in(0,\theta^j)$ of ϕ such that $\phi'(\alpha^*)=\nabla F(x^t+\alpha^* d^t)^T d^t=0$ and $\phi(\alpha^*)<\phi(0)$, which implies that $F(x^t+\alpha^* d^t)<F(x^t)\leq F(u^0)$. Hence, (6.1) holds for $x=x^t$ and $y=x^t+\alpha^* d^t$. Using this, $0<\alpha^*<\theta^j\leq 1$, and $\nabla F(x^t+\alpha^* d^t)^T d^t=0$, we obtain

$$\begin{aligned}
&\tfrac{(\alpha^*)^2L_H^F}{2}\|d^t\|^3 \overset{(6.1)}{\geq} \|d^t\|\|\nabla F(x^t+\alpha^* d^t)-\nabla F(x^t)-\alpha^*\nabla^2 F(x^t)d^t\| \\
&\geq (d^t)^T(\nabla F(x^t+\alpha^* d^t)-\nabla F(x^t)-\alpha^*\nabla^2 F(x^t)d^t) \\
&= -(d^t)^T\nabla F(x^t)-\alpha^*(d^t)^T\nabla^2 F(x^t)d^t \\
&\overset{(6.5)}{=} (1-\alpha^*)(d^t)^T(\nabla^2 F(x^t)+2\epsilon_H I)d^t+2\alpha^*\epsilon_H\|d^t\|^2 \\
&\overset{(6.3)}{\geq} (1+\alpha^*)\epsilon_H\|d^t\|^2 \geq \epsilon_H\|d^t\|^2,
\end{aligned}$$

which along with $d^t\neq 0$ implies that $(\alpha^*)^2\geq 2\epsilon_H(L_H^F)^{-1}\|d^t\|^{-1}$. Using this and $\theta^j>\alpha^*$, we conclude that (6.10) holds in this case.

Case 2. $F(x^t+\theta^j d^t)\leq F(x^t)$. This together with $F(x^t)\leq F(u^0)$ implies that (6.2) holds for $x=x^t$ and $y=x^t+\theta^j d^t$. Then, because j violates (3.10), we obtain

$$
\begin{aligned}
-\eta\epsilon_H\theta^{2j}\|d^t\|^2 &\le F(x^t+\theta^j d^t)-F(x^t)\\
&\overset{(6.2)}{\le} \theta^j\nabla F(x^t)^T d^t+\frac{\theta^{2j}}{2}(d^t)^T\nabla^2F(x^t)d^t+\frac{L_H^F}{6}\theta^{3j}\|d^t\|^3\\
&\overset{(6.5)}{=} -\theta^j(d^t)^T(\nabla^2F(x^t)+2\epsilon_H I)d^t+\frac{\theta^{2j}}{2}(d^t)^T\nabla^2F(x^t)d^t+\frac{L_H^F}{6}\theta^{3j}\|d^t\|^3\\
&= -\theta^j\left(1-\frac{\theta^j}{2}\right)(d^t)^T(\nabla^2F(x^t)+2\epsilon_H I)d^t-\theta^{2j}\epsilon_H\|d^t\|^2+\frac{L_H^F}{6}\theta^{3j}\|d^t\|^3\\
&\overset{(6.3)}{\le} -\theta^j\left(1-\frac{\theta^j}{2}\right)\epsilon_H\|d^t\|^2-\theta^{2j}\epsilon_H\|d^t\|^2+\frac{L_H^F}{6}\theta^{3j}\|d^t\|^3\\
&\le -\theta^j\epsilon_H\|d^t\|^2+\frac{L_H^F}{6}\theta^{3j}\|d^t\|^3. \qquad (6.11)
\end{aligned}
$$

Recall that $d^t\neq 0$. Dividing both sides of (6.11) by $L_H^F\theta^j\|d^t\|^3/6$ and using $\eta,\theta\in(0,1)$, we obtain that $\theta^{2j}\ge 6(1-\theta^j\eta)\epsilon_H(L_H^F)^{-1}\|d^t\|^{-1}\ge 6(1-\eta)\epsilon_H(L_H^F)^{-1}\|d^t\|^{-1}$. Hence, (6.10) also holds in this case.

Combining the above two cases, we conclude that (6.10) holds for any $j\ge 0$ that violates (3.10). By this and $\theta\in(0,1)$, one can see that all $j\ge 0$ that violate (3.10) must be bounded above. It then follows that the step length α_t associated with (3.10) is well-defined. We next prove (6.8). Observe from the definition of j_t in Algorithm 3.1 that $j=j_t-1$ violates (3.10) and hence (6.10) holds for $j=j_t-1$. Then, by (6.10) with $j=j_t-1$ and $\alpha_t=\theta^{j_t}$, one has

$$
\alpha_t=\theta^{j_t}\ge\sqrt{\min\{6(1-\eta),2\}\epsilon_H(L_H^F)^{-1}}\;\theta\|d^t\|^{-1/2}, \qquad (6.12)
$$

which, along with (6.4) and $\|\nabla F(x^t)\|\le U_g^F$, implies (6.8). This proves statement (i).

We next prove statement (ii) by considering two separate cases.

Case 1. $\alpha_t=1$. By this assertion, one knows that (3.10) holds for $j=0$. It then follows that $F(x^t+d^t)\le F(x^t)\le F(u^0)$, which implies that (6.1) holds for $x=x^t$ and $y=x^t+d^t$. By this and (6.6), one has

$$
\begin{aligned}
\|\nabla F(x^{t+1})\|=\|\nabla F(x^t+d^t)\| &\le \|\nabla F(x^t+d^t)-\nabla F(x^t)-\nabla^2F(x^t)d^t\|\\
&\quad+\|(\nabla^2F(x^t)+2\epsilon_H I)d^t+\nabla F(x^t)\|+2\epsilon_H\|d^t\|\\
&\le \tfrac{L_H^F}{2}\|d^t\|^2+\tfrac{4+\zeta}{2}\epsilon_H\|d^t\|,
\end{aligned}
$$

where the last inequality follows from (6.1) and (6.6). Solving the above inequality for $\|d^t\|$ and using the fact that $\|d^t\|>0$, we obtain that

$$
\begin{aligned}
\|d^t\| &\ge \tfrac{-(4+\zeta)\epsilon_H+\sqrt{(4+\zeta)^2\epsilon_H^2+8L_H^F\|\nabla F(x^{t+1})\|}}{2L_H^F}\\
&\ge \tfrac{-(4+\zeta)\epsilon_H+\sqrt{(4+\zeta)^2\epsilon_H^2+8L_H^F\epsilon_H^2}}{2L_H^F}\min\{\|\nabla F(x^{t+1})\|/\epsilon_H^2,1\}\\
&= \tfrac{4}{4+\zeta+\sqrt{(4+\zeta)^2+8L_H^F}}\min\{\|\nabla F(x^{t+1})\|/\epsilon_H,\epsilon_H\},
\end{aligned}
$$

where the second inequality follows from the inequality $-a+\sqrt{a^2+bs}\ge(-a+\sqrt{a^2+b})\min\{s,1\}$ for all $a,b,s\ge 0$, which can be verified by performing a rationalization to the terms $-a+\sqrt{a^2+b}$ and $-a+\sqrt{a^2+bs}$, respectively. By this, $\alpha_t=1$, (3.10), and (3.13), one can see that (6.9) holds.

Case 2. $\alpha_t < 1$. It then follows that $j = 0$ violates (3.10) and hence (6.10) holds for $j = 0$. Now, letting $j = 0$ in (6.10), we obtain that $\|d^t\| \geq \min\{6(1-\eta), 2\}\epsilon_H / L_H^F$, which together with (3.10) and (6.12) implies that

$$F(x^t) - F(x^{t+1}) \geq \eta \epsilon_H \theta^{2j_t} \|d^t\|^2 \geq \eta \frac{\min\{6(1-\eta), 2\}\epsilon_H^2}{L_H^F} \theta^2 \|d^t\| \geq \eta \left[\frac{\min\{6(1-\eta), 2\}\theta}{L_H^F}\right]^2 \epsilon_H^3.$$

By this and (3.13), one can see that (6.9) also holds in this case. ∎

The following lemma shows that when the search direction d^t in Algorithm 3.1 is of type "NC," the line-search step results in a sufficient reduction on F as well.

LEMMA 6.3. *Suppose that Assumption* 3.1 *holds and the direction d^t results from either the output d of Algorithm* A.1 *with* d_type=NC *or the output v of Algorithm* B.1 *at some iteration t of Algorithm* 3.1. *Let $c_{\rm nc}$ be defined as in* (3.14). *Then the following statements hold:*

(i) *The step length α_t is well-defined, and $\alpha_t \geq \min\{1, \theta/L_H^F, 3(1-\eta)\theta/L_H^F\}$.*

(ii) *The next iterate $x^{t+1} = x^t + \alpha_t d^t$ satisfies $F(x^t) - F(x^{t+1}) \geq c_{\rm nc}\epsilon_H^3$.*

Proof. Observe that F is descent along the iterates (whenever well-defined) generated by Algorithm 3.1. Using this and $x^0 = u^0$, we have $F(x^t) \leq F(u^0)$. By the assumption on d^t, one can see from Algorithm 3.1 that d^t is a negative curvature direction given in (3.7) or (3.9). Also, notice that the vector v returned from Algorithm B.1 satisfies $\|v\| = 1$. By these results, Lemma 6.1(ii), (3.7), and (3.9), one can observe that

$$\nabla F(x^t)^T d^t \leq 0, \quad (d^t)^T \nabla^2 F(x^t) d^t = -\|d^t\|^3 < 0. \tag{6.13}$$

We first prove statement (i). If (3.11) holds for $j = 0$, then $\alpha_t = 1$, which clearly implies that $\alpha_t \geq \min\{1, \theta/L_H^F, 3(1-\eta)\theta/L_H^F\}$. We now suppose that (3.11) fails for $j = 0$. Claim that for all $j \geq 0$ that violate (3.11), it holds that

$$\theta^j \geq \min\{1/L_H^F, 3(1-\eta)/L_H^F\}. \tag{6.14}$$

Indeed, suppose that (3.11) is violated by some $j \geq 0$. We now show that (6.14) holds for such j by considering two separate cases.

Case 1. $F(x^t + \theta^j d^t) > F(x^t)$. Let $\phi(\alpha) = F(x^t + \alpha d^t)$. Then $\phi(\theta^j) > \phi(0)$. Also, by (6.13), one has $\phi'(0) = \nabla F(x^t)^T d^t \leq 0$ and $\phi''(0) = (d^t)^T \nabla^2 F(x^t) d^t < 0$. Using these, we can observe that there exists a local minimizer $\alpha^* \in (0, \theta^j)$ of ϕ such that $\phi(\alpha^*) < \phi(0)$, namely, $F(x^t + \alpha^* d^t) < F(x^t)$. By the second-order optimality condition of ϕ at α^*, one has $\phi''(\alpha^*) = (d^t)^T \nabla^2 F(x^t + \alpha^* d^t) d^t \geq 0$. Since $F(x^t + \alpha^* d^t) < F(x^t) \leq F(u^0)$, it follows that (3.2) holds for $x = x^t$ and $y = x^t + \alpha^* d^t$. Using this, the second relation in (6.13), and $(d^t)^T \nabla^2 F(x^t + \alpha^* d^t) d^t \geq 0$, we obtain that

$$\begin{aligned} L_H^F \alpha^* \|d^t\|^3 &\overset{(3.2)}{\geq} \|d^t\|^2 \|\nabla^2 F(x^t + \alpha^* d^t) - \nabla^2 F(x^t)\| \\ &\geq (d^t)^T (\nabla^2 F(x^t + \alpha^* d^t) - \nabla^2 F(x^t)) d^t \geq -(d^t)^T \nabla^2 F(x^t) d^t = \|d^t\|^3. \end{aligned} \tag{6.15}$$

Recall from (6.13) that $d^t \neq 0$. It then follows from (6.15) that $\alpha^* \geq 1/L_H^F$, which along with $\theta^j > \alpha^*$ implies that $\theta^j > 1/L_H^F$. Hence, (6.14) holds in this case.

Case 2. $F(x^t + \theta^j d^t) \leq F(x^t)$. It follows from this assertion and $F(x^t) \leq F(u^0)$ that (6.2) holds for $x = x^t$ and $y = x^t + \theta^j d^t$. By this result and the fact that j violates (3.11), one has

$$\begin{aligned} -\tfrac{\eta}{2}\theta^{2j}\|d^t\|^3 &\leq F(x^t + \theta^j d^t) - F(x^t) \overset{(6.2)}{\leq} \theta^j \nabla F(x^t)^T d^t + \tfrac{\theta^{2j}}{2}(d^t)^T \nabla^2 F(x^t) d^t + \tfrac{L_H^F}{6}\theta^{3j}\|d^t\|^3 \\ &\overset{(6.13)}{\leq} -\tfrac{\theta^{2j}}{2}\|d^t\|^3 + \tfrac{L_H^F}{6}\theta^{3j}\|d^t\|^3, \end{aligned}$$

which together with $d^t \neq 0$ implies that $\theta^j \geq 3(1-\eta)/L_H^F$. Hence, (6.14) also holds in this case.

Combining the above two cases, we conclude that (6.14) holds for any $j \geq 0$ that violates (3.11). By this and $\theta \in (0,1)$, one can see that all $j \geq 0$ that violate (3.11) must be bounded above. It then follows that the step length α_t associated with (3.11) is well-defined. We next derive a lower bound for α_t. Notice from the definition of j_t in Algorithm 3.1 that $j = j_t - 1$ violates (3.11) and hence (6.14) holds for $j = j_t - 1$. Then, by (6.14) with $j = j_t - 1$ and $\alpha_t = \theta^{j_t}$, one has $\alpha_t = \theta^{j_t} \geq \min\{\theta/L_H^F, 3(1-\eta)\theta/L_H^F\}$, which immediately yields $\alpha_t \geq \min\{1, \theta/L_H^F, 3(1-\eta)\theta/L_H^F\}$ as desired.

We next prove statement (ii) by considering two separate cases.

Case 1. d^t results from the output d of Algorithm A.1 with d_type=NC. It then follows from (6.7) that $\|d^t\| \geq \epsilon_H$. This together with (3.11) and statement (i) implies that statement (ii) holds.

Case 2. d^t results from the output v of Algorithm B.1. Notice from Algorithm B.1 that $\|v\| = 1$ and $v^T \nabla^2 F(x^t) v \leq -\epsilon_H/2$, which along with (3.9) yields $\|d^t\| \geq \epsilon_H/2$. By this, (3.11), and statement (i), one can see that statement (ii) again holds. ∎

Proof of Theorem 3.2. For notational convenience, we let $\{x^t\}_{t\in\mathbb{T}}$ denote all the iterates generated by Algorithm 3.1, where $\mathbb{T}$ is a set of consecutive nonnegative integers starting from 0. Notice that F is descent along the iterates generated by Algorithm 3.1, which together with $x^0 = u^0$ implies that $x^t \in \{x : F(x) \leq F(u^0)\}$. It then follows from (3.3) that $\|\nabla^2 F(x^t)\| \leq U_H^F$ holds for all $t \in \mathbb{T}$.

(i) Suppose for contradiction that the total number of calls of Algorithm B.1 in Algorithm 3.1 is more than T_2. Notice from Algorithm 3.1 and Lemma 6.3(ii) that each of these calls, except the last one, returns a sufficiently negative curvature direction, and each of them results in a reduction on F of at least $c_{\rm nc}\epsilon_H^3$. Hence, $T_2 c_{\rm nc}\epsilon_H^3 \leq \sum_{t\in\mathbb{T}}[F(x^t) - F(x^{t+1})] \leq F(x^0) - F_{\rm low} = F_{\rm hi} - F_{\rm low}$, which contradicts the definition of T_2 given in (3.12). Hence, statement (i) of Theorem 3.2 holds.

(ii) Suppose for contradiction that the total number of calls of Algorithm A.1 in Algorithm 3.1 is more than T_1. Observe that if Algorithm A.1 is called at some iteration t and generates the next iterate x^{t+1} satisfying $\|\nabla F(x^{t+1})\| \leq \epsilon_g$, then Algorithm B.1 must be called at the next iteration $t+1$. In view of this and statement (i) of Theorem 3.2, we see that the total number of such iterations t is at most T_2. Hence, the total number of iterations t of Algorithm 3.1 at which Algorithm A.1 is called and generates the next iterate x^{t+1} satisfying $\|\nabla F(x^{t+1})\| > \epsilon_g$ is at least $T_1 - T_2 + 1$. Moreover, for each of such iterations t, we observe from Lemmas 6.2(ii) and 6.3(ii) that $F(x^t) - F(x^{t+1}) \geq \min\{c_{\rm sol}, c_{\rm nc}\}\min\{\epsilon_g^2\epsilon_H^{-1}, \epsilon_H^3\}$. It then follows that $(T_1 - T_2 + 1)\min\{c_{\rm sol}, c_{\rm nc}\}\min\{\epsilon_g^2\epsilon_H^{-1}, \epsilon_H^3\} \leq \sum_{t\in\mathbb{T}}[F(x^t) - F(x^{t+1})] \leq F_{\rm hi} - F_{\rm low}$, which contradicts the definition of T_1 and T_2 given in (3.12). Hence, statement (ii) of Theorem 3.2 holds.

(iii) Notice that either Algorithm A.1 or B.1 is called at each iteration of Algorithm 3.1. It follows from this and statements (i) and (ii) of Theorem 3.2 that the total number of iterations of Algorithm 3.1 is at most $T_1 + T_2$. In addition, the relation (3.15) follows from (3.13), (3.14), and (3.12). One can also observe that the output x^t of Algorithm 3.1 satisfies $\|\nabla F(x^t)\| \leq \epsilon_g$ deterministically and $\lambda_{\min}(\nabla^2 F(x^t)) \geq -\epsilon_H$ with probability at least $1-\delta$ for some $0 \leq t \leq T_1 + T_2$, where the latter part is due to Algorithm B.1. This completes the proof of statement (ii) of Theorem 3.2.

(iv) By Theorem A.1 with $(H, \varepsilon) = (\nabla^2 F(x^t), \epsilon_H)$ and the fact that $\|\nabla^2 F(x^t)\| \leq U_H^F$, one can observe that the number of Hessian-vector products required by each call of Algorithm A.1 with input $U = 0$ is at most $\widetilde{\mathcal{O}}(\min\{n, (U_H^F/\epsilon_H)^{1/2}\})$. In addition, by Theorem B.1 with $(H,\varepsilon)=(\nabla^2 F(x^t),\epsilon_H)$, $\|\nabla^2 F(x^t)\|\le U_H^F$, and the fact that each iteration of the Lanczos method requires only one matrix-vector product, one can observe that the number of Hessian-vector products required by each call of Algorithm B.1 is also at most $\widetilde{\mathcal{O}}(\min\{n,(U_H^F/\epsilon_H)^{1/2}\})$. Based on these observations and statement (iii) of Theorem 3.2, we see that statement (iv) of this theorem holds. ∎

6.2. Proof of the main results in section 4. Recall from Assumption 4.1(a) that $\|c(z_{\epsilon_1})\|\le\epsilon_1/2<1$. By virtue of this, (4.2), and the definition of $\tilde c$ in (4.4), we obtain that

$$f(x)+\gamma\|\tilde c(x)\|^2\ge f(x)+\gamma\|c(x)\|^2/2-\gamma\|c(z_{\epsilon_1})\|^2\ge f_{\rm low}-\gamma\quad\forall x\in\mathbb{R}^n. \tag{6.16}$$

We now prove the following auxiliary lemma that will be used frequently later.

LEMMA 6.4. *Suppose that Assumption* 4.1 *holds. Let γ, $f_{\rm hi}$, and $f_{\rm low}$ be given in Assumption* 4.1. *Assume that $\rho>2\gamma$, $\lambda\in\mathbb{R}^m$, and $x\in\mathbb{R}^n$ satisfy*

$$\widetilde{\mathcal{L}}(x,\lambda;\rho)\le f_{\rm hi}, \tag{6.17}$$

where $\widetilde{\mathcal{L}}$ is defined as in (4.5). *Then the following statements hold:*

(i) $f(x)\le f_{\rm hi}+\|\lambda\|^2/(2\rho)$.
(ii) $\|\tilde c(x)\|\le\sqrt{2(f_{\rm hi}-f_{\rm low}+\gamma)/(\rho-2\gamma)+\|\lambda\|^2/(\rho-2\gamma)^2}+\|\lambda\|/(\rho-2\gamma)$.
(iii) *If $\rho\ge\|\lambda\|^2/(2\tilde\delta_f)$ for some $\tilde\delta_f>0$, then $f(x)\le f_{\rm hi}+\tilde\delta_f$.*
(iv) *If*

$$\rho\ge 2(f_{\rm hi}-f_{\rm low}+\gamma)\tilde\delta_c^{-2}+2\|\lambda\|\tilde\delta_c^{-1}+2\gamma \tag{6.18}$$

for some $\tilde\delta_c>0$, then $\|\tilde c(x)\|\le\tilde\delta_c$.

Proof. (i) It follows from (6.17) and the definition of $\widetilde{\mathcal{L}}$ in (4.5) that

$$f_{\rm hi}\ge f(x)+\lambda^T\tilde c(x)+\frac{\rho}{2}\|\tilde c(x)\|^2=f(x)+\frac{\rho}{2}\left\|\tilde c(x)+\frac{\lambda}{\rho}\right\|^2-\frac{\|\lambda\|^2}{2\rho}\ge f(x)-\frac{\|\lambda\|^2}{2\rho}.$$

Hence, statement (i) holds.

(ii) In view of (6.16) and (6.17), one has

$$\begin{aligned}f_{\rm hi}&\overset{(6.17)}{\ge} f(x)+\lambda^T\tilde c(x)+\tfrac{\rho}{2}\|\tilde c(x)\|^2=f(x)+\gamma\|\tilde c(x)\|^2+\tfrac{\rho-2\gamma}{2}\left\|\tilde c(x)+\tfrac{\lambda}{\rho-2\gamma}\right\|^2-\tfrac{\|\lambda\|^2}{2(\rho-2\gamma)}\\&\overset{(6.16)}{\ge} f_{\rm low}-\gamma+\tfrac{\rho-2\gamma}{2}\left\|\tilde c(x)+\tfrac{\lambda}{\rho-2\gamma}\right\|^2-\tfrac{\|\lambda\|^2}{2(\rho-2\gamma)}.\end{aligned}$$

It then follows that $\left\|\tilde c(x)+\frac{\lambda}{\rho-2\gamma}\right\|\le\sqrt{\frac{2(f_{\rm hi}-f_{\rm low}+\gamma)}{\rho-2\gamma}+\frac{\|\lambda\|^2}{(\rho-2\gamma)^2}}$, which implies that statement (ii) holds.

(iii) Statement (iii) immediately follows from statement (i) and $\rho\ge\|\lambda\|^2/(2\tilde\delta_f)$.

(iv) Suppose that (6.18) holds. Multiplying both sides of (6.18) by $\tilde\delta_c^2$ and rearranging the terms, we have $(\rho-2\gamma)\tilde\delta_c^2-2\|\lambda\|\tilde\delta_c-2(f_{\rm hi}-f_{\rm low}+\gamma)\ge 0$. Recall that $\rho>2\gamma$ and $\tilde\delta_c>0$. Solving this inequality for $\tilde\delta_c$ yields

$$\tilde\delta_c\ge\sqrt{2(f_{\rm hi}-f_{\rm low}+\gamma)/(\rho-2\gamma)+\|\lambda\|^2/(\rho-2\gamma)^2}+\|\lambda\|/(\rho-2\gamma),$$

which along with statement (ii) implies that $\|\tilde c(x)\|\le\tilde\delta_c$. Hence, statement (iv) holds. ∎

Proof of Lemma 4.4. (i) Let x be any point such that $\widetilde{\mathcal{L}}(x,\lambda^k;\rho_k)\le\widetilde{\mathcal{L}}(x^k_{\rm init},\lambda^k;\rho_k)$. It then follows from (4.9) that $\widetilde{\mathcal{L}}(x,\lambda^k;\rho_k)\le f_{\rm hi}$. By this, $\|\lambda^k\|\le\Lambda$, $\rho_k\ge\rho_0>2\gamma$, $\delta_{f,1}\le\delta_f$, $\delta_{c,1}\le\delta_c$, and Lemma 6.4 with $(\lambda,\rho)=(\lambda^k,\rho_k)$, one has $f(x)\le f_{\rm hi}+\|\lambda^k\|^2/(2\rho_k)\le f_{\rm hi}+\Lambda^2/(2\rho_0)=f_{\rm hi}+\delta_{f,1}\le f_{\rm hi}+\delta_f$ and

$$\begin{aligned}\|\tilde c(x)\|&\le\sqrt{\frac{2(f_{\rm hi}-f_{\rm low}+\gamma)}{\rho_k-2\gamma}+\frac{\|\lambda^k\|^2}{(\rho_k-2\gamma)^2}}+\frac{\|\lambda^k\|}{\rho_k-2\gamma}\\&\le\sqrt{\frac{2(f_{\rm hi}-f_{\rm low}+\gamma)}{\rho_0-2\gamma}+\frac{\Lambda^2}{(\rho_0-2\gamma)^2}}+\frac{\Lambda}{\rho_0-2\gamma}=\delta_{c,1}\le\delta_c.\end{aligned}\tag{6.19}$$

Also, recall from the definition of $\tilde c$ in (4.4) and $\|c(z_{\epsilon_1})\|\le1$ that $\|c(x)\|\le1+\|\tilde c(x)\|$. This together with the above inequalities and (4.3) implies $x\in\mathcal{S}(\delta_f,\delta_c)$. Hence, statement (i) of Lemma 4.4 holds.

(ii) Note that $\inf_{x\in\mathbb{R}^n}\widetilde{\mathcal{L}}(x,\lambda^k;\rho_k)=\inf_{x\in\mathbb{R}^n}\{\widetilde{\mathcal{L}}(x,\lambda^k;\rho_k):\widetilde{\mathcal{L}}(x,\lambda^k;\rho_k)\le\widetilde{\mathcal{L}}(x^k_{\rm init},\lambda^k;\rho_k)\}$. Consequently, to prove statement (ii) of Lemma 4.4, it suffices to show that

$$\inf_{x\in\mathbb{R}^n}\{\widetilde{\mathcal{L}}(x,\lambda^k;\rho_k):\widetilde{\mathcal{L}}(x,\lambda^k;\rho_k)\le\widetilde{\mathcal{L}}(x^k_{\rm init},\lambda^k;\rho_k)\}\ge f_{\rm low}-\gamma-\Lambda\delta_c.\tag{6.20}$$

To this end, let x be any point satisfying $\widetilde{\mathcal{L}}(x,\lambda^k;\rho_k)\le\widetilde{\mathcal{L}}(x^k_{\rm init},\lambda^k;\rho_k)$. We then know from (6.19) that $\|\tilde c(x)\|\le\delta_c$. By this, $\|\lambda^k\|\le\Lambda$, $\rho_k>2\gamma$, and (6.16), one has

$$\begin{aligned}\widetilde{\mathcal{L}}(x,\lambda^k;\rho_k)&=f(x)+\gamma\|\tilde c(x)\|^2+(\lambda^k)^T\tilde c(x)+\tfrac{\rho_k-2\gamma}{2}\|\tilde c(x)\|^2\\&\ge f(x)+\gamma\|\tilde c(x)\|^2-\Lambda\|\tilde c(x)\|\ge f_{\rm low}-\gamma-\Lambda\delta_c,\end{aligned}$$

and hence (6.20) holds as desired. ∎

Proof of Theorem 4.6. Suppose that Algorithm 4.1 terminates at some iteration k, that is, $\tau^g_k\le\epsilon_1$, $\tau^H_k\le\epsilon_2$, and $\|c(x^{k+1})\|\le\epsilon_1$ hold. Then, by $\tau^g_k\le\epsilon_1$, $\tilde\lambda^{k+1}=\lambda^k+\rho_k\tilde c(x^{k+1})$, $\nabla\tilde c=\nabla c$, and the second relation in (4.6), one has $\|\nabla f(x^{k+1})+\nabla c(x^{k+1})\tilde\lambda^{k+1}\|=\|\nabla f(x^{k+1})+\nabla\tilde c(x^{k+1})(\lambda^k+\rho_k\tilde c(x^{k+1}))\|=\|\nabla_x\widetilde{\mathcal{L}}(x^{k+1},\lambda^k;\rho_k)\|\le\tau^g_k\le\epsilon_1$. Hence, $(x^{k+1},\tilde\lambda^{k+1})$ satisfies the first relation in (2.4). In addition, by (4.7) and $\tau^H_k\le\epsilon_2$, one can show that $\lambda_{\min}(\nabla^2_{xx}\widetilde{\mathcal{L}}(x^{k+1},\lambda^k;\rho_k))\ge-\epsilon_2$ with probability at least $1-\delta$, which leads to $d^T\nabla^2_{xx}\widetilde{\mathcal{L}}(x^{k+1},\lambda^k;\rho_k)d\ge-\epsilon_2\|d\|^2$ for all $d\in\mathbb{R}^n$ with probability at least $1-\delta$. Using this, $\tilde\lambda^{k+1}=\lambda^k+\rho_k\tilde c(x^{k+1})$, $\nabla\tilde c=\nabla c$, and $\nabla^2\tilde c_i=\nabla^2 c_i$ for $1\le i\le m$, we see that with probability at least $1-\delta$, it holds that $d^T(\nabla^2 f(x^{k+1})+\sum_{i=1}^m\tilde\lambda^{k+1}_i\nabla^2 c_i(x^{k+1})+\rho_k\nabla c(x^{k+1})\nabla c(x^{k+1})^T)d\ge-\epsilon_2\|d\|^2$ for all $d\in\mathbb{R}^n$, which implies $d^T(\nabla^2 f(x^{k+1})+\sum_{i=1}^m\tilde\lambda^{k+1}_i\nabla^2 c_i(x^{k+1}))d\ge-\epsilon_2\|d\|^2$ for all $d\in\mathcal{C}(x^{k+1})$, where $\mathcal{C}(\cdot)$ is defined in (2.3). Hence, $(x^{k+1},\tilde\lambda^{k+1})$ satisfies (2.5) with probability at least $1-\delta$. Combining these with $\|c(x^{k+1})\|\le\epsilon_1$, we conclude that x^{k+1} is a deterministic ϵ_1-FOSP of (1.1) and an (ϵ_1,ϵ_2)-SOSP of (1.1) with probability at least $1-\delta$. Hence, Theorem 4.6 holds. ∎

Proof of Theorem 4.8. It follows from (4.14) that $\rho_{\epsilon_1}\ge2\rho_0$. By this, one has

$$K_{\epsilon_1}\overset{(4.12)}{=}\lceil\log\epsilon_1/\log\omega_1\rceil\overset{(4.11)}{=}\lceil\log2/\log r\rceil\le\log(\rho_{\epsilon_1}\rho_0^{-1})/\log r+1.\tag{6.21}$$

Notice that $\{\rho_k\}$ is either unchanged or increased by a ratio r as k increases. By this fact and (6.21), we see that

$$\max_{0\le k\le K_{\epsilon_1}}\rho_k\le r^{K_{\epsilon_1}}\rho_0\overset{(6.12)}{\le}r^{\frac{\log(\rho_{\epsilon_1}\rho_0^{-1})}{\log r}+1}\rho_0=r\rho_{\epsilon_1}.\tag{6.22}$$

In addition, notice that $\rho_k > 2\gamma$ and $\|\lambda^k\| \le \Lambda$. Using these, (4.1), the first relation in (4.6), and Lemma 6.4(ii) with $(x,\lambda,\rho) = (x^{k+1},\lambda^k,\rho_k)$, we obtain that

$$\begin{aligned} \|\tilde{c}(x^{k+1})\| &\le \sqrt{\frac{2(f_{\rm hi} - f_{\rm low} + \gamma)}{\rho_k - 2\gamma} + \frac{\|\lambda^k\|^2}{(\rho_k - 2\gamma)^2}} + \frac{\|\lambda^k\|}{\rho_k - 2\gamma} \\ &\le \sqrt{\frac{2(f_{\rm hi} - f_{\rm low} + \gamma)}{\rho_k - 2\gamma} + \frac{\Lambda^2}{(\rho_k - 2\gamma)^2}} + \frac{\Lambda}{\rho_k - 2\gamma}. \end{aligned} \tag{6.23}$$

Also, we observe from $\|c(z_{\epsilon_1})\| \le \epsilon_1/2$ and the definition of $\tilde{c}$ in (4.4) that

$$\|c(x^{k+1})\| \le \|\tilde{c}(x^{k+1})\| + \|c(z_{\epsilon_1})\| \le \|\tilde{c}(x^{k+1})\| + \epsilon_1/2. \tag{6.24}$$

We now prove that $\overline{K}_{\epsilon_1}$ is finite. Suppose for contradiction that $\overline{K}_{\epsilon_1}$ is infinite. It then follows from this and (4.15) that $\|c(x^{k+1})\| > \epsilon_1$ for all $k \ge K_{\epsilon_1}$, which along with (6.24) implies that $\|\tilde{c}(x^{k+1})\| > \epsilon_1/2$ for all $k \ge K_{\epsilon_1}$. It then follows that $\|\tilde{c}(x^{k+1})\| > \alpha\|\tilde{c}(x^k)\|$ must hold for infinitely many k's. Using this fact and the update scheme on $\{\rho_k\}$, we deduce that $\rho_{k+1} = r\rho_k$ holds for infinitely many k's, which together with the monotonicity of $\{\rho_k\}$ implies that $\rho_k \to \infty$ as $k \to \infty$. By this assertion and (6.23), one can see that $\|\tilde{c}(x^{k+1})\| \to 0$ as $k \to \infty$, which contradicts the fact that $\|\tilde{c}(x^{k+1})\| > \epsilon_1/2$ holds for all $k \ge K_{\epsilon_1}$. Hence, $\overline{K}_{\epsilon_1}$ is finite. In addition, notice from (4.11), (4.12), and (4.13) that $(\tau_k^g, \tau_k^H) = (\epsilon_1, \epsilon_2)$ for all $k \ge K_{\epsilon_1}$. This along with the termination criterion of Algorithm 4.1 and the definition of $\overline{K}_{\epsilon_1}$ implies that Algorithm 4.1 must terminate at iteration $\overline{K}_{\epsilon_1}$.

We next show that (4.16) and $\rho_k \le r\rho_{\epsilon_1}$ hold for $0 \le k \le \overline{K}_{\epsilon_1}$ by considering two separate cases.

Case 1. $\|c(x^{K_{\epsilon_1}+1})\| \le \epsilon_1$. By this and (4.15), one can see that $\overline{K}_{\epsilon_1} = K_{\epsilon_1}$, which together with (6.21) and (6.22) implies that (4.16) and $\rho_k \le r\rho_{\epsilon_1}$ hold for $0 \le k \le \overline{K}_{\epsilon_1}$.

Case 2. $\|c(x^{K_{\epsilon_1}+1})\| > \epsilon_1$. By this and (4.15), one can observe that $\overline{K}_{\epsilon_1} > K_{\epsilon_1}$ and also $\|c(x^{k+1})\| > \epsilon_1$ for all $K_{\epsilon_1} \le k \le \overline{K}_{\epsilon_1} - 1$, which together with (6.24) implies

$$\|\tilde{c}(x^{k+1})\| > \epsilon_1/2 \quad \forall K_{\epsilon_1} \le k \le \overline{K}_{\epsilon_1} - 1. \tag{6.25}$$

It then follows from $\|\lambda^k\| \le \Lambda$, (4.1), the first relation in (4.6), and Lemma 6.4(iv) with $(x,\lambda,\rho,\tilde{\delta}_c) = (x^{k+1},\lambda^k,\rho_k,\epsilon_1/2)$ that

$$\begin{aligned} \rho_k &< 8(f_{\rm hi} - f_{\rm low} + \gamma)\epsilon_1^{-2} + 4\|\lambda^k\|\epsilon_1^{-1} + 2\gamma \\ &\le 8(f_{\rm hi} - f_{\rm low} + \gamma)\epsilon_1^{-2} + 4\Lambda\epsilon_1^{-1} + 2\gamma \overset{(4.14)}{\le} \rho_{\epsilon_1} \quad \forall K_{\epsilon_1} \le k \le \overline{K}_{\epsilon_1} - 1. \end{aligned} \tag{6.26}$$

Combining this relation, (6.22), and the fact $\rho_{\overline{K}_{\epsilon_1}} \le r\rho_{\overline{K}_{\epsilon_1}-1}$, we conclude that $\rho_k \le r\rho_{\epsilon_1}$ holds for $0 \le k \le \overline{K}_{\epsilon_1}$. It remains to show that (4.16) holds. To this end, let $\mathbb{K} = \{k : \rho_{k+1} = r\rho_k, K_{\epsilon_1} \le k \le \overline{K}_{\epsilon_1} - 2\}$. It follows from (6.26) and the update scheme of ρ_k that $r^{|\mathbb{K}|}\rho_{K_{\epsilon_1}} = \max_{K_{\epsilon_1} \le k \le \overline{K}_{\epsilon_1}-1}\{\rho_k\} \le \rho_{\epsilon_1}$, which together with $\rho_{K_{\epsilon_1}} \ge \rho_0$ implies that

$$|\mathbb{K}| \le \log(\rho_{\epsilon_1}\rho_{K_{\epsilon_1}}^{-1})/\log r \le \log(\rho_{\epsilon_1}\rho_0^{-1})/\log r. \tag{6.27}$$

Let $\{k_1, k_2, \ldots, k_{|\mathbb{K}|}\}$ denote all the elements of $\mathbb{K}$ arranged in ascending order, and let $k_0 = K_{\epsilon_1}$ and $k_{|\mathbb{K}|+1} = \overline{K}_{\epsilon_1} - 1$. We next derive an upper bound for $k_{j+1} - k_j$ for $j = 0, 1, \ldots, |\mathbb{K}|$. By the definition of $\mathbb{K}$, one can observe that $\rho_k = \rho_{k'}$ for $k_j < k, k' \le k_{j+1}$. Using this and the update scheme of ρ_k, we deduce that

$$\|\tilde{c}(x^{k+1})\| \le \alpha\|\tilde{c}(x^k)\| \quad \forall k_j < k < k_{j+1}. \tag{6.28}$$

On the other hand, by (4.10), (6.23), and $\rho_k \geq \rho_0$, one has $\|\tilde{c}(x^{k+1})\| \leq \delta_{c,1}$ for $0 \leq k \leq \overline{K}_{\epsilon_1}$. By this and (6.25), one can see that

$$\epsilon_1/2 < \|\tilde{c}(x^{k+1})\| \leq \delta_{c,1} \quad \forall K_{\epsilon_1} \leq k \leq \overline{K}_{\epsilon_1} - 1. \tag{6.29}$$

Now, note that either $k_{j+1} - k_j = 1$ or $k_{j+1} - k_j > 1$. In the latter case, we can apply (6.28) with $k = k_{j+1} - 1, \ldots, k_j + 1$ together with (6.29) to deduce that

$$\epsilon_1/2 < \|\tilde{c}(x^{k_{j+1}})\| \leq \alpha \|\tilde{c}(x^{k_{j+1}-1})\| \leq \cdots \leq \alpha^{k_{j+1}-k_j-1}\|\tilde{c}(x^{k_j+1})\| \leq \alpha^{k_{j+1}-k_j-1}\delta_{c,1}$$

for all $j = 0, 1, \ldots, |\mathbb{K}|$. Combining these two cases, we have

$$k_{j+1} - k_j \leq |\log(\epsilon_1(2\delta_{c,1})^{-1})/\log\alpha| + 1 \quad \forall j = 0, 1, \ldots, |\mathbb{K}|. \tag{6.30}$$

Summing up these inequalities, and using (6.21), (6.27), $k_0 = K_{\epsilon_1}$, and $k_{|\mathbb{K}|+1} = \overline{K}_{\epsilon_1} - 1$, we have

$$\begin{aligned}
\overline{K}_{\epsilon_1} &= 1 + k_{|\mathbb{K}|+1} = 1 + k_0 + \sum_{j=0}^{|\mathbb{K}|}(k_{j+1} - k_j) \\
&\overset{(6.30)}{\leq} 1 + K_{\epsilon_1} + (|\mathbb{K}|+1)\left(\left|\frac{\log(\epsilon_1(2\delta_{c,1})^{-1})}{\log\alpha}\right| + 1\right) \\
&\leq 2 + \frac{\log(\rho_{\epsilon_1}\rho_0^{-1})}{\log r} + \left(\frac{\log(\rho_{\epsilon_1}\rho_0^{-1})}{\log r} + 1\right)\left(\left|\frac{\log(\epsilon_1(2\delta_{c,1})^{-1})}{\log\alpha}\right| + 1\right) \qquad (6.31) \\
&= 1 + \left(\frac{\log(\rho_{\epsilon_1}\rho_0^{-1})}{\log r} + 1\right)\left(\left|\frac{\log(\epsilon_1(2\delta_{c,1})^{-1})}{\log\alpha}\right| + 2\right),
\end{aligned}$$

where the second inequality is due to (6.21) and (6.27). Hence, (4.16) also holds in this case. ∎

We next prove Theorem 4.10. Before proceeding, we introduce some notation that will be used shortly. Let $L_{k,H}$ denote the Lipschitz constant of $\nabla^2_{xx}\widetilde{\mathcal{L}}(x, \lambda^k; \rho_k)$ on the convex open neighborhood $\Omega(\delta_f, \delta_c)$ of $\mathcal{S}(\delta_f, \delta_c)$, where $\mathcal{S}(\delta_f, \delta_c)$ is defined in (4.3), and let $U_{k,H} = \sup_{x\in\mathcal{S}(\delta_f,\delta_c)} \|\nabla^2_{xx}\widetilde{\mathcal{L}}(x, \lambda^k; \rho_k)\|$. Notice from (4.4) and (4.5) that

$$\nabla^2_{xx}\widetilde{\mathcal{L}}(x, \lambda^k; \rho_k) = \nabla^2 f(x) + \sum_{i=1}^m \lambda_i^k \nabla^2 c_i(x) + \rho_k\left(\nabla c(x)\nabla c(x)^T + \sum_{i=1}^m \tilde{c}_i(x)\nabla^2 c_i(x)\right). \tag{6.32}$$

By this, $\|\lambda^k\| \leq \Lambda$, the definition of $\tilde{c}$,, and the Lipschitz continuity of $\nabla^2 f$ and $\nabla^2 c_i$ (see Assumption 4.1(c)), one can observe that there exist some constants L_1, L_2, U_1, and U_2, depending only on f, c, Λ, δ_f, and δ_c, such that

$$L_{k,H} \leq L_1 + \rho_k L_2, \quad U_{k,H} \leq U_1 + \rho_k U_2. \tag{6.33}$$

Proof of Theorem 4.10. Let T_k and N_k denote the number of iterations and matrix-vector products, respectively, performed by Algorithm 3.1 at the outer iteration k of Algorithm 4.1. It then follows from Theorem 4.8 that the total number of iterations and matrix-vector products performed by Algorithm 3.1 in Algorithm 4.1 are $\sum_{k=0}^{\overline{K}_{\epsilon_1}} T_k$ and $\sum_{k=0}^{\overline{K}_{\epsilon_1}} N_k$, respectively. In addition, notice from (4.14) and Theorem 4.8 that $\rho_{\epsilon_1} = \mathcal{O}(\epsilon_1^{-2})$ and $\rho_k \leq r\rho_{\epsilon_1}$, which yield $\rho_k = \mathcal{O}(\epsilon_1^{-2})$.

We first claim that $(\tau_k^g)^2/\tau_k^H \geq \min\{\epsilon_1^2/\epsilon_2, \epsilon_2^3\}$ holds for any $k \geq 0$. Indeed, let $\bar{t} = \log \epsilon_1 / \log \omega_1$ and $\psi(t) = \max\{\epsilon_1, \omega_1^t\}^2 / \max\{\epsilon_2, \omega_2^t\}$ for all $t \in \mathbb{R}$. It then follows from (4.13) that $\omega_1^{\bar{t}} = \epsilon_1$ and $\omega_2^{\bar{t}} = \epsilon_2$. By this and $\omega_1, \omega_2 \in (0,1)$, one can observe that $\psi(t) = (\omega_1^2/\omega_2)^t$ if $t \leq \bar{t}$ and $\psi(t) = \epsilon_1^2/\epsilon_2$ otherwise. This along with $\epsilon_2 \in (0,1)$ implies that $\min_{t\in[0,\infty)} \psi(t) = \min\{\psi(0), \psi(\bar{t})\} = \min\{1, \epsilon_1^2/\epsilon_2\} \geq \min\{\epsilon_1^2/\epsilon_2, \epsilon_2^3\}$, which together with (4.11) yields $(\tau_k^g)^2/\tau_k^H = \psi(k) \geq \min\{\epsilon_1^2/\epsilon_2, \epsilon_2^3\}$ for all $k \geq 0$.

(i) From Lemma 4.4(i) and the definitions of $\Omega(\delta_f, \delta_c)$ and $L_{k,H}$, we see that $L_{k,H}$ is a Lipschitz constant of $\nabla_{xx}^2 \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k)$ on a convex open neighborhood of $\{x : \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k) \leq \widetilde{\mathcal{L}}(x_{\text{init}}^k, \lambda^k; \rho_k)\}$. Also, recall from Lemma 4.4(ii) that $\inf_{x\in\mathbb{R}^n} \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k) \geq f_{\text{low}} - \gamma - \Lambda\delta_c$. By these facts, $\widetilde{\mathcal{L}}(x_{\text{init}}^k, \lambda^k; \rho_k) \leq f_{\text{hi}}$ (see (4.9)), and Theorem 3.2(iii) with $(F_{\text{hi}}, F_{\text{low}}, L_H^F, \epsilon_g, \epsilon_H) = (\widetilde{\mathcal{L}}(x_{\text{init}}^k, \lambda^k; \rho_k), f_{\text{low}} - \gamma - \Lambda\delta_c, L_{k,H}, \tau_k^g, \tau_k^H)$, one has

$$\begin{aligned} T_k &= \mathcal{O}((f_{\text{hi}} - f_{\text{low}} + \gamma + \Lambda\delta_c) L_{k,H}^2 \max\{(\tau_k^g)^{-2}\tau_k^H, (\tau_k^H)^{-3}\}) \\ &\overset{(6.33)}{=} \mathcal{O}(\rho_k^2 \max\{(\tau_k^g)^{-2}\tau_k^H, (\tau_k^H)^{-3}\}) = \mathcal{O}(\epsilon_1^{-4} \max\{\epsilon_1^{-2}\epsilon_2, \epsilon_2^{-3}\}), \end{aligned} \tag{6.34}$$

where the last equality is from $(\tau_k^g)^2/\tau_k^H \geq \min\{\epsilon_1^2/\epsilon_2, \epsilon_2^3\}$, $\tau_k^H \geq \epsilon_2$, and $\rho_k = \mathcal{O}(\epsilon_1^{-2})$.

Next, if $c(x) = Ax - b$ for some $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, then $\nabla c(x) = A^T$ and $\nabla^2 c_i(x) = 0$ for $1 \leq i \leq m$. By these facts and (6.32), one has $L_{k,H} = \mathcal{O}(1)$. Using this and similar arguments as for (6.34), we obtain that $T_k = \mathcal{O}(\max\{\epsilon_1^{-2}\epsilon_2, \epsilon_2^{-3}\})$. By this result, (6.34), and $\overline{K}_{\epsilon_1} = \mathcal{O}(|\log \epsilon_1|^2)$ (see Remark 4.9), we conclude that statement (i) of Theorem 4.10 holds.

(ii) In view of Lemma 4.4(i) and the definition of $U_{k,H}$, one can see that $U_{k,H} \geq \sup_{x\in\mathbb{R}^n}\{\|\nabla_{xx}^2 \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k)\| : \widetilde{\mathcal{L}}(x, \lambda^k; \rho_k) \leq \widetilde{\mathcal{L}}(x_{\text{init}}^k, \lambda^k; \rho_k)\}$. Using this, $\widetilde{\mathcal{L}}(x_{\text{init}}^k, \lambda^k; \rho_k) \leq f_{\text{hi}}$, and Theorem 3.2(iv) with $(F_{\text{hi}}, F_{\text{low}}, L_H^F, U_H^F, \epsilon_g, \epsilon_H) = (\widetilde{\mathcal{L}}(x_{\text{init}}^k, \lambda^k; \rho_k), f_{\text{low}} - \gamma - \Lambda\delta_c, L_{k,H}, U_{k,H}, \tau_k^g, \tau_k^H)$, we obtain that

$$\begin{aligned} N_k &= \widetilde{\mathcal{O}}((f_{\text{hi}} - f_{\text{low}} + \gamma + \Lambda\delta_c) L_{k,H}^2 \max\{(\tau_k^g)^{-2}\tau_k^H, (\tau_k^H)^{-3}\} \min\{n, (U_{k,H}/\tau_k^H)^{1/2}\}) \\ &\overset{(6.33)}{=} \widetilde{\mathcal{O}}(\rho_k^2 \max\{(\tau_k^g)^{-2}\tau_k^H, (\tau_k^H)^{-3}\} \min\{n, (\rho_k/\tau_k^H)^{1/2}\}) \\ &= \widetilde{\mathcal{O}}(\epsilon_1^{-4} \max\{\epsilon_1^{-2}\epsilon_2, \epsilon_2^{-3}\} \min\{n, \epsilon_1^{-1}\epsilon_2^{-1/2}\}), \end{aligned} \tag{6.35}$$

where the last equality is from $(\tau_k^g)^2/\tau_k^H \geq \min\{\epsilon_1^2/\epsilon_2, \epsilon_2^3\}$, $\tau_k^H \geq \epsilon_2$, and $\rho_k = \mathcal{O}(\epsilon_1^{-2})$.

On the other hand, if c is assumed to be affine, it follows from the above discussion that $L_{k,H} = \mathcal{O}(1)$. Using this, $U_{k,H} \leq U_1 + \rho_k U_2$, and similar arguments as for (6.35), we obtain that $N_k = \widetilde{\mathcal{O}}(\max\{\epsilon_1^{-2}\epsilon_2, \epsilon_2^{-3}\} \min\{n, \epsilon_1^{-1}\epsilon_2^{-1/2}\})$. By this, (6.35), and $\overline{K}_{\epsilon_1} = \mathcal{O}(|\log \epsilon_1|^2)$ (see Remark 4.9), we conclude that statement (ii) of Theorem 4.10 holds.
$\square$

Next, we provide a proof of Theorem 4.14. To proceed, we first observe from Assumptions 4.1(c) and 4.12 that there exist $U_g^f > 0$, $U_g^c > 0$, and $\sigma > 0$ such that

$$\|\nabla f(x)\| \leq U_g^f, \quad \|\nabla c(x)\| \leq U_g^c, \quad \lambda_{\min}(\nabla c(x)^T \nabla c(x)) \geq \sigma^2 \quad \forall x \in \mathcal{S}(\delta_f, \delta_c). \tag{6.36}$$

We next establish several technical lemmas that will be used shortly.

LEMMA 6.5. *Suppose that Assumptions* 4.1 *and* 4.12 *hold and that ρ_0 is sufficiently large such that $\delta_{f,1} \leq \delta_f$ and $\delta_{c,1} \leq \delta_c$, where $\delta_{f,1}$ and $\delta_{c,1}$ are defined as in* (4.10)*. Let $\{(x^k, \lambda^k, \rho_k)\}$ be generated by Algorithm* 4.1*. Suppose that*

$$\rho_k \geq \max\{\Lambda^2(2\delta_f)^{-1}, 2(f_{\text{hi}} - f_{\text{low}} + \gamma)\delta_c^{-2} + 2\Lambda\delta_c^{-1} + 2\gamma, 2(U_g^f + U_g^c\Lambda + 1)(\sigma\epsilon_1)^{-1}\} \tag{6.37}$$

for some $k \geq 0$, where γ, f_{hi}, f_{low}, δ_f, and δ_c are given in Assumption 4.1*, and U_g^f, U_g^c, and σ are given in* (6.36)*. Then it holds that $\|c(x^{k+1})\| \leq \epsilon_1$.*

Proof. By (6.37) and $\|\lambda^k\| \le \Lambda$ (see step 6 of Algorithm 4.1), one can see that $\rho_k \ge \max\{\|\lambda^k\|^2(2\delta_f)^{-1}, 2(f_{\text{hi}}-f_{\text{low}}+\gamma)\delta_c^{-2}+2\|\lambda^k\|\delta_c^{-1}+2\gamma\}$. Using this, (4.1), the first relation in (4.6), and Lemma 6.4(iii) and (iv) with $(x,\lambda,\rho,\tilde{\delta}_f,\tilde{\delta}_c)=(x^{k+1},\lambda^k,\rho_k,\delta_f,\delta_c)$, we obtain that $f(x^{k+1}) \le f_{\text{hi}}+\delta_f$ and $\|\tilde{c}(x^{k+1})\| \le \delta_c$. In addition, recall from $\|c(z_{\epsilon_1})\| \le 1$ and the definition of $\tilde{c}$ in (4.4) that $\|c(x^{k+1})\| \le 1+\|\tilde{c}(x^{k+1})\|$. These together with (4.3) show that $x^{k+1} \in \mathcal{S}(\delta_f,\delta_c)$. It then follows from (6.36) that $\|\nabla f(x^{k+1})\| \le U_g^f$, $\|\nabla c(x^{k+1})\| \le U_g^c$, and $\lambda_{\min}(\nabla c(x^{k+1})^T\nabla c(x^{k+1})) \ge \sigma^2$. By $\|\nabla f(x^{k+1})\| \le U_g^f$, $\|\nabla c(x^{k+1})\| \le U_g^c$, $\tau_k^g \le 1$, $\|\lambda^k\| \le \Lambda$, (4.4), and (4.6), one has

$$\rho_k\|\nabla c(x^{k+1})\tilde{c}(x^{k+1})\| \le \|\nabla f(x^{k+1})+\nabla c(x^{k+1})\lambda^k\|+\|\nabla_x\widetilde{\mathcal{L}}(x^{k+1},\lambda^k;\rho_k)\|$$

$$\overset{(4.6)}{\le} \|\nabla f(x^{k+1})\|+\|\nabla c(x^{k+1})\|\|\lambda^k\|+\tau_k^g \le U_g^f+U_g^c\Lambda+1. \tag{6.38}$$

In addition, note that $\lambda_{\min}(\nabla c(x^{k+1})^T\nabla c(x^{k+1})) \ge \sigma^2$ implies that $\nabla c(x^{k+1})^T\nabla c(x^{k+1})$ is invertible. Using this fact and [6.38], we obtain

$$\|\tilde{c}(x^{k+1})\| \le \|(\nabla c(x^{k+1})^T\nabla c(x^{k+1}))^{-1}\nabla c(x^{k+1})^T\|\|\nabla c(x^{k+1})\tilde{c}(x^{k+1})\|$$

$$= \lambda_{\min}(\nabla c(x^{k+1})^T\nabla c(x^{k+1}))^{-\frac{1}{2}}\|\nabla c(x^{k+1})\tilde{c}(x^{k+1})\| \overset{(6.38)}{\le} \frac{U_g^f+U_g^c\Lambda+1}{\sigma\rho_k}. \tag{6.39}$$

We also observe from (6.37) that $\rho_k \ge 2(U_g^f+U_g^c\Lambda+1)(\sigma\epsilon_1)^{-1}$, which along with (6.39) proves $\|\tilde{c}(x^{k+1})\| \le \epsilon_1/2$. Combining this with the definition of $\tilde{c}$ in (4.4) and $\|c(z_{\epsilon_1})\| \le \epsilon_1/2$, we conclude that $\|c(x^{k+1})\| \le \epsilon_1$ holds as desired. ☐

The next lemma provides a stronger upper bound for $\{\rho_k\}$ than the one in Theorem 4.8.

LEMMA 6.6. *Suppose that Assumptions* 4.1 *and* 4.12 *hold and that* ρ_0 *is sufficiently large such that* $\delta_{f,1} \le \delta_f$ *and* $\delta_{c,1} \le \delta_c$*, where* $\delta_{f,1}$ *and* $\delta_{c,1}$ *are defined as in* (4.10)*. Let* $\{\rho_k\}$ *be generated by Algorithm* 4.1 *and*

$$\tilde{\rho}_{\epsilon_1} := \max\{\Lambda^2(2\delta_f)^{-1}, 2(f_{\text{hi}}-f_{\text{low}}+\gamma)\delta_c^{-2}+2\Lambda\delta_c^{-1}+2\gamma, 2(U_g^f+U_g^c\Lambda+1)(\sigma\epsilon_1)^{-1}, 2\rho_0\}, \tag{6.40}$$

where γ*,* f_{hi}*,* f_{low}*,* δ_f*, and* δ_c *are given in Assumption* 4.1*, and* U_g^f*,* U_g^c*, and* σ *are given in* (6.36)*. Then* $\rho_k \le r\tilde{\rho}_{\epsilon_1}$ *holds for* $0 \le k \le \overline{K}_{\epsilon_1}$*, where* $\overline{K}_{\epsilon_1}$ *is defined in* (4.15)*.*

Proof. It follows from (6.40) that $\tilde{\rho}_{\epsilon_1} \ge 2\rho_0$. By this and similar arguments as for (6.21), one has $K_{\epsilon_1} \le \log(\tilde{\rho}_{\epsilon_1}\rho_0^{-1})/\log r+1$, where K_{ϵ_1} is defined in (4.12). Using this, the update scheme for $\{\rho_k\}$, and similar arguments as for (6.22), we obtain

$$\max_{0\le k\le K_{\epsilon_1}} \rho_k \le r\tilde{\rho}_{\epsilon_1}. \tag{6.41}$$

If $\|c(x^{K_{\epsilon_1}+1})\| \le \epsilon_1$, it follows from (4.15) that $\overline{K}_{\epsilon_1}=K_{\epsilon_1}$, which together with (6.41) implies that $\rho_k \le r\tilde{\rho}_{\epsilon_1}$ holds for $0 \le k \le \overline{K}_{\epsilon_1}$. On the other hand, if $\|c(x^{K_{\epsilon_1}+1})\| > \epsilon_1$, it follows from (4.15) that $\|c(x^{k+1})\| > \epsilon_1$ for $K_{\epsilon_1} \le k \le \overline{K}_{\epsilon_1}-1$. This together with Lemma 6.5 and (6.40) implies that for all $K_{\epsilon_1} \le k \le \overline{K}_{\epsilon_1}-1$,

$$\rho_k < \max\{\Lambda^2(2\delta_f)^{-1}, 2(f_{\text{hi}}-f_{\text{low}}+\gamma)\delta_c^{-2}+2\Lambda\delta_c^{-1}+2\gamma, 2(U_g^f+U_g^c\Lambda+1)(\sigma\epsilon_1)^{-1}\} \overset{(6.40)}{\le} \tilde{\rho}_{\epsilon_1}.$$

By this, [6.41], and $\rho_{\overline{K}_{\epsilon_1}} \le r\rho_{\overline{K}_{\epsilon_1}-1}$, we also see that $\rho_k \le r\tilde{\rho}_{\epsilon_1}$ holds for $0 \le k \le \overline{K}_{\epsilon_1}$. ☐

Proof of Theorem 4.14. Notice from (6.40) and Lemma 6.6 that $\tilde{\rho}_{\epsilon_1} = \mathcal{O}(\epsilon_1^{-1})$ and $\rho_k \le r\tilde{\rho}_{\epsilon_1}$, which yield $\rho_k = \mathcal{O}(\epsilon_1^{-1})$. The conclusion of Theorem 4.14 then follows from this and the same arguments as for the proof of Theorem 4.10 with $\rho_k = \mathcal{O}(\epsilon_1^{-2})$ replaced by $\rho_k = \mathcal{O}(\epsilon_1^{-1})$. ◻

7. Future work.

There are several possible future studies on this work. First, it would be interesting to extend our AL method to seek an approximate SOSP of nonconvex optimization with inequality or more general constraints. Indeed, for nonconvex optimization with inequality constraints, one can reformulate it as an equality constrained problem using squared slack variables (e.g., see [7]). It can be shown that an SOSP of the latter problem induces a weak SOSP of the original problem and also the linear independence constraint qualification holds for the latter problem if it holds for the original problem. As a result, it is promising to find an approximate weak SOSP of an inequality constrained problem by applying our AL method to the equivalent equality constrained problem. Second, it is worth studying whether the enhanced complexity results in section 4.3 can be derived under weaker constraint qualification (e.g., see [5]). Third, the development of our AL method is based on a strong assumption that a nearly feasible solution of the problem is known. It would make the method applicable to a broader class of problems if such an assumption could be removed by modifying the method possibly through the use of infeasibility detection techniques (e.g., see [19]). Lastly, more numerical studies would be helpful to further improve our AL method from a practical perspective.

Appendix A. A capped conjugate gradient method.

In this part we present the capped CG method proposed in [56, Algorithm 1] for finding either an approximate solution to the linear system (3.6) or a sufficiently negative curvature direction of the associated matrix H, which has been briefly discussed in section 3.1. The details can be found in [56, section 3.1].

The following theorem presents the iteration complexity of Algorithm A.1.

THEOREM A.1 (iteration complexity of Algorithm A.1). *Consider applying Algorithm* A.1 *with input $U = 0$ to the linear system* (3.6) *with $g \neq 0$, $\varepsilon > 0$, and H being an $n \times n$ symmetric matrix. Then the number of iterations of Algorithm* A.1 *is* $\widetilde{\mathcal{O}}(\min\{n, \sqrt{\|H\|/\varepsilon}\})$.

Proof. From [56, Lemma 1], we know that the number of iterations of Algorithm A.1 is bounded by $\min\{n, J(U,\varepsilon,\zeta)\}$, where $J(U,\varepsilon,\zeta)$ is the smallest integer J such that $\sqrt{T}\tau^{J/2} \le \widehat{\zeta}$, with $U, \widehat{\zeta}, T$, and τ being the values returned by Algorithm A.1. In addition, it was shown in [56, section 3.1] that $J(U,\varepsilon,\zeta) \le \left\lceil \left(\sqrt{\kappa} + \frac{1}{2}\right) \ln\left(\frac{144(\sqrt{\kappa}+1)^2\kappa^6}{\zeta^2}\right)\right\rceil$, where $\kappa = \mathcal{O}(U/\varepsilon)$ is an output by Algorithm A.1. Then one can see that $J(U,\varepsilon,\zeta) = \widetilde{\mathcal{O}}(\sqrt{U/\varepsilon})$. Notice from Algorithm A.1 that the output $U \le \|H\|$. Combining these results, we obtain the conclusion as desired. ◻

Appendix B. A randomized Lanczos based minimum eigenvalue oracle.

Here we present the randomized Lanczos method proposed in [56, section 3.2], which can be used as a minimum eigenvalue oracle for Algorithm 3.1. As briefly discussed in section 3.1, this oracle outputs either a sufficiently negative curvature direction of H or a certificate that H is nearly positive semidefinite with high probability. More detailed motivation and explanation of the oracle can be found in [56, section 3.2].

The following theorem justifies that Algorithm B.1 is a suitable minimum eigenvalue oracle for Algorithm 3.1. Its proof is identical to that of [56, Lemma 2] and thus omitted.

Algorithm A.1 A capped conjugate gradient method.

```
Inputs: symmetric matrix $H \in \mathbb{R}^{n\times n}$, vector $g \neq 0$, damping parameter $\varepsilon \in (0,1)$,
desired relative accuracy $\zeta \in (0,1)$.
Optional input: scalar $U \geq 0$ (set to 0 if not provided).
Outputs: d_type, $d$.
Secondary outputs: final values of $U, \kappa, \widehat{\zeta}, \tau$, and $T$.
Set
    $\bar{H} := H + 2\varepsilon I, \quad \kappa := \frac{U+2\varepsilon}{\varepsilon}, \quad \widehat{\zeta} := \frac{\zeta}{3\kappa}, \quad \tau := \frac{\sqrt{\kappa}}{\sqrt{\kappa}+1}, \quad T := \frac{4\kappa^4}{(1-\sqrt{\tau})^2},$
$y^0 \leftarrow 0, r^0 \leftarrow g, p^0 \leftarrow -g, j \leftarrow 0.$
if $(p^0)^T \bar{H} p^0 < \varepsilon \|p^0\|^2$ then
    Set $d \leftarrow p^0$ and terminate with d_type=NC;
else if $\|Hp^0\| > U\|p^0\|$ then
    Set $U \leftarrow \|Hp^0\|/\|p^0\|$ and update $\kappa, \widehat{\zeta}, \tau, T$ accordingly;
end if
while TRUE do
    $\alpha_j \leftarrow (r^j)^T r^j/(p^j)^T \bar{H} p^j$; {Begin Standard CG Operations}
    $y^{j+1} \leftarrow y^j + \alpha_j p^j$;
    $r^{j+1} \leftarrow r^j + \alpha_j \bar{H} p^j$;
    $\beta_{j+1} \leftarrow \|r^{j+1}\|^2/\|r^j\|^2$;
    $p^{j+1} \leftarrow -r^{j+1} + \beta_{j+1} p^j$; {End Standard CG Operations}
    $j \leftarrow j+1$;
    if $\|Hp^j\| > U\|p^j\|$ then
        Set $U \leftarrow \|Hp^j\|/\|p^j\|$ and update $\kappa, \widehat{\zeta}, \tau, T$ accordingly;
    end if
    if $\|Hy^j\| > U\|y^j\|$ then
        Set $U \leftarrow \|Hy^j\|/\|y^j\|$ and update $\kappa, \widehat{\zeta}, \tau, T$ accordingly;
    end if
    if $\|Hr^j\| > U\|r^j\|$ then
        Set $U \leftarrow \|Hr^j\|/\|r^j\|$ and update $\kappa, \widehat{\zeta}, \tau, T$ accordingly;
    end if
    if $(y^j)^T \bar{H} y^j < \varepsilon \|y^j\|^2$ then
        Set $d \leftarrow y^j$ and terminate with d_type=NC;
    else if $\|r^j\| \leq \widehat{\zeta}\|r^0\|$ then
        Set $d \leftarrow y^j$ and terminate with d_type=SOL;
    else if $(p^j)^T \bar{H} p^j < \varepsilon \|p^j\|^2$ then
        Set $d \leftarrow p^j$ and terminate with d_type=NC;
    else if $\|r^j\| > \sqrt{T}\tau^{j/2}\|r^0\|$ then
        Compute $\alpha_j, y^{j+1}$ as in the main loop above;
        Find $i \in \{0,\ldots,j-1\}$ such that
            $(y^{j+1}-y^i)^T \bar{H}(y^{j+1}-y^i) < \varepsilon \|y^{j+1}-y^i\|^2;$
        Set $d \leftarrow y^{j+1} - y^i$ and terminate with d_type=NC;
    end if
end while
```

Algorithm B.1 A randomized Lanczos based minimum eigenvalue oracle.

Input: symmetric matrix $H \in \mathbb{R}^{n\times n}$, tolerance $\varepsilon > 0$, and probability parameter $\delta \in (0,1)$.

Output: a sufficiently negative curvature direction v satisfying $v^T Hv \le -\varepsilon/2$ and $\|v\| = 1$; or a certificate that $\lambda_{\min}(H) \ge -\varepsilon$ with probability at least $1-\delta$.

Apply the Lanczos method [44] to estimate $\lambda_{\min}(H)$ starting with a random vector uniformly generated on the unit sphere, and run it for at most

$$N(\varepsilon,\delta) := \min\left\{ n, 1 + \left\lceil \frac{\ln(2.75n/\delta^2)}{2}\sqrt{\frac{\|H\|}{\varepsilon}} \right\rceil \right\} \tag{B.1}$$

iterations. If a unit vector v with $v^T Hv \le -\varepsilon/2$ is found at some iteration, terminate immediately and return v.

THEOREM B.1 (iteration complexity of Algorithm B.1). *Consider Algorithm* B.1 *with tolerance $\varepsilon > 0$, probability parameter $\delta \in (0,1)$, and symmetric matrix $H \in \mathbb{R}^{n\times n}$ as its input. Then it either finds a sufficiently negative curvature direction v satisfying $v^T Hv \le -\varepsilon/2$ and $\|v\| = 1$ or certifies that $\lambda_{\min}(H) \ge -\varepsilon$ holds with probability at least $1-\delta$ in at most $N(\varepsilon,\delta)$ iterations, where $N(\varepsilon,\delta)$ is defined in* (B.1).

Notice that $\|H\|$ is required in Algorithm B.1. In general, computing $\|H\|$ may not be cheap when n is large. Nevertheless, $\|H\|$ can be efficiently estimated via a randomization scheme with high confidence (e.g., see the discussion in [56, Appendix B3]).

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, 2017, pp. 1195–1199.

[2] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt, *On augmented Lagrangian methods with general lower-level constraints*, SIAM J. Optim., 18 (2007), pp. 1286–1309, https://doi.org/10.1137/060654797.

[3] R. Andreani, G. Haeser, and J. M. Martínez, *On sequential optimality conditions for smooth constrained optimization*, Optimization, 60 (2011), pp. 627–641.

[4] R. Andreani, G. Haeser, A. Ramos, and P. J. Silva, *A second-order sequential optimality condition associated to the convergence of optimization algorithms*, IMA J. Numer. Anal., 37 (2017), pp. 1902–1929.

[5] R. Andreani, G. Haeser, M. L. Schuverdt, and P. J. S. Silva, *Two new weak constraint qualifications and applications*, SIAM J. Optim., 22 (2012), pp. 1109–1135, https://doi.org/10.1137/110843939.

[6] P. Armand and N. N. Tran, *An augmented Lagrangians method for equality constrained optimization with rapid infeasibility detection capabilities*, J. Optim. Theory Appl., 181 (2019), pp. 197–215.

[7] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999.

[8] W. Bian, X. Chen, and Y. Ye, *Complexity analysis of interior point algorithms for non-Lipschitz and nonconvex minimization*, Math. Program., 149 (2015), pp. 301–327.

[9] E. G. Birgin, J. Gardenghi, J. M. Martínez, S. A. Santos, and Ph. L. Toint, *Evaluation complexity for nonlinear constrained optimization using unscaled KKT conditions and high-order models*, SIAM J. Optim., 26 (2016), pp. 951–967, https://doi.org/10.1137/15M1031631.

[10] E. G. Birgin, G. Haeser, and A. Ramos, *Augmented Lagrangians with constrained subproblems and convergence to second-order stationary points*, Comput. Optim. Appl., 69 (2018), pp. 51–75.

[11] E. G. Birgin and J. M. Martínez, *Practical Augmented Lagrangian Methods for Constrained Optimization*, SIAM, 2014, https://doi.org/10.1137/1.9781611973365.

[12] 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, https://doi.org/10.1137/16M110280X.

[13] E. G. Birgin and J. M. Martínez, *Complexity and performance of an augmented Lagrangian algorithm*, Optim. Methods Softw., 35 (2020), pp. 885–920.

[14] J. F. Bonnans and G. Launay, *Sequential quadratic programming with penalization of the displacement*, SIAM J. Optim., 5 (1995), pp. 792–812, https://doi.org/10.1137/0805038.

[15] N. Boumal, V. Voroninski, and A. S. Bandeira, *The non-convex Burer-Monteiro approach works on smooth semidefinite programs*, in Advances in Neural Information Processing Systems 29, Curran Associates, 2016, pp. 2757–2765.

[16] L. F. Bueno and J. M. Martínez, *On the complexity of an inexact restoration method for constrained optimization*, SIAM J. Optim., 30 (2020), pp. 80–101, https://doi.org/10.1137/18M1216146.

[17] S. Burer and R. D. C. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Math. Program., 95 (2003), pp. 329–357.

[18] S. Burer and R. D. C. Monteiro, *Local minima and convergence in low-rank semidefinite programming*, Math. Program., 103 (2005), pp. 427–444.

[19] J. V. Burke, F. E. Curtis, and H. Wang, *A sequential quadratic optimization algorithm with rapid infeasibility detection*, SIAM J. Optim., 24 (2014), pp. 839–872, https://doi.org/10.1137/120880045.

[20] R. H. Byrd, F. E. Curtis, and J. Nocedal, *Infeasibility detection and SQP methods for nonlinear optimization*, SIAM J. Optim., 20 (2010), pp. 2281–2299, https://doi.org/10.1137/080738222.

[21] R. H. Byrd, R. B. Schnabel, and G. A. Shultz, *A trust region algorithm for nonlinearly constrained optimization*, SIAM J. Numer. Anal., 24 (1987), pp. 1152–1170, https://doi.org/10.1137/0724076.

[22] Y. Carmon and J. Duchi, *Gradient descent finds the cubic-regularized nonconvex Newton step*, SIAM J. Optim., 29 (2019), pp. 2146–2178, https://doi.org/10.1137/17M1113898.

[23] 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 International Conference on Machine Learning, PMLR, 2017, pp. 654–663.

[24] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, *Accelerated methods for nonconvex optimization*, SIAM J. Optim., 28 (2018), pp. 1751–1772, https://doi.org/10.1137/17M1114296.

[25] 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.

[26] C. Cartis, N. I. M. Gould, and Ph. L. Toint, *On the evaluation complexity of cubic regularization methods for potentially rank-deficient nonlinear least-squares problems and its relevance to constrained nonlinear optimization*, SIAM J. Optim., 23 (2013), pp. 1553–1574, https://doi.org/10.1137/120869687.

[27] C. Cartis, N. I. M. Gould, and P. L. Toint, *On the complexity of finding first-order critical points in constrained nonlinear optimization*, Math. Program., 144 (2014), pp. 93–106.

[28] C. Cartis, N. I. M. Gould, and Ph. L. Toint, *On the evaluation complexity of constrained nonlinear least-squares and general constrained nonlinear optimization using second-order methods*, SIAM J. Numer. Anal., 53 (2015), pp. 836–851, https://doi.org/10.1137/130915546.

[29] C. Cartis, N. I. M. Gould, and P. L. Toint, *Evaluation complexity bounds for smooth constrained nonlinear optimization using scaled KKT conditions and high-order models,* in Approximation and Optimization: Algorithms, Complexity and Applications, Springer, 2019, pp. 5–26.

[30] C. Cartis, N. I. M. Gould, and Ph. L. Toint, *Optimality of orders one to three and beyond: Characterization and evaluation complexity in constrained nonconvex optimization*, J. Complexity, 53 (2019), pp. 68–94.

[31] X. Chen, L. Guo, Z. Lu, and J. J. Ye, *An augmented Lagrangian method for non-Lipschitz nonconvex programming*, SIAM J. Numer. Anal., 55 (2017), pp. 168–193, https://doi.org/10.1137/15M1052834.

[32] D. Cifuentes and A. Moitra, *Polynomial Time Guarantees for the Burer-Monteiro Method*, preprint, arXiv:1912.01745, 2019.

[33] T. F. Coleman, J. Liu, and W. Yuan, *A new trust-region algorithm for equality constrained optimization*, Comput. Optim. Appl., 21 (2002), pp. 177–199.

[34] F. E. Curtis, D. P. Robinson, C. W. Royer, and S. J. Wright, *Trust-region Newton-CG with strong second-order complexity guarantees for nonconvex optimization*, SIAM J. Optim., 31 (2021), pp. 518–544, https://doi.org/10.1137/19M130563X.

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

[36] F. E. Curtis, D. P. Robinson, and M. Samadi, *Complexity analysis of a trust funnel algorithm for equality constrained optimization*, SIAM J. Optim., 28 (2018), pp. 1533–1563, https://doi.org/10.1137/16M1108650.

[37] G. N. Grapiglia and Y. Yuan, *On the complexity of an augmented Lagrangian method for nonconvex optimization*, IMA J. Numer. Anal., 41 (2021), pp. 1546–1568.

[38] G. Haeser, H. Liu, and Y. Ye, *Optimality condition and complexity analysis for linearly-constrained optimization without differentiability on the boundary*, Math. Program., (2019), pp. 1–37.

[39] M. R. Hestenes, *Multiplier and gradient methods*, J. Optim. Theory Appl., 4 (1969), pp. 303–320.

[40] M. Hong, D. Hajinezhad, and M.-M. Zhao, *Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks*, in International Conference on Machine Learning, PMLR, 2017, pp. 1529–1538.

[41] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, *How to escape saddle points efficiently*, in International Conference on Machine Learning, PMLR, 2017, pp. 1724–1732.

[42] C. Kanzow and D. Steck, *An example comparing the standard and safeguarded augmented Lanczos methods*, Oper. Res. Lett., 45 (2017), pp. 598–603.

[43] W. Kong, J. G. Melo, and R. D. C. Monteiro, *Complexity of a quadratic penalty accelerated inexact proximal point method for solving linearly constrained nonconvex composite programs*, SIAM J. Optim., 29 (2019), pp. 2566–2593, https://doi.org/10.1137/18M1171011.

[44] 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, https://doi.org/10.1137/0613066.

[45] Z. Li, P.-Y. Chen, S. Liu, S. Lu, and Y. Xu, *Rate-improved inexact augmented Lagrangian method for constrained nonconvex optimization*, in International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 2170–2178.

[46] S. Lu, *A single-loop gradient descent and perturbed ascent algorithm for nonconvex functional constrained optimization*, in International Conference on Machine Learning, PMLR, 2022, pp. 14315–14357.

[47] S. Lu, M. Razaviyayn, B. Yang, K. Huang, and M. Hong, *Finding second-order stationary points efficiently in smooth nonconvex linearly constrained optimization problems*, in Advances in Neural Information Processing Systems 33, Curran Associates, 2020, pp. 2811–2822.

[48] Z. Lu and X. Li, *Sparse recovery via partial regularization: Models, theory, and algorithms*, Math. Oper. Res., 43 (2018), pp. 1290–1316.

[49] Z. Lu and Y. Zhang, *An augmented Lagrangian approach for sparse principal component analysis*, Math. Program., 135 (2012), pp. 149–193.

[50] 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.

[51] J. G. Melo, R. D. Monteiro, and W. Kong, *Iteration-Complexity of an Inner Accelerated Inexact Proximal Augmented Lagrangian Method Based on the Classical Lagrangian Function and a Full Lagrange Multiplier Update*, preprint, arXiv:2008.00562, 2020.

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

[53] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer, 2006.

[54] M. O'Neill and S. J. Wright, *A log-barrier Newton-CG method for bound constrained optimization with complexity guarantees*, IMA J. Numer. Anal., 41 (2021), pp. 84–121.

[55] R. T. Rockafellar, *Lagrange multipliers and optimality*, SIAM Rev., 35 (1993), pp. 183–238, https://doi.org/10.1137/1035044.

[56] C. W. Royer, M. O'Neill, and S. J. Wright, *A Newton-CG algorithm with complexity guarantees for smooth unconstrained optimization*, Math. Program., 180 (2020), pp. 451–488.

[57] 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, https://doi.org/10.1137/17M1134329.

[58] M. F. Sahin, A. Eftekhari, A. Alacaoglu, F. Latorre, and V. Cevher, *An inexact augmented Lagrangian framework for nonconvex optimization with nonlinear constraints*,

in Advances in Neural Information Processing Systems 32, Curran Associates, 2019, pp. 13943–13955.

[59] Y. Xie and S. J. Wright, *Complexity of Projected Newton Methods for Bound-Constrained Optimization*, preprint, arXiv:2103.15989, 2021.

[60] Y. Xie and S. J. Wright, *Complexity of proximal augmented Lagrangian for nonconvex optimization with nonlinear equality constraints*, J. Sci. Comput., 86 (2021), pp. 1–30.

[61] L. Yang, D. Sun, and K.-C. Toh, *SDPNAL+: A majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints*, Math. Program. Comput., 7 (2015), pp. 331–366.

[62] X.-Y. Zhao, D. Sun, and K.-C. Toh, *A Newton-CG augmented Lagrangian method for semidefinite programming*, SIAM J. Optim., 20 (2010), pp. 1737–1765, https://doi.org/10.1137/080718206.