An accelerated first-order method with complexity analysis for solving cubic regularization subproblems

Rujun Jiang* Man-Chung Yue† Zhishuo Zhou‡

Abstract

We propose a first-order method to solve the cubic regularization subproblem (CRS) based on a novel reformulation. The reformulation is a constrained convex optimization problem whose feasible region admits an easily computable projection. Our reformulation requires computing the minimum eigenvalue of the Hessian. To avoid the expensive computation of the exact minimum eigenvalue, we develop a surrogate problem to the reformulation where the exact minimum eigenvalue is replaced with an approximate one. We then apply first-order methods such as the Nesterov's accelerated projected gradient method (APG) and projected Barzilai-Borwein method to solve the surrogate problem. As our main theoretical contribution, we show that when an ϵ-approximate minimum eigenvalue is computed by the Lanczos method and the surrogate problem is approximately solved by APG, our approach returns an ϵ-approximate solution to CRS in $\tilde{O}(\epsilon^{-1/2})$ matrix-vector multiplications (where $\tilde{O}(\cdot)$ hides the logarithmic factors). Numerical experiments show that our methods are comparable to and outperform the Krylov subspace method in the easy and hard cases, respectively. We further implement our methods as subproblem solvers of adaptive cubic regularization methods, and numerical results show that our algorithms are comparable to the state-of-the-art algorithms.

1 Introduction

Motivated by applications in machine learning and signal processing, optimization problems of the following form have attracted significant attention:

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

*School of Data Science, Fudan University, Shanghai, China, rjjiang@fudan.edu.cn

†Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong, manchung.yue@polyu.edu.hk

‡School of Data Science, Fudan University, Shanghai, China, zhouzs18@fudan.edu.cn

where F is a twice continuously differentiable function that is possibly non-convex. The cubic regularization method [19, 9] is among the most successful algorithms for solving problem (1). At each iteration of the cubic regularization method, the subproblem takes the form

$$\min_{x\in\mathbb{R}^n} f_1(x) := \frac{1}{2}x^T Ax + b^T x + \frac{\rho}{3}\|x\|^3, \tag{CRS}$$

where $\|\cdot\|$ denotes the Euclidean norm, A is an $n \times n$ symmetric matrix (not necessarily positive semidefinite) and ρ is a regularization parameter. In particular, A and b represent the Hessian and gradient of the function F at the current iterate, respectively. It was first proved by Nesterov and Polyak [19] that the cubic regularization method enjoys an iteration complexity of $O(\epsilon^{-3/2})$ if each subproblem is solved exactly. Cartis et al. [9] developed a generalization of the cubic regularization method, called ARC, which allows the subproblems to be solved inexactly and the regularization parameter $\rho > 0$ to be chosen adaptively. In the same paper, they showed that the iteration complexity of ARC is again $O(\epsilon^{-3/2})$. Complementing to these global complexity results, Yue et al. [25] showed that the cubic regularization method enjoys a local quadratic convergence rate under an error bound-type condition.

Despite the above strong theoretical guarantees, the practical performance of the cubic regularization method depends critically on the efficiency of solving its subproblems. As such, there have been considerable endeavors on developing fast algorithms for solving (CRS). One of the most successful algorithms for solving large-scale instances of (CRS) in practice is the Krylov subspace method [9]. Carmon and Duchi [7] provided the first the convergence rate analysis of the Krylov subspace method. In particular, they showed that the Krylov subspace method achieves an ϵ-approximate optimal solution in $O(\epsilon^{-1/2})$ or $O(\sqrt{\kappa}\log\epsilon^{-1})$ operations (matrix-vector multiplications) in the easy case[1], where κ is the condition number of (CRS). Unfortunately, the Krylov subspace method may fail to converge to the optimal solution when the problem (CRS) is in the hard case or close to being in the hard case [7]. Carmon and Duchi also showed in another paper [6] that the gradient descent method is able to converge to the global minimizer if the step size is sufficiently small, and the convergence rate is $\tilde{O}(\epsilon^{-1})$ (where $\tilde{O}(\cdot)$ hides the logarithmic factors). Although, for the problem (CRS), the convergence rate of the gradient descent method is worse than that of the Krylov subspace method, it works in both the easy and hard cases. On the other hand, based on the cubic regularization method, Agarwal et al. [1] derived an algorithm with $\tilde{O}(\epsilon^{-7/4})$ operations for finding an approximate local minimum of problem (1), i.e., a point $x \in \mathbb{R}^n$ satisfying

$$\|\nabla F(x)\| \le \epsilon \quad \text{and} \quad \nabla^2 F(x) + \sqrt{\epsilon} I \succeq 0,$$

[1]For the problem (CRS), it is said to be in the easy if the optimal solution x^* satisfies $\rho\|x^*\| > -\lambda_1$, where λ_1 is the minimum eigenvalue of A, and hard case otherwise.

where I denotes the identity matrix of appropriate dimension and, for any symmetric matrix M, the inequality $M \succeq 0$ means that M is positive semidefinite. A key component of their result is an algorithm for computing an approximate solution to the problem (CRS) in $\tilde{O}(\epsilon^{-1/4})$ operations. However, the approximate solution returned by this algorithm is not an ϵ-approximate global minimizer of the problem (CRS) in the traditional sense (see [1, Theorem 2] for details). Furthermore, the algorithm in [1] for solving (CRS) requires sophisticated parameter tuning, and no numerical results had been provided in the paper. Finally, a Newton-like method for solving problems of the form (1) had been recently developed by Birgin and Martínez [5]. Each subproblem of their algorithm, which is similar to but not the same as (CRS), is constructed and can be efficiently solved by using the so-called mixed factorization (see [5, Section 2] for details) of the (approximate) Hessian of F at the current point. Birgin and Martínez [5] advocated in particular the mixed factorization obtained from the Bunch-Parlett-Kaufman factorization [13], a matrix factorization whose computational cost is similar to that of the Cholesky factorization.

From the above discussion, it is desirable to have an algorithm for solving the problem (CRS) that works efficiently in practice for both the hard and easy cases and enjoys theoretical guarantees. In this paper, we achieve this goal by developing a first-order method for solving arbitrary instances of (CRS) with $\tilde{O}(\epsilon^{-1/2})$ matrix-vector multiplications. Our approach is based on a novel reformulation of the problem (CRS), which is a constrained convex optimization problem built using the minimum eigenvalue of the matrix A. The feasible region of the reformulation admits an efficient, closed-form projection. Therefore, when the exact computation of the minimum eigenvalue is viable, we can apply any algorithm for solving constrained convex optimization problems to solve the reformulation to global optimality. The optimal solution to the problem (CRS) can then be constructed by using the optimal solution of the reformulation. In practice, it is often prohibitively expensive to compute the exact minimum eigenvalue of the matrix A, if not impossible. We circumvent this limitation by developing a surrogate problem to the reformulation. The surrogate problem is again a constrained convex optimization problem with an easily computable projection onto its feasible region. More importantly, the surrogate problem requires only an approximate minimum eigenvalue, which can be computed efficiently by using, e.g., the Lanczos method [13]. Similarly, an ϵ-approximate optimal solution of the problem (CRS) can be constructed from an ϵ-approximate solution of the surrogate problem.

The said bound $\tilde{O}(\epsilon^{-1/2})$ on the number of operations is proved by combining the follow two ideas. First, for any $\delta \in (0, 1)$, the Lanczos method returns an ϵ-approximate minimum eigenvalue in $O(\epsilon^{-1/2} \log(n/\delta))$ matrix-vector multiplications with probability at least $1 - \delta$. Second, solving the surrogate problem by the Nesterov's accelerated projected gradient descent method [20, 3] (APG) requires $O(\epsilon^{-1/2})$ iterations, where each iteration consists of

one gradient and Hessian evaluations and one matrix-vector multiplication. Therefore, the total number of operations of our method is bounded by $O(\epsilon^{-1/2}\log(n/\delta))$ (see Theorem 3.4). This bound is similar to the sublinear bound for the Krylov subspace method proved in [7] in the easy case and better than that of the gradient descent method in [6]. Note also that our bound is for the subproblem and hence not directly comparable with that of [1]. Besides, our algorithm has the advantage that it is easily implementable. Furthermore, as we shall see in our numerical section, the proposed algorithm works efficiently in practice for high-dimensional problems—our algorithm shows a comparable performance to the Krylov subspace method in the easy case. An another advantage of our algorithm is that, unlike the Krylov subspace method, it works in both the easy and hard cases. This saves us from the computational overhead due to the need of detecting the hard case.

We remark that our approach is inspired by the recent line of research [15, 24] on linear-time algorithms for the trust region subproblem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \frac{1}{2}x^TAx + b^Tx \\ \text{subject to} \quad & \|x\|^2 \le 1, \end{aligned} \tag{TRS}$$

and the close resemblance between the problems (CRS) and (TRS). More specifically, the algorithms in [15, 24] are based on a convex reformulation for the (TRS) derived in [11]. Motivated by the works [15, 24], Jiang and Li [17] recently derived a novel convex reformulation for the generalized trust region subproblem, which further inspires us to explore hidden convexity for (CRS) in this paper. It should also be pointed out that our reformulation and its surrogate problem offer great potential and flexibility for the design of fast algorithms to solve the problem (CRS). Indeed, one can apply any algorithm for constrained convex optimization problems to solve these two optimization problems. Proving theoretical guarantees for other algorithms for solving these two models is left as a future research.

The remaining of this paper is organized as follows. In Section 2, we derive our convex reformulation based on the minimum eigenvalue of matrix A and discuss the computation of the projection to its feasible region. In Section 3, we present a surrogate problem for (CRS) and theoretically analyze the complexity of our method when applying the APG to solve the surrogate problem with an approximate minimum eigenvalue computed by the Lanczos method. In Section 4, we first compare the numerical performance of our methods with the Krylov subspace method and then compare our methods against others as a subproblem solver for ARC. We conclude our paper in Section 5.

2 Convex reformulation

We first record the optimality condition of (CRS) [19, 9], which is given by the following system of equations in x and λ:

$$Ax + b + \lambda x = 0, \quad A + \lambda I \succeq 0, \quad \text{and} \quad \lambda = \rho\|x\|. \tag{2}$$

This optimality condition will be frequently used in this paper. It is obvious that (CRS) is equivalent to the following problem:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n, y\in\mathbb{R}} & \frac{1}{2}x^TAx + b^Tx + \frac{\rho}{3}y^{\frac{3}{2}} \\ \text{subject to} & \|x\|^2 \le y. \end{array} \tag{RP}$$

Note that the feasible region $\{(x, y) \in \mathbb{R}^n \times \mathbb{R} : \|x\|^2 \le y\}$ of the problem (RP) is convex. Therefore, when $A \succeq 0$, (RP) is a convex optimization problem and can be solved efficiently by various methods, e.g., APG or projected Barzilai-Borwein method (BBM) [2, 23]. Hence, from now on, we assume that the minimum eigenvalue of matrix A, denoted by λ_1, is negative, i.e., $\lambda_1 < 0$. Consider the optimization problem

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n, y\in\mathbb{R}} & f_2(x, y) := \frac{1}{2}x^T(A - \lambda_1 I)x + b^Tx + \frac{\rho}{3}y^{\frac{3}{2}} + \frac{\lambda_1}{2}y \\ \text{subject to} & \|x\|^2 \le y. \end{array} \tag{CP}$$

Problem (CP) is a convex problem because f_2 is separable in x and y and is convex in each of these two variables. The following theorem shows that problem (CRS) is equivalent to problem (CP).

Theorem 2.1. *Problem* (CRS) *is equivalent to* (CP) *in the following sense. First, the two problems have the same optimal value. Second, if x^* is an optimal solution to* (CRS)*, then $(x^*, \|x^*\|^2)$ is an optimal solution to* (CP)*. Third, if $(\tilde{x}, \tilde{y})$ is an optimal solution to* (CP)*, then an optimal solution to* (CRS) *is given by*

$$\hat{x} = \begin{cases} \tilde{x} & \text{if } \|\tilde{x}\|^2 = \tilde{y}, \\ \tilde{x} + \zeta v & \text{if } \|\tilde{x}\|^2 < \tilde{y}, \end{cases}$$

where ζ is a root of the quadratic equation $\|\tilde{x} + \zeta v\|^2 = \tilde{y}$ and v is an eigenvector associated with λ_1.

Proof. Denote by Val(CRS) and Val(CP) the optimal values of problems (CRS) and (CP), respectively. We first observe that (CP) is a convex problem and satisfies the Slater condition. Assume that x^* is an optimal solution to (CRS). By using the optimality condition (2), we

can easily show that the triplet $(x, y, \mu) = (x^*, \|x^*\|^2, \frac{1}{2}(\rho\|x^*\| + \lambda_1))$ satisfies the KKT system of (CP):

$$(A - \lambda_1 I)x + b + 2\mu x = 0 \text{ and } \frac{\rho}{2}y^{\frac{1}{2}} + \frac{\lambda_1}{2} - \mu = 0 \tag{3}$$

This implies that $(x^*, \|x^*\|^2)$ is an optimal solution to (CP) and that Val(CRS) $\geq$ Val(CP). On the other hand, because of the assumption $\lambda_1 < 0$ and the constraint $\|x\|^2 \leq y$, we have that Val(CRS) $\leq$ Val(CP). Therefore, Val(CRS) = Val(CP). This completes the proof of the first and second claims.

To prove the third claim, assume that (CP) has an optimal solution $(\tilde{x}, \tilde{y})$. Suppose μ is a Lagrangian multiplier associated with the constraint in (CP). If $\|\tilde{x}\|^2 = \tilde{y}$, from the KKT system (3), we have that

$$A\tilde{x} - \lambda_1\tilde{x} + b + 2\mu\tilde{x} = 0 \tag{4}$$

and

$$\frac{1}{2}\rho\sqrt{\tilde{y}} + \frac{1}{2}\lambda_1 - \mu = 0. \tag{5}$$

Equation (5) implies $\mu = \rho\sqrt{\tilde{y}}/2 + \lambda_1/2 \geq 0$. This, together with $\|\tilde{x}\|^2 = \tilde{y}$ and $A\tilde{x} - \lambda_1\tilde{x} + b + 2\mu\tilde{x} = 0$, implies that $A\tilde{x} + b + \bar{\lambda}\tilde{x} = 0$ and $A + \bar{\lambda}I \succeq 0$, for $\bar{\lambda} = 2\mu - \lambda_1 = \rho\|\tilde{x}\|$. Hence, due to (2), $\tilde{x}$ is also optimal for (CRS) and the objective values of (CRS) and (CP) are the same due to $\|\tilde{x}\|^2 = \tilde{y}$.

Next, we consider the case of $\|\tilde{x}\|^2 < \tilde{y}$. Let v be an eigenvector of matrix A associated with the minimum eigenvalue λ_1. By complementary slackness, $\mu = 0$. Then, equation (4) implies that $b^T v = 0$. Hence, there exists ζ such that $\|\tilde{x} + \zeta v\| = \sqrt{\tilde{y}}$ and $(\tilde{x} + \zeta v, \tilde{y})$ is still a solution to (CP). Using the same argument for the case of $\|\tilde{x}\|^2 = \tilde{y}$, we can show that $\tilde{x} + \zeta v$ is an optimal solution for (CRS). This completes the proof. □

Optimization problems of the form

$$\min_{x\in\mathbb{R}^n, y\in\mathbb{R}} g(x, y) + h(x, y), \tag{6}$$

where g is a smooth convex function and h is a non-smooth convex function, are called convex composite minimization problems. Letting $S = \{(x, y) : \|x\|^2 \leq y\}$, problem (CP) can be written as a convex composite minimization problem:

$$\min_{x\in\mathbb{R}^n, y\in\mathbb{R}} f_2(x, y) + \iota_S(x, y),$$

where ι_S is the indicator function

$$\iota_S(x, y) := \begin{cases} 0, & \text{if } (x, y) \in S, \\ +\infty, & \text{otherwise.} \end{cases}$$

General convex composite minimization problems (6) can be solved by many different algorithms such as APG, BBM, proximal quasi-Newton methods [12] and proximal Newton methods [26]. In order to apply these methods, we need to efficiently compute the proximal mapping with respect to the non-smooth function h in (6). In our situation, $h = \iota_S$ and hence the proximal mapping reduces to the orthogonal projection $\Pi_S(x, y)$ onto the closed convex set S, i.e.,

$$\Pi_S(x, y) = \operatorname*{argmin}_{(x',y')\in S} \|(x', y') - (x, y)\|^2.$$

The following theorem shows that such a projection can be done in $O(n)$ time.

Theorem 2.2. *For any point $(x_0, y_0) \in \mathbb{R}^n \times \mathbb{R}$, the projection $\Pi_S(x_0, y_0)$ is given by*

$$\Pi_S(x_0, y_0) = \begin{cases} (x_0, y_0), & \text{if } \|x_0\|^2 \le y_0, \\ \left(\dfrac{x_0}{1+\mu^*}, y_0 + \dfrac{\mu^*}{2}\right), & \text{otherwise,} \end{cases} \tag{7}$$

where μ^* is the unique solution in the interval $[\max\{0, -2y_0\}, \infty)$ of the univariate cubic equation

$$\frac{1}{2}\mu^3 + (y_0 + 1)\mu^2 + (2y_0 + \frac{1}{2})\mu - x_0^T x_0 + y_0 = 0. \tag{8}$$

Proof. The case of $x_0^T x_0 \le y_0$ is trivial. So, we consider the case that $x_0^T x_0 > y_0$. The projection is defined as the solution to the (strongly convex) optimization problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n, y\in\mathbb{R}} \quad & \|(x, y) - (x_0, y_0)\|^2 \\ \text{subject to} \quad & \|x\|^2 \le y. \end{aligned} \tag{9}$$

The KKT optimality condition of problem (9) can be written as

$$2(x - x_0) + 2\mu x = 0, \tag{10}$$
$$2y - 2y_0 - \mu = 0, \tag{11}$$
$$\mu(\|x\|^2 - y) = 0,$$
$$\|x\|^2 \le y,$$
$$\mu \ge 0.$$

We have $x = \frac{x_0}{1+\mu}$ and $y = y_0 + \frac{\mu}{2}$ from (10) and (11), respectively. Suppose that $\mu = 0$. The optimality condition reduces to $x = x_0$ and $y = y_0$, which contradicts to the constraint $\|x\|^2 > y$ of problem (9). Therefore, we have $\mu > 0$ and hence $\|x\|^2 = y$ by complementary slackness. This leads to the univariate cubic equation

$$\left(\frac{x_0}{1+\mu}\right)^T \frac{x_0}{1+\mu} = y_0 + \frac{\mu}{2},$$

which is equivalent to (8) and implies, in particular, that $2y_0 + \mu \geq 0$. Define

$$h(\mu) = \frac{1}{2}\mu^3 + (y_0+1)\mu^2 + \left(2y_0 + \frac{1}{2}\right)\mu - x_0^T x_0 + y_0.$$

Since $2y_0 + \mu \geq 0$ and $\mu \geq 0$, the derivative h' satisfies

$$h'(\mu) = \frac{3}{2}\mu^2 + 2(y_0+1)\mu + \left(2y_0 + \frac{1}{2}\right) = \frac{1}{2}\mu^2 + (2y_0+\mu)\mu + (2y_0 + 2\mu) + \frac{1}{2} \geq \frac{1}{2}.$$

Hence $h(\mu)$ is strictly increasing on $[\max\{0, -2y_0\}, \infty)$. Observing that $h(0) = y_0 - x_0^T x_0 < 0$, $h(-2y_0) = -x_0^T x_0 < 0$ and $h(+\infty) = +\infty$, there exists exactly one root in the interval $[\max\{0, -2y_0\}, \infty)$. Denote the solution of equation $h(\mu) = 0$ in this interval by μ^*. Then, we have

$$x = \frac{x_0}{1+\mu^*} \quad \text{and} \quad y = y_0 + \frac{\mu^*}{2},$$

which completes the proof. □

In practice, to find a root of the cubic equation (8) in the interval $[\max\{0, -2y_0\}, \infty)$, we use a hybrid method obtained by combining the bisection method and the Newton's method. Numerically, our hybrid method is faster and more stable than the function `roots` in MATLAB. The projection can be done in runtime $O(n)$ as formulating the cubic equation cost $O(n)$ and solving the univariate cubic equation costs $O(1)$.

3 Complexity to achieve an ϵ-optimal solution of (CRS)

3.1 Another Equivalent Convex Reformulation

To achieve a theoretical complexity for solving convex composite optimization problem (6) with first-order methods such as APG [20], the function g is often required to have a Lipschitz continuous gradient on its domain $\mathrm{dom}(g)$, i.e., there exists a constant $L > 0$ such that

$$\|\nabla g(x) - \nabla g(y)\| \leq L\|x - y\|, \ \forall x, y \in \mathrm{dom}(g).$$

However, one can easily check that the gradient ∇f_2 of the objective f_2 of (CP) is not Lipschitz continuous at those points (x, y) with $y = 0$. To remedy this, instead of (CP), we consider the following problem, which ensures y is bounded below from 0 by imposing an extra constrain $y \geq l$:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n, y\in\mathbb{R}} & f_2(x,y) \\ \text{subject to} & \|x\|^2 \leq y, \ y \geq l, \end{array} \tag{BCP}$$

where $l = \lambda_1^2/\rho^2$. To justify the choice of the lower bound l in (BCP), we note that the function $\frac{\rho}{3}y^{\frac{3}{2}} + \frac{\lambda_1}{2}y$ is decreasing when $\sqrt{y} \leq -\lambda_1/\rho$. Therefore, any optimal solution $(\tilde{x}, \tilde{y})$

of (CP) must satisfy $\tilde{y} \geq (-\lambda_1/\rho)^2 = l$, and hence problem (BCP) has the same objective value and optimal solutions as problem (CP).

Problem (BCP) is again in the form of a convex composite minimization problem (6). Denote by $B = \{(x, y) \in \mathbb{R}^n \times \mathbb{R} : \|x\|^2 \leq y, y \geq l\}$ the feasible region of problem (BCP). The next theorem shows that the projection Π_B onto the feasible region B is again easily computable.

Theorem 3.1. *For any point $(x_0, y_0) \in \mathbb{R}^n \times \mathbb{R}$, the projection $\Pi_B(x_0, y_0)$ is given by*

$$\Pi_B(x_0, y_0) = \begin{cases} (x_1, y_1) & \text{if } y_1 \geq l, \\ (x_0, l) & \text{if } y_1 < l \text{ and } \|x_0\| < \sqrt{l}, \\ (\sqrt{l}x_0/\|x_0\|, l) & \text{otherwise,} \end{cases}$$

where $(x_1, y_1) = \Pi_S(x_0, y_0)$.

Proof. Let (x_2, y_2) be the projection of (x_0, y_0) onto B. If $y_1 \geq l$, then $(x_1, y_1) = \Pi_S(x_0, y_0)$ is the solution to the problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n, y \in \mathbb{R}} \quad & \|(x, y) - (x_0, y_0)\|^2 \\ \text{subject to} \quad & \|x\|^2 \leq y, \ y \geq l. \end{aligned}$$

Next, we consider the case of $y_1 < l$. In this case, we must have $y_2 = l$ since otherwise (x_2, y_2) is also the projection of (x_0, y_0) onto S, which contradicts with $y_1 < l$. Hence, x_2 is actually the solution to the problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \|x - x_0\|^2 \\ \text{subject to} \quad & \|x\|^2 \leq l. \end{aligned}$$

We thus have the following two implications: if $\|x_0\| < \sqrt{l}$, then $x_2 = x_0$; and if $\|x_0\| \geq \sqrt{l}$, then $x_2 = \sqrt{l}x_0/\|x_0\|$. This completes the proof. □

For Theorem 3.1, the projection onto B is as cheap as the projection onto S because the former costs at most two more scalar comparisons, which are negligible, than the latter (note that $\|x_0\|$ is already computed in the computation of the projection onto S).

3.2 A Surrogate Problem

When the dimension n is high, the exact computation of the minimum eigenvalue is prohibitively expensive, if not impossible. For computational efficiency, an approximate eigenvalue is preferred when only an approximate solution of (CRS) is needed, which is often the

case in practice. When an approximate minimum eigenvalue $\theta \approx \lambda_1$ is used in the problem (BCP), the objective $\frac{1}{2}x^T(A-\theta I)x + b^T x + \frac{\rho}{3}y^{\frac{3}{2}} + \frac{\theta}{2}y$ could be non-convex. Therefore, we need to slightly modify the problem (BCP). Let the approximate minimum eigenvalue θ satisfies $\lambda_1 \le \theta \le \lambda_1 + \epsilon$ and define $\eta := -\theta + \epsilon + \lambda_1 \ge 0$. Noting that $-\theta + \epsilon = -\lambda_1 + \eta$ (we will frequently use this equality in subsequent analysis), we obtain the following problem as a surrogate problem to (CRS):

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n, y\in\mathbb{R}} & f_3(x,y) := \dfrac{1}{2}x^T\left(A + (-\theta+\epsilon)I\right)x + b^T x + \dfrac{\rho}{3}y^{\frac{3}{2}} - \dfrac{-\theta+\epsilon}{2}y \\ \text{subject to} & \|x\|^2 \le y,\ y \ge \hat{l}, \end{array} \tag{SP}$$

where $\hat{l} = (-\theta+\epsilon)^2/\rho^2$. To justify the lower bound $\hat{l}$ for y, we note that $\frac{\rho}{3}y^{\frac{3}{2}} - \frac{-\theta+\epsilon}{2}y$ is decreasing when $y \le \hat{l}$, and hence $\hat{l}$ is a lower bound for any optimal y. From now on, we denote by $\hat{B} := \{(x,y) : \|x\|^2 \le y,\ y \ge \hat{l}\}$ and (x^η, y^η) the feasible region and an optimal solution to (SP), respectively. By Theorem 3.1, the feasible region $\hat{B}$ admits an easily computable projection.

Our theoretical convergence rate of solving problem (CRS) is based on the surrogate problem (SP). Specifically, we shall specialize the backtracking line search version of APG [3] to problem (SP) (see Algorithm 1) and show in Theorem 3.4 below that the sequence of iterates converges sublinearly to an optimal solution of problem (BCP) (which is also an optimal solution to problem (CP)). In view of Theorem 2.1, a convergence rate for solving (CRS) is thus obtained. It should be pointed out that, unlike the original APG, we reset the final solution returned by APG (in Lines 8–12 of Algorithm 1) to achieve an equal or smaller objective value (see the proof in Theorem 3.4).

Remark: If we directly use the approximate minimum eigenvalue θ to replace the exact minimum eigenvalue λ_1 in (BCP), we get the following problem:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n, y\in\mathbb{R}} & \dfrac{1}{2}x^T(A-\theta I)x + b^T x + \dfrac{\rho}{3}y^{\frac{3}{2}} + \dfrac{\theta}{2}y \\ \text{subject to} & \|x\|^2 \le y,\ y \ge l, \end{array} \tag{AP}$$

In Appendix A, we show that solving (AP) yields an approximate optimal solution to (CRS) if ϵ is sufficiently small, i.e., the eigenvalue computation is sufficiently accurate. We also show in Appendix A that either all the stationary points, which are approximate optimal solutions of (AP), share the same objective value, or there is a unique stationary point that is the optimal solution of (AP) if $-\theta > \bar{\lambda}$, where $\bar{\lambda}$ is some constant such that $\bar{\lambda} < -\lambda_1$. Note that when $\epsilon \le -\lambda_1 - \bar{\lambda}$, we always have that $\theta < \lambda_1 + \epsilon < -\bar{\lambda}$ and hence that $-\theta > \bar{\lambda}$. However, the constant $\bar{\lambda}$ is unknown a priori and hence our formulation (AP) may have a non-optimal stationary point if we choose a θ that is not close enough to λ_1. This is why we focus on (SP) in this paper. Nevertheless, we will compare the empirical performance between (SP) and (AP) in the numerical section.

Algorithm 1 APG for (SP)

Input: f_3, ∇f_3, $L_0 > 0, \xi > 1$, $\epsilon > 0$, $\theta < 0$, $x_0 \in \mathbb{R}^n$ and $y_0 \in \mathbb{R}$.

choose $\beta_1 = \alpha_0 = (x_0^T, y_0)^T$ and $t_1 = 1$

for $k = 1, 2, ..., k_{\max}$ **do**

find the smallest non-negative integer i_k such that $\bar{L} = \xi^{i_k} L_{k-1}$ and

$$f_3(\alpha_k) \geq f_3(\beta_k) + \nabla f_3(\beta_k)^T(\alpha_k - \beta_k) + \frac{\bar{L}}{2}\|\alpha_k - \beta_k\|^2,$$

where $\alpha_k = \Pi_{\hat{B}}(\beta_k - \frac{1}{\bar{L}}\nabla f_3(\beta_k))$

set $L_k = \xi^{i_k} L_{k-1}$

compute $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$

compute $\beta_{k+1} = \alpha_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(\alpha_k - \alpha_{k-1})$

end for

if $\alpha_k(n+1) > \|\alpha_k(1:n)\|^2$ and $\sqrt{\alpha_k(n+1)} > (-\theta + \epsilon)/\rho$ **then**

set $x_k = \alpha_k(1:n)$ and $y_k = \max\{\|\alpha_k(1:n)\|^2, (-\theta + \epsilon)^2/\rho^2\}$

else

set $(x_k^T, y_k)^T = \alpha_k$

end if

3.2.1 Approximate Computation of Eigenpairs

To obtain an approximate eigenpair, we recall the Lanczos method for approximately finding the minimum eigenvalue and its associated eigenvector [13]. The Lanczos method achieves a fast complexity bound for eigenvalue computation [18] and is an important component for proving complexity bounds for non-convex unconstrained optimization in the literature [1, 8, 22]. The specific result on the Lanczos method we need is the following lemma.

Lemma 3.2 ([18] and Lemma 9 in [22])**.** *Let H be a symmetric matrix satisfying $\|H\|_2 \leq U_H$ for some $U_H > 0$, where $\|\cdot\|_2$ denotes the operator 2-norm of a matrix, and λ_1 its minimum eingenvalue. Suppose that the Lanczos procedure is applied to find the largest eigenvalue of $U_H I - H$ starting at a random vector distributed uniformly over the unit sphere. Then, for any $\epsilon > 0$ and $\delta \in (0, 1)$, there is a probability at least $1 - \delta$ that the procedure outputs a unit vector v such that $v^T H v \leq \lambda_1 + \epsilon$ in at most* $\min\left\{n, \frac{\log(n/\delta^2)}{2\sqrt{2}}\sqrt{\frac{U_H}{\epsilon}}\right\}$ *iterations.*

3.2.2 Convergence Rate of APG for (SP)

We first collect some basic properties of APG.

Lemma 3.3 ([20, 3]). *Consider a function $G(x) = g(x) + h(x)$, where g is continuously differentiable, convex function with the gradient ∇g being L-Lipschitz continuous on its domain $\mathrm{dom}(g)$ and h is a proper, closed, and convex function that can possibly be non-smooth. Let $\{x_k\}_{k=1}^{\infty}$ be the sequence generated by APG. Then, we have*

$$G(x_k) - G^* \leq \frac{2\xi L\|x^* - x_0\|^2}{(k+1)^2},$$

where x^* is an optimal solution and G^* is the optimal value of $G(x)$. Equivalently, in order to guarantee $G(x_k) - G^* \leq \epsilon$, we need at most $k = \sqrt{2\xi L}\|x^* - x_0\|\epsilon^{-1/2} - 1$ iterations.

Restricting the objective function f_3 in (SP) to the set $\hat{B}$, the gradient ∇f_3 is then γ-Lipschitz continuous, where

$$\gamma = \max\left\{\|A + (\epsilon - \theta)I\|_2, \frac{\rho}{4\sqrt{\hat{l}}}\right\}. \tag{12}$$

Applying Lemma 3.3 to problem (SP) with $g = f_3$ and $h = \iota_{\hat{B}}$, we obtain that

$$f_3(x_k, y_k) - f(x^\eta, y^\eta) \leq \epsilon$$

after at most $k = \sqrt{2\xi\gamma}\sqrt{\|x^\eta - x_0\|^2 + (y^\eta - y_0)^2}\epsilon^{-1/2} - 1$ iterations.

The next theorem shows that with probability at least $1 - \delta$, our algorithm returns an ϵ-approximate optimal solution to problem (CRS) using at most $O(\epsilon^{-1/2}\log(n/\delta))$ operations (including those in the approximate eigenpiar computation and the APG).

Theorem 3.4. *Let X^* be the optimal solution set, (x^η, y^η) be any optimal solution to problem* (SP)*, $R = \inf_{(x,y)\in X^*}\|(x, y) - (x_0, y_0)\|$ the initial distance to the optimal solution, (x^*, y^*) an optimal solution to problem* (BCP) *with $\|x^*\|^2 = y^*$ (which always exists) and (x_k, y_k) the solution returned by Algorithm 1, where $k \geq \sqrt{2\xi\gamma}R\epsilon^{-1/2} - 1$ and γ is as defined in* (12)*. Define*

$$\tilde{x} = \begin{cases} x_k, & \text{if } \|x_k\|^2 = y_k, \\ x_k + tv, & \text{otherwise,} \end{cases}$$

where v is an approximate eigenvector that satisfies $v^T Av \leq \lambda_1 + \epsilon$ and $\|v\| = 1$, and t is chosen such that $t(v^T Ax_k + b^T v + (-\lambda_1 + \eta)x_k^T v) \leq 0$ and $\|x_k + tv\|^2 = y_k$ (which also always exists). Then, we have

$$f_1(\tilde{x}) - f_1(x^*) \leq \epsilon + (-\lambda_1 + \eta)^2\epsilon/\rho^2 = O(\epsilon),$$

where f_1 is the objective function in (CRS)*. Furthermore, when the approximate eigenpair is computed by the Lanczos method, the output is correct with probability at least $1 - \delta$ and the total number of matrix-vector products is at most*

$$\sqrt{2\xi\gamma}R\epsilon^{-1/2} - 1 + \frac{\log(n/\delta^2)}{2\sqrt{2}}\sqrt{\frac{\|A\|_2}{\epsilon}} = O(\epsilon^{-1/2}\log(n/\delta)).$$

Proof. Recall that f_2 and f_3 are the objective functions of (BCP) and (SP), respectively. For any optimal solution x^* of (CRS), $(x^*, \|x^*\|^2)$ is an optimal solution of (BCP). Therefore, an optimal solution (x^*, y^*) satisfying $\|x^*\|^2 = y^*$ always exists. Let $E_k = f_3(x_k, y_k) - f_3(x^\eta, y^\eta)$. From Lemma 3.3, we obtain that $f_3(\alpha_k) - f_3(x^\eta, y^\eta) < \epsilon$. If $\alpha_k(n+1) > \|\alpha_k(1:n)\|^2$ and $\sqrt{\alpha_k(n+1)} > (-\lambda_1 + \eta)/\rho$, we then go to Line 8 and Algorithm 1 outputs (x_k, y_k) instead of α_k. The y-part of the objective function f_3, i.e.,

$$\frac{\rho}{3} y^{\frac{3}{2}} - \frac{-\theta + \epsilon}{2} y,$$

is increasing when $\sqrt{y} \geq (-\theta + \epsilon)/\rho$, and hence Line 8 outputs a solution whose objective value is at most $f_3(\alpha_k)$. Hence $E_k \leq \epsilon$ for all $k \geq \sqrt{2\xi\gamma} R \epsilon^{-1/2} - 1$. Using this, we have

$$\begin{aligned} & f_3(x_k, y_k) - f_2(x^*, y^*) \\ =& f_3(x_k, y_k) - f_3(x^\eta, y^\eta) + f_3(x^\eta, y^\eta) - f_3(x^*, y^*) + f_3(x^*, y^*) - f_2(x^*, y^*) \\ \leq& E_k + 0 + \frac{\eta}{2}(\|x^*\|^2 - y^*) \\ =& E_k, \end{aligned} \tag{13}$$

where the inequality follows from the fact $f_3(x^\eta, y^\eta) - f_3(x^*, y^*) \leq 0$ because (x^η, y^η) is an optimal solution to (SP) and the last equality from the fact that $\|x^*\|^2 = y^*$.

If $\|x_k\|^2 = y_k$, we have that $\tilde{x} = x_k$ and hence that $f_3(x_k, y_k) = f_1(\tilde{x})$. Substituting $f_3(x_k, y_k) = f_1(\tilde{x})$ to (13) and noting that $f_1(x^*) = f_2(x^*, y^*)$, we have that $f_1(\tilde{x}) - f_1(x^*) \leq \epsilon$. If $\|x_k\|^2 < y_k$, we have $\tilde{x} = x_k + tv$ with $\|\tilde{x}\|^2 = y_k$ and hence

$$\begin{aligned} & f_1(\tilde{x}) - f_3(x_k, y_k) \\ =& \frac{1}{2}(x_k + tv)^T A(x_k + tv) + b^T(x_k + tv) + \frac{\rho}{3}\|(x_k + tv)\|^3 \\ & - \left(\frac{1}{2} x_k^T A x_k + b^T x_k + \frac{\rho}{3} y_k^{3/2} + \frac{-\lambda_1 + \eta}{2}(\|x_k\|^2 - y_k) \right) \\ =& tv^T A x_k + \frac{t^2}{2} v^T A v + t b^T v - \frac{-\lambda_1 + \eta}{2}(\|x_k\|^2 - \|x_k + tv\|^2) \\ =& t(v^T A x_k + b^T v) + \frac{t^2}{2}(\lambda_1 + \epsilon - \eta) - \frac{-\lambda_1 + \eta}{2}(-2t x_k^T v - t^2) \\ =& t(v^T A x_k + b^T v + (-\lambda_1 + \eta) x_k^T v) + \epsilon t^2/2 \\ \leq& \epsilon t^2/2, \end{aligned} \tag{14}$$

where the third equality follows from $v^T A v = \theta = \lambda_1 + \epsilon - \eta$ and the inequality from $t(v^T A x_k + b^T v + (-\lambda_1 + \eta) x_k^T v) \leq 0$. Note that a constant t satisfying such an inequality always exists. Indeed, since $\|x_k\|^2 < y_k$, the equation $\|x_k + tv\|^2 = y_k$ (in t) have two roots of opposite signs. Hence, we can always choose a t such that $t(v^T A x_k + b^T v + (-\lambda_1 + \eta) x_k^T v) \leq 0$. Using the

inequalities (13), (14) and the fact that $f_1(x^*) = f_2(x^*, y^*)$, we get $f_1(\tilde{x}) - f_1(x^*) \leq \epsilon + \epsilon t^2/2$. Also, $\|x_k + tv\|^2 = y_k$ implies that $t \leq \|x_k\| + \sqrt{y_k} \leq 2\left(\frac{-\lambda_1+\eta}{\rho}\right)$. Thus, we have

$$f_1(\tilde{x}) - f_1(x^*) \leq \epsilon + 2\epsilon\left(\frac{-\lambda_1+\eta}{\rho}\right)^2 \leq \epsilon + 2\left(\frac{-\lambda_1+\epsilon}{\rho}\right)^2\epsilon,$$

where the last inequality follows from $0 \leq \eta := -\theta + \lambda_1 + \epsilon \leq \epsilon$.

From Lemma 3.2, with probability at least $1-\delta$, such θ and v can be computed in at most $\frac{\log(n/\delta^2)}{2\sqrt{2}}\sqrt{\frac{\|A\|_2}{\epsilon}}$ iterations. And Lemma 3.3 shows that the number of operations required by Algorithm 1 is at most $\sqrt{2\xi\gamma}R\epsilon^{-1/2} - 1$. This completes the proof. □

4 Numerical experiments

In this section, we first compare performance of our subproblem solver to the Krylov subspace method on randomly generated instances whose matrix A in the quadratic term has at least one negative eigenvalue. We then compare ARC ([9]) algorithms with different subproblem solvers on test problems from the CUTEst collection ([14]).

4.1 Comparison for subproblem solvers

In this subsection, we compare the numerical performance between our methods and the Krylov subspace method [9] using randomly generated instances. The problem instances are generated in the same manner as in [7], except that we replace both the original diagonal matrix A and vector b by Q^TAQ and Q^Tb, respectively to make the problem more computationally involved and less trivial. The matrix Q is a random block diagonal matrix (with n/K blocks) and each block is generated by the MATLAB command `orth(rand(K))` with K being a positive integer. Note that the random matrices generated in this manner are of full rank almost surely. As pointed out in [7], by construction, the optimal values are -1 for all cases. Problems with different dimensions n and different sparsity levels were tested. The sparsity of matrix A is then K/n, i.e., a proportion K/n of the total entries are nonzero. For fixed K and n, problems with different condition numbers κ and eigen-gaps `gap` (to be defined later) in the easy and hard cases were also tested, which are believed to strongly affect the hardness of problem (CRS) and the Krylov subspace method [7]. In the easy case, we tested problems with the condition number $\kappa = \frac{\lambda_n+\lambda^*}{\lambda_1+\lambda^*}$, where λ_n is the largest eigenvalue of A and λ^* is the optimal Lagrangian multiplier, which is an indicator for the hardness of the problem [7]. In the hard case, we tested problems with different eigen-gap $\texttt{gap} = \lambda_2 - \lambda_1$, where λ_2 is the second smallest eigenvalue of matrix A. All experiments were run on a Windows workshop with 16 Intel Xeon W-2145 cores (3.70GHz) and 64GB of RAM.

$K = 10,\ n = 2000$

Methods	$\kappa = 10$				$\kappa = 10^2$				$\kappa = 10^3$				$\kappa = 10^4$			
	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}
BBM(AP)	5.5e-06	13.2	2.85e-03	1.95e-03	8.3e-06	52.4	3.58e-03	1.76e-03	6.9e-06	120.4	1.11e-02	5.79e-03	8.3e-06	338.8	1.71e-02	7.61e-03
BBM(SP)	4.3e-06	15.0	2.70e-03	1.95e-03	6.3e-06	42.2	3.15e-03	1.76e-03	6.8e-06	123.8	1.08e-02	5.79e-03	6.4e-04	361.6	1.73e-02	7.61e-03
APG(AP)	4.1e-06	15.2	2.85e-03	1.95e-03	8.6e-06	55.4	3.77e-03	1.76e-03	8.6e-06	99.8	1.01e-02	5.79e-03	2.1e-03	276.4	1.60e-02	7.61e-03
APG(SP)	6.4e-06	15.2	2.71e-03	1.95e-03	8.0e-06	54.0	3.69e-03	1.76e-03	9.0e-06	108.8	9.56e-03	5.79e-03	3.2e-03	326.0	1.75e-02	7.61e-03
Krylov	6.8e-06	9.0	1.69e-03	0	7.7e-06	26.6	3.12e-03	0	9.5e-06	61.8	7.31e-03	0	8.9e-06	90.8	9.58e-03	0

$K = 10,\ n = 10000$

Methods	$\kappa = 10$				$\kappa = 10^2$				$\kappa = 10^3$				$\kappa = 10^4$			
	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}
BBM(AP)	3.2e-06	11.0	1.31e-02	8.11e-03	7.4e-06	55.0	2.57e-02	8.22e-03	8.6e-06	168.0	1.11e-01	6.77e-02	8.5e-06	348.0	1.59e-01	7.49e-02
BBM(SP)	8.4e-06	11.0	1.25e-02	8.11e-03	9.9e-06	57.0	2.28e-02	8.22e-03	7.5e-06	152.0	1.04e-01	6.77e-02	2.8e-06	415.0	1.69e-01	7.49e-02
APG(AP)	2.2e-06	29.0	2.02e-02	8.11e-03	1.0e-05	68.0	2.87e-02	8.22e-03	9.6e-06	120.0	1.01e-01	6.77e-02	9.9e-06	283.0	1.49e-01	7.49e-02
APG(SP)	6.9e-06	12.0	1.35e-02	8.11e-03	9.5e-06	70.0	2.78e-02	8.22e-03	9.4e-06	135.0	1.06e-01	6.77e-02	1.0e-05	324.0	1.59e-01	7.49e-02
Krylov	6.1e-06	9.0	4.05e-03	0	8.2e-06	27.0	9.30e-03	0	9.2e-06	68.0	2.17e-02	0	1.0e-05	146.0	4.55e-02	0

$K = 100,\ n = 100$

Methods	$\kappa = 10$				$\kappa = 10^2$				$\kappa = 10^3$				$\kappa = 10^4$			
	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}
BBM(AP)	5.4e-06	13.6	1.07e-03	7.94e-04	6.6e-06	55.2	1.60e-03	8.09e-04	1.2e-05	104.1	2.35e-03	1.11e-03	5.5e-04	276.6	4.23e-03	1.39e-03
BBM(SP)	4.6e-06	15.2	1.06e-03	7.94e-04	6.9e-06	56.3	1.53e-03	8.09e-04	7.8e-06	101.8	2.20e-03	1.11e-03	8.8e-04	286.4	4.21e-03	1.39e-03
APG(AP)	5.4e-06	17.2	1.13e-03	7.94e-04	8.1e-06	60.2	1.61e-03	8.09e-04	8.7e-06	96.5	2.21e-03	1.11e-03	1.4e-03	266.8	4.05e-03	1.39e-03
APG(SP)	5.8e-06	18.0	1.11e-03	7.94e-04	7.9e-06	60.1	1.60e-03	8.09e-04	8.4e-06	90.3	2.12e-03	1.11e-03	2.8e-03	295.4	4.38e-03	1.39e-03
Krylov	5.9e-06	9.0	7.99e-04	0	7.2e-06	21.9	1.80e-03	0	7.0e-06	33.1	2.32e-03	0	6.9e-06	34.4	2.15e-03	0

$K = 100,\ n = 1000$

Methods	$\kappa = 10$				$\kappa = 10^2$				$\kappa = 10^3$				$\kappa = 10^4$			
	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}
BBM(AP)	5.2e-06	13.3	4.97e-03	3.20e-03	6.9e-06	69.8	9.76e-03	3.61e-03	8.8e-06	150.7	2.09e-02	8.10e-03	1.3e-05	246.1	3.94e-02	1.71e-02
BBM(SP)	4.5e-06	15.1	4.92e-03	3.20e-03	6.7e-06	70.5	9.76e-03	3.61e-03	8.0e-06	142.6	2.00e-02	8.10e-03	1.1e-03	337.9	4.43e-02	1.71e-02
APG(AP)	5.0e-06	17.3	5.29e-03	3.20e-03	8.6e-06	86.3	1.13e-02	3.61e-03	9.2e-06	127.6	1.92e-02	8.10e-03	2.4e-03	288.6	4.09e-02	1.71e-02
APG(SP)	7.8e-06	16.7	5.21e-03	3.20e-03	8.6e-06	76.7	1.05e-02	3.61e-03	9.5e-06	120.8	1.85e-02	8.10e-03	3.2e-03	285.5	4.06e-02	1.71e-02
Krylov	6.8e-06	9.0	2.52e-03	0	7.9e-06	26.1	6.69e-03	0	9.0e-06	60.6	1.50e-02	0	9.1e-06	87.0	2.08e-02	0

Table 1: Comparison between Krylov subspace methods and our methods for solving (CRS) for different dimensions and sparsity levels in the **easy** case. Time unit: second.

$K = 100,\ n = 5000$

Methods	$\kappa = 10$				$\kappa = 10^2$				$\kappa = 10^3$				$\kappa = 10^4$			
	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}
BBM(AP)	4.9e-06	12.6	2.74e-02	1.56e-02	7.1e-06	57.6	5.39e-02	1.75e-02	6.7e-06	143.2	1.26e-01	5.29e-02	2.6e-04	382.8	3.06e-01	1.21e-01
BBM(SP)	5.0e-06	12.7	2.68e-02	1.56e-02	8.0e-06	58.2	4.95e-02	1.75e-02	5.7e-06	159.6	1.32e-01	5.29e-02	2.6e-04	390.9	3.07e-01	1.21e-01
APG(AP)	6.3e-06	14.1	2.77e-02	1.56e-02	8.3e-06	69.0	5.64e-02	1.75e-02	9.7e-06	115.0	1.13e-01	5.29e-02	3.9e-04	344.1	2.90e-01	1.21e-01
APG(SP)	4.0e-06	18.5	3.04e-02	1.56e-02	8.2e-06	57.8	5.07e-02	1.75e-02	9.4e-06	121.4	1.16e-01	5.29e-02	6.4e-04	305.8	2.71e-01	1.21e-01
Krylov	6.4e-06	9.0	8.01e-03	0	8.4e-06	26.5	2.37e-02	0	9.4e-06	68.6	6.02e-02	0	9.5e-06	134.9	1.16e-01	0

$K = 100,\ n = 10000$

Methods	$\kappa = 10$				$\kappa = 10^2$				$\kappa = 10^3$				$\kappa = 10^4$			
	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}
BBM(AP)	6.4e-06	13.1	4.91e-02	2.70e-02	7.2e-06	45.0	8.61e-02	2.83e-02	8.1e-06	144.1	2.68e-01	1.02e-01	1.0e-05	383.1	7.73e-01	3.38e-01
BBM(SP)	4.5e-06	13.0	4.86e-02	2.70e-02	8.6e-06	47.8	8.87e-02	2.83e-02	8.3e-06	153.0	2.78e-01	1.02e-01	1.3e-04	361.7	7.46e-01	3.38e-01
APG(AP)	5.1e-06	17.1	5.50e-02	2.70e-02	7.7e-06	46.1	8.92e-02	2.83e-02	8.9e-06	130.9	2.57e-01	1.02e-01	9.0e-04	381.9	7.80e-01	3.38e-01
APG(SP)	5.6e-06	18.2	5.68e-02	2.70e-02	7.0e-06	47.3	9.08e-02	2.83e-02	9.5e-06	123.9	2.50e-01	1.02e-01	9.4e-06	380.9	7.76e-01	3.38e-01
Krylov	6.5e-06	9.0	1.76e-02	0	8.2e-06	26.1	5.12e-02	0	9.5e-06	69.9	1.37e-01	0	9.7e-06	147.3	2.91e-01	0

$K = 1000,\ n = 1000$

Methods	$\kappa = 10$				$\kappa = 10^2$				$\kappa = 10^3$				$\kappa = 10^4$			
	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}
BBM(AP)	6.0e-06	12.3	4.33e-02	2.38e-02	6.3e-06	59.9	9.52e-02	2.54e-02	7.8e-06	132.9	1.93e-01	5.60e-02	7.0e-04	332.9	4.57e-01	1.16e-01
BBM(SP)	7.0e-06	13.9	4.46e-02	2.38e-02	5.8e-06	55.3	8.77e-02	2.54e-02	7.9e-06	127.0	1.87e-01	5.60e-02	9.8e-04	283.1	4.05e-01	1.16e-01
APG(AP)	5.0e-06	15.8	4.76e-02	2.38e-02	7.5e-06	61.1	9.46e-02	2.54e-02	8.9e-06	102.7	1.65e-01	5.60e-02	1.2e-03	312.1	4.37e-01	1.16e-01
APG(SP)	5.6e-06	16.4	4.79e-02	2.38e-02	7.7e-06	57.1	9.07e-02	2.54e-02	9.5e-06	109.3	1.71e-01	5.60e-02	1.8e-03	358.6	4.83e-01	1.16e-01
Krylov	6.4e-06	9.0	1.69e-02	0	8.0e-06	26.5	5.11e-02	0	9.0e-06	59.9	1.13e-01	0	8.8e-06	81.4	1.58e-01	0

$K = 1000,\ n = 5000$

Methods	$\kappa = 10$				$\kappa = 10^2$				$\kappa = 10^3$				$\kappa = 10^4$			
	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}
BBM(AP)	4.9e-06	13.6	2.86e-01	1.61e-01	6.0e-06	54.9	5.46e-01	1.66e-01	8.1e-06	144.1	1.45e+00	5.33e-01	9.7e-06	367.2	3.42e+00	1.15e+00
BBM(SP)	5.8e-06	13.7	2.83e-01	1.61e-01	7.6e-06	51.3	5.19e-01	1.66e-01	7.3e-06	163.1	1.55e+00	5.33e-01	7.8e-04	394.9	3.58e+00	1.15e+00
APG(AP)	5.7e-06	15.6	2.99e-01	1.61e-01	8.8e-06	54.5	5.44e-01	1.66e-01	9.2e-06	120.8	1.31e+00	5.33e-01	9.9e-06	333.7	3.22e+00	1.15e+00
APG(SP)	5.4e-06	17.4	3.11e-01	1.61e-01	8.1e-06	55.8	5.49e-01	1.66e-01	9.4e-06	128.0	1.35e+00	5.33e-01	5.3e-04	328.1	3.18e+00	1.15e+00
Krylov	6.6e-06	9.0	1.02e-01	0	8.3e-06	26.1	2.98e-01	0	9.4e-06	69.2	7.92e-01	0	9.6e-06	131.7	1.52e+00	0

$K = 1000,\ n = 10000$

Methods	$\kappa = 10$				$\kappa = 10^2$				$\kappa = 10^3$				$\kappa = 10^4$			
	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}
BBM(AP)	4.9e-06	12.7	5.21e-01	2.91e-01	7.5e-06	51.4	1.02e+00	3.01e-01	7.7e-06	140.1	2.68e+00	8.24e-01	1.5e-05	419.7	8.25e+00	2.95e+00
BBM(SP)	5.2e-06	13.5	5.27e-01	2.91e-01	8.3e-06	46.1	9.51e-01	3.01e-01	8.9e-06	131.8	2.56e+00	8.24e-01	1.5e-05	428.3	8.33e+00	2.95e+00
APG(AP)	6.1e-06	14.1	5.50e-01	2.91e-01	7.5e-06	51.6	1.04e+00	3.01e-01	9.3e-06	107.8	2.26e+00	8.24e-01	1.4e-05	364.3	7.58e+00	2.95e+00
APG(SP)	5.5e-06	15.3	5.65e-01	2.91e-01	8.6e-06	50.1	1.01e+00	3.01e-01	9.3e-06	105.3	2.23e+00	8.24e-01	2.5e-04	364.5	7.58e+00	2.95e+00
Krylov	6.6e-06	9.0	2.02e-01	0	8.1e-06	26.5	6.12e-01	0	9.3e-06	65.4	1.53e+00	0	9.6e-06	149.5	3.47e+00	0

Table 2: Comparison between Krylov subspace methods and our methods for solving (CRS) for different dimensions and sparsity levels in the **easy** case. Time unit: second.

$K = 1000,\ n = 10000$																
Methods	$\mathsf{gap} = 10^{-1}$				$\mathsf{gap} = 10^{-2}$				$\mathsf{gap} = 10^{-3}$				$\mathsf{gap} = 10^{-4}$			
	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}	fval-opt	iter	time	time_{eig}
`BBM(AP)`	8.2e-06	8.8	6.49e-01	4.47e-01	7.7e-06	27.1	1.52e+00	1.09e+00	8.6e-06	53.7	3.61e+00	2.83e+00	8.8e-06	63.4	8.07e+00	7.18e+00
`BBM(SP)`	8.2e-06	8.8	6.41e-01	4.47e-01	7.6e-06	27.4	1.52e+00	1.09e+00	8.0e-06	54.7	3.59e+00	2.83e+00	9.6e-06	62.2	8.04e+00	7.18e+00
`APG(AP)`	5.6e-06	7.6	6.43e-01	4.47e-01	7.5e-06	16.0	1.39e+00	1.09e+00	9.2e-06	36.3	3.39e+00	2.83e+00	9.9e-06	41.1	7.79e+00	7.18e+00
`APG(SP)`	5.6e-06	7.6	6.41e-01	4.47e-01	7.5e-06	16.0	1.39e+00	1.09e+00	9.7e-06	35.8	3.38e+00	2.83e+00	1.0e-05	39.1	7.78e+00	7.18e+00
`Krylov`	8.8e-01	455.9	1.07e+01	0	1.9e+01	175.1	4.12e+00	0	3.7e-03	442.5	1.04e+01	0	1.1e-03	500.0	1.17e+01	0

Table 3: Comparison between Krylov subspace methods and our methods for solving (CRS) for different dimensions and sparsity levels in the **hard** case. Time unit: second.

The approximate eigenvalue in formulating the surrogate problem was computed by the MATLAB function `eigs`. We found empirically that setting the tolerance (an input argument of the MATLAB function `eigs`) to be $5/\kappa$ in the easy case and 10^{-6} in the hard case yields a reasonable trade-off between accuracy and efficiency. Both (SP) and (AP) were tested. Besides APG, we have also applied BBM to solve the problems (SP) and (AP). For APG, we used a restarting strategy, which is a common method for speeding up the algorithm [21, 16]. For BBM, we used a simple line search rule to guarantee the decrease of the objective function values. As we know the optimal value is -1, we terminate our algorithm and the Krylov subspace method[2] if the objective value is less than `-1+1e-6`.

Tables 1–3 show the performance comparison of our methods and the Krylov subspace method. In the tables, `BBM(AP)` denotes the method that solves problem (AP) by BBM; `BBM(SP)` denotes the method that solves problem (SP) by BBM; `APG(AP)` denotes the method that solves problem (AP) by APG; `APG(SP)` denotes the method that solves problem (SP) by APG; and `Krylov` denotes the Krylov subspace methods for directly solving problem (CRS). In the tables, `fval-opt` denotes the objective value accuracy, which is the objective value returned by the algorithm minus the optimal value; `iter` denotes the iteration number of each algorithm; `time` denotes the total time of each algorithm; $\texttt{time}_{\texttt{eig}}$ denotes the time cost for approximately computing the minimum eigenvalue, which is 0 for the Krylov subspace method.

From Tables 1 and 2 , we see that in the easy case, our methods achieved the prescribed accuracy when $\kappa < 10^4$ and were a bit slower than the Krylov subspace method. All our four methods took more iterations and CPU time as the condition number κ increases, as expected. We also obtain that in our methods the eigenvalue computation took about 1/3 to 1/2 of the total CPU time and the ratio of $\texttt{time}_{\texttt{eig}}$ over `time` becomes slightly smaller as the condition number increases. From Table 3, we see that in the hard case, our methods performed much better than the Krylov subspace method in terms of solution quality, iteration number and CPU time. All our four methods took more iterations and CPU time as the eigen-gap κ increases, as expected. We also observe that the eigenvalue computation took more than 2/3 of total time and the ratio of time_{eig} over time becomes larger if the eigen-gap decreases. For the test problems with `gap`$= 10^{-4}$, the Krylov subspace method attained the maximum time 500 seconds and failed to return a solution satisfying the stopping criteria, while our methods sufficiently solved all the problems in less than 10 seconds on average. As our methods always outperform the Krylov subspace method in the hard case, we do not report more results for the hard case. In fact, the Krylov subspace method fails to find an approximate solution, while our methods always find a

[2]The authors are indebted to Coralia Cartis for her kind sharing of the MATLAB codes for the Krylov subspace method.

good approximate solution with an accuracy 10^{-6}. We also notice that, in both the easy and hard cases, APG are slightly better than BBM, especially for instances with a large condition number, and each of APG and BBM has a similar performance on solving (AP) and (SP). Comparing the ratio $\text{time}_{\text{eig}}/\text{time}$, we conclude that the two considered first-order methods performs on par in terms of solving the surrogate problems (AP) and (SP). For future research, we would like to develop more efficient methods for solving the surrogate problem.

4.2 Numerical tests on CUTEst problems

In this subsection, we compare the numerical performance of ARC algorithms ([9]) implemented with different subproblem solvers on unconstrained test problems of the CUTEst collections.

Towards that end, we describe a variant of ARC, Algorithm 2, whose subproblem solver is based on our reformulation. Denoting the function to minimize by F, in each iteration, we compute an approximate solution for the cubic regularization model function

$$\min_s m_k(s) = s^T B_k s + g_k^T s + \frac{\sigma_k}{3}\|s\|^3,$$

where B_k is an approximation of the Hessian $\nabla^2 F(x_k)$, $g_k = \nabla F(x_k)$ and σ_k is an adaptive parameter. Suppose $\mathcal{S}$ is an arbitrary solver for (CRS) and $\mathcal{A}$ is an arbitrary solver for the surrogate problem (AP). In our algorithm, we call $\mathcal{A}$ if the following condition is met:

$$\|g_k\| \leq \max\left(F(x_k), 1\right) \cdot \epsilon_1 \quad \text{and} \quad \lambda_1(B_k) < -\epsilon_2, \tag{15}$$

where ϵ_1 and ϵ_2 are some small positive real numbers and $\lambda_1(B_k)$ is the minimum eigenvalue of B_k; and otherwise we call $\mathcal{S}$ to solve the model function directly. Condition (15) is motivated by the facts that the Cauchy point is a good initial point when the norm of the gradient is large and that the subproblem solver $\mathcal{A}$ is designed for cases where B_k at current iterate has at least one negative eigenvalue. We use the Cauchy point [9] as an initial point:

$$s_k^C = -\alpha_k^C \text{ and } \alpha_k^C = \operatorname*{argmin}_{\alpha \in \mathbb{R}_+} m_k(-\alpha g_k).$$

The (approximate) solution s_k to the model function returned by the solver $\mathcal{S}$ or $\mathcal{A}$ is accepted as the trial step if the model function value at s_k is smaller than that at the Cauchy point s_k^C; otherwise the Cauchy point s_k^C is used. From [9, Lemma 2.1], the above choice of the trial step guarantees that our variant of ARC (Algorithm 2) converges to a first-order stationary point (i.e., $\lim_{k\to\infty} \|g_k\| = 0$) under some mild conditions, e.g., F is a continuously differentiable function, $\|g_{t_i} - g_{l_i}\| \to 0$ whenever $\|x_{t_i} - x_{l_i}\| \to 0$ for any subsequences $\{t_i\}$ and $\{l_i\}$ of

Algorithm 2 ARC using reformulation (AP)

Input: $x_0,\ \gamma_2 \geq \gamma_1 > 1,\ 1 > \eta_2 \geq \eta_1 > 0$, and $\sigma_0 > 0$, for $k = 0, 1, ...$ until convergence

1: compute the Cauchy point s_k^C

2: **if** condition (15) is satisfied **then**

3: compute a trial step $\bar{s}_k$ using $\mathcal{A}$ with an initial point $(s_k^C, \|s_k^C\|)$

4: **else**

5: compute a trial step $\bar{s}_k$ using $\mathcal{S}$ with an initial point s_k^C

6: **end if**

7: set

$$s_k = \begin{cases} \bar{s}_k & \text{if } m_k(\bar{s}_k) \leq m_k(s_k^C) \\ s_k^C & \text{otherwise} \end{cases}$$

8: compute $f(x_k + s_k)$ and

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

9: set

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

10: set

$$\sigma_{k+1} \in \begin{cases} (0, \sigma_k] & \text{if } \rho_k > \eta_2 & \text{(very successful iteration)} \\ [\sigma_k, \gamma_1 \sigma_k] & \text{if } \eta_1 \leq \rho_k \leq \eta_2 & \text{(successful iteration)} \\ [\gamma_1 \sigma_k, \gamma_2 \sigma_k] & \text{otherwise} & \text{(unsuccessful iteration)} \end{cases}$$

natural numbers and the norm of the approximate Hessian B_k is upper bounded by some positive constant for all k.

We implemented two different subproblem solvers $\mathcal{A}$ in Algorithm 2. In our implementation, if condition (15) is not satisfied, we still solve the original cubic model (CRS) by BBM so that the effect of solving problem (CRS) via our reformulation can be shown by comparing the overall time and solution quality. For the cases where condition (15) is satisfied, we implemented both APG and BBM to solve the surrogate problem (AP). We call the former ARC-RAPG and the latter ARC-RBB. We compare our algorithms against the ARC algorithm in [9], denoted by ARC-GLRT, where the subproblems are solved by the Krylov subspace method and another ARC algorithm with subproblems solved by BBM, denoted by ARC-BB, which is a simplified version of the algorithm in [4].

We tested medium-size ($n \in [500, 1500]$) problems from the CUTEst collections as in [4]. For condition (15), we set $\epsilon_1 = 10^{-2}$ and $\epsilon_2 = 10^{-4}$. In our numerical tests, we found that condition (15) is never met for some problems, and thus ARC-RAPG and ARC-RBB reduce to ARC-BB. Therefore, we only report results on those instances where condition (15) is

satisfied in at lease one iteration. There are 20 such instances. We implemented all the ARC algorithms in MATLAB 2017a. All the experiments were run on a Macbook Pro laptop. The parameters in ARC are chosen as described in [9]. We set $B_k = H_k := \nabla^2 F(x_k)$ and the minimum eigenvalue of H_k is approximately computed by the MATLAB command `eigs` with tolerance `1e-4`. We terminate each ARC algorithm if either the iteration counter k reaches 5000 or the stopping criteria

$$\|g_k\| \leq 10^{-5} \quad \text{and} \quad \lambda_1(H_k) \geq -10^{-3}$$

are met. In our test, all algorithms were terminated before iteration counter reaches 5000. When using APG to solve (AP) in ARC-RAPG, we again used the restarting strategy as in Section 4.1. For BBM to solve (AP) or (CRS) in ARC-BB, ARC-RAPG and ARC-RBB, we used the simple line search rule to guarantee the decrease of the objective function values as in Section 4.1. In all implementations, we terminate the subproblem solver in each inner iteration when the iteration number of the subproblem reaches 150, or the following stopping criterion is met:

$$\|\nabla m_k(s_k)\| \leq \min\{1, \|s_k\|\}\|g(x_k)\|.$$

We report the numerical results for ARC-RAPG, ARC-RBB, ARC-GLRT and ARC-BB in Table 4. The first column shows the name of the problem instance. The number below the problem name represents its dimension. The column f^* shows the final objective function value. The columns n_i, n_{prod} , n_f , n_g and n_{eig} show the iteration number, number of Hessian-vector products, number of function evaluations, number of gradient evaluations and the number of eigenvalue computations. The columns time, time_{eig} and $\text{time}_{\text{loop}}$, show in seconds the overall CPU time, eigenvalue computation time and difference between the last two, respectively. From Table 4, we see that with our stopping criteria, the algorithms return the same final objective values on almost all the problem instances (except the problem CHAINWOO). The quantities n_i, n_{prod}, n_f, n_g and n_{eig} of the four algorithms are comparable. From the table, we see that ARC-GLRT slightly outperforms the other three algorithms. The table also shows that for some problems, our methods ARC-RAPG and ARC-RBB take much more CPU time than ARC-GLRT and ARC-BB. We found that this is mainly because time_{eig} is too large and dominates the total runtime in these problems. This can be further divided into two situations: either there are many eigenvalue computations, or the number of eigenvalue computations is small but each eigenvalue computation takes a large amount of time (in these cases, the MATLAB function `eigs` is difficult to converge, and we used `eig` instead to compute full eigenvalues). Excluding the time to compute the eigenvalues, the actual times of the four algorithms do not differ much, which is evidenced by the column $\text{time}_{\text{loop}}$.

To get more insights from the numerical tests, we also use performance profiles ([10]) to

illustrate the experimental results in Figure 1–3. We note that, although ARC-GLRT has the best performance, the iteration numbers and the gradient evaluation numbers required by our algorithm are less than 2 times of those by ARC-GLRT on over 90% of the tests, and the numbers of Hessian-vector products required by our algorithms are also less than 2 times of those by ARC-GLRT on about 80% of the tests. We further note that ARC-RAPG, ARC-RBB and ARC-BB have the similar performance in terms of iteration numbers and gradient evaluations. We also plot the performance profiles on test problems where algorithms require a CPU time of more than 1 second in Figure 4. Compared with the previous results, the gap between our algorithms and ARC-GLRT has narrowed.

To see more advantages of our reformulation, we also investigate the numerical results for all the 10 realizations with different initial points for each problem. In Table 5, we report the number that ARC-RAPG or ARC-RBB outperforms ARC-GLRT and ARC-BB out of the 10 realizations for each problem. We see that our methods outperform ARC-GLRT and ARC-BB frequently in terms of iteration number, the numbers of Hessian-vector products and gradient evaluations. This shows that our new reformulation may bring advantages in ARC algorithms.

Problem	Method	n_i	n_{prod}	n_f	n_g	n_{eig}	f^*	time	time_{eig}	$\text{time}_{\text{loop}}$
BROYDN7D (1000)	ARC-GLRT	43.6	804.8	44.6	35.7	-	2.45E+02	0.270	-	0.270
	ARC-BB	42.6	864.5	43.6	36.3	1.0	2.42E+02	0.386	0.051	0.335
	ARC-RAPG	42.5	917.5	43.5	36.6	15.7	2.42E+02	0.888	0.558	0.331
	ARC-RBB	43.9	941.2	44.9	36.7	16.9	2.42E+02	0.929	0.597	0.332
BRYBND (1000)	ARC-GLRT	40.3	629.5	41.3	31.2	-	5.92E-14	0.300	-	0.300
	ARC-BB	37.3	644.5	38.3	30.4	1.0	3.24E-13	0.308	0.019	0.289
	ARC-RAPG	37.3	670.4	38.3	30.4	2.2	3.10E-13	0.327	0.035	0.292
	ARC-RBB	37.5	692.4	38.5	30.6	2.4	4.56E-13	0.340	0.037	0.303
CHAINWOO (1000)	ARC-GLRT	208.1	5591.5	209.1	155.9	-	1.07E+03	1.460	-	1.460
	ARC-BB	310.4	12186.4	311.4	228.5	1.0	1.11E+03	4.031	0.324	3.707
	ARC-RAPG	330.2	14027.7	331.2	231.8	199.6	1.11E+03	26.609	22.560	4.049
	ARC-RBB	311.2	11232.0	312.2	230.6	183.6	1.13E+03	24.114	20.940	3.174
DIXMAANF (1500)	ARC-GLRT	24.9	631.7	25.9	23.1	-	1.00E+00	0.409	-	0.409
	ARC-BB	20.9	534.4	21.9	20.7	1.0	1.00E+00	0.541	0.116	0.426
	ARC-RAPG	22.7	628.1	23.7	20.9	10.9	1.00E+00	1.399	0.971	0.428
	ARC-RBB	21.7	523.2	22.7	20.8	9.8	1.00E+00	1.399	0.971	0.428
DIXMAANG (1500)	ARC-GLRT	23.6	523.4	24.6	22.6	-	1.00E+00	0.351	-	0.351
	ARC-BB	24.0	576.8	25.0	22.8	1.0	1.00E+00	0.578	0.117	0.461
	ARC-RAPG	25.8	806.0	26.8	23.4	11.4	1.00E+00	1.518	1.007	0.511
	ARC-RBB	24.9	578.2	25.9	23.0	10.6	1.00E+00	1.388	0.975	0.413
DIXMAANH (1500)	ARC-GLRT	26.5	602.7	27.5	24.3	-	1.00E+00	0.402	-	0.402
	ARC-BB	26.7	720.8	27.7	24.7	1.0	1.00E+00	0.660	0.113	0.547
	ARC-RAPG	29.0	1005.8	30.0	25.0	12.7	1.00E+00	1.712	1.086	0.626
	ARC-RBB	26.7	668.7	27.7	24.7	10.5	1.00E+00	1.416	0.934	0.481
DIXMAANJ (1500)	ARC-GLRT	46.9	5031.1	47.9	39.6	-	1.00E+00	2.458	-	2.458
	ARC-BB	46.2	3309.2	47.2	38.7	1.0	1.00E+00	3.419	1.545	1.873
	ARC-RAPG	52.8	3736.0	53.8	40.0	36.1	1.00E+00	34.097	32.223	1.873

	ARC-RBB	48.3	3104.6	49.3	41.9	31.4	1.00E+00	33.251	31.679	1.572
DIXMAANK (1500)	ARC-GLRT	62.2	6228.8	63.2	50.6	-	1.00E+00	3.058	-	3.058
	ARC-BB	70.3	5865.4	71.3	55.9	1.0	1.00E+00	4.773	1.530	3.242
	ARC-RAPG	82.3	6898.9	83.3	56.9	62.2	1.00E+00	55.014	51.676	3.338
	ARC-RBB	74.6	5836.4	75.6	61.0	54.3	1.00E+00	52.804	49.958	2.846
DIXNAANL (1500)	ARC-GLRT	66.4	6159.8	67.4	54.0	-	1.00E+00	2.981	-	2.981
	ARC-BB	73.0	5840.7	74.0	58.4	1.0	1.00E+00	4.714	1.505	3.209
	ARC-RAPG	83.0	7475.0	84.0	59.8	60.0	1.00E+00	52.685	49.103	3.582
	ARC-RBB	76.1	6458.4	77.1	63.3	53.4	1.00E+00	55.178	52.098	3.080
EXTROSNB (1000)	ARC-GLRT	1762.6	51785.4	1763.6	1233.4	-	1.62E-08	13.566	-	13.566
	ARC-BB	1368.5	196671.1	1369.5	1139.3	1.0	2.97E-06	49.540	0.007	49.533
	ARC-RAPG	1386.6	199155.1	1387.6	1116.3	1277.4	2.98E-06	57.130	8.713	48.417
	ARC-RBB	1393.9	200360.0	1394.9	1118.7	1284.9	2.99E-06	57.412	8.750	48.662
FLETCHCR (1000)	ARC-GLRT	1790.8	42046.0	1791.8	1215.2	-	7.97E-01	10.844	-	10.844
	ARC-BB	1775.8	49084.9	1776.8	1245.5	1.0	7.97E-01	17.060	0.012	17.048
	ARC-RAPG	1781.6	49874.9	1782.6	1248.8	567.9	7.97E-01	28.072	9.042	19.030
	ARC-RBB	1830.3	51176.3	1831.3	1272.7	662.8	7.97E-01	30.574	11.081	19.493
FREUROTH (1000)	ARC-GLRT	35.8	338.2	36.8	30.5	-	1.17E+05	0.254	-	0.254
	ARC-BB	36.4	1378.7	37.4	31.2	1.0	1.17E+05	0.487	0.008	0.479
	ARC-RAPG	35.0	1261.3	36.0	30.5	23.0	1.17E+05	0.681	0.199	0.482
	ARC-RBB	36.1	1425.4	37.1	30.4	23.8	1.17E+05	0.704	0.198	0.506
GENHUMPS (1000)	ARC-GLRT	1676.4	49839.0	1677.4	1027.0	-	2.47E-11	12.462	-	12.462
	ARC-BB	1529.6	41024.1	1530.6	926.6	1.0	1.55E-11	14.545	0.008	14.537
	ARC-RAPG	1529.5	41090.9	1530.5	926.6	9.1	1.59E-11	15.137	0.132	15.006
	ARC-RBB	1529.5	41081.9	1530.5	926.4	9.1	1.58E-11	14.887	0.134	14.753
GENROSE (500)	ARC-GLRT	978.8	20138.0	979.8	660.7	-	1.00E+00	1.953	-	1.953
	ARC-BB	1097.5	28322.9	1098.5	748.3	1.0	1.00E+00	2.024	0.004	2.020
	ARC-RAPG	1097.5	28850.6	1098.5	746.1	221.3	1.00E+00	3.142	0.804	2.338
	ARC-RBB	1093.1	28496.0	1094.1	747.0	218.1	1.00E+00	3.450	0.935	2.515
NONCVXU2 (1000)	ARC-GLRT	66.7	8319.3	67.7	62.4	-	2.32E+03	1.866	-	1.866
	ARC-BB	123.7	8467.1	124.7	122.6	1.0	2.32E+03	3.067	0.697	2.369
	ARC-RAPG	116.1	7497.0	117.1	116.0	114.1	2.32E+03	67.010	64.964	2.046
	ARC-RBB	128.5	7936.8	129.5	122.6	125.5	2.32E+03	71.118	68.930	2.188
NONCVXUN (1000)	ARC-GLRT	264.2	213656.6	265.2	258.6	-	2.32E+03	43.103	-	43.103
	ARC-BB	1719.2	240580.4	1720.2	1716.4	1.0	2.32E+03	62.662	0.753	61.909
	ARC-RAPG	1968.8	277075.0	1969.8	1967.6	1966.5	2.32E+03	1255.385	1195.682	59.704
	ARC-RBB	2220.7	312062.4	2221.7	2214.2	2217.5	2.32E+03	1413.231	1345.627	67.604
OSCIPATH (500)	ARC-GLRT	41.7	7781.1	42.7	33.0	-	4.55E-01	0.774	-	0.774
	ARC-BB	72.0	7779.9	73.0	66.1	1.0	4.55E-01	0.718	0.215	0.502
	ARC-RAPG	99.7	11934.9	100.7	75.7	66.4	4.55E-01	15.259	14.305	0.954
	ARC-RBB	75.8	8345.5	76.8	64.4	42.2	4.55E-01	10.241	9.719	0.522
TOINTGSS (1000)	ARC-GLRT	15.7	93.9	16.7	13.0	-	1.00E+01	0.086	-	0.086
	ARC-BB	14.9	270.8	15.9	12.1	1.0	1.00E+01	0.128	0.008	0.121
	ARC-RAPG	17.4	613.9	18.4	13.7	12.1	1.00E+01	0.319	0.090	0.229
	ARC-RBB	17.6	511.2	18.6	13.4	12.3	1.00E+01	0.291	0.096	0.196
TQUARTIC (1000)	ARC-GLRT	67.1	306.9	68.1	55.2	-	9.77E-14	0.354	-	0.354
	ARC-BB	75.6	600.4	76.6	59.0	1.0	1.16E-11	0.467	0.010	0.457
	ARC-RAPG	76.0	907.8	77.0	59.1	7.5	3.72E-10	0.677	0.107	0.570
	ARC-RBB	76.5	730.8	77.5	59.8	6.6	1.74E-11	0.570	0.081	0.489
	ARC-GLRT	294.8	4663.0	295.8	217.5	-	2.16E-15	1.625	-	1.625

WOODS (1000)	ARC-BB	393.0	9877.9	394.0	276.5	1.0	1.8E-14	3.164	0.009	3.154
	ARC-RAPG	393.7	10102.2	394.7	275.9	5.8	6.9E-14	3.559	0.070	3.489
	ARC-RBB	392.8	9847.9	393.8	276.5	5.5	1.99E-16	3.535	0.064	3.471

Table 4: Results on the CUTEst problems

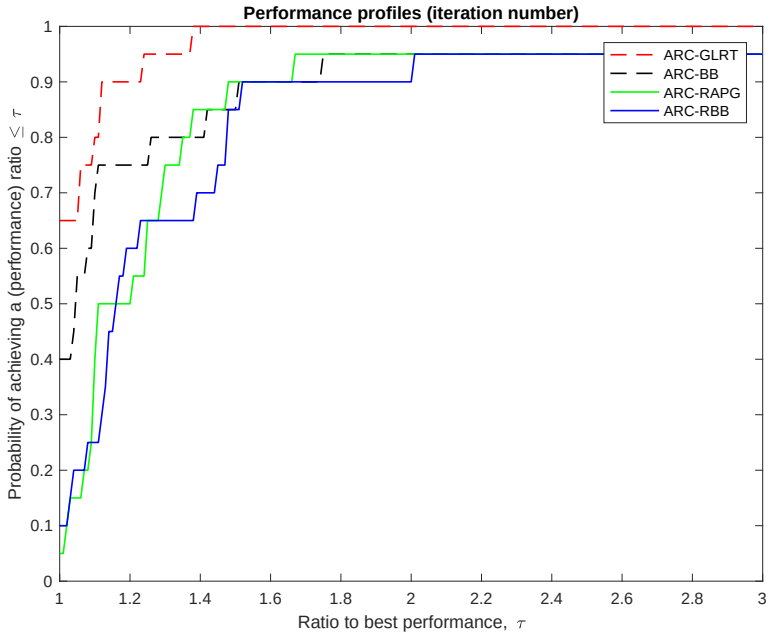

Figure 1: Performance profiles for iteration number for ARC-GLRT, ARC-BB, ARC-RAPG and ARC-RBB on the CUTEst problems

Problem	Index	ARC-RAPG		ARC-RBB	
		ARC-GLRT	ARC-BB	ARC-GLRT	ARC-BB
BROYDN7D	n_i	6	4	6	0
	n_{prod}	4	3	4	2
	n_g	5	0	4	2
BRYBND	n_i	6	0	6	0
	n_{prod}	4	4	4	3
	n_g	6	0	6	0
CHAINWOO	n_i	0	1	0	4
	n_{prod}	0	0	0	6
	n_g	0	2	0	3
DIXMAANF	n_i	8	1	9	2
	n_{prod}	5	3	7	6

	n_g	9	2	9	3
DIXMAANG	n_i	3	1	3	1
	n_{prod}	3	2	3	5
	n_g	3	1	3	1
DIXMAANH	n_i	5	1	5	3
	n_{prod}	3	1	4	6
	n_g	4	2	5	2
DIXMAANJ	n_i	2	2	2	4
	n_{prod}	8	3	8	5
	n_g	4	2	2	0
DIXMAANK	n_i	1	1	0	3
	n_{prod}	3	3	6	6
	n_g	2	2	0	0
DIXMAANL	n_i	1	2	1	4
	n_{prod}	4	3	6	4
	n_g	2	3	0	2
EXTROSNB	n_i	10	4	10	2
	n_{prod}	0	4	0	2
	n_g	9	7	10	7
FLETCHCR	n_i	6	5	4	1
	n_{prod}	2	3	1	2
	n_g	3	4	2	1
FREUROTH	n_i	7	7	6	4
	n_{prod}	0	6	0	6
	n_g	6	6	4	5
GENHUMPS	n_i	10	1	10	2
	n_{prod}	10	0	10	0
	n_g	10	1	10	2
GENROSE	n_i	3	6	2	5
	n_{prod}	0	2	2	5
	n_g	2	7	1	6
NONCVXU2	n_i	0	7	0	2
	n_{prod}	6	8	7	5
	n_g	0	7	0	4
NONCVXUN	n_i	0	3	0	2

	n_{prod}	2	3	2	2
	n_g	0	3	0	2
OSCIPATH	n_i	0	1	0	3
	n_{prod}	2	1	4	3
	n_g	0	2	0	5
TOINTGSS	n_i	4	2	3	0
	n_{prod}	0	2	0	0
	n_g	4	3	3	0
TQUARTIC	n_i	2	5	2	2
	n_{prod}	0	1	0	2
	n_g	3	5	0	2
WOODS	n_i	0	5	0	5
	n_{prod}	0	2	0	5
	n_g	0	4	0	3

Table 5: The number of times ARC-RAPG (or ARC-RBB) performs better than the other two algorithms in 10 realizations with different initial points on each CUTEst problem, considering the number of iterations, the number of Hessian-vector products and the number of gradient evaluations

5 Conclusion

In this paper, we developed a novel approach for solving the problem (CRS). We first equivalently reformulate the problem (CRS) to a convex constrained optimization problem, where the feasible region admits an easy projection and the objective function is formed by using the minimum eigenvalue of the Hessian matrix. To circumvent the expensive cost due to the exact computation of the minimum eigenvalue, we then constructed a surrogate problem which is again a convex constrained optimization problem with a feasible region that admits an easy projection and can be solved by a variety of methods such as APG and BBM. Furthermore, we proved that an ϵ-approximate solution to (CRS) can be obtained in at most $\tilde{O}(\epsilon^{-1/2})$ matrix-vector multiplications if we use the Lanczos method for approximate eigenvalue computation and APG to approximately solve the surrogate problem. Numerical results showed that our methods are comparable to the Krylov subspace method in the easy case and significantly outperform the Krylov subspace method in the hard case. We also implemented variants of ARC where the subproblem solver uses our approaches. The resulting ARC algorithms showed good numerical performance on the problem instances from the CUTEst datasets. As future work, we plan to investigate the complexity of cubic regularization or ARC variants with subproblem solver based on our reformulations for finding

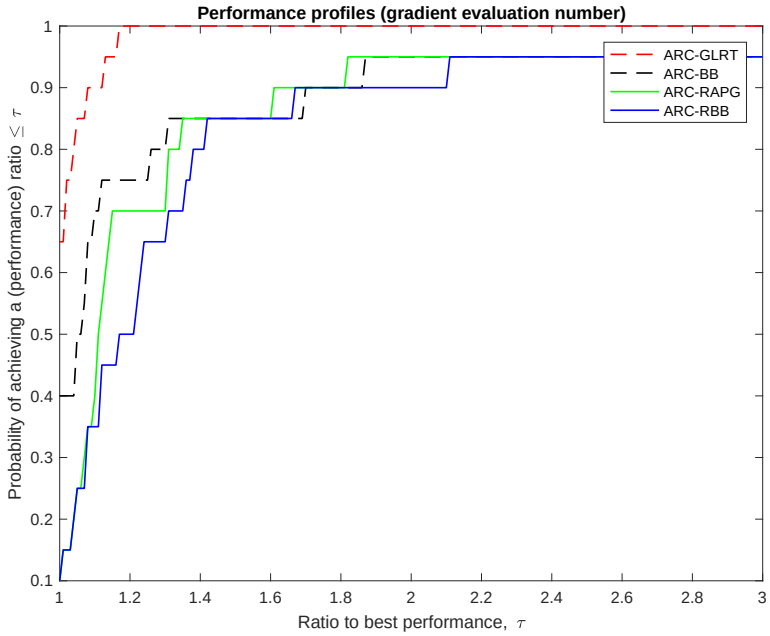

Figure 2: Performance profiles for gradient evaluations for ARC-GLRT, ARC-BB, ARC-RAPG and ARC-RBB on the CUTEst problems

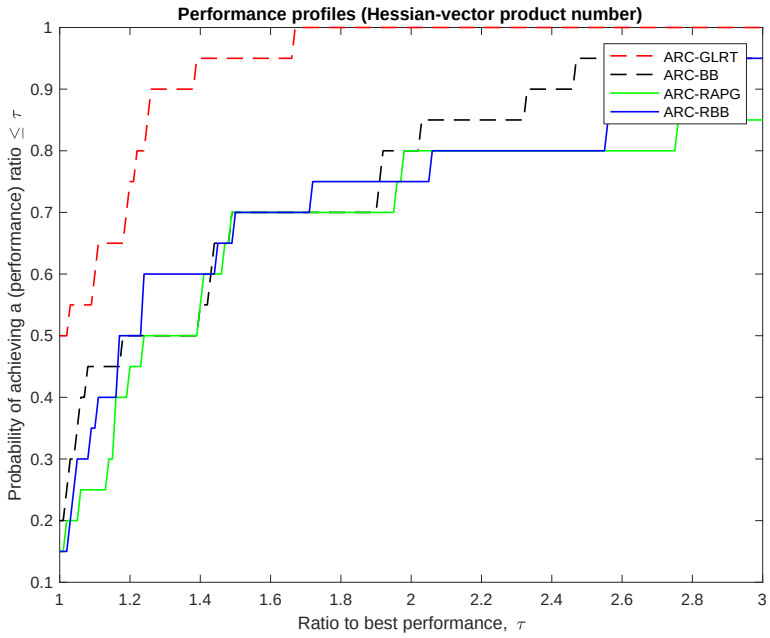

Figure 3: Performance profiles for Hessian-vector products for ARC-GLRT, ARC-BB, ARC-RAPG and ARC-RBB on the CUTEst problems

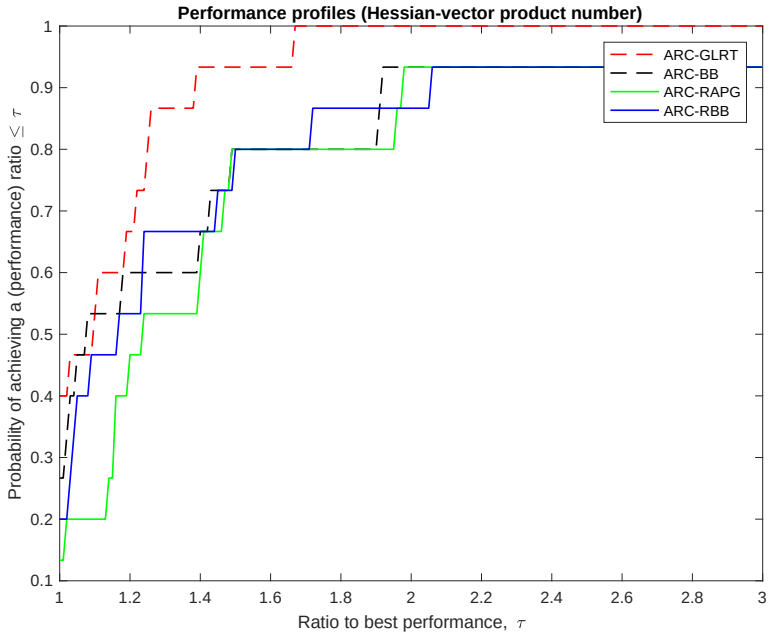

Figure 4: Performance profiles of Hessian-vector products for ARC-GLRT, ARC-BB, ARC-RAPG and ARC-RBB on the CUTEst problems where algorithms require a CPU time more than 1 second

an approximate stationary point and local minimizer of smooth non-convex minimization problems.

A Analysis for problem (AP)

The purpose of this appendix is to show that when the approximate minimum eigenvalue θ of A is close enough to the exact minimum eigenvalue λ_1 ($\lambda_1 \leq \theta < -\bar{\lambda}$ for some $\bar{\lambda}$ defined in the following paragraphs), problem (AP) can be used to construct an approximate solution to (CRS). Define $\hat{B} := \{(x, y) : \|x\|^2 \leq y,\ y \geq \hat{l}\}$. We claim that the problem (AP) simplifies to

$$\begin{aligned} \min_{x\in\mathbb{R}^n, y\in\mathbb{R}} \quad & \frac{1}{2}x^T(A-\theta I)x + b^T x + \frac{\rho}{3}y^{\frac{3}{2}} + \frac{\theta}{2}y \\ \text{subject to} \quad & \|x\|^2 \leq y, \end{aligned} \tag{AP$_0$}$$

i.e., the constraint $y \geq l$ is redundant. This is because when $y \leq l$ in (AP$_0$), $\frac{\rho}{3}y^{\frac{3}{2}} + \frac{\theta}{2}y$ is decreasing in y, and thus the optimal solution of (AP$_0$) must satisfy $y \geq l$. We consider the following two cases.

The hard case: Recall the optimality condition (2) for (CRS). Note that $\|(A+\lambda I)^\dagger b\|^2 - \lambda^2/\rho^2 \leq 0$ and $\|(A+\lambda I)^\dagger b\|^2 - \lambda^2/\rho^2$ is a decreasing function in λ, where $(\cdot)^\dagger$ denotes the *Moore–Penrose pseudoinverse*, when $\lambda \geq -\lambda_1$ because we are in the hard case [9]. First

consider the case that $\|(A-\lambda_1 I)^\dagger b\|^2 - \lambda_1^2/\rho^2 < 0$. Let $\bar{\lambda}$ be the largest $\lambda \in [0, -\lambda_1)$ such that $\|(A+\lambda I)^\dagger b\|^2 - \lambda^2/\rho^2 = 0$, if it exists. If such $\bar{\lambda}$ does not exist, we set $\bar{\lambda} = 0$. Using the fact $\|(A+\lambda I)^\dagger b\|^2 - \lambda^2/\rho^2 < 0$ for $\lambda \geq -\lambda_1$, we have

$$\|(A+\lambda I)^\dagger b\|^2 - \frac{\lambda^2}{\rho^2} < 0, \ \forall \lambda > \bar{\lambda}. \tag{16}$$

Suppose that $\lambda_1 \leq \theta < -\bar{\lambda}$ and that (x^θ, y^θ) is an optimal solution of (AP_0). Let μ be the Lagrange multiplier corresponding to the constraint $\left\|x^\theta\right\|^2 \leq y^\theta$. Then, the KKT condition of (AP_0) implies $Ax^\theta - \theta x^\theta + b + 2\mu x^\theta = 0$ and $\rho\sqrt{y^\theta}/2 + \theta/2 - \mu = 0$. Due to $-\theta + 2\mu > \bar{\lambda}$, we have $\|x^\theta\|^2 < y^\theta, \forall \mu \geq 0$ from (16). Using the complementary slackness $\mu(\|x^\theta\|^2 - y^\theta) = 0$, we have that $\mu = 0$. Thus, every possible stationary point of (AP_0) can be written as $(x^\theta, y^\theta) = ((A-\theta I)^\dagger b + tv, \ (-\theta/\rho)^2)$, where t is a scalar satisfying $\|x^\theta + tv\| \leq \sqrt{y^\theta}$. This in turn yields an approximate optimal solution $x^\theta + tv$ to (CRS), where one should note that different t yields the same objective value. Next, we consider the remaining case that $\|(A+\lambda_1 I)^\dagger b\|^2 - \lambda_1^2/\rho^2 = 0$. This case is similar to the easy case, where we can recover an optimal solution if $\theta \in [\lambda_1, -\bar{\lambda})$, where $\bar{\lambda}$ is the largest $\lambda \in [0, -\lambda_1)$ such that $\|(A+\lambda I)^\dagger b\|^2 - \lambda^2/\rho^2 = 0$. (If such $\bar{\lambda}$ does not exist, we take $\bar{\lambda} = 0$.) The analysis is similar to the easy case below and hence omitted here.

The easy case: Recall that in the easy case we have a unique optimal solution x^* satisfying $\rho\|x^*\| > -\lambda_1$ and $x^* = (A + \rho\|x^*\| I)^{-1} b$; see, e.g., Theorem 3.1 in [9]. Let $\bar{\lambda}$ be the largest $\lambda \in [0, -\lambda_1)$ such that $h(\lambda) := \|(A+\lambda I)^\dagger b\|^2 - \lambda^2/\rho^2 = 0$, if it exists. If such $\bar{\lambda}$ does not exist, we take $\bar{\lambda} = 0$. Then for all $\lambda \in (\bar{\lambda}, -\lambda_1)$, $h(\lambda) > 0$ as $\lim_{\lambda \to -\lambda_1} h(\lambda) = +\infty$. This, together with the definition of $\bar{\lambda}$ and the fact that $h(\lambda)$ is a decreasing function on $(-\lambda_1, +\infty)$, implies that there is only one point, denoting $\tilde{\lambda}$ in $(\bar{\lambda}, +\infty)$ satisfying $\|(A+\lambda I)^\dagger b\|^2 - \lambda^2/\rho^2 = 0$. Let $\tilde{x} = (A + \tilde{\lambda} I)^\dagger b$. The optimality condition (2) implies that $\tilde{x}$ is the unique optimal solution of (CRS). We again suppose that $\lambda_1 \leq \theta < -\bar{\lambda}$. If $\theta = \lambda_1$, (AP_0) reduces to the exact reformulation (CP). Next, we consider the case that $\lambda_1 < \theta < -\bar{\lambda}$. Assuming that $\mu = 0$, the inequality $\lambda_1 < \theta < -\bar{\lambda}$ implies that $\|x^\theta\| = \|(A-\theta I)^\dagger b\| > \sqrt{y^\theta} = -\theta/\rho$, which violates the constraint $\|x\|^2 \leq y$. Hence, we must have $\mu > 0$, and thus we always have $\|x^\theta\| = \sqrt{y^\theta}$, i.e., $\|(A + (-\theta+\mu) I)^\dagger b\| = (-\theta+\mu)/\rho$. This implies $-\theta + \mu = \lambda^*$. That is, we recover the optimal solution if $\theta < -\bar{\lambda}$.

Acknowledgments

Rujun Jiang is supported by National Natural Science Foundation of China under Grant 11801087. Man-Chung Yue is supported by the Hong Kong Research Grants Council under the grant 25302420.

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*, pages 1195–1199. ACM, 2017.

[2] J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[4] T. Bianconcini, G. Liuzzi, B. Morini, and M. Sciandrone. On the use of iterative methods in cubic regularization for unconstrained optimization. *Computational Optimization and Applications*, 60(1):35–57, 2015.

[5] E. Birgin and J. Martínez. A Newton-like method with mixed factorizations and cubic regularization for unconstrained minimization. *Computational Optimization and Applications*, 73(3):707–753, 2019.

[6] Y. Carmon and J. Duchi. Gradient descent finds the cubic-regularized nonconvex Newton step. *SIAM Journal on Optimization*, 29(3):2146–2178, 2019.

[7] Y. Carmon and J. C. Duchi. Analysis of Krylov subspace solutions of regularized nonconvex quadratic problems. In *Advances in Neural Information Processing Systems*, pages 10705–10715, 2018.

[8] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Accelerated methods for nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1751–1772, 2018.

[9] C. Cartis, N. I. Gould, and P. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, 2011.

[10] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[11] O. E. Flippo and B. Jansen. Duality and sensitivity in nonconvex quadratic optimization over an ellipsoid. *European Journal of Operational Research*, 94(1):167–178, 1996.

[12] H. Ghanbari and K. Scheinberg. Proximal quasi-Newton methods for regularized convex optimization with linear and accelerated sublinear convergence rates. *Computational Optimization and Applications*, 69(3):597–627, 2018.

[13] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2013.

[14] N. I. Gould, D. Orban, and P. L. Toint. Cutest: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015.

[15] N. Ho-Nguyen and F. Kilinc-Karzan. A second-order cone based approach for solving the trust-region subproblem and its variants. *SIAM Journal on Optimization*, 27(3):1485–1512, 2017.

[16] N. Ito, A. Takeda, and K.-C. Toh. A unified formulation and fast accelerated proximal gradient method for classification. *The Journal of Machine Learning Research*, 18(1):510–558, 2017.

[17] R. Jiang and D. Li. Novel reformulations and efficient algorithms for the generalized trust region subproblem. *SIAM Journal on Optimization*, 29(2):1603–1633, 2019.

[18] J. Kuczyński and H. Woźniakowski. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1094–1122, 1992.

[19] Y. Nesterov and B. T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[20] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[21] B. O'Donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, 2015.

[22] C. W. Royer and S. J. Wright. Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1448–1477, 2018.

[23] J. A. Tropp and S. J. Wright. Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, 98(6):948–958, 2010.

[24] J. Wang and Y. Xia. A linear-time algorithm for the trust region subproblem based on hidden convexity. *Optimization Letters*, 11(8):1639–1646, 2017.

[25] M.-C. Yue, Z. Zhou, and A. Man-Cho So. On the quadratic convergence of the cubic regularization method under a local error bound condition. *SIAM Journal on Optimization*, 29(1):904–932, 2019.

[26] M.-C. Yue, Z. Zhou, and A. M.-C. So. A family of inexact SQA methods for non-smooth convex minimization with provable convergence guarantees based on the Luo–Tseng error bound property. *Mathematical Programming*, 174(1-2):327–358, 2019.