An Alternating Direction Method of Multipliers with a Worst-case $O(1/n^2)$ Convergence Rate

WENYI TIAN[*] XIAOMING YUAN[†]

Abstract. The alternating direction method of multipliers (ADMM) is being widely used for various convex programming models with separable structures arising in many scientific computing areas. The ADMM's worst-case $O(1/n)$ convergence rate measured by the iteration complexity has been established in the literature when its penalty parameter is a constant, where n is the iteration counter. Research on ADMM's worst-case $O(1/n^2)$ convergence rate, however, is still in its infancy. In this paper, we suggest applying a rule proposed recently by Chambolle and Pock to iteratively updating the penalty parameter and show that the ADMM with this adaptive penalty parameter has a worst-case $O(1/n^2)$ convergence rate in the ergodic sense. Without strong convexity requirement on the objective function, our assumptions on the model are mild and can be satisfied by some representative applications. We test the LASSO model and numerically verify the significant acceleration effectiveness of the faster ADMM with a worst-case $O(1/n^2)$ convergence rate. Moreover, the faster ADMM is more favorable than the ADMM with a constant penalty parameter, as the former is much less sensitive to the initial value of the penalty parameter and can sometimes produce very high-accuracy solutions.

Keywords. Convex programming, Alternating direction method of multipliers, Convergence rate, Acceleration, First order methods

1 Introduction

Many applications can be modeled as such a convex minimization problem whose objective function is separable and representable by the sum of two or more functions. A representative case is where one function represents a data-fidelity term and the other is a regularization term; this case arises frequently in areas such as inverse problems, image processing and machine learning. For such a separable convex minimization model, the alternating direction method of multipliers (ADMM) proposed originally in [18] (see also [6, 16]) turns out to be a benchmark solver and it is being widely used for many applications in a broad spectrum of areas. We refer the reader to [3, 14, 17] for some review papers on the ADMM.

The convergence of ADMM has been well studied in earlier literature, e.g., [15, 16], and recently its worst-case $O(1/n)$ convergence rate measured by the iteration complexity has also been established in [24, 25, 29]. Here, n is the iteration counter and we refer to [31, 32, 33] for some seminal work of convergence rate analysis in terms of the iteration complexity. The main goal of this paper is to investigate under which scenario the ADMM has a faster worst-case $O(1/n^2)$ convergence rate. Note that it still remains open whether or not the ADMM can achieve a worst-case $O(1/n^2)$ convergence rate under the general setting where no special assumptions on the model are posed. This can be partially understood by the result in [8] (see Theorem 8 therein).

[*]Center for Applied Mathematics, Tianjin University, Tianjin 300072, China. This author was partially supported by the National Natural Science Foundation of China (No. 11701416). Email: twymath@gmail.com

[†]Department of Mathematics, The University of Hong Kong, Hong Kong, China. This author was partially supported by the General Research Fund from Hong Kong Research Grants Council:12300515. Email: xmyuan@hku.hk

To discuss the possibility of deriving a worst-case $O(1/n^2)$ convergence rate for the ADMM, we concentrate on the convex minimization model

$$\min_{x\in\mathcal{X}} f(x)+g(Ax) \tag{1.1}$$

where $\mathcal{X}\subset\mathbb{R}^d$ is closed and convex, $f:\mathcal{X}\to(-\infty,\infty]$ and $g:\mathbb{R}^m\to(-\infty,\infty]$ are closed, proper, and convex functions, g is smooth with a Lipschitz continuous gradient and $A\in\mathbb{R}^{m\times d}$ is full column rank. Throughout the solution set of (1.1) is assumed to be nonempty. The model (1.1) can be written as

$$\begin{aligned} \min\ & f(x)+g(y)\\ \text{s.t.}\ & Ax-y=0\\ & x\in\mathcal{X}, y\in\mathbb{R}^m, \end{aligned} \tag{1.2}$$

where $y\in\mathbb{R}^m$ is an auxiliary variable. Then, the iterative scheme of ADMM for (1.2) reads as

$$\begin{cases} x_{n+1}=\operatorname*{argmin}_{x\in\mathcal{X}}\Big\{f(x)+\dfrac{\sigma}{2}\Big\|Ax-y_n+\dfrac{\lambda_n}{\sigma}\Big\|^2\Big\}\\ y_{n+1}=\operatorname*{argmin}_{y\in\mathbb{R}^m}\Big\{g(y)+\dfrac{\sigma}{2}\Big\|Ax_{n+1}-y+\dfrac{\lambda_n}{\sigma}\Big\|^2\Big\}\\ \lambda_{n+1}=\lambda_n+\sigma(Ax_{n+1}-y_{n+1}), \end{cases} \tag{1.3}$$

with $\sigma>0$ the penalty parameter and $\lambda\in\mathbb{R}^m$ the Lagrange multiplier. There are different ways to understand the ADMM. For example, it can be regarded as a splitting version of the classical augmented Lagrangian method in [26, 34]; it was also explained in [15] as an application of the Douglas-Rachford splitting method (DRSM), which was first proposed in [11] for linear heat equations and then generalized in [27] to the nonlinear case, to the dual problem of (1.1); and it was further analyzed in [13] that the ADMM is an application of the proximal point algorithm (PPA) [28, 30] from the maximal monotone operator perspective.

An important variant of ADMM is the partially proximal version:

$$\begin{cases} x_{n+1}=\operatorname*{argmin}_{x\in\mathcal{X}}\Big\{f(x)+\dfrac{\sigma}{2}\Big\|Ax-y_n+\dfrac{\lambda_n}{\sigma}\Big\|^2+\dfrac{1}{2}\|x-x_n\|_Q^2\Big\}\\ y_{n+1}=\operatorname*{argmin}_{y\in\mathbb{R}^m}\Big\{g(y)+\dfrac{\sigma}{2}\Big\|Ax_{n+1}-y+\dfrac{\lambda_n}{\sigma}\Big\|^2\Big\}\\ \lambda_{n+1}=\lambda_n+\sigma(Ax_{n+1}-y_{n+1}), \end{cases} \tag{1.4}$$

where $Q\in\mathbb{R}^{d\times d}$ is a positive definite matrix; see e.g. [21]. In particular, when $Q=\mu I-\sigma A^TA$ with $\mu>\sigma\|A^TA\|$, it is easy to see that the x-subproblem in (1.4) reduces to

$$x_{n+1}=\operatorname*{argmin}_{x\in\mathcal{X}}\Big\{f(x)+\frac{\mu}{2}\Big\|x-x_n+\frac{1}{\mu}A^T\big(\lambda_n+\sigma(Ax_n-y_n)\big)\Big\|^2\Big\}.$$

When $\mathcal{X}=\mathbb{R}^d$, the minimization problem in the equation above amounts to computing the proximity operator of f:

$$\operatorname{prox}_{\gamma f}(x):=\operatorname*{argmin}_{y}\Big\{f(y)+\frac{1}{2\gamma}\|y-x\|^2\Big\}, \tag{1.5}$$

with $\gamma>0$. Note that the proximity operator (1.5) has a closed-form solution for some interesting cases such as $f(x)=\|x\|_1$. In this case, the proximal version of ADMM (1.4) reduces to the linearized version of ADMM for (1.2):

$$
\left\{
\begin{aligned}
&x_{n+1} = \operatorname*{argmin}_{x\in\mathcal{X}}\Big\{f(x) + \frac{\mu}{2}\big\|x - x_n + \frac{1}{\mu}A^T\big(\lambda_n + \sigma(Ax_n - y_n)\big)\big\|^2\Big\}\\
&y_{n+1} = \operatorname*{argmin}_{y\in\mathbb{R}^m}\Big\{g(y) + \frac{\sigma}{2}\big\|Ax_{n+1} - y + \frac{\lambda_n}{\sigma}\big\|^2\Big\}\\
&\lambda_{n+1} = \lambda_n + \sigma(Ax_{n+1} - y_{n+1}).
\end{aligned}
\right. \tag{1.6}
$$

We refer to, e.g., [40, 41, 43], for some efficient applications of the linearized version of ADMM. Note that we only consider the case where the x-subproblem is proximally regularized because it is useful enough for most of the ADMM's applications. Technically, there is no difficulty if both the subproblems are proximally regularized, see e.g., [12, 21].

In [4], the problem (1.1) with $\mathcal{X} = \mathbb{R}^d$ was written as the saddle-point problem

$$
\min_{x\in\mathbb{R}^d}\max_{\lambda\in\mathbb{R}^m}\big\{f(x) + (Ax,\lambda) - g^*(\lambda)\big\}, \tag{1.7}
$$

where $g^*(\lambda) := \sup_y\{(y,\lambda) - g(y)\}$ is the Fenchel conjugate of $g(y)$ (see, e.g., [35]). Then, the following generalized primal-dual algorithm was proposed to solve (1.7):

$$
\left\{
\begin{aligned}
&\lambda_{n+1} = \operatorname{prox}_{\sigma g^*}\big(\lambda_n + \sigma A\tilde{x}_n\big)\\
&x_{n+1} = \operatorname{prox}_{\tau f}\big(x_n - \tau A^T\lambda_{n+1}\big)\\
&\tilde{x}_{n+1} = x_{n+1} + \theta(x_{n+1} - x_n),
\end{aligned}
\right. \tag{1.8}
$$

where "prox" is defined in (1.5), $\theta\in[0,1]$ is a combination parameter, $\tau > 0$ and $\sigma > 0$ are two constants. For the scheme (1.8) with $\theta = 1$, its worst-case $O(1/n)$ convergence rate in the ergodic sense was established in [4] under the condition $\tau\sigma\|A\|^2 < 1$. Then, the scheme (1.8) was extended in [23] with the combination parameter $\theta\in[-1,1]$; some correction steps were combined with the primal-dual step and the convergence was established under the condition

$$
\tau\sigma\frac{(1+\theta)^2}{4}\|A^TA\| < 1.
$$

Convergence analysis for the scheme (1.8) with $\theta\in[-1,1]$ were further established in [22, 38] for the case where f is strongly convex. As analyzed in [4], the primal-dual algorithm (1.8) with $\theta = 1$ is equivalent to the application of the linearized (also called preconditioned) version of ADMM with $Q = \sigma^{-1}I - \tau AA^T$ and $\tau\sigma\|A\|^2 < 1$ for the dual problem of (1.1).

Moreover, in [4] (see also [5]), the authors suggested choosing the involved parameters θ, τ and σ dynamically, instead of constants, in (1.8). That is, they considered the generalized primal-dual algorithm with dynamically adjusted parameters:

$$
\left\{
\begin{aligned}
&\lambda_{n+1} = \operatorname{prox}_{\sigma_n g^*}\big(\lambda_n + \sigma_n A\tilde{x}_n\big)\\
&x_{n+1} = \operatorname{prox}_{\tau_n f}\big(x_n - \tau_n A^T\lambda_{n+1}\big)\\
&\tilde{x}_{n+1} = x_{n+1} + \theta_{n+1}(x_{n+1} - x_n).
\end{aligned}
\right. \tag{1.9}
$$

A worst-case $O(1/n^2)$ convergence rate of (1.9) in the ergodic sense was established in [4] provided that these parameters satisfy the conditions:

$$
\theta_{n+1} = \frac{1}{\sqrt{1+2\gamma\tau_n}},\ \tau_{n+1} = \theta_{n+1}\tau_n,\ \sigma_{n+1} = \sigma_n/\theta_{n+1}. \tag{1.10}
$$

It turns out that these conditions are crucial for establishing the desired $O(1/n^2)$ convergence rates in [4, 5]. Thus, compared with the primal-dual scheme (1.8) with constant parameters, the scheme (1.9)-(1.10) possesses a higher convergence rate in terms of the iteration complexity despite that it additionally requires to determine three parameter sequences. These existing results strongly inspire us to consider under which conditions the ADMM (1.3) with dynamically adjusted penalty parameters has a worst-case $O(1/n^2)$ convergence rate.

In the literature, there are some works related to how to derive a worst-case $O(1/n^2)$ convergence rate for the ADMM; which either require stronger assumptions or are eligible only for some variants of the original ADMM scheme (1.3). In [9, 19], the following more general problem was considered:

$$\begin{aligned} &\min\ f(x) + g(y) \\ &\text{s.t.}\ \ Ax + By = b, \end{aligned} \tag{1.11}$$

where $f : \mathbb{R}^{d_1} \to (-\infty, \infty]$ and $g : \mathbb{R}^{d_2} \to (-\infty, \infty]$ are closed, proper and convex functions, $A \in \mathbb{R}^{m\times d_1}$, $B \in \mathbb{R}^{m\times d_2}$ and $b \in \mathbb{R}^m$. It was shown in [9] that if g is strongly convex and the ADMM's penalty parameter σ is less than a constant depending on the strong convexity modulus of g and the norm of the matrix B, then the residual of the constraint in (1.11) is decreasing in order of $o(1/n^2)$ in a nonergodic sense while the measurement of the error of the objective function in the primal model (1.11) is still in order of $o(1/n)$[1]. In [19], a modified version of the ADMM was proposed by combining it with the acceleration technique in [33]. Then, it was shown that the error of the objective function of the dual problem of (1.11) has a worst-case $O(1/n^2)$ convergence rate in a nonergodic sense, under the conditions that both f and g are strongly convex with g being further assumed to be quadratic, and the penalty parameter σ is less than some constant depending on the strong convexity modulus of f and g, the spectral radius of matrices A and B.

In this paper, we will establish a worst-case $O(1/n^2)$ convergence rate in the ergodic sense for both the original ADMM scheme (1.3) and the proximal version (1.4) with the penalty parameter σ replaced by adaptive ones σ_n, under the condition that the penalty parameter σ_n is iteratively adjusted by a specific rule similar as the one in [4, 5]. The restriction of the penalty parameter is mild and can be automatically determined with a given initial value (see (2.3) and (2.4)). We also show that the proximity of the Lagrange multiplier $\{\lambda_n\}$ to the optimal value is reduced on an $O(1/n^2)$ rate. Our assumptions on the model (1.1) are given at the beginning of Section 2. Note that we do not assume any strong convexity on the objective function of the model (1.1) as in existing work such as [7, 9, 19, 37].

Finally, we would mention that some convergence rate results in the asymptotic sense can be established for the ADMM if further assumptions are made. For example, the asymptotic linear convergence rate of ADMM was established in [2, 20] for the special case of (1.2) where both functions are quadratic. But this type of analysis is not the focus of this paper.

The rest of the paper is organized as follows. In Section 2, a faster ADMM is proposed and some remarks are given. We first prove the convergence for the faster ADMM in Section 3 and then establish its worst-case $O(1/n^2)$ convergence rate in Section 4. In Section 5, we elaborate on the connection between the faster ADMM and the primal-dual algorithm in [4] and it's variants. In Section 6, we test the LASSO model and report some preliminary numerical results; some conclusions are also drawn based on these numerical results.

[1]In our discussion, for simplicity, we do not differentiate the orders of $O(1/n)$ and $o(1/n)$ (also $O(1/n^2)$ and $o(1/n^2)$) because of two reasons. First, they are of the same order in the worst-case nature; thus usually their difference is not that significant. Second, technically, for some basic operator splitting methods it is not hard to improve an $O(1/n)$ rate to $o(1/n)$ or from $O(1/n^2)$ to $o(1/n^2)$, see, e.g., [7].

2 A Faster ADMM

As mentioned, we consider the original ADMM (1.3) and its proximal version (1.4) simultaneously for (1.2) with adaptive penalty parameters $\{\sigma_n\}$. So we relax the restriction of Q in (1.4) and only require it to be positive semi-definite in our discussion. We slightly abuse the notation $\|x\|_Q^2$ to denote the number x^TQx even if Q may only be positive semi-definite. To derive a worst-case $O(1/n^2)$ convergence rate for the ADMM, our assumptions on the model (1.1) are summarized as follows.

Assumption: Both $f(x)$ and $g(x)$ are closed, proper, and convex functions; $g(x)$ is smooth and ∇g is Lipschitz continuous with constant $\frac{1}{\gamma}$; and A is full column rank.

Note that the model (1.1) with this assumption still includes a variety of applications such as the LASSO problem given in (6.1), despite that our assumption here is the reverse of some usual ones in the literature in which f is assumed to be smooth with a Lipschitz gradient while g is convex. Such a difference results in different orders of the subproblems for the implementation of ADMM; but our analysis for deriving a worst-case $O(1/n^2)$ convergence rate for the faster ADMM is based on this assumption. Furthermore, we notice that with the above assumption, it follows from [1, Theorem 18.15] that the following inequality holds:

$$g(y_1) \geq g(y_2) + \langle \nabla g(y_2), y_1 - y_2\rangle + \frac{\gamma}{2}\|\nabla g(y_1) - \nabla g(y_2)\|^2, \ \ \forall\, y_1, y_2 \in \mathbb{R}^m. \tag{2.1}$$

We propose the faster ADMM with a worst-case $O(1/n^2)$ convergence rate in Algorithm 1.

Algorithm 1: Faster ADMM with a worst-case $O(1/n^2)$ convergence rate.

Specify an integer $\kappa > 0$ as the frequency of adjusting the penalty parameter σ_n; an initial value of $\sigma_0 > 0$; and $(x_0, y_0, \lambda_0) \in \mathcal{X} \times \mathbb{R}^m \times \mathbb{R}^m$. Choose a positive semi-definite matrix $Q \in \mathbb{R}^{d\times d}$. For the $(n+1)$-th iteration, perform the following steps:

$$\begin{cases} x_{n+1} = \underset{x\in\mathcal{X}}{\operatorname{argmin}}\Big\{ f(x) + \dfrac{\sigma_n}{2}\Big\|Ax - y_n + \dfrac{\lambda_n}{\sigma_n}\Big\|^2 + \dfrac{\sigma_n}{2}\|x - x_n\|_Q^2 \Big\} \\ y_{n+1} = \underset{y\in\mathbb{R}^m}{\operatorname{argmin}}\Big\{ g(y) + \dfrac{\sigma_n}{2}\Big\|Ax_{n+1} - y + \dfrac{\lambda_n}{\sigma_n}\Big\|^2 \Big\} \\ \lambda_{n+1} = \lambda_n + \sigma_n(Ax_{n+1} - y_{n+1}), \end{cases} \tag{2.2}$$

where the penalty parameter σ_n is updated every κ iterations by the rule

$$\sigma_n = \tilde{\sigma}_{\lfloor \frac{n}{\kappa} \rfloor}, \tag{2.3}$$

where $\lfloor \frac{n}{\kappa} \rfloor$ is the largest integer no greater than $\frac{n}{\kappa}$ and the sequence $\{\tilde{\sigma}_i\}$ is given by

$$\tilde{\sigma}_{i+1} = \frac{\tilde{\sigma}_i}{\sqrt{1+\gamma\tilde{\sigma}_i}}, \text{ with } \tilde{\sigma}_0 = \sigma_0 > 0. \tag{2.4}$$

Remark 2.1. *Note that the sequence $\{\tilde{\sigma}_i\}$ is specified with a given σ_0 and the rule (2.3) adjusts the penalty parameter $\{\sigma_n\}$ after every κ iterations by assigning each $\tilde{\sigma}_i$ to κ iterations of the faster ADMM (2.2) consecutively. Thus, the sequence $\{\sigma_n\}$ is also automatically determined with a given initial value σ_0 and a frequency κ. For the extreme case* *where $\kappa = 1$, then we have*

$$\kappa = 1, \quad \begin{Bmatrix} \sigma_0 & \sigma_1 & \sigma_2 & \cdots & \sigma_n & \cdots \\ \downarrow & \downarrow & \downarrow & & \downarrow & \\ \tilde{\sigma}_0 & \tilde{\sigma}_1 & \tilde{\sigma}_2 & \cdots & \tilde{\sigma}_n & \cdots \end{Bmatrix},$$

which means the sequence $\{\sigma_n\}$ is iteratively updated by

$$\sigma_{n+1} = \frac{\sigma_n}{\sqrt{1+\gamma\sigma_n}}. \tag{2.5}$$

This is precisely the formula for updating the parameters of the accelerated primal-dual scheme in [4, 5]. If we choose $\kappa > 1$, e.g., $\kappa = 10$, then we have

$$\underbrace{\sigma_0 \ \ \sigma_1 \ \ \cdots \ \ \sigma_9}_{\tilde{\sigma}_0} \ \ \underbrace{\sigma_{10} \ \ \sigma_{11} \ \ \cdots \ \ \sigma_{19}}_{\tilde{\sigma}_1} \ \ \cdots;$$

and for the general κ, we have

$$\underbrace{\sigma_0 \ \ \cdots \ \ \sigma_{\kappa-1}}_{\tilde{\sigma}_0} \ \ \underbrace{\sigma_\kappa \ \ \cdots \ \ \sigma_{2\kappa-1}}_{\tilde{\sigma}_1} \ \ \cdots \ \ \underbrace{\sigma_{s\kappa} \ \ \cdots \ \ \sigma_{(s+1)\kappa-1}}_{\tilde{\sigma}_s} \ \ \cdots.$$

For an integer n, it can be decomposed as $n = s\kappa + j$ with $0 \le j \le \kappa - 1$. Thus, it follows

$$\sigma_{n+1} = \begin{cases} \sigma_n, & 0 \le j \le \kappa - 2, \\ \dfrac{\sigma_n}{\sqrt{1+\gamma\sigma_n}}, & j = \kappa - 1. \end{cases} \tag{2.6}$$

Clearly, the sequence $\{\sigma_n\}$ is monotonically non-increasing. Thus, it is easy to understand that if the sequence $\{\sigma_n\}$ is updated on a too high frequency, i.e., κ is small, then the sequence $\{\sigma_n\}$ decreases too fast and the step size for updating the dual variable becomes too small, especially when γ is relatively large. In this case, the efficiency of the scheme (2.2) may be deteriorated. On the other hand, if the sequence $\{\sigma_n\}$ is updated on a too low frequency, i.e., κ is huge, as we shall show in Theorems 4.1 and 4.2 (see (4.13) and (4.16)), the coefficient of the $O(1/n^2)$ convergence rate to be established is too large and it deteriorates the convergence also. So, in general we do not recommend too extreme values of κ for the proposed faster ADMM (2.2) with large γ. As we shall numerically verify later, medium values such as $\kappa = 5$ or 10, usually can result in very good numerical results even though the "optimal" choice, we believe, still depends on the specific application of the abstract model (1.2) and the data set under consideration.

3 Convergence

Recall our main goal is to establish a worst-case $O(1/n^2)$ convergence rate for Algorithm 1. First of all, in this section we prove the convergence of Algorithm 1.

Let the Lagrangian function of (1.2) be defined as

$$L(x, y; \lambda) := f(x) + g(y) + (\lambda, Ax - y), \tag{3.1}$$

with $\lambda \in \mathbb{R}^m$ the Lagrange multiplier. Further, we define $\Omega := \mathcal{X} \times \mathbb{R}^m \times \mathbb{R}^m$. Then, solving (1.2) is equivalent to finding a saddle point of $L(x, y; \lambda)$. This is equivalent to solving the variational inequality: finding $(x^*, y^*, \lambda^*) \in \Omega$ such that

$$\Theta(x, y) - \Theta(x^*, y^*) + \begin{pmatrix} A^T\lambda^* \\ -\lambda^* \\ -Ax^* + y^* \end{pmatrix}^T \begin{pmatrix} x - x^* \\ y - y^* \\ \lambda - \lambda^* \end{pmatrix} \ge 0, \ \ \forall\, (x, y, \lambda) \in \Omega, \tag{3.2}$$

where $\Theta(x,y)=f(x)+g(y)$.

To prove the convergence of the sequence $\{(x_n,y_n,\lambda_n)\}$ generated by Algorithm 1, we first give a lemma.

Lemma 3.1. *Let the sequence $\{(x_n,y_n,\lambda_n)\}$ be generated by Algorithm 1. Then we have*

$$\begin{aligned}&\Theta(x,y)-\Theta(x_{n+1},y_{n+1})+\left\{\begin{pmatrix}A^T\lambda_{n+1}\\-\lambda_{n+1}\\-Ax_{n+1}+y_{n+1}\end{pmatrix}^T\right.\\ &+\left.\begin{pmatrix}\sigma_nA^T(y_{n+1}-y_n)\\-\sigma_n(y_{n+1}-y_n)\\0\end{pmatrix}^T+\sigma_n\begin{pmatrix}Q(x_{n+1}-x_n)\\y_{n+1}-y_n\\\frac{1}{\sigma_n^2}(\lambda_{n+1}-\lambda_n)\end{pmatrix}^T\right\}\begin{pmatrix}x-x_{n+1}\\y-y_{n+1}\\\lambda-\lambda_{n+1}\end{pmatrix}\\ &\geq\frac{\gamma}{2}\|\nabla g(y)-\nabla g(y_{n+1})\|^2,\ \forall\,(x,y,\lambda)\in\Omega.\end{aligned}\tag{3.3}$$

Proof. First, the optimality condition of the x-subproblem in (2.2) is

$$f(x)-f(x_{n+1})+\Big(A^T\big(\sigma_n(Ax_{n+1}-y_n)+\lambda_n\big)+\sigma_nQ(x_{n+1}-x_n),x-x_{n+1}\Big)\geq0,\ \forall\,x\in\mathcal{X}.\tag{3.4}$$

Using the updating scheme for λ_{n+1} in (2.2), we obtain

$$f(x)-f(x_{n+1})+\Big(A^T\lambda_{n+1}+\sigma_nA^T(y_{n+1}-y_n)+\sigma_nQ(x_{n+1}-x_n),x-x_{n+1}\Big)\geq0,\ \forall\,x\in\mathcal{X}.\tag{3.5}$$

In addition, it yields from the optimality condition of the y-subproblem in (2.2) and the updating formula for λ_{n+1} that

$$\lambda_{n+1}=\nabla g(y_{n+1}),$$

then it follows from (2.1) that

$$g(y)-g(y_{n+1})-\big(\lambda_{n+1},y-y_{n+1}\big)\geq\frac{\gamma}{2}\|\nabla g(y)-\nabla g(y_{n+1})\|^2,\ \forall\,y\in\mathbb{R}^m.\tag{3.6}$$

Together with (3.5), (3.6) and the following identity

$$-Ax_{n+1}+y_{n+1}+\frac{1}{\sigma_n}(\lambda_{n+1}-\lambda_n)=0,\tag{3.7}$$

we obtain the result (3.3). □

Theorem 3.1. *Let (x^*,y^*,λ^*) be a saddle point of (3.1) and the sequence $\{(x_n,y_n,\lambda_n)\}$ be generated by Algorithm 1. Then we have*

$$\begin{aligned}&\|x_{n+1}-x^*\|_Q^2+\|y_{n+1}-y^*\|^2+\frac{1}{\sigma_{n+1}^2}\|\lambda_{n+1}-\lambda^*\|^2\\ &\leq\Big(\|x_n-x^*\|_Q^2+\|y_n-y^*\|^2+\frac{1}{\sigma_n^2}\|\lambda_n-\lambda^*\|^2\Big)\\ &\quad-\Big(\|x_{n+1}-x_n\|_Q^2+\|y_{n+1}-y_n\|^2+\frac{1}{\sigma_n^2}\|\lambda_{n+1}-\lambda_n\|^2\Big).\end{aligned}\tag{3.8}$$

Proof. From the y-subproblem in (2.2), it holds

$$\lambda_{n+1}=\nabla g(y_{n+1}),\quad\lambda^*=\nabla g(y^*).$$

Then, it follows from (3.3) with $(x, y, \lambda) = (x^*, y^*, \lambda^*)$ that

$$\begin{aligned}
&\sigma_n \begin{pmatrix} Q(x_{n+1} - x_n) \\ y_{n+1} - y_n \\ \frac{1}{\sigma_n^2}(\lambda_{n+1} - \lambda_n) \end{pmatrix}^T \begin{pmatrix} x^* - x_{n+1} \\ y^* - y_{n+1} \\ \lambda^* - \lambda_{n+1} \end{pmatrix} \\
&\geq \frac{\gamma}{2}\|\lambda^* - \lambda_{n+1}\|^2 + \Theta(x_{n+1}, y_{n+1}) - \Theta(x^*, y^*) \\
&\quad + \left\{ \begin{pmatrix} A^T\lambda_{n+1} \\ -\lambda_{n+1} \\ -Ax_{n+1} + y_{n+1} \end{pmatrix}^T + \begin{pmatrix} \sigma_n A^T(y_{n+1} - y_n) \\ -\sigma_n(y_{n+1} - y_n) \\ 0 \end{pmatrix}^T \right\} \begin{pmatrix} x_{n+1} - x^* \\ y_{n+1} - y^* \\ \lambda_{n+1} - \lambda^* \end{pmatrix}.
\end{aligned} \tag{3.9}$$

Taking $(x, y, \lambda) = (x_{n+1}, y_{n+1}, \lambda_{n+1})$ in (3.2) and adding the identity

$$\left\{ \begin{pmatrix} A^T\lambda_{n+1} \\ -\lambda_{n+1} \\ -Ax_{n+1} + y_{n+1} \end{pmatrix}^T - \begin{pmatrix} A^T\lambda^* \\ -\lambda^* \\ -Ax^* + y^* \end{pmatrix}^T \right\} \begin{pmatrix} x_{n+1} - x^* \\ y_{n+1} - y^* \\ \lambda_{n+1} - \lambda^* \end{pmatrix} = 0$$

to both sides, we have

$$\Theta(x_{n+1}, y_{n+1}) - \Theta(x^*, y^*) + \begin{pmatrix} A^T\lambda_{n+1} \\ -\lambda_{n+1} \\ -Ax_{n+1} + y_{n+1} \end{pmatrix}^T \begin{pmatrix} x_{n+1} - x^* \\ y_{n+1} - y^* \\ \lambda_{n+1} - \lambda^* \end{pmatrix} \geq 0. \tag{3.10}$$

By the updating formula for λ_{n+1} in (2.2) and the monotonicity of the gradient ∇g of the smooth function g, we obtain

(3.11)
$$\begin{aligned}
\begin{pmatrix} \sigma_n A^T(y_{n+1} - y_n) \\ -\sigma_n(y_{n+1} - y_n) \\ 0 \end{pmatrix}^T \begin{pmatrix} x_{n+1} - x^* \\ y_{n+1} - y^* \\ \lambda_{n+1} - \lambda^* \end{pmatrix} &= \big(y_{n+1} - y_n, \sigma_n(Ax_{n+1} - y_{n+1})\big) \\
&= (y_{n+1} - y_n, \lambda_{n+1} - \lambda_n) \\
&= \big(y_{n+1} - y_n, \nabla g(y_{n+1}) - \nabla g(y_n)\big) \geq 0.
\end{aligned}$$

Therefore, it follows from (3.9), (3.10) and (3.11) that

$$\begin{pmatrix} Q(x_{n+1} - x_n) \\ y_{n+1} - y_n \\ \frac{1}{\sigma_n^2}(\lambda_{n+1} - \lambda_n) \end{pmatrix}^T \begin{pmatrix} x^* - x_{n+1} \\ y^* - y_{n+1} \\ \lambda^* - \lambda_{n+1} \end{pmatrix} \geq \frac{\gamma}{2\sigma_n}\|\lambda^* - \lambda_{n+1}\|^2. \tag{3.12}$$

Using the identities

$$\begin{aligned}
\big(Q(x_{n+1} - x_n), x^* - x_{n+1}\big) &= \frac{1}{2}\big(\|x^* - x_n\|_Q^2 - \|x^* - x_{n+1}\|_Q^2 - \|x_n - x_{n+1}\|_Q^2\big), \\
\big(y_{n+1} - y_n, y^* - y_{n+1}\big) &= \frac{1}{2}\big(\|y^* - y_n\|^2 - \|y^* - y_{n+1}\|^2 - \|y_n - y_{n+1}\|^2\big), \\
\big(\lambda_{n+1} - \lambda_n, \lambda^* - \lambda_{n+1}\big) &= \frac{1}{2}\big(\|\lambda^* - \lambda_n\|^2 - \|\lambda^* - \lambda_{n+1}\|^2 - \|\lambda_n - \lambda_{n+1}\|^2\big),
\end{aligned}$$

and (3.12), we have

$$\begin{aligned}
&\|x_{n+1} - x^*\|_Q^2 + \|y_{n+1} - y^*\|^2 + \frac{1 + \gamma\sigma_n}{\sigma_n^2}\|\lambda_{n+1} - \lambda^*\|^2 \\
&\leq \Big(\|x_n - x^*\|_Q^2 + \|y_n - y^*\|^2 + \frac{1}{\sigma_n^2}\|\lambda_n - \lambda^*\|^2\Big) \\
&\quad - \Big(\|x_{n+1} - x_n\|_Q^2 + \|y_{n+1} - y_n\|^2 + \frac{1}{\sigma_n^2}\|\lambda_{n+1} - \lambda_n\|^2\Big).
\end{aligned} \tag{3.13}$$

Moreover, according to (2.6), it holds

$$\frac{1+\gamma\sigma_n}{\sigma_n^2} \geq \frac{1}{\sigma_{n+1}^2}. \tag{3.14}$$

Thus, it follows from (3.13) and (3.14) that the assertion (3.8) holds. □

The assertion (3.8) essentially implies the convergence of the sequence $\{(x_n, y_n, \lambda_n)\}$. We prove a lemma and then present the convergence result.

Recall the special case of the rule (2.3)-(2.4) with $\kappa = 1$, i.e., the sequence $\{\sigma_n\}$ is updated by (2.5). Then, as proved in [4], we have $\lim\limits_{n\to\infty} \frac{\gamma}{2} n\sigma_n = 1$, which means $\sigma_n \sim O(1/n)$. Now, we generalize this result to the general rule (2.3)-(2.4) with a general frequency κ.

Lemma 3.2. *For $\{\sigma_n\}$ updated by the rule (2.3)-(2.4) in Algorithm 1, we have $\sigma_n \sim O(\kappa/n)$.*

Proof. For $\{\sigma_n\}$ updated by the rule (2.3)-(2.4), we have $\lim\limits_{s\to\infty} \frac{\gamma}{2} s\tilde{\sigma}_s = 1$. For an integer n, it can be written as $n = s\kappa + j$, $0 \leq j \leq \kappa - 1$, where $s = \lfloor \frac{n}{\kappa} \rfloor$. Thus, we have $\lim\limits_{n\to\infty} \frac{\gamma}{2} n\sigma_n = \kappa$, because κ is a fixed integer. This yields $\sigma_n = \tilde{\sigma}_{\lfloor \frac{n}{\kappa} \rfloor} \sim O(\kappa/n)$ and completes the proof. □

Theorem 3.2. *Let $\{(x_n, y_n, \lambda_n)\}$ be the sequence generated by Algorithm 1. Then, the sequence $\{(x_n, y_n, \lambda_n)\}$ converges to a saddle point (x^*, y^*, λ^*) of (3.1). In addition, we have*

$$\lim_{n\to\infty} (Ax_n - y_n) = 0, \ \ \textit{and} \ \ \lim_{n\to\infty} \{f(x_n) + g(y_n)\} = f(x^*) + g(y^*). \tag{3.15}$$

Proof. Taking the summation of (3.8) for n from 0 to N, we have

$$\begin{aligned}&\sum_{n=0}^{N} \Big(\|x_{n+1} - x_n\|_Q^2 + \|y_{n+1} - y_n\|^2 + \frac{1}{\sigma_n^2} \|\lambda_{n+1} - \lambda_n\|^2 \Big) \\ &\leq \Big(\|x_0 - x^*\|_Q^2 + \|y_0 - y^*\|^2 + \frac{1}{\sigma_0^2} \|\lambda_0 - \lambda^*\|^2 \Big),\end{aligned}$$

which indicates

$$\lim_{n\to\infty} \Big(\|x_{n+1} - x_n\|_Q^2 + \|y_{n+1} - y_n\|^2 + \frac{1}{\sigma_n^2} \|\lambda_{n+1} - \lambda_n\|^2 \Big) = 0. \tag{3.16}$$

Then we have

$$\lim_{n\to\infty} Q(x_n - x_{n+1}) = 0, \ \lim_{n\to\infty} (y_n - y_{n+1}) = 0 \ \text{ and } \ \lim_{n\to\infty} \frac{1}{\sigma_n} (\lambda_n - \lambda_{n+1}) = 0. \tag{3.17}$$

It follows from (3.8) that $\big(\|x_n - x^*\|_Q^2 + \|y_n - y^*\|^2 + \frac{1}{\sigma_n^2} \|\lambda_n - \lambda^*\|^2\big)$ is bounded. Recall the identity (3.7), we know that $\|Ax_n - Ax^*\|^2$ is bounded. Since A is full column rank and $\sigma_n \sim O(\kappa/n)$ shown in Lemma 3.2, we conclude that the sequence $\{(x_n, y_n, \lambda_n)\}$ has a cluster point. Let us denote it by (x^*, y^*, λ^*). Then, substituting it into (3.3) and using (3.17), we obtain

$$\Theta(x, y) - \Theta(x^*, y^*) + \begin{pmatrix} A^T\lambda^* \\ -\lambda^* \\ -Ax^* + y^* \end{pmatrix}^T \begin{pmatrix} x - x^* \\ y - y^* \\ \lambda - \lambda^* \end{pmatrix} \geq 0, \ \forall \, (x, y, \lambda) \in \Omega,$$

which implies that (x^*, y^*, λ^*) is a saddle point of (3.1). Furthermore, by (3.7) and (3.17), it immediately yields

$$\lim_{n\to\infty} (Ax_n - y_n) = \lim_{n\to\infty} \frac{1}{\sigma_{n-1}} (\lambda_n - \lambda_{n-1}) = 0. \tag{3.18}$$

Taking $(x, y, \lambda) = (x_n, y_n, \lambda_n)$ in (3.2), we get

$$f(x_n) + g(y_n) \geq f(x^*) + g(y^*) - (\lambda^*, Ax_n - y_n),$$

and thus

$$\liminf_{n\to\infty} \{f(x_n) + g(y_n)\} \geq f(x^*) + g(y^*). \tag{3.19}$$

In addition, we set $(x, y, \lambda) = (x^*, y^*, \lambda^*)$ in (3.3) and simplify it as

$$\begin{aligned} f(x^*) + g(y^*) \geq\ & f(x_{n+1}) + g(y_{n+1}) + (\lambda^*, Ax_{n+1} - y_{n+1}) \\ & + \sigma_n (y_{n+1} - y_n, Ax_{n+1} - y_{n+1}) + \sigma_n \big(Q(x_{n+1} - x_n), x_{n+1} - x^*\big) \\ & + \sigma_n \big(y_{n+1} - y_n, y_{n+1} - y^*\big) + \frac{1}{\sigma_n} \big(\lambda_{n+1} - \lambda_n, \lambda_{n+1} - \lambda^*\big). \end{aligned}$$

Thus, using the boundedness of the sequence $\big(\|x_n - x^*\|_Q^2 + \|y_n - y^*\|^2 + \frac{1}{\sigma_n^2}\|\lambda_n - \lambda^*\|^2\big)$, the results in (3.16), (3.17) and (3.18), and $\sigma_n \sim O(\kappa/n)$ shown in Lemma 3.2, we obtain

$$f(x^*) + g(y^*) \geq \limsup_{n\to\infty} \{f(x_n) + g(y_n)\}. \tag{3.20}$$

Therefore, (3.18), (3.19) and (3.20) imply the assertion (3.15) and the proof is complete. □

Remark 3.1. *If the matrix A in the problem (1.1) is not full rank but Q in Algorithm 1 is chosen to be positive definite, then similar results as Theorem 3.2 can be derived.*

4 Worst-case $O(1/n^2)$ Convergence Rate

In this section, we establish a worst-case $O(1/n^2)$ convergence rate for Algorithm 1. Our analysis is based on the saddle-point reformulation of the model (1.1)

$$\min_{x\in\mathcal{X}} \max_{\lambda\in\mathbb{R}^m} \big\{\mathcal{L}(x, \lambda) := f(x) + (Ax, \lambda) - g^*(\lambda)\big\}. \tag{4.1}$$

As mentioned in [4], the following partial primal-dual gap can be used to measure the accuracy of an iterate generated by Algorithm 1:

$$\mathcal{G}_{\mathcal{B}_1\times\mathcal{B}_2}(x, \lambda) := \max_{\lambda'\in\mathcal{B}_2} \mathcal{L}(x, \lambda') - \min_{x'\in\mathcal{B}_1} \mathcal{L}(x', \lambda), \tag{4.2}$$

where $\mathcal{B}_1 \times \mathcal{B}_2$ is a bounded subset of the space $U := \mathcal{X} \times \mathbb{R}^m$ containing a solution point (x^*, λ^*) of the saddle-point reformulation (4.1). According to [4], we know that for $(\hat{x}, \hat{\lambda})$ in $\mathcal{B}_1 \times \mathcal{B}_2$, if $\mathcal{G}_{\mathcal{B}_1\times\mathcal{B}_2}(\hat{x}, \hat{\lambda}) \leq 0$, then $(\hat{x}, \hat{\lambda})$ is also a solution point of (4.1). Hence, we can define $(\tilde{x}, \tilde{\lambda}) \in \mathcal{B}_1 \times \mathcal{B}_2$ as an approximate solution to (4.1) with an accuracy of ϵ if

$$\mathcal{G}_{\mathcal{B}_1\times\mathcal{B}_2}(\tilde{x}, \tilde{\lambda}) \leq \epsilon$$

with $\epsilon > 0$.

Now we start to establish a worst-case $O(1/n^2)$ convergence rate for Algorithm 1. First, based on the first-order optimality conditions of the subproblems in (2.2), we prove a lemma.

Lemma 4.1. *Let $\{(x_n, y_n, \lambda_n)\}$ be the sequence generated by Algorithm 1. Then, we have*

$$f(x) - f(x_{n+1}) + \Big(A^T\Big(\sigma_n A(x_{n+1} - x_n) + \frac{\sigma_n}{\sigma_{n-1}}(\lambda_n - \lambda_{n-1}) + \lambda_n\Big) \\ + \sigma_n Q(x_{n+1} - x_n), x - x_{n+1}\Big) \geq 0, \;\; \forall\, x \in \mathcal{X}; \tag{4.3}$$

$$\lambda_{n+1} = \nabla g(y_{n+1}); \tag{4.4}$$

$$g^*(\lambda) - g^*(\lambda_{n+1}) - \big(Ax_{n+1} - \frac{1}{\sigma_n}(\lambda_{n+1} - \lambda_n), \lambda - \lambda_{n+1}\big) \\ - \frac{\gamma}{2}\|\lambda - \lambda_{n+1}\|^2 \geq 0, \;\; \forall\, \lambda \in \mathbb{R}^m. \tag{4.5}$$

Proof. The assertion (4.3) is an immediate result by inserting the updating formula for the multiplier $\lambda_n = \lambda_{n-1} + \sigma_{n-1}(Ax_n - y_n)$ into the optimality condition (3.4) of the x-subproblem. The second assertion (4.4) can be derived from the optimality condition of the y-subproblem by using the updating formula for λ_{n+1} in (2.2). Furthermore, because of (4.4), we have

$$y_{n+1} \in \partial g^*(\lambda_{n+1}). \tag{4.6}$$

Since ∇g is Lipschitz continuous with constant $\frac{1}{\gamma}$, it follows from [1, Theorem 18.15] that g^* is γ-strongly convex. Then, together with (4.6), we obtain

$$g^*(\lambda) - g^*(\lambda_{n+1}) - \big(y_{n+1}, \lambda - \lambda_{n+1}\big) - \frac{\gamma}{2}\|\lambda - \lambda_{n+1}\|^2 \geq 0, \;\; \forall\, \lambda \in \mathbb{R}^m.$$

Thus, using the updating scheme for λ in (2.2), we prove the assertion (4.5). □

Then, we prove one more lemma, based on which the worst-case $O(1/n^2)$ convergence rate of Algorithm 1 can be easily obtained.

Lemma 4.2. *Let $\{(x_n, \lambda_n)\}$ be generated by Algorithm 1 and $U = \mathcal{X} \times \mathbb{R}^m$. Then, we have*

$$\big(\mathcal{L}(x_{n+1}, \lambda) - \mathcal{L}(x, \lambda_{n+1})\big) + \frac{1}{\theta_{n+1}} S_{n+1}(x, \lambda) \leq S_n(x, \lambda), \;\; \forall\, (x, \lambda) \in U, \tag{4.7}$$

where

$$\theta_n := \frac{\sigma_n}{\sigma_{n-1}}$$

and

$$S_n(x, \lambda) := \frac{\sigma_n}{2}\|A(x - x_n)\|^2 + \frac{\sigma_n}{2}\|x - x_n\|_Q^2 + \frac{1}{2\sigma_n}\|\lambda - \lambda_n\|^2 \\ + \frac{\theta_n^2}{2\sigma_n}\|\lambda_n - \lambda_{n-1}\|^2 + \theta_n\big(A(x - x_n), \lambda_n - \lambda_{n-1}\big). \tag{4.8}$$

Proof. First, it follows from (4.3) and (4.5) that

$$\Big(f(x) + (Ax, \lambda_{n+1}) - g^*(\lambda_{n+1})\Big) - \Big(f(x_{n+1}) + (Ax_{n+1}, \lambda) - g^*(\lambda)\Big) \\ + \sigma_n\big((A^TA + Q)(x_{n+1} - x_n), x - x_{n+1}\big) + \frac{1}{\sigma_n}(\lambda_{n+1} - \lambda_n, \lambda - \lambda_{n+1}) \\ \geq \big(A(x - x_{n+1}), \lambda_{n+1} - \lambda_n\big) - \frac{\sigma_n}{\sigma_{n-1}}\big(A(x - x_{n+1}), \lambda_n - \lambda_{n-1}\big) \\ + \frac{\gamma}{2}\|\lambda - \lambda_{n+1}\|^2, \;\; \forall\, (x, \lambda) \in U. \tag{4.9}$$

With the definition of $\mathcal{L}(\cdot,\cdot)$ in (4.1) and the equality

$$\big(A(x-x_{n+1}),\lambda_n-\lambda_{n-1}\big)=\big(A(x-x_n),\lambda_n-\lambda_{n-1}\big)-\big(A(x_{n+1}-x_n),\lambda_n-\lambda_{n-1}\big),$$

we can reformulate (4.9) as

$$\begin{aligned}
&-\big(\mathcal{L}(x_{n+1},\lambda)-\mathcal{L}(x,\lambda_{n+1})\big)+\sigma_n\big(A(x_{n+1}-x_n),A(x-x_{n+1})\big)\\
&+\sigma_n\big(Q(x_{n+1}-x_n),x-x_{n+1}\big)+\frac{1}{\sigma_n}\big(\lambda_{n+1}-\lambda_n,\lambda-\lambda_{n+1}\big)\\
\geq\ &\big(A(x-x_{n+1}),\lambda_{n+1}-\lambda_n\big)-\frac{\sigma_n}{\sigma_{n-1}}\big(A(x-x_n),\lambda_n-\lambda_{n-1}\big)\\
&+\frac{\sigma_n}{\sigma_{n-1}}\big(A(x_{n+1}-x_n),\lambda_n-\lambda_{n-1}\big)+\frac{\gamma}{2}\|\lambda-\lambda_{n+1}\|^2\\
\geq\ &\big(A(x-x_{n+1}),\lambda_{n+1}-\lambda_n\big)-\frac{\sigma_n}{\sigma_{n-1}}\big(A(x-x_n),\lambda_n-\lambda_{n-1}\big)\\
&-\frac{\sigma_n}{2}\|A(x_{n+1}-x_n)\|^2-\frac{\sigma_n^2}{2\sigma_{n-1}^2\sigma_n}\|\lambda_n-\lambda_{n-1}\|^2+\frac{\gamma}{2}\|\lambda-\lambda_{n+1}\|^2,\ \ \forall\,(x,\lambda)\in U.
\end{aligned}$$

Further, using the identities

$$\begin{aligned}
\big(A(x_{n+1}-x_n),A(x-x_{n+1})\big)&=\frac{1}{2}\big(\|A(x-x_n)\|^2-\|A(x-x_{n+1})\|^2-\|A(x_n-x_{n+1})\|^2\big),\\
\big(Q(x_{n+1}-x_n),x-x_{n+1}\big)&=\frac{1}{2}\big(\|x-x_n\|_Q^2-\|x-x_{n+1}\|_Q^2-\|x_n-x_{n+1}\|_Q^2\big),\\
\big(\lambda_{n+1}-\lambda_n,\lambda-\lambda_{n+1}\big)&=\frac{1}{2}\big(\|\lambda-\lambda_n\|^2-\|\lambda-\lambda_{n+1}\|^2-\|\lambda_n-\lambda_{n+1}\|^2\big),
\end{aligned}$$

we have

$$\begin{aligned}
&\frac{\sigma_n}{2}\|A(x-x_n)\|^2+\frac{\sigma_n}{2}\|x-x_n\|_Q^2+\frac{1}{2\sigma_n}\|\lambda-\lambda_n\|^2\\
&+\frac{\sigma_n^2}{2\sigma_{n-1}^2\sigma_n}\|\lambda_n-\lambda_{n-1}\|^2+\frac{\sigma_n}{\sigma_{n-1}}\big(A(x-x_n),\lambda_n-\lambda_{n-1}\big)\\
\geq\ &\frac{\sigma_n}{2}\|A(x-x_{n+1})\|^2+\frac{\sigma_n}{2}\|x-x_{n+1}\|_Q^2+\frac{1+\gamma\sigma_n}{2\sigma_n}\|\lambda-\lambda_{n+1}\|^2\\
&+\frac{\sigma_n}{2}\|x_{n+1}-x_n\|_Q^2+\frac{1}{2\sigma_n}\|\lambda_{n+1}-\lambda_n\|^2+\big(A(x-x_{n+1}),\lambda_{n+1}-\lambda_n\big)\\
&+\big(\mathcal{L}(x_{n+1},\lambda)-\mathcal{L}(x,\lambda_{n+1})\big),\ \ \forall\,(x,\lambda)\in U.
\end{aligned}\tag{4.10}$$

Recall (2.3), (2.4) and (2.6). We have

$$\frac{1+\gamma\sigma_n}{\sigma_n}\geq\frac{1}{\theta_{n+1}\sigma_{n+1}},\qquad\sigma_n\theta_{n+1}=\sigma_{n+1}.\tag{4.11}$$

Then, it easily yields from (4.10) and the definition of $S_n(x,\lambda)$ in (4.8) that

$$S_n(x,\lambda)\geq\frac{1}{\theta_{n+1}}S_{n+1}(x,\lambda)+\big(\mathcal{L}(x_{n+1},\lambda)-\mathcal{L}(x,\lambda_{n+1})\big),\ \ \forall\,(x,\lambda)\in U.\tag{4.12}$$

The proof is complete. □

Now, we establish a worst-case $O(1/n^2)$ convergence rate in the ergodic sense for Algorithm 1.

Theorem 4.1 ($O(1/n^2)$ convergence rate in the ergodic sense)**.** *Let $\{(x_n, \lambda_n)\}$ be the sequence generated by Algorithm 1. Let*

$$T_n = \sum_{i=0}^{n} \frac{\sigma_0}{\sigma_i}, \quad \tilde{x}_n = \frac{1}{T_n} \sum_{i=0}^{n} \frac{\sigma_0}{\sigma_i} x_{i+1} \ \textit{and} \ \tilde{\lambda}_n = \frac{1}{T_n} \sum_{i=0}^{n} \frac{\sigma_0}{\sigma_i} \lambda_{i+1}.$$

Then, for any bounded subset $\mathcal{B}_1 \times \mathcal{B}_2$ of $U = \mathcal{X} \times \mathbb{R}^m$ containing the sequence $\{(\tilde{x}_n, \tilde{\lambda}_n)\}$, we have

$$\mathcal{G}_{\mathcal{B}_1 \times \mathcal{B}_2}(\tilde{x}_n, \tilde{\lambda}_n) \le c \frac{\kappa}{n^2} = O(\frac{1}{n^2}), \tag{4.13}$$

where $c > 0$ is a constant.

Proof. Multiplying the inequality (4.7) with index i in Lemma 4.2 by $\frac{\sigma_0}{\sigma_i}$, summing it over $i = 0, 1, \cdots, n$, and using the relationship $\sigma_i \theta_{i+1} = \sigma_{i+1}$ in (4.11), we obtain

$$\sum_{i=0}^{n} \frac{\sigma_0}{\sigma_i} \big(\mathcal{L}(x_{i+1}, \lambda) - \mathcal{L}(x, \lambda_{i+1})\big) + \sum_{i=0}^{n} \frac{\sigma_0}{\sigma_{i+1}} S_{i+1}(x, \lambda) \le \sum_{i=0}^{n} \frac{\sigma_0}{\sigma_i} S_i(x, \lambda), \ \ \forall\, (x, \lambda) \in U.$$

Because f and g are convex functions, we have

$$T_n \big(\mathcal{L}(\tilde{x}_n, \lambda) - \mathcal{L}(x, \tilde{\lambda}_n)\big) + \frac{\sigma_0}{\sigma_{n+1}} S_{n+1}(x, \lambda) \le S_0(x, \lambda), \ \ \forall\, (x, \lambda) \in U. \tag{4.14}$$

It follows from Lemma 3.2 that $\sigma_n \sim O(\kappa/n)$. With the definition of T_n, this immediately yields that $T_n \sim O(n^2/\kappa)$. Since $S_{n+1}(x, \lambda)$ is nonnegative, we have

$$\mathcal{L}(\tilde{x}_n, \lambda) - \mathcal{L}(x, \tilde{\lambda}_n) \le \frac{1}{T_n} S_0(x, \lambda) \le c \frac{\kappa}{n^2} S_0(x, \lambda), \ \ \forall\, (x, \lambda) \in U. \tag{4.15}$$

From the result of Theorem 3.1, we know that the sequence $\{(x_n, \lambda_n)\}$ is bounded, then its linear average $(\tilde{x}_n, \tilde{\lambda}_n)$ is also bounded. Therefore, for some open bounded subset $\mathcal{B}_1 \times \mathcal{B}_2$ of U containing the sequence $\{(\tilde{x}_n, \tilde{\lambda}_n)\}$, $S_0(x, \lambda)$ is bounded over $\mathcal{B}_1 \times \mathcal{B}_2$. Then, taking the maximum of both sides of (4.15) with (x, λ) over $\mathcal{B}_1 \times \mathcal{B}_2$ and recalling the definition of $\mathcal{G}_{\mathcal{B}_1 \times \mathcal{B}_2}(x, \lambda)$ in (4.2) give us

$$\mathcal{G}_{\mathcal{B}_1 \times \mathcal{B}_2}(\tilde{x}_n, \tilde{\lambda}_n) = \max_{(x,\lambda) \in \mathcal{B}_1 \times \mathcal{B}_2} \Big\{ \mathcal{L}(\tilde{x}_n, \lambda) - \mathcal{L}(x, \tilde{\lambda}_n) \Big\} \le c \frac{\kappa}{n^2} = O(\frac{1}{n^2}).$$

The proof is complete. □

The assertion (4.13) means that $(\tilde{x}_n, \tilde{\lambda}_n)$ calculated by n iterations of Algorithm 1 is an approximate solution of the saddle-point reformulation (4.1) with an accuracy of $O(1/n^2)$. Therefore, a worst-case $O(1/n^2)$ convergence rate in the ergodic sense is established for Algorithm 1. Moreover, we can obtain a stronger convergence rate for the sequence of $\{\lambda_n\}$ in terms of the proximity to the optimal value λ^* in the following theorem.

Theorem 4.2 (Convergence rate of the dual variable)**.** *Let (x^*, λ^*) be a solution point of the saddle-point reformulation (4.1), and $\{(x_n, \lambda_n)\}$ be generated by Algorithm 1. Then, we have*

$$\|\lambda_{n+1} - \lambda^*\|^2 \le \frac{2\sigma_{n+1}^2}{\sigma_0} S_0(x^*, \lambda^*) = O(\frac{\kappa^2}{n^2}). \tag{4.16}$$

Proof. Taking $(x, \lambda) = (x^*, \lambda^*)$ in (4.14), and recalling the fact

$$\mathcal{L}(\tilde{x}_n, \lambda^*) - \mathcal{L}(x^*, \tilde{\lambda}_n) \geq 0,$$

we have

$$\frac{\sigma_0}{\sigma_{n+1}} S_{n+1}(x^*, \lambda^*) \leq S_0(x^*, \lambda^*).$$

It follows from the definition of S_n in (4.8) that

$$\begin{aligned}
S_{n+1}(x^*, \lambda^*) &= \frac{\sigma_{n+1}}{2}\|A(x^* - x_{n+1})\|^2 + \frac{\sigma_{n+1}}{2}\|x^* - x_{n+1}\|_Q^2 + \frac{1}{2\sigma_{n+1}}\|\lambda^* - \lambda_{n+1}\|^2 \\
&\quad + \frac{\theta_{n+1}^2}{2\sigma_{n+1}}\|\lambda_{n+1} - \lambda_n\|^2 + \theta_{n+1}\big(A(x^* - x_{n+1}), \lambda_{n+1} - \lambda_n\big) \\
&= \frac{1}{2}\Big\|\sqrt{\sigma_{n+1}}A(x^* - x_{n+1}) + \frac{\theta_{n+1}}{\sqrt{\sigma_{n+1}}}(\lambda_{n+1} - \lambda_n)\Big\|^2 \\
&\quad + \frac{\sigma_{n+1}}{2}\|x^* - x_{n+1}\|_Q^2 + \frac{1}{2\sigma_{n+1}}\|\lambda^* - \lambda_{n+1}\|^2 \\
&\geq \frac{1}{2\sigma_{n+1}}\|\lambda^* - \lambda_{n+1}\|^2.
\end{aligned}$$

Therefore, the result (4.16) follows from the above two inequalities and Lemma 3.2. □

5 Connection with Primal-Dual Algorithms

In this section, we elaborate on the connection between Algorithm 1 with the primal-dual algorithm proposed in [4] and a variant of it for the special case of (1.1) with $\mathcal{X} = \mathbb{R}^d$. Recall the relationship between the primal-dual algorithm (1.9) and the linearized version of ADMM for the dual problem of (1.1) with $\mathcal{X} = \mathbb{R}^d$.

5.1 (x-λ-λ) Primal-Dual Scheme

First, we consider a variant version of (1.9) which updates the dual variable λ twice at each iteration and thus is called the (x-λ-λ) scheme

$$\text{(5.1)} \qquad \begin{cases} x_{n+1} = \operatorname{prox}_{\tau_n f}\big(x_n - \tau_n A^T\tilde{\lambda}_n\big) \\ \lambda_{n+1} = \operatorname{prox}_{\sigma_n g^*}\big(\lambda_n + \sigma_n A x_{n+1}\big) \\ \tilde{\lambda}_{n+1} = \lambda_{n+1} + \theta_{n+1}(\lambda_{n+1} - \lambda_n). \end{cases}$$

First, using the Moreau's identity in [35]:

$$\operatorname{prox}_{\sigma g^*}(\lambda) + \sigma \operatorname{prox}_{g/\sigma}(\lambda/\sigma) = \lambda,$$

we can reformulate the λ-subproblem in (5.1) as

$$\text{(5.2)} \qquad \lambda_{n+1} = \lambda_n + \sigma_n(Ax_{n+1} - y_{n+1}),$$

where

$$y_{n+1} = \operatorname*{argmin}_y \Big\{g(y) + \frac{\sigma_n}{2}\|y - Ax_{n+1} - \frac{\lambda_n}{\sigma_n}\|^2\Big\}.$$

Second, by the definition of proximal operator in (1.5) and the relationships $\tilde{\lambda}_n = \lambda_n + \theta_n(\lambda_n - \lambda_{n-1})$ in (5.1) and $\lambda_n - \lambda_{n-1} = \sigma_{n-1}(Ax_n - y_n)$ in (5.2), the x-subproblem in (5.1) can be rewritten as

$$\begin{aligned}
x_{n+1} &= \operatorname{prox}_{\tau_n f}\big(x_n - \tau_n A^T\tilde{\lambda}_n\big)\\
&= \underset{x}{\operatorname{argmin}}\left\{f(x) + \frac{1}{2\tau_n}\big\|x - x_n + \tau_n A^T\tilde{\lambda}_n\big\|^2\right\}\\
&= \underset{x}{\operatorname{argmin}}\left\{f(x) + \frac{1}{2\tau_n}\|x - x_n\|^2 + \big(A(x-x_n), \tilde{\lambda}_n\big) + \frac{\tau_n}{2}\|A^T\tilde{\lambda}_n\|^2\right\}\\
&= \underset{x}{\operatorname{argmin}}\left\{\begin{array}{c} f(x) + \frac{1}{2\tau_n}\|x-x_n\|^2 - \frac{\sigma_n}{2}\|A(x-x_n)\|^2 + \frac{\sigma_n}{2}\|A(x-x_n)\|^2\\ +\big(A(x-x_n),\tilde{\lambda}_n\big) + \frac{1}{2\sigma_n}\|\tilde{\lambda}_n\|^2 - \frac{1}{2\sigma_n}\|\tilde{\lambda}_n\|^2 + \frac{\tau_n}{2}\|A^T\tilde{\lambda}_n\|^2\end{array}\right\}\\
&= \underset{x}{\operatorname{argmin}}\left\{\begin{array}{c} f(x) + \frac{\sigma_n}{2}\big\|A(x-x_n) + \frac{\tilde{\lambda}_n}{\sigma_n}\big\|^2\\ +\frac{1}{2}\|x-x_n\|^2_{\tau_n^{-1}I - \sigma_n A^TA} + \frac{\tau_n}{2}\|A^T\tilde{\lambda}_n\|^2 - \frac{1}{2\sigma_n}\|\tilde{\lambda}_n\|^2\end{array}\right\}\\
&= \underset{x}{\operatorname{argmin}}\left\{\begin{array}{c} f(x) + \frac{\sigma_n}{2}\big\|Ax - y_n + (1-\vartheta_n)\frac{\lambda_n}{\sigma_n} + \vartheta_n\frac{\lambda_{n-1}}{\sigma_n}\big\|^2\\ +\frac{1}{2}\|x-x_n\|^2_{\tau_n^{-1}I-\sigma_nA^TA} + C_n\end{array}\right\},
\end{aligned}$$

where

$$\vartheta_n := \frac{\sigma_n - \theta_n\sigma_{n-1}}{\sigma_{n-1}} \text{ and } C_n = \frac{\tau_n}{2}\big\|A^T\tilde{\lambda}_n\big\|^2 - \frac{1}{2\sigma_n}\big\|\tilde{\lambda}_n\big\|^2.$$

Thus, the primal-dual algorithm (5.1) is equivalent to the following scheme:

$$\text{(5.3)}\qquad \begin{cases} x_{n+1} = \underset{x}{\operatorname{argmin}}\left\{\begin{array}{c} f(x) + \frac{\sigma_n}{2}\big\|Ax - y_n + (1-\vartheta_n)\frac{\lambda_n}{\sigma_n} + \vartheta_n\frac{\lambda_{n-1}}{\sigma_n}\big\|^2\\ +\frac{1}{2}\|x-x_n\|^2_{\tau_n^{-1}I-\sigma_nA^TA}\end{array}\right\}\\ y_{n+1} = \underset{y}{\operatorname{argmin}}\Big\{g(y) + \dfrac{\sigma_n}{2}\big\|Ax_{n+1} - y + \dfrac{\lambda_n}{\sigma_n}\big\|^2\Big\}\\ \lambda_{n+1} = \lambda_n + \sigma_n(Ax_{n+1} - y_{n+1}),\end{cases}$$

which can be viewed as a linearized version of the ADMM with varying penalty parameters

$$\text{(5.4)}\qquad \begin{cases} x_{n+1} = \underset{x}{\operatorname{argmin}}\Big\{f(x) + \dfrac{\sigma_n}{2}\big\|Ax - y_n + (1-\vartheta_n)\dfrac{\lambda_n}{\sigma_n} + \vartheta_n\dfrac{\lambda_{n-1}}{\sigma_n}\big\|^2\Big\}\\ y_{n+1} = \underset{y}{\operatorname{argmin}}\Big\{g(y) + \dfrac{\sigma_n}{2}\big\|Ax_{n+1} - y + \dfrac{\lambda_n}{\sigma_n}\big\|^2\Big\}\\ \lambda_{n+1} = \lambda_n + \sigma_n(Ax_{n+1} - y_{n+1}),\end{cases}$$

whose x-subproblem is proximally regularized by the term $\frac{1}{2}\|x - x_n\|^2_{\tau_n^{-1}I-\sigma_nA^TA}$.

Furthermore, $\vartheta_n = 0$ if $\theta_n = \sigma_n/\sigma_{n-1}$. Therefore, the scheme (5.4) can be simplified as

$$\text{(5.5)}\qquad \begin{cases} x_{n+1} = \underset{x}{\operatorname{argmin}}\Big\{f(x) + \dfrac{\sigma_n}{2}\big\|Ax - y_n + \dfrac{\lambda_n}{\sigma_n}\big\|^2\Big\}\\ y_{n+1} = \underset{y}{\operatorname{argmin}}\Big\{g(y) + \dfrac{\sigma_n}{2}\big\|Ax_{n+1} - y + \dfrac{\lambda_n}{\sigma_n}\big\|^2\Big\}\\ \lambda_{n+1} = \lambda_n + \sigma_n(Ax_{n+1} - y_{n+1}).\end{cases}$$

This is precisely the application of the standard ADMM scheme (1.3) with varying penalty parameters σ_n to (1.1). Therefore, the conclusion is that if $\sigma_n = \theta_n\sigma_{n-1}$ is satisfied, then the primal-dual algorithm (5.1) is the case of Algorithm 1 with

$$Q = (\tau_n\sigma_n)^{-1}I - A^TA.$$

This connection can be regarded as a generalization of the elaboration in [36] on these two methods with constant parameters.

5.2 (λ-x-x) Primal-Dual Scheme

With a similar analysis, we can also show that the scheme (1.9) with $\theta_n = \tau_n/\tau_{n-1}$ is equivalent to a linearized version of the ADMM for the dual problem (5.10) of model (1.1).

Different from the (x-λ-λ) scheme in (5.1), the accelerated scheme (1.9)

$$\begin{cases} \lambda_{n+1} = \mathrm{prox}_{\sigma_n g^*}\big(\lambda_n + \sigma_n A\tilde{x}_n\big) \\ x_{n+1} = \mathrm{prox}_{\tau_n f}\big(x_n - \tau_n A^T\lambda_{n+1}\big) \\ \tilde{x}_{n+1} = x_{n+1} + \theta_{n+1}(x_{n+1} - x_n) \end{cases}$$

discussed in [4] performs iterations in order of (λ-x-x).

Taking $x = x_n - \tau_n A^T\lambda_{n+1}$ and $\tau = \tau_n$ in the Moreau's identity (see, e.g. [35]):

$$\mathrm{prox}_{\tau f}(x) + \tau\mathrm{prox}_{f^*/\tau}(x/\tau) = x,$$

we obtain

$$x_{n+1} = x_n - \tau_n(A^T\lambda_{n+1} + z_{n+1}), \tag{5.6}$$

where

$$z_{n+1} = \mathrm{prox}_{f^*/\tau_n}\Big(\frac{x_n}{\tau_n} - A^T\lambda_{n+1}\Big) = \operatorname*{argmin}_z\Big\{f^*(z) + \frac{\tau_n}{2}\Big\|A^T\lambda_{n+1} + z - \frac{x_n}{\tau_n}\Big\|^2\Big\}.$$

For the λ-subproblem, according to the definition of the proximity operator in (1.5), the relationships $\tilde{x}_n = x_n + \theta_n(x_n - x_{n-1})$ in (1.9) and $x_n - x_{n-1} = -\tau_{n-1}(A^T\lambda_n + z_n)$ in (5.6), we have

$$\begin{aligned} \lambda_{n+1} &= \mathrm{prox}_{\sigma_n g^*}\big(\lambda_n + \sigma_n A\tilde{x}_n\big) \\ &= \operatorname*{argmin}_\lambda\Big\{g^*(\lambda) + \frac{1}{2\sigma_n}\big\|\lambda - \lambda_n - \sigma_n A\tilde{x}_n\big\|^2\Big\} \\ &= \operatorname*{argmin}_\lambda\Big\{g^*(\lambda) + \frac{\tau_n}{2}\Big\|A^T(\lambda - \lambda_n) - \frac{\tilde{x}_n}{\tau_n}\Big\|^2 + \frac{1}{2}\|\lambda - \lambda_n\|^2_{\sigma_n^{-1}I - \tau_n AA^T} + \tilde{C}_n\Big\} \\ &= \operatorname*{argmin}_\lambda\left\{\begin{array}{c} g^*(\lambda) + \frac{\tau_n}{2}\Big\|A^T\lambda + z_n - (1-\delta_n)\frac{x_n}{\tau_n} - \delta_n\frac{x_{n-1}}{\tau_n}\Big\|^2 \\ +\frac{1}{2}\|\lambda - \lambda_n\|^2_{\sigma_n^{-1}I - \tau_n AA^T} \end{array}\right\}, \end{aligned}$$

where

$$\delta_n = \frac{\tau_n - \theta_n\tau_{n-1}}{\tau_{n-1}} \quad \text{and} \quad \tilde{C}_n = \frac{\sigma_n}{2}\|A\tilde{x}_n\|^2 - \frac{1}{2\tau_n}\|\tilde{x}_n\|^2.$$

With the above discussions, we can derive that the scheme (1.9) is equivalent to

$$\begin{cases} \lambda_{n+1} = \operatorname*{argmin}_\lambda\left\{\begin{array}{c} g^*(\lambda) + \frac{\tau_n}{2}\Big\|A^T\lambda + z_n - (1-\delta_n)\frac{x_n}{\tau_n} - \delta_n\frac{x_{n-1}}{\tau_n}\Big\|^2 \\ +\frac{1}{2}\|\lambda - \lambda_n\|^2_{\sigma_n^{-1}I - \tau_n AA^T} \end{array}\right\} \\ z_{n+1} = \operatorname*{argmin}_z\Big\{f^*(z) + \frac{\tau_n}{2}\Big\|A^T\lambda_{n+1} + z - \frac{x_n}{\tau_n}\Big\|^2\Big\} \\ x_{n+1} = x_n - \tau_n(A^T\lambda_{n+1} + z_{n+1}). \end{cases} \tag{5.7}$$

If $\theta_n = \tau_n/\tau_{n-1}$, then (5.7) can be simplified as
(5.8)

$$\begin{cases} \lambda_{n+1} = \underset{\lambda}{\operatorname{argmin}} \Big\{ g^*(\lambda) + \frac{\tau_n}{2} \big\| A^T\lambda + z_n - \frac{x_n}{\tau_n} \big\|^2 + \frac{1}{2} \|\lambda - \lambda_n\|^2_{\sigma_n^{-1} I - \tau_n A A^T} \Big\} \\ z_{n+1} = \underset{z}{\operatorname{argmin}} \Big\{ f^*(z) + \frac{\tau_n}{2} \big\| A^T\lambda_{n+1} + z - \frac{x_n}{\tau_n} \big\|^2 \Big\} \\ x_{n+1} = x_n - \tau_n(A^T\lambda_{n+1} + z_{n+1}), \end{cases}$$

which is an application of the linearized version of the ADMM with varying penalty parameters τ_n to the problem

$$\begin{aligned} \min_{\lambda \in \mathbb{R}^m, z \in \mathbb{R}^d} \quad & f^*(z) + g^*(\lambda) \\ \text{s.t.} \quad & A^T\lambda + z = 0. \end{aligned} \tag{5.9}$$

Note that the dual problem of (1.1) with $\mathcal{X} = \mathbb{R}^d$ can be written as

$$\max_{\lambda \in \mathbb{R}^m} \Big\{ - \big(f^*(-A^T\lambda) + g^*(\lambda) \big) \Big\}. \tag{5.10}$$

Thus, the problem (5.9) is equivalent to the dual problem (5.10) of (1.1) by introducing variable $z = -A^T\lambda$ and regarding x in (5.8) as the Lagrange multiplier of the constraint in (5.9).

Note that (5.3) and (5.7) are not equivalent to each other with non-identity matrix A and varying parameters τ_n, σ_n.

Remark 5.1. *The primal-dual scheme* (1.9) *discussed in [4] with iterations in order of (λ-x-x) corresponds to a linearized version of the ADMM for the dual problem* (5.10) *of model* (1.1)*, while the so called (x-λ-λ) scheme* (5.1) *corresponds to a linearized version of the ADMM for the primal problem* (1.1)*. Therefore, the (x-λ-λ) scheme* (5.1) *is different from the (λ-x-x) scheme* (1.9) *discussed in [4].*

6 Numerical Results

As mentioned, the ADMM has found many applications in a broad spectrum of areas. In this section, we take the LASSO model in [39] as an illustrative example to numerically verify the efficiency of the proposed faster ADMM. The LASSO model is probably the simplest application representative of those to which the ADMM has been successfully applied. All the codes were written by MATLAB R2012a and all experiments were performed on a desktop with Windows 8 system and an Intel(R) Core(TM) i5-4570s CPU processor (2.9GHz) with a 8GB memory.

6.1 Experiment Setup

The LASSO model in [39] is

$$\min_x \Big\{ \alpha\|x\|_1 + \frac{1}{2}\|Dx - c\|^2 \Big\}, \tag{6.1}$$

where $\|x\|_1 := \sum_{i=1}^d |x_i|$, $D \in \mathbb{R}^{l\times d}$ is a design matrix usually with $l \ll d$, l is the number of data points, d is the number of features, $c \in \mathbb{R}^l$ is the response vector and $\alpha > 0$ is a regularization parameter. The LASSO model provides a sparse estimation of x when there are more features than data points. It can also be explained as a model for finding a sparse solution of the under-determined system of linear equations $Dx = c$.

Obviously, the model (6.1) can be rewritten as

$$\begin{aligned} &\min_{x,y} \ \ \alpha\|x\|_1 + \frac{1}{2}\|Dy - c\|^2 \\ &\ \text{s.t.} \ \ x - y = 0, \end{aligned} \tag{6.2}$$

which is a special case of model (1.2) with $f(x) = \alpha\|x\|_1$, $g(y) = \frac{1}{2}\|Dy - c\|^2$ and $m = d$, $\mathcal{X} = \mathbb{R}^d$, $A = I_{d\times d}$. For (6.2), it is easy to see that the assumptions posed in Section 2 are satisfied. Particularly, the Lipschitz continuity constant of ∇g is $\|D^TD\|$, then $\gamma = 1/\|D^TD\|$ is set in (2.4) for Algorithm 1 as γ denotes the inverse of the Lipschitz continuity constant of ∇g. Thus, applying the proposed faster ADMM (2.2) with $Q = 0$ to (6.2), we obtain the scheme

$$\begin{cases} x_{n+1} = \underset{x}{\operatorname{argmin}}\Big\{\alpha\|x\|_1 + \dfrac{\sigma_n}{2}\|x - y_n + \dfrac{\lambda_n}{\sigma_n}\|^2\Big\} \\ y_{n+1} = \underset{y}{\operatorname{argmin}}\Big\{\dfrac{1}{2}\|Dy - c\|^2 + \dfrac{\sigma_n}{2}\|x_{n+1} - y + \dfrac{\lambda_n}{\sigma_n}\|^2\Big\} \\ \lambda_{n+1} = \lambda_n + \sigma_n(x_{n+1} - y_{n+1}) \end{cases} \tag{6.3}$$

where the penalty parameter $\{\sigma_n\}$ is updated by

$$\sigma_{n+1} = \tilde{\sigma}_{\lfloor \frac{n}{\kappa} \rfloor}$$

with κ being a given integer and

$$\tilde{\sigma}_{s+1} = \frac{\tilde{\sigma}_s}{\sqrt{1 + \tilde{\sigma}_s/\|D^TD\|}},$$

starting from a given $\tilde{\sigma}_0 = \sigma_0$. As mentioned in many literatures, the x-subproblem in (6.3) has its closed-form solution given by

$$x_{n+1} = S_{\alpha/\sigma_n}(y_n - \frac{\lambda_n}{\sigma_n}),$$

where $S_\delta(x)$ is the soft-thresholding operator [10] defined as

$$(S_\delta(x))_i = (1 - \delta/|x_i|)_+ \cdot x_i, \ \ i = 1, 2, \cdots, d,$$

and the y-subproblem has its solution given by

$$y_{n+1} = (\sigma_n I + D^TD)^{-1}(\sigma_n x_{n+1} + \lambda_n + D^Tc).$$

To implement the scheme (6.3) and avoid loss of efficiency possibly caused by coding skills, we use the widely-used MATLAB ADMM package downloaded at `http://web.stanford.edu/~boyd/papers/admm/`, with the only slight revision of using an adaptive penalty parameter. We use the stopping criterion in [14]

$$\operatorname{dist}_\infty\Big(0, \partial_x\Big[\alpha\|x\|_1 + \frac{1}{2}\|Dx - c\|^2\Big]_{x=x_n}\Big) \le \varepsilon, \tag{6.4}$$

where $\operatorname{dist}_\infty(t, S) := \inf\{\|t - s\|_\infty \mid s \in S\}$, and $\varepsilon > 0$ is a tolerance.

6.2 Synthetic Data

In this experiment, we specify the LASSO model (6.1) with $l = 1500$ and $d = 5000$; the matrix D in (6.1) is generated by the MATLAB function `randn` with 1500 by 5000 entries; all columns are normalized afterwards; the sparse vector $x \in \mathbb{R}^{5000}$ is generated by the MATLAB function `sprandn` with 100 nonzero entries; the vector c is set as $Dx + \eta$ with noise vector $\eta \sim N(0, 0.001)$; the regularization parameter α is set as $\|D^T c\|_\infty/10$.

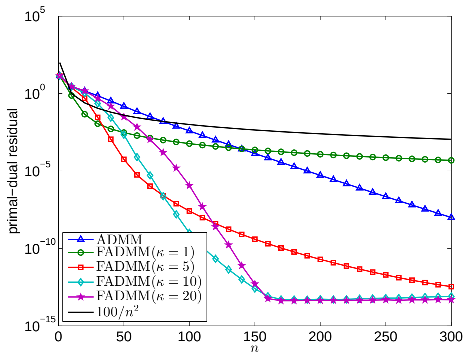
We use the original ADMM (1.3) with a constant penalty parameter as the benchmark to verify the $O(1/n^2)$ convergence rate of Algorithm 1. We test different constant penalty parameters for the original ADMM (1.3) and Algorithm 1 also starting from the same constant for each comparison. In Figure 1, we plot the evaluation of the primal-dual residual $\max\left\{\sigma_n\|y_{n+1} - y_n\|, \frac{1}{\sigma_n}\|\lambda_{n+1} - \lambda_n\|\right\}$ from (3.3) taking $A := I$ and $Q := \mathbf{0}$ for LASSO model (6.1) with respect to the iteration number for the original ADMM (1.3) with a constant penalty parameter σ_0 (denoted by "ADMM") and Algorithm 1 with varying penalty parameter starting from the same σ_0 (denoted by "FADMM"). We report the results for $\sigma_0 = 10, 50, \cdots$, up to 1000. We have tested a number of other cases of σ_0 and the comparison is similar except for some extreme cases where σ_0 is very small. For Algorithm 1, four choices of $\kappa = 1, 5, 10, 20$ are tested. For each case, we plot the evolution of the primal-dual residual in Figure 1. These curves show clearly that Algorithm 1 converges absolutely faster than the original ADMM (1.3) with a given constant parameter. To discern the actual rate more clearly, we also set a benchmark rate of $100/n^2$ and plot its decay. As we can see, the speed of the decay of the primal-dual residuals is even faster than the benchmark rate of $100/n^2$; so the established $O(1/n^2)$ convergence rate is just a worse-case estimate of the speed and we can easily witness faster speed empirically. Moreover, the values of the objective function in each iteration are plotted in Figure 2, where the decay of the objective function by Algorithm 1 is also much faster than that of the original ADMM, especially when the initial penalty parameter is large.

In Table 1, we compare the iteration numbers of these two ADMM schemes for some choices of σ_0 and different tolerance in the stopping criterion (6.4). In this table, "–" means the stopping criterion (6.4) is not satisfied after 5000 iterations. These data show that the original ADMM (1.3) with a given constant penalty parameter can easily fail, especially when producing high-precision solutions; while Algorithm 1 usually performs very well except for the extreme case where $\kappa = 1$. Generally, we do not suggest choosing $\kappa = 1$ if the Lipschitz constant of ∇g is small.

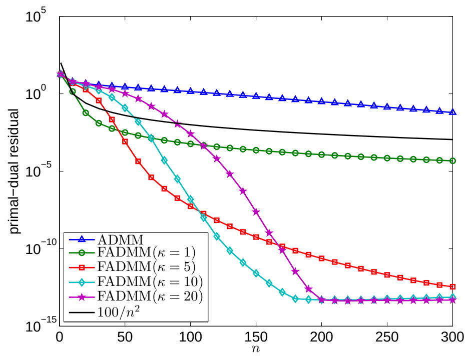
In Figure 3, we test the sensitivity to the initial value σ_0 of Algorithm 1. We only report the results when $\kappa = 10$ for succinctness. For different cases of the tolerance ϵ in the stopping criterion (6.4), we plot the iteration numbers with respect to different choices of σ_0 whose values vary from 10^0 to 10^3 with an equal exponential distance of 0.1, where the horizontal axis is in logarithmic scale. It is clearly demonstrated that Algorithm 1 is much more robust to the value of the initial penalty parameter σ_0 especially when it is larger equal than 10.0, even when high-precision solutions are produced. This is a significant advantage for implementing ADMM-type algorithms.

6.3 Real Datasets

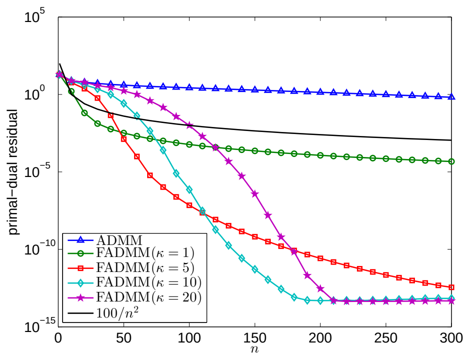
In this subsection, we test the proposed Algorithm 1 for the LASSO model (6.1) in which the matrix D and vector c are given by the following six real gene microarray datasets: GLIOMA, LEU, LUNG, MLL, PROSTATE and SRBCT. These datasets are available in MATLAB format at the website `http://www.biomedcentral.com/1471-2105/7/228/additional/`. The sample and feature numbers of these datasets are listed in Table 2; more details can be found in, e.g., [42]. All rows of the matrix D are normalized in our experiments, and the regularization parameter α is set as $\|D^T c\|_\infty/10$.



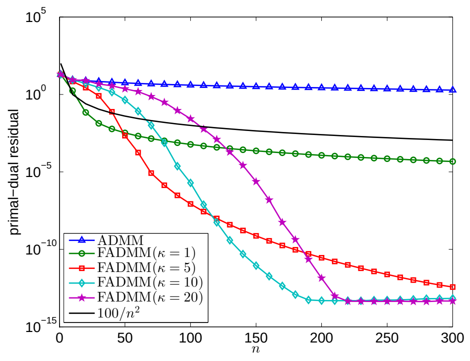
(a) Initial $\sigma_0 = 10.0$



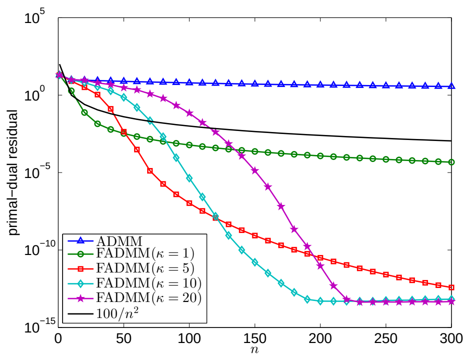
(b) Initial $\sigma_0 = 50.0$



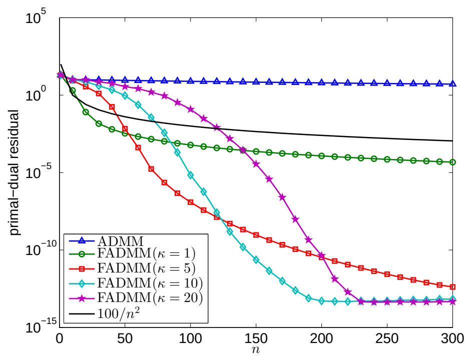
(c) Initial $\sigma_0 = 100.0$



(d) Initial $\sigma_0 = 200.0$



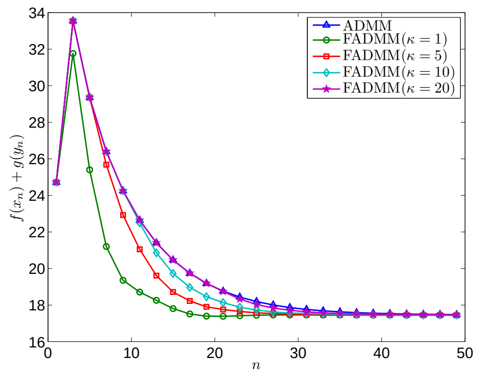
(e) Initial $\sigma_0 = 500.0$



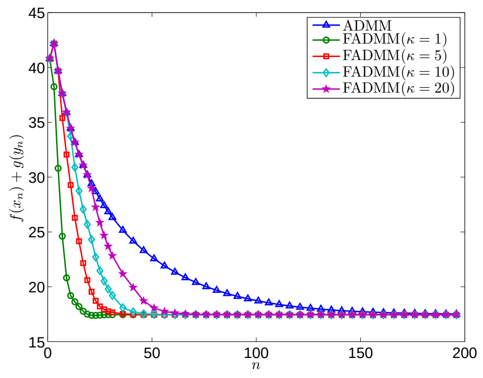
(f) Initial $\sigma_0 = 1000.0$

Figure 1: The decay of the primal-dual residual for LASSO model with the synthetic data generated in Section 6.2 by ADMM and faster ADMM with $\kappa = 1, 5, 10, 20$ and different initial penalty parameter σ_0.

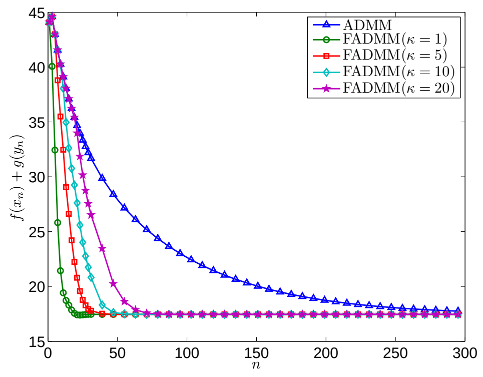
We compare the iteration numbers of the original ADMM (1.3) with a constant penalty parameter σ_0 (denoted by "ADMM") and Algorithm 1 with varying penalty parameter starting from the same σ_0 (denoted by "FADMM"), four choices of $\kappa = 1, 2, 5, 10$ are tested for Algorithm 1. The numerical results for these real datasets are listed in Tables 3-8, respectively, with $\sigma_0 = 1.0, 5.0, 10.0, 20.0, 50.0, 100.0, 200.0, 500.0, 1000.0$ and



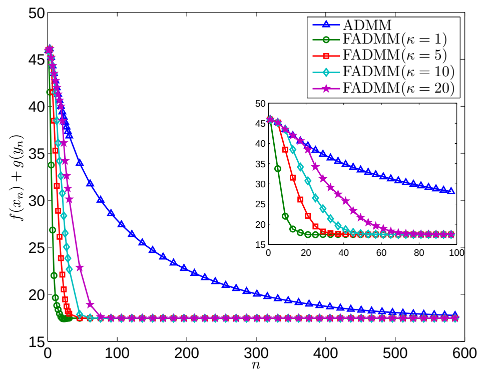
(g) Initial $\sigma_0 = 10.0$



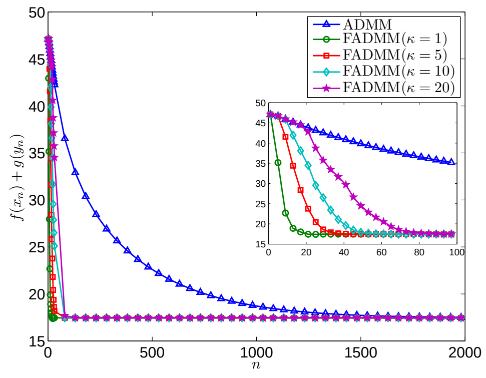
(h) Initial $\sigma_0 = 50.0$



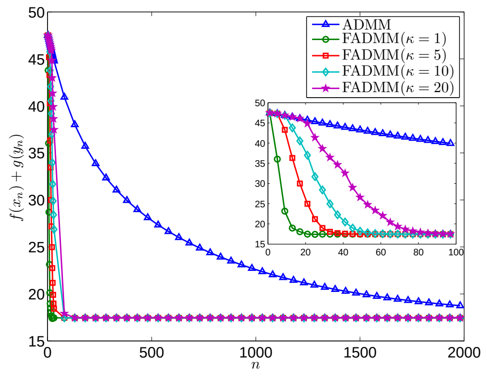
(i) Initial $\sigma_0 = 100.0$



(j) Initial $\sigma_0 = 200.0$



(k) Initial $\sigma_0 = 500.0$



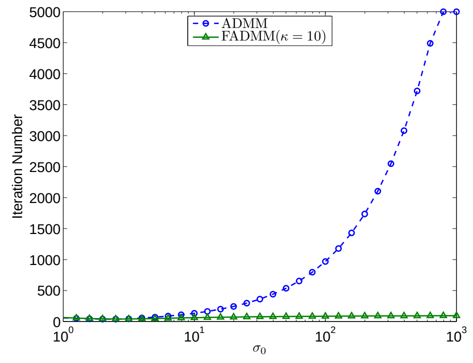
(l) Initial $\sigma_0 = 1000.0$

Figure 2: The decay of the values of the objective function of LASSO model with the synthetic data generated in Section 6.2 by ADMM and faster ADMM with $\kappa = 1, 5, 10, 20$ and different initial penalty parameter σ_0.

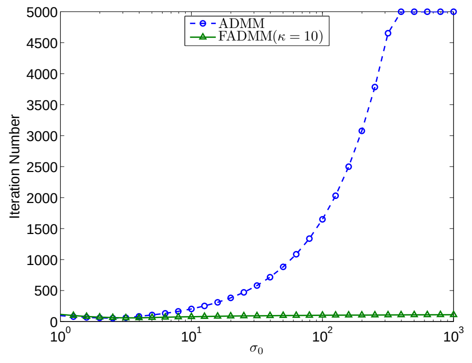
$\varepsilon = 10^{-3}, 10^{-4}, 10^{-5}$. The results in these tables show the same observations as those mentioned in Section 6.2: Algorithm 1 performs faster than the original ADMM (1.3) and it is more robust with respect to the initial value of σ_0.

Table 1: Iteration numbers of ADMM and faster ADMM for solving LASSO model with the synthetic data generated in Section 6.2 with different initial parameter σ_0 and different tolerance ε in the stopping criterion (6.4). ("–" means the iteration number is beyond 5000)

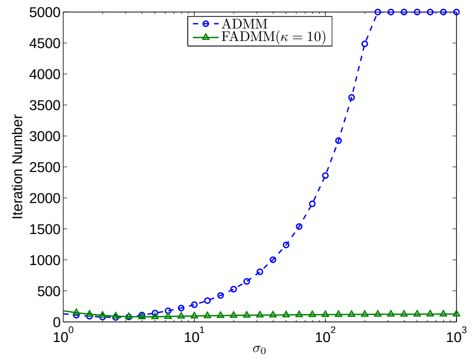
ε	10^{-4}	10^{-6}	10^{-8}	10^{-4}	10^{-6}	10^{-8}	10^{-4}	10^{-6}	10^{-8}	10^{-4}	10^{-6}	10^{-8}
	$\sigma_0 = 1.0$			$\sigma_0 = 10.0$			$\sigma_0 = 20.0$			$\sigma_0 = 50.0$		
ADMM	50	84	121	134	204	277	268	407	553	665	1010	1373
FADMM($\kappa = 1$)	276	2232	"–"	150	1133	"–"	149	1117	"–"	150	1112	"–"
FADMM($\kappa = 5$)	65	138	256	44	64	100	49	68	103	53	72	107
FADMM($\kappa = 10$)	56	106	170	54	71	86	64	81	96	72	89	105
FADMM($\kappa = 20$)	52	93	140	74	93	109	93	113	129	110	130	147
	$\sigma_0 = 100.0$			$\sigma_0 = 200.0$			$\sigma_0 = 500.0$			$\sigma_0 = 1000.0$		
ADMM	1326	2015	2739	2647	4023	"–"	"–"	"–"	"–"	"–"	"–"	"–"
FADMM($\kappa = 1$)	149	1107	"–"	150	1110	"–"	150	1111	"–"	150	1109	"–"
FADMM($\kappa = 5$)	56	74	109	57	76	110	59	78	112	60	79	113
FADMM($\kappa = 10$)	77	93	110	81	97	113	84	101	116	86	103	119
FADMM($\kappa = 20$)	119	140	156	125	146	163	133	153	170	137	158	174



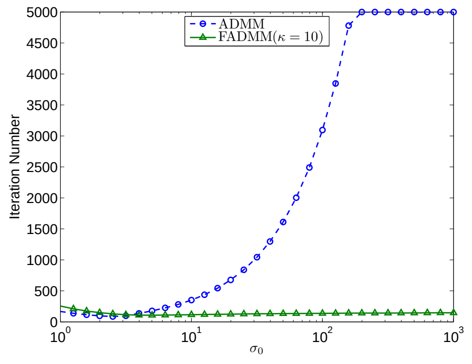
(a) $\varepsilon = 1.0 \times 10^{-6}$



(b) $\varepsilon = 1.0 \times 10^{-8}$



(c) $\varepsilon = 1.0 \times 10^{-10}$



(d) $\varepsilon = 1.0 \times 10^{-12}$

Figure 3: The iteration numbers of ADMM and faster ADMM with $\kappa = 10$ and different initial parameter σ_0 for solving LASSO model with the synthetic data generated in Section 6.2 with different tolerance ε in the stopping criterion (6.4).

6.4 Conclusions

In the literature, it is well known that the efficiency of the original ADMM (1.3) with a constant penalty parameter heavily depends on the value of this parameter and it seems we still lack of any general strategy to tune this constant. This can also be seen from the

Table 2: The informations about the six gene microarray datasets.

Dataset	GLIOMA	LEU	LUNG	MLL	PROSTATE	SRBCT
Samples	50	72	203	57	102	83
Gene features	4434	3571	3312	5848	5966	2308

Table 3: GLIOMA dataset: iteration numbers of ADMM and faster ADMM. ("–" means the iteration number is beyond 5000)

	ADMM			FADMM($\kappa = 1$)			FADMM($\kappa = 2$)			FADMM($\kappa = 5$)			FADMM($\kappa = 10$)		
σ_0 \ ε	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}
1.0	21	1286	"–"	22	467	1879	21	555	2677	21	743	4069	21	892	"–"
5.0	28	"–"	"–"	11	535	1954	14	687	2830	10	1037	4441	17	1418	"–"
10.0	78	"–"	"–"	15	545	1964	29	707	2850	47	1085	4492	58	1511	"–"
20.0	169	"–"	"–"	20	550	1970	37	717	2860	64	1111	4518	87	1561	"–"
50.0	426	"–"	"–"	23	553	1973	43	724	2867	79	1128	4535	114	1595	"–"
100.0	851	"–"	"–"	24	555	1974	45	726	2870	85	1134	4541	126	1608	"–"
200.0	1700	"–"	"–"	25	556	1975	47	728	2871	89	1138	4545	134	1616	"–"
500.0	4249	"–"	"–"	25	556	1976	48	729	2873	92	1142	4549	141	1623	"–"
1000.0	"–"	"–"	"–"	26	557	1976	49	730	2874	94	1144	4551	144	1627	"–"

Table 4: LEU dataset: iteration numbers of ADMM and faster ADMM. ("–" means the iteration number is beyond 5000)

	ADMM			FADMM($\kappa = 1$)			FADMM($\kappa = 2$)			FADMM($\kappa = 5$)			FADMM($\kappa = 10$)		
σ_0 \ ε	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}
1.0	39	175	2795	29	117	602	29	137	839	39	153	1225	39	161	1556
5.0	158	763	"–"	61	191	677	80	227	984	107	308	1561	124	411	2173
10.0	315	1524	"–"	70	182	687	96	245	1004	139	350	1610	175	488	2270
20.0	629	3048	"–"	74	187	692	105	255	1015	160	375	1636	214	535	2322
50.0	1569	"–"	"–"	78	209	696	111	262	1022	176	391	1654	244	567	2356
100.0	3137	"–"	"–"	79	192	697	114	265	1024	182	398	1660	257	580	2370
200.0	"–"	"–"	"–"	80	211	698	116	266	1026	186	402	1664	265	589	2378
500.0	"–"	"–"	"–"	81	212	698	117	268	1028	190	405	1668	272	596	2385
1000.0	"–"	"–"	"–"	81	194	699	118	269	1028	192	407	1670	276	599	2389

Table 5: LUNG dataset: iteration numbers of ADMM and faster ADMM. ("–" means the iteration number is beyond 5000)

	ADMM			FADMM($\kappa = 1$)			FADMM($\kappa = 2$)			FADMM($\kappa = 5$)			FADMM($\kappa = 10$)		
σ_0 \ ε	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}
1.0	85	1096	1593	88	585	1596	86	783	1483	111	838	1435	85	906	1601
5.0	434	4905	"–"	185	906	1813	247	1106	1706	312	1718	2216	353	2119	2615
10.0	886	"–"	"–"	225	943	1952	302	1180	1779	420	1890	2290	528	2437	2934
20.0	1783	"–"	"–"	243	962	1870	337	1218	1818	497	1984	2384	663	2618	3115
50.0	4453	"–"	"–"	255	975	1882	360	1243	1842	552	1950	2444	767	2736	3234
100.0	"–"	"–"	"–"	259	979	1988	368	1251	1851	573	2065	2564	808	2779	3276
200.0	"–"	"–"	"–"	261	981	1787	373	1256	1756	584	1983	2477	830	2802	3299
500.0	"–"	"–"	"–"	263	983	1992	376	1259	1859	592	2085	2584	847	2819	3316
1000.0	"–"	"–"	"–"	264	984	1891	377	1261	1860	596	2089	2489	854	2826	3323

tables in this section. Indeed, it is the main disadvantage of the ADMM (1.3) and it usually requires users to tune this parameter to find a specific value appropriate to a given problem. Based on the numerical experiments, we find that for most of the cases, Algorithm 1 with a given initial value of the penalty parameter outperforms the original ADMM (1.3) with the same constant penalty parameter; thus the theoretically faster $O(1/n^2)$ convergence rate is

Table 6: MLL dataset: iteration numbers of ADMM and faster ADMM. (“–” means the iteration number is beyond 5000)

	ADMM			FADMM($\kappa = 1$)			FADMM($\kappa = 2$)			FADMM($\kappa = 5$)			FADMM($\kappa = 10$)		
σ_0 \ ε	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}
1.0	10	905	“–”	10	228	1751	10	431	2170	10	381	2634	10	658	3192
5.0	4	4546	“–”	4	421	1839	4	576	2336	4	873	3028	4	1224	3938
10.0	2	“–”	“–”	2	432	1851	2	598	2359	2	926	3084	2	1327	4049
20.0	4	“–”	“–”	4	313	1857	4	610	2370	4	955	3113	4	1383	4107
50.0	3	“–”	“–”	5	441	1860	4	617	2378	3	974	3132	3	1420	4145
100.0	8	“–”	“–”	6	443	1862	4	620	2381	6	981	3139	8	1435	4160
200.0	16	“–”	“–”	3	319	1863	5	622	2383	10	985	3144	13	1444	4169
500.0	38	“–”	“–”	4	320	1863	7	623	2384	13	989	3147	20	1451	4176
1000.0	74	“–”	“–”	7	445	1864	8	624	2385	16	991	3149	23	1455	4180

Table 7: PROSTATE dataset: iteration numbers of ADMM and faster ADMM. (“–” means the iteration number is beyond 5000)

	ADMM			FADMM($\kappa = 1$)			FADMM($\kappa = 2$)			FADMM($\kappa = 5$)			FADMM($\kappa = 10$)		
σ_0 \ ε	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}
1.0	38	926	4259	41	500	1151	39	605	1498	38	689	2253	38	754	2707
5.0	26	4601	“–”	25	622	1279	26	770	1841	26	1126	2908	26	1558	3853
10.0	161	“–”	“–”	26	641	1299	47	806	1879	83	1213	3001	124	1728	4038
20.0	334	“–”	“–”	30	576	1309	74	826	1899	137	1261	3050	176	1823	4136
50.0	835	“–”	“–”	38	657	1315	85	839	1912	163	1292	3081	224	1884	4199
100.0	1670	“–”	“–”	40	660	1317	90	843	1916	174	1303	3093	245	1907	4222
200.0	3338	“–”	“–”	52	586	1319	92	846	1919	180	1310	3099	258	1921	4235
500.0	“–”	“–”	“–”	53	587	1320	94	848	1921	185	1315	3104	268	1931	4246
1000.0	“–”	“–”	“–”	53	663	1320	95	849	1922	188	1318	3107	273	1936	4251

Table 8: SRBCT dataset: iteration numbers of ADMM and faster ADMM. (“–” means the iteration number is beyond 5000)

	ADMM			FADMM($\kappa = 1$)			FADMM($\kappa = 2$)			FADMM($\kappa = 5$)			FADMM($\kappa = 10$)		
σ_0 \ ε	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}
1.0	56	90	182	48	67	124	52	84	142	54	88	159	55	89	170
5.0	186	411	900	87	122	186	109	161	249	125	221	363	145	270	466
10.0	368	819	1798	98	132	198	128	180	271	163	266	415	205	349	560
20.0	733	1634	3592	103	138	204	139	192	283	189	294	444	252	401	616
50.0	1828	4081	“–”	107	142	208	146	199	291	208	313	463	288	439	654
100.0	3654	“–”	“–”	109	144	210	149	202	294	215	321	471	302	454	670
200.0	“–”	“–”	“–”	110	145	210	151	204	296	220	325	476	312	463	679
500.0	“–”	“–”	“–”	111	145	211	153	206	297	224	329	480	319	471	687
1000.0	“–”	“–”	“–”	111	146	212	154	207	298	226	331	482	323	475	691

numerically verified. Moreover, our preliminary numerical results show that Algorithm 1 can sometimes produce very high-accuracy solutions with few iterations; this can be hardly achieved by the original ADMM (1.3) with a constant penalty parameter unless it is very well tuned. In our experiments, we also show that Algorithm 1 is much less sensitive to the initial value of the penalty parameter. Indeed, for the LASSO model, Algorithm 1 is very robust with respect to the initial value of σ_0. These features indicate that compared with the original ADMM (1.3) with a constant penalty parameter, theoretically Algorithm 1 has a higher order of worst-case convergence rate and numerically it performs more efficiently and robustly. These favorable features make Algorithm 1 more attractive to a number of applications.

References

[1] H. H. Bauschke and P. L. Combettes, Convex analysis and monotone operator theory in Hilbert spaces, Springer, New York, 2011.

[2] D. Boley, Local linear convergence of the alternating direction method of multipliers on quadratic or linear programs, SIAM J. Optim. **23** (2013), no. 4, 2183–2207.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learning **3** (2010), no. 1, 1–122.

[4] A. Chambolle and T. Pock, A first-order primal-dual algorithm for convex problems with applications to imaging, J. Math. Imaging Vision **40** (2011), no. 1, 120–145.

[5] A. Chambolle and T. Pock, On the ergodic convergence rates of a first-order primal-dual algorithm, Math. Program. **159** (2016), no. 1, 253–287.

[6] T. F. Chan and R. Glowinski, Finite element approximation and iterative solution of a class of mildly nonlinear elliptic equations, Technical Report, Computer Science Department, Stanford University, CA, 1978.

[7] E. Corman and X. M. Yuan, A generalized proximal point algorithm and its convergence rate, SIAM J. Optim. **24** (2014), no. 4, 1614–1638.

[8] D. Davis and W. T. Yin, Convergence rate analysis of several splitting schemes, Splitting Methods in Communication, Imaging, Science, and Engineering (R. Glowinski, S. J. Osher, and W. T. Yin, eds.), Springer International Publishing, 2016, pp. 115–163.

[9] D. Davis and W. T. Yin, Faster convergence rates of relaxed Peaceman-Rachford and ADMM under regularity assumptions, Math. Oper. Res. **42** (2017), no. 3, 783–805.

[10] D. L. Donoho and Y. Tsaig, Fast solution of ℓ_1-norm minimization problems when the solution may be sparse, IEEE Trans. Inform. Theory **54** (2008), no. 11, 4789–4812.

[11] J. Douglas, Jr. and H. H. Rachford, Jr., On the numerical solution of heat conduction problems in two and three space variables, Trans. Amer. Math. Soc. **82** (1956), no. 2, 421–439.

[12] J. Eckstein, Some saddle-function splitting methods for convex programming, Optim. Methods Softw. **4** (1994), no. 1, 75–83.

[13] J. Eckstein and D. P. Bertsekas, On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators, Math. Program. **55** (1992), no. 3, 293–318.

[14] J. Eckstein and W. Yao, Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives, Pac. J. Optim. **11** (2015), no. 4, 619–644.

[15] D. Gabay, Applications of the method of multipliers to variational inequalities, Augmented Lagrangian Methods: Applications to the Solution of Boundary-Valued Problems (M. Fortin and R. Glowinski, eds.), North-Holland Publishing Co., Amsterdam, 1983, pp. 299–331.

[16] D. Gabay and B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximation, Comput. Math. Appl. **2** (1976), no. 1, 17–40.

[17] R. Glowinski, On alternating direction methods of multipliers: A historical perspective, Modeling, Simulation and Optimization for Science and Technology (W. Fitzgibbon, Y. A. Kuznetsov, P. Neittaanmäki, and O. Pironneau, eds.), Springer Netherlands, 2014, pp. 59–82 (English).

[18] R. Glowinski and A. Marrocco, Sur l'approximation par éléments finis et la résolution par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires, R.A.I.R.O. **R2** (1975), 41–76.

[19] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, Fast alternating direction optimization methods, SIAM J. Imaging Sci. **7** (2014), no. 3, 1588–1623.

[20] D. R. Han and X. M. Yuan, Local linear convergence of the alternating direction method of multipliers for quadratic programs, SIAM J. Numer. Anal. **51** (2013), no. 6, 3446–3457.

[21] B. S. He, L.-Z. Liao, D. R. Han, and H. Yang, A new inexact alternating directions method for monotone variational inequalities, Math. Program. **92** (2002), no. 1, 103–118.

[22] B. S. He, Y. F. You, and X. M. Yuan, On the convergence of primal-dual hybrid gradient algorithm, SIAM J. Imaging Sci. **7** (2014), no. 4, 2526–2537.

[23] B. S. He and X. M. Yuan, Convergence analysis of primal-dual algorithms for a saddle-point problem: from contraction perspective, SIAM J. Imaging Sci. **5** (2012), no. 1, 119–149.

[24] B. S. He and X. M. Yuan, On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method, SIAM J. Numer. Anal. **50** (2012), no. 2, 700–709.

[25] B. S. He and X. M. Yuan, On non-ergodic convergence rate of Douglas-Rachford alternating direction method of multipliers, Numer. Math. **130** (2015), no. 3, 567–577.

[26] M. R. Hestenes, Multiplier and gradient methods, J. Optim. Theory Appl. **4** (1969), no. 5, 303–320.

[27] P.-L. Lions and B. Mercier, Splitting algorithms for the sum of two nonlinear operators, SIAM J. Numer. Anal. **16** (1979), no. 6, 964–979.

[28] B. Martinet, Régularisation dinéquations variationnelles par approximations successives, Rev. Franç. Inform. Rech. Opér. **4** (1970), 154–158.

[29] R. D. C. Monteiro and B. F. Svaiter, Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers, SIAM J. Optim. **23** (2013), no. 1, 475–507.

[30] J.-J. Moreau, Proximité et dualité dans un espace hilbertien, Bull. Soc. Math. France **93** (1965), 273–299.

[31] A. Nemirovski, Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems, SIAM J. Optim. **15** (2004), no. 1, 229–251.

[32] Y. Nesterov, Gradient methods for minimizing composite functions, Math. Program. **140** (2013), no. 1, 125–161.

[33] Y. E. Nesterov, A method for solving the convex programming problem with convergence rate $O(1/k^2)$, Dokl. Akad. Nauk SSSR **269** (1983), no. 3, 543–547, (In Russian. Translated in Soviet Math. Dokl., 27 (1983), pp. 372-376.).

[34] M. J. D. Powell, A method for nonlinear constraints in minimization problems, Optimization (R. Fletcher, ed.), Academic Press, New York, 1969, pp. 283–298.

[35] R. T. Rockafellar, Convex analysis, Princeton University Press, Princeton, NJ, 1997.

[36] R. Shefi, Rate of convergence analysis for convex nonsmooth optimization algorithms, PhD Thesis, Tel Aviv University, Israel, 2015.

[37] M. Tao and X. M. Yuan, On the optimal linear convergence rate of a generalized proximal point algorithm, J. Sci. Comput. **74** (2018), no. 2, 826–850.

[38] W. Y. Tian and X. M. Yuan, Convergence analysis of primal-dual based methods for total variation minimization with finite element approximation, J. Sci. Comput. **76** (2018), no. 1, 243–274.

[39] R. Tibshirani, Regression shrinkage and selection via the lasso, J. Roy. Statist. Soc. Ser. B **58** (1996), no. 1, 267–288.

[40] X. F. Wang and X. M. Yuan, The linearized alternating direction method of multipliers for Dantzig selector, SIAM J. Sci. Comput. **34** (2012), no. 5, A2792–A2811.

[41] J. F. Yang and X. M. Yuan, Linearized augmented Lagrangian and alternating direction methods for nuclear norm minimization, Math. Comp. **82** (2012), no. 281, 301–329.

[42] K. Yang, Z. P. Cai, J. Z. Li, and G. H. Lin, A stable gene selection in microarray data analysis, BMC Bioinformatics **7** (2006), no. 1, 228.

[43] X. Q. Zhang, M. Burger, and S. Osher, A unified primal-dual algorithm framework based on Bregman iteration, J. Sci. Comput. **46** (2011), no. 1, 20–46.