Line Search for Averaged Operator Iteration

Pontus Giselsson, Mattias Fält, and Stephen Boyd

Abstract

Many popular first order algorithms for convex optimization, such as forward-backward splitting, Douglas-Rachford splitting, and the alternating direction method of multipliers (ADMM), can be formulated as averaged iteration of a nonexpansive mapping. In this paper we propose a line search for averaged iteration that preserves the theoretical convergence guarantee, while often accelerating practical convergence. We discuss several general cases in which the additional computational cost of the line search is modest compared to the savings obtained.

1 Introduction

First-order algorithms such as forward-backward splitting, Douglas-Rachford splitting, and the alternating direction methods of multipliers (ADMM) are often used for large-scale convex optimization. While the theory tells us that these methods converge, practical convergence can be very slow for some problem instances. One effective method to reduce the number of iterations is to precondition the problem data. This approach has been extensively studied in the literature and has proven very successful in practice; see, e.g., [4, 7, 22, 16, 18, 19] for a limited selection of such approaches.

Another general approach to improving practical efficiency is to carry out a line search, i.e., to first compute a tentative next iterate and then to select the next iterate on the ray from the current iterate passing through the tentative iterate. Typical line searches are based on some readily computed quantity such as the function value or norm of the gradient or residual. A well designed line search preserves the theoretical convergence of the base method, while accelerating the practical convergence. Line search is widely used in gradient descent or Newton methods; see [6, 24]. These line search methods cannot be applied to all first-order methods mentioned above, however, since in general there is no readily computed quantity that is decreasing. (The convergence proofs for these methods typically rely on quantities related to the distance to an optimal point, which cannot be evaluated while the algorithm is running.) In this paper we propose a general line search scheme that is applicable to most first-order convex optimization methods, including those mentioned above whose convergence proofs are not based on the decrease of an observable quantity.

We exploit the fact that many first-order optimization algorithms can be viewed as averaged iterations of some nonexpansive operator, i.e., they can be written in the form

$$x^{k+1} = (1 - \bar{\alpha})x^k + \bar{\alpha}Sx^k = x^k + \bar{\alpha}(Sx^k - x^k), \qquad (1)$$

where $\bar{\alpha} \in (0, 1)$ and $S : \mathbb{R}^n \to \mathbb{R}^n$ is nonexpansive, i.e., it satisfies $\|Su - Sv\|_2 \leq \|u - v\|_2$ for all u, v. The superscript k denotes iteration number. The middle expression shows that the next point is a weighted average of the current point x^k and Sx^k. The expression on the righthand side of (1) shows that the iteration can be interpreted as a taking a step of length $\bar{\alpha}$ in the direction of the fixed-point residual $r^k = Sx^k - x^k$. Assuming a fixed-point exists, the iteration (1) converges to the set of fixed-points.

In this paper we will show how steps sometimes much larger than $\bar{\alpha}$ can be taken, which typically accelerates practical convergence. This iteration has the form

$$x^{k+1} = x^k + \alpha_k(Sx^k - x^k), \qquad (2)$$

where $\alpha_k > 0$ is chosen according to line search rules described below. We refer to α_k as the *step length* in the kth iteration, and $\bar{\alpha}$ as the *nominal step length*. The choice $\alpha_k = \bar{\alpha}$ recovers the basic averaged iteration (1). We refer to the selection of α_k as a line search, since we are selecting the next iterate as a point on the line or ray passing through x^k in the direction of the residual.

The merit function used to accept a step length α_k in the line search is the norm of the fixed-point residual $\|r\|_2 = \|Sx - x\|_2$. To evaluate this merit function for a candidate point, we must compute Sx, which corresponds to the dominant cost of taking a full iteration of the nominal algorithm. In the general case, then, the line search is computationally expensive, and there is a trade-off between the cost of the line search (which depends on the number of candidate points examined), and the savings in iterations due to the line search. But we have identified many common and interesting problem and algorithm combinations for which the fixed-point residual can be computed at low additional cost along the candidate ray. In these situations, performing one iteration with line search is roughly as expensive as performing one standard iteration of the nominal algorithm, so the additional cost of the line search is minimal. This happens when the nonexpansive operator S can be written as $S = S_2 S_1$ where $S_1 : \mathbb{R}^n \to \mathbb{R}^n$ is affine and $S_2 : \mathbb{R}^n \to \mathbb{R}^n$ is relatively cheap to evaluate.

The paper is organized as follows. In Section 2, we state the line search method and prove its convergence. In Section 3, we show that the line search can be carried out efficiently when $S = S_2 S_1$ and S_2 is cheap to evaluate and S_1 is affine. In Section 4, we show how to implement the line search for some popular algorithms. Finally, in Section 6 we provide numerical examples that show the efficiency of the proposed line search.

2 The line search method

2.1 Line search test

The line search method first computes the nominal next iterate $\bar{x}^k$ according to the basic averaged iteration (1), and then (possibly) selects a different value of α_k. The algorithm has the following form.

$$r^k := Sx^k - x^k \tag{3}$$

$$\bar{x}^k := x^k + \bar{\alpha}r^k \tag{4}$$

$$\bar{r}^k := S\bar{x}^k - \bar{x}^k \tag{5}$$

$$x^{k+1} := x^k + \alpha_k r^k \tag{6}$$

In the first step we compute the current residual, in the second step we compute the nominal next iterate, and in the third step we compute the nominal next residual. In the last step, we form the actual next iterate.

In (6) the step length α_k must satisfy the following. Either $\alpha_k = \bar{\alpha}$, i.e., we take the nominal step, or $\alpha_k \in (\bar{\alpha}, \alpha^{\max}]$ is such that

$$\|r^{k+1}\|_2 = \|Sx^{k+1} - x^{k+1}\|_2 \le (1 - \epsilon)\|\bar{r}^k\|_2, \tag{7}$$

where $\epsilon \in (0,1)$ and $\alpha^{\max} \ge \bar{\alpha}$ are fixed algorithm parameters. Thus we either take the nominal step, or one that reduces the norm of the fixed point residual compared to the nominal step.

We will discuss the details of the computation and give some specific methods to choose α_k later; but for now we observe that to verify the line search test (7), we must evaluate r^{k+1}, which is the first step (3) of the next iteration. In a similar way, if we take the nominal step, i.e., choose $\alpha_k = \bar{\alpha}$, then step (5) is the first step of the next iteration. In either case, there is no additional computational cost.

2.2 Convergence analysis

We analyze the proposed line search method and provide residual and iterate convergence results. All results are proven in Appendix A.

Theorem 1 *Suppose that $S : \mathbb{R}^n \to \mathbb{R}^n$ is nonexpansive and let $\bar{\alpha} \in (0,1)$. Then the iteration (3)-(6) satisfies $\|r^k\|_2 \to c$ as $k \to \infty$.*

So, the norm of the residual converges. Next, we show that the residual converges to zero if a fixed-point to S exists, i.e., if $\mathrm{fix}S = \{x \in \mathbb{R}^n \mid x = Sx\} \ne \emptyset$.

Theorem 2 *Suppose that $S : \mathbb{R}^n \to \mathbb{R}^n$ is nonexpansive, that $\mathrm{fix}S \ne \emptyset$, and that $\bar{\alpha} \in (0,1)$. Then the iteration (3)-(6) satisfies $r^k \to 0$ and $x^{k+1} \to x^k$ as $k \to \infty$.*

If a fixed-point to S exists, the fixed-point residual will converge to zero. Next, we establish what happens when no fixed-point to S exists.

Theorem 3 *Suppose that $S : \mathbb{R}^n \to \mathbb{R}^n$ is nonexpansive, that $\mathrm{fix}S = \emptyset$, that $\inf \|Sx - x\| = c > 0$, and that $\bar{\alpha} \in (0,1)$. Then the iteration (3)-(6) satisfies $r^k \to d$ and $x^{k+1} - x^k \to \bar{\alpha}d$ with $\|d\| = c$ as $k \to \infty$.*

This result relies heavily on [3, Proposition 4.5] (which is a specification of more general results in [8, Corollary 1.5] and [1, Corollary 2.3]). It says that, in the limit, the residual converges to a vector with smallest fixed-point residual. So the iterates converge to a line. This can, e.g., be used to devise infeasibility detection methods for these methods.

Next, we establish a rate bound for a difference of residuals.

Theorem 4 *Suppose that $S : \mathbb{R}^n \to \mathbb{R}^n$ is nonexpansive and $\bar{\alpha} \in (0,1)$. Then the iteration (3)-(6) satisfies*

$$\sum_{k=0}^{n} \|\bar{r}^k - r^k\|_2^2 \le \frac{\bar{\alpha}}{1 - \bar{\alpha}} \|r^0\|_2^2. \tag{8}$$

Let $k_{\mathrm{best}}^n \in \{0, \ldots, n\}$ be the iterate k (up to n) for which $\|\bar{r}^k - r^k\|_2$ is smallest. Then

$$\|\bar{r}^{k_{\mathrm{best}}^n} - r^{k_{\mathrm{best}}^n}\|_2^2 \le \frac{\bar{\alpha}}{(n+1)(1 - \bar{\alpha})} \|r^0\|_2^2. \tag{9}$$

If S is a δ-contraction with $\delta \in [0,1)$, i.e., $\|Sx - Sy\| \le \delta\|x - y\|$ for all $x, y \in \mathbb{R}^n$, stronger convergence results can be obtained.

Theorem 5 *Assume that $S : \mathbb{R}^n \to \mathbb{R}^n$ is δ-contractive with $\delta \in [0,1)$ and $\bar{\alpha} \in (0,1)$. Then the iteration (3)-(6) satisfies*

$$\|r^{k+1}\|_2 \le (1 - \bar{\alpha} + \bar{\alpha}\delta)\|r^k\|_2$$

for all iterations k.

So, the fixed-point residual converges linearly to zero (which it can since contractive operators always have a unique fixed-point).

Remark 1 *All results in this section are stated in the Euclidean setting with the standard 2-norm. But they also hold in general finite-dimensional real Hilbert space settings.*

3 Computational cost

The fixed-point residual must be evaluated to carry out the line search test (7). In the general case this requires us to evaluate the operator S, which has the same cost as a full iteration of the algorithm. Therefore, in the general case it may be too expensive to evaluate many (or even just more than one) candidate step lengths α_k compared to the savings in iterations due to the line search.

In this section we consider a special case in which the line search can be carried out more efficiently, i.e., many candidate points along the ray can be evaluated with low additional cost. Suppose that $S = S_2 S_1$, where $S_2 : \mathbb{R}^n \to \mathbb{R}^n$ is cheap to evaluate compared to S_1, and $S_1 : \mathbb{R}^n \to \mathbb{R}^n$ is affine. The algorithm (3)-(6) in this case becomes:

$$r^k := S_2 S_1 x^k - x^k \tag{10}$$

$$\bar{x}^k := x^k + \bar{\alpha} r^k \tag{11}$$

$$\bar{r}^k := S_2 S_1 \bar{x}^k - \bar{x}^k \tag{12}$$

$$x^{k+1} := x^k + \alpha_k r^k \tag{13}$$

In between (12) and (13), we perform the line search test (7),

$$\|r^{k+1}\|_2 = \|S_2 S_1 x^{k+1} - x^{k+1}\|_2 \leq (1 - \epsilon)\|\bar{r}^k\|_2, \tag{14}$$

for multiple candidate values of α_k.

We now analyze the complexity, assuming that the cost of evaluating S_2, and vector-vector operations, are negligible (or at least, dominated by the cost of evaluating S_1). In one iteration with line search we need to compute $S_1 x^k$ in (10), $S_1 \bar{x}^k$ in (12), and $S_1(x^k + \alpha_k r^k)$ for each candidate α_k in (14). Since S_1 is affine, i.e., of the form

$$S_1(x) = Fx + h \tag{15}$$

with $F \in \mathbb{R}^{n \times n}$ and $h \in \mathbb{R}^n$, we have for any α,

$$S_1(x^k + \alpha r^k) = Fx^k + h + \alpha F r^k.$$

So once we evaluate $F_2 x^k$ and $F_2 r^k$, we can evaluate $S_1(x^k + \alpha r^k)$ for any number of values of α, at the cost of only vector operations. In particular, we can evaluate $S_1 \bar{x}^k$ in step (12), and $S_1 x^{k+1}$ for multiple values of α_k in the line search test (14), with no further evaluations of S_1. We can express the first three steps of the algorithm as

$$r^k := S_2(Fx^k + h) - x^k \tag{16}$$

$$\bar{x}^k := x^k + \bar{\alpha} r^k \tag{17}$$

$$\bar{r}^k := S_2\left(Fx^k + h + \bar{\alpha} F r^k\right) - \bar{x}^k \tag{18}$$

which involves two evaluations of F (and two evaluations of S_2), and some vector operations. The next step is the line search, in which we evaluate the residual r using

$$r^{k+1} = S_2\left(Fx^k + h + \alpha_k F r^k\right) - (x^k + \alpha_k r^k) \tag{19}$$

for p candidate values of α_k. Each of these involves a few vector operations, and one evaluation of S_2, since we use the cached values of Fr^k and Fx^k. One

iteration costs $2 + p$ evaluations of S_2, 2 evaluations of F, and order p vector operations.

Finally, as observed above, we will have already evaluated the step (10) for the next iteration, so one evaluation of F (and S_2) does not count (or rather, counts towards the next iteration). Thus the computational cost of one iteration with p candidate values of α_k is one evaluation of S_1 (hence F) and $p + 1$ evaluations of S_2. If the cost of evaluating S_1 dominates the cost of evaluating S_2 (and vector operations), the computational cost of the iteration with line search is the same as the basic iteration without line search.

A variation. For some algorithms such as forward-backward splitting the averaged iteration (1) is more conveniently written as

$$x^{k+1} := T_2 T_1 x^k \tag{20}$$

where $T_2 : \mathbb{R}^n \to \mathbb{R}^n$ and $T_1 : \mathbb{R}^n \to \mathbb{R}^n$. So, in this case $(1-\bar{\alpha})x^k + \bar{\alpha}S_2 S_1 x^k = T_2 T_1 x^k$. (The nominal $\bar{\alpha}$ is hidden in the composition between T_2 and T_1.)

Instead of using $S_2 S_1 x - x$ as residuals in (10)-(13), we can use $\bar{\alpha}(S_2 S_1 x - x) = T_2 T_1 x - x$. An equivalent algorithm then becomes

$$r^k := T_2 T_1 x^k - x^k \tag{21}$$
$$\bar{x}^k := x^k + r^k \tag{22}$$
$$\bar{r}^k := T_2 T_1 \bar{x}^k - \bar{x}^k \tag{23}$$
$$x^{k+1} := x^k + \alpha_k r^k \tag{24}$$

where $\alpha_k \in [1, \alpha_{\max}]$.

Now, let T_1 be affine, i.e., of the form

$$T_1 x = Fx + h. \tag{25}$$

Then the steps (16)-(18) (with the x^{k+1} update) becomes

$$r^k := T_2(Fx^k + h) - x^k \tag{26}$$
$$\bar{x}^k := x^k + r^k \tag{27}$$
$$\bar{r}^k := T_2\left(Fx^k + h + Fr^k\right) - \bar{x}^k \tag{28}$$
$$x^{k+1} := x^k + \alpha_k r^k \tag{29}$$

The residual for the line search that is evaluated between (28) and (29) is computed as

$$r^{k+1} = T_2\left(Fx^k + h + \alpha_k Fr^k\right) - (x^k + \alpha_k r^k) \tag{30}$$

for multiple candidate values of α_k.

Evaluating affine operators. To evaluate the affine operator $S_1 : \mathbb{R}^n \to \mathbb{R}^n$ typically involves a matrix multiplication or a matrix inversion, where the matrix is the same in all iterations.

There are two main methods for repeated matrix inversion. The first is to factorize the matrix to be inverted once before the algorithm starts. Then forward and backward solves are used in every iteration. The cost of the forward and backward solves depends on the sparsity of the factors, but is typically more than $O(n)$ up to $O(n^2)$. The second option is to use an iterative method (with warm start). This requires a number of multiplications with the matrix to invert and is hence more expensive than $O(n)$.

Assuming that the cost of evaluating $S_2 : \mathbb{R}^n \to \mathbb{R}^n$ is $O(n)$, the cost of evaluating S_1 dominates the one of evaluating S_2 in this setting.

4 Optimization algorithms

Many popular optimization algorithms can be implemented with the proposed line search method. In this section, we show how S, S_2 and S_1 (or T_2 and T_1) look for some of these. Before this, we introduce some operators.

The proximal operator associated with a proper closed and convex $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is defined as

$$\text{prox}_{\gamma f}(z) := \underset{x}{\text{argmin}}\{f(x) + \tfrac{1}{2\gamma}\|x - z\|_2^2\} \tag{31}$$

where $\gamma > 0$. The reflected proximal operator is defined as

$$R_{\gamma f} := 2\text{prox}_{\gamma f} - I. \tag{32}$$

If f is the indicator function of a nonempty closed and convex set C, i.e.,

$$f(x) = \iota_C(x) := \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{else} \end{cases} \tag{33}$$

then the proximal operator in (31) is a projection:

$$\text{prox}_{\gamma f}(z) = \Pi_C(z) := \underset{x \in C}{\text{argmin}} \|x - z\|_2 \tag{34}$$

and the reflected proximal operator in (32) is $R_{\gamma \iota_C} = R_{\iota_C} = 2\Pi_C - I$.

4.1 Forward-backward splitting

The forward-backward splitting method (see, e.g., [10]) solves composite optimization problems of the form

$$\text{minimize } f(x) + g(x), \tag{35}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable with an L-Lipschitz continuous gradient ∇f and $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is proper closed and convex.

The forward-backward algorithm for this problem is

$$x^{k+1} := \text{prox}_{\gamma g}(x^k - \gamma \nabla f(x^k)), \tag{36}$$

where $\gamma \in (0, \frac{2}{L})$ is the step size and $\text{prox}_{\gamma g}$ is defined in (31).

If $\gamma \in (0, \frac{2}{L})$, it can be shown (by combining [2, Proposition 4.33], [2, Proposition 23.7, Remark 4.24)(iii)], and [11, Proposition 2.4] or [17, Proposition 3]) that

$$\text{prox}_{\gamma g}(I - \gamma \nabla f) = (1 - \bar{\alpha})I + \bar{\alpha}S$$

with $\bar{\alpha} = \frac{2}{4 - \gamma L}$, where

$$S = (1 - \frac{1}{\alpha})I + \frac{1}{\alpha}\text{prox}_{\gamma g}(I - \gamma \nabla f)$$

is nonexpansive. So, the forward-backward splitting algorithm (36) is an averaged iteration of a nonexpansive mapping with $\bar{\alpha} = \frac{2}{4 - \gamma L}$. So, if $\gamma \in (0, \frac{2}{L})$, we can do line search in forward-backward splitting.

We identify $T_2 = \text{prox}_{\gamma g}$ and $T_1 = (I - \gamma \nabla f)$ in (20). With these definitions, forward-backward splitting with line search is implemented as (21)-(24).

T_1 **affine.** The operator $T_1 = (I - \gamma \nabla f)$ is affine if $f : \mathbb{R}^n \to \mathbb{R}$ is convex quadratic, i.e., if

$$f(x) = \frac{1}{2}x^T P x + q^T x$$

with $P \in \mathbb{R}^{n \times n}$ positive semi-definite and $q \in \mathbb{R}^n$. The operator T_1 becomes

$$T_1 = (I - \gamma P)x - \gamma q.$$

Comparing to (25), we identify $F = I - \gamma P$ and $h = -\gamma q$. With these F and h, forward-backward splitting with line search can be implemented as in (26)-(29).

So a full iteration with line search needs only one multiplication with $F = (I - \gamma P)$. If in addition $T_2 = \text{prox}_{\gamma g}$ is cheap to evaluate, one full line search iteration can be evaluated roughly at the same cost as a basic iteration of the algorithm.

4.2   Douglas-Rachford splitting

The Douglas-Rachford splitting method [23] solves problems of the form

$$\text{minimize } f(x) + g(x),$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ are proper closed and convex.

The algorithm is given by the following iteration

$$x^k := \text{prox}_{\gamma f}(z^k) \tag{37}$$

$$y^k := \text{prox}_{\gamma g}(2x^k - z^k) \tag{38}$$

$$z^{k+1} := z^k + 2\alpha(y^k - x^k) \tag{39}$$

where γ is a positive scalar and $\alpha \in (0, 1)$.

Using the reflected proximal operator defined in (32) the Douglas-Rachford algorithm can be written as

$$z^{k+1} := ((1 - \alpha)I + \alpha R_{\gamma g} R_{\gamma f})z^k. \tag{40}$$

The reflected proximal operators $R_{\gamma g}$ and $R_{\gamma f}$ are nonexpansive [2, Corollary 23.10], and so is their composition $R_{\gamma g} R_{\gamma f}$.

The algorithm (40) is exactly on the form used in Section 3 where $S_2 = R_{\gamma g}$, $S_1 = R_{\gamma f}$, $S = R_{\gamma g} R_{\gamma f}$, and $\bar{\alpha} = \alpha$. With these definitions, Douglas-Rachford with line search can be implemented as (10)-(13).

Note that $R_{\gamma f} z^k = 2x^k - z^k$ in (37)-(39), $R_{\gamma g} R_{\gamma f} = 2y^k - 2x^k + z^k$ and the residual $r^k = R_{\gamma g} R_{\gamma f} z^k - z^k = 2(y^k - x^k)$.

S_1 **affine.** If $S_1 = R_{\gamma f}$ is affine and $S_2 = R_{\gamma g}$ is cheap to evaluate, the line search can be done almost for free, see Section 3.

The operator $S_1 = R_{\gamma f} = 2\mathrm{prox}_{\gamma f} - I$ is affine if $\mathrm{prox}_{\gamma f}$ is affine, which it is if f is of the form

$$f(x) = \begin{cases} \frac{1}{2}x^T P x + q^T x & \text{if } Ax = b \\ \infty & \text{else} \end{cases}$$

with $P \in \mathbb{R}^{n \times n}$ positive semi-definite, $q \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. (Any of the quadratic or linear functions, or the affine constraint can be removed, and the operator S_1 is still affine.) The proximal and reflected proximal operators of f become

$$\mathrm{prox}_{\gamma f}(z) = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} P + \gamma^{-1}I & A^T \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} \gamma^{-1}z - q \\ b \end{bmatrix}$$

$$R_{\gamma f}(z) = 2\mathrm{prox}_{\gamma f}(z) - z = 2 \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} P + \gamma^{-1}I & A^T \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} \gamma^{-1}z - q \\ b \end{bmatrix} - z$$

$$=: Fz + h$$

where $F \in \mathbb{R}^{n \times n}$ and $h \in \mathbb{R}^n$.

In this situation, the first three steps of the line search algorithm are (16)-(18) with $S_2 = R_{\gamma g}$ and the residual is (19). As shown in Section 3, we only need one evaluation of F per full iteration.

Note that in practice, the matrix F is typically not stored explicitly. One alternative is to factorize $\begin{bmatrix} P + \gamma^{-1}I & A^T \\ A & 0 \end{bmatrix}$ before the algorithm starts. This factorization is cached and used in all consecutive iterations to compute Fr^k (and Fz^0). Another option is to use an iterative method (with warm-start) to solve the corresponding linear system of equations.

4.3 ADMM

The alternating direction method of multipliers [20, 15, 5] solves problems of the form

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c, \end{array} \tag{41}$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ are proper closed convex, and $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, and $c \in \mathbb{R}^p$.

A standard form of ADMM (with scaled dual variable u and relaxation $\alpha \in (0, 1)$) is:

$$x^{k+1} = \operatorname*{argmin}_{x}\{f(x) + \tfrac{\rho}{2}\|Ax + Bz^k - c + u^k\|_2^2\} \tag{42}$$

$$x_A^{k+1} = 2\alpha Ax^{k+1} - (1 - 2\alpha)(Bz^k - c) \tag{43}$$

$$z^{k+1} = \operatorname*{argmin}_{z}\{g(z) + \tfrac{\rho}{2}\|x_A^{k+1} + Bz - c + u^k\|_2^2\} \tag{44}$$

$$u^{k+1} = u^k + (x_A^{k+1} + Bz^{k+1} - c) \tag{45}$$

where $\alpha = \tfrac{1}{2}$ gives standard ADMM without relaxation. This form of ADMM does not have a variable for which the algorithm is an averaged iteration of a nonexpansive mapping.

In Appendix B it is shown that ADMM is Douglas-Rachford splitting applied to a specific problem formulation. (This is a well known fact, see, e.g., [14, 13].) Therefore, ADMM is α-averaged and can be written on the form

$$v^{k+1} = (1 - \alpha)v^k + \alpha R_1 R_2 v^k \tag{46}$$

where $R_1 : \mathbb{R}^p \to \mathbb{R}^p$ and $R_2 : \mathbb{R}^p \to \mathbb{R}^p$ are reflected proximal operators. These reflected proximal operators are given by (see (74) and (76) in Appendix B where $\rho = \tfrac{1}{\gamma}$):

$$R_1(v) = 2A\operatorname*{argmin}_{x}\{f(x) + \tfrac{\rho}{2}\|Ax - v - c\|_2^2\} - 2c - v, \tag{47}$$

$$R_2(v) = -2B\operatorname*{argmin}_{z}\{g(z) + \tfrac{\rho}{2}\|Bz + v\|_2^2\} - v. \tag{48}$$

The algorithm (46) (and therefore ADMM in (42)-(45)) can then be implemented as (see Appendix B):

$$z^k := \operatorname*{argmin}_{z}\{g(z) + \tfrac{\rho}{2}\|Bz + v^k\|_2^2\} \tag{49}$$

$$x^k := \operatorname*{argmin}_{x}\{f(x) + \tfrac{\rho}{2}\|Ax + 2Bz^k + v^k - c\|_2^2\} \tag{50}$$

$$v^{k+1} := v^k + 2\alpha(Ax^k + Bz^k - c) \tag{51}$$

The iteration (46) is on the form discussed in Section 3 with $S_2 = R_1$, $S_1 = R_2$, $S = R_1 R_2$, and $\bar{\alpha} = \alpha$. With these definitions, ADMM with line search can be implemented as (10)-(13).

Note that $R_2 v^k = -2Bz^k - v^k$ in (49)-(51), $R_1 R_2 v^k = 2Ax^k - 2c + 2Bz^k + v^k$, and the residual $r^k = 2(Ax^k + Bz^k - c)$ in (51).

R_2 **affine.** If R_2 is affine and R_1 is cheap to evaluate, then line search can be performed efficiently, see Section 3.

The operator R_2 is affine if g is of the form

$$g(z) = \begin{cases} \frac{1}{2} z^T P z + q^T z & \text{if } Lz = b \\ \infty & \text{else} \end{cases}$$

with $P \in \mathbb{R}^{m \times m}$ positive semi-definite, $q \in \mathbb{R}^m$, $A \in \mathbb{R}^{s \times m}$, and $b \in \mathbb{R}^s$. The operator R_2 in (48) becomes

$$R_2(v) = \begin{bmatrix} -2B & 0 \end{bmatrix} \begin{bmatrix} P + \rho B^T B & L^T \\ L & 0 \end{bmatrix}^{-1} \begin{bmatrix} -(q + \rho B^T v) \\ b \end{bmatrix} - v$$

$$=: Fv + h$$

where $F \in \mathbb{R}^{p \times p}$ and $h \in \mathbb{R}^p$.

With these definitions of F and h, the first three steps of ADMM with line search is (16)-(18) with $S_2 = R_1$ and the residual is (19). Therefore, only one application of R_2 (and F) is needed per full line search iteration, see Section 3.

Also here, the matrix F is typically not stored explicitly. Instead, either a cached factorization of $\begin{bmatrix} P + \rho B^T B & L^T \\ L & 0 \end{bmatrix}$ or an iterative method (with warm-start) is used to compute Fr^k (and Fv^0).

4.4 Consensus

The consensus algorithm [5, Section 7] solves problems of the form

$$\text{minimize } f(x) = \sum_{i=1}^{N} f_i(x) \tag{52}$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and all $f_i : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ are proper closed and convex. An equivalent formulation is

$$\text{minimize } f_i(x_i) + \iota_C(x_1, \dots, x_N) \tag{53}$$

where the consensus constraint set C is

$$C = \{(x_1, \dots, x_N) \in \mathbb{R}^n \times \cdots \times \mathbb{R}^n \mid x_1 = \cdots = x_N\}$$

and ι_C is an indicator function defined in (33). That is, every $x_i \in \mathbb{R}^n$ in (53) is a local version of the global $x \in \mathbb{R}^n$ in (52).

We use the following formulation of the consensus algorithm:

$$x_i^k := \text{prox}_{\gamma f_i}(2z_{\text{av}}^k - z_i^k) \tag{54}$$

$$z_i^{k+1} := z_i^k + (x_i^k - z_{\text{av}}^k) \tag{55}$$

where $z_{\text{av}} = \frac{1}{N} \sum_{i=1}^{N} z_i$ is the average of the z_i's.

This consensus algorithm is obtained by applying Douglas-Rachford splitting with $\alpha = \frac{1}{2}$ to (53). (To use ADMM as in [5] would give an equivalent algorithm, see [13], but without a variable for which the algorithm is an averaged iteration.) Therefore, it is $\frac{1}{2}$-averaged and can be written on the form

$$\mathbf{z}^{k+1} := \tfrac{1}{2}(\mathbf{z}^k + R_{\gamma f} R_{\iota_C} \mathbf{z}^k) = \tfrac{1}{2}\left(\mathbf{z}^k + R_{\gamma f}(2z_{\mathrm{av}}^k - \mathbf{z}^k)\right)$$

where $\mathbf{z} = (z_1, \ldots, z_N)$. Using local variables, it can instead be written as

$$z_i^{k+1} := \tfrac{1}{2}\left(z_i^k + R_{\gamma f_i}(2z_{\mathrm{av}}^k - z_i^k)\right)$$

for all $i = \{1, \ldots, N\}$.

The local updates of the algorithm with line search become:

$$r_i^k := R_{\gamma f_i}(2z_{\mathrm{av}} - z_i^k) - z_i^k \tag{56}$$
$$\bar{z}_i^k := z_i^k + \tfrac{1}{2}r_i^k \tag{57}$$
$$\bar{r}_i^k := R_{\gamma f}(2\bar{z}_{\mathrm{av}}^k - \bar{z}_i^k) - \bar{z}_i^k \tag{58}$$
$$z_i^{k+1} := z_i^k + \alpha_k r_i^k \tag{59}$$

where either $\alpha_k = \frac{1}{2}$, or $\alpha_k \in (\frac{1}{2}, \alpha_{\max}]$ is chosen in accordance with (7), i.e., such that

$$\|\mathbf{r}^{k+1}\|_2 \leq (1 - \epsilon)\|\bar{\mathbf{r}}^k\|_2.$$

where $\mathbf{r}^k = (r_1^k, \ldots, r_N^k)$.

Note that the local residual r_i^k in (56) is given by $2(x_i^k - z_{\mathrm{av}}^k)$ in (55) (and similarly for $\bar{r}_i^k$ in (58)).

The operator R_{ι_C} is always affine. Therefore, a full iteration with line search can be performed with only one evaluation of R_{ι_C}, see Section 3. However, R_{ι_C} is often cheaper to evaluate than $R_{\gamma f}$. So, to evaluate a candidate point in the line search involves the costly operator $R_{\gamma f}$ and may be almost as costly as a full iteration of the algorithm.

4.5 Alternating projection methods

We consider the problem of finding a point in the intersection of two nonempty closed and convex sets C and D. That is, we want to find any $x \in C \cap D$. This can equivalently be written as solving the optimization problem

$$\text{minimize } \iota_C(x) + \iota_D(x) \tag{60}$$

where $\iota_C : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and $\iota_D : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ are indicator functions (defined in (33)) for C and D respectively .

There are numerous algorithms for finding such x. We focus on alternating projections and Douglas-Rachford splitting.

Alternating projections. The alternating projections [27] is given by

$$x^{k+1} = \Pi_C \Pi_D x^k. \tag{61}$$

Since Π_C and Π_D are $\frac{1}{2}$-averaged [2, Proposition 23.7], the composition is $\frac{2}{3}$-averaged [11, Proposition 2.4] or [17, Proposition 3]. Therefore, alternating projections is an averaged iteration with $\bar{\alpha} = \frac{2}{3}$ and of the form $x^{k+1} = T_2 T_1 x^k$ where $T_2 = \Pi_C$ and $T_1 = \Pi_D$.

Since alternating projections is an instance of (20), we can implement alternating projections with line search as (21)-(24) (with $T_2 = \Pi_C$ and $T_1 = \Pi_D$).

Douglas-Rachford. The problem (60) can also be solved using Douglas-Rachford splitting. The algorithm becomes

$$z^{k+1} = (1 - \alpha)z^k + \alpha R_{\iota_C} R_{\iota_D} z^k$$

where $\alpha \in (0, 1)$. That is, we have a composition of two reflections.

This algorithm is treated in Section 4.2 where we identified $R_{\iota_C} = S_2$ and $R_{\iota_D} = S_1$.

Remark 2 *Note that the γ parameter used in standard Douglas-Rachford is not present here (since the projection is independent of this). Therefore, the only parameter to be tuned is α, i.e., the one we perform line search over.*

D **affine.** When D is affine, i.e., $D = \{x \mid Ax = b\}$, then

$$\Pi_D(x) = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} x \\ b \end{bmatrix},$$

$$R_{\iota_D}(x) = 2\Pi_D(x) - x = \begin{bmatrix} 2I & 0 \end{bmatrix} \begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix}^{-1} \begin{bmatrix} x \\ b \end{bmatrix} - x.$$

Both these operators are affine.

Assume that Π_C (and hence $R_{\iota_C} = 2\Pi_C - I$) is cheap to evaluate. Then the line search can be implemented in alternating projections and in Douglas-Rachford splitting with almost no additional cost compared to a their basic iterations (see Section 3).

Alternating projections with line search is implemented as (26)-(29) with $T_2 = P_C$ and $Fx + h = \Pi_D$. The residual used for the line search is (30). The three first steps of Douglas-Rachford with line search is (16)-(18) with $S_2 = R_{\iota_C}$ and $Fx + h = R_{\iota_D}$. The residual used for the line search is (19).

4.6   Other algorithms

There are numerous other optimization algorithms that are averaged iterations of some nonexpansive mapping. For instance, forward-backward splitting for solving monotone inclusion problems and for solving Fenchel dual problems,

as well as projected and standard gradient methods fit the framework. The line search can also be used in Douglas-Rachford splitting for solving monotone inclusion problems. Also, preconditioned ADMM methods [9] can be interpreted as an averaged iteration of some nonexpansive mapping [21]. The recently proposed three operator splitting method in [12] is another example. Finally, the proximal point algorithm [25] for finding the zero of one maximally monotone operator is an averaged iteration. Actually, an algorithm is an averaged iteration of a nonexpansive mapping if and only if it is an instance of the proximal point method. Many of the methods mentioned above are discussed in [26].

5 Line search variations

There are numerous ways to create variations of the line search method. In this section, we list some that can improve practical convergence.

Line search activation. We do not need to perform line search in every iteration. Line search can be used in a subset of the iterations only. If a cheap test can indicate if a line search is beneficial, this can be used as activation rule for the line search.

Let $v^k = x^k - x^{k-1}$ be the difference between consecutive iterates. We have observed that if v^{k+1} and v^k are almost aligned, large step lengths α_k are typically accepted. If they are not aligned, we are typically restricted to smaller α_k. So, an activation rule could be that the cosine between the vectors v^{k+1} and v^k is large, i.e., that

$$\frac{(v^{k+1})^T v^k}{\|v^{k+1}\|_2 \|v^k\|_2} > 1 - \hat{\epsilon} \tag{62}$$

for some small $\hat{\epsilon} > 0$.

This is particularly useful for methods where the affine operator S_1 is not dominating (as in consensus). Even for methods where S_1 is dominating, this can be useful. In some cases we get fewer iterations when this activation rule is used, than if not.

Other candidate points. We are not restricted to perform the line search along the residual direction r^k. We can accept any candidate point $\hat{x}^{k+1}$ as the next iterate if its fixed-point residual is smaller than for the nominal point.

We introduce the residual function

$$r(x) = Sx - x. \tag{63}$$

Then we can replace the test in (7) with

$$\|r(\hat{x}^{k+1})\|_2 \leq (1 - \epsilon)\|r(\bar{x}^k)\|_2. \tag{64}$$

The full algorithm becomes

$$r^k := Sx^k - x^k$$

$$\bar{x}^k := x^k + \bar{\alpha} r^k$$

$$\bar{r}^k := S\bar{x}^k - \bar{x}^k$$

$$x^{k+1} := \begin{cases} \hat{x}^{k+1} & \text{if (64) holds} \\ x^k + \bar{\alpha} r^k & \text{else} \end{cases}$$

It is straightforward to verify that all convergence results for the residuals r^k in Section 2.2 still hold in this more general setting.

One special case is to perform line search along another direction d^k. Then the candidate point is $\hat{x}^{k+1} = x^k + \alpha_k d^k$. To evaluate the test in (64), we need to compute $S_2 S_1(x^k + \alpha_k d^k)$. One evaluation is in the general case as expensive as one iteration of the method. However, if $d^k = r^k$ and S_1 is affine, we saw in Section 3 that no additional S_1 applications are needed to perform the line search. If the direction d^k instead is a linear combination of previous residuals, i.e., $d^k = \sum_{i=0}^k \theta_i r^i$ where $\theta_i \in \mathbb{R}$, also no additional applications of S_1 are needed due to it being affine.

Another line search condition. Here, we present another line search test that does not compare progress with a nominal step, but with the last iterate that was decided by a line search. The progress is not measured with the residual function r in (63), but with a different function s.

To state the line search test, we let i_k be the index of the last iterate (up to the current iterate k) that was decided by a line search, i.e., that was not the result of a nominal step. Then any candidate point $\hat{x}^{k+1}$ can be accepted as the next iterate if the following conditions hold

$$\|s(\hat{x}^{k+1})\|_2 \leq (1-\epsilon)\|s(x^{i_k})\|_2 \quad \text{and} \quad \|r(\hat{x}^{k+1})\|_2 \leq C\|s(\hat{x}^{k+1})\|_2,$$

where C is a positive scalar, ϵ is a small positive scalar, and r is the residual function in (63). If these conditions are not satisfied, the algorithm instead takes a nominal step $x^{k+1} = x^k + \bar{\alpha} r^k$.

The convergence results in this setting become weaker. The rate results in Theorem 4 and 5 cannot be guaranteed. The results concerning the residual sequence r^k in Theorem 1, Theorem 2, and Theorem 3 can, however, be shown to hold. Let $k_0, k_1, k_2, \ldots$ be the iteration indices whose iterates have been decided by accepting a candidate line search point. Then

$$\|s(x^{k_p})\|_2 \leq (1-\epsilon)\|s(x^{k_{p-1}})\|_2 \leq (1-\epsilon)^p\|s(x^{k_0})\|_2,$$

which implies for iteration indices $k \in [k_{p+1}, k_p]$ that

$$\|r(x^k)\|_2 \leq \|r(x^{k_p})\|_2 \leq C\|s(x^{k_p})\|_2 \leq (1-\epsilon)^p\|s(x^{k_0})\|_2,$$

since $\{\|r(x^k)\|_2\}$ is a nonincreasing sequence in the basic method. If the tests are satisfied an infinite number of times, then $p \to \infty$ and $\|r(x^k)\|_2 \to 0$ as
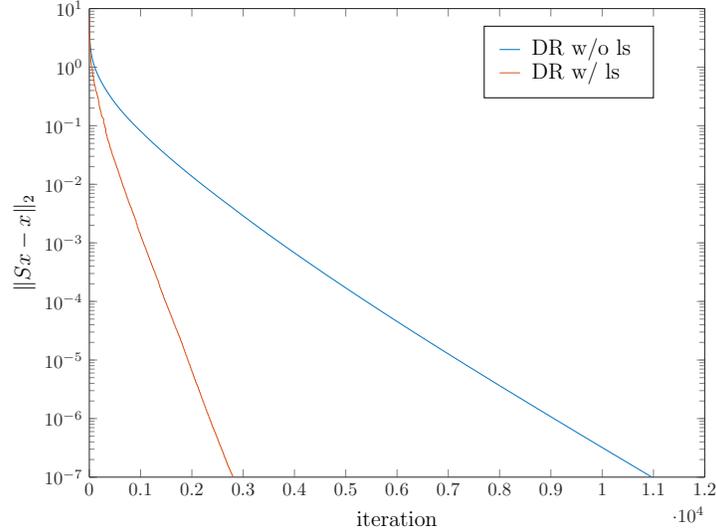
Figure 1: Fixed-point residual vs iteration for Douglas-Rachford with and without line search.

$k \to \infty$. If the tests are satisfied a finite number of times (which they are if, e.g., $\inf_x \|Sx - x\|_2 > 0$), the algorithm reduces to the basic iteration after a finite number of steps. Using these insights, the proofs to the results concerning the residual r^k in Theorem 1, Theorem 2, and Theorem 3 can easily be modified to show that the results hold also in this setting.

To improve performance, one might want to add a condition that accepts a candidate point if there is an improvement compared to the previous iterate, i.e., if the following condition is satisfied

$$\|s(\hat{x}^{k+1})\|_2 \le (1 - \epsilon)\|s(x^k)\|_2.$$

This condition is, however, not needed to guarantee convergence of the method.

6 Numerical examples

6.1 Nonnegative least squares

To evaluate the efficiency of the line search, we solve a nonnegative least squares problem using the Douglas-Rachford algorithm. The problem is of the form

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2^2 \\ \text{subject to} & x \ge 0 \end{array}$$

where $A \in \mathbb{R}^{1000 \times 1000}$ is dense and $b \in \mathbb{R}^{1000}$.
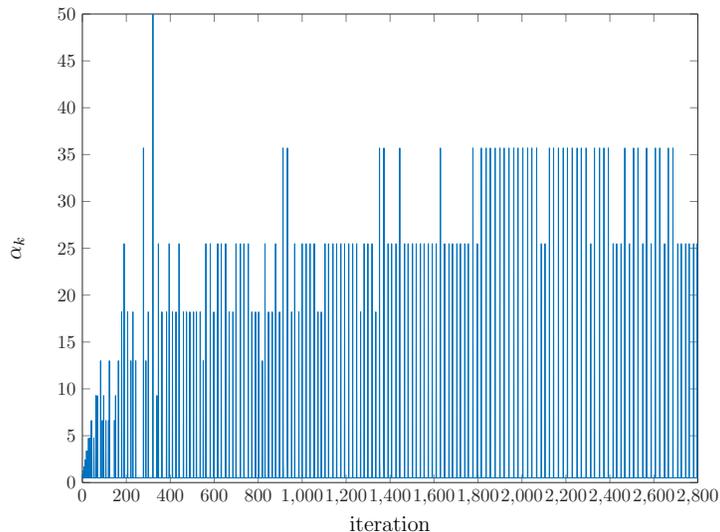
Figure 2: Step length α_k vs iteration in the line search method.

The entries in the data matrix A are drawn from a normal distribution with zero mean and unit variance. Then, each row of A is scaled with a uniformly distributed random number between 0.1 and 1.1 to worsen the conditioning of the problem. The entries in b are drawn from a normal distribution with zero mean and unit variance.

To fit the Douglas-Rachford framework, we let $f(x) = \|Ax - b\|_2^2$ and $g(x) = \iota(x \geq 0)$. The operator $\mathrm{prox}_{\gamma f}$ is affine and the operator $\mathrm{prox}_{\gamma g}$ is (very) cheap to evaluate compared to $\mathrm{prox}_{\gamma f}$. Therefore, this problem is on the form discussed in Section 3. So an iteration with line search is just slightly more expensive than performing a basic iteration of the algorithm.

In the line search test (14), we let $\epsilon = 0.03$ (which may or may not be a good choice in other examples) and α_k is decided using back-tracking from $\alpha_{\max} = 50$ with a factor $1/1.4$ for each candidate α. The back-tracking is stopped either when the test is satisfied, or when the candidate $\alpha \leq \bar{\alpha}$, in which case $\alpha_k = \bar{\alpha}$. This gives a worst case of 14 line search test points.

The computational cost for $\mathrm{prox}_{\gamma f}$ is roughly $2n^2$ after an initial matrix factorization. The cost for $\mathrm{prox}_{\gamma g}$ is, on the other hand, roughly n. To evaluate the line search test, no additional $\mathrm{prox}_{\gamma f}$ computations are needed. But about 10 vector additions or multiplications with scalars and one $\mathrm{prox}_{\gamma g}$ is needed for every candidate point (the same as in the standard algorithm). So, evaluating one candidate point costs approximately $10n$. A worst case of 14 candidate points costs $140n$ for a full line search. Comparing this to the cost for one basic iteration, $2n^2 + 10n$, gives, when $n = 1000$, that one iteration with line search costs, in the worst case, 1.07 times a basic iteration.

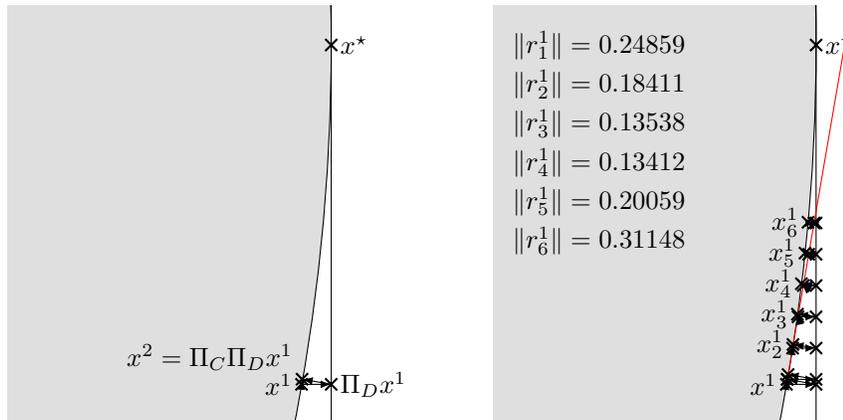Figure 1 shows the fixed-point residual vs iteration number for Douglas-

Figure 3: The left figure shows one iteration of alternating projections. The residual in this figure is $r^1 = x^2 - x^1$. In the right figure, an alternating projections step with line search is performed. The residual direction is shown in red. We evaluate six candidate points x_i^1, $i \in \{1, \ldots, 6\}$, along this line. (The points themselves, $\Pi_D x_i^1$ and $\Pi_C \Pi_D x_i^1$ are marked with crosses in the figure.) The norm of each residual $r_i^1 = \Pi_C \Pi_D x_i^1 - x_i^1$ is printed in the figure. The 4th point x_4^1 has smallest residual norm. This corresponds to $\alpha_k = 19.75$. Another option is to choose the farthest candidate point with residual norm smaller than $\|r_1^1\|$. This holds for the fifth point with $\alpha_k = 26$. Both these choices are convergent. In this case we get closer to the intersection point by choosing the farthest point.

Rachford with and without line search (the Douglas-Rachford parameters are chosen to be $\bar{\alpha} = \frac{1}{2}$ and $\gamma = 3$). For this example, the number of iterations is reduced by roughly a factor four. The improvement in execution time is roughly the same because of the modest 7% increase in computational cost due to the line search.

Figure 2 shows what values α_k that are chosen in the line search. An $\alpha_k = \bar{\alpha}$ corresponds to a standard Douglas-Rachford iteration. In 175 out of the 2800 iterations, an $\alpha_k > \bar{\alpha}$ was selected. Among these 158 had $\alpha_k > 5$.

6.2 An alternating projections example

To visualize the line search, we solve a two dimensional feasibility problem using alternating projections.

We want to find a point in the intersection between two sets $C = \{x \in \mathbb{R}^2 \mid \|x\| \leq 1\}$ and $D = \{x \in \mathbb{R}^2 \mid x = (x_1, x_2), x_1 = 1\}$. So C is the unit circle, and D is a vertical line that touches the boundary of C at $(1, 0)$. The unique intersection point is $x^\star = (1, 0)$.

In Figure 3 we show one iteration of the standard alternating projections algorithm and one iteration with line search. In Figure 4 we show 50 steps of alternating projections.
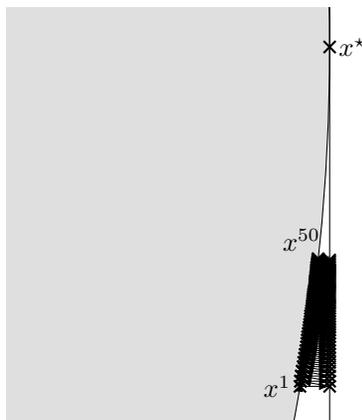
Figure 4: This figure shows 50 iterations of alternating projections. Comparing to Figure 3 reveals that roughly 50 steps of alternating projections give the same progression as one step with line search (when the farthest acceptable point is chosen) in this example.

We see that the progression in 50 steps of alternating projections is roughly the same as the progression of one step with line search (when the farthest acceptable candidate point is chosen). The line search scheme does, on the other hand, compute six candidate points to advance this far. (Or really five, since the first is the basic next step.) So, we gain roughly a factor 10 in this step.

This is just a simple example where both projections are very cheap. If the cost of projecting onto the subspace is dominating the other cost of the other projection. Then the cost of performing one iteration with line search is roughly the same as the cost of one basic iteration. In such cases, we can gain a lot by performing line search.

7   Acknowledgments

References

[1] J. B. Baillon, R. E. Bruck, and S. Reich. On the asymptotic behavior of nonexpansive mappings and semigroups in Banach spaces. *Houston Journal of Mathematics*, 4:1–9, 1978.

[2] H. H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces.* Springer, 2011.

[3] H. H. Bauschke and W. M. Moursi. The Douglas-Rachford algorithm for two (not necessarily intersecting) affine subspaces. `http://arxiv.org/abs/1504.03721`, 2015.

[4] M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, New York, NY, 2004.

[7] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev. Analysis of the inexact Uzawa algorithm for saddle point problems. *SIAM Journal on Numerical Analysis*, 34(3):1072–1092, 1997.

[8] R. E. Bruck and S. Reich. Nonexpansive projections and resolvents of accretive operators in Banach spaces. *Houston Journal of Mathematics*, 3:459–470, 1977.

[9] 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(1):120–145, 2011.

[10] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM journal on Multiscale Modeling and Simulation*, 4(4):1168–1200, 2005.

[11] P. L. Combettes and I. Yamada. Compositions and convex combinations of averaged nonexpansive operators. *Journal of Mathematical Analysis and Applications*, 425(1):55–70, 2015.

[12] D. Davis and W. Yin. A three-operator splitting scheme and its optimization applications. `http://arxiv.org/abs/1504.01032`, 2015.

[13] J. Eckstein. *Splitting methods for monotone operators with applications to parallel optimization.* PhD thesis, MIT, 1989.

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

[15] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17–40, 1976.

[16] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, March 2015.

[17] P. Giselsson. Tight global linear convergence rate bounds for Douglas-Rachford splitting. 2015. Submitted. Available: `http://arxiv.org/abs/1506.01556`.

[18] P. Giselsson and S. Boyd. Metric selection in fast dual forward-backward splitting. *Automatica*, 62:1–10, 2015.

[19] P. Giselsson and S. Boyd. Linear convergence and metric selection in Douglas-Rachford splitting and ADMM. 2016. Accepted for publication in Transactions on Automatic Control. Available: `http://arxiv.org/abs/1410.8479`.

[20] R. Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problémes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 9:41–76, 1975.

[21] B. He and X. Yuan. Convergence analysis of primal-dual algorithms for a saddle-point problem: From contraction perspective. *SIAM Journal on Imaging Sciences*, 5(1):119–149, 2012.

[22] Q. Hu and J. Zou. Nonlinear inexact Uzawa algorithms for linear and nonlinear saddle-point problems. *SIAM Journal on Optimization*, 16(3):798–825, 2006.

[23] P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

[24] J. Nocedal and S. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2nd edition, 2006.

[25] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[26] E. K. Ryu and S. Boyd. Primer on monotone operator methods. *Appl. Comput. Math.*, 15(1), 2016. To appear.

[27] J. von Neumann. *Functional Operators. Volume II. The Geometry of Orthogonal Spaces*. Princeton University Press: Annals of Mathematics Studies, 1950. Reprint of 1933 lecture notes.

A Proofs to results in Section 2

A.1 Proof to Theorem 1

First, we show that $\|r^k\|_2 = \|x^k - Sx^k\|_2 \to c$ as $k \to \infty$. We show this by considering the cases $\alpha_k = 1$ and $\alpha_k > 1$ separately.

First, we consider the case $\alpha_k = 1$. For convenience, we introduce the operator $T = (1 - \bar{\alpha})I + \bar{\alpha}S$. Then the update for $\bar{x}^k$ in (4) can be written as

$$\bar{x}^k = x^k + \bar{\alpha}(Sx^k - x^k) = (1 - \bar{\alpha})x^k + \bar{\alpha}Sx^k = Tx^k.$$

Noting that $\|x - Tx\|_2 = \|x - (1 - \bar{\alpha})x - \bar{\alpha}Sx\|_2 = \bar{\alpha}\|x - Sx\|_2$ implies

$$\|r^{k+1}\|_2 = \|\bar{r}^k\|_2 = \|\bar{x}^k - S\bar{x}^k\|_2 = \tfrac{1}{\bar{\alpha}}\|\bar{x}^k - T\bar{x}^k\|_2 = \tfrac{1}{\bar{\alpha}}\|Tx^k - TTx^k\|_2.$$

Therefore, since T is nonexpansive:

$$\|r^{k+1}\|_2 \leq \tfrac{1}{\bar{\alpha}}\|x^k - Tx^k\|_2 = \|x^k - Sx^k\|_2 = \|r^k\|_2. \tag{65}$$

Next, we consider the case where $\alpha_k > 1$. Since $\|\bar{r}^k\|_2 \leq \|r^k\|_2$, we get from the line search test (7) that

$$\|r^{k+1}\|_2 \leq (1 - \epsilon)\|\bar{r}^k\|_2 \leq (1 - \epsilon)\|r^k\|_2. \tag{66}$$

So $\{\|r^k\|_2\}_{k=1}^{\infty}$ is a decreasing sequence which is bounded below (by 0). Hence it converges. This completes the proof.

A.2 Proof to Theorem 2

Combining (65) and (66), we get

$$\|r^{k+1}\|_2 \leq (1 - \epsilon)^{k_0}\|r^0\|_2 \tag{67}$$

where k_0 is the number of times that α_k satisfies $\alpha_k > 1$. If $k_0 \to \infty$ as $k \to \infty$, then $\|r^{k+1}\|_2 \to 0$ as $k \to \infty$. On the other hand, if k_0 stays finite as $k \to \infty$, there exists a finite $k_{\max}$ after which the line search is not activated again. Then for $k \geq k_{\max}$, the algorithm reduces to $x^{k+1} = Tx^k$, which satisfies $\|r^k\|_2 = \|x^k - Sx^k\|_2 = \tfrac{1}{\bar{\alpha}}\|x^k - Tx^k\|_2 \to 0$ as $k \to \infty$, see [2, Theorem 5.14]. This concludes the proof.

A.3 Proof to Theorem 3

Combining (65) and (66), we get

$$\|r^{k+1}\|_2 \leq (1 - \epsilon)^{k_0}\|r^0\|_2 \tag{68}$$

where k_0 is the number of times that α_k satisfies $\alpha_k > 1$. If $k_0 \to \infty$ as $k \to \infty$, then $\|r^{k+1}\|_2 \to 0$ as $k \to \infty$. This is a contradiction to that $\inf \|Sx - x\|_2 > 0$.

Hence k_0 must be finite and there exists a $k_{\max}$ after which the algorithm reduces to the basic averaged iteration.

Let $T = (1 - \bar{\alpha})I + \bar{\alpha}S$, $x^{k_{\max}} = \tilde{x}_0$ and $\Delta k = k - k_{\max}$. Then a straightforward generalization of [3, Proposition 4.5] to allow for averaged operators (instead of only firmly nonexpansive or $\frac{1}{2}$-averaged) gives that

$$\|\bar{\alpha}r^k - v\| = \|x^k - x^{k+1} - v\| = \|T^{\Delta k}\tilde{x}_0 - T^{\Delta k+1}\tilde{x}_0 - v\| \to 0$$

for a specific v. Therefore $r^k \to \frac{1}{\bar{\alpha}}v =: d$ as $k \to \infty$. Further, $x^{k+1} - x^k = \bar{\alpha}r^k \to \bar{\alpha}d$ as $k \to \infty$.

The v is the *infimal displacement vector* (see [3, Fact 2.2]) that satisfies $v \in \overline{\mathrm{ran}}(I - T)$ (i.e., v is in the closure of the range of $I - T$) and $\|v\|_2 = \inf_x \|x - Tx\|_2$. Therefore $\|d\|_2 = \frac{1}{\bar{\alpha}}\|v\|_2 = \frac{1}{\bar{\alpha}}\inf_x \|x - Tx\|_2 = \inf_x \|x - Sx\|_2$. This concludes the proof.

A.4   Proof to Theorem 4

We need the following lemma for this proof.

Lemma 1 *Suppose that* $S : \mathbb{R}^n \to \mathbb{R}^n$ *nonexpansive and that* $\bar{\alpha} \in (0,1)$. *Then every iteration of* (3)-(6) *satisfies*

$$\bar{\alpha}(1 - \bar{\alpha})\|\bar{r}^k - r^k\|_2^2 \le \|x^k - \bar{x}^k\|_2^2 - \|x^{k+1} - \bar{x}^{k+1}\|_2^2. \tag{69}$$

Proof.   Let $T = (1 - \bar{\alpha})I + \bar{\alpha}S$. Then T is $\bar{\alpha}$-averaged, and it satisfies [2, Proposition 4.25(iii)]

$$\tfrac{1-\bar{\alpha}}{\bar{\alpha}}\|(I - T)\bar{x}^k - (I - T)x^k\|_2^2 \le \|x^k - \bar{x}^k\|_2^2 - \|Tx^k - T\bar{x}^k\|_2^2.$$

Now, since $(I - T)x = (I - (1 - \bar{\alpha})I - \bar{\alpha}S)x = \bar{\alpha}(I - S)x$, we have $(I - T)x^k = \bar{\alpha}r^k$ and $(I - T)\bar{x}^k = \bar{\alpha}\bar{r}^k$. Therefore

$$\bar{\alpha}(1 - \bar{\alpha})\|\bar{r}^k - r^k\|_2^2 \le \|x^k - \bar{x}^k\|_2^2 - \|Tx^k - T\bar{x}^k\|_2^2.$$

The algorithm chooses either $\alpha_k = \bar{\alpha}$ or $\alpha_k > \bar{\alpha}$. If $\alpha_k = \bar{\alpha}$, we have $Tx^k = \bar{x}^k = x^{k+1}$ and $T\bar{x}^k = Tx^{k+1} = \bar{x}^{k+1}$. Therefore

$$\begin{aligned}
\bar{\alpha}(1 - \bar{\alpha})\|\bar{r}^k - r^k\|_2^2 &\le \|x^k - \bar{x}^k\|_2^2 - \|Tx^k - T\bar{x}^k\|_2^2 \\
&= \|x^k - \bar{x}^k\|_2^2 - \|x^{k+1} - \bar{x}^{k+1}\|_2^2.
\end{aligned}$$

If instead $\alpha_k > \bar{\alpha}$, we get

$$\begin{aligned}
\bar{\alpha}(1 - \bar{\alpha})\|\bar{r}^k - r^k\|_2^2 &\le \|x^k - \bar{x}^k\|_2^2 - \|Tx^k - T\bar{x}^k\|_2^2 \\
&= \|x^k - \bar{x}^k\|_2^2 - \|\bar{x}^k - T\bar{x}^k\|_2^2 \\
&\le \|x^k - \bar{x}^k\|_2^2 - \tfrac{1}{(1-\epsilon)^2}\|x^{k+1} - Tx^{k+1}\|_2^2 \\
&\le \|x^k - \bar{x}^k\|_2^2 - \|x^{k+1} - Tx^{k+1}\|_2^2 \\
&= \|x^k - \bar{x}^k\|_2^2 - \|x^{k+1} - \bar{x}^{k+1}\|_2^2
\end{aligned}$$

where the second inequality holds due to the line search test in (7) and the third inequality holds since $\epsilon \in (0,1)$. Therefore (69) holds for all k and the proof is complete. $\qquad\square$

Now we are ready to prove the result. A telescope summation of (69) gives

$$\bar{\alpha}(1-\bar{\alpha})\sum_{k=0}^{n}\|\bar{r}^k - r^k\|_2^2 \leq \|x^0 - \bar{x}^0\|_2^2 = \bar{\alpha}^2\|r^0\|_2^2.$$

This proves (8). To prove (9), we note that $k_{\text{best}}^n \in \{0,\ldots,n\}$ is the iteration k (up till n) with smallest $\|\bar{r}^k - r^k\|_2$. Therefore

$$(n+1)\|\bar{r}^{k_{\text{best}}^n} - r^{k_{\text{best}}^n}\|_2^2 \leq \sum_{k=0}^{n}\|\bar{r}^k - r^k\|_2^2 \leq \frac{\bar{\alpha}}{1-\bar{\alpha}}\|r^0\|_2^2.$$

This concludes the proof.

A.5 Proof to Theorem 5

First, we introduce $T = (1-\bar{\alpha})I + \bar{\alpha}S$ which is $\bar{\alpha}$-averaged, and satisfies $\|x - Sx\|_2 = \frac{1}{\bar{\alpha}}\|x - Tx\|_2$. Let's consider the case when $\alpha_k = \bar{\alpha}$. Then $\bar{x}^k = Tx^k$ and

$$\|r^{k+1}\|_2 = \|\bar{r}^k\|_2 = \|\bar{x}^k - S\bar{x}^k\|_2 = \tfrac{1}{\bar{\alpha}}\|\bar{x}^k - T\bar{x}^k\|_2 = \tfrac{1}{\bar{\alpha}}\|Tx^k - TTx^k\|_2$$
$$= \tfrac{1}{\bar{\alpha}}\|(1-\bar{\alpha})(x^k - Tx^k) + \bar{\alpha}(Sx^k - STx^k)\|_2.$$

The triangle inequality gives that

$$\begin{aligned}
\|r^{k+1}\|_2 &\leq \tfrac{1}{\bar{\alpha}}((1-\bar{\alpha})\|x^k - Tx^k\|_2 + \bar{\alpha}\|Sx^k - STx^k\|_2)\\
&\leq \tfrac{1}{\bar{\alpha}}((1-\bar{\alpha})\|x^k - Tx^k\|_2 + \bar{\alpha}\delta\|x^k - Tx^k\|_2)\\
&= \tfrac{1}{\bar{\alpha}}(1-\bar{\alpha}+\bar{\alpha}\delta)\|x^k - Tx^k\|_2\\
&= (1-\bar{\alpha}+\bar{\alpha}\delta)\|x^k - Sx^k\|_2\\
&= (1-\bar{\alpha}+\bar{\alpha}\delta)\|r^k\|_2.
\end{aligned}$$

Next, we consider the case when $\alpha_k > \bar{\alpha}$. Since $\|\bar{r}^k\|_2 \leq (1-\bar{\alpha}+\bar{\alpha}\delta)\|r^k\|_2$ the line search test (7) implies that

$$\|r^{k+1}\|_2 \leq (1-\epsilon)\|\bar{r}^k\|_2 \leq (1-\epsilon)(1-\bar{\alpha}+\bar{\alpha}\delta)\|r^k\|_2 \leq (1-\bar{\alpha}+\bar{\alpha}\delta)\|r^k\|_2.$$

That is, the algorithm is linearly convergent with factor (at most) $(1-\bar{\alpha}+\bar{\alpha}\delta)$ in both situations. This concludes the proof.

B ADMM derivation

In this section, we show the equivalence between the standard ADMM formulation (42)-(45) and the ADMM version used for line search (49)-(51). We also

show that the version used for line search, (49)-(51), is an α-averaged iteration of a nonexpansive mapping.

We do this by showing that the ADMM iterations can be derived by applying Douglas-Rachford splitting to a specific problem formulation. This derivation is not new [14, 13], but we include it here for completeness and to explicitly arrive that the ADMM variation (49)-(51) that we need for the line search.

ADMM solves problems of the form

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array} \qquad (70)$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ are proper closed convex, $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, and $c \in \mathbb{R}^p$.

Using image functions (that are also called infimal postcompositions) defined as

$$(L \triangleright \psi)(y) = \inf\{\psi(x) \mid Lx = y\}$$

where $L \in \mathbb{R}^{n \times m}$ is a linear operator and $\psi : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper function, it is straightforward to verify that (70) is equivalent to

$$\text{minimize } (-A \triangleright f)(-u - c) + (-B \triangleright g)(u).$$

Let $p_1(u) = (-A \triangleright f)(-u - c)$ and $p_2(u) = (-B \triangleright g)(u)$ to get the equivalent problem

$$\text{minimize } p_1(u) + p_2(u). \qquad (71)$$

To arrive at the standard ADMM iterations, we apply Douglas-Rachford splitting to (71). The algorithm becomes

$$v^{k+1} = (1 - \alpha)v^k + \alpha R_{\gamma p_1} R_{\gamma p_2} v^k \qquad (72)$$

where the reflected proximal operators $R_{\gamma p_1}$ and $R_{\gamma p_2}$ are given by $R_{\gamma p_1} = 2\text{prox}_{\gamma p_1} - I$ and $R_{\gamma p_2} = 2\text{prox}_{\gamma p_2} - I$. Under the assumption that the infimum over x is attained in the following prox evaluation, we have

$$\begin{aligned} \text{prox}_{\gamma p_1}(v) &= \operatorname*{argmin}_{u}\{\inf_{x}\{f(x) \mid -Ax = -u - c\} + \tfrac{1}{2\gamma}\|u - v\|_2^2\} \\ &= A \operatorname*{argmin}_{x}\{f(x) + \tfrac{1}{2\gamma}\|Ax - v - c\|_2^2\} - c. \end{aligned} \qquad (73)$$

The reflected proximal operator becomes

$$R_{\gamma p_1}(v) = 2A \operatorname*{argmin}_{x}\{f(x) + \tfrac{1}{2\gamma}\|Ax - v - c\|_2^2\} - 2c - v. \qquad (74)$$

Again, assuming that the following infimum is attained, we get

$$\begin{aligned} \text{prox}_{\gamma p_2}(v) &= \operatorname*{argmin}_{u}\{\inf_{z}\{g(z) \mid -Bz = u\} + \tfrac{1}{2\gamma}\|u - v\|_2^2\} \\ &= -B \operatorname*{argmin}_{z}\{g(z) + \tfrac{1}{2\gamma}\|Bz + v\|_2^2\} \end{aligned} \qquad (75)$$

and reflected proximal operator

$$R_{\gamma p_2}(v) = -2B \operatorname*{argmin}_z \{g(z) + \tfrac{1}{2\gamma}\|Bz + v\|_2^2\} - v. \tag{76}$$

Using the prox expressions (73) and (75), and defining $\rho = \frac{1}{\gamma}$, we find that the Douglas-Rachford algorithm (37)-(39) applied to (71) becomes

$$z^k = \operatorname*{argmin}_z \{g(z) + \tfrac{\rho}{2}\|Bz + v^k\|_2^2\} \tag{77}$$

$$x^k = \operatorname*{argmin}_x \{f(x) + \tfrac{\rho}{2}\|Ax + 2Bz^k + v^k - c\|_2^2\} \tag{78}$$

$$v^{k+1} = v^k + 2\alpha(Ax^k + Bz^k - c) \tag{79}$$

This is exactly the iteration (49)-(51) which is used in the line search. This algorithm is equivalent to ADMM, but keeps the v^k variables in which the algorithm can be interpreted as an averaged iteration of a nonexpansive mapping, see (72).

To derive the ADMM iterations (42)-(45), we next substitute $v^{k+1} = u^k + 2\alpha(Ax^k - c) - (1 - 2\alpha)Bz^k$. Let $x_A^k = 2\alpha Ax^k - (1 - 2\alpha)(Bz^k - c)$ to get $v^{k+1} = u^k + x_A^k - c$ and

$$z^k = \operatorname*{argmin}_z \{g(z) + \tfrac{\rho}{2}\|x_A^{k-1} + Bz - c + u^{k-1}\|_2^2\}$$

$$x^k = \operatorname*{argmin}_x \{f(x) + \tfrac{\rho}{2}\|Ax + 2Bz^k + u^{k-1} + x_A^{k-1} - 2c\|_2^2\}$$

$$u^k = u^{k-1} + (x_A^{k-1} + Bz^k - c)$$

since $v^{k+1} = u^k + x_A^k - c$ inserted in (79) implies

$$u^k = u^{k-1} + x_A^{k-1} - x_A^k + 2\alpha(Ax^k + Bz^k - c)$$
$$= u^{k-1} + x_A^{k-1} - (2\alpha Ax^k - (1 - 2\alpha)(Bz^k - c)) + 2\alpha(Ax^k + Bz^k - c)$$
$$= u^{k-1} + (x_A^{k-1} + Bz^k - c)$$

(This implies that $v^k = u^k - Bz^k$.) Next, insert the third equation into the second to get

$$z^k = \operatorname*{argmin}_z \{g(z) + \tfrac{\rho}{2}\|x_A^{k-1} + Bz - c + u^{k-1}\|_2^2\}$$

$$x^k = \operatorname*{argmin}_x \{f(x) + \tfrac{\rho}{2}\|Ax + Bz^k - c + u^k\|_2^2\}$$

$$u^k = u^{k-1} + (x_A^{k-1} + Bz^k - c)$$

Now, change order of the x^k update and the u^k update and move the x^k update to the first line and insert x_A^{k-1} to get

$$x^{k-1} = \operatorname*{argmin}_x \{f(x) + \tfrac{\rho}{2}\|Ax + Bz^{k-1} - c + u^{k-1}\|_2^2\}$$

$$x_A^{k-1} = 2\alpha Ax^{k-1} - (1 - 2\alpha)(Bz^{k-1} - c)$$

$$z^k = \operatorname*{argmin}_z \{g(z) + \tfrac{\rho}{2}\|x_A^{k-1} + Bz - c + u^{k-1}\|_2^2\}$$

$$u^k = u^{k-1} + (x_A^{k-1} + Bz^k - c)$$

Now, let $x^k \to x^{k+1}$ and $x_A^k \to x_A^{k+1}$ to get

$$x^k = \underset{x}{\operatorname{argmin}}\{f(x) + \tfrac{\rho}{2}\|Ax + Bz^{k-1} - c + u^{k-1}\|_2^2\}$$
$$x_A^k = 2\alpha A x^k - (1 - 2\alpha)(Bz^{k-1} - c)$$
$$z^k = \underset{z}{\operatorname{argmin}}\{g(z) + \tfrac{\rho}{2}\|x_A^k + Bz - c + u^{k-1}\|_2^2\}$$
$$u^k = u^{k-1} + (x_A^k + Bz^k - c)$$

Letting $k \to k+1$ gives ADMM on the standard form (42)-(45).

Remark 3 *ADMM can also be derived by applying Douglas-Rachford to the Fenchel dual of (70), see [14]. The Fenchel dual is*

$$minimize\ f^*(-A^T\mu) + c^T\mu + g^*(-B^T\mu).$$

Letting $d_1(\mu) := f^*(-A^T\mu) + c^T\mu$ and $d_2(\mu) := g^*(-B^T\mu)$, this is equivalent to

$$minimize\ d_1(\mu) + d_2(\mu).$$

It holds that $p_1^* = d_1$ and $p_2^* = d_2$, see [2, Corollary 15.28]. It is also known that Douglas-Rachford when applied to minimize $p_1 + p_2$ is equivalent to applying Douglas-Rachford to minimize $p_1^* + p_2^*$ (which is $d_1 + d_2$), see [13]. Therefore we can also apply Douglas-Rachford to this dual formulation to get ADMM. This derivation is longer and therefore not used here.