Noname manuscript No.
(will be inserted by the editor)

An efficient adaptive accelerated inexact proximal point method for solving linearly constrained nonconvex composite problems

Weiwei Kong · Jefferson G. Melo · Renato D.C. Monteiro

December 5, 2019

Abstract This paper proposes an efficient adaptive variant of a quadratic penalty accelerated inexact proximal point (QP-AIPP) method proposed earlier by the authors. Both the QP-AIPP method and its variant solve linearly set constrained nonconvex composite optimization problems using a quadratic penalty approach where the generated penalized subproblems are solved by a variant of the underlying AIPP method. The variant, in turn, solves a given penalized subproblem by generating a sequence of proximal subproblems which are then solved by an accelerated composite gradient algorithm. The main difference between AIPP and its variant is that the proximal subproblems in the former are always convex while the ones in the latter are not necessarily convex due to the fact that their prox parameters are chosen as aggressively as possible so as to improve efficiency. The possibly nonconvex proximal subproblems generated by the AIPP variant are also tentatively solved by a novel adaptive accelerated composite gradient algorithm based on the validity of some key convergence inequalities. As a result, the variant generates a sequence of proximal subproblems where the stepsizes are adaptively changed according to the responses obtained from the calls to the accelerated composite gradient algorithm. Finally, numerical results are given to demonstrate the efficiency of the proposed AIPP and QP-AIPP variants.

2000 Mathematics Subject Classification: 47J22, 90C26, 90C30, 90C60, 65K10.

Key words: quadratic penalty method, nonconvex program, iteration-complexity, proximal point method, first-order accelerated methods.

1 Introduction

This paper presents a computationally efficient variant of the quadratic penalty accelerated inexact proximal point (QP-AIPP) method studied in [15].

The QP-AIPP method of [15] is designed for solving the linearly–constrained nonconvex composite optimization problem

$$\min \{f(z) + h(z) : Az = b,\ z \in \Re^n\}, \tag{1.1}$$

where $A : \Re^n \mapsto \Re^p$ is a linear operator, $b \in \Re^p$, $h : \Re^n \to (-\infty, \infty]$ is a closed proper convex function, and f is a real-valued differentiable (possibly nonconvex) function whose gradient is L–Lipschitz and which, for some $0 < m \leq L$, satisfies

$$f(u) \geq f(z) + \langle \nabla f(z), u - z\rangle - \frac{m}{2}\|u - z\|^2 \quad \forall\, z, u \in \text{dom } h. \tag{1.2}$$

Weiwei Kong · Renato D.C. Monteiro
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0205. (E-mail: wkong37@gatech.edu & monteiro@isye.gatech.edu). The works of these authors were partially supported by ONR Grant N00014-18-1-2077.

Jefferson G. Melo
Institute of Mathematics and Statistics, Federal University of Goias, Campus II- Caixa Postal 131, CEP 74001-970, Goiânia-GO, Brazil. (E-mail: jefferson@ufg.br). The work of this author was supported in part by CNPq Grant 406975/2016-7.

The QP-AIPP method solves (1.1) via a quadratic penalty method, i.e., a sequence of penalty subproblems of the form

$$\min\left\{f(z)+h(z)+\frac{c}{2}\|Az-b\|^2 : z\in\Re^n\right\}, \tag{1.3}$$

for an increasing sequence of positive penalty parameters c, is solved by the accelerated inexact proximal point (AIPP) method (discussed below) in which each penalty subproblem is solved using a common starting point $z_0 \in \operatorname{dom} h$ (i.e., a cold–start strategy is adopted).

We briefly outline the AIPP method of [15]. First, note that (1.3) is a special case of

$$\phi_* := \min\{\phi(z) := g(z)+h(z) : z\in\Re^n\} \tag{1.4}$$

where $g(z) := f(z)+c\|Az-b\|^2/2$ is a function satisfying

$$-\frac{m}{2}\|u-z\|^2 \le g(u)-[g(z)+\langle\nabla g(z),u-z\rangle] \le \frac{M}{2}\|u-z\|^2 \quad \forall\, z,u\in\operatorname{dom}\, h, \tag{1.5}$$

where $M = L+c\|A\|^2$. In the general setting of (1.4)–(1.5), the AIPP method generates a sequence $\{z_k\}$ using an inexact proximal point (IPP) framework (see for example [30,31]), i.e., given $z_{k-1}\in\operatorname{dom} h$, it computes z_k as a suitable approximate solution of the proximal subproblem

$$\min\left\{g(z)+h(z)+\frac{1}{2\lambda_k}\|z-z_{k-1}\|^2 : z\in\Re^n\right\} \tag{1.6}$$

for some prox-parameter $\lambda_k > 0$. Note that the first inequality in (1.5) implies that the objective function of (1.6) is convex as long as λ_k is not larger than $1/m$. The AIPP method sets $\lambda_k = 1/(2m)$ for every k and uses an accelerated composite gradient (ACG) variant (see for example [4, 22, 26]) to approximately solve (1.6).

Since the larger λ_k is the faster the above IPP framework converges to a desirable approximate solution, the main goal of this paper is to develop an aggressive AIPP variant, and subsequently an aggressive QP-AIPP variant, which possibly chooses λ_k substantially larger than $1/m$ despite potential loss of convexity of (1.6). An important ingredient in obtaining this aggressive AIPP variant is the development of a relaxed ACG (R-ACG) algorithm that approximately solves (1.6) according to a more relaxed termination criterion. More specifically, within a reasonably number of iterations, the algorithm: (i) either solves the possibly nonconvex subproblem (1.6) according to the relaxed criterion or stops with failure due to λ_k being too large; and (ii) always solves (1.6) according to the relaxed criterion when its objective function is convex. The aforementioned relaxed AIPP (R-AIPP) variant starts with a relatively large initial prox parameter and, in each one of its steps, calls the R-ACG algorithm to solve the corresponding prox subproblem. If a key descent inequality fails, then the prox parameter λ_k is halved, the prox center z_{k-1} is maintained, and the R-ACG algorithm is invoked once again to solve the resulting prox subproblem; otherwise, the prox parameter λ_k is preserved and z_k takes the place of z_{k-1}.

This paper also considers a more general version of (1.1) in which the linear constraint $Az = b$ is replaced by the linear set constraint $Az\in S$, where $S\subseteq\Re^p$ is a closed convex set. Clearly, when $S = \{b\}$, the more general problem reduces to (1.1). Under the assumption that $\operatorname{dom} h$ is bounded and all penalty subproblems are solved by the AIPP variant using the aforementioned cold–start strategy, it turns out that the iteration complexity of the QP-AIPP variant for finding the desired approximate solution is considerably worse than that of the QP-AIPP method of [15]. If, on the other hand, the QP-AIPP variant adopts the warm–start strategy in which the R-AIPP method for solving the current penalty subproblem starts from the approximate solution found for the previous subproblem, then the iteration complexity of this relaxed QP-AIPP (R-QP-AIPP) variant is shown to be the same as that of the QP-AIPP method of [15], up to a logarithmic factor.

The proposed AIPP and QP-AIPP variants are compared with three state-of-the-art optimization methods on five different optimization problems. The computational results obtained show that the variants can substantially outperform most of the competing methods on many problem instances.

Related works. We first discuss papers dealing with related algorithms for solving the convex version of (1.1) and other related monotone problems. Iteration-complexity analysis of quadratic penalty methods for solving (1.1) under the assumption that f is convex and h is a convex indicator function was first studied in [16] and further explored in [2, 25]. Iteration-complexity of first-order augmented Lagrangian methods for solving the latter class of linearly constrained convex programs was studied in [3, 17, 20, 21, 29, 32]. Inexact proximal point methods using accelerated gradient algorithms to solve their prox-subproblems were previously

considered in [7,12,13,14,24] in the setting of convex-concave saddle point problems and monotone variational inequalities.

We now discuss papers dealing with related algorithms for solving (1.1) when f is nonconvex and the assumptions mentioned after (1.1) hold. Paper [15] is, up to our knowledge, the first one to consider a proximal method with acceleration strategy for solving (1.1). Previous works using acceleration strategies were concerned with the unconstrained problem (1.4). Namely, [9] proposed an accelerated gradient framework to solve (1.4) with better iteration complexity than the usual composite gradient method. Since then, many authors have proposed other accelerated frameworks for solving (1.4) under different assumptions on the functions g and h (see, for example, [5, 8, 10, 18, 28]). In particular, by exploiting the lower curvature m, [5, 8, 28] proposed some algorithms which improve the iteration-complexity bound of [9] in terms of the dependence on the upper curvature M. Finally, there has been a growing interest in the iteration complexity of methods for solving optimization problems using second order information (see, for example, [5, 6, 23, 27]).

Organization of the paper. Subsection 1.1 provides some basic definitions and notation. Section 2 begins with presenting some background materials and transitions into defining a general descent (GD) framework for solving the nonconvex optimization problem (1.4). Section 3 presents and derives the complexity of an R-ACG algorithm which attempts to solve (1.6) even when it is not convex. Section 4 presents a relaxed variant of the AIPP method proposed in [15]. Section 5 presents a relaxed variant of the QP-AIPP method proposed in [15]. Section 6 presents numerical results to illustrate the efficiency of the AIPP and QP-AIPP variants. Finally, Section 7 presents some concluding remarks.

1.1 Basic definitions and notation

This subsection provides some basic definitions and notation used in this paper.

The set of natural numbers is denoted by $\mathbb{N}$. The set of real numbers is denoted by $\Re$. The set of non-negative real numbers and the set of positive real numbers are denoted by $\Re_+$ and $\Re_{++}$, respectively. Let $\Re^n$ denote a real valued n–dimension inner product space, whose inner product and its associated induced norm are denoted by $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$, respectively. Let $\langle\cdot,\cdot\rangle_F$ denote the Frobenius inner product. Let S^n_+ denote the cone of positive semidefinite n–by–n matrices. For $t > 0$, define $\log^+_1(t) := \max\{\log t, 1\}$. The set of proper lower semi-continuous convex functions defined on $\Re^n$ is denoted by $\overline{\mathrm{Conv}}(\Re^n)$. Given a linear operator $A : \Re^n \mapsto \Re^p$, the operator norm of A is denoted by $\|A\| := \sup\{\|Az\|/\|z\| : z \in \Re^n, z \neq 0\}$.

Let $\psi : \Re^n \to (-\infty, +\infty]$ be given. The effective domain of ψ is denoted by $\operatorname{dom}\psi := \{x \in \Re^n : \psi(x) < \infty\}$ and ψ is proper if $\operatorname{dom}\psi \neq \emptyset$. If ψ is differentiable at $\bar z \in \Re^n$, then its affine approximation $\ell_\psi(\cdot;\bar z)$ at $\bar z$ is denoted by

$$\ell_\psi(z;\bar z) := \psi(\bar z) + \langle \nabla\psi(\bar z), z - \bar z\rangle \quad \forall z \in \Re^n. \tag{1.7}$$

Also, for $\varepsilon \geq 0$, its *ε-subdifferential* at $z \in \operatorname{dom}\psi$ is denoted by

$$\partial_\varepsilon\psi(z) := \{v \in \Re^n : \psi(u) \geq \psi(z) + \langle v, u - z\rangle - \varepsilon, \forall u \in \Re^n\}. \tag{1.8}$$

The subdifferential of ψ at $z \in \operatorname{dom}\psi$, denoted by $\partial\psi(z)$, corresponds to $\partial_0\psi(z)$.

For a given $X \subseteq \Re^n$, the closure of the set X is denoted by $\operatorname{cl} X$, the indicator function of X, denoted by δ_X, is defined as $\delta_X(x) = 0$ if $x \in X$ and $\delta_X(x) = \infty$ if $x \notin X$. Moreover, the normal cone of X at a point $x \in X$ is denoted by

$$N_X(x) := \{u \in \Re^{n\times n} : \langle u, x' - x\rangle \leq 0, \forall x' \in X\} = \partial\delta_X(x).$$

2 A general descent framework

As discussed in Section 1, all the penalized subproblems (see (1.2)) that arise during the execution of the QP-AIPP method, as well as the R-QP-AIPP method, are of the form (1.4). Hence, efficiently obtaining a solution of (1.4) is of paramount importance for both the QP-AIPP and R-QP-AIPP methods. While the QP-AIPP method uses the AIPP method to solve (1.4), the R-QP-AIPP method uses the R-AIPP method which will be discussed in Section 4. The discussion of this section (as well as Section 3) will essentially pave the way towards the presentation of the R-AIPP method.

More specifically, this section presents and analyzes a GD framework for solving (1.4) that makes use of a black box (see step 1 of the GD framework below). In addition, it describes: the assumptions and relevant

quantities underlying problem (1.4), the notion of approximate stationary point of (1.4) adopted in this section and Section 4, and the relationship between the GD framework and the GIPP framework of [15], of which the AIPP method is an instance of.

Our problem of interest in this section and Section 4 is (1.4) which is assumed to satisfy the following assumptions:

(A1) $h \in \overline{\text{Conv}}(\Re^n)$;
(A2) g is a nonconvex differentiable function on $\text{dom}\, h$ and there exist a scalar $M > 0$ such that

$$\|\nabla g(u) - \nabla g(z)\| \le M\|u - z\| \quad \forall u, z \in \text{dom}\, h; \tag{2.1}$$

(A3) $\phi_* > -\infty$.

In addition, the analysis in Section 4 makes use of the quantity

$$\underline{m} := \inf\left\{m \in \Re_{++} : g(u) \ge \ell_g(u; z) - \frac{m}{2}\|u - z\|^2 \quad \forall u, z \in \text{dom}\, h\right\}, \tag{2.2}$$

which is positive in view of assumption (A2). While it is generally difficult to compute the above quantity, it is well known that assumption (A2) implies that $\underline{m} \in (0, M]$. Moreover, it is shown in Proposition 4.2 below that the smaller $\underline{m}$ is, the better the iteration complexity of R-AIPP method in Section 4 becomes.

It is well-known that a necessary condition for $z^* \in \text{dom}\, h$ to be a local minimum of (1.4) is that z^* be a stationary point of ϕ, i.e., $0 \in \nabla g(z^*) + \partial h(z^*)$. A relaxation of this inclusion leads to the following definition of an approximate stationary point of (1.4): given a tolerance $\hat\rho > 0$, a pair $(\hat z, \hat v)$ is said to be a $\hat\rho$–approximate stationary point of (1.4) if

$$\hat v \in \nabla g(\hat z) + \partial h(\hat z), \qquad \|\hat v\| \le \hat\rho. \tag{2.3}$$

Given a general quadruple $(\lambda, z^-, z, v) \in \Re_{++} \times \Re^n \times \text{dom}\, h \times \Re^n$, the following simple refinement procedure shows how to obtain a pair $(\hat z, \hat v)$ satisfying the inclusion in (2.3) with a technically useful bound on the residual $\hat v$ (see Proposition 2.1 below).

Refinement procedure.

Input: a scalar $M > 0$, a pair of functions (g, h) satisfying assumptions (A1) and (A2), and a quadruple $(\lambda, z^-, z, v) \in \Re_{++} \times \Re^n \times \text{dom}\, h \times \Re^n$;

Output: a triple $(\hat z, \hat v, \Delta) \in \text{dom}\, h \times \Re^n \times \Re_{++}$ satisfying (2.8);

(0) set

$$M_\lambda := \lambda M + 1, \quad f_\lambda := \lambda g + \frac{1}{2}\|\cdot - z^-\|^2 - \langle v, \cdot\rangle, \quad h_\lambda := \lambda h; \tag{2.4}$$

(1) compute

$$\hat z := \arg\min_u \left\{\langle \nabla f_\lambda(z), u - z\rangle + \frac{M_\lambda}{2}\|u - z\|^2 + h_\lambda(u)\right\}, \tag{2.5}$$

$$\hat v := \frac{1}{\lambda}\left[(v + z^- - z) + M_\lambda(z - \hat z)\right] + \nabla g(\hat z) - \nabla g(z), \tag{2.6}$$

$$\Delta := (f_\lambda + h_\lambda)(z) - (f_\lambda + h_\lambda)(\hat z); \tag{2.7}$$

(2) return the triple $(\hat z, \hat v, \Delta)$.

For the sake of brevity, we write $(\hat z, \hat v, \Delta) = RP(\lambda, z^-, z, v)$ to indicate that the triple $(\hat z, \hat v, \Delta)$ is the output of the above refinement procedure with inputs M, (g, h), and (λ, z^-, z, v). We now state an important property of this procedure, whose proof can be found in Appendix A.

Proposition 2.1 *Let a pair of functions (g, h) satisfying (A1)–(A3) and a quadruple $(\lambda, z^-, z, v) \in \Re_{++} \times \Re^n \times \text{dom}\, h \times \Re^n$ be given and let $(\hat z, \hat v, \Delta) = RP(\lambda, z^-, z, v)$. Then, $\Delta \ge 0$ and*

$$\hat v \in \nabla g(\hat z) + \partial h(\hat z), \quad \lambda\|\hat v\| \le \|v + z^- - z\| + 2\sqrt{2M_\lambda \Delta} \tag{2.8}$$

where M_λ is as in (2.4).

The above proposition shows that the pair $(\hat{z}, \hat{v})$, computed as in (2.5) and (2.6), clearly satisfies the inclusion in (2.3) and that the quantity $\lambda\|\hat{v}\|$ has an upper bound expressed in terms of the two quantities: $\|v + z^- - z\|$ and $\sqrt{M_\lambda \Delta}$. Given a tolerance $\hat{\rho} > 0$, it will be shown in Proposition 2.2 below that the GD framework stated next generates a sequence of iterates $\{(\lambda_k, z_k, v_k)\}$ whose corresponding refined sequence $\{(\hat{z}_k, \hat{v}_k)\}$ obtained as $(\hat{z}_k, \hat{v}_k) = RP(\lambda_k, z_{k-1}, z_k, v_k)$ for every $k \geq 1$ yields a $\hat{\rho}$–approximate stationary point of (1.4).

GD framework.

Input: a scalar $M > 0$, a function pair (g, h) satisfying assumptions (A1)–(A3), an initial point $z_0 \in \operatorname{dom} h$, and a scalar pair $(\theta, \tau) \in \Re^2_{++}$;

(0) set $\phi := g + h$ and $k = 1$;
(1) find a triple $(\lambda_k, z_k, v_k) \in \Re_{++} \times \operatorname{dom} h \times \Re^n$ such that its corresponding refined triple

$$(\hat{z}_k, \hat{v}_k, \Delta_k) = RP(\lambda_k, z_{k-1}, z_k, v_k)$$

satisfies

$$\|v_k + z_{k-1} - z_k\|^2 \leq \theta\lambda_k[\phi(z_{k-1}) - \phi(z_k)], \tag{2.9}$$

$$2\left(\lambda_k M + 1\right)\Delta_k \leq \tau\|v_k + z_{k-1} - z_k\|^2; \tag{2.10}$$

(2) set $k = k + 1$ and go to step 1.

We now make three remarks about the GD framework. First, no termination criterion is added to the GD framework so as to be able to discuss convergence rate results about its generated sequence. A discussion of how to terminate it is given after Proposition 2.2 below. Second, step 1 should be viewed as an oracle in that it does not specify how to compute the triple (λ_k, z_k, v_k). Third, Corollary 2.5 below shows that if the stepsize λ_k is chosen so that the prox subproblem (1.6) is a strongly convex composite problem, i.e., $\lambda_k \in (0, 1/\underline{m})$ where $\underline{m}$ is as in (2.2), the point z_k is chosen as its unique optimal solution, and v_k is set to zero, then the triple (λ_k, z_k, v_k) satisfies (2.9) and (2.10) with $\theta = 2$ and $\tau = 0$. Thus, when $(\theta, \tau) \in [2, \infty) \times [0, \infty)$, we conclude that: (i) there always exists a triple satisfying (2.9) and (2.10); and, (ii) the GD framework can be viewed as an IPP method. Fourth, the R-AIPP of Section 4, being a special instance of the GD framework, can also be viewed as an IPP method which chooses (θ, τ) in the open rectangle $(2, \infty) \times (0, \infty)$ and applies an ACG variant, such as the one described in Section 3, to problem (1.6) in order to obtain a triple (λ_k, z_k, v_k) satisfying (2.9) and (2.10).

The following result shows an important property about the sequence of iterates $\{(\lambda_k, \hat{z}_k, \hat{v}_k)\}$.

Proposition 2.2 *The sequences of stepsizes $\{\lambda_k\}$ and iterate pairs $\{(\hat{z}_k, \hat{v}_k)\}$ satisfy*

$$\hat{v}_k \in \nabla g(\hat{z}_k) + \partial h(\hat{z}_k), \quad \min_{i \leq k} \|\hat{v}_i\|^2 \leq \theta\left(1 + 2\sqrt{\tau}\right)^2 \frac{[\phi(z_0) - \phi_*]}{\Lambda_k}, \tag{2.11}$$

for every $k \geq 1$, where $\Lambda_k := \sum_{i=1}^k \lambda_i$.

Proof Let $k \geq 1$ be fixed. The inclusion in (2.11) follows from Proposition 2.1 with $(\hat{z}, \hat{v}) = (\hat{z}_k, \hat{v}_k)$ and the definitions of $\hat{z}_k$ and $\hat{v}_k$ in step 1 of the GD framework. To show the inequality in (2.11), first observe that (2.9) and the definition of ϕ_* in (1.4) implies that

$$\phi(z_0) - \phi_* \geq \sum_{i=1}^k [\phi(z_{i-1}) - \phi(z_i)] \geq \sum_{i=1}^k \frac{\|v_i + z_{i-1} - z_i\|^2}{\theta\lambda_i} \geq \frac{\Lambda_k}{\theta} \min_{i \leq k} \frac{1}{\lambda_i^2} \|v_i + z_{i-1} - z_i\|^2. \tag{2.12}$$

Now, let $i \geq 1$ be arbitrary. In view of step 1 of the GD framework we have $(\hat{z}_i, \hat{v}_i, \Delta_i) = RP(\lambda_i, z_{i-1}, z_i, v_i)$. Hence Proposition 2.1 with $(\lambda, z^-, z, v, \hat{v}) = (\lambda_i, z_{i-1}, z_i, v_i, \hat{v}_i)$ and (2.10) with $k = i$ imply that

$$\|\hat{v}_i\| \leq \left(1 + 2\sqrt{\tau}\right) \frac{\|v_i + z_{i-1} - z_i\|}{\lambda_i}. \tag{2.13}$$

The inequality in (2.11) now follows by combining (2.12) and (2.13). ∎

We now make three remarks about the GD framework in light of Proposition 2.2. First, if the GD framework stops when a pair $(\hat{z}_k, \hat{v}_k)$ such that $\|\hat{v}_k\| \leq \hat{\rho}$ is found, then it follows from (2.3) and the inclusion in (2.11) that $(\hat{z}_k, \hat{v}_k)$ is a $\hat{\rho}$–approximate stationary point of (1.4). Second, if the sequence of stepsizes $\{\lambda_i\}$ satisfies $\lim_{k\to\infty} \Lambda_k = \infty$, then it follows from the inequality in (2.11) and assumption (A3) that the GD framework indeed stops according to the above termination criterion. Third, (2.11) indicates that the larger the stepsizes λ_k are, the faster the quantity $\min_{i\leq k} \|\hat{v}_i\|$ approaches zero.

For the remainder of this section, our goal is to show that the GD framework can be seen as a relaxation of the GIPP framework studied in [15]. The proof of this fact is not essential in establishing any results pertaining to the R-AIPP method in Section 4 or the R-QP-AIPP method in Section 5 and may skipped without any loss of continuity.

Recall that, for a given $z_0 \in \operatorname{dom} h$ and $\sigma \in [0, 1)$, the GIPP framework in [15] considers a sequence $\{(\lambda_k, z_k, v_k, \varepsilon_k)\} \subseteq \Re_{++} \times \operatorname{dom}\phi \times \Re^n \times \Re_+$ satisfying

$$v_k \in \partial_{\varepsilon_k}\left(\lambda_k\phi + \frac{1}{2}\|\cdot - z_{k-1}\|^2\right)(z_k), \quad \|v_k\|^2 + 2\varepsilon_k \leq \sigma\|v_k + z_{k-1} - z_k\|^2, \tag{2.14}$$

for every $k \geq 1$. We now state a simple technical result which will not only be used in this section but also later in the analysis of the R-ACG algorithm (see Section 3).

Lemma 2.3 *Assume that $\varepsilon \geq 0$ and $(\lambda, z^-, z, v) \in \Re_{++} \times \Re^n \times \operatorname{dom} h \times \Re^n$ satisfy*

$$v \in \partial_\varepsilon\left(\lambda\phi + \frac{1}{2}\|\cdot - z^-\|^2\right)(z). \tag{2.15}$$

Then, the quantity Δ defined in (2.7) *satisfies $\Delta \leq \varepsilon$.*

Proof Let $(\hat{z}, \Delta)$ be computed as in (2.5) and (2.7). It follows from (1.8) and (2.15) that

$$\lambda\phi(z') + \frac{1}{2}\|z' - z^-\|^2 \geq \lambda\phi(z) + \frac{1}{2}\|z - z^-\|^2 + \langle v, z' - z\rangle - \varepsilon \quad \forall z' \in \Re^n.$$

Considering the above inequality at the point $z' = \hat{z}$, along with some algebraic manipulation, we have

$$\varepsilon \geq \left[\lambda\phi(z) + \frac{1}{2}\|z - z^-\|^2 - \langle v, z\rangle\right] - \left[\lambda\phi(\hat{z}) + \frac{1}{2}\|\hat{z} - z^-\|^2 - \langle v, \hat{z}\rangle\right] = \Delta$$

where the last equality is due to the definitions of ϕ and Δ given in (1.4) and (2.7), respectively. ∎

The following result shows the relationship between the GIPP framework of [15] and the GD framework of this section.

Proposition 2.4 *If, for some $z_{k-1} \in \operatorname{dom} h$, constant $\sigma \in [0, 1)$, and index $k \geq 1$, the quadruple $(\lambda_k, z_k, v_k, \varepsilon_k)$ satisfies* (2.14)*, then (λ_k, z_k, v_k) satisfies* (2.9) *and* (2.10) *for any $\theta \geq 2/(1-\sigma)$ and $\tau \geq \sigma(\lambda_k M + 1)$. As a consequence, if $\sup\{\lambda_k : k \geq 1\} < \infty$, then every instance of the GIPP framework is an instance of the GD framework for any (θ, τ) satisfying*

$$\theta \geq \frac{2}{1-\sigma}, \quad \tau \geq \sup\left\{\sigma(\lambda_k M + 1) : k \geq 1\right\}. \tag{2.16}$$

Proof The proof that (λ_k, z_k, v_k) satisfies (2.9) with $\theta = 2/(1-\sigma)$ can be found in [15, Proposition 5(a)]. Now, let $k \geq 1$ and observe that from Lemma 2.3 with $(\lambda, z^-, z, v) = (\lambda_k, z_{k-1}, z_k, v_k)$ and $\varepsilon = \varepsilon_k$ we have $\Delta \leq \varepsilon_k$. It follows from the last inequality and the inequality in (2.14) that $2\Delta \leq \sigma\|v_k + z_{k-1} - z_k\|^2$. Combining the previous inequality with the assumption on τ now shows that (λ_k, z_k, v_k) satisfies (2.10). The second part of the proposition follows immediately from the first part and condition (2.16). ∎

The above proposition shows that if $\{\lambda_k\}$ is bounded and the parameter triple (σ, θ, τ) satisfies (2.16), then the condition for finding an iterate (λ_k, z_k, v_k) in the GD framework is more relaxed than the condition for finding an iterate $(\lambda_k, z_k, v_k, \varepsilon_k)$ in the GIPP framework. As a consequence, under the conditions in (2.16), an optimization algorithm (such as the R-ACG algorithm of Section 3) applied to (1.6) is expected to find the triple (λ_k, z_k, v_k) for the GD framework faster than the quadruple $(\lambda_k, z_k, v_k, \varepsilon_k)$ for the GIPP framework.

The following corollary justifies the third remark following the GD framework.

Corollary 2.5 *Let $z_{k-1} \in \operatorname{dom} h$ and $\lambda_k \in (0, 1/\underline{m})$ be given, where $\underline{m}$ is as in* (2.2)*. Then,* (1.6) *has a unique global minimum z_k and the triple $(\lambda_k, z_k, v_k) \in \Re_{++} \times \operatorname{dom} h \times \Re^n$ where $v_k = 0$ satisfies* (2.9) *and* (2.10) *with $\theta = 2$ and $\tau = 0$.*

Proof The existence and unique uniqueness of z_k follows from the fact that $\phi + \|\cdot - z_{k-1}\|^2/\lambda_k$ is strongly convex. Moreover, the fact that z_k is the unique global minimum of (1.6) implies that the quadruple $(\lambda_k, z_k, v_k, \varepsilon_k)$, where $(v_k, \varepsilon_k) = (0, 0)$, satisfies (2.14) with $\sigma = 0$. The conclusion of the corollary now follows immediately from the first part of Proposition 2.4 with $\sigma = 0$. ∎

3 A relaxed accelerated composite gradient algorithm

This section presents and analyzes an ACG variant, namely, the R-ACG algorithm, which is used as an important tool in the development of the R-AIPP method of Section 4. More specifically, the R-AIPP method can be viewed as a special instance of the GD framework where step 1 is implemented by repeatedly calling the ACG variant of this section.

Before describing the variant, we consider its assumptions as well as the problem that it solves. First, we describe the assumptions. Let $\widetilde{\phi} : \Re^n \to (-\infty, \infty]$ be given and assume that it can be decomposed as $\widetilde{\phi} = \widetilde{\phi}^{(s)} + \widetilde{\phi}^{(n)}$ where:

(B1) $\widetilde{\phi}^{(n)} \in \overline{\operatorname{Conv}}(\Re^n)$;

(B2) $\widetilde{\phi}^{(s)}$ is a differentiable function on $\operatorname{dom} \widetilde{\phi}^{(n)}$ such that for some $\widetilde{M} > 0$,

$$\widetilde{\phi}^{(s)}(u) \le \ell_{\widetilde{\phi}^{(s)}}(u; x) + \frac{\widetilde{M}}{2}\|u - x\|^2 \quad \forall u, x \in \operatorname{dom} \psi^{(n)}.$$

We now describe our problem of interest in this section.

Problem A: Given $\widetilde{\phi} : \Re^n \to (-\infty, +\infty]$ satisfying the above assumptions, a point $x_0 \in \Re^n$, and a pair of parameters $(\theta, \tau) \in (2, \infty) \times (0, \infty)$, the problem is to find a triple $(x, u, \eta) \in \Re^n \times \Re^n \times \Re_+$ such that

$$\|x_0 - x + u\|^2 \le \theta \left[\widetilde{\phi}(x_0) - \widetilde{\phi}(x)\right], \tag{3.1}$$

$$u \in \partial_\eta \left(\widetilde{\phi} + \frac{1}{2}\|\cdot - x_0\|^2\right)(x), \quad 2\left(\widetilde{M} + 1\right)\eta \le \tau\|x_0 - x + u\|^2. \tag{3.2}$$

The following simple result shows how the ability to solve Problem A allows us to implement the "step 1" oracle in the GD framework.

Proposition 3.1 *Assume that $\phi = g + h$ satisfies conditions (A1) and (A2), and let $z_{k-1} \in \operatorname{dom} h$ be given. Then the following statements hold:*

(a) if (x, u) satisfies (3.1) *with $(\widetilde{\phi}, \widetilde{M}, x_0) = (\lambda\phi, \lambda M, z_{k-1})$ for some $\lambda > 0$, then the triple $(\lambda_k, z_k, v_k) := (\lambda, x, u)$ satisfies* (2.9)*;*

(b) if (x, u, η) solves Problem A with input $(\widetilde{\phi}, \widetilde{M}, x_0) = (\lambda\phi, \lambda M, z_{k-1})$ for some $\lambda > 0$, then the triple $(\lambda_k, z_k, v_k) = (\lambda, x, u)$ solves step 1 of the GD framework.

Proof (a) Assume that (x, u) satisfies (3.1). It follows from the fact that $(\lambda, x, u) = (\lambda_k, z_k, v_k)$ and the definition of $\widetilde{\phi}$ that

$$\|z_{k-1} - z_k + v_k\|^2 \le \theta\left[\widetilde{\phi}(z_{k-1}) - \widetilde{\phi}(z_k)\right] = \theta\lambda_k\left[\phi(z_{k-1}) - \phi(z_k)\right]$$

and thus the triple (λ_k, z_k, v_k) satisfies (2.9).

(b) Assume that (x, u, η) satisfies (3.2) and define $\varepsilon := \eta$ and $(z^-, z, v) := (x_0, x, u)$. Moreover, let Δ be computed as in (2.7) with $\hat{z}$ as in (2.5). It follows from Lemma 2.3, the definition of $\widetilde{\phi}$, the fact that $\eta = \varepsilon$, and the inclusion in (3.2) that $\Delta \le \eta$. Using the inequality in (3.2) and the fact that $(x_0, x, u) = (z_{k-1}, z_k, v_k)$ gives $2(\widetilde{M} + 1)\Delta \le \tau\|z_{k-1} - z_k + v_k\|^2$ and thus the pair (z_k, v_k) satisfies (2.10) in view of the definition of $\widetilde{M}$. As a consequence, the triple (λ_k, z_k, v_k) solves step 1 of the GD framework. ∎

The R-ACG algorithm presented below, which is a modified ACG variant for minimizing the function $\psi := \widetilde{\phi} + \|\cdot - x_0\|^2/2$, solves Problem A under the assumption that ψ is convex (see Proposition 3.2(c) below). As a consequence, it can be used to implement step 1 of the GD framework whenever λ_k is sufficiently small. More specifically, since $\lambda_k\phi + \|\cdot - z_{k-1}\|^2/2$ is clearly convex whenever λ_k is chosen in $(0, 1/\underline{m}]$, where $\underline{m}$ is as in (2.2), we can use the R-ACG algorithm to solve problem A with $\widetilde{\phi} = \lambda_k\phi$ and $x_0 = z_{k-1}$, and hence the "step 1" oracle in the GD framework in view of Proposition 3.1(b). In fact, the AIPP method developed in [15] is an instance of the GIPP framework (and hence an instance of the GD framework) in which, given an upper bound m on $\underline{m}$, it chooses $\lambda_k = 1/(2m)$ for all k and in which step 1 is implemented with a single call to the R-ACG algorithm presented below.

However, our main goal in this paper is the development of an instance of the GD framework which aggressively chooses λ_k (possibly) much larger than $1/\underline{m}$ since, according to Proposition 2.2, this strategy can potentially reduce its number of iterations. In this regard, the R-ACG algorithm presented below accepts as input a function $\widetilde{\phi}$ of the form $\widetilde{\phi} = \lambda\phi$ for some $\lambda > 0$ in which $\widetilde{\phi} + \|\cdot - x_0\|^2/2$ is not necessarily convex, and terminates with either failure or by finding a triple (x, u, η) satisfying (3.1) within $\mathcal{O}(\widetilde{M}^{1/2}\log_1^+ \widetilde{M})$ iterations (see statements (a) and (b) of Proposition 3.2 below). Clearly, in the second case, the triple $(\lambda_k, z_k, v_k) = (\lambda, x, u)$ is guaranteed to satisfy (2.9) but not necessarily (2.10) (see Proposition 3.1(a)). If (2.10) is satisfied then the R-ACG algorithm clearly provides a solution of the "step 1" oracle of the GD framework; otherwise, the stepsize λ is considered large. The R-AIPP method of Section 4 is an instance of the GD framework which attempts to provide a solution of its "step 1" oracle in this manner and adaptively reduces λ whenever it is found to be large.

R-ACG algorithm.

Input: a scalar $\widetilde{M} > 0$, a function pair $(\widetilde{\phi}^{(s)}, \widetilde{\phi}^{(n)})$ satisfying assumptions (B1) and (B2), an initial point $x_0 \in \operatorname{dom}\widetilde{\phi}^{(n)}$, and a pair of parameters $(\theta, \tau) \in (2, \infty) \times (0, \infty)$;

Output: a triple $(x, u, \eta) \in \operatorname{dom}\widetilde{\phi}^{(n)} \times \Re^n \times \Re_+$ satisfying (3.1) or a **failure** status;

(0) set $y_0 = x_0$, $A_0 = 0$, $\Gamma_0 \equiv 0$, $j = 1$, and define

$$\psi^{(s)} := \widetilde{\phi}^{(s)} + \frac{1}{4}\|\cdot - x_0\|^2, \qquad \psi^{(n)} := \widetilde{\phi}^{(n)} + \frac{1}{4}\|\cdot - x_0\|^2, \qquad \psi := \psi^{(s)} + \psi^{(n)}, \tag{3.3}$$

$$\widetilde{L} := \widetilde{M} + \frac{1}{2}, \qquad \mu := \frac{1}{2}; \tag{3.4}$$

(1) compute

$$A_j = A_{j-1} + \frac{\mu A_{j-1} + 1 + \sqrt{(\mu A_{j-1}+1)^2 + 4\widetilde{L}(\mu A_{j-1}+1)A_{j-1}}}{2\widetilde{L}}, \tag{3.5}$$

$$\widetilde{x}_{j-1} = \frac{A_{j-1}}{A_j}x_{j-1} + \frac{A_j - A_{j-1}}{A_j}y_{j-1}, \quad \Gamma_j = \frac{A_{j-1}}{A_j}\Gamma_{j-1} + \frac{A_j - A_{j-1}}{A_j}\ell_{\psi^{(s)}}(\cdot, \widetilde{x}_{j-1}), \tag{3.6}$$

$$y_j = \arg\min_y \left\{\Gamma_j(y) + \psi^{(n)}(y) + \frac{1}{2A_j}\|y - y_0\|^2\right\}, \tag{3.7}$$

$$x_j = \frac{A_{j-1}}{A_j}x_{j-1} + \frac{A_j - A_{j-1}}{A_j}y_j \tag{3.8}$$

and set

$$u_j = \frac{y_0 - y_j}{A_j}, \quad \eta_j = \max\left\{\psi(x_j) - \Gamma_j(y_j) - \psi^{(n)}(y_j) - \langle u_j, x_j - y_j\rangle, 0\right\}; \tag{3.9}$$

(2) if both inequalities

$$\|A_j u_j + x_j - x_0\|^2 + 2A_j\eta_j \le \|x_j - x_0\|^2, \tag{3.10}$$

$$\psi(x_0) \ge \psi(x_j) + \langle u_j, x_0 - x_j\rangle - \eta_j, \tag{3.11}$$

hold, then go to step 3; otherwise, stop with **failure**;

(3) if both inequalities

$$2\left(\widetilde{M}+1\right)\eta_j \le \tau\|x_0 - x_j + u_j\|^2, \tag{3.12}$$

$$\|x_0 - x_j + u_j\|^2 \le \theta\left[\widetilde{\phi}(x_0) - \widetilde{\phi}(x_j)\right], \tag{3.13}$$

hold, then return $(x, u, \eta) = (x_j, u_j, \eta_j)$; otherwise, increment $j = j + 1$ and go to step 1.

Some comments about the above algorithm are in order. First, step 1 is essentially a standard step of an ACG variant (see, for example, [13, 15]) applied to the problem $\min\{\widetilde{\phi}(x) + \|x - x_0\|^2/2 : x \in \Re^n\}$ with the exception that it also computes in (3.9) the quantities u_j and η_j which, together with x_j, determine the termination criteria for the method. Second, it is shown in [15, Lemma 9] that a simplified version of the above algorithm, namely, one that does not include the two tests performed in step 2 and stops whenever the inequality in (2.14) is satisfied with $(z_{k-1}, z_k, v_k, \varepsilon_k) = (x_0, x_j, u_j, \eta_j)$, implements step 1 of the GIPP framework in [15]. Finally, it is well-known (see, for example, [13, Proposition 2.3]) that the scalar A_j updated according to (3.5) satisfies

$$A_j \ge \frac{1}{L}\max\left\{\frac{j^2}{4}, \left(1 + \sqrt{\frac{\mu}{4\widetilde{L}}}\right)^{2(j-1)}\right\} \quad \forall j \ge 1. \tag{3.14}$$

The next result establishes the iteration-complexity bound and some properties of the R-ACG algorithm.

Proposition 3.2 *The R-ACG algorithm satisfies the following statements:*

(a) it stops (either with success or failure) in at most

$$\left\lceil 1 + \left(\sqrt{2\widetilde{M}+1}\right)\log_1^+\left(C\left[2\widetilde{M}+1\right]\right)\right\rceil \tag{3.15}$$

iterations, where

$$C := \max\left\{\left[1 + \sqrt{\frac{\widetilde{M}+1}{\tau}}\right]^2, \left[1 + \sqrt{\frac{\theta}{\theta-2}}\right]^2\right\}; \tag{3.16}$$

(b) if it stops with success then its output (x, u, η) satisfies

$$\|x_0 - x + u\|^2 \le \theta\left[\widetilde{\phi}(x_0) - \widetilde{\phi}(x)\right]; \tag{3.17}$$

(c) if $\widetilde{\phi}^{(s)} + \|\cdot - x_0\|^2/2$ is convex then it always terminates with success and its output (x, u, η) solves Problem A.

Proof (a) See Appendix A.2.

(b) This follows from the fact that when the R-ACG algorithm stops with success, the last iterate $(x, u) = (x_j, u_j)$ satisfies (3.13).

(c) It follows from [15, Proposition 8(c)] that if $\widetilde{\phi}^{(s)} + \|\cdot - x_0\|^2/2$ is convex, then the iterate (x_j, u_j, η_j, A_j) satisfies (3.10) and the inclusion $u_j \in \partial_{\eta_j}(\widetilde{\phi} + \|\cdot - x_0\|^2/2)(x_j)$ for every $j \ge 1$. Hence, since the aforementioned inclusion and the definition of ψ in (3.3) imply (3.11), we conclude that the R-ACG algorithm does not terminate with failure (see step 2). As a consequence, it follows from statement (a) that it must terminate with success. It then follows from the previous inclusion, and the fact that the last iterate $(x, u, \eta) := (x_j, u_j, \eta_j)$ satisfies (3.12), that η fulfills (3.2). ∎

4 A relaxed accelerated inexact proximal point method

This section states and analyzes a relaxed variant of the AIPP method proposed in [15], namely, the R-AIPP method, for computing an approximate stationary point of (1.4) as in (2.3).

The R-AIPP method stated below is an instance of the GD framework which implements its step 1 by repeatedly invoking the ACG variant in Section 3 and thereby generates the method's iteration sequence. More specifically, if z_{k-1} denotes the previous iterate in the GD framework and $\lambda := \lambda_k$ then the R-ACG algorithm is invoked to attempt to solve Problem A with curvature $\widetilde{M}$, function pair $(\widetilde{\phi}^{(s)}, \widetilde{\phi}^{(n)})$, and initial point x_0 given by

$$\widetilde{M} = \lambda M, \quad \widetilde{\phi}^{(s)} = \lambda g, \quad \widetilde{\phi}^{(n)} = \lambda h, \quad x_0 = z_{k-1}.$$

If it succeeds, it obtains a pair (x, u) which will satisfy condition (3.1) of Problem A. Consequently, if the triple $(\lambda_k, z_k, v_k) = (\lambda, x, u)$ satisfies (2.10), then it is a solution of step 1 of the GD framework. If the R-ACG algorithm declares failure or the triple does not satisfy (2.10), then the stepsize λ is reduced and the above procedure is repeated.

R-AIPP method.

Input: a tolerance $\hat{\rho} > 0$, a scalar $M > 0$, a function pair (g, h) satisfying assumptions (A1)–(A3), an initial point $z_0 \in \operatorname{dom} h$, a scalar $\lambda_0 > 0$, and a pair of parameters $(\theta, \tau) \in (2, \infty) \times (0, \infty)$;

Output: a pair $(\hat{z}, \hat{v}) \in \operatorname{dom} h \times \Re^n$ satisfying (2.3);

(0) set $\lambda = \lambda_0$ and $k = 1$;

(1) apply the R-ACG algorithm to **Problem A** in Section 3 with inputs $\widetilde{M}$, $(\widetilde{\phi}^{(s)}, \widetilde{\phi}^{(n)})$, x_0, and (θ, τ), where

$$\widetilde{M} := \lambda M, \quad \widetilde{\phi}^{(s)} := \lambda g, \quad \widetilde{\phi}^{(n)} := \lambda h, \quad x_0 := z_{k-1}; \tag{4.1}$$

if the R-ACG algorithm stops with **failure** then set $\lambda = \lambda/2$ and repeat this step; otherwise, let (x, u, η) denote its output triple and go to step 2;

(2) compute $(\hat{z}, \hat{v}, \Delta) = RP(\lambda, z_{k-1}, x, u)$ through the refinement procedure; if

$$2(\lambda M + 1)\Delta > \tau \|u + z_{k-1} - x\|^2,$$

then set $\lambda = \lambda/2$ and go to step 1; otherwise, set

$$(\lambda_k, z_k, v_k) = (\lambda, x, u), \quad (\hat{z}_k, \hat{v}_k) = (\hat{z}, \hat{v}),$$

and go to step 3;

(3) if $\hat{v}_k$ satisfies

$$\|\hat{v}_k\| \le \hat{\rho}, \tag{4.2}$$

then return $(\hat{z}, \hat{v}) = (\hat{z}_k, \hat{v}_k)$; otherwise, increment $k = k + 1$ and go to step 1;

We now give some comments about the above method. First, it performs two types of iterations, namely, the outer iterations which are indexed by k and the inner ones which are performed by the R-ACG algorithm every time it is called in step 1. Second, if the call to the R-ACG algorithm in step 1 does not stop with failure then, by Proposition 3.2(b), the triple (x, u, η) output by the R-ACG algorithm together with the stepsize λ will satisfy (3.17) where $\widetilde{\phi} = \lambda(g + h)$. Hence, by Proposition 3.1(a), the triple $(\lambda_k, z_k, v_k) := (\lambda, x, u)$ will satisfy (2.9). If λ is also not halved in step 2 then the definition of $\widetilde{M}$ and Proposition 3.1(b) imply that the triple (λ_k, z_k, v_k) also satisfies (2.10). As a consequence, a single iteration of the R-AIPP method implements step 1 of the GD framework. Third, the termination condition (4.2) and Proposition 2.1, with $(\lambda, z^-, z, v) = (\lambda_k, z_{k-1}, z_k, v_k)$, imply that the required solution, i.e., a pair $(\hat{z}, \hat{v})$ satisfying (2.3), is obtained when the R-AIPP method terminates. Fourth, since the R-AIPP iterates implement step 1 of GD framework, and the sequence $\{\lambda_k\}$ is bounded below (see Lemma 4.1(b) below), Proposition 2.2 implies that the sequence $\{\hat{v}_k\}$ generated by the R-AIPP method has a subsequence approaching zero, and thus the method must terminate in step 3. Fifth, although the R-AIPP method does not necessarily generate proximal subproblems with convex objective functions, it is shown in Proposition 4.2 below that it has an iteration-complexity similar

to that of the AIPP method of [15]. Finally, in contrast to the aforementioned AIPP method, the R-AIPP neither requires an upper bound on the quantity $\underline{m}$ in (2.2) as part of its input nor does it place any restriction on the initial stepsize λ_0.

Each iteration of the R-AIPP method may call the R-ACG algorithm multiple times (possibly just one time). Invocations of the R-ACG algorithm that stop with success are said to be of type S while the other invocations are said to be of type O. Let K_S (resp., K_O) denote the total number of R-ACG calls of type S (resp., type O). The following technical result provides some basic facts about K_S, K_O and the sequence of stepsizes $\{\lambda_k\}$.

Lemma 4.1 *The following statements hold for the R-AIPP method:*

(a) if the stepsize $\lambda_{\bar{k}} \leq 1/(2\underline{m})$ for some $\bar{k} \geq 1$, then every iteration $k \geq \bar{k}$ is of type S and, as a consequence, $\lambda_k = \lambda_{\bar{k}}$ for every $k > \bar{k}$;
(b) K_O can be bounded as $2^{K_O} \leq \max\{1, 4\lambda_0\underline{m}\}$;
(c) $\{\lambda_k\}$ is non-increasing and satisfies $1/\lambda_k \leq \max\{1/\lambda_0, 4\underline{m}\}$ for all $k \geq 1$.

Proof (a) Since $\lambda_{\bar{k}} \leq 1/(2\underline{m})$, the definition of $\underline{m}$ in (2.2) implies that $\widetilde{\phi}^{(s)} + \|\cdot - z_{k-1}\|^2/2$ is convex, where $\widetilde{\phi}^{(s)}$ is as defined in (4.1) with $\lambda := \lambda_{\bar{k}}$. Hence, Proposition 3.2(c) together with Proposition 3.1(b) imply that step 1 and step 2 do not halve λ at the $\bar{k}^{\text{th}}$ iteration, which is to say that this iteration is of type S. Since $\{\lambda_k\}$ is clearly nonincreasing, the same conclusion holds true for every iteration $k \geq \bar{k}$. Moreover, as λ is not halved for subsequent iterations following $\bar{k}$, it follows that $\lambda_k = \lambda_{\bar{k}}$ for every $k > \bar{k}$.

(b) Using the fact that immediately before each iteration of type O, the stepsize λ is halved, we see that the condition $\lambda_{\bar{k}} \leq 1/(2m)$ in part (a) would eventually be satisfied for some iteration $\bar{k} \geq 1$, and hence K_O is finite. Now, note that if $K_O = 0$ then the inequality in part (b) follows immediately. Assume then that $K_O \geq 1$. It now follows from part (a) and the definition of K_O that $\lambda_0/2^{K_O-1} > 1/(2\underline{m})$, which clearly implies the inequality in part (b).

(c) The first statement follows trivially from the update rule of λ_k in the R-AIPP method. Now, note that the definition of K_O together with the update rule for λ_k imply, for every $k \geq 1$, that $\lambda_0/2^{K_O} \leq \lambda_k$. The inequality in part (c) then follows from the inequality in part (b). ∎

In view of Lemma 4.1(a), choosing an initial stepsize λ_0 satisfying $\lambda_0 \leq 1/(2\underline{m})$ results in an R-AIPP variant with constant stepsize, which resembles the AIPP method described in [15].

The next proposition presents a worst-case iteration complexity bound on the number of inner iterations of the R-AIPP method with respect to the inputs M, λ_0, and z_0, the quantity $\underline{m}$ in (2.2), and the tolerance $\hat{\rho}$.

Proposition 4.2 *Defining $\xi_0 := \max\{1/\lambda_0, 4\underline{m}\}$, the R-AIPP method outputs a $\hat{\rho}$–approximate stationary point $(\hat{z}, \hat{v})$ of (1.4) in at most*

$$\mathcal{O}\left(\sqrt{M+\xi_0}\left[\frac{\sqrt{\xi_0}\,[\phi(z_0)-\phi_*]}{\hat{\rho}^2} + \sqrt{\lambda_0}\right]\log_1^+\left(\lambda_0\,[M+\xi_0]\right)\right) \tag{4.3}$$

inner iterations.

Proof Let TI_S (resp. TI_O) denote the total number of inner iterations performed during all calls of type S (resp. type O) (see the paragraph preceding Lemma 4.1). Clearly, the total number of inner iterations is $\text{TI} := \text{TI}_S + \text{TI}_O$. We now bound each one of the quantities TI_S and TI_O separately by using the fact that assumption (A2), (4.1), and Proposition 3.2(a) imply that the number of inner iterations performed during each call to the R-ACG algorithm is bounded by

$$\left\lceil \sqrt{2\lambda M + 1}\log_1^+\left(C\,[2\lambda M+1]\right)\right\rceil,$$

where λ is the value of λ just before the call and C is as in (3.16) with $\widetilde{M} = \bar{\lambda}M$.

We first consider TI_O. Note that Lemma 4.1(b) implies that K_O is finite. Since $\text{TI}_O = 0$ when $K_O = 0$, we may assume without loss of generality that $K_O \geq 1$. Note that the values of λ just before the K_O calls of type O are exactly $\lambda_0, \lambda_0/2, \ldots, \lambda_0/2^{K_O-1}$. Hence, we conclude that

$$\begin{aligned}\text{TI}_O &\leq \sum_{i=1}^{K_O}\left(\sqrt{\frac{2\lambda_0 M}{2^{i-1}}+1}\right)\log_1^+\left(C\left[\frac{2\lambda_0 M}{2^{i-1}}+1\right]\right) \leq \sum_{i=1}^{K_O}\left(\sqrt{\frac{2\lambda_0\,(M+\xi_0)}{2^{i-1}}}\right)\log_1^+\left(2C\lambda_0\,[M+\xi_0]\right)\\ &\leq \left(2+\sqrt{2}\right)\sqrt{2\lambda_0\,(M+\xi_0)}\log_1^+\left(2C\lambda_0\,[M+\xi_0]\right),\end{aligned}$$

where the second inequality is due the fact that Lemma 4.1(b) implies $2^{i-1} \leq 2^{K_O-1} \leq 2\lambda_0\xi_0$ for every $i \leq K_O$. Thus, we obtain

$$\mathrm{TI}_O = \mathcal{O}\left(\sqrt{\lambda_0\,(M+\xi_0)}\log_1^+\left(\lambda_0\,[M+\xi_0]\right)\right). \tag{4.4}$$

We now bound TI_S. Suppose that $K_S > 1$ and observe that the termination criterion (4.2) is not satisfied in the first $K_S - 1$ iterations. Since the R-AIPP method is an instance of the GD framework, it follows from Proposition 2.2 that

$$\hat{\rho}^2 < \min_{j\leq K_L-1}\|\hat{v}_j\|^2 \leq \theta\left(1+2\sqrt{\tau}\right)^2\frac{[\phi(z_0)-\phi_*]}{\sum_{j=1}^{K_L-1}\lambda_j}. \tag{4.5}$$

Using the fact that Lemma 4.1(c) implies $1/\lambda_j \leq \max\{1/\lambda_0, 4\bar{m}\} = \xi_0$ and $\lambda_j \leq \lambda_0$ for every $j \geq 1$, we obtain

$$\begin{aligned}
\mathrm{TI}_S &\leq \sum_{j=1}^{K_L}\left(\sqrt{\lambda_j M+1}\right)\log\left(C\,[\lambda_j M+1]\right) \leq \sum_{j=1}^{K_L}\left(\sqrt{\lambda_j(M+\xi_0)}\right)\log\left(C\lambda_j[M+\xi_0]\right)\\
&\leq \sqrt{M+\xi_0}\left(\sum_{j=1}^{K_L-1}\frac{\lambda_j}{\sqrt{\lambda_j}}+\sqrt{\lambda_{K_L}}\right)\log\left(C\lambda_0[M+\xi_0]\right)\\
&\leq \sqrt{M+\xi_0}\left[\sqrt{\xi_0}\left(\sum_{j=1}^{K_L-1}\lambda_j\right)+\sqrt{\lambda_0}\right]\log\left(C\lambda_0[M+\xi_0]\right)\\
&\leq \sqrt{M+\xi_0}\left[\theta\left(1+2\sqrt{\tau}\right)^2\frac{\sqrt{\xi_0}\,[\phi(z_0)-\phi_*]}{\hat{\rho}^2}+\sqrt{\lambda_0}\right]\log\left(C\lambda_0[M+\xi_0]\right),
\end{aligned}$$

Hence, we conclude that

$$\mathrm{TI}_S = \mathcal{O}\left(\sqrt{M+\xi_0}\left[\frac{\sqrt{\xi_0}\,[\phi(z_0)-\phi_*]}{\hat{\rho}^2}+\sqrt{\lambda_0}\right]\log_1^+\left(\lambda_0\,[M+\xi_0]\right)\right). \tag{4.6}$$

It can be easily seen that the bound in (4.6) trivially holds when $K_S \leq 1$ in view of the last term in it. Indeed, to prove this, just assume that $\sum_{j=1}^{K_S-1}\lambda_j = 0$ in the above argument bounding TI_S. Now, since $\mathrm{TI} = \mathrm{TI}_O + \mathrm{TI}_S$, the bound in (4.3) follows by adding (4.4) and (4.6).

The last statement of the proposition follows due to Proposition 2.1 and the termination condition in step 3 of the R-AIPP method. ∎

Observe that, unless λ_0 is large or $\underline{m}$ is small, the first term in (4.3) dominates the second one.

The numerical experiments in Section 6 consider three variants of the R-AIPP method, two of which are R-AIPP instances with different choices of λ_0. More specifically, given an upper bound m on $\underline{m}$, one of the R-AIPP instances chooses $\lambda_0 = 0.9/(2m)$ while the other one chooses $\lambda_0 = 1$. For the problem instances considered, the former choice of λ_0 is relatively small, while the latter choice is relatively large.

We now end this section by discussing some possible choices of the initial stepsize λ_0 and how the corresponding R-AIPP instances compare to the AIPP method of [15]. First, the AIPP method requires knowledge of an upper bound m on $\underline{m}$ such that $m = \mathcal{O}(M)$, and, as a consequence of a more general iteration complexity bound derived in [15, Corollary 14], its inner iteration complexity can be shown to be

$$\mathcal{O}\left(\sqrt{M}\left[\frac{\sqrt{m}\,[\phi(z_0)-\phi_*]}{\hat{\rho}^2}+\sqrt{\frac{1}{m}}\log_1^+\left(\frac{M}{m}\right)\right]\right). \tag{4.7}$$

Now, if m as above is also known to the R-AIPP and the input λ_0 is set to $1/(4m)$, then its inner iteration complexity (4.3) reduces to

$$\mathcal{O}\left(\sqrt{M}\left[\frac{\sqrt{m}\,[\phi(z_0)-\phi_*]}{\hat{\rho}^2}+\sqrt{\frac{1}{m}}\right]\log_1^+\left(\frac{M}{m}\right)\right), \tag{4.8}$$

which is the same as (4.7) up to a logarithmic factor. On the other hand, if λ_0 is chosen so that $1/\lambda_0 = \mathcal{O}(\underline{m})$ then (4.3) reduces to

$$\mathcal{O}\left(\sqrt{M}\left[\frac{\sqrt{\underline{m}}\,[\phi(z_0)-\phi_*]}{\hat{\rho}^2}+\sqrt{\lambda_0}\right]\log_1^+\left(\lambda_0 M\right)\right), \tag{4.9}$$

whose dominant first term is as good as the dominant first term in (4.7) whenever $\sqrt{\underline{m}}\log_1^+(\lambda_0 M) = \mathcal{O}(\sqrt{m})$.

5 A relaxed quadratic penalty AIPP method

This section presents the R-QP-AIPP method for solving a class of linearly–set–constrained nonconvex composite optimization problems. Similar to the QP-AIPP method of [15], the R-QP-AIPP method is a quadratic penalty–based method that solves a sequence of penalized subproblems, for increasing values of the penalty parameter, using the R-AIPP method of Section 4. The section contains two subsections. The first one describes the main problem of interest, its underlying assumptions, and the notion of a corresponding approximate stationary point which R-QP-AIPP method will provably obtain, and briefly outlines a cold–started quadratic penalty–based method for obtaining such a point. The second one presents a warm–started quadratic penalty–based method, namely, the R-QP-AIPP method, for obtaining the desired stationary point and establishes its ACG iteration complexity.

5.1 The linearly–set–constrained problem

This subsection describes the main problem of interest in this section, namely, the linearly–set–constrained nonconvex composite optimization problem (5.1), its underlying assumptions, and a notion of an approximate stationary point of it. Moreover, it describes the quadratic penalty subproblem (parameterized a penalty parameter) associated with (5.1) and discusses the relationship between their corresponding approximate stationary points. It then outlines a (static and dynamic) cold–started quadratic penalty–based method and its corresponding iteration-complexity bound, which turns out to be larger than that of the QP-AIPP method of [15].

The main problem of interest for this section is the linearly–set–constrained nonconvex composite optimization problem

$$\varphi^* := \min \{\varphi(z) := f(z) + h(z) : Az \in S,\, z \in \Re^n\}, \tag{5.1}$$

where closed convex set $S \subseteq \Re^p$, linear operator $A : \Re^n \mapsto \Re^p$, and functions $f, h : \Re^n \mapsto (-\infty, \infty]$, satisfy the following assumptions:

(C1) $h \in \overline{\mathrm{Conv}}(\Re^n)$ and its diameter

$$D_h := \sup\{\|z - z'\| : z, z' \in \mathrm{dom}\, h\} \tag{5.2}$$

is finite;

(C2) $A \neq 0$ and $\mathcal{F} := \{z \in \mathrm{dom}\, h : Az \in S\} \neq \emptyset$;

(C3) f is a nonconvex differentiable function on $\mathrm{dom}\, h$ and there exist a scalar $L > 0$ such that

$$\|\nabla f(z) - \nabla f(u)\| \leq L\|z - u\| \quad \forall u, z \in \mathrm{dom}\, h; \tag{5.3}$$

(C4) $\varphi_0^* := \inf\{\varphi(z) : z \in \Re^n\} > -\infty$.

We make two remarks about the above assumptions. First, Lemma A.1 in Appendix A.3 shows that (C1), (C3), and the additional assumption that f be lower semicontinuous on $\mathrm{cl}\,(\mathrm{dom}\, h)$ imply (C4). Second, denoting $\underline{m}$ as the quantity (2.2) with $g = f$, assumption (C3) implies that $\underline{m} \in (0, L]$. Moreover, it is shown in Theorem 5.3 below that the smaller $\underline{m}$ is, the better the iteration complexity of the R-QP-AIPP method becomes.

We now discuss a notion of approximate stationary point for (5.1). Clearly, (5.1) is equivalent to the problem

$$\min \{f(z) + h(z) : Az - s = 0,\, s \in S,\, z \in \Re^n\}. \tag{5.4}$$

Moreover, a necessary condition for a point $(\hat{z}, \hat{s}) \in \mathrm{dom}\, h \times S$ to be a local minimum to the above problem is that there exists a multiplier $\hat{q} \in \Re^p$ such that

$$0 \in \nabla f(\hat{z}) + \partial h(\hat{z}) + A^*\hat{q}, \quad A\hat{z} - \hat{s} = 0, \quad \hat{q} \in N_S(\hat{s}), \quad \hat{s} \in S. \tag{5.5}$$

Given a tolerance pair $(\hat{\rho}, \hat{\eta}) \in \Re^2_{++}$, a triple $([\hat{z}, \hat{s}], \hat{q}, \hat{v}) \in [\mathrm{dom}\, h \times S] \times \Re^p \times \Re^n$ is said to be a $(\hat{\rho}, \hat{\eta})$–approximate stationary point of (1.1) if it satisfies

$$\hat{v} \in \nabla f(\hat{z}) + \partial h(\hat{z}) + A^*\hat{q}, \quad \|\hat{v}\| \leq \hat{\rho}, \quad \|A\hat{z} - \hat{s}\| \leq \hat{\eta}, \quad \hat{q} \in N_S(\hat{s}), \quad \hat{s} \in S. \tag{5.6}$$

Clearly, a $(\hat{\rho},\hat{\eta})$–approximate stationary point $([\hat{z},\hat{s}],\hat{q},\hat{v})$ of (5.1) when $(\hat{\rho},\hat{\eta})=(0,0)$ means that the pair $(\hat{z},\hat{s})$ and the multiplier $\hat{q}$ satisfy (5.5).

We now describe the quadratic penalty subproblem (parameterized by a penalty parameter) with respect to (5.1). Defining the quadratic penalty function $p_S:\Re^p\mapsto\Re_+$ as

$$p_S(x):=\frac{1}{2}\|x-\mathit{\Pi}_S(x)\|^2 \tag{5.7}$$

where

$$\mathit{\Pi}_S(x):=\arg\min\{\|u-x\|:u\in S\} \tag{5.8}$$

for every $x\in\Re^p$, the quadratic penalty subproblem parameterized by a penalty parameter $c>0$ with respect to (5.1) is

$$\varphi_c^*:=\min\left\{\varphi_c(z):=\varphi(z)+c\cdot p_S(Az):z\in\Re^n\right\}. \tag{5.9}$$

We now make four remarks regarding (5.9). First, (1.3) is an instance of (5.9) in which $S=\{b\}$. Second, when $c=0$, the optimal value of (5.9) coincides with φ_0^* in (C4), and hence there is no abuse of notation made here. Third, it is easily seen that

$$\varphi^*\geq\varphi_{\bar{c}}^*\geq\varphi_c^*\quad\forall\bar{c}>c\geq 0, \tag{5.10}$$

where φ^* is as in (5.1). Finally, (5.9) is a penalty subproblem involving only the original variable z of formulation (5.1) rather than the one associated with (5.4) (constructed as in Section 1 with $Az=b$ replaced by $Az-s=0$), which involves the pair of variables (z,s).

The following result shows how a $\hat{\rho}$–approximate stationary point of (5.9) yields a $(\hat{\rho},\hat{\eta})$–approximate stationary point of (5.1) when the penalty parameter c is sufficiently large.

Proposition 5.1 *Let $(\hat{\rho},\hat{\eta})\in\Re^2_{++}$ and $c\geq 0$ be given and suppose that $(\hat{z},\hat{v})$ is a $\hat{\rho}$–approximate stationary point of (5.9) as in (2.3) with $g=f+c\cdot(p_S\circ A)$. Moreover, let $\underline{m}$ be as in (2.2) with $g=f$ and define*

$$g_c:=f+c\cdot(p_S\circ A),\quad M_c:=L+c\|A\|^2, \tag{5.11}$$

$$\hat{s}:=\mathit{\Pi}_S(A\hat{z}),\quad \hat{q}:=c(A\hat{z}-\hat{s}),\quad T:=2\left(\varphi^*-\varphi_0^*+\hat{\rho}D_h\right)+\underline{m}D_h^2, \tag{5.12}$$

where φ^* and φ_0^* are as in (5.1) and (C4), respectively. Then, the following statements hold:

(a) for every $u,z\in\operatorname{dom}h$, the pair $(g,M)=(g_c,M_c)$ satisfies (2.1);
(b) the triple $([\hat{z},\hat{s}],\hat{q},\hat{v})$ satisfies the inclusions and the first inequality of (5.6) and

$$\frac{c}{2}\|A\hat{z}-\hat{s}\|^2\leq\varphi^*-\varphi(\hat{z})+\hat{\rho}D_h+\frac{1}{2}\left(\underline{m}D_h^2\right); \tag{5.13}$$

(c) if, in addition, the penalty parameter c satisfies

$$c\geq\frac{T}{\hat{\eta}^2}, \tag{5.14}$$

then $\|A\hat{z}-\hat{s}\|\leq\hat{\eta}$, and hence $([\hat{z},\hat{s}],\hat{q},\hat{v})$ is a $(\hat{\rho},\hat{\eta})$–approximate stationary point of (5.1).

Proof Throughout this proof, we will make use of the well known fact (see, for example, [1, Theorems 6.39 & 6.60]) that p_S is convex, differentiable, its gradient is 1–Lipschitz, and, for every $x\in\Re^p$,

$$\nabla p_S(x)=x-\mathit{\Pi}_S(x)\in N_S\left(\mathit{\Pi}_S(x)\right). \tag{5.15}$$

(a) This follows immediately from the definition of g_c in (5.11), assumption (C3), and the fact that ∇p_S is 1–Lipschitz continuous.

(b) Using the definitions of $\hat{q}$ and $\hat{s}$ given in (5.12), and the fact that (5.15) at $x=A\hat{z}$ implies $c\nabla p_S(A\hat{z})=\hat{q}$, observe that: (i) $c\nabla(p_S\circ A)\hat{z}=cA^*\nabla p_S(A\hat{z})=A^*\hat{q}$; and (ii) $\hat{q}\in N_S(\hat{s})$. It now follows from the definition of a $\hat{\rho}$–approximate stationary point of (5.9) with $g=f+c\cdot(p_S\circ A)$ and the previous observations that

$$\begin{aligned}\hat{v}\in\nabla f(\hat{z})+\partial h(\hat{z})+c\nabla(p_S\circ A)\hat{z}&=\nabla f(\hat{z})+\partial h(\hat{z})+A^*\hat{q}\\&\subseteq\nabla f(\hat{z})+\partial h(\hat{z})+A^*N_S(\hat{s}).\end{aligned} \tag{5.16}$$

Hence, with the additional fact that $\|\hat{v}\| \leq \hat{\rho}$ from (2.3), it follows that the inclusions and first inequality of (5.6) hold. Next, observe that the convexity of p_S and the first inclusion in (5.16) imply that $\hat{v} \in \nabla f(\hat{z}) + \partial\left[h + c \cdot (p_S \circ A)\right](\hat{z})$ or equivalently,

$$h(u) + c \cdot p_S(Au) \geq h(\hat{z}) + c \cdot p_S(A\hat{z}) + \langle \hat{v} - \nabla f(\hat{z}), u - \hat{z}\rangle \quad \forall u \in \operatorname{dom} h. \tag{5.17}$$

Considering (5.17) at any $u \in \mathcal{F}$ and using the fact that $p_S(Au) = 0$ for any $u \in \mathcal{F}$, the definition of $\underline{m}$ in (2.2), and the definitions of p_S and $\hat{s}$, we conclude that

$$\begin{aligned}
\frac{c}{2}\|A\hat{z} - \hat{s}\|^2 &\leq h(u) - h(\hat{z}) + \langle \nabla f(\hat{z}), u - \hat{z}\rangle - \langle \hat{v}, u - \hat{z}\rangle \\
&\leq (f+h)(u) - (f+h)(\hat{z}) + \|\hat{v}\|\|u - \hat{z}\| + \frac{1}{2}\left(\underline{m}\|u - \hat{z}\|^2\right) \\
&\leq \varphi(u) - \varphi(\hat{z}) + \hat{\rho} D_h + \frac{1}{2}\left(\underline{m} D_h^2\right).
\end{aligned}$$

Taking the infimum over $u \in \mathcal{F}$ immediately implies (5.13).

(c) Using (5.14), the fact that $\varphi(\hat{z}) \geq \varphi_0^*$, and the definition of T, it follows from part (b) that

$$\|A\hat{z} - \hat{s}\|^2 \leq \frac{1}{c}\left[2\left(\varphi^* - \varphi_0^* + \hat{\rho} D_h\right) + \underline{m} D_h^2\right] = \frac{T}{c} \leq \hat{\eta}^2.$$

■

In view of the above proposition, we now outline a static penalty method for obtaining a $(\hat{\rho}, \hat{\eta})$–approximate stationary point of (5.1). First, let $z_0 \in \operatorname{dom} h$ be given and select a penalty parameter $c = \mathcal{O}(\hat{\eta}^{-2})$ satisfying (5.14). Second, obtain a $\hat{\rho}$–approximate stationary point $(\hat{z}, \hat{v})$ of (5.9) using the R-AIPP method of Section 4 with starting point z_0 and inputs $M = M_c$ and $(g, h) = (g_c, h)$, which satisfy assumptions (A1)–(A3) in view of Proposition 5.1(a) and assumptions $(C1)$ and $(C3)$. Finally, compute the pair $(\hat{s}, \hat{q})$ according to (5.12) and output the triple $([\hat{z}, \hat{s}], \hat{q}, \hat{v})$, which is a $(\hat{\rho}, \hat{\eta})$–approximate stationary point of (5.1) in view of Proposition 5.1(c). Using (5.11) with $(c, \bar{c}) = (0, c)$, the definitions in (5.11), the fact that $c = \mathcal{O}(\hat{\eta}^{-2})$, and the complexity bound for the R-AIPP method described in Proposition 4.2 with $M = M_c$, it is easy to see that the ACG iteration complexity of the outlined method is

$$\mathcal{O}\left(\sqrt{M_c + \xi_0}\left[\frac{\sqrt{\xi_0}\left[\varphi_c(z_0) - \varphi_0^*\right]}{\hat{\rho}^2} + \sqrt{\lambda_0}\right]\log_1^+\left(\lambda_0\left[M_c + \xi_0\right]\right)\right) = \mathcal{O}\left(\hat{\rho}^{-2}\hat{\eta}^{-3}\log_1^+\hat{\eta}^{-1}\right), \tag{5.18}$$

where $\xi_0 := \max\{1/\lambda_0, 4\underline{m}\}$ and the last quantity ignores any constants aside from the tolerances. A drawback of this static penalty method is that it requires in its first step the selection of a single parameter c, which is generally difficult to obtain. This issue can be circumvented by considering a dynamic cold–started penalty method in which the static penalty method is repeated for a sequence of increasing values of c and common starting point z_0. It can be shown that the resulting cold–started dynamic penalty method has an ACG iteration complexity that is still on the same order as (5.18). Note that the bound (5.18) is actually $\mathcal{O}(\hat{\rho}^{-2}\hat{\eta}^{-1}\log_1^+\hat{\eta}^{-1})$ when $z_0 \in \mathcal{F}$ (see (C2)) but our interest lies in the case where $z_0 \notin \mathcal{F}$ since an initial point $z_0 \in \mathcal{F}$ is generally not known.

The QP-AIPP method of [15] is a modified cold–started dynamic penalty method like the one just outlined, but which replaces the R-AIPP method called in step 2 of the static penalty method with the AIPP method of [15]. It has been shown in [15, Theorem 18] that its ACG iteration complexity bound for finding a $(\hat{\rho}, \hat{\eta})$–approximate stationary point of (1.1) is $\mathcal{O}(\hat{\rho}^{-2}\hat{\eta}^{-1})$. This bound is established without assuming that $\operatorname{dom} h$ is bounded and is clearly better than the one in (5.18).

The next subsection considers a warm–started dynamic penalty method, similar to the one described immediately after Proposition 5.1, in which the input z_0 to the R-AIPP call for solving the next penalty subproblem is chosen to be the output $\hat{z}$ from the R-AIPP call for solving the current one. It is shown in Theorem 5.3 of Subsection 5.2 that its ACG iteration complexity is $\mathcal{O}(\hat{\rho}^{-2}\hat{\eta}^{-1}\log_1^+\hat{\eta}^{-1})$, which is the same as the one for the QP-AIPP method up to a logarithmic factor. As a side remark, we note that although a warm–started version of the QP-AIPP method in [15] can be also considered, the aforementioned $\mathcal{O}(\hat{\rho}^{-2}\hat{\eta}^{-1})$ ACG iteration complexity bound was derived for its cold–started version.

5.2 The R-QP-AIPP method

The goal of this subsection is to describe the R-QP-AIPP method, i.e., the warm–started dynamic penalty method mentioned at the end of Subsection 5.1, and establish its corresponding ACG iteration complexity.

We start by describing the R-QP-AIPP method.

R-QP-AIPP method.

Input: a problem instance of the form in (5.1), a scalar $L > 0$, a tolerance pair $(\hat{\rho}, \hat{\eta}) \in \Re^2_{++}$, an initial point $\hat{z}_0 \in \operatorname{dom} h$, a scalar $\lambda_0 > 0$, and a pair of parameters $(\theta, \tau) \in (2, \infty) \times (0, \infty)$;

Output: a triple $([\hat{z}, \hat{s}], \hat{q}, \hat{v}) \in [\operatorname{dom} h \times S] \times \Re^p \times \Re^n$ satisfying (5.6);

(0) set $c_0 := L/\|A\|^2$ and $l = 1$;
(1) set $(c, z_0) := (c_{l-1}, \hat{z}_{l-1})$ and

$$M_c := L + c\|A\|^2, \quad g_c := f + c \cdot (p_S \circ A);$$

call the R-AIPP method on (1.4) with inputs $\hat{\rho}$, M_c, (g_c, h), z_0, λ_0, and (θ, τ), to obtain a $\hat{\rho}$-approximate stationary point $(\hat{z}, \hat{v})$ of (1.4), and set

$$(\hat{z}_l, \hat{v}_l) = (\hat{z}, \hat{v}), \quad \hat{s}_l = \Pi_S(A\hat{z}_l), \quad \hat{q}_l = c(A\hat{z}_l - \hat{s}_l);$$

(2) if the residual

$$\|A\hat{z}_l - \hat{s}_l\| \leq \hat{\eta},$$

then return $([\hat{z}, \hat{s}], \hat{q}, \hat{v}) = ([\hat{z}_l, \hat{s}_l], \hat{q}_l, \hat{v}_l)$; otherwise, set $c_l = 2c_{l-1}$, increment $l = l + 1$, and go to step 1.

Before giving some remarks about the above method, we discuss its general structure. Every loop of the R-QP-AIPP method invokes in its step 1 the R-AIPP method of Section 4 to compute a $\hat{\rho}$-approximate stationary point of the current penalty subproblem (5.9). The latter method in turn uses the R-ACG algorithm of Section 3 as a subroutine in its implementation (see step 1 of the R-AIPP method). Moreover, step 1 of the R-QP-AIPP implements a warm–start strategy, namely, the input point z_0 of the current R-AIPP call is set to be the output point $\hat{z}_{l-1}$ of the previous R-AIPP call.

We now make three remarks about the R-QP-AIPP method. First, it follows from Proposition 5.1(b) that, for every $l \geq 1$, the triple $([\hat{z}, \hat{s}], \hat{q}, \hat{v}) = ([\hat{z}_l, \hat{s}_l], \hat{q}_l, \hat{v}_l)$ satisfies the inclusions and the first inequality in (5.6). Second, since every loop of the R-QP-AIPP method doubles c, the condition (5.14) will be eventually satisfied. Hence, in view of Proposition 5.1(c), the pair $(\hat{z}, \hat{s})$ corresponding to this c will satisfy the condition $\|A\hat{z} - \hat{s}\| \leq \hat{\eta}$ and the R-QP-AIPP method will stop in view of its stopping criterion in step 2. Finally, in view of the first and second remarks, we conclude that the R-QP-AIPP method outputs a triple $([\hat{z}, \hat{s}], \hat{q}, \hat{v})$ satisfying (5.6).

Before deriving the ACG iteration complexity of the R-QP-AIPP method, we note that the number of ACG iterations needed in the $(l+1)^{\text{th}}$ execution of its step 1 depends on the quantity $\varphi_{c_l}(\hat{z}_l) - \varphi^*_{c_l}$ (see the left–hand–side of (5.18) with $(c, z_0) = (c_l, \hat{z}_l)$). The result below shows that the warm–start strategy in step 1 of the method together with the boundedness of $\operatorname{dom} h$ imply that the aforementioned quantity has an upper bound that is independent of the size of the parameter c_l.

Lemma 5.2 *Let c_0 and $\hat{z}_0$ be as in step 0 and the input of the R-QP-AIPP method, respectively, and define*

$$S_0 := \varphi_{c_0}(\hat{z}_0) - \varphi^*_{c_0}, \quad Q_0 := T + S_0, \tag{5.19}$$

where φ^*_c and T are as in (5.9) *and* (5.12)*, respectively. Then, for every $l \geq 0$, we have*

$$\varphi_{c_l}(\hat{z}_l) - \varphi^*_{c_l} \leq Q_0. \tag{5.20}$$

Proof The case in which $l = 0$ follows trivially from the definition of S_0 in (5.19). Consider now the case in which $l \geq 1$. Remark that $c_l = 2c_{l-1}$ due to step 2 of R-QP-AIPP and (5.9) and that $(\hat{z}_l, \hat{v}_l)$ is a $\hat{\rho}$–approximate stationary point of (5.9) with $c = c_{l-1}$ due to the warm–start strategy in step 1 of the R-QP-AIPP method. It now follows from the aforementioned remarks, the last inequality in (5.10) with $c = c_l$, and Proposition 5.1(b) with $(\hat{z}, c) = (\hat{z}_l, c_{l-1})$, that

$$\begin{aligned}\varphi_{c_l}(\hat{z}_l) - \varphi_{c_l}^* &\leq \varphi_{c_l}(\hat{z}_l) - \varphi_0^* = \varphi(\hat{z}_l) + 2\left[\frac{c_{l-1}}{2}\|A\hat{z}_l - \hat{s}_l\|^2\right] - \varphi_0^* \\ &\leq \varphi(\hat{z}_l) + 2\left[\varphi^* - \varphi(\hat{z}_l) + \hat{\rho}D_h + \frac{1}{2}\left(\underline{m}D_h^2\right)\right] - \varphi_0^*. \end{aligned} \tag{5.21}$$

Grouping terms in the last expression together, using the definition of Q_0 given in (5.19), and the fact that $\varphi(\hat{z}_l) \geq \varphi_0^*$, we conclude that

$$\varphi(\hat{z}_l) + 2\left[\varphi^* - \varphi(\hat{z}_l) + \hat{\rho}D_h + \frac{1}{2}\left(\underline{m}D_h^2\right)\right] - \varphi_0^* \leq 2\left(\varphi^* - \varphi_0^* + \hat{\rho}D_h\right) + \underline{m}D_h^2 = T \leq Q_0. \tag{5.22}$$

Combining (5.21) and (5.22) yields (5.20). ∎

The following result establishes the iteration complexity of the R-QP-AIPP method with respect to the inputs L, λ_0, and z_0, the quantity $\underline{m}$ in (2.2) with $g = f$, and the tolerance pair $(\hat{\rho}, \hat{\eta})$.

Theorem 5.3 *Given a tolerance pair $(\hat{\rho}, \hat{\eta}) \in \Re_+^2$, define*

$$\Xi_{\hat{\eta}} := L + \frac{T\|A\|^2}{\hat{\eta}^2}, \tag{5.23}$$

where T is given in (5.12)*. Then, defining $\xi_0 := \max\{1/\lambda_0, 4\underline{m}\}$, the R-QP-AIPP method outputs a $(\hat{\rho}, \hat{\eta})$–approximate stationary point $([\hat{z}, \hat{s}], \hat{q}, \hat{v})$ of* (5.1) *in at most*

$$\mathcal{O}\left(\sqrt{\Xi_{\hat{\eta}} + \xi_0}\left[\frac{\sqrt{\xi_0}Q_0}{\hat{\rho}^2} + \sqrt{\lambda_0}\right]\log_1^+\left(\lambda_0\left[\Xi_{\hat{\eta}} + \xi_0\right]\right)\right), \tag{5.24}$$

ACG iterations, where Q_0 is as in (5.19)*.*

Proof Define $T_{\hat{\eta}} := T/\hat{\eta}^2$ and let $\bar{l}$ be the smallest index such that $c_{\bar{l}-1} \geq T_{\hat{\eta}}$. Since the R-QP-AIPP invokes the R-AIPP method with $(M, g) = (M_{c_{l-1}}, g_{c_{l-1}})$, it follows from Lemma 5.2 and Proposition 4.2, with $M = M_{c_{l-1}}$, that the total number of ACG iterations at the l^{th} iteration of the R-QP-AIPP method is on the order of

$$\mathcal{O}\left(\sqrt{M_{c_{l-1}} + \xi_0}\left[\frac{\sqrt{\xi_0}Q_0}{\hat{\rho}^2} + \sqrt{\lambda_0}\right]\log_1^+\left(\lambda_0\left[M_{c_{l-1}} + \xi_0\right]\right)\right). \tag{5.25}$$

Hence, the R-QP-AIPP method stops in a total number of ACG iterations bounded above by the sum of the quantity in (5.25) over $l = 1, \ldots, \bar{l}$.

We now focus on simplifying some of the quantities in the aforementioned sum. Using the fact that $L = c_0\|A\|^2$, we obtain the bound

$$M_{c_{l-1}} = L + c_{l-1}\|A\|^2 = L + 2^{l-1}c_0\|A\|^2 \leq 2^{l-1}\left(L + c_0\|A\|^2\right) = 2^l c_0\|A\|^2. \tag{5.26}$$

Now, if $\bar{l} = 1$, then the above inequality implies that $M_{c_{\bar{l}-1}} \leq 2c_0\|A\|^2 = 2L = \mathcal{O}\left(\Xi_{\hat{\eta}}\right)$. Assume then that $\bar{l} \geq 2$. Observe that the definition of $\bar{l}$ implies that $2^{\bar{l}-1}c_0 \leq 2T_{\hat{\eta}}$ or, equivalently, $\sqrt{c_0}\sqrt{2}^{\bar{l}} \leq 2\sqrt{T_{\hat{\eta}}}$. Combining the previous inequality with (5.26), we conclude that

$$\begin{aligned}\sum_{k=1}^{\bar{l}}\sqrt{M_{c_{l-1}} + \xi_0} &\leq \sum_{k=1}^{\bar{l}}\sqrt{2^l c_0\|A\|^2 + \xi_0} \leq \sqrt{2}^{\bar{l}}\left(1 + \sqrt{2}\right)\sqrt{2c_0\|A\|^2 + \xi_0} \\ &\leq 8\sqrt{\|A\|^2 T_{\hat{\eta}} + \xi_0} = \mathcal{O}\left(\sqrt{\Xi_{\hat{\eta}} + \xi_0}\right)\end{aligned} \tag{5.27}$$

and also

$$\log_1^+\left(M_{c_{l-1}} + \xi_0\right) \leq \log_1^+\left(2^{\bar{l}}c_0\|A\|^2 + \xi_0\right) \leq \log_1^+\left(4T_{\hat{\eta}}\|A\|^2 + \xi_0\right) = \mathcal{O}\left(\log_1^+\left[\Xi_{\hat{\eta}} + \xi_0\right]\right). \tag{5.28}$$

It now follows from (5.25), (5.27), and (5.28) that the R-QP-AIPP method stops in a total number of ACG iterations bounded by the quantity in (5.24).

The statement that $([\hat{z}, \hat{s}], \hat{q}, \hat{v})$ is a $(\hat{\rho}, \hat{\eta})$–approximate stationary point follows from Proposition 5.1(b) and the termination condition in step 2 of the R-QP-AIPP method. ∎

We now make three remarks about the complexity bound in (5.24). First, in terms of the tolerance pair $(\hat{\rho}, \hat{\eta})$, it is $\mathcal{O}(\hat{\rho}^{-2}\hat{\eta}^{-1}\log_1^+ \hat{\eta}^{-1})$, which improves upon the complexity in (5.18) by a $\Theta(\hat{\eta}^{-2})$ factor. Second, unless λ_0 is large or $\underline{m}$ is small, the first term in (5.24) dominates the second one.

We now end this section by discussing some possible choices of the initial stepsize λ_0 and how the corresponding R-QP-AIPP instances compare to the QP-AIPP method of [15]. First, recall that the QP-AIPP method requires the knowledge of an upper bound m on $\underline{m}$ such that $m = \mathcal{O}(L)$, and remark that, under the same assumptions of this paper, it can be shown using [15, Theorem 18] that its ACG iteration complexity is

$$\mathcal{O}\left(\sqrt{\Xi_{\hat{\eta}}}\left[\frac{\sqrt{m}Q_0}{\hat{\rho}^2} + \sqrt{\frac{1}{m}}\log_1^+\left(\frac{\Xi_{\hat{\eta}}}{m}\right)\right]\right). \tag{5.29}$$

Now, if m as above is also known to the R-AIPP and the input λ_0 is set to $1/(4m)$, then the ACG iteration complexity (5.24) reduces to

$$\mathcal{O}\left(\sqrt{\Xi_{\hat{\eta}}}\left[\frac{\sqrt{m}Q_0}{\hat{\rho}^2} + \sqrt{\frac{1}{m}}\right]\log_1^+\left(\frac{\Xi_{\hat{\eta}}}{m}\right)\right), \tag{5.30}$$

which is the same as (5.24) up to a logarithmic factor. On the other hand, if λ_0 is chosen so that $1/\lambda_0 = \mathcal{O}(\underline{m})$ then (5.24) reduces to

$$\mathcal{O}\left(\sqrt{\Xi_{\hat{\eta}}}\left[\frac{\sqrt{\underline{m}}Q_0}{\hat{\rho}^2} + \sqrt{\lambda_0}\right]\log_1^+\left(\lambda_0\Xi_{\hat{\eta}}\right)\right), \tag{5.31}$$

whose dominant first term is as good as the dominant first term in (5.29) when $\sqrt{\underline{m}}\log_1^+(\lambda_0\Xi_{\hat{\eta}}) = \mathcal{O}(\sqrt{m})$.

6 Numerical experiments

This section presents computational results that highlight the performance of the R-AIPP and R-QP-AIPP methods. It contains three subsections. The first subsection compares three variants of the R-AIPP method against three state-of-the-art nonconvex composite optimization algorithms. The second subsection uses the six algorithms in the first subsection as subroutines in a quadratic penalty method similar to the one in Section 5. More specifically, given an algorithm A out of the six algorithms in the first subsection, a corresponding quadratic penalty method is considered in which steps 0 to 2 of the R-QP-AIPP method in Section 5 are executed with algorithm A replacing the R-AIPP method. The third subsection presents a summary of the numerical experiments.

We first describe the three different R-AIPP variants considered. While the second variant does not assume knowledge of an upper bound m on the quantity $\underline{m}$ in (2.2), the first and third variants do in order to determine their initial stepsize λ_0. More specifically, the first variant, referred to as R-AIPPc, is the R-AIPP method with initial stepsize chosen to be $\lambda_0 = 0.9/(2m)$. As opposed to the two algorithms explained below, which can adaptively change λ_k between iterations, this algorithm is a constant stepsize method (see Lemma 4.1 and the paragraph following it). The second variant, referred to as R-AIPPv1, is the R-AIPP method with initial stepsize chosen to be $\lambda_0 = 1$. Since λ_0 is relatively large in the experiments considered, λ is halved in some of its outer iterations. The third variant, referred to as R-AIPPv2, is a variant of the R-AIPP method with initial stepsize chosen to be $\lambda_0 = 1/(5m)$. This variant modifies the R-AIPP method by adding conditions that allow the stepsize λ to increase between subproblems. More specifically, the R-AIPPv2 method doubles the value of λ at the end of iteration k when: (a) λ has never been halved in step 1 or 2 and (b) the number of inner iterations performed by the R-ACG algorithm in step 1 is less than 250. All R-AIPP variants are run with $\theta = 4$, a problem–specific value of τ, and adaptively estimate the constant $\widetilde{M}$ that is used in each iteration of the R-ACG algorithm.

We now make three remarks about the above R-AIPP variants and the AIPP method of [15]. First, while both the R-AIPPc and AIPP method choose the stepsizes $\{\lambda_k\}$ to be constant, the former method differs from the latter one in that it uses a more relaxed criterion, i.e., (2.9) and (2.10), for solving the k^{th} prox subproblem (1.6). Moreover, the limited numerical experiments in Appendix A.4 show that this relaxation

drastically improves upon the efficiency of the AIPP method, regardless of the magnitude of the ratio M/m. As we believe that this effect would observed in the other problem instances of this section, we choose not to include the AIPP method as part of our suite of benchmark algorithms for the sake of brevity. Second, the R-AIPPv1 and R-AIPPv2 methods differ from the R-AIPPc method in that they permit the stepsizes $\{\lambda_k\}$ to be significantly larger than the constant ones chosen for the R-AIPPc method. As will be observed in the numerical experiments below, this can drastically improve the efficiency of the adaptive stepsize R-AIPP variants. Third, in view of the descriptions of the R-AIPP variants in the previous paragraph, both the R-AIPPc and R-AIPPv1 methods are instances of the R-AIPP method while the R-AIPPv2 method is not. However, the R-AIPPv2 method is clearly an instance of the GD framework, and hence a similar analysis to the one in Section 4 may be used to establish its ACG iteration complexity. For sake of brevity we omit its analysis in this paper.

We now describe the three other nonconvex composite optimization algorithms considered. The first algorithm is an implementation of the unified problem-parameter free accelerated gradient (UPFAG) method that is proposed and analyzed in [10]. The particular implementation considered is the UPFAG-fullBB method, which utilizes a Barzilai–Borwein type stepsize selection strategy and is described in [10, Section 4]. Its input parameters include $(\gamma_1, \gamma_2, \gamma_3) = (0.4, 0.4, 1.0)$ and $(\delta, \sigma) = (10^{-2}, 10^{-10})$. The second algorithm is an implementation of the NC-FISTA method in [19]. The particular implementation considered uses input parameters $(\xi, \lambda) = (1.05m, 0.99/M)$. The third algorithm is an implementation of the accelerated gradient (AG) method that is proposed and analyzed in [9]. The particular implementation considered is Algorithm 2, which is described in [9, Section 2].

Finally, we state some additional details about the numerical experiments. First, for each linearly–set–constrained problem of the form given in (5.1), the quadratic penalty method used to solve it starts with the initial penalty parameter chosen to be $c_0 = \max\{10^{-10}, (1000m - L)/\|A\|^2\}$. Second, each algorithm is run with a time limit of 4000 seconds. If an algorithm does not terminate with a solution for a particular problem instance, we do not report any details about its iteration count or function value at the point of termination and the runtime for that instance is marked with a [*] symbol. Third, the iterations listed in the tables this section include backtracking iterations if a parameter line search method is used as part of the algorithm. Finally, all algorithms described at the beginning of this section are implemented in MATLAB 2019a and are run on Linux 64-bit machines each containing Xeon E5520 processors and at least 8 GB of memory.

6.1 Unconstrained problems

This subsection examines the performance of the R-AIPP method as a nonconvex composite optimization solver for solving problems of the form given in (1.4). Given a function pair (g, h) satisfying assumptions (A1)–(A3) with $\phi = g + h$, tolerance $\hat{\rho} > 0$, and an initial point $z_0 \in \operatorname{dom} h$, each algorithm seeks a pair $(\hat{z}, \hat{v})$ satisfying

$$\hat{v} \in \nabla g(\hat{z}) + \partial h(\hat{z}), \quad \frac{\|\hat{v}\|}{\|\nabla g(z_0)\| + 1} \leq \hat{\rho}. \tag{6.1}$$

Two problems are considered, namely: (i) the quadratic matrix problem; and (ii) the support vector machine problem in [10].

All methods that terminated within 4000 seconds converged to the same objective value, which, for each table in this subsection, is given in a column labeled $\phi(\hat{z})$. The bold numbers in each of the aforementioned tables highlight the algorithm that performed the most efficiently in terms of iteration count or total runtime.

6.1.1 Quadratic matrix problem

Given a pair of dimensions $(l, n) \in \mathbb{N}^2$, scalar pair $(\alpha_1, \alpha_2) \in \Re^2_{++}$, linear operators $\mathcal{B} : S^n_+ \mapsto \Re^n$ and $\mathcal{C} : S^n_+ \mapsto \Re^l$ defined by

$$[\mathcal{B}(z)]_j = \langle B_j, z \rangle_F, \quad [\mathcal{C}(z)]_i = \langle C_i, z \rangle_F,$$

for matrices $\{B_j\}_{j=1}^n, \{C_i\}_{i=1}^l \subseteq \Re^{n \times n}$, positive diagonal matrix $D \in \Re^{n \times n}$, and vector $d \in \Re^l$, this sub–subsection considers the following quadratic matrix (QM) problem:

$$\begin{aligned} \min_z \ & \frac{\alpha_1}{2}\|\mathcal{C}(z) - d\|^2 - \frac{\alpha_2}{2}\|D\mathcal{B}(z)\|^2 \\ \text{s.t. } & z \in P_n, \end{aligned}$$

where $P_n = \{z \in S_+^n : \operatorname{tr} z = 1\}$ denotes the n–dimensional spectraplex.

We now describe the experiment parameters for the instances considered. First, the dimensions were set to be $(l, n) = (50, 200)$ and only 2.5% of the entries of the submatrices B_j and C_i being nonzero. Second, the entries of B_j, C_i, and d (resp., D) are generated by sampling from the uniform distribution $\mathcal{U}[0, 1]$ (resp., $\mathcal{U}[1, 1000]$). Third, the initial starting point is $z_0 = I_n/n$, where I_n is the n-dimensional identity matrix. Fourth, with respect to the termination criterion (6.1), the inputs, for every $z \in S_+^n$, are

$$g(z) = \frac{\alpha_1}{2}\|\mathcal{C}(z) - d\|^2 - \frac{\alpha_2}{2}\|D\mathcal{B}(z)\|^2, \quad h(z) = \delta_{P_n}(z), \quad \hat{\rho} = 10^{-7}.$$

Fifth, the R-AIPP variants used a parameter value of $\tau = 10000$. Finally, each problem instance considered is based on a specific curvature pair $(m, M) \in \Re^2_{++}$ for which the scalar pair $(\alpha_1, \alpha_2) \in \Re^2_{++}$ is selected so that $M = \lambda_{\max}(\nabla^2 g)$ and $-m = \lambda_{\min}(\nabla^2 g)$.

We now present the numerical tables for this set of problem instances. We start with instances in which m is fixed.

M	m	$\phi(\hat{z})$	Iteration Count					
			UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
10^2	10^0	–1.74E–02	3892	2045	8670	8266	7627	**1093**
10^4	10^0	3.67E–01	9809	8642	7064	3250	3691	**1185**
10^6	10^0	3.84E+01	23388	11861	7039	1270	1268	**1174**

Table 6.1: Iteration counts for QM problems with fixed m.

M	m	$\phi(\hat{z})$	Runtime (seconds)					
			UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
10^2	10^0	–1.74E–02	511.91	151.58	880.19	935.50	981.49	**119.87**
10^4	10^0	3.67E–01	1304.09	683.95	687.16	287.37	334.41	**106.54**
10^6	10^0	3.84E+01	2804.38	774.74	615.91	100.27	102.29	**94.16**

Table 6.2: Runtimes for QM problems with fixed m.

We now present instances where $m = M$.

M	m	$\phi(\hat{z})$	Iteration Count					
			UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
10^2	10^2	-2.06E+01	18	38	79	161	**10**	20
10^4	10^4	-2.06E+03	19	39	80	217	**7**	21
10^6	10^6	-2.06E+05	19	39	80	175	**8**	20

Table 6.3: Iteration counts for QM problems with $m = M$.

M	m	$\phi(\hat{z})$	Runtime (seconds)					
			UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
10^2	10^2	-2.06E+01	1.73	1.71	4.80	16.07	**1.23**	2.20
10^4	10^4	-2.06E+03	1.68	1.91	4.89	19.67	**0.70**	2.35
10^6	10^6	-2.06E+05	2.06	2.06	4.73	16.32	**0.57**	2.08

Table 6.4: Runtimes for QM problems with $m = M$.

6.1.2 Support vector machine problem

Given a pair of dimensions $(n, k) \in \mathbb{N}^2$, matrix $U \in \Re^{n\times k}$, and vector $v \in \{-1, +1\}^n$, this sub–subsection considers the following (sigmoid) support vector machine (SVM) problem

$$\min_z \ \frac{1}{k}\sum_{i=1}^{k}[1 - \tanh(v_i \langle u_i, z\rangle)] + \frac{1}{2k}\|z\|^2,$$

where u_i denotes the i^{th} column of U.

We now describe the experiment parameters for the instances considered. First, the entries of U are generated by sampling from the uniform distribution $\mathcal{U}[0, 1]$, with only 5% of the entries being nonzero, and $v = \text{sgn}(U^T x)$ where the entries of x are sampled from the uniform distribution over the k–dimensional ball centered at 0 with radius 50. Second, the initial starting point is $z_0 = 0$. Third, the curvature parameters for each problem instance are $m = M = (4\sqrt{3}\|U\|_F^2)/(9k) + 1/k$. Fourth, with respect to the termination criterion (6.1), the inputs, for every $z \in \Re^n$, are

$$g(z) = \frac{1}{k}\sum_{i=1}^{k}[1 - \tanh(v_i \langle u_i, z\rangle)] + \frac{1}{2k}\|z\|^2, \quad h(z) = 0, \quad \hat{\rho} = 10^{-3}.$$

Fifth, the R-AIPP variants used a parameter value of $\tau = 5000$. Finally, each problem instance considered is based on a specific dimension pair $(n, k) \in \mathbb{N}^2$.

We now present the numerical tables for this set of problem instances.

n	k	$\phi(\hat{z})$	Iteration Count					
			UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
1000	500	2.37E–01	82	3025	783	8234	889	**57**
2000	1000	1.61E–01	197	8361	1192	22706	1227	**85**
4000	2000	1.05E–01	1128	-	1347	-	1651	**97**
8000	4000	6.67E–02	372	-	1647	-	-	**148**

Table 6.5: Iteration counts for SVM problems.

n	k	$\phi(\hat{z})$	Runtime (seconds)					
			UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
1000	500	2.37E–01	3.58	49.00	12.80	389.39	35.21	**1.74**
2000	1000	1.61E–01	29.05	473.67	65.79	3626.51	164.56	**7.73**
4000	2000	1.05E–01	1076.09	4000.00*	437.80	4000.00*	1059.98	**47.75**
8000	4000	6.67E–02	1118.84	4000.00*	1975.18	4000.00*	4000.00*	**177.03**

Table 6.6: Runtimes for SVM problems.

6.2 Linearly constrained problems

This subsection examines the performance of the R-QP-AIPP method as a nonconvex linearly–set–constrained composite optimization solver for solving problems of the form given in (5.1). Given a linear operator A, convex set S, function pair (f, h) satisfying assumptions (C1)–(C3), tolerance pair $(\hat{\rho}, \hat{\eta}) \in \Re^2_{++}$, and an initial point $z_0 \in \text{dom}\, h$, each algorithm seeks a triple $([\hat{z}, \hat{s}], \hat{p}, \hat{v})$ satisfying

$$\begin{gathered}\hat{v} \in \nabla f(\hat{z}) + \partial h(\hat{z}) + A^*\hat{p}, \quad \frac{\|\hat{v}\|}{\|\nabla f(z_0)\| + 1} \le \hat{\rho}, \\ \|A\hat{z} - \hat{s}\| \le \hat{\eta}, \quad \hat{p} \in N_S(\hat{s}).\end{gathered} \tag{6.2}$$

Three problems are considered, namely: (i) the linearly–constrained quadratic matrix problem; (ii) the sparse principal component analysis problem in [11]; and (iii) the bounded matrix completion problem in [33].

The bold numbers in each of the tables in this subsection highlight the algorithm that performed the most efficiently in terms of iteration count or total runtime.

6.2.1 Linearly–constrained quadratic matrix problem

Given a pair of dimensions $(l, n) \in \mathbb{N}^2$, scalar pair $(\alpha_1, \alpha_2) \in \Re^2_{++}$, linear operators $\mathcal{A} : S^n_+ \mapsto \Re^l$, $\mathcal{B} : S^n_+ \mapsto \Re^n$, and $\mathcal{C} : S^n_+ \mapsto \Re^l$ defined by

$$[\mathcal{A}(z)]_i = \langle A_i, z\rangle_F\,, \quad [\mathcal{B}(z)]_j = \langle B_j, z\rangle_F\,, \quad [\mathcal{C}(z)]_i = \langle C_i, z\rangle_F\,,$$

for matrices $\{A_i\}^l_{i=1}, \{B_j\}^n_{j=1}, \{C_i\}^l_{i=1} \subseteq \Re^{n\times n}$, positive diagonal matrix $D \in \Re^{n\times n}$, and vector pair $(b, d) \in \Re^l \times \Re^l$, this sub–subsection considers the following linearly–constrained quadratic matrix (LCQM) problem:

$$\begin{aligned} \min_z \ & \frac{\alpha_1}{2}\|\mathcal{C}(z) - d\|^2 - \frac{\alpha_2}{2}\|D\mathcal{B}(z)\|^2 \\ \text{s.t. } & \mathcal{A}(z) \in \{b\}, \quad z \in P_n, \end{aligned}$$

where $P_n = \{z \in S^n_+ : \operatorname{tr} z = 1\}$ denotes the n–dimensional spectraplex.

We now describe the experiment parameters for the instances considered. First, the dimensions were set to be $(l, n) = (50, 200)$ and only 1.0% of the entries of the submatrices A_i, B_j, and C_i being nonzero. Second, the entries of A_i, B_j, C_i, b, and d (resp., D) were generated by sampling from the uniform distribution $\mathcal{U}[0, 1]$ (resp., $\mathcal{U}[1, 1000]$). Third, the initial starting point z_0 was chosen to be a random point in S^n_+. More specifically, three unit vectors $\nu_1, \nu_2, \nu_3 \in \Re^n$ and three scalars $e_1, e_2, e_2 \in \Re_+$ are first generated by sampling vectors $\widetilde{\nu}_i \sim \mathcal{U}^n[0, 1]$ and scalars $\widetilde{d}_i \sim \mathcal{U}[0, 1]$ and setting $\nu_i = \widetilde{\nu}_i/\|\widetilde{\nu}_i\|$ and $e_i = \widetilde{e}_i/(\sum^3_{j=1} \widetilde{e}_i)$ for $i = 1, 2, 3$. The initial iterate for the first subproblem is then set to $z_0 = \sum^3_{i=1} e_i\nu_i\nu_i^T$. Fourth, with respect to the termination criterion (6.1), the inputs, for every $z \in S^n_+$, are

$$\begin{aligned} f(z) = \frac{\alpha_1}{2}\|\mathcal{C}(z) - d\|^2 - \frac{\alpha_2}{2}\|D\mathcal{B}(z)\|^2, \quad h(z) = \delta_{P_n}(z), \\ A(z) = \mathcal{A}(z), \quad S = \{b\}, \quad \hat{\rho} = 10^{-3}, \quad \hat{\eta} = 10^{-3}. \end{aligned}$$

Fifth, the R-AIPP variants used a parameter value of $\tau = 5000$. Finally, each problem instance considered is based on a specific curvature pair $(m, M) \in \Re^2_{++}$ for which the scalar pair $(\alpha_1, \alpha_2) \in \Re^2_{++}$ is selected so that $M = \lambda_{\max}(\nabla^2 f)$ and $-m = \lambda_{\min}(\nabla^2 f)$.

We now present the numerical tables for this set of problem instances.

L	m	Iteration Count					
		UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
10^1	10^0	2148	12758	8739	1797	1675	**998**
10^2	10^0	1615	8957	5253	1206	**1103**	1153
10^3	10^0	3967	26383	5926	1570	**1489**	1555

Table 6.7: Iteration counts for LCQM problems.

L	m	Runtime (seconds)					
		UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
10^1	10^0	274.13	958.47	883.50	205.48	192.35	**103.60**
10^2	10^0	218.05	684.10	531.88	124.45	117.54	**117.32**
10^3	10^0	481.51	1997.85	615.14	165.38	**156.69**	164.04

Table 6.8: Runtimes for LCQM problems.

6.2.2 Sparse principal component analysis problem

Given integer k, positive scalar pair $(\nu, b) \in \Re^2_{++}$, and matrix $\Sigma \in S^n_+$, this sub–subsection considers the following sparse principal component analysis (PCA) problem:

$$\begin{aligned} \min_{\Pi,\Phi} \ & \langle \Sigma, \Pi\rangle_F + \sum_{i,j=1}^n q_\nu(\Phi_{ij}) + \nu \sum_{i,j=1}^n |\Phi_{ij}| \\ \text{s.t. } & \Pi - \Phi = 0, \quad (\Pi, \Phi) \in \mathcal{F}^k \times \Re^{n\times n} \end{aligned}$$

where $\mathcal{F}^k = \{z \in S^n_+ : 0 \preceq z \preceq I, \operatorname{tr} M = k\}$ denotes the k–Fantope and q_ν is the minimax concave penalty (MCP) function given by

$$q_\nu(t) := \begin{cases} -t^2/(2b), & \text{if } |t| \le b\nu, \\ b\nu^2/2 - \nu|t|, & \text{if } |t| > b\nu, \end{cases} \quad \forall t \in \Re.$$

We now describe the experiment parameters for the instances considered. First, the scalar parameters are chosen to be $(\nu, b) = (100, 100, 0.1)$. Second, the matrix Σ is generated according to an eigenvalue decomposition $\Sigma = P\Lambda P^T$, based on a parameter pair (s, k), where k is as in the problem description and s is a positive integer. In particular, we choose $\Lambda = (100, 1, ..., 1)$, the first column of P to be a sparse vector whose first s entries are $1/\sqrt{s}$, and the other entries of P to be sampled randomly from the standard Gaussian distribution. Third, the initial starting point is $(\Pi_0, \Phi_0) = (D_k, 0)$ where D_k is a diagonal matrix whose first k entries are 1 and whose remaining entries are 0. Fourth, the curvature parameters for each problem instance are $m = M = 1/b$. Fifth, with respect to the termination criterion (6.1), the inputs, for every $(\Pi, \Phi) \in S^n_+ \times \Re^{n\times n}$, are

$$f(\Pi, \Phi) = \langle \Sigma, \Pi \rangle_F + \sum_{i,j=1}^n q_\nu(\Phi_{ij}), \quad h(\Pi, \Phi) = \delta_{\mathcal{F}^k}(\Pi) + \nu \sum_{i,j=1}^n |\Phi_{ij}|,$$
$$A(\Pi, \Phi) := \Pi - \Phi, \quad S = \{0\}, \quad \hat{\eta} = 10^{-3}, \quad \hat{\rho} = 10^{-6}.$$

Sixth, the R-AIPP variants used a parameter value of $\tau = 100000$. Finally, each problem instance considered is based on a specific parameter pair $(s, k) \in \mathbb{N}^2$ where s is part of the process of generating Σ (see the second description above).

We now present the numerical tables for this set of problem instances.

s	k	Iteration Count					
		UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
5	1	-	21979	34584	**4511**	5735	6071
10	1	-	23574	34712	**4954**	5960	5745
15	1	-	27944	32560	**5197**	5867	5822

Table 6.9: Iteration counts for sparse PCA problems.

s	k	Runtime (seconds)					
		UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
5	1	4000.00*	142.11	349.87	**67.32**	83.23	87.99
10	1	4000.00*	153.18	353.59	**72.72**	86.98	83.67
15	1	4000.00*	180.27	328.69	**75.37**	85.56	84.55

Table 6.10: Runtimes for sparse PCA problems.

6.2.3 Bounded matrix completion problem

Given a dimension pair $(p, q) \in \mathbb{N}^2$, positive scalar triple $(\beta, \mu, \theta) \in \Re^3_{++}$, scalar pair $(u, l) \in \Re^2$, matrix $A \in \Re^{p\times q}$, and indices Ω, this sub–subsection considers the following bounded matrix completion (BMC) problem:

$$\min_X \frac{1}{2}\|P_\Omega(X - A)\|^2 + \mu \sum_{i=1}^{\min\{p,q\}} [\kappa(\sigma_i(X)) - \kappa_0\sigma_i(X)] + \bar{\mu}\|X\|_*$$
$$\text{s.t. } l \le X_{ij} \le u \quad \forall (i, j) \in \{1, ..., p\} \times \{1, ..., q\},$$

where $\|\cdot\|_*$ denotes the nuclear norm, the function P_Ω is the linear operator that zeros out any entry not in Ω, the function $\sigma_i(X)$ denotes the i^{th} largest singular value of X, and

$$\kappa_0 := \frac{\beta}{\theta}, \quad \bar{\mu} := \mu\kappa_0, \quad \kappa(t) := \beta \log\left(1 + \frac{|t|}{\theta}\right) \quad \forall t \in \Re.$$

We now describe the experiment parameters for the instances considered. First, the matrix A is the user–movie ratings data matrix of the MovieLens 100K dataset[1], the index set Ω is the set of nonzero entries in A, and the dimension pair is set to be $(p, q) = (610, 9724)$. Second, the initial starting point was chosen to be $X_0 = 0$. Third, the curvature parameters for each problem instance are $m = 2\beta\mu/\theta^2$ and $M = \max\{1, m\}$ and the bounds are set to $(l, u) = (0, 5)$. Fourth, with respect to the termination criterion (6.1), the inputs, for every $X \in \Re^{n\times n}$, are

$$f(X) = \frac{1}{2}\|P_\Omega(X - A)\|^2 + \mu \sum_{i=1}^{\min\{p,q\}} [\kappa(\sigma_i(X)) - \kappa_0\sigma_i(X)], \quad h(X) = \bar{\mu}\|X\|_*,$$
$$A(X) = X, \quad S = \left\{Z \in \Re^{p\times q} : l \le Z_{ij} \le u,\ (i,j) \in \{1,...,p\} \times \{1,...,q\}\right\},$$
$$\hat{\eta} = 10^{-2}, \quad \hat{\rho} = 5 \times 10^{-2}.$$

Fifth, the R-AIPP variants used a parameter value of $\tau = 1000$. Finally, each problem instance considered is based on a specific parameter triple $(\beta, \mu, \theta) \in \Re^3_{++}$.

We now present the numerical tables for this set of problem instances.

β	μ	θ	Iteration Count					
			UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
1/2	$\sqrt{2}$	2	73	-	229	21	**16**	131
1	$\sqrt{2}$	2	132	-	324	73	77	**70**
2	$\sqrt{2}$	2	**76**	-	-	210	356	83

Table 6.11: Iteration counts for BMC problems.

β	μ	θ	Runtime (seconds)					
			UPFAG	NC-FISTA	AG	R-AIPPc	R-AIPPv1	R-AIPPv2
1/2	$\sqrt{2}$	2	1515.79	4000.00*	2498.02	283.48	**254.04**	1305.06
1	$\sqrt{2}$	2	2619.55	4000.00*	3754.03	881.60	900.00	**801.55**
2	$\sqrt{2}$	2	1938.81	4000.00*	4000.00*	2435.49	3657.56	**943.33**

Table 6.12: Runtimes for BMC problems.

6.3 Summary of the numerical experiments

All three variants of the R-AIPP method perform well (relative to the other methods) in the numerical experiments of this section. The R-AIPPv2 method, in particular, is the best performing method in a large proportion of both the unconstrained and constrained problem instances. A potential explanation is that the stepsizes $\{\lambda_k\}$ generated by this method may become significantly larger than the initial stepsize parameters $\lambda_0 = 1$ and $\lambda_0 = 0.9/(2m)$ used in the R-AIPPv1 and R-AIPPc methods, respectively, which in view of the third remark following Proposition 2, speeds up the convergence of the quantity $\min_{i\le k}\|\hat{v}_i\|$ to zero.

Moreover, the adaptive stepsize R-AIPP variants, namely, the R-AIPPv1 and R-AIPPv2 methods, have been shown to perform well regardless of the size of the ratio M/m (see, for example, Tables 6.1–6.4). This is a significant improvement over the AIPP method of [15] which has only been shown to perform well when the ratio M/m is large (see, for example, Table A.1).

7 Concluding remarks

Observing the arguments used in the proofs of Proposition 5.1, Lemma 5.2, and Theorem 5.3, it is straightforward to see that the assumption of $\operatorname{dom} h$ being bounded can be relaxed to assuming that the iterates $\{\hat{z}_l\}$ generated by R-QP-AIPP method of Section 5 be bounded. Explicitly assuming that the iterates satisfy

[1] See the MovieLens 100K dataset containing 610 users and 9724 movies, which is found in https://grouplens.org/datasets/movielens/.

$\|\hat{z}_l\| \le B$, for every $l \ge 1$ and some $B > 0$, the resulting ACG iteration complexity of R-QP-AIPP method is (5.24) with Q_0 replaced by the quantity

$$\varphi_{c_0}(\hat{z}_0) - \varphi_{c_0}^* + 2\left(\varphi^* - \varphi_0^* + \hat{\rho}\,[d_0 + 2B] + \underline{m}\left[d_0^2 + 4B^2\right]\right),$$

where c_0 is as in step 0 of the method, $d_0 := \inf\{\|u - \hat{z}_0\| : z \in \mathcal{F}\}$, the quantity $\underline{m}$ is as in (2.2) with $g = f$, and the quantities $\hat{z}_0, \varphi_c$, and φ_c^* are from the input of the R-QP-AIPP method and (5.9). It should be noted however that we were not able to show that the iterates $\{\hat{z}_l\}$ is bounded. Hence, it is still an open problem to establish the iteration complexity of R-QP-AIPP when $\operatorname{dom} h$ is unbounded.

Note that the description of the R-AIPP (resp. R-QP-AIPP) method of Section 4 (resp. Section 5) does not actually require knowledge of an upper bound m on the parameter $\underline{m}$ in (2.2). This is in contrast to the AIPP (resp. QP-AIPP) method of [15], which requires m in order to establish its validity and iteration complexity. In addition, one could consider a R-AIPP (resp. R-QP-AIPP) variant in which the quantity M (resp. L) is adaptively inferred from its iterates rather than requiring knowledge of its value beforehand. While for the sake of brevity we omit the formal description and analysis of such a variant in this paper, we conjecture that the iteration complexity of the R-AIPP (resp. R-QP-AIPP) variant is as in (4.3) (resp. (5.24)) with M (resp. L) replaced with a quantity that lower bounds it, e.g., the maximum of the lower estimates of M (resp. L) which are inferred by the generated iterates.

A Appendix

This appendix contains proofs and statements of several technical results used in the main body of the paper. It contains three subsections. The first subsection consists of proofs about the refinement procedure of Section 2; the second subsection consists of proofs about the R-ACG algorithm of Section 3; and the third subsection consists of technical results related to Section 5.

A.1 Properties of the refinement procedure

Proof (of Proposition 2.1) It follows from [15, Lemma 19] with $(f, h, L) = (f_\lambda, h_\lambda, M_\lambda)$ that $\Delta \ge 0$ and

$$M_\lambda(z - \hat{z}) \in \nabla f_\lambda(z) + \partial h_\lambda(\hat{z}) = \lambda \nabla g(z) + (z - z^- - v) + \lambda \partial h(\hat{z}).$$

Dividing by λ and rearranging terms yields

$$\frac{1}{\lambda}\left[M_\lambda(z - \hat{z}) + (v + z^- - z)\right] - \nabla g(z) \in \partial h(\hat{z}).$$

Adding $\nabla g(\hat{z})$ to both sides and using the definition of $\hat{v}$ gives

$$\hat{v} = \frac{1}{\lambda}\left[M_\lambda(z - \hat{z}) + (v + z^- - z)\right] + \nabla g(\hat{z}) - \nabla g(z) \in \nabla g(\hat{z}) + \partial h(\hat{z}),$$

which is the inclusion in (2.8).

We now bound $\lambda\|\hat{v}\|$. Since [15, Lemma 19] implies that $\|z - \hat{z}\| \le \sqrt{2M_\lambda^{-1}\Delta}$ and ∇g is M–Lipschitz continuous then

$$\begin{aligned}
\lambda\|\hat{v}\| &\le \|M_\lambda(z - \hat{z})\| + \|v + z^- - z\| + \lambda\|\nabla g(\hat{z}) - \nabla g(z)\| \\
&\le \sqrt{2M_\lambda\Delta} + \|v + z^- - z\| + \lambda M\|\hat{z} - z\| \le \sqrt{2M_\lambda\Delta} + \|v + z^- - z\| + M_\lambda\|\hat{z} - z\| \\
&\le \sqrt{2M_\lambda\Delta} + \|v + z^- - z\| + M_\lambda\sqrt{2M_\lambda^{-1}\Delta} = \|v + z^- - z\| + 2\sqrt{2M_\lambda\Delta},
\end{aligned}$$

which is the inequality in (2.8). ∎

A.2 Properties of the R-ACG algorithm

Proof (of Proposition 3.2(a)) Let ℓ denote the quantity in (3.15). Assume that the R-ACG algorithm has performed ℓ-iterations without declaring failure. In view of step 2 of the R-ACG algorithm, it follows that both (3.10) and (3.11) hold for every $1 \le j \le \ell$. We will show that it must stop successfully at the end of the ℓ^{th} iteration, and hence that the conclusion of the lemma holds. Indeed, note that (3.14), (3.15), and the fact that $\log(1 + t) \le t$ for all $t \ge 0$ implies that

$$A_\ell \ge \frac{2}{1 + 2\widetilde{M}}\left(1 + \frac{1}{2}\sqrt{\frac{1}{1 + 2\widetilde{M}}}\right)^{2(\ell-1)} \ge 2C > 2. \tag{A.1}$$

Combining the triangle inequality, (3.10), the fact that $2/A_\ell \leq 1/C$ and $(2/A_\ell)^2 < 2/A_\ell < 1$ from (A.1), and the relation $(a+b)^2 \leq 2(a^2+b^2)$ for all $a, b \in \Re$, we obtain

$$\begin{aligned}\|u_\ell\|^2 + 2\eta_\ell &\leq \max\{1/A_\ell^2, 1/(2A_\ell)\}(\|A_\ell u_\ell\|^2 + 4A_\ell\eta_\ell) \\ &\leq \max\{1/A_\ell^2, 1/(2A_\ell)\}(2\|A_\ell u_\ell + x_\ell - x_0\|^2 + 2\|x_\ell - x_0\|^2 + 4A_\ell\eta_\ell) \\ &\leq \max\{(2/A_\ell)^2, 2/A_\ell\}\|x_\ell - x_0\|^2 \leq \frac{1}{C}\|x_\ell - x_0\|^2.\end{aligned}$$

On the other hand, using the triangle inequality and the fact that $(a+b)^2 \leq (1+s)a^2 + (1+1/s)b^2$ for every $(a, b, s) \in \Re \times \Re \times R_{++}$ (under the choice of $s = 1/(\sqrt{C}-1)$), we obtain

$$\|x_\ell - x_0\|^2 \leq \frac{\sqrt{C}}{\sqrt{C}-1}\|x_0 - x_\ell + u_\ell\|^2 + \sqrt{C}\|u_\ell\|^2.$$

Combining the previous estimates, we then conclude that

$$\|u_\ell\|^2 + 2\eta_\ell \leq \frac{1}{C-\sqrt{C}}\|x_0 - x_\ell + u_\ell\|^2 + \frac{1}{\sqrt{C}}\|u_\ell\|^2, \tag{A.2}$$

which, after a simple algebraic manipulation, easily implies that

$$\frac{1}{\sqrt{C}-1}\|x_0 - x_\ell + u_\ell\|^2 \geq 2\sqrt{C}\eta_\ell + \left(\sqrt{C}-1\right)\|u_\ell\|^2 \geq \left(\sqrt{C}-1\right)\left(\|u_\ell\|^2 + 2\eta_\ell\right). \tag{A.3}$$

Using the first term in the maximum of (3.16) together with the second inequality of (A.3) immediately implies that (3.12) holds with $j = \ell$. To show that (3.13) holds at $j = \ell$, observe that the definition of ψ in (3.3), (3.11) with $j = \ell$, the second inequality of (A.3), and the second term in the maximum of (3.16) imply that

$$\begin{aligned}\widetilde{\phi}(x_0) - \widetilde{\phi}(x_\ell) \geq \langle u_\ell, x_0 - x_\ell\rangle + \eta_\ell + \frac{1}{2}\|x_\ell - x_0\|^2 &= \frac{1}{2}\left[\|x_0 - x_\ell + u_\ell\|^2 - \left(\|u_\ell\|^2 + 2\eta_\ell\right)\right] \\ &\geq \frac{1}{2}\left[1 + \left(\sqrt{C}-1\right)^{-2}\right]\|x_0 - x_\ell + u_\ell\|^2 \geq \frac{1}{\theta}\|x_0 - x_\ell + u_\ell\|^2.\end{aligned}$$

■

A.3 Results related to Section 5

Lemma A.1 *Assume that $f, h : \Re^n \mapsto (-\infty, \infty]$ satisfy assumptions (C1) and (C3) in Section 5, and that, in addition, f is lower semicontinuous on $\mathrm{cl}\,(\mathrm{dom}\, h)$. Then, $\varphi := f + h$ is a proper lower semicontinuous function which has a global minimum over $\Re^n$.*

Proof Suppose $\bar{z} \in \Re^n \backslash \mathrm{cl}\,(\mathrm{dom}\, h)$. Since $\mathrm{cl}\,(\mathrm{dom}\, h)$ is closed, there exists $\varepsilon > 0$ such that $h(u) = \infty$ for every $u \in \Re^n \backslash \mathrm{cl}\,(\mathrm{dom}\, h)$ satisfying $\|u - \bar{z}\| < \varepsilon$. Hence, $\liminf_{u\to\bar{z}} \varphi(u) = \infty = \varphi(\bar{z})$. Now suppose $\bar{z} \in \mathrm{cl}\,(\mathrm{dom}\, h)$. By the lower semicontinuity of f and h we have

$$\liminf_{u\to\bar{z}} \varphi(u) \geq \liminf_{u\to\bar{z}} f(u) + \liminf_{u\to\bar{z}} h(u) \geq f(\bar{z}) + h(\bar{z}) = \varphi(\bar{z})$$

and, since f is differentiable on $\mathrm{dom}\, h$, the function φ is proper lower semicontinuous with $\mathrm{dom}\,\varphi = \mathrm{dom}\, h$. The last statement of the lemma follows from the well known fact that infimum of a lower semicontinuous function over a bounded set, namely, $\mathrm{dom}\,\varphi$, is always attained. ■

A.4 Comparison with the AIPP method

This subsection presents some computational results that compare the AIPP method of [15] with the R-AIPPc method described at the beginning of Section 6. The main problem of interest for this sub-subsection is the quadratic matrix problem described in Sub-subsection 6.1.1.

We now describe the particular implementation of the AIPP method used in this sub-subsection, which differs from its description in [15] in two ways. First, its innermost subroutine, namely, the ACG method, stops immediately when a quadruple $(\lambda_k, z_k, v_k, \varepsilon_k)$ satisfying (2.14) is found. Second, for each iteration k of the method, a triple $(\hat{z}, \hat{v}, \Delta)$ is generated from the refinement procedure in Section 2 by assigning $(\hat{z}, \hat{v}, \Delta) = RP(\lambda_k, z_{k-1}, z_k, v_k)$, and the method stops with the desired output when $\hat{v}$ satisfies condition (6.1).

All experiment parameters for the R-AIPPc method and the problem instances are as described in Sub-subsection 6.1.1 below, while the AIPP uses a parameter input of $(\sigma, \lambda) = (0.3, 1/(2m))$ for its results.

We now present the numerical tables for this set of problem instances.

M	m	$\phi(\hat{z})$	Iteration Count		Runtime	
			AIPP	R-AIPPc	AIPP	R-AIPPc
10^1	10^0	-3.65E-02	-	**8920**	4000.00*	**1098.14**
10^2	10^0	-1.74E-02	53276	**8317**	3672.23	**737.76**
10^3	10^0	2.05E-02	23645	**4424**	1547.97	**329.06**
10^4	10^0	3.67E-01	7797	**3250**	505.20	**215.69**
10^5	10^0	3.82E+00	2791	**1420**	176.64	**93.87**
10^6	10^0	3.84E+01	3475	**1270**	222.26	**84.84**

Table A.1: Iteration counts for QM problems with fixed m.

References

1. B. Amir. *First-order methods in optimization*, volume 25. SIAM, 2017.
2. N.S. Aybat and G. Iyengar. A first-order smoothed penalty method for compressed sensing. *SIAM J. Optim.*, 21(1):287–313, 2011.
3. N.S. Aybat and G. Iyengar. A first-order augmented Lagrangian method for compressed sensing. *SIAM J. Optim.*, 22(2):429–459, 2012.
4. A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009.
5. Y. Carmon, J.C. Duchi, O. Hinder, and A. Sidford. Accelerated methods for nonconvex optimization. *SIAM J. Optim.*, 28(2):1751–1772, 2018.
6. C. Cartis, N. Gould, and P. Toint. On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems. *SIAM J. Optim.*, 20(6):2833–2852, 2010.
7. Y. Chen, G. Lan, and Y. Ouyang. Optimal primal-dual methods for a class of saddle point problems. *SIAM J. Optim.*, 24(4):1779–1814, 2014.
8. D. Drusvyatskiy and C. Paquette. Efficiency of minimizing compositions of convex functions and smooth maps. *Math. Program.*, pages 1–56, 2018.
9. S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Math. Program.*, 156:59–99, 2016.
10. S. Ghadimi, G. Lan, and H. Zhang. Generalized uniformly optimal methods for nonlinear programming. *Journal of Scientific Computing*, 79(3):1854–1881, Jun 2019.
11. Q. Gu, Z. Wang, and H. Liu. Sparse pca with oracle property. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1529–1537. Curran Associates, Inc., 2014.
12. Y. He and R.D.C. Monteiro. Accelerating block-decomposition first-order methods for solving composite saddle-point and two-player Nash equilibrium problems. *SIAM J. Optim.*, 25(4):2182–2211, 2015.
13. Y. He and R.D.C. Monteiro. An accelerated HPE-type algorithm for a class of composite convex-concave saddle-point problems. *SIAM J. Optim.*, 26(1):29–56, 2016.
14. O. Kolossoski and R.D.C. Monteiro. An accelerated non-euclidean hybrid proximal extragradient-type algorithm for convex-concave saddle-point problems. *Optim. Methods Softw.*, 32(6):1244–1272, 2017.
15. 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 Journal on Optimization*, 29(4):2566–2593, 2019.
16. G. Lan and R.D.C. Monteiro. Iteration-complexity of first-order penalty methods for convex programming. *Math. Program.*, 138(1):115–139, Apr 2013.
17. G. Lan and R.D.C. Monteiro. Iteration-complexity of first-order augmented Lagrangian methods for convex programming. *Math. Program.*, 155(1):511–547, Jan 2016.
18. H. Li and Z. Lin. Accelerated proximal gradient methods for nonconvex programming. *Adv. Neural Inf. Process. Syst.*, 28:379–387, 2015.
19. J. Liang, R.D.C. Monteiro, and C.K. Sim. A fista-type accelerated gradient algorithm for solving smooth nonconvex composite optimization problems. *arXiv preprint arXiv:1905.07010*, 2019.
20. Y.F. Liu, X. Liu, and S. Ma. On the nonergodic convergence rate of an inexact augmented lagrangian framework for composite convex programming. *Math. Oper. Res.*, 44(2):632–650, 2019.
21. Z. Lu and Z. Zhou. Iteration-complexity of first-order augmented Lagrangian methods for convex conic programming. *Available on arXiv:1803.09941*, 2018.
22. R.D.C. Monteiro, C. Ortiz, and B.F. Svaiter. An adaptive accelerated first-order method for convex optimization. *Comput. Optim. Appl.*, 64:31–73, 2016.
23. R.D.C. Monteiro and B.F. Svaiter. Iteration-complexity of a Newton proximal extragradient method for monotone variational inequalities and inclusion problems. *SIAM J. Optim.*, 22(3):914–935, 2012.
24. R.D.C. Monteiro and B.F. Svaiter. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM J. Optim.*, 23(2):1092–1125, 2013.
25. I. Necoara, A. Patrascu, and F. Glineur. Complexity of first-order inexact Lagrangian and penalty methods for conic convex programming. *Optim. Methods Softw.*, pages 1–31, 2017.
26. Y.E. Nesterov. *Introductory lectures on convex optimization : a basic course*. Kluwer Academic Publ., Boston, 2004.
27. Y.E. Nesterov and B.T. Polyak. Cubic regularization of newton method and its global performance. *Math. Program.*, 108(1):177–205, 2006.

28. C. Paquette, H. Lin, D. Drusvyatskiy, J. Mairal, and Z. Harchaoui. Catalyst for gradient-based nonconvex optimization. In *AISTATS 2018-21st International Conference on Artificial Intelligence and Statistics*, pages 1–10, 2018.
29. A. Patrascu, I. Necoara, and Q. Tran-Dinh. Adaptive inexact fast augmented Lagrangian methods for constrained convex optimization. *Optim. Lett.*, 11(3):609–626, 2017.
30. R.T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.*, 14(5):877–898, 1976.
31. M.V. Solodov and B.F. Svaiter. A hybrid approximate extragradient-proximal point algorithm using the enlargement of a maximal monotone operator. *Set-Valued Var. Anal.*, 7(4):323–345, 1999.
32. Y. Xu. Iteration complexity of inexact augmented Lagrangian methods for constrained convex programming. *Available on arXiv:1711.05812*, 2017.
33. Q. Yao and J.T. Kwok. Efficient learning with a family of nonconvex regularizers by redistributing nonconvexity. *J. Mach. Learn. Res.*, 18:179–1, 2017.