Analyzing ADMM and it's Accelerated Variants

Raunak Shah {raunaks@iitk.ac.in}, Varun Goyal {gvarun@iitk.ac.in}, Ishanh Misra {imisra@iitk.ac.in}
Indian Institute of Technology Kanpur, India

PROJECT MODE : MIXED

I. Introduction

We analyze the alternating direction method of multipliers (ADMM) by focusing on the following problem:

$$\min H(u) + G(v) \text{ s.t. } Au + Bv \leq b$$

We start by analyzing convergence for normal ADMM and deriving a rate for when convergence occurs. We also analyze a restarted ADMM variant which works when one of $H(u)$ or $G(v)$ is weakly convex. Then we derive the basic ADMM updates for a general weakly convex problem $min\ F(u)$ s.t. $Au \leq b$. This is motivated by the simulations we present later in Section III. Then we take a slight detour where we present a novel strategy to improve FBS and FISTA from a "2-split" problem to "R-split". Finally in Section III we focus on a quadratic program with $F(u) = \frac{1}{2}u^TQu + q^Tu$ and the same constraint $Au \leq b$. We present simulation results for standard ADMM, it's faster restarted variant, and compare with three AMA (alternating minimization algorithm) variants that we can use since $F(u)$ is strongly convex. We present several proofs throughout and derive intricate updates and provide justifications that were not mentioned in the original paper. We further used these updates in our python code which was written from scratch and can be found here: [CODE].

II. Theoretical Analysis

A. Convergence of Normal ADMM

We shall analyze proof of convergence (normal ADMM) as given by authors of [1] assuming both H and G are strongly convex functions. But since this assumption is tight, and one of the objective functions might be often weakly convex, in Section II-B we will also analyze authors' proof for convergence (restart method for fast ADMM). This is relevant because the simulations we have done also use the restart method and show its convergence. Note that we shall state the key results here, and not simply restate the proofs here. Instead we shall try to critically analyze them. For the original proofs, refer to [1].

The convex optimization problem is:

$$\min H(u) + G(v) \quad \text{s.t.} \quad Au + Bv = b \tag{1}$$

Minimizing the lagrangian:

$$\min_{u,v} L(u, v, \lambda) = \min_{u,v} H(u) + G(v) + \lambda^T(b - Au - Bv)$$

$$= \min_u \left[H(u) - \lambda^T(Au)\right] + \min_v \left[G(v) - \lambda^T(Bv)\right] + \lambda^T b$$

Denoting H^* by the conjugate of H, we get

$$D(\lambda) = -H^*(A^T\lambda) - G^*(B^T\lambda) + \lambda^T b \tag{2}$$

This dual was stated directly in [1] without the above proof. Note that finding optimum of objectives often involves differentials of the form $\partial H^*(u)$ or $\nabla H^*(u)$. So, due to this repetitive nature, the authors have pre-defined functions:

$$\mathbf{\Psi}(\lambda) := A\nabla H^*(A^Tu) \quad \text{and} \quad \mathbf{\Phi}(\lambda) := B\nabla G^*(B^Tv) \tag{3}$$

Please note that we are using $\nabla H^*(.)$ instead of $\partial H^*(.)$ because H is strongly convex (for the purposes of this paper) and implies continuous differentiability of H^*. Before Lemma 1 in [1], the authors have directly stated (without any proof) Lipschitz constant $L(\mathbf{\Psi}) \leq \rho(A^TA)/\sigma_H$ (similarly for G), $\rho(.)$ being the matrix norm and σ_H being modulus of a strongly convex function H. A short proof is:

$$\begin{aligned}
&||\mathbf{\Psi}(x) - \mathbf{\Psi}(y)|| = ||A\nabla H^*(A^Tx) - A\nabla H^*(A^Ty)|| \\
&\leq \sqrt{\rho(A^TA)}\ ||\nabla H^*(A^Tx) - \nabla H^*(A^Ty)|| \\
&\leq \sqrt{\rho(A^TA)}\ L(\nabla H^*)\ ||A^Tx - A^Ty|| \\
&\leq \sqrt{\rho(A^TA)}\ \sigma_H^{-1}\ \sqrt{\rho(AA^T)}\ ||x - y|| \\
&\leq \rho(A^TA)\ \sigma_H^{-1}\ ||x - y||
\end{aligned}$$

The second line uses the known result $||Ax|| \leq \rho(A)\ ||x||$ that has also been covered in lectures; the third uses definition of Lipschitz constant; the fourth uses $L(H^*) = \sigma_H^{-1}$; the fifth uses $\rho(A^TA) = \rho(AA^T)$ (can prove trivially using Singular Vector Decomposition). One may refer section 1.2 of [1].

Lemma 1 of [1] states that:

$$Au^+ = \mathbf{\Psi}(\lambda^{1/2}) \quad \text{and} \quad Bv^+ = \mathbf{\Phi}(\lambda^+) \tag{4}$$

where $u^+, v^+, \lambda^+, \lambda^{1/2}$ etc. can be referred from section 3 of [1]. The proof for this involves simplifying the optimality conditions for u^+ and v^+. It also uses the identity:

$$x \in \partial F(y) \iff y \in \partial F^*(x) \tag{5}$$

for convex function F. The second part allows the authors to safely say that $Bv = \mathbf{\Phi}(\lambda)$ at the end of each iteration (and beginning of each iteration, second iteration onwards) - an assumption that is also used in Lemma 2.
In Lemma 2, the authors try to establish a bound on $D(\lambda^+) - D(\gamma)$, which represents the difference between the dual at the end of an iteration and the dual at any other value γ. They compute this difference for each term in the dual's expression (see equation (2)). That is, they separately analyze $H^*(A^T\gamma) - H^*(A^T\lambda^+)$ and $G^*(B^T\gamma) - G^*(B^T\lambda^+)$. We

elaborate on these calculations while providing justification for many steps which the original paper skips. For convex functions, note that $q(x) - q(y) \geq \langle x - y, \nabla q(y) \rangle$. Conjugates of a function are convex as covered in class, so G^* is convex. Thus using the above identity:

$$\begin{aligned} G^*(B^T\gamma) - G^*(B^T\lambda^+) &\geq \langle \gamma - \lambda^+, B\nabla G^*(B^T\lambda^+) \rangle \\ &\geq \langle \gamma - \lambda^+, \mathbf{\Phi}(\lambda^+) \rangle \end{aligned}$$

Similarly, $H^*(A^T\gamma) - H^*(A^T\lambda^+) = \Big[H^*(A^T\gamma)$
$-H^*(A^T\lambda^{1/2})\Big] - \Big[H^*(A^T\lambda^+) - H^*(A^T\lambda^{1/2})\Big]$
(now using convexity of H^* and definition of $\mathbf{\Psi}(.)$)
$\geq \langle \gamma - \lambda^{1/2}, \mathbf{\Psi}(\lambda^{1/2}) \rangle - \Big[H^*(A^T\lambda^+) - H^*(A^T\lambda^{1/2})\Big]$

Here, applying Taylor's approximation on the term in square brackets (upto the 2nd order):

$$\Big[H^*(A^T\lambda^+) - H^*(A^T\lambda^{1/2}))\Big] \leq \langle \lambda^+ - \lambda^{1/2}, A\nabla H^*(A^T\lambda^{1/2}) \rangle + \frac{A^2\partial\nabla H^*(A^T\lambda^{1/2})}{2}||\lambda^+ - \lambda^{1/2}||^2$$

Knowing that the Lipschitz constant $L(\nabla H^*) \leq \alpha := \frac{\rho(A^TA)}{\sigma_H}$, we get $\nabla H^*(x) - \nabla H^*(y)|| \leq \alpha||x - y||$. Thus $c \leq \alpha \; \forall \, c \in A^2\partial\nabla H^*(A^T\lambda^{1/2})$.

$$\Big[H^*(A^T\lambda^+) - H^*(A^T\lambda^{1/2})\Big] \leq \langle \lambda^+ - \lambda^{1/2},\; A\nabla H^*(A^T\lambda^{1/2}) \rangle + \frac{\alpha}{2}||\lambda^+ - \lambda^{1/2}||^2$$

Use this in the expression for $H^*(A^T\gamma) - H^*(A^T\lambda^+)$ while substituting $\mathbf{\Psi}(.)$:

$$\implies H^*(A^T\gamma) - H^*(A^T\lambda^+) \geq \langle \gamma - \lambda^{1/2}, \mathbf{\Psi}(\lambda^{1/2}) \rangle - \langle \lambda^+ - \lambda^{1/2}, A\nabla H^*(A^T\lambda^{1/2}) \rangle - \frac{\alpha}{2}||\lambda^+ - \lambda^{1/2}||^2$$

(again using definition of $\mathbf{\Psi}(.)$)

$$= \langle \gamma - \lambda^+, \mathbf{\Psi}(\lambda^{1/2}) \rangle - \frac{\alpha}{2}||\lambda^+ - \lambda^{1/2}||^2$$

A subtle point to note here is that the inequality corresponding to H^* has been derived in a little different manner than the derivation corresponding to G^*. Upon careful observation, we see that the difference for H^* is calculated by applying inequalities so that $\lambda^{1/2}$ is the base (like $f(x + \partial x) \approx f(x) + f'(x)\partial x$ has x as base). This helps introducing $\mathbf{\Psi}(\lambda^{1/2})$ into the inequality. On the other hand, for G^*, we just need the term $\mathbf{\Phi}(\lambda^+)$, so just applying convexity of G^* is enough, leading to a very short derivation. From here the proof continues on to compute $D(\lambda^+) - D(\gamma)$ by adding these expressions and substituting the values of $\mathbf{\Psi}(\lambda^{1/2})$ and $\mathbf{\Phi}(\lambda^+)$ (refer to [1] for the algebra). Finally:

$$2\tau[D(\lambda^+) - D(\gamma)] \geq 2\langle \gamma - \lambda, \lambda - \lambda^+ \rangle + ||\lambda^+ - \lambda||^2 \quad (6)$$

Note that in kth iteration $\lambda = \lambda_k$ and $\lambda^+ = \lambda_{k+1}$. One may be tempted to plug in $\gamma = \lambda^*$ (optimal dual variable):

$$2\tau[D(\lambda_{k+1}) - D(\lambda^*)] \geq 2\langle \lambda^* - \lambda_k, \lambda_k - \lambda_{k+1} \rangle + ||\lambda_{k+1} - \lambda_k||^2 \quad (7)$$

but note that the RHS then becomes dependent on λ_k and λ_{k+1} itself and does not provide useful insight on how to prove $O(1/k)$ limit. Therefore, the authors also find the successive difference $D(\lambda_{k+1}) - D(\lambda_k)$ by taking $\gamma = \lambda_k$:

$$2\tau[D(\lambda_{k+1}) - D(\lambda_k)] \geq ||\lambda_{k+1} - \lambda_k||^2 \quad (8)$$

For brevity, I shall denote $D(\lambda_i) = D_i$ and $D(\lambda^*) = D^*$. To find the general difference $D^* - D_n$, equations (7) and (8) are summed from $k = 0$ to $(n-1)$ and added together to produce:

$$D(\lambda^*) - D(\lambda_n) \leq \frac{||\lambda^* - \lambda_1||^2}{2\tau(n-1)} \quad (9)$$

The subtle point here to note is than we now have an RHS that does not depend on λ_k or λ_{k+1} at all so the dual converges as $O(1/n)$. This essentially proves Theorem 1 in [1]. Note that the authors of [1] have incorrectly used $\sum_{k=1}^{n-1} k||\lambda_k - \lambda_{k+1}||^2$ instead of $\sum_{k=1}^{n-1}(k-1)||\lambda_k - \lambda_{k+1}||^2$ in their derivation. The proof for convergence of normal ADMM also goes on to show that the residuals:

$$r_k := b - Au_k - Bv_k \quad \text{and} \quad d_k := \tau A^T B(v_k - v_{k+1}) \quad (10)$$

also decay as $O(1/k)$. This proof (lemma 3) utilizes equations (8) and (9) and the verified hypothesis $Bv_k = \mathbf{\Phi}(\lambda_k)$. Again, in this proof, in the first line, instead of $(1/2\tau)||\lambda_k - \lambda_{k+1}||^2$, the authors have written $(\tau/2)||\lambda_k - \lambda_{k+1}||^2$. But this is also fine since we are focusing on proving the $O(1/k)$ limit (and not the constants associated with this limit).

B. Accelerated ADMM (without restart method)

The famous algorithms FBS, Nesterov's method and FISTA (algorithms 4, 5, and 6 in [1]) make us observe that FISTA uses FBS with additional updates similar to Nesterov's method. This "Nesterov-like updates" are used by the authors to construct the fast ADMM algorithm. In this case, α_{k+1} is computed iteratively as:

$$\alpha_{k+1} := \frac{1 + \sqrt{1 + 4\alpha_k^2}}{2} \quad (11)$$

Then, new updates for $\hat{v}$ and $\hat{\lambda}$ are computed iteratively:

$$\begin{aligned} \hat{v}_{k+1} &:= v_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(v_k - v_{k-1}) \\ \hat{\lambda}_{k+1} &:= \lambda_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(\lambda_k - \lambda_{k-1}) \end{aligned}$$

Note that values from the previous TWO iterations are need (v_k, v_{k-1} and λ_k, λ_{k-1}) in current iteration as opposed to ONE in normal ADMM (v_k and λ_k). The authors of [1] have given enough proofs showing that rate of convergence is $O(1/k^2)$ if both $H(u)$ and $G(v)$ are convex.

C. Accelerated ADMM with the Restart Method

For the restart method we can prove that the algorithm will converge, but finding an explicit convergence rate is not possible. However experimentally the restart method performs much better than the other variants. Possible paths for the kth iteration: (1) If $\frac{c_k}{c_{k-1}} < \eta$; Updates happen like accelerated

ADMM (2) Else Restart and then unaccelerated updates; we set $\alpha_{k+1} = \alpha_1 = 1, \hat{v}_{k+1} = v_{k-1}, \hat{\lambda}_{k+1} = \lambda_{k-1}$. We will use the following result for unaccelerated ADMM [2],

$$\frac{1}{\tau}||\lambda_{k+1} - \lambda_k||^2 + \tau||B(v_{k+1} - (v_k)||^2 \leq \frac{1}{\tau}||\lambda_k - \lambda_{k-1}||^2 + \tau||B(v_{k+1} - (v_k)||^2 \quad (12)$$

The terms on LHS and RHS denote the combined residuals. We can see that in unaccelerated ADMM the combined residual is non-increasing. Let us assume there have been n-iterations of the restart-method algorithm, of which n_1 were accelerated and n_2 resulted in restarts (hence n_2 were unaccelerated). Hence, $n = n_1 + n_2$. By the inequalities on combined residual explained above, we can say, $c_n \leq \eta^{-n_1} c_0$. Now, if we assume there does not happen any accelerated iteration after a finite i iterations, then in all the upcoming iterations there won't be any accelerated updates, hence it is equivalently running unaccelerated ADMM from then onwards, which we have proven above to converge. If the iterations taking accelerated path are infinite then we can inherently say $c_k \to 0$ as $c_n \leq \eta^{-n_1} c_0$ and $n_1 \to \inf$. Hence we have shown convergence of restart method by converging combined residuals.

D. *Updates for General Convex Problem (Linear Constraints)*

Taking inspiration from the quadratic program discussed later in equation (34), we solve the general convex optimization:

$$\min_u F(u) \qquad \text{s.t.} \quad Au \leq b \qquad (13)$$

Introducing the infinite step function:

$$I_{v \leq b}(v) = \begin{cases} 0 & v \leq b \\ \infty & \text{in other cases} \end{cases} \qquad (14)$$

the problem may be rewritten as:

$$\min_u F(u) + I_{v \leq b}(v) \qquad \text{s.t.} \quad Au - v = 0 \qquad (15)$$

which is possible only when $\sigma_{I_{v \leq b}} = 0$ while we require a strictly positive modulus for convexity. As will be shown in equation (36), $I_{v \leq b}$ is not strongly convex. Still being weakly convex (think of $I_{v \leq b}(v)$ being the extended definition of the zero function over the domain $v \leq b$), we can use either normal ADMM or fast ADMM with restart conditions. Updating $u:$

$$\arg\min_u F(u) - \lambda^T(Au) + \frac{\tau}{2}||0 - Au + v||^2 \qquad (16)$$

Setting gradient to zero:

$$0 \in \partial F(u^*) - A^T\lambda + \tau A^T(Au^* - v) \qquad (17)$$

$$A^T(\lambda + \tau v) \in \tau A^T A \left(I + (\tau A^T A)^{-1}\partial F\right) u^* \qquad (18)$$

$$(A^T A)^{-1} A \left(\frac{\lambda}{\tau} + v\right) \in \left(I + (\tau A^T A)^{-1}\partial F\right) u^* \qquad (19)$$

So,the update equation must be:

$$u^* = J_{(\tau A^T A)^{-1}\partial F}(A^T A)^{-1} A \left(\frac{\lambda}{\tau} + v\right) \qquad (20)$$

where $J_{\partial F} := (I + \partial F)^{-1}$ is defined as the proximal (or resolvent) operator (see section 1.2 of [1]). This gives the advantage that this operator and the matrix $(A^T A)^{-1}A$ can be computed only once in the algorithm's beginning. For $v - update:$

$$\begin{aligned}
&\arg\min_v I_{v \leq b}(v) + \lambda^T v + \frac{\tau}{2}\|0 - Au + v\|_2^2 \\
&\quad \text{(Impose the constraint } v \leq b) \\
&= \arg\min_v 0 + \lambda^T v + \frac{\tau}{2}\left(v^T v - 2u^T A^T v + u^T A^T A u\right) \\
&= \arg\min_v (\lambda - \tau A u)^T v + \frac{\tau v^T v}{2} \\
&= \sum_i \arg\min_{v_i} (\lambda - \tau A u)_i v_i + \frac{\tau}{2} v_i^2
\end{aligned}$$

Inner expressions are convex quadratics and minimized when:

$$\tau v_i^* - \tau(Au)_i + \lambda_i = 0 \implies v_i^* = (Au)_i - \frac{\lambda_i}{\tau} \qquad (21)$$

However this assumes $(Au)_i - \frac{\lambda_i}{\tau} \leq b_i$ as we assumed $v \leq b$ earlier. If not, then we must select a different $v^* \leq b$. Since the objective expression is a quadratic in v, it will be monotonically decreasing for values of v_i that lie to the left of the minima. Note that here b_i is one of these values that is lesser than the minima. Thus we need to choose the maximum possible v_i that satisfies the constraint $v_i \leq b_i$. This implies $v_i = b_i$. We can compactly write this as $v_i = \min\left[(Au)_i - \frac{\lambda_i}{\tau}, b_i\right]$. Or:

$$v^* = \min\left(Au - \frac{\lambda}{\tau},\ b\right) \qquad (22)$$

For λ-update:

$$\lambda^* := \lambda + \tau(0 - Au + v) = \lambda + \tau(v - Au) \qquad (23)$$

The general algorithm for normal ADMM then becomes:

```
for k = 1, 2, 3, ... do
    $u_{k+1} := J_{(\tau A^T A)^{-1}\partial F}\left((A^T A)^{-1} A \left(\frac{\lambda_k}{\tau} + v_k\right)\right)$
    $v_{k+1} := \min\left(Au_{k+1} - \frac{\lambda_k}{\tau}, b\right)$
    $\lambda_{k+1} := \lambda_k + \tau(v_{k+1} - Au_{k+1})$
```

For fast ADMM, only addition lines have to be added, same as lines 5-7 of algorithm 7 (fast ADMM) of [1]. Note that we propose this algorithm for the general convex problem (13) since resolvent operator can be predefined and also since the update steps for λ and v are pretty straight forward. Such update steps can also be introduced for the faster variants of ADMM and AMA using Nesterov's method (see [3]).

E. *Extending FBS and ADMM for R-splits*

[1] uses Nesterov-like updates to accelerate ADMM. We had to briefly comprehend Nesterov's method ([3]), and along with that FISTA and FBS ([4]) too. So, we struck upon a way to improve FBS for R-splits. Objective of FBS splits across two convex functions $H(x) + G(x)$ and does:

$$x_{k+1} = J_{\tau\partial G}(x_k - \tau\partial H(x_k)) \qquad (24)$$

Using the definition of resolvent operator $J_{\tau\partial G} := (I + \tau\partial G)^{-1}$, the above can be written equivalently as:

$$(I + \tau\partial G)x_{k+1} = x_k - \tau\partial H(x_k) \quad (25)$$

$$x_{k+1} + \tau\partial G(x_{k+1}) = x_k - \tau\partial H(x_k) \quad (26)$$

This suggests that if we go backward from the destination x_{k+1} guided by ∂G (LHS), then we are at the same point as if we go forward from the starting point x_k guided by ∂H (RHS). This is essentially the "backward" and "forward" part of FBS. We propose to generalize this split for R splits. The objective may be $F_1(x) + F_2(x) + \cdots + F_R(x)$. Let the start point be x and destination be x^*. Also let the $(R-1)$ split points be $x_1, x_2, \ldots, x_{R-1}$. We start from x to arrive at x_1 :

$$x_1 = x - \tau\partial F_1(x) \quad (27)$$

Now start from last point x^* and propagate back to reach x_1 :

$$x_{R-1} = x^* + \tau\partial F_R(x^*) = (I + \tau\partial F_R)x^* \quad (28)$$

$$x_{R-2} = x_{R-1} + \tau\partial F_{R-1}(x_{R-1}) = (I + \tau\partial F_{R-1})x_{R-1} \quad (29)$$

and so on till: $x_1 = x_2 + \tau\partial F_2(x_2) = (I + \tau\partial F_2)x_2$ (30)

Then combining equations (27), (28), (29) and (30), we get:

$$(I + \tau\partial F_2)\ldots(I + \tau\partial F_R)x^* = x - \tau\partial F_1(x) \quad (31)$$

Again, resolvent operator helps rewriting the above as:

$$x^* = J_{\tau\partial F_R} J_{\tau\partial F_{R-1}} \ldots J_{\tau\partial F_2}(x - \tau\partial F_1(x)) \quad (32)$$

suggesting that extended FBS algorithm may look like:

```
**for** $k = 0, 1, 2, \ldots,$ **do**
    $x_{k+1} = J_{\tau\partial F_R} J_{\tau\partial F_{R-1}} \ldots J_{\tau\partial F_2}(x_k - \tau\partial F_1(x_k))$
```

We justify the above approach by observing equation (31) under the assumptions : (a) $F_1, F_2, \ldots, F_R$ are differentiable upto second order. So $\partial F(x) = \{\nabla F(x)\}$; (b) each update step does not introduce huge changes. Then for any two functions F_i and F_j:

$$\begin{aligned}
&(I + \tau\partial F_i)(I + \tau\partial F_j)x = (I + \tau\nabla F_i)(I + \tau\nabla F_j)x \\
&= (I + \tau\nabla F_i)(x + \tau\nabla F_j(x)) \\
&= x + \tau\nabla F_j(x) + \tau\nabla F_i(x + \tau\nabla F_j(x)) \\
&\quad \text{(assuming updates are not huge, using Taylor's expansion)} \\
&\approx x + \tau\nabla F_j(x) + \tau\nabla F_i(x) + \tau\nabla^2 F_i(\tau\nabla F_j(x)) \\
&\quad \text{(assuming updates are not huge)} \\
&\approx x + \tau\nabla F_j(x) + \tau\nabla F_i(x)
\end{aligned}$$

The above form is also symmetric in F_i and F_j, so the resolvent operators $J_{\tau\partial F_2}, J_{\tau\partial F_3}, \ldots, J_{\tau\partial F_R}$ in the extended FBS algorithm may be used in any order. Since FISTA is an extension of FBS using Nesterov's method, extended FISTA is suggested (see original FISTA in [1]):

```
**for** k = 1, 2, 3, ... **do**
    $x_k = J_{\tau\partial F_R} J_{\tau\partial F_{R-1}} \ldots J_{\tau\partial F_2}(y_k - \tau\partial F_1(y_k))$
    $\alpha_{k+1} = (1 + \sqrt{1 + 4\alpha_k^2})/2$
    $y_{k+1} = x_k + \frac{\alpha_k - 1}{\alpha k+1}(x_k - x_{k-1})$
```

Seeing the form of ADMM in equation (1) in [1], one is naturally tempted to extend ADMM (and also its accelerated methods) to R-splits. Upon reading the comparison of fast ADMM with other methods, we decide to only suggest what may be called R-split ADMM variants (refer to [1] for more details). The following algorithms assume the convex optimization problem of the form (here $u^1, u^2, \ldots, u^R$ are distinct variables, not powers):

$$\min_{u^1, u^2, \ldots, u^R} \sum_{i=1}^{R} F_i(u^i) \quad \text{and} \quad \sum_{i=1}^{R} A_i u^i = b \quad (33)$$

R-split fast ADMM:

```
**for** k = 0, 1, 2, ... **do**
    $u^1_{k+1} = \arg\min_{u^1} F_1(u^1) - \langle\hat{\lambda}_k, A_1 u^1\rangle$
            $+ \frac{\tau}{2}||b - A_1 u^1 - A_2 \hat{u}^2_k - \cdots - A_R \hat{u}^R_k||^2$
    $\vdots$
    $u^R_{k+1} = \arg\min_{u^R} F_R(u^R) - \langle\hat{\lambda}_k, A_R u^R\rangle$
            $+ \frac{\tau}{2}||b - A_1 u^1_{k+1} - A_2 u^2_{k+1} - \cdots - A_R u^R||^2$
    $\lambda_k = \hat{\lambda}_k + \tau(b - A_1 u^1_{k+1} - \cdots - A_R u^R_{k+1})$
    $\alpha_{k+1} = \frac{1 + \sqrt{1 + \alpha_k^2}}{2}$
    $\hat{u}^2_{k+1} = u^2_{k+1} + ((\alpha_k - 1)/\alpha_{k+1}) \cdot (u^2_k - u^2_{k-1})$
    $\vdots$
    $\hat{u}^R_{k+1} = u^R_{k+1} + ((\alpha_k - 1)/\alpha_{k+1}) \cdot (u^R_k - u^R_{k-1})$
    $\hat{\lambda}_{k+1} = \lambda_k + ((\alpha_k - 1)/\alpha_{k+1}) \cdot (\lambda_k - \lambda_{k-1})$
```

Note that the algorithm for R-split normal ADMM can be similarly stated down by removing Nesterov-like updates from the above algorithm. Given: (a) the nature of resolvent operators to get concatenated as seen in (32), (b) direct use of resolvent operator to solve problems like (13) as seen in the algorithm after (36), we also suggest using resolvent operators for R-split ADMM:

```
**for** k = 0, 1, 2, ... **do**
    **for** r = 1, 2, ..., R **do**
        $u^r_{k+1} = J_{(\tau A_r^T A_r)^{-1}\partial F_r}\left((A_r^T A_r)^{-1} A_r(v_k + \lambda_k/\tau)\right)$
    $\lambda_{k+1} = \lambda_k + \tau(v_{k+1} - A_1 u^1_{k+1} - \cdots - A_R u^R_{k+1})$
```

where v is a variable introduced to cover for inequality similar to that in (13).

III. Simulations & Numerical Results

We simulated the standard ADMM and it's faster variants on a quadratic programming problem. We also compared the convergence results with AMA, another alternating optimization method. This comparison is addressed in subsection III-E.

$$\min \frac{1}{2} u^T Q u + q^T u \text{ s.t. } Au \leq b \quad (34)$$

The ADMM template as described in the paper [1] is:

$$\min H(u) + G(v) \text{ s.t. } Au + Bv \leq b$$

We expressed the quadratic program in this form:

$$\min \frac{1}{2} u^T Q u + q^T u + I_{v \leq b}(v) \text{ s.t. } Au - v = 0 \quad (35)$$

where $I_{v\le b}(v)$ has same definition as in (14). Here $H(u) = \frac{1}{2}u^TQu + q^Tu, G(v) = I_{v\le b}(v)$. Since strong convexity for a function P means there exists largest $\sigma_P > 0$ s.t.:

$$tP(x) + (1-t)P(y) - P(tx + (1-t)y) \geq \sigma_P\, t(1-t)||x-y||^2 \quad (36)$$

for all $x, y \in \text{dom}P$ and all $0 \le t \le 1$. Note that $I_{v\le b}(v)$ isn't strongly convex since LHS always evaluates to 0, only if $\sigma_{I_{v\le b}} = 0$. We require $\sigma_{I_{v\le b}} > 0$ for strong convexity, so $I_{v\le b}(v)$ is not strongly convex. Also on substituting $H(u)$ into (36), we get the condition

$$\frac{\lambda(1-\lambda)}{2}(x-y)^TQ(x-y) \geq \sigma_H\lambda(1-\lambda)||x-y||_2^2$$

Use the symmetric eigenvalue decomposition ($Q = E\Lambda E^T$):

$$(x-y)^TE(\Lambda - 2\sigma_H I)E^T(x-y) \geq 0 \quad (37)$$

Since Q is positive definite, it's eigenvalues are positive so we can choose $\sigma_H = \frac{\lambda_{min}(Q)}{2} > 0$ and the above equation will be satisfied. Hence $H(u)$ is strongly convex and $G(v)$ is not, the ADMM acceleration method for strongly convex functions will not work here. Thus we present simulation results for normal ADMM and the accelerated-restart method that applies to weakly convex functions which conforms with the paper. The authors have also omitted using accelerated ADMM for QP, and proceeded with restart method.

Now we derive the updates for ADMM. All the updates follow from the augmented Lagrangian. Note that the form of this QP problem is same as equation (13), so we know automatically that the update steps for v and λ are same as given by equations (22) and (23) respectively. For updating u, we just need to find the resolvent operator in equation (20). For QP, resolvent operator is:

$$\partial H(u) = \frac{\partial}{\partial u}\left(\frac{u^TQu}{2} + q^Tu\right) = Qu + q$$

Writing $J_{(\tau A^TA)^{-1}\partial H} = J$ in short:

$$\begin{aligned}
&J^{-1}(u) = \left(I + (\tau A^TA)^{-1}\partial H\right)(u)\\
&= u + (\tau A^TA)^{-1}Qu + (\tau A^TA)^{-1}q\\
&= \left(I + (\tau A^TA)^{-1}Q\right)u + (\tau A^TA)^{-1}q\\
&\implies J\left((A^TA)^{-1}A(v + \lambda/\tau)\right)\\
&= \left(I + (\tau A^TA)^{-1}Q\right)^{-1}[(\tau A^TA)^{-1}A(\lambda + \tau v)\\
&- (\tau A^TA)^{-1}q]
\end{aligned}$$

$$J_{(\tau A^TA)^{-1}\partial H}(u) = \left(\tau A^TA + Q\right)^{-1}[A(\lambda + \tau v) - q] \quad (38)$$

which is the same as the update step in algorithm 10 of [1]. This confirms that pre-determining the resolvent operator will be much more helpful (as already explained at the end of section II.D of this paper). Note that $(Q + \tau A^TA)$ is invertible as Q is taken as positive definite and A^TA is positive semidefinite, which makes the net sum positive definite.

A. ADMM - Simulations and Analysis

We obtained solutions for the QP by simulating normal ADMM and the accelerated-restart variant, and have plotted results in fig 1 and 2, reproducing fig 9 of the original paper[1]. All our code was written in Python from scratch, using vector sizes, initializations and hyperparameters as specified in [1].

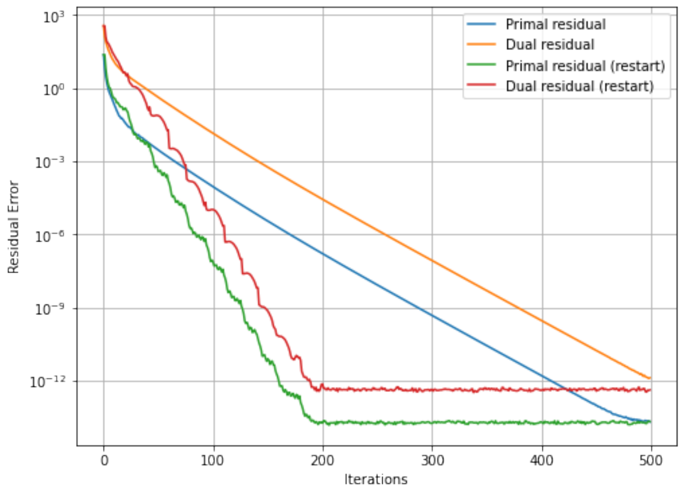

Fig. 1. Plot of primal and dual residuals for normal ADMM and accelerated ADMM with restart method

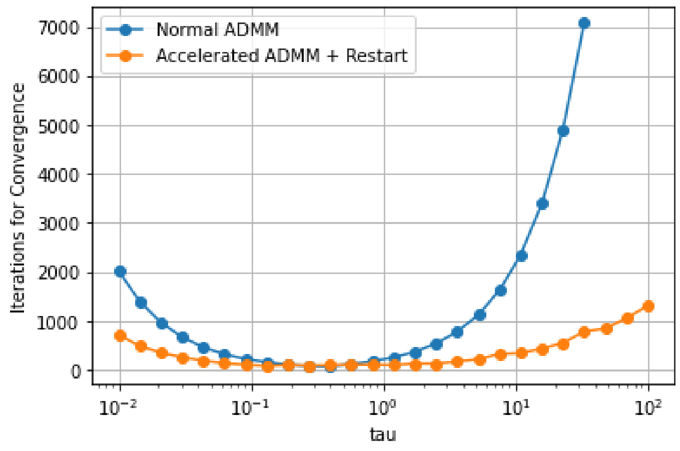

Fig. 2. Plot of τ vs number of iterations required for convergence of ADMM (i.e. when residual hits 10^{-5}). We used $\tau = 1$.

B. Comparisons with AMA

Finally we also compare the ADMM approaches with variants of the AMA algorithms for further analysis. The algorithm for AMA is almost the same as ADMM, but it requires one of the objective functions to be strongly convex. We already proved that $H(u)$ is strongly convex and $G(v)$ isn't in the beginning of section III, so it is valid to apply AMA here. The algorithm for AMA is almost the same as ADMM except we do not include the augmented lagrangian term in the u-update. Thus the updates for u become (set $\tau = 0$ in (38)):

$$u^* = Q^{-1}\left(A^T\lambda - q\right)$$

The v and λ updates remain the same. The fast variant of AMA uses FISTA-based acceleration of the lambda update, and the restart variant is similar to ADMM except we only restart when the dual objective increases in an iteration. In order to measure progress, we track the primal residuals and the difference between the current dual objective and the optimal dual objective $D(\lambda) - D(\lambda^*)$ in each iteration. We derived the dual in (2). For our QP problem this reduces to:

$$D(\lambda) = \min_u \left[\frac{u^T Q u}{2} + q^T u - \lambda^T(Au)\right] + \min_v \left[I_{v \le b}(v) + \lambda^T v\right] + \lambda^T b$$

Note that the first term corresponds to $-H^*(A^T\lambda)$ and second term is $-G^*(B^T\lambda) = -G^*(-\lambda)$ because of definition of conjugate function. Substituting the optima to the individual optimization problems $u^* = Q^{-1}\left(A^T\lambda - q\right)$ and $v^* = 0$:

$$H^*(A^T\lambda) = (A^T\lambda)^T Q^{-1}(A^T\lambda - q)$$
$$-\frac{1}{2}\left(A^T\lambda - q\right)^T Q^{-1}\left(A^T\lambda - q\right) - q^T Q^{-1}(A^T\lambda - q)$$

$$= \frac{1}{2}(A^T\lambda)^T Q^{-1}(A^T\lambda - q) - \frac{1}{2} q^T Q^{-1}(A^T\lambda - q)$$

Also $G^*(B^T\lambda) = G^*(-\lambda) = 0$. So we get:

$$D(\lambda) = -\frac{1}{2}\left(A^T\lambda - q\right)^T Q^{-1}\left(A^T\lambda - q\right) + \lambda^T b$$

For $D(\lambda^*)$ we used the λ^* that we got from converged ADMM. Again all our code was written from scratch. We observed that AMA converges much more slowly than ADMM. We set $N_u = 50$ and $N_b = 25$ as specified in [1]. We used $\tau = \frac{\lambda_{min}(Q)}{\rho(A^T A)}$. The reason for this is described in the analysis section below. The plot of dual objective error successfully reproduces Fig. 10 of [1]. We also provide the primal residual plot for additional reference.

1) ADMM vs AMA - Analysis: Fig. 1 is a plot for $N_u = 500$ and $N_b = 250$, where $A \in R^{N_u \times N_b}$. Our AMA plots correspond to $N_u = 50$ and $N_b = 25$. We observed that even normal ADMM converges much faster (around 500 iterations) even when the optimization problem is 10x larger. The fastest variant of AMA, on the other hand takes about 2000+ iterations to converge properly. We suspect that this is largely due to restrictions imposed by the value of τ.

Based on the convergence analysis for AMA in [1], $\tau < \frac{2\sigma_H}{\rho(A^T A)}$ is required for AMA to converge. We already computed the limiting $\sigma_H = \frac{\lambda_{min}(Q)}{2}$ when we were proving strong convexity of $H(u)$ in (37). Substituting this into the convergence criterion, we get $\tau \le \frac{\lambda_{min}(Q)}{\rho(A^T A)}$, that is of the order 10^{-5}. This is the value of τ that we used for AMA.

τ acts like a 'learning rate' parameter for the augmented lagrangian term in the u, v, and λ updates in ADMM and AMA. The τ-sensitivity plot Fig. 2 shows that the optimal τ for ADMM for fastest convergence is between 10^{-1} and 1. Even when $N_u = 50$ and $N_b = 25$ we observed a similar ADMM sensitivity plot. However, when we try to increase τ for AMA beyond 10^{-5}, the primal residual and dual objective diverge as per the theoretical condition above. So the AMA algorithm is forced to be slower by a smaller value of τ since the step sizes are smaller for the updates. The ADMM tau-sensitivity plot clearly shows that if we use $\tau = 10^{-5}$ for ADMM we will get slower convergence there as well.

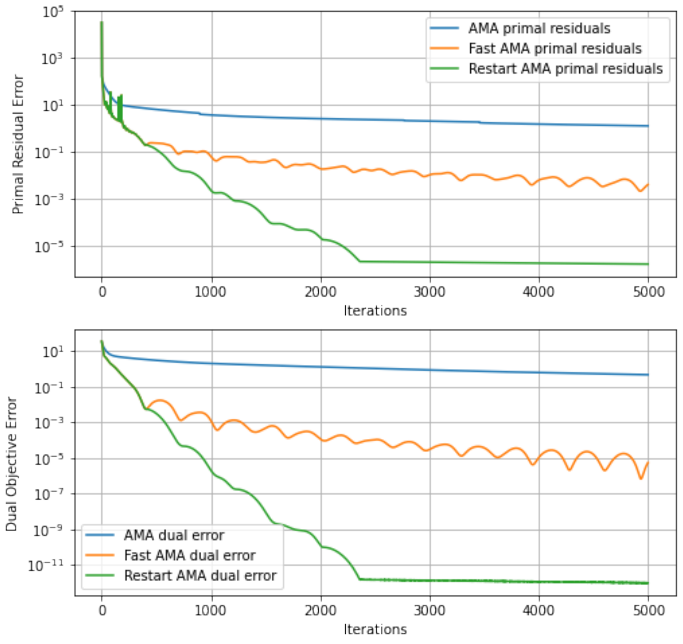

Fig. 3. Plot of primal residuals and dual objective error for normal AMA, fast AMA, and restarted AMA

IV. Conclusion

This concludes our analysis of the ADMM algorithm and it's variants. We analyzed convergence and derived updates for the general problem. Our QP based simulation ultimately compared ADMM to AMA and their accelerated variants. We also proposed an extended FBS-splitting algorithm to work for an objective that can be split into"R-splits". Further work along this direction could be to simulate ADMM/AMA for a problem with both strongly convex objectives (like elastic net regularization) or add equality constraints to the problem. We can also try to simulate the R-split problem to see how well our proposed scheme works.

References

[1] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM Journal on Imaging Sciences*, vol. 7, no. 3, pp. 1588–1623, 2014.

[2] Y. X. He, Bingsheng, "On non-ergodic convergence rate of douglas–rachford alternating direction method of multipliers," *Optimization Online*, 2012.

[3] Y. E. NESTEROV, "A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$," *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1983.

[4] R. E. Bruck, "On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in hilbert space," *Journal of Mathematical Analysis and Applications*, vol. 61, no. 1, pp. 159–164, 1977.