

# Self-adaptive ADMM for semi-strongly convex problems

Tianyun Tang<sup>\*</sup>, Kim-Chuan Toh<sup>†</sup>

October 3, 2023

## Abstract

In this paper, we develop a self-adaptive ADMM that updates the penalty parameter adaptively. When one part of the objective function is strongly convex i.e., the problem is semi-strongly convex, our algorithm can update the penalty parameter adaptively with guaranteed convergence. We establish various types of convergence results including accelerated convergence rate of  $O(1/k^2)$ , linear convergence and convergence of iteration points. This enhances various previous results because we allow the penalty parameter to change adaptively. We also develop a partial proximal point method with the subproblem solved by our adaptive ADMM. This enables us to solve problems without semi-strongly convex property. Numerical experiments are conducted to demonstrate the high efficiency and robustness of our method.

**keywords:** Adaptive ADMM, Semi-strongly convex, Partial proximal point method

**Mathematics subject classification:** 90C06, 90C25, 90C90

## 1 Introduction

### 1.1 Adaptive ADMM

In this paper, we consider the following linearly constrained convex optimization problem

$$\min \{f(y) + g(z) \mid By + Cz = b\}, \quad (1)$$

where  $B \in \mathbb{R}^{m \times n_1}$ ,  $C \in \mathbb{R}^{m \times n_2}$ ,  $f : \mathbb{R}^{n_1} \rightarrow (-\infty, \infty]$  and  $g : \mathbb{R}^{n_2} \rightarrow (-\infty, \infty]$  are proper lower semi-continuous and convex functions. One of the most popular methods to solve the problem (1) is alternating direction method of multiplier i.e., ADMM [13, 15]. The convergence analysis of the traditional ADMM often assumes that the penalty parameter is fixed; see for example [7, 10, 12, 21, 27]. Because the efficiency of ADMM is highly sensitive to the penalty parameter, in practice, one would prefer to adaptively update the penalty parameter to avoid laborious tuning; see for example [31, 43]. Existing works on the convergence of

---

<sup>\*</sup>Department of Mathematics, National University of Singapore, Singapore 119076 (ttang@u.nus.edu).

<sup>†</sup>Department of Mathematics, and Institute of Operations Research and Analytics, National University of Singapore, Singapore 119076 (mattohk@nus.edu.sg). The research of this author is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 3 grant call (MOE-2019-T3-1-010).

adaptive ADMM mostly assume that the ratio between two consecutive parameters tends to 1 rapidly, and the algorithm quickly behaves just like the ADMM with a fixed penalty parameter [20, 47–49]. In this paper, we aim to partially close the gap between theory and practice. In detail, we assume that one of the objective function  $g(\cdot)$  is strongly convex, that is, for any  $x \in \mathbb{R}^{n_2}$ ,  $y \in \mathbb{R}^{n_2}$ ,

$$g(x) - g(y) \geq \langle \xi, x - y \rangle + \sigma_g \|x - y\|^2/2, \text{ for any } \xi \in \partial g(y), \quad (2)$$

where  $\sigma_g > 0$  is the strong convexity parameter and  $\|\cdot\|$  stands for the Euclidean norm. We call this problem semi-strongly convex, which is also used in [41]. With this assumption, we may greatly increase the freedom of adaptively adjusting the penalty parameter with guaranteed convergence. In Section 2, we will propose an adaptive ADMM with a special penalty updating scheme. That is, at every iteration, we define an interval to choose the new penalty parameter. The interval’s length may tend to infinity, with floating lower bound and upper bound. We allow the parameter to increase to infinity at the rate of  $O(k)$ , where  $k$  is the iteration counter, and decrease at a linear rate as long as there is a lower bound. We obtain various convergence results within this framework, which will be described in the next subsection.

## 1.2 Convergence analysis

In Section 3, we will analyse the convergence property of our algorithm. We first prove that our algorithm achieves accelerated convergence rate of  $O(1/k^2)$  in terms of objective function value and primal feasibility. Accelerating algorithms for constrained optimization problems has been an active research area; see [4, 5, 26, 34, 40, 41] for examples. Since Goldstein et al. [16] proposed an accelerated ADMM with the convergence rate of  $O(1/k^2)$  by making rather strong assumptions including one that assumes both  $f$  and  $g$  are strongly convex, various attempts have been made to weaken the assumptions while maintaining the convergence rate of  $O(1/k^2)$ . In [44], Xu proposed an accelerated ADMM by increasing the penalty parameter while assuming that one of the component objective functions  $g(\cdot)$  is strongly convex. This work significantly weakens the assumptions of Goldstein et al. Since then, Xu’s framework of increasing the penalty parameter has been generalised and modified by other researchers, see [37, 39, 46]. Among them, Tran-Dinh increases the penalty parameter at a quadratic rate  $O(k^2)$  to achieve the non-ergodic convergence rate of  $O(1/k^2)$ . Although the technique of increasing the penalty parameter can result in a nice convergence rate, this framework has two issues that prevent it from being practical. First, the convergence analysis focuses on the objective function value and primal feasibility, which doesn’t involve the dual variables. In practice, we cannot check the optimality of a solution without the dual variable because the optimal objective value is unknown in advance. Therefore, it is necessary for us to analyse the convergence of the dual variable. Another issue is that, if we keep increasing the penalty parameter to very large values, the dual feasibility will not be penalised enough, and that will deteriorate the convergence speed of the dual feasibility. This observation will also be illustrated in the numerical experiments. In order to overcome these two issues, we update the penalty parameter adaptively to balance the primal and dual KKT residue. Moreover, we prove the convergence of primal-dual iterates, which has not been shown before for accelerated ADMM or adaptive ADMM. Our result

implies that we can use the KKT residue in the stopping criterion because the dual variable also converges. Apart from the sub-linear convergence rate, we also consider the condition for our algorithm to achieve linear convergence. For this aspect, our algorithm can achieve linear convergence if  $g$  is strongly convex and Lipschitz continuously differentiable, and the matrix  $C$  has full row-rank. Note that the linear convergence of ADMM has been studied before, see [3, 10, 14, 22, 33] to just name a few. However, our analysis allows the penalty parameter to change adaptively. As far as we know, this paper is the first to analyse the linear convergence of an adaptive ADMM.

### 1.3 Partial proximal point method

Because our adaptive method is designed to solve a semi-strongly convex problem, in Section 4, we consider how to apply it to solve problem (1) if neither  $f$  nor  $g$  is strongly convex. The idea is that we add a proximal term to one of the variable and solve a sequence of problems. This method is called partial proximal point method (PPPM), which has been used in [24] by Jiang et al. Note that our formulation is different from that in [24] in the sense that their subproblem is strongly convex while our subproblem is only semi-strongly convex. While the convergence of the PPPM was done in [19], the stopping conditions for solving the subproblems are based on practically unverifiable conditions. Here we prove the convergence of the PPPM with the subproblems solved inexactly under verifiable conditions. Because the subproblem becomes a semi-strongly convex problem, we may use the adaptive ADMM to solve it. In Section 6, we conduct numerical experiments to verify the efficiency of the PPPM against different types of ADMM.

### 1.4 Organization of the paper

In Section 2, we present our main algorithm. In Section 3, we conduct the convergence analysis of the algorithm. In Section 4, we discuss the partial proximal point method and its convergence analysis. In Section 5, we discuss some implementation strategies of our algorithm. In Section 6, we present numerical results to verify the robustness, convergence rate and efficiency of IADMM. In Section 7, we give a brief conclusion. The proofs of some results are put in the appendix.

## 2 Self-adaptive ADMM

### 2.1 Preliminaries

Before we state our algorithm, we provide several useful definitions and notations. Define  $x = (y, z)$ , we say that  $(y, z, \lambda)$  is KKT solution of (1) if the following conditions hold.

$$\begin{cases} 0 \in \partial f(y) + B^\top \lambda \\ 0 \in \partial g(z) + C^\top \lambda \\ By + Cz - b = 0. \end{cases} \quad (3)$$

Define

$$\mathcal{F}(x) := f(y) + g(z), \quad \mathcal{A}x := By + Cz, \quad \mathcal{L}(x, \lambda) := \mathcal{F}(x) + \langle \lambda, \mathcal{A}x - b \rangle.$$

We assume that the KKT solution set for (1) is nonempty and let  $(x^* := (y^*, z^*), \lambda^*)$  be a KKT solution of (1). Then we have  $0 \in \partial_x \mathcal{L}(x^*, \lambda^*)$ . Hence  $\mathcal{L}(x, \lambda^*) - \mathcal{L}(x^*, \lambda^*) \geq 0$ , and  $\mathcal{L}(x, \lambda^*) - \mathcal{L}(x^*, \lambda^*) = 0$  if and only if  $0 \in \partial_x \mathcal{L}(x, \lambda^*)$ . Thus, we have the following result

$$\mathcal{L}(x, \lambda^*) - \mathcal{L}(x^*, \lambda^*) = 0, \quad \|\mathcal{A}x - b\|^2 = 0 \iff (x, \lambda^*) \text{ is a KKT solution.} \quad (4)$$

For a given  $n \times n$  symmetric positive semidefinite matrix  $D$  and vectors  $x, y, z \in \mathbb{R}^n$ , define

$$\eta_D(x, y, z) := \langle D(z - y), x - z \rangle, \quad \xi_D(x, y, z) := \frac{1}{2}\|x - y\|_D^2 - \frac{1}{2}\|x - z\|_D^2,$$

where  $\|w\|_D := \sqrt{\langle w, Dw \rangle}$  for any  $w \in \mathbb{R}^n$ . Simple calculation shows that  $\eta_D(x, y, z) = \xi_D(x, y, z) - \|y - z\|_D^2/2$ .

## 2.2 Algorithm statement

Now we present our algorithm as follows.

---

### Algorithm 1 IADMM

---

**Initialization:** Choose  $(x^1, \lambda^1)$  and set constant parameters  $\gamma \in (1, \frac{1+\sqrt{5}}{2})$ ,  $\epsilon, \beta, \tau \in (0, 1)$ , initial penalty parameter  $\beta_1 \in [\beta, +\infty)$ . Choose a matrix  $Q \succeq 0$ . For all  $k \geq 1$ , choose matrix  $P_k \succeq 0$  such that  $\beta_{k+1}P_{k+1} \preceq \beta_k P_k$ . Let  $Q_k = \beta_k Q$ .

**for**  $k = 1, 2, \dots$  **do**

- 1,  $y^{k+1} \in \arg \min_y \left\{ f(y) + \langle \lambda^k, By \rangle + \frac{\beta_k}{2}\|By + Cz^k - b\|^2 + \frac{1}{2}\|y - y^k\|_{P_k}^2 \right\}$
- 2,  $z^{k+1} \in \arg \min_z \left\{ g(z) + \langle \lambda^k, Cz \rangle + \frac{\beta_k}{2}\|By^{k+1} + Cz - b\|^2 + \frac{1}{2}\|z - z^k\|_{Q_k}^2 \right\}$
- 3,  $\lambda^{k+1} = \lambda^k + \gamma\beta_k(By^{k+1} + Cz^{k+1} - b)$
- 4, Choose  $\beta_{k+1} \in \left[ \max\{\beta, \tau\beta_k\}, \sqrt{\beta_k^2 + \frac{(1-\epsilon)\sigma_g\beta_k}{\lambda_{\max}(C^\top C + Q)}} \right]$

**end for**

---

Algorithm 1 is similar to the traditional (proximal) ADMM. The only difference is that in step 4, we choose a new penalty parameter in an interval containing the current penalty parameter. This is why we call the algorithm IADMM, where "I" stands for interval. The parameters  $(\epsilon, \beta) > 0$  are introduced only for theoretical analysis. In practice, we may choose  $(\epsilon, \beta)$  to be small numbers. For simplicity, we only consider the case where the step-length  $\gamma > 1$  since in practice this choice typically will lead to a faster convergence compared to the case where  $\gamma \in (0, 1)$ . But note that our IADMM still works for the latter case. Some remarks on Algorithm IADMM are in order. First, the algorithm is still applicable to the case where the function  $g(\cdot)$  is not strongly convex, i.e.,  $\sigma_g = 0$ . In this case, the penalty parameters  $\{\beta_k\}$  must be non-increasing. Second, when the parameters  $\beta_k$  is fixed for all  $k$ , IADMM reduces to the proximal ADMM in [12] when we set  $P_k = \beta_k P$  for some given  $P \succeq 0$ . Third, we can also add a smooth function to the  $g$ -part, and perform a majorization in every iteration like the algorithm in [44]. The convergence analysis is similar but includes more tedious details. For simplicity, we only consider problem (1). Last, we assume that every subproblem is well-defined with an optimal solution.

### 3 Convergence rate analysis

In this section, we will analyse the convergence property of IADMM. We first state some useful lemmas. Their proofs are put in the appendix.

#### 3.1 Useful lemmas

The following lemma serves as the foundation in the convergence analysis of our IADMM. Many theorems later are based on this lemma. Note that it holds even if  $\sigma_g = 0$ .

**Lemma 3.1.** *Let  $\delta := 1 + \gamma - \gamma^2 > 0$ . Then for any  $(x, \lambda)$  satisfying  $\mathcal{A}x - b = 0$ , we have*

$$\begin{aligned} & \beta_k \left( \mathcal{L}(x^{k+1}, \lambda) - \mathcal{L}(x, \lambda) \right) + \frac{\delta \beta_{k-1}^2}{2\gamma} \|\mathcal{A}x^k - b\|^2 + \frac{\delta \beta_k^2}{2} \|C(z^{k+1} - z^k)\|^2 \\ & + \frac{\beta_k}{2} \|y^k - y^{k+1}\|_{P_k}^2 + \frac{\beta_k^2}{2} \|z^k - z^{k+1}\|_Q^2 \\ & \leq \Phi_k(x, \lambda) - \Phi_{k+1}(x, \lambda) - \sigma_g \beta_k \|z^k - z^{k+1}\|^2 - \frac{\epsilon \sigma_g \beta_k}{2} \|z - z^{k+1}\|^2, \end{aligned} \quad (5)$$

where

$$\begin{aligned} \Phi_k(x, \lambda) &= \frac{1}{2\gamma} \|\lambda - \lambda^k\|^2 + \frac{(2 - \gamma)\beta_{k-1}^2}{2} \|\mathcal{A}x^k - b\|^2 + \frac{\beta_k^2}{2} \|z - z^k\|_{C^\top C + Q}^2 \\ &+ \frac{\beta_{k-1}^2}{2} \|z^k - z^{k-1}\|_Q^2 + \frac{\beta_k}{2} \|y - y^k\|_{P_k}^2. \end{aligned} \quad (6)$$

In the above lemma,  $\Phi_k(x, \lambda)$  serves as a kind of energy function for us to measure the progress of each IADMM iteration. In particular, the left-hand-side of (5) gives the reduction in the “energy” one can expect at each iteration.

The next lemma also appears in Xu’s convergence analysis of accelerated ADMM in [44].

**Lemma 3.2.** *Consider a continuous function  $D(\lambda)$ . Suppose for any  $\lambda \in \mathbb{R}^m$ ,  $\mathcal{L}(x^k, \lambda) - \mathcal{L}(x^*, \lambda) \leq h(k)D(\lambda)$ , where  $h(k) \geq 0$ . Then  $\|\mathcal{A}x^k - b\| = O(h(k))$ ,  $|\mathcal{F}(x^k) - \mathcal{F}(x^*)| = O(h(k))$*

With Lemma 3.1 and Lemma 3.2, we are able to state and prove the convergence results about accelerated convergence, primal-dual iterative convergence and linear convergence in the following three subsections respectively.

#### 3.2 Ergodic convergence rate of $O(1/k^2)$

**Theorem 3.3.** *Define  $v^k := \frac{\sum_{i=1}^k \beta_i x^{i+1}}{\sum_{i=1}^k \beta_i}$  and  $\gamma_k := \sum_{i=1}^k \beta_i$ . Then*

$$|\mathcal{F}(v^k) - \mathcal{F}(x^*)| = O(1/\gamma_k), \quad \|\mathcal{A}v^k - b\| = O(1/\gamma_k).$$

*Proof.* Since the right-hand-side of (5) in Lemma 3.1 is summable, we choose  $x = x^*$  for some optimal solution and take summation of the inequality (5) from 1 to  $k$ , then we get

$$\sum_{i=1}^k \beta_i (\mathcal{L}(x^{i+1}, \lambda) - \mathcal{L}(x^*, \lambda)) \leq D(\lambda) := \Phi_1(x^*, \lambda). \quad (7)$$

Note that when deriving (7), we have ignored many nonnegative terms in (5). From the convexity of  $\mathcal{L}(x, \lambda)$  as a function of  $x$ , we have

$$\gamma_k \left( \mathcal{L}(v^k, \lambda) - \mathcal{L}(x^*, \lambda) \right) \leq D(\lambda). \quad (8)$$

By applying Lemma 3.2 to the above inequality, we get Theorem 3.3.  $\blacksquare$

Since  $\beta_i \geq \beta$  for any  $i$ , then  $\gamma_k = \Omega(k)^1$ . The following corollary can be derived from Theorem 3.3 directly.

**Corollary 3.4.** *Let  $v^k$ ,  $\gamma_k$  be defined as in Theorem 3.3. Then we have*

$$|\mathcal{F}(v^k) - \mathcal{F}(x^*)| = O(1/k), \quad \|\mathcal{A}v^k - b\| = O(1/k).$$

Moreover, if  $\gamma_k = \Omega(k^2)$ , then

$$|\mathcal{F}(v^k) - \mathcal{F}(x^*)| = O(1/k^2), \quad \|\mathcal{A}v^k - b\| = O(1/k^2).$$

**Remark.** Note that from Corollary 3.4, the convergence rate is at least  $O(1/k)$ , even if  $\sigma_g = 0$ . Also, when  $\sigma_g > 0$ , we see that it is possible to choose  $\beta_i$  such that  $\beta_i = \Omega(i)$ . Indeed, if we choose  $\beta_{i+1}$  to be the upper bound of the interval in Step 4 of Algorithm 1 at every iteration, we get  $\beta_i = \Omega(i)$  and so  $\gamma_k = \Omega(k^2)$ . Thus, our algorithm can achieve the convergence rate of  $O(1/k^2)$  when  $\sigma_g > 0$ . However, we should note that even though the objective value gap and primal feasibility decrease at the rate of  $O(1/k^2)$ , the dual feasibility may not. In the next subsection, we will establish the convergence the the sequence  $(x^k, \lambda^k)$ .

### 3.3 Nonergodic convergence of iteration points

In this section, we give a proof of the convergence of the iteration points of IADMM. Suppose  $(x^*, \lambda^*)$  is a KKT solution. Our convergence theorem is as follows.

**Theorem 3.5.** *Suppose  $\beta_k P_k + \beta_k^2 B^\top B \succeq \Theta \forall k$  for some  $\Theta \succ 0$ . Then  $(y^k, z^k, \lambda^k)$  converges to a KKT solution  $(y^*, z^*, \lambda^*)$  as  $k \rightarrow \infty$ .*

*Proof.* If we choose  $(x, \lambda) = (x^*, \lambda^*)$  in Lemma 3.1 and define the quantity

$$\begin{aligned} \Phi_k := \Phi_k(x^*, \lambda^*) &= \frac{1}{2\gamma} \|\lambda^* - \lambda^k\|^2 + \frac{(2-\gamma)\beta_{k-1}^2}{2} \|\mathcal{A}x^k - b\|^2 + \frac{\beta_k^2}{2} \|z^* - z^k\|_{C^\top C + Q}^2 \\ &\quad + \frac{\beta_{k-1}^2}{2} \|z^k - z^{k-1}\|_Q^2 + \frac{\beta_k}{2} \|y^* - y^k\|_{P_k}^2, \end{aligned} \quad (9)$$

we get

$$\left. \begin{aligned} &\frac{\delta\beta_{k-1}^2}{2\gamma} \|\mathcal{A}x^k - b\|^2 + \frac{\beta_k^2}{2} \|z^k - z^{k+1}\|_Q^2 + \frac{\delta\beta_k^2}{2} \|C(z^k - z^{k+1})\|^2 \\ &+ \frac{\beta_k}{2} \|y^k - y^{k+1}\|_{P_k}^2 + \sigma_g \beta_k \|z^k - z^{k+1}\|^2 + \frac{\epsilon\sigma_g\beta_k}{2} \|z^* - z^{k+1}\|^2 \end{aligned} \right\} \leq \Phi_k - \Phi_{k+1}, \quad (10)$$

---

<sup>1</sup>A sequence  $\{a_k\}_{k \in \mathbb{N}^+}$  is said to be  $\Omega(k)$  if there exists some positive number  $c$  and integer  $N_0$  such that  $a_k \geq ck$  for any  $k \geq N_0$ .

Note that when deriving the above inequality, we have used the inequality  $\mathcal{L}(x^{k+1}, \lambda^*) - \mathcal{L}(x^*, \lambda^*) \geq 0$ . By taking summation in (10), we have that the infinite sum on the left-hand-side sequence is finite. Thus we have the following fact.

**Fact 1.**  $\beta_{k-1}^2 \|\mathcal{A}x^k - b\|^2 = o(1)$ ,  $\beta_k \|y^k - y^{k+1}\|_{P_k}^2 = o(1)$ ,  $\beta_k^2 \|z^k - z^{k+1}\|_{C^\top C + Q}^2 = o(1)$ ,  $\beta_k \|z^{k+1} - z^*\|^2 = o(1)$ , and

$$\sum_{k=1}^{\infty} \frac{\sigma_g}{\beta_k} \left( \beta_k^2 \|z^k - z^{k+1}\|^2 + \frac{\epsilon \beta_k^2}{2} \|z^* - z^{k+1}\|^2 \right) < \infty. \quad (11)$$

Moreover, for the sequence of parameters  $\{\beta_k\}$  in Step 4, we can easily prove that  $\beta_k \leq \beta_1 + \frac{(1-\epsilon)\sigma_g(k-1)}{\lambda_{\max}(C^\top C + Q)} = O(k)$ , which implies that  $\sum_{k=1}^{\infty} 1/\beta_k = \infty$ . Together with the fact that  $\sigma_g > 0$  and (11), we have the following result.

**Fact 2.**  $\liminf_{k \rightarrow \infty} \beta_k^2 \|z^k - z^{k+1}\|^2 + \frac{\epsilon \beta_k^2}{2} \|z^* - z^{k+1}\|^2 = 0$ .

From (10), we know that  $\{\Phi_k\}$  is a nonincreasing sequence and it is bounded. Hence, from the definition of  $\Phi_k$  in (9), we have the boundedness of the following sequences:

$$\{\|\lambda^* - \lambda^k\|\}, \quad \{\sigma_g \beta_k \|z^* - z^k\|^2\}, \quad \{\beta_k^2 \|z^* - z^k\|_{C^\top C + Q}^2\}, \quad \{\beta_k \|y^* - y^k\|_{P_k}^2\}, \quad (12)$$

where the boundedness of the second term comes from Fact 1 and the boundedness of  $\{\beta_{k+1}/\beta_k\}$ . Since  $\{\beta_k/\beta_{k+1}\}$  is bounded, from the third sequence in (12),  $\{\beta_k^2 \|C(z^{k+1} - z^*)\|^2\}$  is also bounded. Next we show that  $\{\|y^* - y^k\|\}$  is bounded. From the convexity of the function  $\|\cdot\|^2$  and  $\mathcal{A}x^{k+1} - b = B(y^{k+1} - y^*) + C(z^{k+1} - z^*)$ , we have the following inequality

$$\beta_k^2 \|B(y^{k+1} - y^*)\|^2 \leq 2\beta_k^2 \|\mathcal{A}x^{k+1} - b\|^2 + 2\beta_k^2 \|C(z^{k+1} - z^*)\|^2,$$

then the boundedness of  $\{\beta_k^2 \|\mathcal{A}x^{k+1} - b\|^2\}$  in Fact 1 and that of  $\{\beta_k^2 \|C(z^{k+1} - z^*)\|^2\}$  (just mentioned above) imply that  $\{\beta_k^2 \|y^{k+1} - y^*\|_{B^\top B}^2\}$  is bounded. Because  $\{\beta_{k+1}/\beta_k\}$  is bounded,  $\{\beta_{k+1}^2 \|y^{k+1} - y^*\|_{B^\top B}^2\}$  is also bounded. From the condition in the theorem, we have the fact that  $\|y^* - y^k\|_{\Theta}^2 \leq \beta_k \|y^* - y^k\|_{P_k}^2 + \beta_k^2 \|y^* - y^k\|_{B^\top B}^2$  and  $\Theta \succ 0$ . This implies that  $\{\|y^* - y^k\|\}$  is bounded.

From the above results, we can conclude that  $\{(y^k, z^k, \lambda^k)\}$  is bounded. From Fact 1, we have that  $z^k \rightarrow z^*$ . Combine these two results and Fact 2, we can see that there exists a sequence  $\{j_k\}_{k \geq 1}$  such that  $(y^{j_k+1}, z^{j_k+1}, \lambda^{j_k+1})$  converges to a limit point  $(y, z^*, \lambda)$ , and  $\beta_{j_k}^2 \|z^{j_k} - z^{j_k+1}\|^2 = o(1)$ ,  $\beta_{j_k}^2 \|z^* - z^{j_k+1}\|^2 = o(1)$ . We summarize the results as follows.

**Fact 3.**  $(y^{j_k+1}, z^{j_k+1}, \lambda^{j_k+1}) \rightarrow (y, z^*, \lambda)$ ,  $\beta_{j_k}^2 \|z^{j_k} - z^{j_k+1}\|^2 = o(1)$ ,  $\beta_{j_k}^2 \|z^* - z^{j_k+1}\|^2 = o(1)$ .

We will next show that  $\|y^{j_k} - y^{j_k+1}\| = o(1)$ . First, from the fact that  $\beta_k^2 \|\mathcal{A}x^{k+1} - b\|^2 = o(1)$  in Fact 1 and the boundedness of  $\{\beta_{k+1}/\beta_k\}$ , we can readily show that  $\beta_k^2 \|\mathcal{A}(x^{k+1} - x^k)\|^2 = o(1)$ . Next, from

$$\begin{aligned} \beta_{j_k}^2 \|B(y^{j_k+1} - y^{j_k})\|^2 &\leq 2\beta_{j_k}^2 \|\mathcal{A}(x^{j_k+1} - x^{j_k})\|^2 + 2\beta_{j_k}^2 \|C(z^{j_k+1} - z^{j_k})\|^2 \\ &\leq 2\beta_{j_k}^2 \|\mathcal{A}(x^{j_k+1} - x^{j_k})\|^2 + 2\lambda_{\max}(C^\top C) \beta_{j_k}^2 \|z^{j_k+1} - z^{j_k}\|^2, \end{aligned}$$

and Fact 3, we get  $\beta_{j_k}^2 \|B(y^{j_k+1} - y^{j_k})\|^2 = o(1)$ . From Fact 1, we also have  $\beta_{j_k} \|y^{j_k} - y^{j_k+1}\|_{P_{j_k}}^2 = o(1)$ . Thus, from the condition in the theorem,  $\|y^{j_k+1} - y^{j_k}\|_\Theta^2 \leq \beta_{j_k} \|y^{j_k} - y^{j_k+1}\|_{P_{j_k}}^2 + \beta_{j_k}^2 \|B(y^{j_k+1} - y^{j_k})\|^2 = o(1)$ . Since  $\Theta \succ 0$ , this implies that  $\|y^{j_k+1} - y^{j_k}\| = o(1)$ .

From (55) and (56) in the Appendix, which are the optimality conditions in Step 1 and Step 2 of Algorithm 1, we know that

$$\begin{aligned} P_{j_k}(y^{j_k} - y^{j_k+1}) + \beta_{j_k} B^\top C(z^{j_k+1} - z^{j_k}) + (\gamma - 1)\beta_{j_k} B^\top (\mathcal{A}x^{j_k+1} - b) &\in \partial f(y^{j_k+1}) + B^\top \lambda^{j_k+1} \\ Q_{j_k}(z^{j_k} - z^{j_k+1}) + (\gamma - 1)\beta_{j_k} C^\top (\mathcal{A}x^{j_k+1} - b) &\in \partial g(z^{j_k+1}) + C^\top \lambda^{j_k+1}. \end{aligned} \quad (13)$$

Since  $\|y^{j_k+1} - y^{j_k}\| = o(1)$  and  $0 \leq P_{j_k} \preceq (\beta_{j_1}/\beta)P_{j_1}$ , we have that  $P_{j_k}(y^{j_k} - y^{j_k+1}) \rightarrow 0$ . From Fact 1, all the following terms,  $\beta_{j_k} B^\top C(z^{j_k+1} - z^{j_k})$ ,  $Q_{j_k}(z^{j_k} - z^{j_k+1})$ ,  $\beta_{j_k} C^\top (\mathcal{A}x^{j_k+1} - b)$  and  $\beta_{j_k} B^\top (\mathcal{A}x^{j_k+1} - b)$  converge to 0. With all of the mentioned convergence results and the demi-closedness of  $\partial f$  and  $\partial g$ , after letting  $k \rightarrow \infty$  in (13), we have that

$$0 \in \partial f(y) + B^\top \lambda, \quad 0 \in \partial g(z^*) + C^\top \lambda.$$

Together with  $\mathcal{A}x - b = \lim_{k \rightarrow \infty} \mathcal{A}x^{j_k+1} - b = 0$ , we know that  $(y, z^*, \lambda)$  is a KKT solution. For convenience, we let  $(y, z^*, \lambda) = (y^*, z^*, \lambda^*)$ .

From (10), we know that  $0 \leq \Phi_{k+1} \leq \Phi_k$  and  $\lim_{k \rightarrow \infty} \Phi_k$  exists. From the condition of  $P_k$  in Algorithm 1, we have that  $\beta_{j_k+1} \|y^* - y^{j_k+1}\|_{P_{j_k+1}}^2 \leq \beta_1 \lambda_{\max}(P_1) \|y^* - y^{j_k+1}\|^2 = o(1)$ . Combine this and Fact 1, Fact 3, the boundedness of  $\{\beta_{j_k+1}/\beta_{j_k}\}$  and the definition of  $\Phi_k$ , we have that  $\lim_{k \rightarrow \infty} \Phi_{j_k+1} = 0$ . Thus  $\lim_{k \rightarrow \infty} \Phi_k = 0$ . From here, we get

$$\beta_k \|y^* - y^k\|_{P_k}^2 = o(1), \quad \beta_k^2 \|z^* - z^k\|_{C^\top C + Q}^2 = o(1), \quad \|\lambda^* - \lambda^k\|^2 = o(1), \quad (14)$$

which implies that  $\lim_{k \rightarrow \infty} \lambda^k = \lambda^*$ . Note that from Fact 1 and  $\beta_k \geq \beta > 0$  for all  $k$ , we have  $\|z^* - z^k\| = o(1)$ , i.e.,  $\lim_{k \rightarrow \infty} z^k = z^*$ . Moreover, from  $\lim_{k \rightarrow \infty} \Phi_k = 0$ , we have that

$$\beta_k^2 \|B(y^k - y^*)\|^2 \leq 2 \left( \frac{\beta_k}{\beta_{k-1}} \right)^2 \beta_{k-1}^2 \|\mathcal{A}x^k - b\|^2 + 2\beta_k^2 \|C(z^k - z^*)\|^2 = o(1). \quad (15)$$

From (14), (15),  $\|y^* - y^k\|_\Theta^2 \leq \beta_k \|y^* - y^k\|_{P_k}^2 + \beta_k^2 \|B(y^k - y^*)\|^2 = o(1)$  and  $\Theta \succ 0$ , we get  $\|y^* - y^k\| = o(1)$ , i.e.,  $\lim_{k \rightarrow \infty} y^k = y^*$ . The proof is completed.  $\blacksquare$

*Remark 3.6.* In Theorem 3.5, we have assumed that  $\sigma_g > 0$ . By modifying the proof slightly, one can prove that the theorem also holds when  $\sigma_g = 0$ , provided  $C^\top C + Q \succ 0$ . Note that in this case,  $\beta_k$  decreases monotonically. More specifically, when  $C^\top C + Q \succ 0$ , one can see from (10) that  $\{\beta_k^2 \|z^* - z^k\|^2\}$  is bounded. From there, one can show that the results in Fact 3 are valid, and the rest of the proof of Theorem 3.5 can carry through.

### 3.4 Nonergodic linear convergence

In this subsection, in addition to assuming that  $g$  is strongly convex with parameter  $\sigma_g > 0$ , we also assume that it is continuously differentiable and  $\nabla g$  is Lipschitz continuous with parameter  $L_g$ . Moreover, we choose  $P_k = 0$  for all  $k \geq 1$  and assume that Step 1 of Algorithm 1 is well defined. To begin with, we choose  $(x, \lambda) = (x^*, \lambda^*)$  in Lemma 3.1 to get the following lemma.



**Lemma 3.7.** *For any KKT solution  $(x^*, \lambda^*)$ ,*

$$\begin{aligned}
& \beta_k \left( \mathcal{L}(x^{k+1}, \lambda^*) - \mathcal{L}(x^*, \lambda^*) \right) + \frac{(2-\gamma)\beta_k^2}{2} \|\mathcal{A}x^{k+1} - b\|^2 + \left( \beta_k^2 + \frac{\sigma_g \beta_k}{\lambda_{\max}(Q)} \right) \|z^{k+1} - z^k\|_Q^2 \\
& + \left( \frac{\beta_{k+1}^2}{2} + \frac{\epsilon \sigma_g \beta_k}{4\lambda_{\max}(C^\top C + Q)} \right) \|z^* - z^{k+1}\|_{C^\top C + Q}^2 + \frac{1}{2\gamma} \|\lambda^* - \lambda^{k+1}\|^2 \\
& \leq \frac{(2-\gamma-\delta/\gamma)\beta_{k-1}^2}{2} \|\mathcal{A}x^k - b\|^2 + \frac{\beta_{k-1}^2}{2} \|z^k - z^{k-1}\|_Q^2 + \frac{\beta_k^2}{2} \|z^* - z^k\|_{C^\top C + Q}^2 + \frac{1}{2\gamma} \|\lambda^* - \lambda^k\|^2 \\
& \quad - \frac{\epsilon \sigma_g \beta_k}{4} \|z^* - z^{k+1}\|^2.
\end{aligned} \tag{16}$$

We also need the following lemma, which is motivated from Lemma 3.2 in [10].

**Lemma 3.8.** *For any  $0 < \alpha < \frac{1}{2}$ ,*

$$\begin{aligned}
& (1-2\alpha)\lambda_{\min}(CC^\top) \|\lambda^{k+1} - \lambda^*\|^2 \\
& \leq \frac{1}{\alpha} \lambda_{\max}(CC^\top) (1-\gamma)^2 \beta_k^2 \|\mathcal{A}x^{k+1} - b\|^2 + \frac{1}{\alpha} \lambda_{\max}(Q) \beta_k^2 \|z^{k+1} - z^k\|_Q^2 + L_g^2 \|z^{k+1} - z^*\|^2.
\end{aligned} \tag{17}$$

Now, we are ready to state the main convergence theorem.

**Theorem 3.9.** *Suppose  $\nabla g$  is Lipschitz continuous with parameter  $L_g$  and  $C$  has full row rank. Choose  $P_k = 0$  for all  $k$  and assume that Step 1 of Algorithm 1 is well-defined. Suppose  $\beta \leq \beta_k \leq \bar{\beta}$  for any  $k$ , then  $\mathcal{L}(x^k, \lambda^*) - \mathcal{L}(x^*, \lambda^*)$ ,  $\|\mathcal{A}x^k - b\|^2$  and  $\|\lambda^k - \lambda^*\|^2$  converges to zero  $R$ -linearly as  $k \rightarrow \infty$ .*

*Proof.* Multiply the inequality in Lemma 3.8 by  $\phi > 0$  such that  $\phi \leq \epsilon \sigma_g \beta / (4L_g^2)$  and add it to the inequality in Lemma 3.7, we get

$$\begin{aligned}
& \beta_k \left( \mathcal{L}(x^{k+1}, \lambda^*) - \mathcal{L}(x^*, \lambda^*) \right) + \left( 2-\gamma - \frac{2\phi}{\alpha} \lambda_{\max}(CC^\top) (1-\gamma)^2 \right) \frac{\beta_k^2}{2} \|\mathcal{A}x^{k+1} - b\|^2 \\
& + \left( \beta_k^2 + \frac{\sigma_g \beta_k}{\lambda_{\max}(Q)} - \frac{\phi}{\alpha} \lambda_{\max}(Q) \beta_k^2 \right) \|z^{k+1} - z^k\|_Q^2 \\
& + \left( \frac{\beta_{k+1}^2}{2} + \frac{\epsilon \sigma_g \beta_k}{4\lambda_{\max}(C^\top C + Q)} \right) \|z^* - z^{k+1}\|_{C^\top C + Q}^2 + \left( \frac{1}{2\gamma} + \phi(1-2\alpha)\lambda_{\min}(CC^\top) \right) \|\lambda^* - \lambda^{k+1}\|^2 \\
& \leq (2-\gamma-\delta/\gamma) \frac{\beta_{k-1}^2}{2} \|\mathcal{A}x^k - b\|^2 + \frac{\beta_{k-1}^2}{2} \|z^k - z^{k-1}\|_Q^2 + \frac{\beta_k^2}{2} \|z^* - z^k\|_{C^\top C + Q}^2 + \frac{1}{2\gamma} \|\lambda^* - \lambda^k\|^2.
\end{aligned}$$

Note that in the last inequality, we removed the nonpositive term  $(\phi L_g^2 - \frac{\epsilon \sigma_g \beta}{4}) \|z^{k+1} - z^*\|^2$ .

By using the fact that  $\beta \leq \beta_k \leq \bar{\beta}$  for any  $k \geq 0$  in the above inequality, we get

$$\begin{aligned}
& \beta_k \left( \mathcal{L}(x^{k+1}, \lambda^*) - \mathcal{L}(x^*, \lambda^*) \right) + \left( 1 + \frac{\delta/\gamma - 2\phi\lambda_{\max}(CC^\top)(\gamma-1)^2/\alpha}{2-\gamma-\delta/\gamma} \right) \frac{(2-\gamma-\delta/\gamma)\beta_k^2}{2} \|\mathcal{A}x^{k+1} - b\|^2 \\
& + \left( 2 + \frac{2\sigma_g}{\lambda_{\max}(Q)\bar{\beta}} - \frac{2\phi}{\alpha}\lambda_{\max}(Q) \right) \frac{\beta_k^2}{2} \|z^{k+1} - z^k\|_Q^2 \\
& + \left( 1 + \frac{\epsilon\sigma_g\beta}{2\lambda_{\max}(C^\top C + Q)\bar{\beta}^2} \right) \frac{\beta_{k+1}^2}{2} \|z^* - z^{k+1}\|_{C^\top C + Q}^2 + \left( 1 + 2\gamma\phi(1-2\alpha)\lambda_{\min}(CC^\top) \right) \frac{1}{2\gamma} \|\lambda^* - \lambda^{k+1}\|^2 \\
& \leq \mathcal{E}(k) := \frac{(2-\gamma-\delta/\gamma)\beta_{k-1}^2}{2} \|\mathcal{A}x^k - b\|^2 + \frac{\beta_{k-1}^2}{2} \|z^k - z^{k-1}\|_Q^2 + \frac{\beta_k^2}{2} \|z^* - z^k\|_{C^\top C + Q}^2 + \frac{1}{2\gamma} \|\lambda^* - \lambda^k\|^2.
\end{aligned}$$

Note that from  $\delta = 1 + \gamma - \gamma^2$  and  $\gamma > 1$ , we have  $2 - \gamma - \delta/\gamma > 0$ . If we choose  $\phi > 0$  to be sufficiently small so that all coefficients in the parentheses on the left-hand-side are positive, then we have that

$$\mathcal{E}(k+1) \leq M(\phi)\mathcal{E}(k), \quad (18)$$

where

$$M(\phi) := \max \left\{ \frac{1}{1 + \frac{\delta/\gamma - 2\phi\lambda_{\max}(CC^\top)(\gamma-1)^2/\alpha}{2-\gamma-\delta/\gamma}}, \frac{1}{2 + \frac{2\sigma_g}{\lambda_{\max}(Q)\bar{\beta}} - \frac{2\phi}{\alpha}\lambda_{\max}(Q)}, \right. \\
\left. \frac{1}{1 + \frac{\epsilon\sigma_g\beta}{2\lambda_{\max}(C^\top C + Q)\bar{\beta}^2}}, \frac{1}{1 + 2\gamma\phi(1-2\alpha)\lambda_{\min}(CC^\top)} \right\}.$$

Note that we have ignored the nonnegative term  $\beta_k (\mathcal{L}(x^{k+1}, \lambda^*) - \mathcal{L}(x^*, \lambda^*))$ . It is easy to see that if  $\phi > 0$  is sufficiently small, then  $0 < M(\phi) < 1$ , which implies that  $\mathcal{E}(k) \rightarrow 0$  linearly. From the definition of  $\mathcal{E}(k)$  and the lower boundedness of  $\beta_k$ , we can see that  $\mathcal{L}(x^k, \lambda^*) - \mathcal{L}(x^*, \lambda^*)$ ,  $\|\mathcal{A}x^k - b\|^2$  and  $\|\lambda^k - \lambda^*\|^2$  all converge R-linearly to zero.  $\blacksquare$

## 4 A partial proximal point method with IADMM for solving non-semi-strongly convex problem

The theoretical analysis in the previous section is based on the semi-strongly convexity of the problem (1). However, in practice, many composite programming problems may not be semi-strongly convex. In this section, in order to resolve this issue, we introduce a partial proximal point method to solve (1) where in each iteration of the algorithm, we solve the following perturbed subproblem with a proximal term inexactly by our IADMM:

$$(y_{k+1}, z_{k+1}) \approx \min \left\{ f(y) + g(z) + \frac{\sigma}{2} \|z - z_k\|^2 : By + Cz = b \right\}, \quad (19)$$

where  $\sigma > 0$ . The subproblem (19) is similar to the subproblem of a proximal point method. However, we only add the proximal term to one of the variables. Since the subproblem (19) is not equivalent to problem (1), we will solve a sequence of such subproblems inexactly to obtain a solution of (1).

---

**Algorithm 2** PPPM

---

**Initialization:** Choose  $(y_1, z_1) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ .

**for**  $k = 1, 2, \dots$  **do**

1, Choose parameter  $\sigma_k \geq 0$

2, Compute an approximate KKT solution  $(y_{k+1}, z_{k+1}, \lambda_{k+1})$  of (19) with  $\sigma = \sigma_k$  by IADMM.

**end for**

---

We use the following inexact KKT condition to measure the accuracy of the subproblem.

$$\begin{cases} \epsilon_k^1 \in \partial f(y_k) + B^\top \lambda_k \\ \epsilon_k^2 \in \partial g(z_k) + C^\top \lambda_k + \sigma_k(z_k - z_{k-1}) \\ \epsilon_k^3 = By_k + Cz_k - b \end{cases} \quad (20)$$

where  $(y_k, z_k, \lambda_k)$  is the outcome of the  $k$ th subproblem and  $(\epsilon_k^1, \epsilon_k^2, \epsilon_k^3) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times \mathbb{R}^m$  is the error term. Note that we use subscripts to differentiate the iterations in the partial proximal point method and IADMM.

The convergence analysis of inexact proximal point methods have been a popular research topic because of its wide applications and connection to augmented Lagrangian methods [9, 11, 29, 35, 50]. Here we give the convergence theorem of the PPPM to make the paper self-contained.

**Theorem 4.1.** *Suppose  $0 \leq \sigma_{k-1} \leq \sigma_k$ ,  $\sum_{k=1}^{\infty} (1 + \|y_k\|) \|\epsilon_k^1\| < \infty$ ,  $\sum_{k=1}^{\infty} (1 + \|z_k\|) \|\epsilon_k^2\| < \infty$  and  $\sum_{k=1}^{\infty} (1 + \|\lambda_k\|) \|\epsilon_k^3\| < \infty$ , then the sequence  $(y_k, z_k, \lambda_k)$  generated from Algorithm 2 satisfies*

$$\lim_{k \rightarrow \infty} \max \left\{ \text{dist}(0, \partial f(y_k) + B^\top \lambda_k), \text{dist}(0, \partial g(z_k) + C^\top \lambda_k), \|Ax^k - b\|, |\mathcal{F}(x_k) - \mathcal{F}(x^*)| \right\} = 0,$$

where  $x^*$  is an optimal solution of (1).

*Proof.* Let  $(x^*, \lambda^*)$  be a KKT solution of problem (1). Combine (20) and the convexity of  $f$  and  $g$ , we obtain the following inequalities

$$f(y_k) + \langle \epsilon_k^1 - B^\top \lambda_k, y^* - y_k \rangle \leq f(y^*), \quad (21)$$

$$g(z_k) + \langle \epsilon_k^2 - C^\top \lambda_k - \sigma_k(z_k - z_{k-1}), z^* - z_k \rangle \leq g(z^*). \quad (22)$$

Adding (21) and (22), we have that

$$\begin{aligned} & \mathcal{F}(x_k) + \langle \epsilon_k^1, y^* - y_k \rangle + \langle \epsilon_k^2, z^* - z_k \rangle + \langle \lambda_k, B(y_k - y^*) + C(z_k - z^*) \rangle \\ & \leq \mathcal{F}(x^*) + \sigma_k \eta(z^*, z_{k-1}, z_k). \end{aligned} \quad (23)$$

Using  $By^* + Cz^* = b$  in (23) together with Cauchy-Schwarz inequality, we get

$$\begin{aligned} \mathcal{F}(x_k) - \mathcal{F}(x^*) + \sigma_k \|z_{k-1} - z_k\|^2 / 2 & \leq (\|y^*\| + \|y_k\|) \|\epsilon_k^1\| + (\|z^*\| + \|z_k\|) \|\epsilon_k^2\| + \|\lambda_k\| \|\epsilon_k^3\| \\ & \quad + \sigma_k \|z_{k-1} - z^*\|^2 / 2 - \sigma_k \|z_k - z^*\|^2 / 2. \end{aligned} \quad (24)$$

Moreover, from the convexity of  $\mathcal{L}(x, \lambda^*)$  and  $0 \in \partial_x \mathcal{L}(x^*, \lambda^*)$  we have that  $\mathcal{L}(x_k, \lambda^*) \geq \mathcal{L}(x^*, \lambda^*)$ . This implies

$$\mathcal{F}(x_k) \geq \mathcal{F}(x^*) - \langle \lambda^*, \mathcal{A}x_k - b \rangle \geq \mathcal{F}(x^*) - \|\lambda^*\| \|\mathcal{A}x_k - b\| \geq \mathcal{F}(x^*) - \|\lambda^*\| \|\epsilon_k^3\|. \quad (25)$$

Substitute (25) into (24) and  $\sigma_{k-1} \geq \sigma_k$ , we get

$$\begin{aligned} 0 \leq & \mathcal{F}(x_k) - \mathcal{F}(x^*) + \sigma_k \|z_{k-1} - z_k\|^2/2 + \|\lambda^*\| \|\epsilon_k^3\| \leq (\|y^*\| + \|y_k\|) \|\epsilon_k^1\| \\ & + (\|z^*\| + \|z_k\|) \|\epsilon_k^2\| + (\|\lambda^*\| + \|\lambda_k\|) \|\epsilon_k^3\| + \sigma_{k-1} \|z_{k-1} - z^*\|^2/2 - \sigma_k \|z_k - z^*\|^2/2. \end{aligned} \quad (26)$$

From the condition in the theorem, we have that the sum of right-hand-side of (26) is upper bounded. Thus, we get

$$\lim_{k \rightarrow \infty} \mathcal{F}(x_k) - \mathcal{F}(x^*) + \sigma_k \|z_{k-1} - z_k\|^2/2 + \|\lambda^*\| \|\epsilon_k^3\| = 0. \quad (27)$$

Using (25) in (27), we get

$$\lim_{k \rightarrow \infty} \mathcal{F}(x_k) - \mathcal{F}(x^*) + \|\lambda^*\| \|\epsilon_k^3\| = 0, \quad \lim_{k \rightarrow \infty} \sigma_k \|z_{k-1} - z_k\|^2 = 0. \quad (28)$$

Because  $0 \leq \sigma_k \leq \sigma_1$ , we have that  $\lim_{k \rightarrow \infty} \sigma_k^2 \|z_{k-1} - z_k\|^2 = 0$ . This implies that  $\lim_{k \rightarrow \infty} \sigma_k \|z_{k-1} - z_k\| = 0$ . Substitute this into (20), we have that

$$\lim_{k \rightarrow \infty} \max \left\{ \text{dist}(0, \partial f(y_k) + B^\top \lambda_k), \text{dist}(0, \partial g(z_k) + C^\top \lambda_k), \|\mathcal{A}x^k - b\| \right\} = 0. \quad (29)$$

Because  $\lim_{k \rightarrow \infty} \|\lambda^*\| \|\epsilon_k^3\| = 0$ , from (28), we get

$$\lim_{k \rightarrow \infty} \mathcal{F}(x_k) - \mathcal{F}(x^*) = 0. \quad (30)$$

Combining (29) and (30), we get Theorem 4.1. ■

Because from Theorem 3.5, the sequence generated of IADMM is convergent. We can control the size of  $(1 + \|y_k\|) \|\epsilon_k^1\|$ ,  $(1 + \|z_k\|) \|\epsilon_k^2\|$  and  $(1 + \|\lambda_k\|) \|\epsilon_k^3\|$  by solving the subproblem accurately enough. Therefore, the condition in Theorem 4.1 can be guaranteed.

## 5 Practical implementation

In this section, we will move on to consider the practical usage of Algorithm 1 and Algorithm 2.

### 5.1 Application in LASSO type problems

In this subsection, we consider the problem

$$\min \{f(\lambda) + g(b - A\lambda)\}, \quad (31)$$

where  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^m$  are given data. We assume that  $f$  and  $g$  are lower semi-continuous and convex,  $g$  is differentiable with gradient that is Lipschitz with modulus  $L_g$ .

We can rewrite the above problem equivalently as  $\min \{f(\lambda) + g(\mu) : A\lambda + \mu = b\}$ . The corresponding dual problem is

$$\min_{\lambda} \left\{ f^*(A^\top z) + g^*(z) - \langle z, b \rangle \right\},$$

which is equivalent to

$$\min \left\{ f^*(y) + g^*(z) - \langle z, b \rangle \mid y - A^\top z = 0 \right\}. \quad (32)$$

The following lemma shows the relation between (31) and (32).

**Lemma 5.1.** *If  $(y^*, z^*, \lambda^*)$  is a KKT solution to (32), then  $\lambda^*$  is an optimal solution to (31).*

*Proof.* From the optimality conditions of (32), we have  $0 \in \partial f^*(y^*) - \lambda^*$ ,  $0 \in \partial g^*(z^*) - b + A\lambda^*$  and  $y^* - A^\top z^* = 0$ . Then  $A^\top z^* \in A^\top \partial g(b - A\lambda^*)$ ,  $y^* \in \partial f(\lambda^*)$ , from which we deduce  $0 \in \partial f(\lambda^*) - A^\top \partial g(b - A\lambda^*)$ . Then  $\lambda^*$  is an optimal solution to  $\min \{f(\lambda) + g(b - A\lambda)\}$ , so  $\lambda^*$  is an optimal solution to (31).  $\blacksquare$

By Proposition 12.60 of [36], we know that  $g^*$  is strongly convex with parameter  $\sigma_g := 1/L_g$ . Thus, we can use Algorithm 1 to solve problem (32). To avoid computing the conjugate function, we may use the following identity (see Theorem 14.3 in [2]) due to Moreau to solve the subproblem.

**Proposition 5.2.** *Let  $F$  be a convex lower semi-continuous function. For any  $\alpha > 0$ , we have*

$$\mathbf{Prox}_{\alpha^{-1}F^*}(x) = x - \frac{1}{\alpha} \mathbf{Prox}_{\alpha F}(\alpha x).$$

With Proposition 5.2, IADMM for solving (32) is presented as follows.

---

**Algorithm 3** IADMM for solving (31)

---

**Initialization:** Given  $(x^1, \lambda^1)$  and constants  $\gamma \in (1, \frac{1+\sqrt{5}}{2})$ ,  $\epsilon, \beta, \tau \in (0, 1)$ , choose the initial penalty parameter  $\beta_1 \in [\beta, \infty)$  and positive semidefinite matrix  $Q$  such that  $AA^\top + Q \succ 0$ . Set  $P_k = 0$ ,  $Q_k = \beta_k Q$ .

**for**  $k = 1, 2, \dots$  **do**

1.  $y^{k+1} = A^\top z^k - \frac{\lambda^k}{\beta_k} - \frac{1}{\beta_k} \mathbf{Prox}_{\beta_k f}(\beta_k A^\top z^k - \lambda^k)$
2.  $z^{k+1} = \arg \min_z \left\{ g^*(z) - \langle z, b \rangle + \langle \lambda^k, -A^\top z \rangle + \frac{\beta_k}{2} \|y^{k+1} - A^\top z\|^2 + \frac{1}{2} \|z - z^k\|_{Q_k}^2 \right\}$
3.  $\lambda^{k+1} = \lambda^k + \gamma \beta_k (y^{k+1} - A^\top z^{k+1})$
4. Choose  $\beta_{k+1} \in \left[ \max\{\beta, \tau \beta_k\}, \sqrt{\beta_k^2 + \frac{(1-\epsilon)\sigma_g \beta_k}{\lambda_{\max}(C^\top C + Q)}} \right]$

**end for**

---

Note that to make subproblem in Step 2 of Algorithm 3 easier to solve, we may choose  $Q := \lambda_{\max}(AA^\top)I - AA^\top$ . Then Step 2 becomes

$$z^{k+1} = a^k - \frac{1}{\beta_k \lambda_{\max}(AA^\top)} \mathbf{Prox}_{\beta_k \lambda_{\max}(AA^\top)g} \left( \beta_k \lambda_{\max}(AA^\top) a^k \right),$$

where

$$a^k = \frac{b + A\lambda^k + \beta_k (\lambda_{\max}(AA^\top)I - AA^\top) z^k + \beta_k A y^{k+1}}{\beta_k \lambda_{\max}(AA^\top)}.$$

## 5.2 Strategies for updating $\beta_k$

In this subsection, we consider how to update the penalty parameter  $\beta_k$ . Note that if we fix  $\beta_k$  as a constant, then IADMM is equivalent to the traditional ADMM. Also, Xu's accelerated ADMM in [44] is essentially IADMM with the following monotone updating strategy:

$$\beta_1 = \beta, \quad \beta_{k+1} = \sqrt{\beta_k^2 + \frac{(1-\epsilon)\sigma_g\beta_k}{\lambda_{\max}(C^\top C + Q)}}, \quad \forall k \in \mathbb{N}^+. \quad (33)$$

It is called accelerated ADMM because the function value gap and primal feasibility achieve the ergodic convergence rate of  $O(1/n^2)$  as shown in Section 3. But as we will see later in the numerical experiments, a better strategy is to adaptively adjust the penalty parameter  $\beta_k$  in IADMM based on the ratio between the normalised primal feasibility  $R_p^{k+1}$  and normalised dual feasibility  $R_d^{k+1}$  for the computed iterate  $(y^{k+1}, z^{k+1}, \lambda^{k+1})$ , where

$$R_p^{k+1} := \frac{\|By^{k+1} + Cz^{k+1} - b\|}{\max\{\|By^{k+1}\|, \|Cz^{k+1}\|, \|b\|\}},$$

$$R_d^{k+1} := \max \left\{ \frac{\|y^{k+1} - \text{Prox}_f(y^{k+1} - B^\top \lambda^{k+1})\|}{\max\{\|y^{k+1}\|, \|B^\top \lambda^{k+1}\|\}}, \frac{\|z^{k+1} - \text{Prox}_g(z^{k+1} - C^\top \lambda^{k+1})\|}{\max\{\|z^{k+1}\|, \|C^\top \lambda^{k+1}\|\}} \right\}.$$

The KKT residue of the computed  $(y^{k+1}, z^{k+1}, \lambda^{k+1})$  is defined to be  $\max\{R_p^{k+1}, R_d^{k+1}\}$ . Our adaptive strategy is that after iteration  $k$ , we check the ratio between  $R_p^{k+1}$  and  $R_d^{k+1}$ , and update  $\beta_k$  as follows:

$$\beta_{k+1} := \begin{cases} \sqrt{\beta_k^2 + \frac{(1-\epsilon)\sigma_g\beta_k}{\lambda_{\max}(C^\top C + Q)}} & R_p^{k+1} > R_d^{k+1}, \\ \max\{\beta, \beta_k/1.5\} & R_p^{k+1} < R_d^{k+1}/10, \\ \beta_k & R_d^{k+1}/10 \leq R_p^{k+1} \leq R_d^{k+1}. \end{cases} \quad (34)$$

It is easy to see that this strategy belongs to the framework of Algorithm 1. Note that updating the penalty according to the ratio between primal and dual feasibility is popular in the literature (see [20, 47]). However, our adaptive strategy has guaranteed convergence. The adaptive strategy for IADMM is not necessarily restricted to the one presented in (34). One can choose any other adaptive strategy, and as long as the new penalty parameter lies in the interval in Step 4 of Algorithm 1, the convergence is guaranteed. In practice, we choose  $\gamma = 1.618$  and  $\epsilon = 10^{-4}$ ,  $\beta = 10^{-6}$ .

## 5.3 Strategies for partial proximal point method

In this section, we consider the implementation of Algorithm 2. Because the convergence analysis in Theorem 4.1 requires  $\sigma_k$  to be monotonically decreasing, we will choose  $\sigma_k = \max\{1/2^k, 10^{-6}\}$ . A geometrically decreasing proximal parameter usually reduces the number of outer iterations in the partial proximal point method. Let  $\hat{R}_p^k, \hat{R}_d^k$  be the primal and dual KKT residue of the  $k$ th subproblem (19). Let  $R_p^k, R_d^k$  be the KKT residue of the original problem (1). Because problems (1) and (19) have the same constraints, we get  $\hat{R}_p^k = R_p^k$ . Also, when the subproblem (19) is solved exactly, we will have that

$R_p^k = \widehat{R}_p^k = \widehat{R}_d^k = 0$  but  $R_d^k$  may not equal to zero. This is because problem (1) is generally not equivalent to subproblem (19). We stop the IADMM for solving the subproblem when the following conditions are satisfied:

$$R_p^k < R_d^k/10, \max \left\{ \widehat{R}_p^k, \widehat{R}_d^k \right\} < 1/(10k^3). \quad (35)$$

The first condition in (35) uses the relation between the primal and dual KKT residues. It has been used in augment Lagrangian method [11]. The second condition in (35) ensures that the KKT residues of the subproblems tend to zero rapidly with the rate  $O(1/k^3)$ . It corresponds to the condition of the error term in Theorem 4.1.

We should add that the initial penalty parameter  $\beta_1$  used by the IADMM to solve the  $k$ -th PPPM subproblem in Algorithm 2 is adaptively adjusted as follows. In detail, if the penalty parameter of the previous IADMM inner loop keeps increasing in the last few iterations, it is likely to be too small and will continue to increase in the new IADMM inner loop. In this case, we set the initial penalty parameter to be  $\eta$  times the last penalty parameter of the previous IADMM inner loop for some constant  $\eta > 1$ . Otherwise, we simply choose the initial penalty parameter to be the same as the last penalty parameter of the previous IADMM inner loop. By doing so, Algorithm 2 can increase the penalty parameter drastically between different outer iterations and refine it by the scheme in (34) at each inner loop. From our numerical experiment, Algorithm 2 can always identify a good penalty parameter after only a few outer iterations and becomes stable after that.

## 6 Numerical experiments

In this section, we apply our algorithms to different problems to demonstrate the robustness, convergence results and efficiency. Although in our theoretical analysis, the proximal term of  $y$  can be nonzero, we simply choose  $P = 0$  in the following numerical experiments. This is because the ADMM subproblem for  $y$  already has a closed form solution and we don't have to add a proximal term to simplify it. All the experiments are run using MATLAB R2021b on a Workstation with a Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz Processor and 128GB RAM.

### 6.1 Testing the robustness of IADMM

In this section, we test several regression problems to verify the robustness of adaptive IADMM.

#### Example 6.1. Total variation regularized least squares problem.

We consider following problem, which is considered in example 6.3 of [25]:

$$\min \left\{ \|Hx - b\|^2/2 + \gamma \sum_{i=1}^{n-1} |x_{i+1} - x_i| : x \in \mathbb{R}^n \right\}, \quad (36)$$

where  $H \in \mathbb{R}^{r \times n}$ ,  $b \in \mathbb{R}^r$ . To solve (36), we apply IADMM to its dual problem as in Subsection 5.1. In this case,  $f(x) = \gamma \sum_{i=1}^{n-1} |x_{i+1} - x_i|$  and  $g(x) = \|x\|^2/2$ . To compute the proximal mapping of  $f$ , we use Condat's direct algorithm in [8]. We use synthetic data similar

to example 6.3 in [25]. We randomly generate the matrix  $H$  as  $H = \text{randn}(r, n) / \text{sqrt}(n)$  and  $b$  as  $b = H * x_{\text{true}} + \text{nf} * \text{randn}(r, 1)$ , where  $x_{\text{true}}$  is a randomly generated vector such that  $(x_2 - x_1, x_3 - x_2, \dots, x_n - x_{n-1})$  only has a few non-zero elements. We choose the noise factor  $\text{nf} = 1 / \text{norm}(H, 'fro')$ .

**Example 6.2. Elastic net regularized support vector machine**

We consider the following problem which is considered in section 3.3 of [44]:

$$\min \left\{ \frac{1}{m} e^\top [y]_+ + \mu_1 \|x\|_1 + \frac{\mu_2}{2} \|x\|^2 : Bx + y = e, x \in \mathbb{R}^p, y \in \mathbb{R}^m \right\}, \quad (37)$$

where  $B \in \mathbb{R}^{m \times p}$  and  $e \in \mathbb{R}^m$  is the vector of all ones. We choose  $\mu_1 = \mu_2 = 0.01$  and generate synthetic data matrix  $B$  in the same way as section 3.3 of [44] (also see section 3.1.1 of [45]).

**Example 6.3. Elastic net regularized problem with square-root loss**

We consider the following problem which is considered in section 5.2. of [39]:

$$\min \left\{ \|x\|_2 + \mu_1 \|y\|_1 + \frac{\mu_2}{2} \|y\|^2 : -x + By = c, x \in \mathbb{R}^n, y \in \mathbb{R}^p \right\}, \quad (38)$$

where  $B \in \mathbb{R}^{n \times p}$  and  $c \in \mathbb{R}^n$ . We choose  $\mu_1 = 0.01$ ,  $\mu_2 = 0.1$  and generate synthetic data matrix  $B$ ,  $c$  in the same way as section 5.2 of [39].

For all the above problems, we test three algorithms: traditional ADMM with fixed-penalty parameter, IADMM with adaptive strategy (34) and ADMM with a heuristic adaptive strategy (which we denote as Heu-ADMM) as follows:

$$\beta_{k+1} := \begin{cases} 1.5\beta_k & R_p^{k+1} > 10R_d^{k+1}, \\ \beta_k / 1.5 & R_p^{k+1} < R_d^{k+1} / 10, \\ \beta_k & R_d^{k+1} / 10 \leq R_p^{k+1} \leq 10R_d^{k+1}. \end{cases} \quad (39)$$

We choose the initial penalty parameter to be from  $10^{-5}$  to  $10^5$ . We stop the algorithms when  $\max\{R_p^{k+1}, R_d^{k+1}\} < 10^{-5}$  or when the maximum iteration number is reached.

The results of this experiment are shown in Figure 1. The  $x$ -axis is the initial penalty parameter and the  $y$ -axis is the number of iterations that the algorithm needs to achieve the accuracy of  $10^{-5}$ . From the plots in Figure 1, we can see that the fixed parameter ADMM is very fast once the penalty parameter is optimally tuned. However, its speed is very sensitive to the initial penalty parameter. However, IADMM and Heu-ADMM can solve all the problems efficiently, without being affected seriously by the initial parameter, because they can adaptively tune their penalty parameters. Apart from the convergence guarantee, our adaptive strategy (34) is also more stable than the heuristic strategy (39). Moreover, when the initial penalty parameter is too small, IADMM is usually faster than Heu-ADMM. One possible reason is that in the strategy (34),  $\beta_k$  has a super-exponential growth when it is too small.



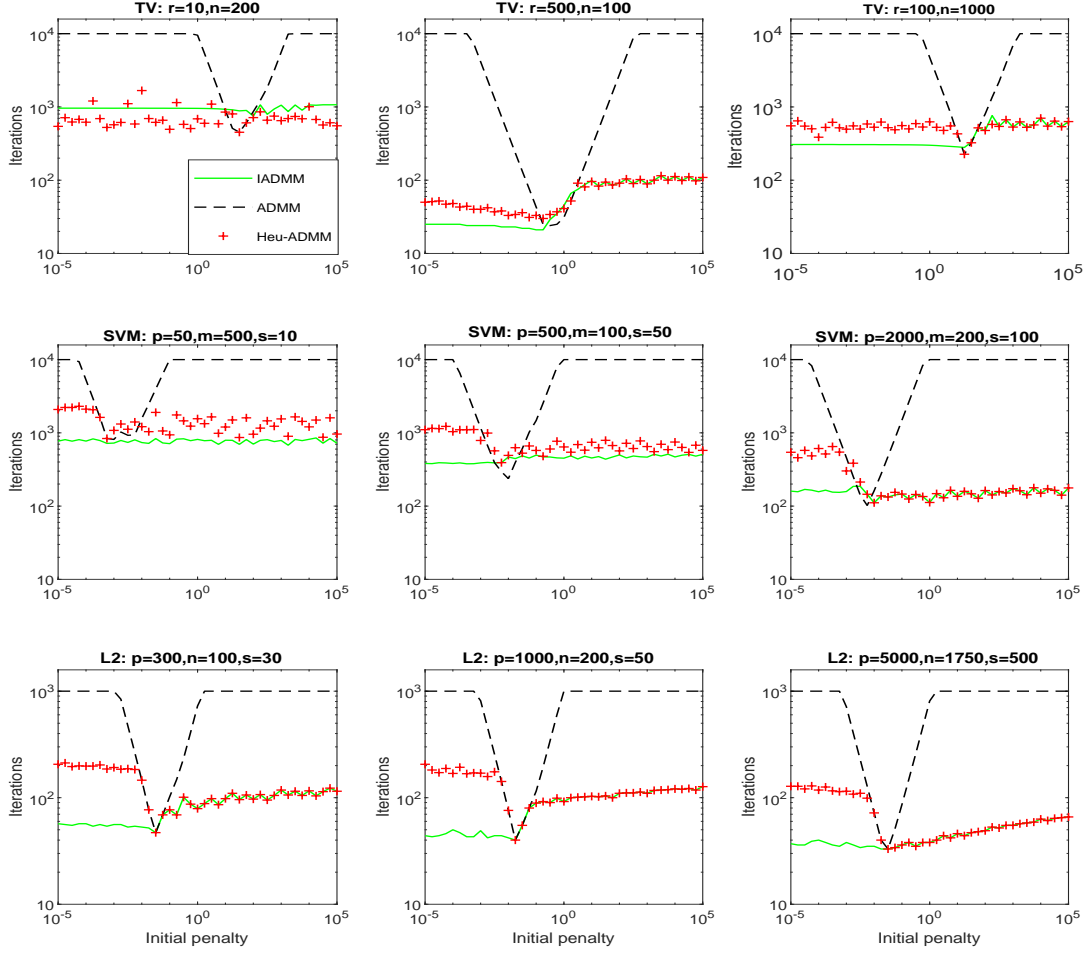


Figure 1: Results for Example 5.1, 5.2 and 5.3. “IADMM” is adaptive ADMM with strategy (34), “ADMM” is ADMM with a fixed-penalty parameter, “Heu-ADMM” is adaptive ADMM with strategy (39). “s” is a parameter for the generating data, readers may refer to [39, 44].

## 6.2 Verifying the convergence rate of IADMM

In this section, we consider the following dense convex quadratic program, which is considered in section 5.1 of [39].

**Example 6.4. Dense convex quadratic programs**

$$\min \left\{ \frac{1}{2} y^\top H y + h^\top y : y \in \mathbb{R}^m, a \leq B y \leq b \right\}, \quad (40)$$

where  $a, b \in \mathbb{R}^n$ ,  $h \in \mathbb{R}^m$ ,  $B \in \mathbb{R}^{n \times m}$  and  $H \in \mathbb{S}^m$  is a positive definite matrix. After introducing another variable  $z \in \mathbb{R}^n$ , (40) becomes:

$$\min \left\{ \frac{1}{2} y^\top H y + h^\top y + \delta_{[a,b]}(z) : B y - z = 0, y \in \mathbb{R}^m \right\}, \quad (41)$$

where  $\delta_{[a,b]}(\cdot)$  is the indicator function of the set  $\{x \in \mathbb{R}^n : a \leq x \leq b\}$ . Since  $\frac{1}{2} y^\top H y + h^\top y$  is strongly convex, we can use IADMM to solve (41). We generate data in the same way as section 5.1 of [39]. We choose  $n = m = 2000$  and the smallest eigenvalue of  $H$  is 1, 0.5, 0.1 for different choices of  $H$ . Moreover, we choose  $B$  as a matrix with full row rank. Therefore, from Theorem 3.9, IADMM has linear convergence as long as the penalty parameters are bounded.

We will compare IADMM with Tran-Dinh and Zhu’s accelerated ADMM in section 4.4 of [41], Kim’s accelerated ADMM in section 6.4 of [25] and Xu’s accelerated ADMM in [44]. We call them “acc1-ADMM”, “acc2-ADMM”, “acc3-ADMM”, respectively. For acc1-ADMM, we update the parameters as suggested in [41]<sup>2</sup>. For acc2-ADMM, we choose the initial penalty parameter to be  $25/(2\|B\|_2^2)$ , which is nearly optimally-tuned. For acc3-ADMM, we choose  $\beta_1 = \beta = 1/(2\|B\|_2^2)$  and use the strategy presented in (33) to update  $\beta_k$ . For IADMM, we choose  $\beta_1 = \beta = 1/(2\|B\|_2^2)$  and use the strategy in (34) to update  $\beta_k$ .

The results of this experiment are shown in Figure 2. The first column of plots is on the primal and dual infeasibility. We use the matlab function `semilogy` to visualize the linear convergence of IADMM. The second column of plots is on the KKT-residue. We use the matlab function `loglog` to show the sub-linear convergence rate of different types of ADMM algorithms. The third column of plots is on the penalty parameters. We use `loglog` to show the growth rate of the penalty parameters. From the plots of Figure 2, we can see that IADMM is the most efficient algorithm. The convergence rate of acc2-ADMM is close to  $O(1/n)$ , which is consistent with the theory in [25]. This example also implies that the convergence rate of  $O(1/n)$  for acc2-ADMM cannot be improved even if one part of the objective function is strongly convex. We also see that the convergence rate of acc1-ADMM is close to  $O(1/n^2)$ , which verifies the theory in [41]. For acc3-ADMM, its convergence rate is surprisingly much faster than the ergodic convergence rate of  $O(1/n^2)$  proved in Section 3. IADMM is the only algorithm which has linear convergence. This is consistent with Theorem 3.9 because from the third plot, the penalty parameter for IADMM is bounded. However, since the penalty parameter for acc3-ADMM tends to

<sup>2</sup>The parameters used in acc1-ADMM are quite different from the traditional ADMM, so we omit the details here. Readers may refer to [41] (31) case 2 and section 5.1 case 2 for details.

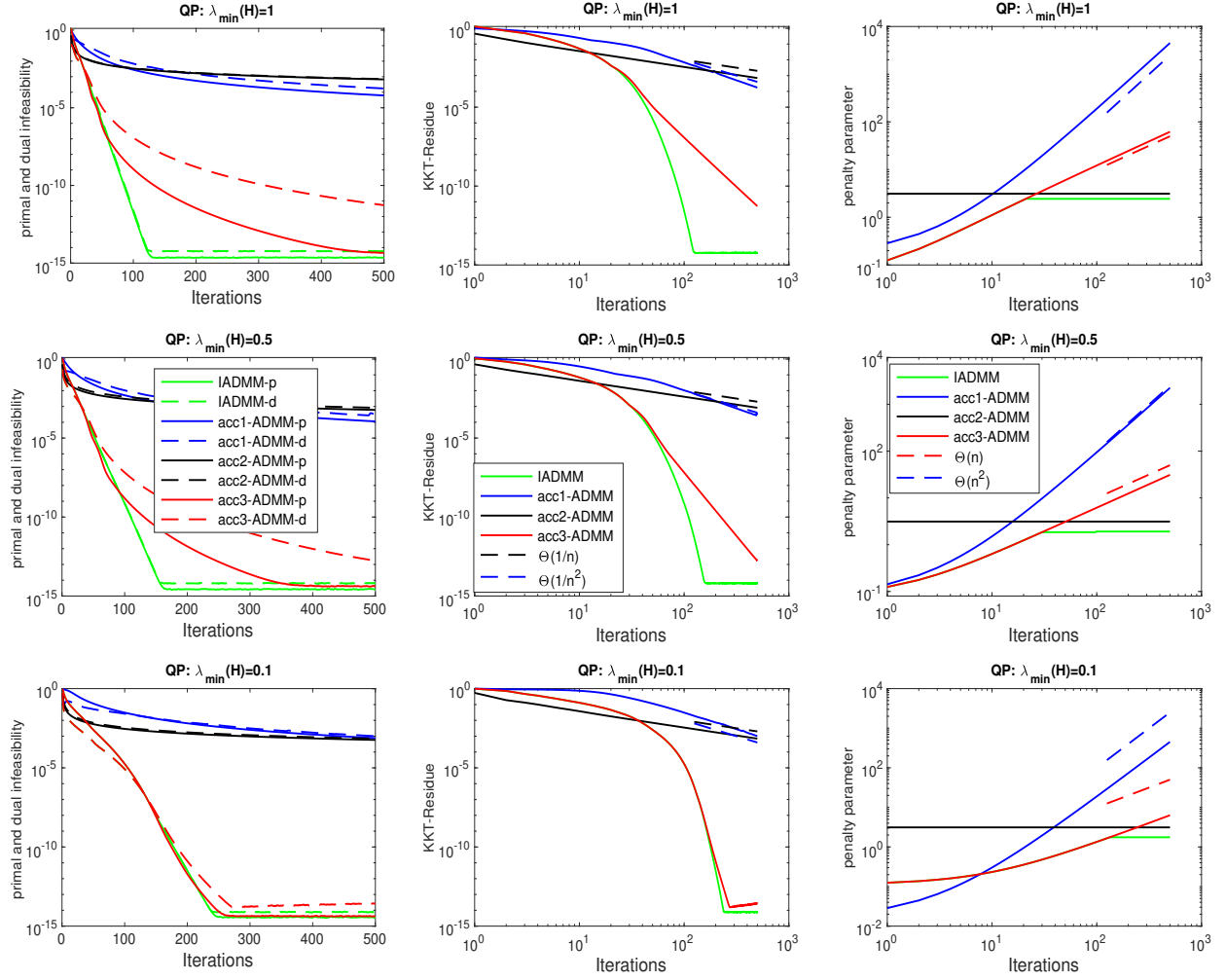


Figure 2: Results for the quadratic program (41) with different  $H$ . In the left panel, the curves \*\*\*-p and \*\*\*-d correspond to the primal feasibility and dual feasibility, respectively.

infinity, the condition of Theorem 3.9 is not satisfied and acc3-ADMM does not exhibit linear convergence. Moreover, it is easy to see that when the penalty parameter is too large, the primal feasibility of acc3-ADMM is much smaller than the dual feasibility of acc3-ADMM<sup>3</sup>. This is because the penalty on the dual feasibility becomes weaker with a larger  $\beta_k$ . That's why we had better choose the penalty parameter adaptively.

We have also tested (41) on other random instances, but the behaviours of acc1-ADMM, acc2-ADMM, acc3-ADMM and IADMM are similar to the ones presented in Figure 2.

### 6.3 Testing the efficiency of partial proximal point method for solving non-semi-strongly convex problems

In this section, we verify the efficiency of Algorithm 2. We consider the following two examples.

#### Example 6.5. Standard convex quadratic programming

$$\min \left\{ \frac{1}{2} x^\top Q x + c^\top x : Ax = b, l \leq x \leq u, x \in \mathbb{R}^n \right\}, \quad (42)$$

where  $Q \in \mathbb{S}_+^n$ ,  $c \in \mathbb{R}^n$ ,  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^m$ ,  $l, u \in (\mathbb{R} \cup \{\pm\infty\})^n$ .

Problem (43) is the standard form of a convex quadratic programming problem. We reformulate problem (43) into the following problem:

$$\min \left\{ \frac{1}{2} z^\top Q z + c^\top z + \delta_{[l,u]}(z) + \delta_\Omega(y) : y = z, y, z \in \mathbb{R}^n \right\}, \quad (43)$$

where  $\Omega = \{y \in \mathbb{R}^n \mid Ay = b\}$ . The above problem can be solved by ADMM-type algorithms. Because the matrix  $Q$  may not be positive definite, we use the partial proximal point method to solve it. For the IADMM subproblem with respect to  $z$ , we use the semi-smooth Newton method to solve its dual problem. In detail, after some simplification, we subproblem with respect to  $z$  can be written as

$$\min \left\{ z^\top Q z / 2 + \beta \|z - \hat{z}\|^2 / 2 + \delta_{[l,u]}(z) : z \in \mathbb{R}^n \right\}, \quad (44)$$

for some  $\beta > 0$  and  $\hat{z} \in \mathbb{R}^n$ . Without loss of generality, we assume that  $Q$  is positive definite. This is because otherwise we can reformulate the above problem as

$$\min \left\{ z^\top (Q + \epsilon I) z / 2 + (\beta - \epsilon) \|z - \beta \hat{z} / (\beta - \epsilon)\|^2 / 2 + \delta_{[l,u]}(z) : z \in \mathbb{R}^n \right\} \quad (45)$$

for some small constant  $\epsilon > 0$ . Problem (44) and (45) only differ by a constant. Since  $Q$  is positive definite, we can apply Cholesky factorization to decompose it as  $Q := LL^\top$  for some lower triangular matrix  $L$ . Note that we only have to do this once at the beginning because  $Q$  is a constant matrix. Thus, problem (44) can be further written as

$$\min \left\{ \|L^\top z\|^2 / 2 + \beta \|z - \hat{z}\|^2 / 2 + \delta_{[l,u]}(z) : z \in \mathbb{R}^n \right\}. \quad (46)$$

---

<sup>3</sup>Note that the acc1-ADMM is quite different from the traditional ADMM. Its primal and dual feasibility is close to each other even if its penalty parameter increases rapidly.

After introducing another variable  $y = L^\top z$ , problem (46) becomes

$$\min \left\{ \|y\|^2/2 + \beta \|z - \hat{z}\|^2/2 + \delta_{[l,u]}(z) : y = L^\top z, y \in \mathbb{R}^n, z \in \mathbb{R}^n \right\}, \quad (47)$$

whose dual problem is

$$\min \left\{ \beta \left( \|\hat{z} - L\lambda/\beta\|^2 - \text{dist}([l, u], \hat{z} - L\lambda/\beta)^2 \right) / 2 + \|\lambda\|^2/2 : \lambda \in \mathbb{R}^n \right\}, \quad (48)$$

which is an unconstrained optimization problem with the objective function being strongly convex and Lipschitz continuous differentiable. One can use semi-smooth newton method to solve (48) efficiently. In our experiment, we set the tolerance of the gradient norm to be less than  $10^{-10}$  to terminate the semi-smooth Newton method.

We compare Algorithm 2 with the barrier method of Gurobi [18] 9.5.2 with 2 threads on the modified Maros-Mészáros benchmark dataset [32]. Because the matrix  $Q$  of some of the instances in [32] is diagonal, Gurobi can solve such problems very efficiently by making use of this highly special structure. In order to make the problems more challenging, we add a small perturbation to  $Q$  whenever it is diagonal as follows:

$$B = \text{randn}(n, n); \text{BB} = B * B'; Q = Q + 1e-6 * \text{norm}(Q, 'fro') * \text{BB} / \text{norm}(\text{BB}, 'fro').$$

We use “\*” to indicate the perturbed instances in the following table. After the random perturbation,  $Q$  becomes a dense matrix. We only test the problems in [32] with standard form and  $1000 \leq n \leq 10000$ . When  $n$  is greater than 10000, the dense Cholesky factorization is too expensive for both Algorithm 2 and Gurobi. We use the following standard KKT residue for convex QP problem to measure the accuracy:

$$\text{Resp} := \frac{\|Ax - b\|}{1 + \|b\|}, \text{Resd} := \frac{\|\text{Proj}_{[l,u]}(x - (Qx + c - A^\top \lambda)) - x\|}{1 + \|Qx + c\|} \quad (49)$$

Since Gurobi use a different stopping criterion instead of KKT residue, we first use Gurobi to solve one problem with tolerance  $10^{-5}$  to get the outcome. Then we compute the KKT residue i.e.,  $\max\{\text{Resp}, \text{Resd}\}$  of the outcome, say  $\alpha$ , and set the tolerance of PPPM to be  $\min\{10^{-5}, \alpha\}$ . By doing so, the outcome of our algorithm will be more accurate than Gurobi and the comparison of running time will be fair.

Table 1: Comparison of PPPM(IADMM), and Gurobi for modified Maros-Mészáros convex QP problem;  $(n, m)$  denotes the number of variables and affine constraints.

Problem	Algorithm	Resp	Resd	Fval	Time [s]
AUG3DCQP* (3873,1000)	PPPM	9.61e-06	1.81e-06	-9.4314271e+02	4.06e+00
	Gurobi	4.64e-11	4.64e-04	-9.4313492e+02	9.93e+01
AUG3DQP* (3873,1000)	PPPM	9.38e-06	4.06e-06	-6.6126864e+02	1.44e+01
	Gurobi	1.82e-14	1.31e-03	-6.6125924e+02	9.95e+01
CONT-050* (2597,2401)	PPPM	7.79e-07	8.39e-07	-4.5638482e+00	2.77e+02
	Gurobi	2.45e-12	8.51e-07	-4.5638481e+00	2.41e+01
CVXQP1.L (10000,5000)	PPPM	4.58e-09	6.78e-09	1.0870480e+08	4.71e+00
	Gurobi	7.37e-09	4.40e-09	1.0870481e+08	1.56e+01

Table 1: Comparison of PPPM(IADMM), and Gurobi for modified Maros-Mészáros convex QP problem;  $(n, m)$  denotes the number of variables and affine constraints.

Problem	Algorithm	Resp	Resd	Fval	Time [s]
CVXQP1.M (1000,500)	PPPM	1.95e-08	6.07e-08	1.0875116e+06	1.51e-01
	Gurobi	6.90e-08	2.90e-08	1.0875127e+06	1.88e-01
CVXQP2.L (10000,2500)	PPPM	1.53e-09	8.70e-09	8.1842458e+07	5.50e+00
	Gurobi	2.85e-12	8.90e-09	8.1842458e+07	7.12e+00
CVXQP2.M (1000,250)	PPPM	2.42e-08	1.22e-07	8.2015543e+05	2.16e-01
	Gurobi	1.85e-10	1.24e-07	8.2015543e+05	1.01e-01
CVXQP3.L (10000,7500)	PPPM	9.90e-10	8.38e-09	1.1571110e+08	6.25e+00
	Gurobi	8.45e-09	5.54e-09	1.1571112e+08	2.03e+01
CVXQP3.M (1000,750)	PPPM	7.54e-08	2.04e-07	1.3628288e+06	2.19e+00
	Gurobi	2.18e-07	4.41e-08	1.3628301e+06	2.60e-01
HUES-MOD* (10000,2)	PPPM	8.98e-06	2.81e-06	3.4824562e+07	3.01e+01
	Gurobi	4.00e-15	2.59e-05	3.4824489e+07	1.27e+03
HUESTIS* (10000,2)	PPPM	4.38e-08	1.38e-08	3.4824489e+11	3.34e+01
	Gurobi	3.66e-15	5.53e-08	3.4824489e+11	1.74e+03
QSCSD6 (1350,147)	PPPM	9.95e-06	8.17e-06	5.0808222e+01	1.00e+00
	Gurobi	2.99e-13	1.08e-05	5.0808253e+01	1.48e-02
QSCSD8 (2750,397)	PPPM	9.38e-06	6.29e-06	9.4076384e+02	1.11e+00
	Gurobi	2.68e-14	3.30e-05	9.4076594e+02	2.40e-02
STCQP1 (4097,939)	PPPM	7.14e-07	1.59e-06	1.5514388e+05	1.76e+00
	Gurobi	3.70e-11	1.60e-06	1.5514356e+05	1.73e-01
STCQP2 (4097,2052)	PPPM	1.11e-06	9.89e-06	2.2327448e+04	6.63e-01
	Gurobi	3.57e-13	1.69e-05	2.2327319e+04	3.27e-01

From Table 1, we can see that Algorithm 2 can solve all the instances to the required accuracy. Gurobi does not reach the tolerance for some instances because it uses a different stopping criterion. Algorithm 2 is faster than Gurobi for more than half of the instances. This shows that when  $Q$  is not well structured, our algorithm (PPPM with IADMM as subproblem solver) is efficient enough to compete favourably with Gurobi, which is a well-developed solver for convex QP problems.

Next we consider another class of non-semi-strongly convex problems, rank lasso problems, to evaluate the efficiency of IADMM within the PPPM framework.

**Example 6.6. Rank LASSO problem**

$$\min \left\{ \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} \left| (b_i - a_i^\top x) - (b_j - a_j^\top x) \right| + \lambda \|x\|_1 : x \in \mathbb{R}^m \right\}. \quad (50)$$

Problem (50) was proposed by Wang et al. in [42]. Its computational aspect is recently studied in [1, 38]. Let  $A := (a_1, a_2, \dots, a_n)^\top$  and  $b = (b_1, b_2, \dots, b_m)^\top$ . Problem (50) is also called tuning-free robust regression because its regularization parameter has the following formula (see (7) of [42]).

$$\lambda = cG_{\|S_n\|_\infty}^{-1} (1 - \alpha_0), \quad (51)$$

where  $\alpha_0 = 0.1$ ,  $c = 1.1$  and  $G_{\|S_n\|_\infty}^{-1}(1-\alpha_0)$  denotes the  $(1-\alpha_0)$ -quantile of the distribution of  $\|S_n\|_\infty$ . Here  $S_n = -2A^\top \xi / n(n-1)$  and  $\xi = 2r - (n+1)$  with  $r$  following the uniform distribution on the permutations of the integers  $\{1, 2, \dots, n\}$ . Problem (50) can be written as the following problem:

$$\min \{h(y) + \lambda \|z\|_1 : Az - b = y, z \in \mathbb{R}^m, y \in \mathbb{R}^n\}, \quad (52)$$

where  $h(y) := \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} |y_i - y_j|$ . Because problem (52) is not semi-strongly convex, we use the partial proximal point method to solve it. When we use IADMM to solve the proximal subproblem of (52), updating  $z$  corresponds to solving the following problem:

$$\min \left\{ \lambda \|z\|_1 + \beta \|Az - b - y - \lambda/\beta\|^2/2 + \sigma \|z - z^k\|^2/2 : z \in \mathbb{R}^m \right\}, \quad (53)$$

which is similar to the proximal point subproblem of the square LASSO problem. We can use the semi-smooth Newton method [28] to solve its dual problem efficiently. Updating  $y$  can be done easily by computing the proximal mapping of  $h(y)$ . We can use the direct solver developed in [30] to solve it in nearly linear time. Because updating the variable  $z$  doesn't have a closed form solution for a general matrix  $A$ , we compare Algorithm 2 two variants of ADMM's. The first one is linearized ADMM, which is presented in Algorithm 4.

---

#### Algorithm 4 LADMM

---

**Initialization:** Given  $(x^1, \lambda^1)$  and constants  $\gamma \in (1, \frac{1+\sqrt{5}}{2})$ .

**for**  $k = 1, 2, \dots$  **do**

1.  $y^{k+1} = \arg \min_y \left\{ h(y) + \frac{\beta_k}{2} \|Az^k - y - b - \lambda^k/\beta_k\|^2 \right\}$
2.  $z^{k+1} = \arg \min_z \left\{ \lambda \|z\|_1 + \frac{\beta_k}{2} \|Az - y^{k+1} - b + \lambda^k/\beta_k\|^2 + \frac{\beta_k}{2} \|z - z^k\|_{\lambda_{\max}(A^\top A)I - A^\top A}^2 \right\}$
3.  $\lambda^{k+1} = \lambda^k - \gamma \beta_k (A^\top z^{k+1} - y^{k+1} - b)$
4. Update  $\beta_{k+1}$  from (39)

**end for**

---

In Step 2 of Algorithm 4, the proximal term with the weighted matrix  $\beta_k(\lambda_{\max}(A^\top A)I - A^\top A)$  can simplify the subproblem into computing the proximal mapping of  $\lambda \|\cdot\|_1$ , which has a closed form solution. The second variant is that we introduce another variable  $u \in \mathbb{R}^m$  and write problem (52) equivalently as follows:

$$\min \{h(y) + \lambda \|z\|_1 : Au - b = y, u = z, z \in \mathbb{R}^m, y \in \mathbb{R}^n, u \in \mathbb{R}^m\}. \quad (54)$$

We can use the ADMM presented in Algorithm 5 to solve (54).

In Algorithm 5, Step 1 is essentially solving a positive definite linear system with the coefficient matrix  $I_m + A^\top A$ . Note that we can apply the Sherman-Morrison-Woodbury formula [17] when  $n$  is smaller than  $m$ . We also store the Cholesky decomposition of  $I + AA^\top$  to use it to solve the linear system in Step 1. Step 2 is equivalent to computing the proximal mappings of  $h(y)$  and  $\lambda \|z\|_1$  independently. Apart from ADMM, we can also formulate problem (50) as a linear programming problem as mentioned in section 4.2 of [42]. However,

---

**Algorithm 5** ADMM

---

**Initialization:** Given  $(y^1, z^1, u^1, \lambda^1, \mu^1)$  and constants  $\gamma \in (1, \frac{1+\sqrt{5}}{2})$ .

**for**  $k = 1, 2, \dots$  **do**

1.  $u^{k+1} = \arg \min_y \left\{ \frac{\beta_k}{2} \|Au - b - y^k - \lambda^k / \beta_k\|^2 + \frac{\beta_k}{2} \|u - z^k - \mu^k / \beta_k\|^2 \right\}$
- 2.

$$(y^{k+1}, z^{k+1}) = \arg \min_{(y, z)} \left\{ \begin{array}{l} h(y) + \frac{\beta_k}{2} \|Au^{k+1} - b - y - \lambda^k / \beta_k\|^2 \\ + \lambda \|z\|_1 + \frac{\beta_k}{2} \|u^{k+1} - z - \mu^k / \beta_k\|^2 \end{array} \right\}$$

$$3. \lambda^{k+1} = \lambda^k - \gamma \beta_k (A^\top u^{k+1} - y^{k+1} - b), \mu^{k+1} = \mu^k - \gamma \beta_k (u^{k+1} - z^{k+1})$$

4. Update  $\beta_{k+1}$  from (39)

**end for**

---

the linear programming formulation has huge number of variables and constraints. Thus we only test the linear programming model for small problems.

For ADMM solvers, we terminate the algorithms when the KKT residue is smaller than  $10^{-5}$ . We choose the initial penalty parameter to be  $\max\{100/\lambda_{\max}(B)^2, 10^{-6}\}$ . We set the maximum running time to be 3600s. For the linear programming model, we use the barrier method in Gurobi 9.5.2 with 2 threads. We also set the tolerance to be  $10^{-5}$  and the maximum running time to be 3600s;

We first consider synthetic dataset. We generated the data  $A$  and  $b$  in the same way as mentioned in example 1 of [42]. In detail, the rows of  $A$  are generated from a  $p$ -dimensional multivariable normal distribution  $N_p(0, \Sigma)$ , where the covariance matrix satisfies  $\Sigma_{i,j} = 0.5$  for  $i \neq j$  and  $\Sigma_{i,j} = 1$  for  $i = j$ . The ground truth  $\hat{x} := (\sqrt{3}, \sqrt{3}, \sqrt{3}, 0, \dots, 0)^\top$ . We generate  $b$  as  $b = A\hat{x} + \epsilon$ , where  $\epsilon_i$  satisfies six different distributions: (1)  $N(0, 0.25)$ ; (2)  $N(0, 1)$ ; (3)  $N(0, 2)$ ; (4)  $0.95N(0, 1) + 0.05N(0, 100)$  (denoted by MN); (5)  $\sqrt{2}t(4)$ , where  $t(4)$  denotes the  $t$  distribution with 4 degree of freedom; (6) Cauchy(0,1). We choose  $\lambda$  from the formula (51) by randomly generating 1000 permutations of the integers  $\{1, 2, \dots, n\}$  and computing the approximate  $(1 - \alpha_0)$ -quantile.

We first test small problems where  $n = 200$  and  $m = 1000$ . In this case, we may compared the above mentioned four algorithms.

Table 2: Comparison of PPPM(IADMM), LADMM, ADMM and Gurobi for randomly generated rank LASSO problem.  $n = 200$ ,  $m = 1000$ . For Algorithm 2, “Iter” means number of ADMM iterations.

Problem	Algorithm	Resp	Resd	Iter	Fval	Time [s]
n = 200	PPPM	4.18e-06	7.67e-06	80	1.9747008e+00	1.60e-01
m = 1000	LADMM	9.92e-06	9.79e-06	3357	1.9747001e+00	2.07e+00
$\kappa = 0.32$	ADMM	8.35e-06	1.00e-05	4206	1.9747004e+00	3.13e+00
$N(0, 0.25)$	Gurobi	9.99e-19	4.74e-06	28	1.9747009e+00	6.13e+01
n = 200	PPPM	5.84e-06	9.98e-06	138	2.8465270e+00	1.69e-01
m = 1000	LADMM	9.17e-06	9.99e-06	3560	2.8465223e+00	1.96e+00
$\kappa = 0.32$	ADMM	9.95e-06	9.27e-06	178321	2.8465172e+00	1.31e+02
$N(0, 1)$	Gurobi	6.30e-18	1.39e-06	22	2.8465242e+00	4.99e+01



Table 2: Comparison of PPPM(IADMM), LADMM, ADMM and Gurobi for randomly generated rank LASSO problem.  $n = 200$ ,  $m = 1000$ . For Algorithm 2, “Iter” means number of ADMM iterations.

Problem	Algorithm	Resp	Resd	Iter	Fval	Time [s]
$n = 200$	PPPM	7.45e-06	9.46e-06	154	4.0089871e+00	1.87e-01
$m = 1000$	LADMM	9.98e-06	1.00e-05	6154	4.0089518e+00	3.60e+00
$\kappa = 0.32$	ADMM	9.72e-06	9.93e-06	22269	4.0089440e+00	1.62e+01
$N(0, 2)$	Gurobi	1.32e-17	2.36e-06	17	4.0089541e+00	4.00e+01
$n = 200$	PPPM	7.19e-06	9.92e-06	195	6.8206437e+00	1.51e+00
$m = 1000$	LADMM	9.61e-06	9.95e-06	6743	6.8203775e+00	3.51e+00
$\kappa = 0.32$	ADMM	9.81e-06	3.59e-06	18699	6.8203800e+00	1.32e+01
MN	Gurobi	2.38e-17	5.35e-07	18	6.8203778e+00	4.45e+01
$n = 200$	PPPM	1.50e-06	9.90e-06	152	3.8513292e+00	1.32e-01
$m = 1000$	LADMM	9.88e-06	9.96e-06	3918	3.8513145e+00	2.19e+00
$\kappa = 0.32$	ADMM	9.99e-06	7.73e-06	28135	3.8513183e+00	2.01e+01
$\sqrt{2}t_4$	Gurobi	6.43e-18	1.16e-05	26	3.8513196e+00	5.75e+01
$n = 200$	PPPM	4.27e-06	7.18e-06	73	9.5835201e+00	8.57e-02
$m = 1000$	LADMM	8.52e-06	9.99e-06	4190	9.5833821e+00	2.41e+00
$\kappa = 0.32$	ADMM	9.98e-06	8.92e-06	11156	9.5834183e+00	8.04e+00
Cauchy	Gurobi	6.95e-18	1.31e-07	38	9.5833823e+00	8.37e+01

From Table 2, we can see that all the algorithms can solve the small size problems to the required accuracy. Gurobi is slow compared with ADMM-type methods except for the second instance. This is because the linear programming formulation of problem (50) contains too many constraints and variables. Among the ADMM-type algorithms, the partial proximal point method (with IADMM as its subproblems solver) is more than 10 times faster than the other two algorithms. The main reason is that with the proximal term, we can use the powerful semi-smooth Newton method to solve the subproblem (53) to update the variable  $z$ .

Now, we move on to test some large randomly generated problems. We do not consider Gurobi because it suffers from memory issue for large problems. For convenience, we only consider the first type of noise, i.e.,  $N(0, 0.25)$  when we generate the dataset.

Table 3: Comparison of PPPM(IADMM) and LADMM and ADMM for randomly generated rank LASSO problem.

Problem	Algorithm	Resp	Resd	Iter	Fval	Time [s]
$n = 1000$	PPPM	4.37e-06	1.53e-06	51	1.0338039e+00	6.75e-01
$m = 2000$	LADMM	6.05e-06	1.00e-05	2193	1.0338041e+00	8.70e+00
$\kappa = 0.15$	ADMM	5.52e-06	9.99e-06	4350	1.0338050e+00	2.70e+01
$n = 1000$	PPPM	5.83e-06	5.73e-06	53	1.0627181e+00	1.45e+00
$m = 4000$	LADMM	5.02e-06	9.99e-06	2606	1.0627179e+00	3.03e+01
$\kappa = 0.15$	ADMM	4.56e-06	1.00e-05	6802	1.0627190e+00	9.94e+01
$n = 2000$	PPPM	4.30e-06	6.66e-06	52	8.3477708e-01	3.91e+00
$m = 4000$	LADMM	9.98e-06	6.53e-06	3469	8.3477689e-01	1.73e+02
$\kappa = 0.11$	ADMM	9.77e-06	9.46e-06	7317	8.3477629e-01	4.23e+02
$n = 2000$	PPPM	3.48e-06	4.34e-06	60	8.5790776e-01	9.32e+00
$m = 8000$	LADMM	9.94e-06	9.10e-06	3750	8.5790787e-01	3.75e+02

Table 3: Comparison of PPPM(IADMM) and LADMM and ADMM for randomly generated rank LASSO problem.

Problem	Algorithm	Resp	Resd	Iter	Fval	Time [s]
$\kappa = 0.11$	ADMM	9.88e-06	9.63e-06	31187	8.5791123e-01	3.40e+03
$n = 4000$	PPPM	1.88e-06	6.87e-06	56	6.8255971e-01	1.94e+01
$m = 8000$	LADMM	9.95e-06	6.99e-06	4465	6.8255971e-01	8.12e+02
$\kappa = 0.08$	ADMM	2.21e-05	7.79e-06	17604	6.8255852e-01	3.60e+03
$n = 4000$	PPPM	2.57e-06	6.28e-06	58	6.9040873e-01	2.21e+01
$m = 10000$	LADMM	9.93e-06	7.22e-06	4078	6.9040871e-01	9.08e+02
$\kappa = 0.08$	ADMM	3.32e-05	1.26e-05	14239	6.9040667e-01	3.60e+03

From Table 3, we can see that the Algorithm 2 is readily more than 10 times faster than the other two algorithms. For the randomly generated datasets, the LADMM and ADMM can return a solution of moderate accuracy for all of the instances. This is because random problems are usually well-conditioned. Apart from synthetic data, we also test on some real data instances, which are collected from LIBSVM datasets [6]. We expand the features of the original data using the polynomial basis functions mentioned in [23]. Different from randomly generated problems, the coefficient matrices  $A$  for real datasets are usually ill-conditioned (with condition numbers ranging from the order of  $10^3$  to the order of more than  $10^{12}$ ) and problem (19) is difficult to be solved by traditional first order method. We don't show the results of certain algorithm if it reaches the maximum running time but the solution is very inaccurate.

Table 4: Comparison of PPPM(IADMM) and LADMM and ADMM for real datasets.  $(n, m)$  denotes the sample size and expanded feature size.

Problem	Algorithm	Resp	Resd	Iter	Fval	Time [s]
abalone7 (4177,6435)	PPPM	1.20e-06	8.53e-06	59	2.6709065e+00	7.87e+00
$\kappa = 0.03$	LADMM	9.97e-06	8.17e-06	3706	2.6596874e+00	5.77e+02
	ADMM	6.06e-06	9.95e-06	10843	2.6597018e+00	1.98e+03
bodyfat7 (252,116280)	PPPM	1.88e-06	8.48e-06	64	4.3262740e-03	6.31e+00
$\kappa = 0.06$	LADMM	-	-	-	-	-
	ADMM	-	-	-	-	-
E2006.test (3308,150358)	PPPM	7.74e-06	5.76e-06	62	3.6089303e-01	6.50e+00
$\kappa = 0.02$	LADMM	9.99e-06	1.89e-06	655	3.5726244e-01	1.15e+02
	ADMM	1.46e-06	9.87e-06	2537	3.5701485e-01	4.76e+02
housing7 (506,77520)	PPPM	1.47e-06	6.96e-06	66	7.2903775e+00	1.20e+01
$\kappa = 0.10$	LADMM	1.76e-05	3.32e-05	16685	7.2707283e+00	3.60e+03
	ADMM	1.43e-04	3.06e-05	15777	7.2704069e+00	3.60e+03
mpg7 (392,3432)	PPPM	1.46e-06	5.35e-06	69	5.0256912e+00	2.16e-01
$\kappa = 0.10$	LADMM	9.15e-06	6.95e-06	1472	5.0191357e+00	3.07e+00
	ADMM	9.88e-06	5.25e-06	7906	5.0191178e+00	2.14e+01
pyrim5 (74,201376)	PPPM	7.32e-06	9.29e-06	93	1.3603816e-01	5.79e+00
$\kappa = 0.37$	LADMM	4.43e-05	1.82e-04	38099	1.3603915e-01	3.60e+03
	ADMM	6.88e-04	1.78e-03	36569	1.3910683e-01	3.60e+03
space_ga9 (3107,5005)	PPPM	3.70e-06	8.81e-06	53	1.6569455e-01	3.16e+00
$\kappa = 0.02$	LADMM	4.99e-06	9.93e-06	281	1.6569433e-01	2.74e+01
	ADMM	-	-	-	-	-

Table 4: Comparison of PPPM(IADMM) and LADMM and ADMM for real datasets.  $(n, m)$  denotes the sample size and expanded feature size.

Problem	Algorithm	Resp	Resd	Iter	Fval	Time [s]
triazines4 (186,635376) $\kappa = 0.29$	PPPM	7.41e-06	2.67e-06	95	1.6628905e-01	3.08e+01
	LADMM	2.67e-05	9.53e-05	6381	1.6632803e-01	3.60e+03
	ADMM	-	-	-	-	-
E2006.train (16087,150360) $\kappa = 0.01$	PPPM	7.96e-06	9.66e-06	85	3.7853164e-01	3.76e+01
	LADMM	-	-	-	-	-
	ADMM	8.19e-06	9.85e-06	353	3.7842288e-01	3.52e+02
log1p.E2006.test (3308,4272226) $\kappa = 0.10$	PPPM	9.98e-06	4.28e-06	69	4.3580711e-01	6.68e+01
	LADMM	8.32e-04	4.17e-03	3027	4.1710211e-01	3.60e+03
	ADMM	-	-	-	-	-
log1p.E2006.train (16087,4272227) $\kappa = 0.05$	PPPM	8.92e-06	1.72e-06	71	4.1277825e-01	2.40e+02
	LADMM	-	-	-	-	-
	ADMM	-	-	-	-	-

From Table 4, we can see that Algorithm 2 is the only solver that can solve all the problems to the required accuracy. Moreover, Algorithm 2 is much more efficient than the other two algorithms. For the instances `housing7` and `pyrim5`, Algorithm 2 is more than 100 times faster than the other two algorithms. This implies that our algorithm is less affected by ill-conditioning of the dataset. This verifies the efficiency and robustness of the partial proximal point method (with IADMM as its subproblems solver) for solving non-semi-strongly convex problems of the form (1).

## 7 Conclusion

We have proposed an adaptive ADMM which can adjust the penalty parameters adaptively with a large degree of freedom. Various types of convergence results for IADMM have been established under the semi-strongly convex condition. We have also proposed a partial proximal point method (together with IADMM as its subproblems solver) to solve problems without semi-strongly convexity. Numerical experiments show that the convergence of IADMM with self-adaptive parameters adjustment is insensitive to the initial parameter chosen as compared to the fixed-parameter ADMM. Also, the partial proximal point method is much more efficient compared with other ADMM-type methods. There are further research questions that we can explore, and these include analyzing the convergence rate of partial proximal point method and applying this method to solve other problems with two non-smooth functions.

## Acknowledgement

We thank the reviewers and Associate Editor for many helpful suggestions to improve the quality of the paper.

## References

- [1] X. Bai and Q. Li. A highly efficient adaptive-sieving-based algorithm for the high-dimensional rank lasso problem. *arXiv preprint arXiv:2207.12753*, 2022.
- [2] H. H. Bauschke, P. L. Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.
- [3] D. Boley. Local linear convergence of the alternating direction method of multipliers on quadratic or linear programs. *SIAM Journal on Optimization*, 23(4):2183–2207, 2013.
- [4] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40:120–145, 2011.
- [5] A. Chambolle and T. Pock. On the ergodic convergence rates of a first-order primal-dual algorithm. *Mathematical Programming*, 159(1-2):253–287, 2016.
- [6] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.
- [7] L. Chen, D. Sun, and K.-C. Toh. A note on the convergence of admm for linearly constrained convex optimization problems. *Computational Optimization and Applications*, 66:327–343, 2017.
- [8] L. Condat. A direct algorithm for 1-d total variation denoising. *IEEE Signal Processing Letters*, 20(11):1054–1057, 2013.
- [9] Y. Cui, D. Sun, and K.-C. Toh. On the r-superlinear convergence of the kkt residuals generated by the augmented lagrangian method for convex composite conic programming. *Mathematical Programming*, 178:381–415, 2019.
- [10] W. Deng and W. Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, 66:889–916, 2016.
- [11] J. Eckstein and P. J. Silva. A practical relative error criterion for augmented lagrangians. *Mathematical Programming*, 141(1-2):319–348, 2013.
- [12] M. Fazel, T. K. Pong, D. Sun, and P. Tseng. Hankel matrix rank minimization with applications to system identification and realization. *SIAM Journal on Matrix Analysis and Applications*, 34(3):946–977, 2013.
- [13] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & mathematics with applications*, 2(1):17–40, 1976.
- [14] P. Giselsson and S. Boyd. Linear convergence and metric selection for douglas-rachford splitting and admm. *IEEE Transactions on Automatic Control*, 62(2):532–544, 2016.

- [15] R. Glowinski and A. Marroco. Sur l’approximation, par éléments finis d’ordre un, et la résolution, par pénalisation-dualité d’une classe de problèmes de dirichlet non linéaires. *Revue française d’automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(R2):41–76, 1975.
- [16] T. Goldstein, B. O’Donoghue, S. Setzer, and R. Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014.
- [17] G. H. Golub et al. Cf vanloan, matrix computations. *The Johns Hopkins*, 113(10):23–36, 1996.
- [18] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.
- [19] C. D. Ha. A generalization of the proximal point algorithm. *SIAM Journal on Control and Optimization*, 28(3):503–512, 1990.
- [20] B. He, H. Yang, and S. Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and applications*, 106:337–356, 2000.
- [21] B. He and X. Yuan. On the  $o(1/n)$  convergence rate of the douglas–rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.
- [22] M. Hong and Z.-Q. Luo. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, 162(1-2):165–199, 2017.
- [23] L. Huang, J. Jia, B. Yu, B.-G. Chun, P. Maniatis, and M. Naik. Predicting execution time of computer programs using sparse polynomial regression. *Advances in neural information processing systems*, 23, 2010.
- [24] K. Jiang, D. Sun, and K.-C. Toh. Solving nuclear norm regularized and semidefinite matrix least squares problems with linear equality constraints. *Discrete Geometry and Optimization*, pages 133–162, 2013.
- [25] D. Kim. Accelerated proximal point method for maximally monotone operators. *Mathematical Programming*, 190(1-2):57–87, 2021.
- [26] H. Li and Z. Lin. Accelerated alternating direction method of multipliers: An optimal  $o(1/k)$  nonergodic analysis. *Journal of Scientific Computing*, 79:671–699, 2019.
- [27] M. Li, D. Sun, and K.-C. Toh. A majorized admm with indefinite proximal terms for linearly constrained convex composite optimization. *SIAM Journal on Optimization*, 26(2):922–950, 2016.
- [28] X. Li, D. Sun, and K.-C. Toh. A highly efficient semismooth newton augmented lagrangian method for solving lasso problems. *SIAM Journal on Optimization*, 28(1):433–458, 2018.
- [29] L. Liang, D. Sun, and K.-C. Toh. An inexact augmented lagrangian method for second-order cone programming with applications. *SIAM Journal on Optimization*, 31(3):1748–1773, 2021.

- [30] M. Lin, Y.-J. Liu, D. Sun, and K.-C. Toh. Efficient sparse semismooth newton methods for the clustered lasso problem. *SIAM Journal on Optimization*, 29(3):2026–2052, 2019.
- [31] Z. Lin, R. Liu, and Z. Su. Linearized alternating direction method with adaptive penalty for low-rank representation. *Advances in neural information processing systems*, 24, 2011.
- [32] I. Maros and C. Mészáros. A repository of convex quadratic programming problems. *Optimization Methods and Software*, 11(1-4):671–681, 1999.
- [33] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. Jordan. A general analysis of the convergence of admm. In *International conference on machine learning*, pages 343–352. PMLR, 2015.
- [34] Y. Ouyang, Y. Chen, G. Lan, and E. Pasiliao Jr. An accelerated linearized alternating direction method of multipliers. *SIAM Journal on Imaging Sciences*, 8(1):644–681, 2015.
- [35] R. T. Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of operations research*, 1(2):97–116, 1976.
- [36] R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.
- [37] S. Sabach and M. Teboulle. Faster lagrangian-based methods in convex optimization. *SIAM Journal on Optimization*, 32(1):204–227, 2022.
- [38] P. Tang, C. Wang, and B. Jiang. A proximal-proximal majorization-minimization algorithm for nonconvex tuning-free robust regression problems. *arXiv preprint arXiv:2106.13683*, 2021.
- [39] Q. Tran-Dinh. Proximal alternating penalty algorithms for nonsmooth constrained convex optimization. *Computational Optimization and Applications*, 72:1–43, 2019.
- [40] Q. Tran-Dinh, O. Fercoq, and V. Cevher. A smooth primal-dual optimization framework for nonsmooth composite convex minimization. *SIAM Journal on Optimization*, 28(1):96–134, 2018.
- [41] Q. Tran-Dinh and Y. Zhu. Non-stationary first-order primal-dual algorithms with faster convergence rates. *SIAM Journal on Optimization*, 30(4):2866–2896, 2020.
- [42] L. Wang, B. Peng, J. Bradic, R. Li, and Y. Wu. A tuning-free robust and efficient approach to high-dimensional regression. *Journal of the American Statistical Association*, 115(532):1700–1714, 2020.
- [43] B. Wohlberg. Admm penalty parameter selection by residual balancing. *arXiv preprint arXiv:1704.06209*, 2017.

- [44] Y. Xu. Accelerated first-order primal-dual proximal methods for linearly constrained composite convex programming. *SIAM Journal on Optimization*, 27(3):1459–1484, 2017.
- [45] Y. Xu, I. Akrotirianakis, and A. Chakraborty. Proximal gradient method for huberized support vector machine. *Pattern Analysis and Applications*, 19:989–1005, 2016.
- [46] Y. Xu and S. Zhang. Accelerated primal–dual proximal block coordinate updating methods for constrained convex optimization. *Computational Optimization and Applications*, 70:91–128, 2018.
- [47] Z. Xu, M. Figueiredo, and T. Goldstein. Adaptive admm with spectral penalty parameter selection. In *Artificial Intelligence and Statistics*, pages 718–727. PMLR, 2017.
- [48] Z. Xu, M. A. Figueiredo, X. Yuan, C. Studer, and T. Goldstein. Adaptive relaxed admm: Convergence theory and practical implementation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7389–7398, 2017.
- [49] Z. Xu, G. Taylor, H. Li, M. A. Figueiredo, X. Yuan, and T. Goldstein. Adaptive consensus admm for distributed optimization. In *International Conference on Machine Learning*, pages 3841–3850. PMLR, 2017.
- [50] L. Yang and K.-C. Toh. Bregman proximal point algorithm revisited: A new inexact version and its inertial variant. *SIAM Journal on Optimization*, 32(3):1523–1554, 2022.

## A Proof details

### A.1 Proof of Lemma 3.1

*Proof.* From the optimality conditions in step 1 and 2, we have that

$$-\left(B^\top \lambda^k + \beta_k B^\top (By^{k+1} + Cz^k - b) + P_k(y^{k+1} - y^k)\right) \in \partial f(y^{k+1}) \quad (55)$$

$$-\left(C^\top \lambda^k + \beta_k C^\top (By^{k+1} + Cz^{k+1} - b) + Q_k(z^{k+1} - z^k)\right) \in \partial g(z^{k+1}). \quad (56)$$

From (55) and the convexity of  $f$ , we have that

$$\begin{aligned} f(y^{k+1}) - f(y) &\leq \langle B^\top \lambda^k + \beta_k B^\top (By^{k+1} + Cz^k - b) + P_k(y^{k+1} - y^k), y - y^{k+1} \rangle \\ &= \langle \lambda^k + \beta_k (By^{k+1} + Cz^k - b), By - By^{k+1} \rangle + \eta_{P_k}(y, y^k, y^{k+1}). \end{aligned} \quad (57)$$

Similarly, from (56) and (2), we have that

$$\begin{aligned} g(z^{k+1}) - g(z) &\leq \langle \lambda^k + \beta_k (By^{k+1} + Cz^{k+1} - b), Cz - Cz^{k+1} \rangle + \eta_{Q_k}(z, z^k, z^{k+1}) - \frac{\sigma_g}{2} \|z^{k+1} - z\|^2 \\ &= \langle \lambda^k + \beta_k (By^{k+1} + Cz^k - b), Cz - Cz^{k+1} \rangle + \eta_{\beta_k C^\top C + Q_k}(z, z^k, z^{k+1}) - \frac{\sigma_g}{2} \|z^{k+1} - z\|^2. \end{aligned} \quad (58)$$

From (57), (58) we have that

$$\begin{aligned}
& \mathcal{F}(x^{k+1}) - \mathcal{F}(x) \\
& \leq \langle \lambda^k + \beta_k(B y^{k+1} + C z^k - b), b - \mathcal{A}x^{k+1} \rangle + \eta_{\beta_k C^\top C + Q_k}(z, z^k, z^{k+1}) + \eta_{P_k}(y, y^k, y^{k+1}) \\
& \quad - \frac{\sigma_g}{2} \|z - z^{k+1}\|^2 \\
& = \langle \lambda^k + \beta_k(\mathcal{A}x^{k+1} - b), b - \mathcal{A}x^{k+1} \rangle + \beta_k \langle C(z^k - z^{k+1}), b - \mathcal{A}x^{k+1} \rangle \\
& \quad + \eta_{\beta_k C^\top C + Q_k}(z, z^k, z^{k+1}) + \eta_{P_k}(y, y^k, y^{k+1}) - \frac{\sigma_g}{2} \|z - z^{k+1}\|^2 \\
& = \langle \lambda^{k+1}, b - \mathcal{A}x^{k+1} \rangle + (\gamma - 1)\beta_k \|\mathcal{A}x^{k+1} - b\|^2 + \beta_k \langle C(z^k - z^{k+1}), b - \mathcal{A}x^{k+1} \rangle \\
& \quad + \eta_{\beta_k C^\top C + Q_k}(z, z^k, z^{k+1}) + \eta_{P_k}(y, y^k, y^{k+1}) - \frac{\sigma_g}{2} \|z - z^{k+1}\|^2, \tag{59}
\end{aligned}$$

where we have used step 3 to get the last equality. From (59), we have

$$\begin{aligned}
& \mathcal{L}(x^{k+1}, \lambda) - \mathcal{L}(x, \lambda) \\
& \leq \langle \lambda^{k+1} - \lambda, b - \mathcal{A}x^{k+1} \rangle + (\gamma - 1)\beta_k \|\mathcal{A}x^{k+1} - b\|^2 + \beta_k \langle C(z^k - z^{k+1}), b - \mathcal{A}x^{k+1} \rangle \\
& \quad + \eta_{\beta_k C^\top C + Q_k}(z, z^k, z^{k+1}) + \eta_{P_k}(y, y^k, y^{k+1}) - \frac{\sigma_g}{2} \|z - z^{k+1}\|^2 \\
& = \left\langle \lambda^{k+1} - \lambda, \frac{\lambda^k - \lambda^{k+1}}{\beta_k \gamma} \right\rangle + (\gamma - 1)\beta_k \|\mathcal{A}x^{k+1} - b\|^2 + \beta_k \langle C(z^k - z^{k+1}), b - \mathcal{A}x^{k+1} \rangle \\
& \quad + \eta_{\beta_k C^\top C + Q_k}(z, z^k, z^{k+1}) + \eta_{P_k}(y, y^k, y^{k+1}) - \frac{\sigma_g}{2} \|z - z^{k+1}\|^2. \tag{60}
\end{aligned}$$

Now, we need to estimate  $\beta_k \langle C(z^k - z^{k+1}), b - \mathcal{A}x^{k+1} \rangle$ . From (56), we know that

$$\begin{aligned}
& -C^\top \lambda^k - \beta_k C^\top (\mathcal{A}x^{k+1} - b) - Q_k(z^{k+1} - z^k) \in \partial g(z^{k+1}) \\
& -C^\top \lambda^{k-1} - \beta_{k-1} C^\top (\mathcal{A}x^k - b) - Q_{k-1}(z^k - z^{k-1}) \in \partial g(z^k)
\end{aligned}$$

Combining the above two equations together with the strongly convexity of  $g$ , we get

$$\left\langle \begin{aligned} & C^\top (\lambda^{k-1} - \lambda^k) - \beta_k C^\top (\mathcal{A}x^{k+1} - b) - Q_k(z^{k+1} - z^k) \\ & + \beta_{k-1} C^\top (\mathcal{A}x^k - b) + Q_{k-1}(z^k - z^{k-1}) \end{aligned}, z^{k+1} - z^k \right\rangle \geq \sigma_g \|z^k - z^{k+1}\|^2,$$

which, together with step 3, implies that

$$\begin{aligned}
& \sigma_g \|z^k - z^{k+1}\|^2 \\
& \leq \langle \lambda^{k-1} + \beta_{k-1}(\mathcal{A}x^k - b) - \lambda^k - \beta_k(\mathcal{A}x^{k+1} - b), C(z^{k+1} - z^k) \rangle \\
& \quad + \langle -Q_k(z^{k+1} - z^k) + Q_{k-1}(z^k - z^{k-1}), z^{k+1} - z^k \rangle \\
& = \langle (1 - \gamma)\beta_{k-1}(\mathcal{A}x^k - b) - \beta_k(\mathcal{A}x^{k+1} - b), C(z^{k+1} - z^k) \rangle \\
& \quad + \langle -Q_{k-1}(z^{k+1} - z^k) + Q_{k-1}(z^k - z^{k-1}), z^{k+1} - z^k \rangle - (\beta_k - \beta_{k-1}) \|z^{k+1} - z^k\|_{Q_k}^2.
\end{aligned}$$



The above inequality implies that

$$\begin{aligned}
& \beta_k \langle \mathcal{A}x^{k+1} - b, C(z^{k+1} - z^k) \rangle \\
& \leq (\gamma - 1) \langle \beta_{k-1}(b - \mathcal{A}x^k), C(z^{k+1} - z^k) \rangle - \|z^{k+1} - z^k\|_{Q_k}^2 \\
& \quad + \beta_{k-1} \langle Q(z^k - z^{k-1}), z^{k+1} - z^k \rangle - \sigma_g \|z^k - z^{k+1}\|^2 \\
& \leq \frac{(\gamma - 1)\beta_{k-1}^2}{2\gamma\beta_k} \|\mathcal{A}x^k - b\|^2 + \frac{(\gamma - 1)\gamma\beta_k}{2} \|C(z^{k+1} - z^k)\|^2 - \|z^{k+1} - z^k\|_{Q_k}^2 \\
& \quad + \frac{\beta_{k-1}^2}{2\beta_k} \|z^k - z^{k-1}\|_Q^2 + \frac{\beta_k}{2} \|z^{k+1} - z^k\|_Q^2 - \sigma_g \|z^k - z^{k+1}\|^2. \\
& = \frac{(\gamma - 1)\beta_{k-1}^2}{2\gamma\beta_k} \|\mathcal{A}x^k - b\|^2 + \frac{(1 - \delta)\beta_k}{2} \|C(z^{k+1} - z^k)\|^2 + \frac{\beta_{k-1}^2}{2\beta_k} \|z^k - z^{k-1}\|_Q^2 \\
& \quad - \frac{\beta_k}{2} \|z^{k+1} - z^k\|_Q^2 - \sigma_g \|z^k - z^{k+1}\|^2,
\end{aligned}$$

In the above, we use the fact that  $\gamma(\gamma - 1) = 1 - \delta$ . Now, we plug the above inequality into (60), we get

$$\begin{aligned}
& \mathcal{L}(x^{k+1}, \lambda) - \mathcal{L}(x, \lambda) \\
& \leq \left\langle \lambda^{k+1} - \lambda, \frac{\lambda^k - \lambda^{k+1}}{\beta_k \gamma} \right\rangle + (\gamma - 1)\beta_k \|\mathcal{A}x^{k+1} - b\|^2 + \frac{(\gamma - 1)\beta_{k-1}^2}{2\gamma\beta_k} \|\mathcal{A}x^k - b\|^2 + \eta_{P_k}(y, y^k, y^{k+1}) \\
& \quad + \eta_{\beta_k C^\top C + Q_k}(z, z^k, z^{k+1}) + \frac{(1 - \delta)\beta_k}{2} \|C(z^{k+1} - z^k)\|^2 + \frac{\beta_{k-1}^2}{2\beta_k} \|z^k - z^{k-1}\|_Q^2 - \frac{\beta_k}{2} \|z^{k+1} - z^k\|_Q^2 \\
& \quad - \sigma_g \|z^k - z^{k+1}\|^2 - \frac{\sigma_g}{2} \|z - z^{k+1}\|^2 \\
& \leq \left\langle \lambda^{k+1} - \lambda, \frac{\lambda^k - \lambda^{k+1}}{\beta_k \gamma} \right\rangle + (\gamma - 1)\beta_k \|\mathcal{A}x^{k+1} - b\|^2 + \frac{(\gamma - 1)\beta_{k-1}^2}{2\gamma\beta_k} \|\mathcal{A}x^k - b\|^2 + \eta_{P_k}(y, y^k, y^{k+1}) \\
& \quad + \xi_{\beta_k C^\top C + Q_k}(z, z^k, z^{k+1}) - \frac{\delta\beta_k}{2} \|C(z^{k+1} - z^k)\|^2 + \frac{\beta_{k-1}^2}{2\beta_k} \|z^k - z^{k-1}\|_Q^2 - \beta_k \|z^{k+1} - z^k\|_Q^2 \\
& \quad - \sigma_g \|z^k - z^{k+1}\|^2 - \frac{\sigma_g}{2} \|z - z^{k+1}\|^2. \tag{61}
\end{aligned}$$

Note that from step 4, we can derive that

$$\beta_k \left( \xi_{\beta_k C^\top C + Q_k}(z, z^k, z^{k+1}) - \frac{(1 - \epsilon)\sigma_g}{2} \|z - z^{k+1}\|^2 \right) \leq \frac{\beta_k^2}{2} \|z - z^k\|_{C^\top C + Q}^2 - \frac{\beta_{k+1}^2}{2} \|z - z^{k+1}\|_{C^\top C + Q}^2.$$

Multiply (61) by  $\beta_k$  and use the above inequality, we obtain that

$$\begin{aligned}
& \beta_k \left( \mathcal{L}(x^{k+1}, \lambda) - \mathcal{L}(x, \lambda) \right) \\
& \leq \frac{1}{\gamma} \langle \lambda^{k+1} - \lambda, \lambda^k - \lambda^{k+1} \rangle + (\gamma - 1) \beta_k^2 \| \mathcal{A}x^{k+1} - b \|^2 + \frac{(\gamma - 1) \beta_{k-1}^2}{2\gamma} \| \mathcal{A}x^k - b \|^2 \\
& \quad + \beta_k \eta_{P_k}(y, y^k, y^{k+1}) + \frac{\beta_k^2}{2} \| z - z^k \|_{C^\top C + Q}^2 - \frac{\beta_{k+1}^2}{2} \| z - z^{k+1} \|_{C^\top C + Q}^2 - \frac{\delta \beta_k^2}{2} \| C(z^{k+1} - z^k) \|^2 \\
& \quad + \frac{\beta_{k-1}^2}{2} \| z^k - z^{k-1} \|_Q^2 - \beta_k^2 \| z^{k+1} - z^k \|_Q^2 - \sigma_g \beta_k \| z^k - z^{k+1} \|^2 - \frac{\epsilon \sigma_g \beta_k}{2} \| z - z^{k+1} \|^2 \\
& = \frac{1}{\gamma} \xi(\lambda, \lambda^k, \lambda^{k+1}) - \frac{(2 - \gamma) \beta_k^2}{2} \| \mathcal{A}x^{k+1} - b \|^2 + \frac{(\gamma - 1) \beta_{k-1}^2}{2\gamma} \| \mathcal{A}x^k - b \|^2 \\
& \quad + \beta_k \eta_{P_k}(y, y^k, y^{k+1}) + \frac{\beta_k^2}{2} \| z - z^k \|_{C^\top C + Q}^2 - \frac{\beta_{k+1}^2}{2} \| z - z^{k+1} \|_{C^\top C + Q}^2 - \frac{\delta \beta_k^2}{2} \| C(z^{k+1} - z^k) \|^2 \\
& \quad + \frac{\beta_{k-1}^2}{2} \| z^k - z^{k-1} \|_Q^2 - \beta_k^2 \| z^{k+1} - z^k \|_Q^2 - \sigma_g \beta_k \| z^k - z^{k+1} \|^2 - \frac{\epsilon \sigma_g \beta_k}{2} \| z - z^{k+1} \|^2
\end{aligned}$$

where we have used the fact that  $\langle \lambda_{k+1} - \lambda, \lambda_k - \lambda_{k+1} \rangle = \frac{1}{2} \| \lambda - \lambda_k \|^2 - \frac{1}{2} \| \lambda - \lambda_{k+1} \|^2 - \frac{1}{2} \| \lambda_k - \lambda_{k+1} \|^2$  and  $\lambda^{k+1} - \lambda^k = \gamma \beta_k (\mathcal{A}x^{k+1} - b)$ . Note that since  $\gamma \in (1, \frac{1+\sqrt{5}}{2})$ ,  $\delta = 1 + \gamma - \gamma^2 > 0$ . Using the identity,  $\frac{\gamma-1}{2\gamma} = \frac{(2-\gamma)}{2} - \frac{\delta}{2\gamma}$ , we deduce that

$$\begin{aligned}
& \beta_k \left( \mathcal{L}(x^{k+1}, \lambda) - \mathcal{L}(x, \lambda) \right) + \frac{\delta \beta_{k-1}^2}{2\gamma} \| \mathcal{A}x^k - b \|^2 \\
& \leq \frac{1}{\gamma} \xi(\lambda, \lambda^k, \lambda^{k+1}) + \frac{(2 - \gamma) \beta_{k-1}^2}{2} \| \mathcal{A}x^k - b \|^2 - \frac{(2 - \gamma) \beta_k^2}{2} \| \mathcal{A}x^{k+1} - b \|^2 \\
& \quad + \beta_k \eta_{P_k}(y, y^k, y^{k+1}) + \frac{\beta_k^2}{2} \| z - z^k \|_{C^\top C + Q}^2 - \frac{\beta_{k+1}^2}{2} \| z - z^{k+1} \|_{C^\top C + Q}^2 - \frac{\delta \beta_k^2}{2} \| C(z^{k+1} - z^k) \|^2 \\
& \quad + \frac{\beta_{k-1}^2}{2} \| z^k - z^{k-1} \|_Q^2 - \beta_k^2 \| z^{k+1} - z^k \|_Q^2 - \sigma_g \beta_k \| z^k - z^{k+1} \|^2 - \frac{\epsilon \sigma_g \beta_k}{2} \| z - z^{k+1} \|^2.
\end{aligned}$$

From here, one can readily get the required inequality in Lemma 3.1.  $\blacksquare$

## A.2 Proof of Lemma 3.2

*Proof.* Because  $(x^*, \lambda^*)$  is a KKT solution, we have that  $0 \in \partial_x \mathcal{L}(x^*, \lambda^*)$ . Since  $\mathcal{L}(x, \lambda^*)$  is a convex function of  $x$ , we then have that

$$\mathcal{L}(x^*, \lambda^*) \leq \mathcal{L}(x, \lambda^*) \text{ for any } x, \quad (62)$$

from which we get

$$-\langle \lambda^*, \mathcal{A}x^k - b \rangle \leq \mathcal{F}(x^k) - \mathcal{F}(x^*). \quad (63)$$

Consider all  $\lambda \in \mathbb{R}^m$  such that  $\|\lambda\| \leq \|\lambda^*\| + 1$  in  $\mathcal{L}(x^k, \lambda) - \mathcal{L}(x^*, \lambda) \leq h(k)D(\lambda)$ , we have

$$\mathcal{F}(x^k) - \mathcal{F}(x^*) + (\|\lambda^*\| + 1) \|\mathcal{A}x^k - b\| \leq h(k) \max_{\|\lambda\| \leq \|\lambda^*\| + 1} D(\lambda). \quad (64)$$

Using (63) in (64), we get

$$\|\mathcal{A}x^k - b\| \leq h(k) \max_{\|\lambda\| \leq \|\lambda^*\|+1} D(\lambda) = O(h(k)). \quad (65)$$

Now, using (65) in (63) and (64) respectively, we get  $|\mathcal{F}(x^k) - \mathcal{F}(x^*)| = O(h(k))$ .  $\blacksquare$

### A.3 Proof of Lemma 3.7

*Proof.* Substitute  $(x^*, \lambda^*)$  into (5), we get the following long inequality

$$\begin{aligned} & \beta_k \left( \mathcal{L}(x^{k+1}, \lambda^*) - \mathcal{L}(x^*, \lambda^*) \right) + \overbrace{\frac{\delta\beta_{k-1}^2}{2\gamma} \|\mathcal{A}x^k - b\|^2}^1 + \overbrace{\frac{\delta\beta_k^2}{2} \|C(z^{k+1} - z^k)\|^2}^0 \\ & + \overbrace{\frac{\beta_k}{2} \|y^k - y^{k+1}\|_{P_k}^2}^0 + \frac{\beta_k^2}{2} \|z^k - z^{k+1}\|_Q^2 + \frac{1}{2\gamma} \|\lambda^* - \lambda^{k+1}\|^2 + \frac{(2-\gamma)\beta_k^2}{2} \|\mathcal{A}x^{k+1} - b\|^2 \\ & + \frac{\beta_{k+1}^2}{2} \|z^* - z^{k+1}\|_{C^\top C+Q}^2 + \frac{\beta_k^2}{2} \|z^{k+1} - z^k\|_Q^2 + \overbrace{\frac{\beta_{k+1}}{2} \|y^* - y^{k+1}\|_{P_{k+1}}^2}^0 \\ & \leq \frac{1}{2\gamma} \|\lambda^* - \lambda^k\|^2 + \frac{(2-\gamma)\beta_{k-1}^2}{2} \|\mathcal{A}x^k - b\|^2 + \frac{\beta_k^2}{2} \|z^* - z^k\|_{C^\top C+Q}^2 + \frac{\beta_{k-1}^2}{2} \|z^k - z^{k-1}\|_Q^2 \\ & + \overbrace{\frac{\beta_k}{2} \|y^* - y^k\|_{P_k}^2}^0 - \overbrace{\sigma_g \beta_k \|z^k - z^{k+1}\|^2}^2 - \overbrace{\frac{\epsilon \sigma_g \beta_k}{2} \|z^* - z^{k+1}\|^2}^3 \end{aligned} \quad (66)$$

Now, we apply several operations to the above inequality: 1, ignore terms under “0” since  $P = 0$ ; 2, move the term under “1” to the right hand side; 3, move the term under “2” to the left hand side and apply  $\|z^{k+1} - z^k\|_Q^2 / \lambda_{\max}(Q) \leq \|z^k - z^{k+1}\|^2$ ; 4, move one half of “4” to the left hand side and apply  $\|z^{k+1} - z^*\|^2 \leq \|z^{k+1} - z^*\|_{C^\top C+Q}^2 / \lambda_{\max}(C^\top C+Q)$ . After all these operations, we will get the inequality (16).  $\blacksquare$

### A.4 Proof of Lemma 3.8

*Proof.* From step 2 and step 3 in IADMM, we have that

$$0 = \nabla g(z^{k+1}) + C^\top \lambda^{k+1} + (1-\gamma)\beta_k C^\top (\mathcal{A}x^{k+1} - b) + Q_k(z^{k+1} - z^k).$$

Since  $(y^*, z^*, \lambda^*)$  is a KKT solution, we have  $0 = \nabla g(z^*) + C^\top \lambda^*$ . Combining these two equations together with the Lipschitz continuity of  $\nabla g$ , we have

$$\begin{aligned} & \|C^\top (\lambda^{k+1} - \lambda^*) + (1-\gamma)\beta_k C^\top (\mathcal{A}x^{k+1} - b) + Q_k(z^{k+1} - z^k)\|^2 = \|\nabla g(z^{k+1}) - \nabla g(z^*)\|^2 \\ & \leq L_g^2 \|z^{k+1} - z^*\|^2. \end{aligned} \quad (67)$$

For  $0 < \alpha < \frac{1}{2}$ , by using the inequality  $\|u + v + w\|^2 \geq (1 - 2\alpha)\|u\|^2 - \frac{1}{\alpha}\|v\|^2 - \frac{1}{\alpha}\|w\|^2$ , we have that

$$\begin{aligned}
& \|C^\top(\lambda^{k+1} - \lambda^*) + (1 - \gamma)\beta_k C^\top(\mathcal{A}x^{k+1} - b) + Q_k(z^{k+1} - z^k)\|^2 \\
& \geq (1 - 2\alpha)\|C^\top(\lambda^{k+1} - \lambda^*)\|^2 - \frac{1}{\alpha}\|(1 - \gamma)\beta_k C^\top(\mathcal{A}x^{k+1} - b)\|^2 - \frac{1}{\alpha}\|Q_k(z^{k+1} - z^k)\|^2 \\
& \geq (1 - 2\alpha)\lambda_{\min}(CC^\top)\|\lambda^{k+1} - \lambda^*\|^2 - \frac{1}{\alpha}\lambda_{\max}(CC^\top)(1 - \gamma)^2\beta_k^2\|\mathcal{A}x^{k+1} - b\|^2 - \frac{1}{\alpha}\|Q_k(z^{k+1} - z^k)\|^2.
\end{aligned}$$

Plug this into (67), we get

$$\begin{aligned}
& (1 - 2\alpha)\lambda_{\min}(CC^\top)\|\lambda^{k+1} - \lambda^*\|^2 \\
& \leq \frac{1}{\alpha}\lambda_{\max}(CC^\top)(1 - \gamma)^2\beta_k^2\|\mathcal{A}x^{k+1} - b\|^2 + \frac{1}{\alpha}\|Q_k(z^{k+1} - z^k)\|^2 + L_g^2\|z^{k+1} - z^*\|^2 \\
& \leq \frac{1}{\alpha}\lambda_{\max}(CC^\top)(1 - \gamma)^2\beta_k^2\|\mathcal{A}x^{k+1} - b\|^2 + \frac{1}{\alpha}\lambda_{\max}(Q)\beta_k^2\|z^{k+1} - z^k\|_Q^2 + L_g^2\|z^{k+1} - z^*\|^2.
\end{aligned}$$

This completes the proof. ■