Self Equivalence of the Alternating Direction Method of Multipliers

Ming Yan* Wotao Yin*

March 5, 2015

Abstract

The alternating direction method of multipliers (ADM or ADMM) breaks a complex optimization problem into much simpler subproblems. The ADM algorithms are typically short and easy to implement yet exhibit (nearly) state-of-the-art performance for large-scale optimization problems.

To apply ADM, we first formulate a given problem into the "ADM-ready" form, so the final algorithm depends on the formulation. A problem like $\text{minimize}_{\mathbf{x}}\, u(\mathbf{x}) + v(\mathbf{C}\mathbf{x})$ has six different "ADM-ready" formulations. They can be in the primal or dual forms, and they differ by how dummy variables are introduced. To each "ADM-ready" formulation, ADM can be applied in two different orders depending on how the primal variables are updated. Finally, we get twelve different ADM algorithms! How do they compare to each other? Which algorithm should one choose?

In this chapter, we show that many of the different ways of applying ADM are equivalent. Specifically, we show that ADM applied to a primal formulation is equivalent to ADM applied to its Lagrange dual; ADM is equivalent to a primal-dual algorithm applied to the saddle-point formulation of the same problem. These results are surprising since the primal and dual variables in ADM are seemingly treated very differently, and some previous work exhibit preferences in one over the other on specific problems. In addition, when one of the two objective functions is quadratic, possibly subject to an affine constraint, we show that swapping the update order of the two primal variables in ADM gives the same algorithm. These results identify the few truly different ADM algorithms for a problem, which generally have different forms of subproblems from which it is easy to pick one with the most computationally friendly subproblems.

Keywords: alternating direction method of multipliers, ADM, ADMM, Douglas-Rachford splitting (DRS), Peaceman-Rachford splitting (PRS), primal-dual algorithm

*Department of Mathematics, University of California, Los Angeles, CA 90095, USA. Emails: yanm@math.ucla.edu and wotaoyin@math.ucla.edu

1 Introduction

The Alternating Direction Method of Multipliers (ADM or ADMM) is a very popular algorithm with wide applications in signal and image processing, machine learning, statistics, compressive sensing, and operations research. Combined with problem reformulation tricks, the method can reduce a complicated problem into much simpler subproblems.

The vanilla ADM applies to a linearly-constrained problem with separable convex objective functions in the following "ADM-ready" form:

$$\begin{cases} \underset{\mathbf{x},\mathbf{y}}{\text{minimize}} & f(\mathbf{x})+g(\mathbf{y}) \\ \text{subject to} & \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}=\mathbf{b}, \end{cases} \tag{P1}$$

where functions f, g are proper, closed, convex but not necessarily differentiable. ADM reduces (P1) into two simpler subproblems and then iteratively updates $\mathbf{x}$, $\mathbf{y}$, as well as a multiplier (dual) variable $\mathbf{z}$. Given $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k)$, ADM generates $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{z}^{k+1})$ as follows

1. $\mathbf{y}^{k+1} \in \arg\min_{\mathbf{y}} g(\mathbf{y}) + (2\lambda)^{-1}\|\mathbf{A}\mathbf{x}^k + \mathbf{B}\mathbf{y} - \mathbf{b} + \lambda\mathbf{z}^k\|_2^2$,
2. $\mathbf{x}^{k+1} \in \arg\min_{\mathbf{x}} f(\mathbf{x}) + (2\lambda)^{-1}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y}^{k+1} - \mathbf{b} + \lambda\mathbf{z}^k\|_2^2$,
3. $\mathbf{z}^{k+1} = \mathbf{z}^k + \lambda^{-1}(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{y}^{k+1} - \mathbf{b})$,

where $\lambda > 0$ is a fixed parameter. We use "$\in$" since the subproblems do not necessarily have unique solutions.

Since $\{f, \mathbf{A}, \mathbf{x}\}$ and $\{g, \mathbf{B}, \mathbf{y}\}$ are in symmetric positions in (P1), swapping them does not change the problem. This corresponds to switching the order that $\mathbf{x}$ and $\mathbf{y}$ are updated in each iteration. But, since the variable updated first is used in the updating of the other variable, this swap leads to a different sequence of variables and thus a different algorithm.

Note that the order switch does not change the per-iteration cost of ADM. Also note that one, however, cannot mix the two update orders at different iterations because it will generally cause divergence, even when the primal-dual solution to (P1) is unique.

1.1 ADM works in many different ways

In spite of its popularity and vast literature, there are still simple unanswered questions about ADM: how many ways can ADM be applied? and which ways work better? Before answering these questions, let us examine the following problem, to which we can find **twelve different ways to apply ADM:**

$$\underset{\mathbf{x}}{\text{minimize}}\ u(\mathbf{x}) + v(\mathbf{C}\mathbf{x}), \tag{1}$$

where u and v are proper, closed, convex functions and $\mathbf{C}$ is a linear mapping. Problem (1) generalizes a large number of signal and image processing, inverse problem, and machine learning models.

We shall reformulate (1) into the form of (P1). By introducing dummy variables in two different ways, we obtain two ADM-ready formulations of problem (1):

$$\left\{\begin{array}{ll}\underset{\mathbf{x},\mathbf{y}}{\text{minimize}} & u(\mathbf{x})+v(\mathbf{y})\\ \text{subject to} & \mathbf{C}\mathbf{x}-\mathbf{y}=0\end{array}\right. \quad \text{and} \quad \left\{\begin{array}{ll}\underset{\mathbf{x},\bar{\mathbf{y}}}{\text{minimize}} & u(\mathbf{x})+v(\mathbf{C}\bar{\mathbf{y}})\\ \text{subject to} & \mathbf{x}-\bar{\mathbf{y}}=0.\end{array}\right. \tag{2}$$

In addition, we can derive the dual problem of (1):

$$\underset{\mathbf{v}}{\text{minimize}}\ u^*(-\mathbf{C}^*\mathbf{v})+v^*(\mathbf{v}), \tag{3}$$

where u^*, v^* are the convex conjugates (i.e., Legendre transforms) of functions u, v, respectively, $\mathbf{C}^*$ is the adjoint of $\mathbf{C}$, and $\mathbf{v}$ is the dual variable. (The steps to derive (3) from (1) are standard and thus omitted.) Then, we also reformulate (3) into two ADM-ready forms, which use different dummy variables:

$$\left\{\begin{array}{ll}\underset{\mathbf{u},\mathbf{v}}{\text{minimize}} & u^*(\mathbf{u})+v^*(\mathbf{v})\\ \text{subject to} & \mathbf{u}+\mathbf{C}^*\mathbf{v}=0\end{array}\right. \quad \text{and} \quad \left\{\begin{array}{ll}\underset{\bar{\mathbf{u}},\mathbf{v}}{\text{minimize}} & u^*(\mathbf{C}^*\bar{\mathbf{u}})+v^*(\mathbf{v})\\ \text{subject to} & \bar{\mathbf{u}}+\mathbf{v}=0.\end{array}\right. \tag{4}$$

Clearly, ADM can be applied to all of the four formulations in (2) and (4), and including the update order swaps, there are *eight different ways* to apply ADM.

Under some technical conditions such as the existence of saddle-point solutions, all the eight ADM will converge to a saddle-point solution or solutions for problem (1). In short, they all work.

It is worth noting that by the Moreau identity, the subproblems involving u^* and v^* can be easily reduced to subproblems involving u and v, respectively. No significant computing is required.

The two formulations in (2), however, lead to significantly different ADM subproblems. In the ADM applied to the left formulation, u and $\mathbf{C}$ will appear in one subproblem and v in the other subproblem. To the right formulation, u will be alone while v and $\mathbf{C}$ will appear in the same subproblem. This difference applies to the two formulations in (4) as well. It depends on the structures of $u, v, \mathbf{C}$ to determine the better choices. Therefore, out of the eight, four will have (more) difficult subproblems than the rest.

There are *another four ways* to apply ADM to problem (1). Every one of them will have three subproblems that separately involve $u, v, \mathbf{C}$, so they are all different from the above eight. To get the first two, let us take the left formulation in (2) and introduce a dummy variable $\mathbf{s}$, obtaining a new equivalent formulation

$$\left\{\begin{array}{ll}\underset{\mathbf{x},\mathbf{y},\mathbf{s}}{\text{minimize}} & u(\mathbf{s})+v(\mathbf{y})\\ \text{subject to} & \mathbf{C}\mathbf{x}-\mathbf{y}=0,\\ & \mathbf{x}-\mathbf{s}=0.\end{array}\right. \tag{5}$$

It turns out that the same "dummy variable" trick applied to the right formulation in (2) also gives (5), up to a change of variable names. Although there are three variables, we can group $(\mathbf{y}, \mathbf{s})$ and

treat $\mathbf{x}$ and $(\mathbf{y}, \mathbf{s})$ as the two variables. Then problem (5) has the form (P1). Hence, we have two ways to apply ADM to (5) with two different update orders. Note that $\mathbf{y}$ and $\mathbf{s}$ do not appear together in any equation or function, so the ADM subproblem that updates $(\mathbf{y}, \mathbf{s})$ will further decouple to two separable subproblems of $\mathbf{y}$ and $\mathbf{s}$; in other words, the resulting ADM has three subproblems involving $\{\mathbf{x}, \mathbf{C}\}$,$\{\mathbf{y}, v\}$, $\{\mathbf{s}, u\}$ separately. The other two ways are results of the same "dummy variable" trick applied to the either formulation in (4). Again, since now $\mathbf{C}$ has its own subproblem, these four ways are distinct from the previous eight ways.

As demonstrated through an example, there are quite many ways to formulate the same optimization problem into "ADM-ready" forms and obtain different ADM algorithms. While most ADM users choose just one way without paying much attention to the other choices, some show preferences toward a specific formulation. For example, some prefer (5) over those in (2) and (4) since $\mathbf{C}$, u, v all end up in separate subproblems. When applying ADM to certain ℓ_1 minimization problems, the authors of [24, 25] emphasize on the dual formulations, and later the authors of [23] show a preference over the primal formulations. When ADM was proposed to solve a traffic equilibrium problem, it was first applied to the dual formulation in [13] and, years later, to the primal formulation in [12]. Regarding which one of the two variables should be updated first in ADM, neither a rule nor an equivalence claim is found in the literature. Other than giving preferences to ADM with simpler subproblems, there is no results that compare the different formulations.

1.2 Contributions

This chapter shows that, applied to certain pairs of different formulations of the same problem, ADM will generate equivalent sequences of variables that can be mapped exactly from one to another at every iteration. Specifically, between the sequence of an ADM algorithm on a primal formulation and that on the corresponding dual formulation, such maps exist.

We also show that whenever at least one of f and g is a quadratic function (including affine function as a special case), possibly subject to an affine constraint, the sequence of an ADM algorithm can be mapped to that of the ADM algorithm using the opposite order for updating their variables.

Abusing the word "equivalence", we say that ADM has "primal-dual equivalence" and "update-order equivalence (with a quadratic objective function)." Equivalent ADM algorithms take the same number of iterations to reach the same accuracy. (However, it is possible that one algorithm is slightly better than the other in terms of numerical stability, for example, against round-off errors.)

Equipped with these equivalence results, the first eight ways to apply ADM to problem (1) that were discussed in section 1.1 are reduced to four ways in light of primal-dual equivalence, and the four will further reduce to two whenever u or v, or both, is a quadratic function.

The last four ways to apply ADM on problem (1) discussed in section 1.1, which yield three subproblems that separately involve u, v, and $\mathbf{C}$, are all equivalent and reduce to just one due to primal-dual equivalence and one variable in them is associated with 0 objective (for example, variable

$\mathbf{x}$ has 0 objective in problem (5)).

Take the ℓ_p-regularization problem, $p \in [1, \infty]$,

$$\underset{\mathbf{x}}{\text{minimize}} \ \|\mathbf{x}\|_p + f(\mathbf{C}\mathbf{x}) \tag{6}$$

as an example, which is special case of problem (1) with a quadratic function u when $p = 2$. We list its three different formulations, whose ADM algorithms are truly different, as follows. When $p \neq 2$ *and* f is non-quadratic, each of the first two formulations leads to a pair of different ADM algorithms with different orders of variable update; otherwise, each pair of algorithms is equivalent.

1. Left formulation of (2):
$$\begin{cases} \underset{\mathbf{x},\mathbf{y}}{\text{minimize}} & \|\mathbf{x}\|_p + f(\mathbf{y}) \\ \text{subject to} & \mathbf{C}\mathbf{x} - \mathbf{y} = 0. \end{cases}$$
The subproblem for $\mathbf{x}$ involves ℓ_p-norm and $\mathbf{C}$. The other one for $\mathbf{y}$ involves f.

2. Right formulation of (2):
$$\begin{cases} \underset{\mathbf{x},\mathbf{y}}{\text{minimize}} & \|\mathbf{x}\|_p + f(\mathbf{C}\mathbf{y}) \\ \text{subject to} & \mathbf{x} - \mathbf{y} = 0. \end{cases}$$
The subproblem for $\mathbf{x}$ involves ℓ_p-norm and, for $p = 1$ and 2, has a closed-form solution. The other subproblem for $\mathbf{y}$ involves $f(\mathbf{C}\cdot)$.

3. Formulation (5): for any $\mu > 0$,
$$\begin{cases} \underset{\mathbf{x},\mathbf{y},\mathbf{s}}{\text{minimize}} & \|\mathbf{s}\|_p + f(\mathbf{y}) \\ \text{subject to} & \mathbf{C}\mathbf{x} - \mathbf{y} = 0, \\ & \mu(\mathbf{x} - \mathbf{s}) = 0. \end{cases}$$
The subproblem for $\mathbf{x}$ is quadratic program involving $\mathbf{C}^*\mathbf{C} + \mu\mathbf{I}$. The subproblem for $\mathbf{s}$ involves ℓ_p-norm. The subproblem for $\mathbf{y}$ involves f. The subproblems for $\mathbf{s}$ and $\mathbf{y}$ are independent.

The best choice depends on which has the simplest subproblems.

The result of ADM's primal-dual equivalence is surprising for three reasons. Firstly, ADM iteration updates *two* primal variable, $\mathbf{x}^k$ and $\mathbf{y}^k$ in (P1) and *one* dual variable, all in different manners. The updates to the primal variables are done in a Gauss-Seidel manner and involve minimizing functions f and g, but the update to the dual variable is explicit and linear. Surprisingly, ADM actually treats one of the two primal variables and the dual variable equally as we will later show. Secondly, most literature describes ADM as an inexact version of the Augmented Lagrangian Method (ALM) [17], which updates $(\mathbf{x}, \mathbf{y})$ together rather than one after another. Although ALM maintains the primal variables, under the hood ALM is the dual-only proximal-point algorithm that iterates the dual variable. It is commonly believed that ADM is an inexact dual algorithm. Thirdly, primal and dual

problems typically have different sizes and regularity properties, causing the same algorithm, even if it is applicable to both, to exhibit different performance. For example, the primal and dual variables may have different dimensions. If the primal function f is Lipschitz differentiable, the dual function f^* is strongly convex but can be non-differentiable, and vice versa. Such primal-dual differences often mean that it is numerically advantageous to solve one rather than the other, yet our result means that there is no such primal-dual difference on ADM.

Our maps between equivalent ADM sequences have very simple forms, as the reader will see below. Besides the technical proofs that establish the maps, it is interesting to mention the operator-theoretic perspective of our results. It is shown in [13] that the dual-variable sequence of ADM coincides with a sequence of the Douglas-Rachford splitting (DRS) algorithm [7, 18]. Our ADM's primal-dual equivalence can be obtained through the above ADM–DRS relation and the Moreau identity: $\mathbf{prox}_h + \mathbf{prox}_{h^*} = \mathbf{I}$, applied to the proximal maps of f and f^* and those of g and g^*. The details are omitted in this chapter. Here, $\mathbf{prox}_h(x) := \arg\min_s h(s) + \frac{1}{2}\|s - x\|^2$.

Our results of primal-dual equivalence for ADM extends to the Peaceman-Rachford splitting (PRS) algorithm. Let the PRS operator [19] be denoted as $\mathbf{T}_{\mathrm{PRS}} = (2\mathbf{prox}_f - \mathbf{I}) \circ (2\mathbf{prox}_g - \mathbf{I})$. The DRS operator is the average of the identity map and the PRS operator: $\mathbf{T}_{\mathrm{DRS}} = \frac{1}{2}\mathbf{I} + \frac{1}{2}\mathbf{T}_{\mathrm{PRS}}$, and the Relaxed PRS (RPRS) operator is a weighted-average: $\mathbf{T}_{\mathrm{RPRS}} = (1 - \alpha)\mathbf{I} + \alpha\mathbf{T}_{\mathrm{PRS}}$, where $\alpha \in (0, 1]$. The DRS and PRS algorithms that iteratively apply their operators to find a fixed point were originally proposed for evolving PDEs with two spatial dimensions in the 1950s and then extended to finding a root of the sum of two maximal monotone (set-valued) mappings by Lions and Mercier [18]. Eckstein showed, in [8, Chapter 3.5], that DRS/PRS applied to the primal problem (1) is equivalent to DRS/PRS applied to the dual problem (4) when $\mathbf{C} = \mathbf{I}$. We will show that RPRS applied to (1) is equivalent to RPRS applied to (3) for all $\mathbf{C}$.

In addition to the aforementioned primal-dual and update-order equivalence, we obtain a primal-dual algorithm for the saddle-point formulation of (P1) that is also equivalent to the ADM. This primal-dual algorithm is generally *different* from the primal-dual algorithm proposed by Chambolle and Pock [3], while they become the same in a special case. The connection between these two algorithms will be explained.

Even when using the same number of dummy variables, truly different ADM algorithms can have different iteration complexities (do not confuse them with the difficulties of their subproblems). The convergence analysis of ADM, such as conditions for sublinear or linear convergence, involves many different scenarios [4–6]. The discussion of convergence rates of ADM algorithms is beyond the scope of this chapter. Our focus is on the equivalence.

1.3 Organization

This chapter is organized as follows. Section 2 specifies our notation, definitions, and basic assumptions. The three equivalence results for ADM are shown in sections 4, 5, and 6: The primal-dual

equivalence of ADM is discussed in sections 4; ADM is shown to be equivalent to a primal-dual algorithm applied to the saddle-point formulation in section 5; In section 6, we show the update-order equivalence of ADM if f or g is a quadratic function, possibly subject to an affine constraint. The primal-dual equivalence of RPRS is shown in section 7. We conclude this chapter with the application of our results on total variation image denoising in section 8.

2 Notation, definitions, and assumptions

Let $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{G}$ be (possibly infinite dimensional) Hilbert spaces. Bold lowercase letters such as $\mathbf{x}$, $\mathbf{y}$, $\mathbf{u}$, and $\mathbf{v}$ are used for points in the Hilbert spaces. In the example of (P1), we have $\mathbf{x} \in \mathcal{H}_1$, $\mathbf{y} \in \mathcal{H}_2$, and $\mathbf{b} \in \mathcal{G}$. When the Hilbert space a point belongs to is clear from the context, we do not specify it for the sake of simplicity. The inner product between points $\mathbf{x}$ and $\mathbf{y}$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle$, and $\|\mathbf{x}\|_2 := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ is the corresponding norm. $\|\cdot\|_1$ and $\|\cdot\|_\infty$ denote the ℓ_1 and ℓ_∞ norms, respectively. Bold uppercase letters such as $\mathbf{A}$ and $\mathbf{B}$ are used for both continuous linear mappings and matrices. $\mathbf{A}^*$ denotes the adjoint of $\mathbf{A}$. $\mathbf{I}$ denotes the identity mapping.

If $\mathcal{C}$ is a convex and nonempty set, the indicator function $\iota_{\mathcal{C}}$ is defined as follows:

$$\iota_{\mathcal{C}}(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x} \in \mathcal{C}, \\ \infty, & \text{if } \mathbf{x} \notin \mathcal{C}. \end{cases}$$

Both lower and upper case letters such as f, g, F, and G are used for functions. Let $\partial f(\mathbf{x})$ be the subdifferential of function f at $\mathbf{x}$. The proximal operator $\mathbf{prox}_{f(\cdot)}$ of function f is defined as

$$\mathbf{prox}_{f(\cdot)}(\mathbf{x}) = \arg\min_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2,$$

where the minimization has the unique solution. The convex conjugate f^* of function f is defined as

$$f^*(\mathbf{v}) = \sup_{\mathbf{x}} \{\langle \mathbf{v}, \mathbf{x} \rangle - f(\mathbf{x})\}.$$

Let $\mathbf{L} : \mathcal{H} \to \mathcal{G}$, the *infimal postcomposition* [1, Def. 12.33] of $f : \mathcal{H} \to (-\infty, +\infty]$ by $\mathbf{L}$ is given by

$$\mathbf{L} \triangleright f : \mathbf{s} \mapsto \inf f(\mathbf{L}^{-1}(\mathbf{s})) = \inf_{\mathbf{x}:\mathbf{L}\mathbf{x}=\mathbf{s}} f(\mathbf{x}),$$

with $\mathrm{dom}(\mathbf{L} \triangleright f) = \mathbf{L}(\mathrm{dom}(f))$.

Lemma 1. *If f is convex and $\mathbf{L}$ is affine and expressed as $\mathbf{L}(\cdot) = \mathbf{A} \cdot + \mathbf{b}$, then $\mathbf{L} \triangleright f$ is convex and the convex conjugate of $\mathbf{L} \triangleright f$ can be found as follows:*

$$(\mathbf{L} \triangleright f)^*(\cdot) = f^*(\mathbf{A}^* \cdot) + \langle \cdot, \mathbf{b} \rangle.$$

Proof. Following from the definitions of convex conjugate and infimal postcomposition, we have

$$\begin{aligned}(\mathbf{L} \triangleright f)^*(\mathbf{v}) &= \sup_{\mathbf{y}} \langle \mathbf{v}, \mathbf{y} \rangle - \mathbf{L} \triangleright f(\mathbf{y}) = \sup_{\mathbf{x}} \langle \mathbf{v}, \mathbf{A}\mathbf{x} + \mathbf{b} \rangle - f(\mathbf{x}) \\ &= \sup_{\mathbf{x}} \langle \mathbf{A}^*\mathbf{v}, \mathbf{x} \rangle - f(\mathbf{x}) + \langle \mathbf{v}, \mathbf{b} \rangle = f^*(\mathbf{A}^*\mathbf{v}) + \langle \mathbf{v}, \mathbf{b} \rangle.\end{aligned}$$

□

Definition 1. *An algorithm on one problem is* equivalent to *another algorithm on the same or another equivalent problem means that the steps in one algorithm can be recovered from the steps in another algorithm, with proper initial conditions and parameters.*

Definition 2. *An optimization algorithm is called* primal-dual equivalent *if this algorithm applied to the primal formulation is equivalent to the same algorithm applied to its Lagrange dual.*

It is important to note that most algorithms are not primal-dual equivalent. ALM applied to the primal problem is equivalent to proximal point method applied to the dual problem [20], but both algorithms are not primal-dual equivalent. In this chapter, we will show that ADM and RPRS are primal-dual equivalent.

We make the following assumptions throughout the chapter:

Assumption 1. *Functions in this chapter are assumed to be proper, closed, and convex.*

Assumption 2. *The saddle-point solutions to all the optimization problems in this chapter are assumed to exist.*

3 Equivalent problems

A *primal formulation* equivalent to (P1) is

$$\begin{cases} \underset{\mathbf{s},\mathbf{t}}{\text{minimize}} & F(\mathbf{s}) + G(\mathbf{t}) \\ \text{subject to} & \mathbf{s} + \mathbf{t} = \mathbf{0}, \end{cases} \tag{P2}$$

where $\mathbf{s}, \mathbf{t} \in \mathcal{G}$ and

$$F(\mathbf{s}) := \min_{\mathbf{x}} f(\mathbf{x}) + \iota_{\{\mathbf{x}:\mathbf{A}\mathbf{x}=\mathbf{s}\}}(\mathbf{x}), \tag{7a}$$

$$G(\mathbf{t}) := \min_{\mathbf{y}} g(\mathbf{y}) + \iota_{\{\mathbf{y}:\mathbf{B}\mathbf{y}-\mathbf{b}=\mathbf{t}\}}(\mathbf{y}). \tag{7b}$$

Remark 1. *If we define* $\mathbf{L}_f$ *and* $\mathbf{L}_g$ *as* $\mathbf{L}_f(\mathbf{x}) = \mathbf{A}\mathbf{x}$ *and* $\mathbf{L}_g(\mathbf{y}) = \mathbf{B}\mathbf{y} - \mathbf{b}$*, respectively, then*

$$F = \mathbf{L}_f \triangleright f, \qquad G = \mathbf{L}_g \triangleright g.$$

The Lagrange dual of (P1) is

$$\underset{\mathbf{v}}{\text{minimize}} \quad f^*(-\mathbf{A}^*\mathbf{v}) + g^*(-\mathbf{B}^*\mathbf{v}) + \langle \mathbf{v}, \mathbf{b} \rangle, \tag{8}$$

which can be derived from $\underset{\mathbf{v}}{\text{minimize}} \left(-\min_{\mathbf{x},\mathbf{y}} L(\mathbf{x}, \mathbf{y}, \mathbf{v}) \right)$ with the Lagrangian defined as follows:

$$L(\mathbf{x}, \mathbf{y}, \mathbf{v}) = f(\mathbf{x}) + g(\mathbf{y}) + \langle \mathbf{v}, \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{b} \rangle.$$

An *ADM-ready* formulation of (8) is

$$\begin{cases} \underset{\mathbf{u},\mathbf{v}}{\text{minimize}} & f^*(-\mathbf{A}^*\mathbf{u}) + g^*(-\mathbf{B}^*\mathbf{v}) + \langle \mathbf{v}, \mathbf{b} \rangle \\ \text{subject to} & \mathbf{u} - \mathbf{v} = \mathbf{0}. \end{cases} \tag{D1}$$

When ADM is applied to an ADM-ready formulation of the Lagrange dual problem, we call it *Dual ADM*. The original ADM is called *Primal ADM*.

Following similar steps, the ADM ready formulation of the Lagrange dual of (P2) is

$$\begin{cases} \underset{\mathbf{u},\mathbf{v}}{\text{minimize}} & F^*(-\mathbf{u}) + G^*(-\mathbf{v}) \\ \text{subject to} & \mathbf{u} - \mathbf{v} = \mathbf{0}. \end{cases} \tag{D2}$$

The equivalence between (D1) and (D2) is trivial since

$$\begin{aligned} F^*(\mathbf{u}) &= f^*(\mathbf{A}^*\mathbf{u}), \\ G^*(\mathbf{v}) &= g^*(\mathbf{B}^*\mathbf{v}) - \langle \mathbf{v}, \mathbf{b} \rangle, \end{aligned}$$

which follows from Lemma 1.

Although there can be multiple equivalent formulations of the same problem (e.g., (P1), (P2), (8), and (D1)/(D2) are equivalent), an algorithm may or may not be applicable to some of them. Even when they are, on different formulations, their behaviors such as convergence and speed of convergence are different. In particular, most algorithms have different behaviors on primal and dual formulations of the same problem. An algorithm applied to a primal formulation does not dictate the behavior of the same algorithm applied to the related dual formulation. The simplex method in linear programming has different performance when applied to both the primal and dual problems, i.e., the primal simplex method starts with a primal basic feasible solution (dual infeasible) until the dual feasibility conditions are satisfied, while the dual simplex method starts with a dual basic feasible solution (primal infeasible) until the primal feasibility conditions are satisfied. The ALM also has different performance when applied to the primal and dual problems, i.e., ALM applied to the primal problem is equivalent to proximal point method applied to the related dual problem, and proximal point method is, in general, different from ALM on the same problem.

4 Primal-dual equivalence of ADM

In this section we show the primal-dual equivalence of ADM. Algorithms 1-3 describe how ADM is applied to (P1), (P2), and (D1)/ (D2) [14, 15].

Algorithm 1 ADM on (P1)

initialize $\mathbf{x}_1^0$, $\mathbf{z}_1^0$, $\lambda > 0$

for $k = 0, 1, \cdots$ **do**

$\mathbf{y}_1^{k+1} \in \arg\min_{\mathbf{y}} g(\mathbf{y}) + (2\lambda)^{-1}\|\mathbf{A}\mathbf{x}_1^k + \mathbf{B}\mathbf{y} - \mathbf{b} + \lambda\mathbf{z}_1^k\|_2^2$

$\mathbf{x}_1^{k+1} \in \arg\min_{\mathbf{x}} f(\mathbf{x}) + (2\lambda)^{-1}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y}_1^{k+1} - \mathbf{b} + \lambda\mathbf{z}_1^k\|_2^2$

$\mathbf{z}_1^{k+1} = \mathbf{z}_1^k + \lambda^{-1}(\mathbf{A}\mathbf{x}_1^{k+1} + \mathbf{B}\mathbf{y}_1^{k+1} - \mathbf{b})$

end for

Algorithm 2 ADM on (P2)

initialize $\mathbf{s}_2^0$, $\mathbf{z}_2^0$, $\lambda > 0$

for $k = 0, 1, \cdots$ **do**

$\mathbf{t}_2^{k+1} = \arg\min_{\mathbf{t}} G(\mathbf{t}) + (2\lambda)^{-1}\|\mathbf{s}_2^k + \mathbf{t} + \lambda\mathbf{z}_2^k\|_2^2$

$\mathbf{s}_2^{k+1} = \arg\min_{\mathbf{s}} F(\mathbf{s}) + (2\lambda)^{-1}\|\mathbf{s} + \mathbf{t}_2^{k+1} + \lambda\mathbf{z}_2^k\|_2^2$

$\mathbf{z}_2^{k+1} = \mathbf{z}_2^k + \lambda^{-1}(\mathbf{s}_2^{k+1} + \mathbf{t}_2^{k+1})$

end for

Algorithm 3 ADM on (D1)/(D2)

initialize $\mathbf{u}_3^0$, $\mathbf{z}_3^0$, $\lambda > 0$

for $k = 0, 1, \cdots$ **do**

$\mathbf{v}_3^{k+1} = \arg\min_{\mathbf{v}} G^*(-\mathbf{v}) + \frac{\lambda}{2}\|\mathbf{u}_3^k - \mathbf{v} + \lambda^{-1}\mathbf{z}_3^k\|_2^2$

$\mathbf{u}_3^{k+1} = \arg\min_{\mathbf{u}} F^*(-\mathbf{u}) + \frac{\lambda}{2}\|\mathbf{u} - \mathbf{v}_3^{k+1} + \lambda^{-1}\mathbf{z}_3^k\|_2^2$

$\mathbf{z}_3^{k+1} = \mathbf{z}_3^k + \lambda(\mathbf{u}_3^{k+1} - \mathbf{v}_3^{k+1})$

end for

The $\mathbf{y}_1^k$ and $\mathbf{x}_1^k$ in Algorithm 1 may not be unique because of the matrices $\mathbf{A}$ and $\mathbf{B}$, while $\mathbf{A}\mathbf{x}_1^k$ and $\mathbf{B}\mathbf{y}_1^k$ are unique. In addition, $\mathbf{A}\mathbf{x}_1^k$ and $\mathbf{B}\mathbf{y}_1^k$ are calculated for twice and thus stored in the implementation of Algorithm 1 to save the second calculation. Following the equivalence of Algorithms 1 and 2 in Part 1 of the following theorem 1, we can view problem (P2) as the *master problem* of (P1). We can say that ADM is essentially an algorithm applied only to the master problem

(P2), which is Algorithm 2; this fact has been obscured by the often-seen Algorithm 1, which integrates ADM on the master problem with the independent subproblems in (7).

Theorem 1 (Equivalence of Algorithms 1-3)**.** *Suppose* $\mathbf{A}\mathbf{x}_1^0 = \mathbf{s}_2^0 = \mathbf{z}_3^0$ *and* $\mathbf{z}_1^0 = \mathbf{z}_2^0 = \mathbf{u}_3^0$ *and that the same parameter* λ *is used in Algorithms 1-3. Then, their equivalence can be established as follows:*

1. *From* $\mathbf{x}_1^k$, $\mathbf{y}_1^k$, $\mathbf{z}_1^k$ *of Algorithm 1, we obtain* $\mathbf{t}_2^k$, $\mathbf{s}_2^k$, $\mathbf{z}_2^k$ *of Algorithm 2 through:*

$$\mathbf{t}_2^k = \mathbf{B}\mathbf{y}_1^k - \mathbf{b}, \tag{9a}$$
$$\mathbf{s}_2^k = \mathbf{A}\mathbf{x}_1^k, \tag{9b}$$
$$\mathbf{z}_2^k = \mathbf{z}_1^k. \tag{9c}$$

   From $\mathbf{t}_2^k$, $\mathbf{s}_2^k$, $\mathbf{z}_2^k$ *of Algorithm 2, we obtain* $\mathbf{y}_1^k$, $\mathbf{x}_1^k$, $\mathbf{z}_1^k$ *of Algorithm 1 through:*

$$\mathbf{y}_1^k = \arg\min_{\mathbf{y}} \{g(\mathbf{y}) : \mathbf{B}\mathbf{y} - \mathbf{b} = \mathbf{t}_2^k\}, \tag{10a}$$
$$\mathbf{x}_1^k = \arg\min_{\mathbf{x}} \{f(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{s}_2^k\}, \tag{10b}$$
$$\mathbf{z}_1^k = \mathbf{z}_2^k. \tag{10c}$$

2. *We can recover the iterates of Algorithm 2 and 3 from each other through*

$$\mathbf{u}_3^k = \mathbf{z}_2^k, \qquad \mathbf{z}_3^k = \mathbf{s}_2^k. \tag{11}$$

Proof. Part 1. Proof by induction.
We argue that under (9b) and (9c), Algorithms 1 and 2 have *essentially identical* subproblems in their *first* steps at the kth iteration. Consider the following problem, which is obtained by plugging the definition of $G(\cdot)$ into the $\mathbf{t}_2^{k+1}$-subproblem of Algorithm 2:

$$(\mathbf{y}_1^{k+1}, \mathbf{t}_2^{k+1}) = \arg\min_{\mathbf{y},\mathbf{t}} g(\mathbf{y}) + \iota_{\{(\mathbf{y},\mathbf{t}):\mathbf{B}\mathbf{y}-\mathbf{b}=\mathbf{t}\}}(\mathbf{y},\mathbf{t}) + (2\lambda)^{-1}\|\mathbf{s}_2^k + \mathbf{t} + \lambda\mathbf{z}_2^k\|_2^2. \tag{12}$$

If one minimizes over $\mathbf{y}$ first while keeping $\mathbf{t}$ as a variable, one eliminates $\mathbf{y}$ and recovers the $\mathbf{t}_2^{k+1}$-subproblem of Algorithm 2. If one minimizes over $\mathbf{t}$ first while keeping $\mathbf{y}$ as a variable, then after plugging in (9b) and (9c), problem (12) reduces to the $\mathbf{y}_1^{k+1}$-subproblem of Algorithm 1. In addition, $(\mathbf{y}_1^{k+1}, \mathbf{t}_2^{k+1})$ obeys

$$\mathbf{t}_2^{k+1} = \mathbf{B}\mathbf{y}_1^{k+1} - \mathbf{b}, \tag{13}$$

which is (9a) at $k+1$. Plugging $\mathbf{t} = \mathbf{t}_2^{k+1}$ into (12) yields problem (10a) for $\mathbf{y}_1^{k+1}$, which must be equivalent to the $\mathbf{y}_1^{k+1}$-subproblem of Algorithm 2. Therefore, the $\mathbf{y}_1^{k+1}$-subproblem of Algorithm 1 and the $\mathbf{t}_2^{k+1}$-subproblem of Algorithm 2 are equivalent through (9a) and (10a) at $k+1$, respectively.

Similarly, under (13) and (9c), we can show that the $\mathbf{x}_1^{k+1}$-subproblem of Algorithm 1 and the $\mathbf{s}_2^{k+1}$-subproblem of Algorithm 2 are equivalent through the formulas for (9b) and (10b) at $k+1$, respectively.

Finally, under (9a) and (9b) at $k+1$ and $\mathbf{z}_2^k = \mathbf{z}_1^k$, the formulas for $\mathbf{z}_1^{k+1}$ and $\mathbf{z}_2^{k+1}$ in Algorithms 1 and 2 are identical, and they return $\mathbf{z}_1^{k+1} = \mathbf{z}_2^{k+1}$, which is (9c) and (10c) at $k+1$.

Part 2. Proof by induction. Suppose that (11) holds. We shall show that (11) holds at $k+1$. Starting from the optimality condition of the $\mathbf{t}_2^{k+1}$-subproblem of Algorithm 2, we derive

$$
\begin{aligned}
&\mathbf{0} \in \partial G(\mathbf{t}_2^{k+1}) + \lambda^{-1}(\mathbf{s}_2^k + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k) \\
\iff &\mathbf{t}_2^{k+1} \in \partial G^*(-\lambda^{-1}(\mathbf{s}_2^k + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k)) \\
\iff &\lambda\left[\lambda^{-1}(\mathbf{s}_2^k + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k)\right] - (\lambda \mathbf{z}_2^k + \mathbf{s}_2^k) \in \partial G^*(-\lambda^{-1}(\mathbf{s}_2^k + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k)) \\
\iff &-\lambda\left[\lambda^{-1}(\mathbf{s}_2^k + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k)\right] + (\lambda \mathbf{u}_3^k + \mathbf{z}_3^k) \in -\partial G^*(-\lambda^{-1}(\mathbf{s}_2^k + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k)) \\
\iff &\mathbf{0} \in -\partial G^*(-\lambda^{-1}(\mathbf{s}_2^k + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k)) - \lambda\left[\mathbf{u}_3^k - \lambda^{-1}(\mathbf{s}_2^k + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k) + \lambda^{-1}\mathbf{z}_3^k\right] \\
\iff &\mathbf{v}_3^{k+1} = \lambda^{-1}(\mathbf{s}_2^k + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k) = \lambda^{-1}(\mathbf{z}_3^k + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k),
\end{aligned}
$$

where the last equivalence follows from the optimality condition for the $\mathbf{v}_3^{k+1}$-subproblem of Algorithm 3.

Starting from the optimality condition of the $\mathbf{s}_2^{k+1}$-subproblem of Algorithm 2, and applying the update, $\mathbf{z}_2^{k+1} = \mathbf{z}_2^k + \lambda^{-1}(\mathbf{s}_2^{k+1} + \mathbf{t}_2^{k+1})$, in Algorithm 2 and the identity of $\mathbf{t}_2^{k+1}$ obtained above, we derive

$$
\begin{aligned}
&\mathbf{0} \in \partial F(\mathbf{s}_2^{k+1}) + \lambda^{-1}(\mathbf{s}_2^{k+1} + \mathbf{t}_2^{k+1} + \lambda \mathbf{z}_2^k) \\
\iff &\mathbf{0} \in \partial F(\mathbf{s}_2^{k+1}) + \mathbf{z}_2^{k+1} \\
\iff &\mathbf{0} \in \mathbf{s}_2^{k+1} - \partial F^*(-\mathbf{z}_2^{k+1}) \\
\iff &\mathbf{0} \in \lambda(\mathbf{z}_2^{k+1} - \mathbf{z}_2^k) - \mathbf{t}_2^{k+1} - \partial F^*(-\mathbf{z}_2^{k+1}) \\
\iff &\mathbf{0} \in \lambda(\mathbf{z}_2^{k+1} - \mathbf{z}_2^k) + \mathbf{z}_3^k + \lambda(\mathbf{z}_2^k - \mathbf{v}_3^{k+1}) - \partial F^*(-\mathbf{z}_2^{k+1}) \\
\iff &\mathbf{0} \in -\partial F^*(-\mathbf{z}_2^{k+1}) + \lambda(\mathbf{z}_2^{k+1} - \mathbf{v}_3^{k+1} + \lambda^{-1}\mathbf{z}_3^k) \\
\iff &\mathbf{z}_2^{k+1} = \mathbf{u}_3^{k+1}.
\end{aligned}
$$

where the last equivalence follows from the optimality condition for the $\mathbf{u}_3^{k+1}$-subproblem of Algorithm 3. Finally, combining the update formulas of $\mathbf{z}_2^{k+1}$ and $\mathbf{z}_3^{k+1}$ in Algorithm 2 and 3, respectively, as well as the identities for $\mathbf{u}_3^{k+1}$ and $\mathbf{v}_3^{k+1}$ obtained above, we obtain

$$
\begin{aligned}
\mathbf{z}_3^{k+1} &= \mathbf{z}_3^k + \lambda(\mathbf{u}_3^{k+1} - \mathbf{v}_3^{k+1}) = \mathbf{s}^k + \lambda(\mathbf{z}_2^{k+1} - \mathbf{z}_2^k - \lambda^{-1}(\mathbf{s}_2^k + \mathbf{t}_2^{k+1})) \\
&= \lambda(\mathbf{z}_2^{k+1} - \mathbf{z}_2^k) - \mathbf{t}_2^{k+1} = \mathbf{s}_2^{k+1}.
\end{aligned}
$$

□

Remark 2. *Part 2 of the theorem (ADM's primal-dual equivalence) can also be derived by combining the following two equivalence results: (i) the equivalence between ADM on the primal problem and the Douglas-Rachford splitting (DRS) algorithm [7, 18] on the dual problem [13], and (ii) the equivalence result between DRS algorithms applied to the master problem (P2) and its dual problem (cf. [8, Chapter 3.5] [9]). In this chapter, however, we provide an elementary algebraic proof in order to derive the formulas in theorem 1 that recover the iterates of one algorithm from another.*

Part 2 of the theorem shows that ADM is a symmetric primal-dual algorithm. The reciprocal positions of parameter λ indicates its function to "balance" the primal and dual progresses.

Part 2 of the theorem also shows that Algorithms 2 and 3 have no difference, in terms of per-iteration complexity and the number of iterations needed to reach an accuracy. However, Algorithms 1 and 2 have difference in terms of per-iteration complexity. In fact, Algorithm 2 is implemented for Algorithm 1 because Algorithm 2 has smaller complexity than Algorithm 1. See the examples in sections 4.2 and 4.3.

4.1 Primal-dual equivalence of ADM on (1) with three subproblems

In section 1.1, we introduced four different ways to apply ADM on (1) with three subproblems. The ADM-ready formulation for the primal problem is (5), and the ADM applied to this formulation is

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{s}^k + \lambda \mathbf{z}_{\mathbf{s}}^k\|_2^2 + \|\mathbf{C}\mathbf{x} - \mathbf{y}^k + \lambda \mathbf{z}_{\mathbf{y}}^k\|_2^2, \tag{14a}$$

$$\mathbf{s}^{k+1} = \arg\min_{\mathbf{s}} u(\mathbf{s}) + (2\lambda)^{-1}\|\mathbf{x}^{k+1} - \mathbf{s} + \lambda \mathbf{z}_{\mathbf{s}}^k\|_2^2, \tag{14b}$$

$$\mathbf{y}^{k+1} = \arg\min_{\mathbf{y}} v(\mathbf{y}) + (2\lambda)^{-1}\|\mathbf{C}\mathbf{x}^{k+1} - \mathbf{y} + \lambda \mathbf{z}_{\mathbf{y}}^k\|_2^2, \tag{14c}$$

$$\mathbf{z}_{\mathbf{s}}^{k+1} = \mathbf{z}_{\mathbf{s}}^k + \lambda^{-1}(\mathbf{x}^{k+1} - \mathbf{s}^{k+1}), \tag{14d}$$

$$\mathbf{z}_{\mathbf{y}}^{k+1} = \mathbf{z}_{\mathbf{y}}^k + \lambda^{-1}(\mathbf{C}\mathbf{x}^{k+1} - \mathbf{y}^{k+1}). \tag{14e}$$

Similarly, we can introduce a dummy variable $\mathbf{t}$ into the left formulation in (4) and obtain a new equivalent formulation

$$\begin{cases} \underset{\mathbf{u},\mathbf{v},\mathbf{t}}{\text{minimize}} & u^*(\mathbf{u}) + v^*(\mathbf{t}) \\ \text{subject to} & \mathbf{C}^*\mathbf{v} + \mathbf{u} = 0,\ \mathbf{v} - \mathbf{t} = 0. \end{cases} \tag{15}$$

The ADM applied to (15) is

$$\mathbf{v}^{k+1} = \arg\min_{\mathbf{v}} \|\mathbf{C}^*\mathbf{v} + \mathbf{u}^k + \lambda^{-1}\mathbf{z}_{\mathbf{u}}^k\|_2^2 + \|\mathbf{v} - \mathbf{t}^k + \lambda^{-1}\mathbf{z}_{\mathbf{t}}^k\|_2^2, \tag{16a}$$

$$\mathbf{u}^{k+1} = \arg\min_{\mathbf{u}} u^*(\mathbf{u}) + \frac{\lambda}{2}\|\mathbf{C}^*\mathbf{v}^{k+1} + \mathbf{u} + \lambda^{-1}\mathbf{z}_{\mathbf{u}}^k\|_2^2, \tag{16b}$$

$$\mathbf{t}^{k+1} = \arg\min_{\mathbf{t}} v^*(\mathbf{t}) + \frac{\lambda}{2}\|\mathbf{v}^{k+1} - \mathbf{t} + \lambda^{-1}\mathbf{z}_{\mathbf{t}}^k\|_2^2, \tag{16c}$$

$$\mathbf{z}_{\mathbf{u}}^{k+1} = \mathbf{z}_{\mathbf{u}}^k + \lambda(\mathbf{C}^*\mathbf{v}^{k+1} + \mathbf{u}^{k+1}), \tag{16d}$$

$$\mathbf{z}_{\mathbf{t}}^{k+1} = \mathbf{z}_{\mathbf{t}}^k + \lambda(\mathbf{v}^{k+1} - \mathbf{t}^{k+1}). \tag{16e}$$

Interestingly, as shown in the following corollary, ADM algorithms (14) and (16) applied to (5) and (15) are equivalent.

Corollary 1. *If the initialization for algorithms* (14) *and* (16) *satisfies* $\mathbf{z}_{\mathbf{y}}^0 = \mathbf{t}^0$, $\mathbf{z}_{\mathbf{s}}^0 = \mathbf{u}^0$, $\mathbf{s}^0 = -\mathbf{z}_{\mathbf{u}}^0$, *and* $\mathbf{y}^0 = \mathbf{z}_{\mathbf{t}}^0$. *Then for* $k \geq 1$, *we have the following equivalence results between the iterations of the two algorithms:*

$$\mathbf{z}_{\mathbf{y}}^k = \mathbf{t}^k, \quad \mathbf{z}_{\mathbf{s}}^k = \mathbf{u}^k, \quad \mathbf{s}^k = -\mathbf{z}_{\mathbf{u}}^k, \quad \mathbf{y}^k = \mathbf{z}_{\mathbf{t}}^k.$$

The proof is similar to the proof of Theorem 1 and is omitted here.

4.2 Example: basis pursuit

The basis pursuit problem seeks for the minimal ℓ_1 solution to a set of linear equations:

$$\underset{\mathbf{u}}{\text{minimize}} \|\mathbf{u}\|_1 \quad \text{subject to } \mathbf{A}\mathbf{u} = \mathbf{b}. \tag{17}$$

Its Lagrange dual is

$$\underset{\mathbf{x}}{\text{minimize}} -\mathbf{b}^T\mathbf{x} \quad \text{subject to } \|\mathbf{A}^*\mathbf{x}\|_\infty \leq 1. \tag{18}$$

The YALL1 algorithms [24] implement ADMs on a set of primal and dual formulations for basis pursuit and LASSO, yet ADM for (17) is not given (however, a linearized ADM is given for (17)). Although seemingly awkward, problem (17) can be turned equivalently into the ADM-ready form

$$\underset{\mathbf{u},\mathbf{v}}{\text{minimize}} \|\mathbf{v}\|_1 + \iota_{\{\mathbf{u}:\mathbf{A}\mathbf{u}=\mathbf{b}\}}(\mathbf{u}) \quad \text{subject to } \mathbf{u} - \mathbf{v} = \mathbf{0}. \tag{19}$$

Similarly, problem (18) can be turned equivalently into the ADM-ready form

$$\underset{\mathbf{x},\mathbf{y}}{\text{minimize}} -\mathbf{b}^T\mathbf{x} + \iota_{B_1^\infty}(\mathbf{y}) \quad \text{subject to } \mathbf{A}^*\mathbf{x} - \mathbf{y} = \mathbf{0}, \tag{20}$$

where $B_1^\infty = \{\mathbf{y} : \|\mathbf{y}\|_\infty \leq 1\}$.

For simplicity, let us suppose that $\mathbf{A}$ has full row rank so the inverse of $\mathbf{A}\mathbf{A}^*$ exists. (Otherwise, $\mathbf{A}\mathbf{u} = \mathbf{b}$ are redundant whenever they are consistent; and $(\mathbf{A}\mathbf{A}^*)^{-1}$ shall be replaced by the pseudo-inverse below.) ADM for problem (19) can be simplified to the iteration:

$$\mathbf{v}_3^{k+1} = \arg\min_{\mathbf{v}} \|\mathbf{v}\|_1 + \frac{\lambda}{2}\|\mathbf{u}_3^k - \mathbf{v} + \frac{1}{\lambda}\mathbf{z}_3^k\|_2^2, \tag{21a}$$

$$\mathbf{u}_3^{k+1} = \mathbf{v}_3^{k+1} - \frac{1}{\lambda}\mathbf{z}_3^k - \mathbf{A}^*(\mathbf{A}\mathbf{A}^*)^{-1}(\mathbf{A}(\mathbf{v}_3^{k+1} - \frac{1}{\lambda}\mathbf{z}_3^k) - \mathbf{b}), \tag{21b}$$

$$\mathbf{z}_3^{k+1} = \mathbf{z}_3^k + \lambda(\mathbf{u}_3^{k+1} - \mathbf{v}_3^{k+1}). \tag{21c}$$

And ADM for problem (20) can be simplified to the iteration:

$$\mathbf{y}_1^{k+1} = \mathcal{P}_{B_1^\infty}(\mathbf{A}^*\mathbf{x}_1^k + \lambda\mathbf{z}_1^k), \tag{22a}$$

$$\mathbf{x}_1^{k+1} = (\mathbf{A}\mathbf{A}^*)^{-1}(\mathbf{A}\mathbf{y}_1^{k+1} - \lambda(\mathbf{A}\mathbf{z}_1^k - \mathbf{b})), \tag{22b}$$

$$\mathbf{z}_1^{k+1} = \mathbf{z}_1^k + \lambda^{-1}(\mathbf{A}^*\mathbf{x}_1^{k+1} - \mathbf{y}_1^{k+1}), \tag{22c}$$

where $\mathcal{P}_{B_1^\infty}$ is the projection onto B_1^∞. Looking into the iteration in (22), we can find that $\mathbf{A}^*\mathbf{x}_1^k$ is used in both the kth and $k+1$st iterations. To save the computation, we can store $\mathbf{A}^*\mathbf{x}_1^k$ as $\mathbf{s}_2^k$. In addition, let $\mathbf{t}_2^k = \mathbf{y}_1^k$ and $\mathbf{z}_2^k = \mathbf{z}_1^k$, we have

$$\mathbf{t}_2^{k+1} = \mathcal{P}_{B_1^\infty}(\mathbf{s}_2^k + \lambda\mathbf{z}_2^k), \tag{23a}$$

$$\mathbf{s}_2^{k+1} = \mathbf{A}^*(\mathbf{A}\mathbf{A}^*)^{-1}(\mathbf{A}(\mathbf{t}_2^{k+1} - \lambda\mathbf{A}\mathbf{z}_2^k) + \lambda\mathbf{b})), \tag{23b}$$

$$\mathbf{z}_2^{k+1} = \mathbf{z}_2^k + \lambda^{-1}(\mathbf{s}_2^{k+1} - \mathbf{t}_2^{k+1}), \tag{23c}$$

which is exactly Algorithm 2 for (20). Thus, Algorithm 2 has smaller complexity than Algorithm 1, i.e., one matrix vector multiplication $\mathbf{A}^*\mathbf{x}_1^k$ is saved from Algorithm 2.

The corollary below follows directly from Theorem 1 by associating (20) and (19) as (P1) and (D2), and (22) and (21) with the iterations of Algorithms 1 and 3, respectively.

Corollary 2. *Suppose that* $\mathbf{A}\mathbf{u} = \mathbf{b}$ *are consistent. Consider ADM iterations* (21) *and* (22). *Let* $\mathbf{u}_3^0 = \mathbf{z}_1^0$ *and* $\mathbf{z}_3^0 = \mathbf{A}^*\mathbf{x}_1^0$. *Then, for* $k \geq 1$, *iterations* (21) *and* (22) *are equivalent. In particular,*

- *From* $\mathbf{x}_1^k$, $\mathbf{z}_1^k$ *in* (22), *we obtain* $\mathbf{u}_3^k$, $\mathbf{z}_3^k$ *in* (21) *through:*

$$\mathbf{u}_3^k = \mathbf{z}_1^k, \qquad \mathbf{z}_3^k = \mathbf{A}^*\mathbf{x}_1^k.$$

- *From* $\mathbf{u}_3^k$, $\mathbf{z}_3^k$ *in* (21), *we obtain* $\mathbf{x}_1^k$, $\mathbf{z}_1^k$ *in* (22) *through:*

$$\mathbf{x}_1^k = (\mathbf{A}\mathbf{A}^*)^{-1}\mathbf{A}\mathbf{z}_3^k, \qquad \mathbf{z}_1^k = \mathbf{u}_3^k.$$

4.3 Example: basis pursuit denoising

The basis pursuit denoising problem is

$$\underset{\mathbf{u}}{\text{minimize}}\ \|\mathbf{u}\|_1 + \frac{1}{2\alpha}\|\mathbf{A}\mathbf{u}-\mathbf{b}\|_2^2 \tag{24}$$

and its Lagrange dual, in the ADM-ready form, is

$$\underset{\mathbf{x},\mathbf{y}}{\text{minimize}}\ -\langle \mathbf{b},\mathbf{x}\rangle + \frac{\alpha}{2}\|\mathbf{x}\|_2^2 + \iota_{B_1^\infty}(\mathbf{y}) \quad \text{subject to } \mathbf{A}^*\mathbf{x}-\mathbf{y}=\mathbf{0}. \tag{25}$$

The iteration of ADM for (25) is

$$\begin{aligned}
\mathbf{y}_1^{k+1} =& \mathcal{P}_{B_1^\infty}(\mathbf{A}^*\mathbf{x}_1^k + \lambda\mathbf{z}_1^k), && \text{(26a)}\\
\mathbf{x}_1^{k+1} =& (\mathbf{A}\mathbf{A}^* + \alpha\lambda\mathbf{I})^{-1}(\mathbf{A}\mathbf{y}_1^{k+1} - \lambda(\mathbf{A}\mathbf{z}_1^k - \mathbf{b})), && \text{(26b)}\\
\mathbf{z}_1^{k+1} =& \mathbf{z}_1^k + \lambda^{-1}(\mathbf{A}^*\mathbf{x}_1^{k+1} - \mathbf{y}_1^{k+1}). && \text{(26c)}
\end{aligned}$$

Looking into the iteration in (26), we can find that $\mathbf{A}^*\mathbf{x}_1^k$ is used in both the kth and $k+1$st iterations. To save the computation, we can store $\mathbf{A}^*\mathbf{x}_1^k$ as $\mathbf{s}_2^k$. In addition, let $\mathbf{t}_2^k = \mathbf{y}_1^k$ and $\mathbf{z}_2^k = \mathbf{z}_1^k$, we have

$$\begin{aligned}
\mathbf{t}_2^{k+1} =& \mathcal{P}_{B_1^\infty}(\mathbf{s}_2^k + \lambda\mathbf{z}_2^k), && \text{(27a)}\\
\mathbf{s}_2^{k+1} =& \mathbf{A}^*(\mathbf{A}\mathbf{A}^* + \alpha\lambda\mathbf{I})^{-1}(\mathbf{A}(\mathbf{t}_2^{k+1} - \lambda\mathbf{z}_2^k) + \lambda\mathbf{b})), && \text{(27b)}\\
\mathbf{z}_2^{k+1} =& \mathbf{z}_2^k + \lambda^{-1}(\mathbf{s}_2^{k+1} - \mathbf{t}_2^{k+1}), && \text{(27c)}
\end{aligned}$$

which is exactly Algorithm 2 for (25). Thus, Algorithm 2 has smaller complexity than Algorithm 1, i.e., one matrix vector multiplication $\mathbf{A}^*\mathbf{x}_1^k$ is saved from Algorithm 2. In addition, if $\mathbf{A}^*\mathbf{A} = \mathbf{I}$, (27b) becomes

$$\mathbf{s}_2^{k+1} = (\alpha\lambda+1)^{-1}(\mathbf{t}_2^{k+1} - \lambda\mathbf{z}_2^k + \lambda\mathbf{A}^*\mathbf{b}), \tag{28}$$

and no matrix vector multiplications is needed during the iteration because $\lambda\mathbf{A}^*\mathbf{b}$ can be precalculated.

The ADM-ready form of the original problem (24) is

$$\underset{\mathbf{u},\mathbf{v}}{\text{minimize}}\ \|\mathbf{v}\|_1 + \frac{1}{2\alpha}\|\mathbf{A}\mathbf{u}-\mathbf{b}\|_2^2 \quad \text{subject to } \mathbf{u}-\mathbf{v}=\mathbf{0}, \tag{29}$$

whose ADM iteration is

$$\begin{aligned}
\mathbf{v}_3^{k+1} =& \arg\min_{\mathbf{v}} \|\mathbf{v}\|_1 + \frac{\lambda}{2}\|\mathbf{u}_3^k - \mathbf{v} + \frac{1}{\lambda}\mathbf{z}_3^k\|_2^2, && \text{(30a)}\\
\mathbf{u}_3^{k+1} =& (\mathbf{A}^*\mathbf{A} + \alpha\lambda\mathbf{I})^{-1}(\mathbf{A}^*\mathbf{b} + \alpha\lambda\mathbf{v}_3^{k+1} - \alpha\mathbf{z}_3^k), && \text{(30b)}\\
\mathbf{z}_3^{k+1} =& \mathbf{z}_3^k + \lambda(\mathbf{u}_3^{k+1} - \mathbf{v}_3^{k+1}). && \text{(30c)}
\end{aligned}$$

The corollary below follows directly from Theorem 1.

Corollary 3. *Consider ADM iterations (26) and (30). Let* $\mathbf{u}_3^0 = \mathbf{z}_1^0$ *and* $\mathbf{z}_3^0 = \mathbf{A}^*\mathbf{x}_1^0$. *For* $k \geq 1$, *ADM on the dual and primal problems (26) and (30) are equivalent in the following way:*

- *From* $\mathbf{x}_1^k$, $\mathbf{z}_1^k$ *in (26), we recover* $\mathbf{u}_3^k$, $\mathbf{z}_3^k$ *in (30) through:*

$$\mathbf{u}_3^k = \mathbf{z}_1^k, \qquad \mathbf{z}_3^k = \mathbf{A}^*\mathbf{x}_1^k.$$

- *From* $\mathbf{u}_3^k$, $\mathbf{z}_3^k$ *in (30), we recover* $\mathbf{x}_1^k$, $\mathbf{z}_1^k$ *in (26) through:*

$$\mathbf{x}_1^k = -(\mathbf{A}\mathbf{u}_3^k - \mathbf{b})/\alpha, \qquad \mathbf{z}_1^k = \mathbf{u}_3^k.$$

Remark 3. *Iteration (30) is different from that of ADM for another ADM-ready form of (24)*

$$\underset{\mathbf{u},\mathbf{v}}{\text{minimize}}\, \|\mathbf{u}\|_1 + \frac{1}{2\alpha}\|\mathbf{v}\|_2^2 \quad \text{subject to } \mathbf{A}\mathbf{u} - \mathbf{v} = \mathbf{b}, \tag{31}$$

which is used in [24]. In general, there are different ADM-ready forms and their ADM algorithms yield different iterates. ADM on one ADM-ready form is equivalent to it on the corresponding dual ADM-ready form.

5 ADM as a primal-dual algorithm on the saddle-point problem

As shown in section 4, ADM on a pair of convex primal and dual problems are equivalent, and there is a connection between $\mathbf{z}_1^k$ in Algorithm 1 and dual variable $\mathbf{u}_3^k$ in Algorithm 3. This primal-dual equivalence naturally suggests that ADM is also equivalent to a primal-dual algorithm involving both primal and dual variables.

We derive problem (P1) into an equivalent primal-dual saddle-point problem (33) as follows:

$$\begin{aligned}
&\min_{\mathbf{y},\mathbf{x}} g(\mathbf{y}) + f(\mathbf{x}) + \iota_{\{(\mathbf{x},\mathbf{y}):\mathbf{A}\mathbf{x}=\mathbf{b}-\mathbf{B}\mathbf{y}\}}(\mathbf{x},\mathbf{y}) \\
=&\min_{\mathbf{y}} g(\mathbf{y}) + F(\mathbf{b}-\mathbf{B}\mathbf{y}) \\
=&\min_{\mathbf{y}}\max_{\mathbf{u}} g(\mathbf{y}) + \langle -\mathbf{u}, \mathbf{b}-\mathbf{B}\mathbf{y}\rangle - F^*(-\mathbf{u}) \qquad (32)\\
=&\min_{\mathbf{y}}\max_{\mathbf{u}} g(\mathbf{y}) + \langle \mathbf{u}, \mathbf{B}\mathbf{y}-\mathbf{b}\rangle - f^*(-\mathbf{A}^*\mathbf{u}). \qquad (33)
\end{aligned}$$

A primal-dual algorithm for solving (33) is described in Algorithm 4. Theorem 2 establishes the equivalence between Algorithms 1 and 4.

Algorithm 4 Primal-dual formulation of ADM on Problem (33)

initialize $\mathbf{u}_4^0$, $\mathbf{u}_4^{-1}$, $\mathbf{y}_4^0$, $\lambda > 0$
for $k = 0, 1, \cdots$ **do**
  $\bar{\mathbf{u}}_4^k = 2\mathbf{u}_4^k - \mathbf{u}_4^{k-1}$
  $\mathbf{y}_4^{k+1} = \arg\min_{\mathbf{y}} g(\mathbf{y}) + (2\lambda)^{-1}\|\mathbf{B}\mathbf{y} - \mathbf{B}\mathbf{y}_4^k + \lambda\bar{\mathbf{u}}_4^k\|_2^2$
  $\mathbf{u}_4^{k+1} = \arg\min_{\mathbf{u}} f^*(-\mathbf{A}^*\mathbf{u}) - \langle \mathbf{u}, \mathbf{B}\mathbf{y}_4^{k+1} - \mathbf{b}\rangle + \lambda/2\|\mathbf{u} - \mathbf{u}_4^k\|_2^2$
end for

Remark 4. *Paper [3] proposed a primal-dual algorithm for (32) and obtained the connection between ADM and that primal-dual algorithm [10]: When* $\mathbf{B} = \mathbf{I}$*, ADM is equivalent to the primal-dual algorithm in [3]; When* $\mathbf{B} \neq \mathbf{I}$*, the primal-dual algorithm is a preconditioned ADM as an additional proximal term* $\delta/2\|\mathbf{y} - \mathbf{y}_4^k\|_2^2 - (2\lambda)^{-1}\|\mathbf{B}\mathbf{y} - \mathbf{B}\mathbf{y}_4^k\|_2^2$ *is added to the subproblem for* $\mathbf{y}_4^{k+1}$*. This is also a special case of inexact ADM in [6]. Our Algorithm 4 is a primal-dual algorithm that is equivalent to ADM in the general case.*

Theorem 2 (Equivalence between Algorithms 1 and 4)**.** *Suppose that* $\mathbf{A}\mathbf{x}_1^0 = \lambda(\mathbf{u}_4^0 - \mathbf{u}_4^{-1}) + \mathbf{b} - \mathbf{B}\mathbf{y}_4^0$ *and* $\mathbf{z}_1^0 = \mathbf{u}_4^0$*. Then, Algorithms 1 and 4 are equivalent with the identities:*

$$\mathbf{A}\mathbf{x}_1^k = \lambda(\mathbf{u}_4^k - \mathbf{u}_4^{k-1}) + \mathbf{b} - \mathbf{B}\mathbf{y}_4^k, \qquad \mathbf{z}_1^k = \mathbf{u}_4^k, \tag{34}$$

for all $k > 0$*.*

Proof. By assumption, (34) holds at iteration $k = 0$.
Proof by induction. Suppose that (34) holds at iteration $k \geq 0$. We shall establish (34) at iteration $k + 1$. From the first step of Algorithm 1, we have

$$\begin{aligned}\mathbf{y}_1^{k+1} &= \arg\min_{\mathbf{y}} g(\mathbf{y}) + (2\lambda)^{-1}\|\mathbf{A}\mathbf{x}_1^k + \mathbf{B}\mathbf{y} - \mathbf{b} + \lambda\mathbf{z}_1^k\|_2^2 \\ &= \arg\min_{\mathbf{y}} g(\mathbf{y}) + (2\lambda)^{-1}\|\lambda(\mathbf{u}_4^k - \mathbf{u}_4^{k-1}) + \mathbf{B}\mathbf{y} - \mathbf{B}\mathbf{y}_4^k + \lambda\mathbf{u}_4^k\|_2^2,\end{aligned}$$

which is the same as the first step in Algorithm 4. Thus we have $\mathbf{y}_1^{k+1} = \mathbf{y}_4^{k+1}$.

Combing the second and third steps of Algorithm 1, we have

$$\mathbf{0} \in \partial f(\mathbf{x}_1^{k+1}) + \lambda^{-1}\mathbf{A}^*(\mathbf{A}\mathbf{x}_1^{k+1} + \mathbf{B}\mathbf{y}_1^{k+1} - \mathbf{b} + \lambda\mathbf{z}_1^k) = \partial f(\mathbf{x}_1^{k+1}) + \mathbf{A}^*\mathbf{z}_1^{k+1}.$$

Therefore,

$$
\begin{aligned}
& \mathbf{x}_1^{k+1} \in \partial f^*(-\mathbf{A}^*\mathbf{z}_1^{k+1}) \\
\Longrightarrow\ & \mathbf{A}\mathbf{x}_1^{k+1} \in \partial F^*(-\mathbf{z}_1^{k+1}) \\
\Longleftrightarrow\ & \lambda(\mathbf{z}_1^{k+1} - \mathbf{z}_1^k) + \mathbf{b} - \mathbf{B}\mathbf{y}_1^{k+1} \in \partial F^*(-\mathbf{z}_1^{k+1}) \\
\Longleftrightarrow\ & \mathbf{z}_1^{k+1} = \arg\min_{\mathbf{z}} F^*(-\mathbf{z}) - \langle \mathbf{z}, \mathbf{B}\mathbf{y}_1^{k+1} - \mathbf{b}\rangle + \lambda/2\|\mathbf{z} - \mathbf{z}_1^k\|_2^2 \\
\Longleftrightarrow\ & \mathbf{z}_1^{k+1} = \arg\min_{\mathbf{z}} f^*(-\mathbf{A}^*\mathbf{z}) - \langle \mathbf{z}, \mathbf{B}\mathbf{y}_4^{k+1} - \mathbf{b}\rangle + \lambda/2\|\mathbf{z} - \mathbf{u}_4^k\|_2^2,
\end{aligned}
$$

where the last line is the second step of Algorithm 4. Therefore, we have $\mathbf{z}_1^{k+1} = \mathbf{u}_4^{k+1}$ and $\mathbf{A}\mathbf{x}_1^{k+1} = \lambda(\mathbf{z}_1^{k+1} - \mathbf{z}_1^k) + \mathbf{b} - \mathbf{B}\mathbf{y}_1^{k+1} = \lambda(\mathbf{u}_4^{k+1} - \mathbf{u}_4^k) + \mathbf{b} - \mathbf{B}\mathbf{y}_4^{k+1}$. □

6 Equivalence of ADM for different orders

In both problem (P1) and Algorithm 1, we can swap $\mathbf{x}$ and $\mathbf{y}$ and obtain Algorithm 5, which is still an algorithm of ADM. In general, the two algorithms are different. In this section, we show that for a certain type of functions f (or g), Algorithms 1 and 5 become equivalent.

Algorithm 5 ADM2 on (P1)

initialize $\mathbf{y}_5^0$, $\mathbf{z}_5^0$, $\lambda > 0$

for $k = 0, 1, \cdots$ **do**

$\mathbf{x}_5^{k+1} = \arg\min_{\mathbf{x}} f(\mathbf{x}) + (2\lambda)^{-1}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y}_5^k - \mathbf{b} + \lambda\mathbf{z}_5^k\|_2^2$

$\mathbf{y}_5^{k+1} = \arg\min_{\mathbf{y}} g(\mathbf{y}) + (2\lambda)^{-1}\|\mathbf{A}\mathbf{x}_5^{k+1} + \mathbf{B}\mathbf{y} - \mathbf{b} + \lambda\mathbf{z}_5^k\|_2^2$

$\mathbf{z}_5^{k+1} = \mathbf{z}_5^k + \lambda^{-1}(\mathbf{A}\mathbf{x}_5^{k+1} + \mathbf{B}\mathbf{y}_5^{k+1} - \mathbf{b})$

end for

The assumption that we need is that either $\mathbf{prox}_{F(\cdot)}$ or $\mathbf{prox}_{G(\cdot)}$ is affine (cf. (7) for the definitions of F and G).

Definition 3. *A mapping T is affine if, for any $\mathbf{r}_1$ and $\mathbf{r}_2$,*

$$T\left(\frac{1}{2}\mathbf{r}_1 + \frac{1}{2}\mathbf{r}_2\right) = \frac{1}{2}T\mathbf{r}_1 + \frac{1}{2}T\mathbf{r}_2.$$

Proposition 1. *Let $\lambda > 0$. The following statements are equivalent:*

1. $\mathbf{prox}_{G(\cdot)}$ *is affine;*
2. $\mathbf{prox}_{\lambda G(\cdot)}$ *is affine;*

3. $a\mathbf{prox}_{G(\cdot)} \circ b\mathbf{I} + c\mathbf{I}$ *is affine for any scalars* a, b *and* c;

4. $\mathbf{prox}_{G^*(\cdot)}$ *is affine;*

5. G *is convex quadratic (or, affine or constant) and its domain* $\mathrm{dom}(G)$ *is either* $\mathcal{G}$ *or the intersection of hyperplanes in* $\mathcal{G}$.

In addition, if function g *is convex quadratic and its domain is the intersection of hyperplanes, then function* G *defined in* (7b) *satisfies Part 5 above.*

Proposition 2. *If* $\mathbf{prox}_{G(\cdot)}$ *is affine, then the following holds for any* $\mathbf{r}_1$ *and* $\mathbf{r}_2$*:*

$$\mathbf{prox}_{G(\cdot)}(2\mathbf{r}_1 - \mathbf{r}_2) = 2\mathbf{prox}_{G(\cdot)}\mathbf{r}_1 - \mathbf{prox}_{G(\cdot)}\mathbf{r}_2. \tag{35}$$

Proof. Equation (35) is obtained by defining $\bar{\mathbf{r}}_1 = 2\mathbf{r}_1 - \mathbf{r}_2$ and $\bar{\mathbf{r}}_2 := \mathbf{r}_2$ and rearranging

$$\mathbf{prox}_{G(\cdot)}\left(\frac{1}{2}\bar{\mathbf{r}}_1 + \frac{1}{2}\bar{\mathbf{r}}_2\right) = \frac{1}{2}\mathbf{prox}_{G(\cdot)}\bar{\mathbf{r}}_1 + \frac{1}{2}\mathbf{prox}_{G(\cdot)}\bar{\mathbf{r}}_2.$$

□

Theorem 3 (Equivalence of Algorithms 1 and 5)**.**

1. *Assume that* $\mathbf{prox}_{\lambda G(\cdot)}$ *is affine. Given the sequences* $\mathbf{y}_5^k$, $\mathbf{z}_5^k$, *and* $\mathbf{x}_5^k$ *of Algorithm 5, if* $\mathbf{y}_5^0$ *and* $\mathbf{z}_5^0$ *satisfy* $-\mathbf{z}_5^0 \in \partial G(\mathbf{B}\mathbf{y}_5^0 - \mathbf{b})$, *then we can initialize Algorithm 1 with* $\mathbf{x}_1^0 = \mathbf{x}_5^1$ *and* $\mathbf{z}_1^0 = \mathbf{z}_5^0 + \lambda^{-1}(\mathbf{A}\mathbf{x}_5^1 + \mathbf{B}\mathbf{y}_5^0 - \mathbf{b})$, *and recover the sequences* $\mathbf{x}_1^k$ *and* $\mathbf{z}_1^k$ *of Algorithm 1 through*

$$\mathbf{x}_1^k = \mathbf{x}_5^{k+1}, \tag{36a}$$
$$\mathbf{z}_1^k = \mathbf{z}_5^k + \lambda^{-1}(\mathbf{A}\mathbf{x}_5^{k+1} + \mathbf{B}\mathbf{y}_5^k - \mathbf{b}). \tag{36b}$$

2. *Assume that* $\mathbf{prox}_{\lambda F(\cdot)}$ *is affine. Given the sequences* $\mathbf{x}_1^k$, $\mathbf{z}_1^k$, *and* $\mathbf{y}_1^k$ *of Algorithm 1, if* $\mathbf{x}_1^0$ *and* $\mathbf{z}_1^0$ *satisfy* $-\mathbf{z}_1^0 \in \partial F(\mathbf{A}\mathbf{x}_1^0)$, *then we can initialize Algorithm 5 with* $\mathbf{y}_5^0 = \mathbf{y}_1^1$ *and* $\mathbf{z}_5^0 = \mathbf{z}_1^0 + \lambda^{-1}(\mathbf{A}\mathbf{x}_1^0 + \mathbf{B}\mathbf{y}_1^1 - \mathbf{b})$, *and recover the sequences* $\mathbf{y}_5^k$ *and* $\mathbf{z}_5^k$ *of Algorithm 5 through*

$$\mathbf{y}_5^k = \mathbf{y}_1^{k+1}, \tag{37a}$$
$$\mathbf{z}_5^k = \mathbf{z}_1^k + \lambda^{-1}(\mathbf{A}\mathbf{x}_1^k + \mathbf{B}\mathbf{y}_1^{k+1} - \mathbf{b}). \tag{37b}$$

Proof. We prove Part 1 only by induction. (The proof for the other part is similar.) The initialization of Algorithm 1 clearly follows (36) at $k = 0$. Suppose that (36) holds at $k \geq 0$. We shall show that (36) holds at $k + 1$. We first show from the affine property of $\mathbf{prox}_{\lambda G(\cdot)}$ that

$$\mathbf{B}\mathbf{y}_1^{k+1} = 2\mathbf{B}\mathbf{y}_5^{k+1} - \mathbf{B}\mathbf{y}_5^k. \tag{38}$$

The optimization subproblems for $\mathbf{y}_1$ and $\mathbf{y}_5$ in Algorithms 1 and 5, respectively, are as follows:

$$\mathbf{y}_1^{k+1} = \arg\min_{\mathbf{y}} g(\mathbf{y}) + (2\lambda)^{-1}\|\mathbf{A}\mathbf{x}_1^k + \mathbf{B}\mathbf{y} - \mathbf{b} + \lambda\mathbf{z}_1^k\|_2^2,$$
$$\mathbf{y}_5^{k+1} = \arg\min_{\mathbf{y}} g(\mathbf{y}) + (2\lambda)^{-1}\|\mathbf{A}\mathbf{x}_5^{k+1} + \mathbf{B}\mathbf{y} - \mathbf{b} + \lambda\mathbf{z}_5^k\|_2^2.$$

Following the definition of G in (7), we have

$$\mathbf{B}\mathbf{y}_1^{k+1} - \mathbf{b} = \mathbf{prox}_{\lambda G(\cdot)}(-\mathbf{A}\mathbf{x}_1^k - \lambda\mathbf{z}_1^k), \tag{39a}$$
$$\mathbf{B}\mathbf{y}_5^{k+1} - \mathbf{b} = \mathbf{prox}_{\lambda G(\cdot)}(-\mathbf{A}\mathbf{x}_5^{k+1} - \lambda\mathbf{z}_5^k), \tag{39b}$$
$$\mathbf{B}\mathbf{y}_5^{k} - \mathbf{b} = \mathbf{prox}_{\lambda G(\cdot)}(-\mathbf{A}\mathbf{x}_5^{k} - \lambda\mathbf{z}_5^{k-1}). \tag{39c}$$

The third step of Algorithm 5 is

$$\mathbf{z}_5^k = \mathbf{z}_5^{k-1} + \lambda^{-1}(\mathbf{A}\mathbf{x}_5^k + \mathbf{B}\mathbf{y}_5^k - \mathbf{b}). \tag{40}$$

(Note that for $k = 0$, the assumption $-\mathbf{z}_5^0 \in \partial G(\mathbf{B}\mathbf{y}_5^0 - \mathbf{b})$ ensures the existence of $\mathbf{z}_5^{-1}$ in (39c) and (40).) Then, (36) and (40) give us

$$\begin{aligned}\mathbf{A}\mathbf{x}_1^k + \lambda\mathbf{z}_1^k &\overset{(36)}{=} \mathbf{A}\mathbf{x}_5^{k+1} + \lambda\mathbf{z}_5^k + \mathbf{A}\mathbf{x}_5^{k+1} + \mathbf{B}\mathbf{y}_5^k - \mathbf{b}\\ &= 2(\mathbf{A}\mathbf{x}_5^{k+1} + \lambda\mathbf{z}_5^k) - (\lambda\mathbf{z}_5^k - \mathbf{B}\mathbf{y}_5^k + \mathbf{b})\\ &\overset{(40)}{=} 2(\mathbf{A}\mathbf{x}_5^{k+1} + \lambda\mathbf{z}_5^k) - (\mathbf{A}\mathbf{x}_5^k + \lambda\mathbf{z}_5^{k-1}).\end{aligned}$$

Since $\mathbf{prox}_{\lambda G(\cdot)}$ is affine, we have (35). Once we plug in (35): $\mathbf{r}_1 = -\mathbf{A}\mathbf{x}_5^{k+1} - \lambda\mathbf{z}_5^k$, $\mathbf{r}_2 = -\mathbf{A}\mathbf{x}_5^k - \lambda\mathbf{z}_5^{k-1}$, and $2\mathbf{r}_1 - \mathbf{r}_2 = -\mathbf{A}\mathbf{x}_1^k - \lambda\mathbf{z}_1^k$ and then apply (39), we obtain (38).

Next, the third step of Algorithm 5 and (38) give us

$$\begin{aligned}\mathbf{B}\mathbf{y}_1^{k+1} - \mathbf{b} + \lambda\mathbf{z}_1^k &\overset{(38)}{=} 2(\mathbf{B}\mathbf{y}_5^{k+1} - \mathbf{b}) - (\mathbf{B}\mathbf{y}_5^k - \mathbf{b}) + \lambda\mathbf{z}_5^k + (\mathbf{A}\mathbf{x}_5^{k+1} + \mathbf{B}\mathbf{y}_5^k - \mathbf{b})\\ &= (\mathbf{B}\mathbf{y}_5^{k+1} - \mathbf{b}) + \lambda\mathbf{z}_5^k + (\mathbf{A}\mathbf{x}_5^{k+1} + \mathbf{B}\mathbf{y}_5^{k+1} - \mathbf{b})\\ &= (\mathbf{B}\mathbf{y}_5^{k+1} - \mathbf{b}) + \lambda\mathbf{z}_5^{k+1}.\end{aligned}$$

This identity shows that the updates of $\mathbf{x}_1^{k+1}$ and $\mathbf{x}_5^{k+2}$ in Algorithms 1 and 5, respectively, have identical data, and therefore, we recover $\mathbf{x}_1^{k+1} = \mathbf{x}_5^{k+2}$.

Lastly, from the third step of Algorithm 1 and the identities above, it follows that

$$\begin{aligned}\mathbf{z}_1^{k+1} &= \mathbf{z}_1^k + \lambda^{-1}(\mathbf{A}\mathbf{x}_1^{k+1} + \mathbf{B}\mathbf{y}_1^{k+1} - \mathbf{b})\\ &= \mathbf{z}_1^k + \lambda^{-1}\left(\mathbf{A}\mathbf{x}_5^{k+2} + (\mathbf{B}\mathbf{y}_5^{k+1} - \mathbf{b} + \lambda\mathbf{z}_5^{k+1} - \lambda\mathbf{z}_1^k)\right)\\ &= \mathbf{z}_5^{k+1} + \lambda^{-1}(\mathbf{A}\mathbf{x}_5^{k+2} + \mathbf{B}\mathbf{y}_5^{k+1} - \mathbf{b}).\end{aligned}$$

Therefore, we obtain (36) at $k+1$. □

Remark 5. *We can avoid the technical condition* $-\mathbf{z}_5^0 \in \partial G(\mathbf{B}\mathbf{y}_5^0 - \mathbf{b})$ *on Algorithm 5 in Part 1 of Theorem 3. When it does not hold, we can use the always-true relation* $-\mathbf{z}_5^1 \in \partial G(\mathbf{B}\mathbf{y}_5^1 - \mathbf{b})$ *instead; correspondingly, we shall add 1 iteration to the iterates of Algorithm 5, namely, initialize Algorithm 1 with* $\mathbf{x}_1^0 = \mathbf{x}_5^2$ *and* $\mathbf{z}_1^0 = \mathbf{z}_5^1 + \lambda^{-1}(\mathbf{A}\mathbf{x}_5^2 + \mathbf{B}\mathbf{y}_5^1 - \mathbf{b})$ *and recover the sequences* $\mathbf{x}_1^k$ *and* $\mathbf{z}_1^k$ *of Algorithm 1 through*

$$\mathbf{x}_1^k = \mathbf{x}_5^{k+2}, \tag{41a}$$

$$\mathbf{z}_1^k = \mathbf{z}_5^{k+1} + \lambda^{-1}(\mathbf{A}\mathbf{x}_5^{k+2} + \mathbf{B}\mathbf{y}_5^{k+1} - \mathbf{b}). \tag{41b}$$

Similar arguments apply to the other part of Theorem 3.

7 Primal-dual equivalence of RPRS

In this section, we consider the following convex problem:

$$\underset{\mathbf{x}}{\text{minimize}} \quad f(\mathbf{x}) + g(\mathbf{A}\mathbf{x}), \tag{P3}$$

and its corresponding Lagrangian dual

$$\underset{\mathbf{v}}{\text{minimize}} \quad f^*(\mathbf{A}^*\mathbf{v}) + g^*(-\mathbf{v}). \tag{D3}$$

In addition, we introduce another primal-dual pair equivalent to (P3)-(D3):

$$\underset{\mathbf{y}}{\text{minimize}} \quad (f^* \circ \mathbf{A}^*)^*(\mathbf{y}) + g(\mathbf{y}), \tag{P4}$$

$$\underset{\mathbf{u}}{\text{minimize}} \quad f^*(\mathbf{u}) + (g \circ \mathbf{A})^*(-\mathbf{u}). \tag{D4}$$

Lemma 2 below will establish the equivalence between the two primal-dual pairs.

Remark 6. *When* $\mathbf{A} = \mathbf{I}$*, we have* $(f^* \circ \mathbf{A}^*)^* = f$*, and problem (P3) is exactly the same as problem (P4). Similarly, problem (D3) is exactly the same as problem (D4).*

Lemma 2. *Problems (P3) and (P4) are equivalent in the following sense:*

- *Given any solution* $\mathbf{x}^*$ *to (P3),* $\mathbf{y}^* = \mathbf{A}\mathbf{x}^*$ *is a solution to (P4),*
- *Given any solution* $\mathbf{y}^*$ *to (P4),* $\mathbf{x}^* \in \underset{\mathbf{x}:\mathbf{A}\mathbf{x}=\mathbf{y}^*}{\arg\min} f(\mathbf{x})$ *is a solution to (P3).*

The equivalence between problems (D3) and (D4) is similar:

- *Given any solution* $\mathbf{v}^*$ *to (D3),* $\mathbf{A}^*\mathbf{v}^*$ *is a solution to (D4),*
- *Given any solution* $\mathbf{u}^*$ *to (D4),* $\mathbf{v}^* \in \underset{\mathbf{v}:\mathbf{A}^*\mathbf{v}=\mathbf{u}^*}{\arg\min} g^*(-\mathbf{v})$ *is a solution to (D3).*

Proof. We prove only the equivalence of (P3) and (P4), the proof for the equivalence of (D3) and (D4) is similar.

Part 1: If $\mathbf{x}^*$ is a solution to (P3), we have $\mathbf{0} \in \partial f(\mathbf{x}^*) + \mathbf{A}^* \partial g(\mathbf{A}\mathbf{x}^*)$. Assume that there exists $\mathbf{q}$ such that $-\mathbf{q} \in \partial g(\mathbf{A}\mathbf{x}^*)$ and $\mathbf{A}^*\mathbf{q} \in \partial f(\mathbf{x}^*)$. Then we have

$$\begin{aligned}\mathbf{A}^*\mathbf{q} \in \partial f(\mathbf{x}^*) \Longleftrightarrow & \mathbf{x}^* \in \partial f^*(\mathbf{A}^*\mathbf{q}) \\ \Longrightarrow & \mathbf{A}\mathbf{x}^* \in \mathbf{A}\partial f^*(\mathbf{A}^*\mathbf{q}) = \partial(f^* \circ \mathbf{A}^*)(\mathbf{q}) \\ \Longleftrightarrow & \mathbf{q} \in \partial(f^* \circ \mathbf{A}^*)^*(\mathbf{A}\mathbf{x}^*).\end{aligned}$$

Therefore,

$$\mathbf{0} \in \partial(f^* \circ \mathbf{A}^*)^*(\mathbf{A}\mathbf{x}^*) + \partial g(\mathbf{A}\mathbf{x}^*)$$

and $\mathbf{A}\mathbf{x}^*$ is a solution to (P4).

Part 2: If $\mathbf{y}^*$ is a solution to (P4), the optimality condition gives us

$$\mathbf{0} \in \partial(f^* \circ \mathbf{A}^*)^*(\mathbf{y}^*) + \partial g(\mathbf{y}^*).$$

Assume that there exists $\mathbf{q}$ such that $-\mathbf{q} \in \partial g(\mathbf{y}^*)$ and $\mathbf{q} \in \partial(f^* \circ \mathbf{A}^*)^*(\mathbf{y}^*)$. Then we have

$$\mathbf{q} \in \partial(f^* \circ \mathbf{A}^*)^*(\mathbf{y}^*) \Longleftrightarrow \mathbf{y}^* \in \partial(f^* \circ \mathbf{A}^*)(\mathbf{q}). \tag{42}$$

Consider the following optimization problem for finding $\mathbf{x}^*$ from $\mathbf{y}^*$

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}) \qquad \text{subject to } \mathbf{A}\mathbf{x} = \mathbf{y}^*,$$

and the corresponding dual problem

$$\underset{\mathbf{v}}{\text{maximize}} -f^*(\mathbf{A}^*\mathbf{v}) + \langle \mathbf{v}, \mathbf{y}^* \rangle.$$

It is easy to obtain from (42) that $\mathbf{q}$ is a solution of the dual problem. The optimal duality gap is zero and the strong duality gives us

$$f(\mathbf{x}^*) = f(\mathbf{x}^*) - \langle \mathbf{q}, \mathbf{A}\mathbf{x}^* - \mathbf{y}^* \rangle = -f^*(\mathbf{A}^*\mathbf{q}) + \langle \mathbf{q}, \mathbf{y}^* \rangle. \tag{43}$$

Thus $\mathbf{x}^*$ is a solution of $\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}) - \langle \mathbf{A}^*\mathbf{q}, \mathbf{x} \rangle$ and

$$\mathbf{A}^*\mathbf{q} \in \partial f(\mathbf{x}^*) \Longleftrightarrow \mathbf{0} \in \partial f(\mathbf{x}^*) - \mathbf{A}^*\mathbf{q}. \tag{44}$$

Because $-\mathbf{q} \in \partial g(\mathbf{y}^*) = \partial g(\mathbf{A}\mathbf{x}^*)$,

$$\mathbf{0} \in \partial f(\mathbf{x}^*) + \mathbf{A}^* \partial \mathbf{g}(\mathbf{A}\mathbf{x}^*) = \partial f(\mathbf{x}^*) + \partial(\mathbf{g} \circ \mathbf{A})(\mathbf{x}^*) \tag{45}$$

Therefore $\mathbf{x}^*$ is a solution of (P3). □

Next we will show the equivalence between the RPRS to the primal and dual problems:

$$\boxed{\text{RPRS on (P3)}} \Longleftrightarrow \boxed{\text{RPRS on (D4)}}$$
$$\boxed{\text{RPRS on (P4)}} \Longleftrightarrow \boxed{\text{RPRS on (D3)}}$$

We describe the RPRS on (P3) in Algorithm 6, and the RPRS on other problems can be obtained in the same way.

Algorithm 6 RPRS on (P3)

initialize $\mathbf{w}^0$, $\lambda > 0$, $0 < \alpha \leq 1$.
for $k = 0, 1, \cdots$ **do**
  $\mathbf{x}^{k+1} = \mathbf{prox}_{\lambda f(\cdot)}\mathbf{w}^k$
  $\mathbf{w}^{k+1} = (1-\alpha)\mathbf{w}^k + \alpha(2\mathbf{prox}_{\lambda g\circ\mathbf{A}(\cdot)} - \mathbf{I})(2\mathbf{x}^{k+1} - \mathbf{w}^k)$
end for

Theorem 4. *[Primal-dual equivalence of RPRS] RPRS on (P3) is equivalent to RPRS on (D4). RPRS on (P4) is equivalent to RPRS on (D3).*

Before proving this theorem, we introduce a lemma, which was also given in [8, Proposition 3.34]. Here, we prove it in a different way using the generalized Moreau decomposition.

Lemma 3. *For $\lambda > 0$, we have*

$$\lambda^{-1}(2\mathbf{prox}_{\lambda F(\cdot)} - \mathbf{I})\mathbf{w} = (\mathbf{I} - 2\mathbf{prox}_{\lambda^{-1}F^*(\cdot)})(\mathbf{w}/\lambda) = (2\mathbf{prox}_{\lambda^{-1}F^*(-\cdot)} - \mathbf{I})(-\mathbf{w}/\lambda). \tag{46}$$

Proof. We prove it using the generalized Moreau decomposition [11, Theorem 2.3.1]

$$\mathbf{w} = \mathbf{prox}_{\lambda F(\cdot)}(\mathbf{w}) + \lambda\mathbf{prox}_{\lambda^{-1}F^*(\cdot)}(\mathbf{w}/\lambda). \tag{47}$$

Using the generalized Moreau decomposition, we have

$$\begin{aligned}\lambda^{-1}(2\mathbf{prox}_{\lambda F(\cdot)} - \mathbf{I})\mathbf{w} =& 2\lambda^{-1}\mathbf{prox}_{\lambda F(\cdot)}(\mathbf{w}) - \mathbf{w}/\lambda \overset{(47)}{=} 2\lambda^{-1}(\mathbf{w} - \lambda\mathbf{prox}_{\lambda^{-1}F^*(\cdot)}(\mathbf{w}/\lambda)) - \mathbf{w}/\lambda \\ =& \mathbf{w}/\lambda - 2\mathbf{prox}_{\lambda^{-1}F^*(\cdot)}(\mathbf{w}/\lambda) = (\mathbf{I} - 2\mathbf{prox}_{\lambda^{-1}F^*(\cdot)})(\mathbf{w}/\lambda).\end{aligned}$$

The last equality of (46) comes from

$$\mathbf{prox}_{\lambda^{-1}F^*(-\cdot)}(-\mathbf{w}/\lambda) = -\mathbf{prox}_{\lambda^{-1}F^*(\cdot)}(\mathbf{w}/\lambda).$$

□

Proof of Theorem 4. We will prove only the equivalence of RPRS on (P3) and (D4). The proof for the other equivalence is the same. The RPRS on (P3) and (D4) can be formulated as

$$\mathbf{w}_1^{k+1} = (1-\alpha)\mathbf{w}_1^k + \alpha(2\mathbf{prox}_{\lambda g\circ\mathbf{A}(\cdot)} - \mathbf{I})(2\mathbf{prox}_{\lambda f(\cdot)} - \mathbf{I})\mathbf{w}_1^k, \tag{48}$$

and

$$\mathbf{w}_2^{k+1} = (1-\alpha)\mathbf{w}_2^k + \alpha(2\mathbf{prox}_{\lambda^{-1}(g\circ\mathbf{A})^*(-\cdot)} - \mathbf{I})(2\mathbf{prox}_{\lambda^{-1}f^*(\cdot)} - \mathbf{I})\mathbf{w}_2^k, \tag{49}$$

respectively. In addition, we can recover the variables $\mathbf{x}^k$ (or $\mathbf{v}^k$) from $\mathbf{w}_1^k$ (or $\mathbf{w}_2^k$) using the following forms:

$$\mathbf{x}^{k+1} = \mathbf{prox}_{\lambda f(\cdot)}\mathbf{w}_1^k, \tag{50}$$

$$\mathbf{v}^{k+1} = \mathbf{prox}_{\lambda^{-1}f^*(\cdot)}\mathbf{w}_2^k. \tag{51}$$

Proof by induction. Suppose $\mathbf{w}_2^k = \mathbf{w}_1^k/\lambda$ holds. We next show that $\mathbf{w}_2^{k+1} = \mathbf{w}_1^{k+1}/\lambda$.

$$\begin{aligned}
\mathbf{w}_2^{k+1} =& (1-\alpha)\mathbf{w}_1^k/\lambda + \alpha(2\mathbf{prox}_{\lambda^{-1}(g\circ\mathbf{A})^*(-\cdot)} - \mathbf{I})(2\mathbf{prox}_{\lambda^{-1}f^*(\cdot)} - \mathbf{I})(\mathbf{w}_1^k/\lambda)\\
\overset{(46)}{=}& (1-\alpha)\mathbf{w}_1^k/\lambda + \alpha(2\mathbf{prox}_{\lambda^{-1}(g\circ\mathbf{A})^*(-\cdot)} - \mathbf{I})(-\lambda^{-1}(2\mathbf{prox}_{\lambda f(\cdot)} - \mathbf{I})\mathbf{w}_1^k)\\
\overset{(46)}{=}& (1-\alpha)\mathbf{w}_1^k/\lambda + \alpha\lambda^{-1}(2\mathbf{prox}_{\lambda(g\circ\mathbf{A})(-\cdot)} - \mathbf{I})(2\mathbf{prox}_{\lambda f(\cdot)} - \mathbf{I})\mathbf{w}_1^k\\
=& \lambda^{-1}[(1-\alpha)\mathbf{w}_1^k + \alpha(2\mathbf{prox}_{\lambda(g\circ\mathbf{A})(\cdot)} - \mathbf{I})(2\mathbf{prox}_{\lambda f(\cdot)} - \mathbf{I})\mathbf{w}_1^k]\\
=& \mathbf{w}_1^{k+1}/\lambda.
\end{aligned}$$

In addition we have

$$\begin{aligned}
\mathbf{x}^{k+1} + \lambda\mathbf{v}^{k+1} =& \mathbf{prox}_{\lambda f(\cdot)}\mathbf{w}_1^k + \lambda\mathbf{prox}_{\lambda^{-1}f^*(\cdot)}\mathbf{w}_2^k\\
=& \mathbf{prox}_{\lambda f(\cdot)}\mathbf{w}_1^k + \lambda\mathbf{prox}_{\lambda^{-1}f^*(\cdot)}(\mathbf{w}_1^k/\lambda) = \mathbf{w}_1^k.
\end{aligned}$$

□

Remark 7. *Eckstein showed in [8, Chapter 3.5] that DRS/PRS on (P3) is equivalent to DRS/PRS on (D3) when* $\mathbf{A} = \mathbf{I}$. *This special case can be obtained from this theorem immediately because when* $\mathbf{A} = \mathbf{I}$, *(D3) is exactly the same as (D4) and we have*

DRS/PRS on (P3) $\Longleftrightarrow$ DRS/PRS on (D4) $\Longleftrightarrow$ DRS/PRS on (D3) $\Longleftrightarrow$ DRS/PRS on (P4).

Remark 8. *In order to make sure that RPRS on the primal and dual problems are equivalent, the initial conditions and parameters have to satisfy conditions described in the proof of Theorem 4. We need the initial condition to satisfy* $\mathbf{w}_2^0 = \mathbf{w}_1^0/\lambda$ *and the parameter for RPRS on the dual problem has to be chosen as* λ^{-1}, *see the differences in (48) and (49).*

8 Application: total variation image denoising

ADM (or split Bregman [16]) has been applied on many image processing applications, and we apply the previous equivalence results of ADM to derive several equivalent algorithms for total variation denoising.

The total variation (ROF model [21]) applied on image denoising is

$$\underset{x\in BV(\Omega)}{\text{minimize}} \int_\Omega |Dx| + \frac{\alpha}{2}\|x-b\|_2^2$$

where x stands for an image, and $BV(\Omega)$ is the set of all bounded variation functions on Ω. The first term is known as the total variation of x, minimizing which tends to yield a piece-wise constant solution. The discrete version is as follows:

$$\underset{\mathbf{x}}{\text{minimize}}\, \|\nabla \mathbf{x}\|_{2,1} + \frac{\alpha}{2}\|\mathbf{x}-\mathbf{b}\|_2^2.$$

Without loss of generality, we consider the two-dimensional image $\mathbf{x}$, and the discrete total variation $\|\nabla \mathbf{x}\|_{2,1}$ of image $\mathbf{x}$ is defined as

$$\|\nabla \mathbf{x}\|_{2,1} = \sum_{ij} |(\nabla \mathbf{x})_{ij}|,$$

where $|\cdot|$ is the 2-norm of a vector. The equivalent ADM-ready form [16, Equation (3.1)] is

$$\underset{\mathbf{x},\mathbf{y}}{\text{minimize}}\ \|\mathbf{y}\|_{2,1} + \frac{\alpha}{2}\|\mathbf{x}-\mathbf{b}\|_2^2 \qquad \text{subject to } \mathbf{y}-\nabla\mathbf{x}=\mathbf{0}, \tag{52}$$

and its dual problem in ADM-ready form [2, Equation (8)] is

$$\underset{\mathbf{v},\mathbf{u}}{\text{minimize}}\ \frac{1}{2\alpha}\|\text{div}\ \mathbf{u}+\alpha\mathbf{b}\|_2^2 + \iota_{\{\mathbf{v}:\|\mathbf{v}\|_{2,\infty}\le 1\}}(\mathbf{v}) \qquad \text{subject to } \mathbf{u}-\mathbf{v}=\mathbf{0}, \tag{53}$$

where $\|\mathbf{v}\|_{2,\infty} = \max_{ij} |(\mathbf{v})_{ij}|$. In addition, the equivalent saddle-point problem is

$$\underset{\mathbf{x}}{\text{minimize}}\,\underset{\mathbf{v}}{\text{maximize}} \quad \frac{1}{2\alpha}\|\mathbf{x}-\mathbf{b}\|_2^2 + \langle \mathbf{v}, \nabla\mathbf{x}\rangle - \iota_{\{\mathbf{v}:\|\mathbf{v}\|_{2,\infty}\le 1\}}(\mathbf{v}). \tag{54}$$

We list the following equivalent algorithms for solving the total variation image denoising problem. The equivalence result stated in Corollary 4 can be obtained from theorems 1-3.

1. Algorithm 1 (primal ADM) on (52) is

$$\mathbf{x}_1^{k+1} = \arg\min_{\mathbf{x}} \frac{\alpha}{2}\|\mathbf{x}-\mathbf{b}\|_2^2 + (2\lambda)^{-1}\|\nabla\mathbf{x}-\mathbf{y}_1^k+\lambda\mathbf{z}_1^k\|_2^2, \tag{55a}$$

$$\mathbf{y}_1^{k+1} = \arg\min_{\mathbf{y}} \|\mathbf{y}\|_{2,1} + (2\lambda)^{-1}\|\nabla\mathbf{x}_1^{k+1}-\mathbf{y}+\lambda\mathbf{z}_1^k\|_2^2, \tag{55b}$$

$$\mathbf{z}_1^{k+1} = \mathbf{z}_1^k + \lambda^{-1}(\nabla\mathbf{x}_1^{k+1}-\mathbf{y}_1^{k+1}). \tag{55c}$$

2. Algorithm 3 (dual ADM) on (53) is

$$\mathbf{u}_3^{k+1} = \arg\min_{\mathbf{u}} \frac{1}{2\alpha}\|\text{div}\ \mathbf{u}+\alpha\mathbf{b}\|_2^2 + \frac{\lambda}{2}\|\mathbf{v}_3^k-\mathbf{u}+\lambda^{-1}\mathbf{z}_3^k\|_2^2, \tag{56a}$$

$$\mathbf{v}_3^{k+1} = \arg\min_{\mathbf{v}} \iota_{\{\mathbf{v}:\|\mathbf{v}\|_{2,\infty}\le 1\}}(\mathbf{v}) + \frac{\lambda}{2}\|\mathbf{v}-\mathbf{u}_3^{k+1}+\lambda^{-1}\mathbf{z}_3^k\|_2^2, \tag{56b}$$

$$\mathbf{z}_3^{k+1} = \mathbf{z}_3^k + \lambda(\mathbf{v}_3^{k+1}-\mathbf{u}_3^{k+1}). \tag{56c}$$

3. Algorithm 4 (primal-dual) on (54) is

$$\bar{\mathbf{v}}_4^k = 2\mathbf{v}_4^k - \mathbf{v}_4^{k-1}, \tag{57a}$$

$$\mathbf{x}_4^{k+1} = \arg\min_{\mathbf{x}} \frac{\alpha}{2}\|\mathbf{x}-\mathbf{b}\|_2^2 + (2\lambda)^{-1}\|\nabla\mathbf{x} - \nabla\mathbf{x}_4^k + \lambda\bar{\mathbf{v}}_4^k\|_2^2, \tag{57b}$$

$$\mathbf{v}_4^{k+1} = \arg\min_{\mathbf{v}} \iota_{\{\mathbf{v}:\|\mathbf{v}\|_{2,\infty}\le 1\}}(\mathbf{v}) - \langle \mathbf{v}, \nabla\mathbf{x}_4^{k+1}\rangle + \frac{\lambda}{2}\|\mathbf{v}-\mathbf{v}^k\|_2^2. \tag{57c}$$

4. Algorithm 5 (primal ADM with order swapped) on (52) is

$$\mathbf{y}_5^{k+1} = \arg\min_{\mathbf{y}} \|\mathbf{y}\|_{2,1} + (2\lambda)^{-1}\|\nabla\mathbf{x}_5^k - \mathbf{y} + \lambda\mathbf{z}_5^k\|_2^2, \tag{58a}$$

$$\mathbf{x}_5^{k+1} = \arg\min_{\mathbf{x}} \frac{\alpha}{2}\|\mathbf{x}-\mathbf{b}\|_2^2 + (2\lambda)^{-1}\|\nabla\mathbf{x} - \mathbf{y}_5^{k+1} + \lambda\mathbf{z}_5^k\|_2^2, \tag{58b}$$

$$\mathbf{z}_5^{k+1} = \mathbf{z}_5^k + \lambda^{-1}(\nabla\mathbf{x}_5^{k+1} - \mathbf{y}_5^{k+1}). \tag{58c}$$

Corollary 4. *Let* $\mathbf{x}_5^0 = \mathbf{b} + \alpha^{-1}\mathrm{div}\,\mathbf{z}_5^0$. *If the initialization for all algorithms* (55)-(58) *satisfies* $\mathbf{y}_1^0 = -\mathbf{z}_3^0 = \nabla\mathbf{x}_4^0 - \lambda(\mathbf{v}_4^0 - \mathbf{v}_4^{-1}) = \mathbf{y}_5^1$ *and* $\mathbf{z}_1^0 = \mathbf{v}_3^0 = \mathbf{v}_4^0 = \mathbf{z}_5^0 + \lambda^{-1}(\nabla\mathbf{x}_5^0 - \mathbf{y}_5^1)$. *Then for* $k \ge 1$, *we have the following equivalence results between the iterations of the four algorithms:*

$$\begin{array}{llll} \mathbf{y}_1^k & = -\mathbf{z}_3^k & = \nabla\mathbf{x}_4^k - \lambda(\mathbf{v}_4^k - \mathbf{v}_4^{k-1}) & = \mathbf{y}_5^{k+1}, \\ \mathbf{z}_1^k & = \mathbf{v}_3^k & = \mathbf{v}_4^k & = \mathbf{z}_5^k + \lambda^{-1}(\nabla\mathbf{x}_5^k - \mathbf{y}_5^{k+1}). \end{array}$$

Remark 9. *In any of the four algorithms, the* ∇ *or* div *operator is separated in a different subproblem from the term* $\|\cdot\|_{2,1}$ *or its dual norm* $\|\cdot\|_{2,\infty}$. *The* ∇ *or* div *operator is translation invariant so their subproblems can be solved by a diagonalization trick [22]. The subproblems involving the term* $\|\cdot\|_{2,1}$ *or the indicator function* $\iota_{\{\mathbf{v}:\|\mathbf{v}\|_{2,\infty}\le 1\}}$ *have closed-form solutions. Therefore, in addition to the equivalence results, all the four algorithms have essentially the same per-iteration costs.*

Acknowledgments

This work is supported by NSF Grants DMS-1349855 and DMS-1317602 and ARO MURI Grant W911NF-09-1-0383. We thank Jonathan Eckstein for bringing his early work [8, Chapter 3.5] and [9] to our attention.

References

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

[2] A. Chambolle, *An algorithm for total variation minimization and applications*, Journal of Mathematical Imaging and Vision, 20 (2004), pp. 89–97.

[3] 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 (2011), pp. 120–145.

[4] D. Davis and W. Yin, *Convergence rate analysis of several splitting schemes*, (2014).

[5] ——, *Convergence rates of relaxed Peaceman-Rachford and ADMM under regularity assumptions*, (2014).

[6] W. Deng and W. Yin, *On the global and linear convergence of the generalized alternating direction method of multipliers*, (2012).

[7] J. Douglas, Jim and J. Rachford, H. H., *On the numerical solution of heat conduction problems in two and three space variables*, Transactions of the American Mathematical Society, 82 (1956), pp. pp. 421–439.

[8] J. Eckstein, *Splitting Methods for Monotone Operators with Applications to Parallel Optimization*, PhD thesis, Massachusetts Institute of Technology, 1989.

[9] J. Eckstein and M. Fukushima, *Some reformulations and applications of the alternating direction method of multipliers*, in Large scale optimization, Springer US, 1994, pp. 115–134.

[10] E. Esser, X. Zhang, and T. Chan, *A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science*, SIAM Journal on Imaging Sciences, 3 (2010), pp. 1015–1046.

[11] J. Esser, *Primal Dual Algorithms for Convex Models and Applications to Image Restoration, Registration and Nonlocal Inpainting*, PhD thesis, University of California, Los Angeles, 2010.

[12] M. Fukushima, *The primal Douglas-Rachford splitting algorithm for a class of monotone mappings with application to the traffic equilibrium problem*, Mathematical Programming, 72 (1996), pp. 1–15.

[13] D. Gabay, *Applications of the method of multipliers to variational inequalities*, in Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems, M. Fortin and R. Glowinski, eds., North-Holland: Amsterdam, Amsterdam, 1983.

[14] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, Computers & Mathematics with Applications, 2 (1976), pp. 17–40.

[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*, Rev. Française d'Automat. Inf. Recherche Opérationelle, 9 (1975), pp. 41–76.

[16] T. Goldstein and S. Osher, *The split Bregman method for l1-regularized problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 323–343.

[17] M. Hestenes, *Multiplier and gradient methods*, Journal of Optimization Theory and Applications, 4 (1969), pp. 303–320.

[18] P. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 964–979.

[19] D. W. Peaceman and H. H. J. Rachford, *The numerical solution of parabolic and elliptic differential equations*, Journal of the Society for Industrial and Applied Mathematics, 3 (1955), pp. pp. 28–41.

[20] R. T. Rockafellar, *A dual approach to solving nonlinear programming problems by unconstrained optimization*, Mathematical Programming, 5 (1973), pp. 354–373.

[21] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D: Nonlinear Phenomena, 60 (1992), pp. 259 – 268.

[22] Y. Wang, J. Yang, W. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM Journal on Imaging Sciences, 1 (2008), pp. 248–272.

[23] Y. Xiao, H. Zhu, and S.-Y. Wu, *Primal and dual alternating direction algorithms for l1-l1-norm minimization problems in compressive sensing*, Computational Optimization and Applications, 54 (2013), pp. 441–459.

[24] J. Yang and Y. Zhang, *Alternating direction algorithms for ℓ_1-problems in compressive sensing*, SIAM Journal on Scientific Computing, 33 (2011), pp. 250–278.

[25] Y. Yang, M. Möller, and S. Osher, *A dual split Bregman method for fast ℓ^1 minimization*, Mathematics of Computation, 82 (2013), pp. 2061–2085.