Finding best approximation pairs for two intersections of closed convex sets

Heinz H. Bauschke,* Shambhavi Singh,† and Xianfu Wang‡

October 15, 2021

Abstract

The problem of finding a best approximation pair of two sets, which in turn generalizes the well known convex feasibility problem, has a long history that dates back to work by Cheney and Goldstein in 1959.

In 2018, Aharoni, Censor, and Jiang revisited this problem and proposed an algorithm that can be used when the two sets are finite intersections of halfspaces. Motivated by their work, we present alternative algorithms that utilize projection and proximity operators. Our modeling framework is able to accommodate even convex sets. Numerical experiments indicate that these methods are competitive and sometimes superior to the one proposed by Aharoni et al.

2020 Mathematics Subject Classification: Primary 65K05; Secondary 47H09, 90C25.

Keywords: Aharoni–Censor–Jiang algorithm, best approximation pair, Douglas–Rachford algorithm, dual-based proximal method, proximal distance algorithm, stochastic subgradient descent.

*Mathematics, University of British Columbia, Kelowna, B.C. V1V 1V7, Canada. E-mail: heinz.bauschke@ubc.ca.
†Mathematics, University of British Columbia, Kelowna, B.C. V1V 1V7, Canada. E-mail: sambha@student.ubc.ca.
‡Mathematics, University of British Columbia, Kelowna, B.C. V1V 1V7, Canada. E-mail: shawn.wang@ubc.ca.
1 Introduction

Throughout this paper, we assume that $Y$ is a finite-dimensional real Hilbert space with inner product $\langle \cdot, \cdot \rangle : Y \times Y \to \mathbb{R}$ and induced norm $\| \cdot \|$. Let $m \in \{1, 2, \ldots \}$, set $I := \{1, \ldots, m\}$ and suppose that $(\forall i \in I)$ $A_i$ and $B_i$ are nonempty closed convex subsets of $Y$ such that $A := \bigcap_{i \in I} A_i \neq \emptyset$ and $B := \bigcap_{i \in I} B_i \neq \emptyset$.

(We assume here without loss of generality that there are as many sets $A_i$ as $B_j$; otherwise, we can either “copy” sets or use the full space $Y$ itself.) It will occasionally be convenient to work with the convention $A_{m+1} = A_1$, $A_{m+2} = A_2$, etc.; or, more formally, $A_n = A_{1 + \text{rem}(n-1,m)}$ and $B_n = B_{1 + \text{rem}(n-1,m)}$. We also assume that the projection operators $P_{A_i}$ and $P_{B_i}$ are “easy” to compute while the projections $P_A$ and $P_B$ are “hard” and not readily available (unless $m = 1$). The problem we are interested in is to find a best approximation pair, i.e., to find $(\bar{a}, \bar{b}) \in A \times B$ such that $\|\bar{a} - \bar{b}\| = \inf_{(a, b) \in A \times B} \|a - b\|$.

(Note that this problem is actually a generalization of the famous convex feasibility problem which asks to find a point in $A \cap B$ provided that this intersection is nonempty which we do not assume here!) This problem has a long history, and the first systematic study was given by Cheney and Goldstein in 1959 [12]; see also, e.g., [3], [4], and [6]. These works, however, assume that the projection operators $P_A$ and $P_B$ are explicitly available, which essentially means that $m = 1$. Recently, Aharoni, Censor, and Jiang (see [1]) tackled the general case. Indeed, assuming that the sets $A_i$ and $B_i$ are halfspaces, they presented a new algorithm — which we call ACJ for simplicity — for solving (1) where they do not require knowledge of the projectors $P_A$ and $P_B$ onto the corresponding polyhedra $A$ and $B$.

The purpose of this paper is to provide other approaches to solving (1). We also provide the required proximity operators as well as numerical comparisons. The algorithms considered will rely only on the operators $P_{A_i}$ and $P_{B_i}$ and some other operators that are available in closed form.

The algorithms presented will work for general closed convex sets, not just polyhedra as long as the projection operators onto the individual sets making up the intersections are available. Implementable formulae for the underlying algorithmic operators are provided. Numerical experiments, similar to one in Aharoni et al.’s paper, are also performed. Our results show that other algorithms should be seriously considered for solving (1), especially if $m$ is small.
The remainder of the paper is organized as follows. In Section 2, we consider reformulations of (1) that are more amenable to the algorithms discussed in Section 3. These algorithms rely on computable formulae which we present in Section 4. We present a small example on which these algorithms are run and convergence is observed using different metrics in Section 5. In Section 5.1 and Section 5.2, we consider examples where the solutions are known or unknown, respectively. We also discuss the (positive) effect of pairing up constraints (see Section 5.3). We conclude the paper in Section 6 with a brief summary of our findings.

The notation employed in this paper is fairly standard and follows largely [5] to which we also refer the reader for general background material. For the reader’s convenience, let us review some notions from convex analysis that are of fundamental importance to this paper. The indicator function of a set \( C \) is written as \( \iota_C \); we have \( \iota_C(x) = 0 \) if \( x \in C \) and \( \iota_C(x) = +\infty \) otherwise. The corresponding distance function is \( d_C(x) = \inf_{c \in C} \| x - c \| \). If \( C \) is convex, closed, and nonempty, then for every \( x \), there exists a unique point \( P_C(x) \in C \) such that \( d_C(x) = \| x - P_C(x) \| \). The corresponding operator \( P_C \) is called the projection operator or simply projector of \( C \). For instance, if \( A \) is the halfspace \( A = \{ x \in Y \mid \langle a, x \rangle \leq \alpha \} \), where \( a \neq 0 \) and \( \alpha \in \mathbb{R} \), then

\[
P_A(x) = x - \frac{\max\{0, \langle a, x \rangle - \alpha\}}{\|a\|^2} a.
\]  

(2)

More generally, if \( f \) is a function that is convex, lower semicontinuous, and proper, then for every \( x \), the function \( y \mapsto f(y) + \frac{1}{2}\| x - y \|^2 \) has a unique minimizer which is the celebrated proximal mapping or prox operator of \( f \), written \( \text{ProxF} \). Note that if \( f = \iota_C \), then we recover the projection operator \( P_C : \text{ProxF}_C = P_C \). A vector \( y \) is a subgradient of \( f \) at \( x \) if for every \( h \), we have \( f(x) + \langle y, h \rangle \leq f(x + h) \); the set of all subgradients at \( x \) is the subdifferential of \( f \) at \( x \), written \( \partial f(x) \).

### 2 Modeling (1)

In the product Hilbert space

\[
X := Y \times Y,
\]

we define nonempty closed convex sets by

\[
(\forall i \in I) \ C_i := A_i \times B_i,
\]

along with their intersection

\[
(\forall i \in I) \ C := \bigcap_{i \in I} C_i = A \times B.
\]
Note that the projector onto $C_i$ is still easy to compute; indeed, $P_{C_i} : (x, y) \mapsto (P_{A_i} x, P_{B_i} y)$. The problem \( (1) \) is thus equivalent to
\[
\text{minimize } h(x, y) \text{ subject to } (x, y) \in C,
\]
where
\[
h(x, y) = \alpha \|x - y\|^p,
\]
\(\alpha > 0\), and \(p \geq 1\). Note that \(h\) has full domain and it is convex — but \(h\) is not strictly convex because it has many minimizers: \(\{(x, x) \mid x \in Y\}\). Also note that if \(p > 1\), then \(h\) is differentiable.

The problem \( (1) \) can thus also be alternatively thought of as
\[
\text{minimize } h(x, y) + \sum_{i \in I} \iota_{C_i}(x, y),
\]
which features a nonsmooth objective function. In the next section, we survey various algorithms that could be used to solve the problem \( (1) \) or its reformulations. We also consider the case when \( (5) \) is approximated by
\[
\text{minimize } h(x, y) + \sum_{i \in I} L d_{C_i}(x, y),
\]
for some "large" constant \(L\).

3 Algorithms for solving \( (1) \)

In this section, we discuss various algorithms. The algorithms in Section 3.1 and Section 3.2 are able to solve the original problem exactly while those in the remaining subsections can be viewed as attempting to solve a perturbed problem.

3.1 The Aharoni–Censor–Jiang Algorithm

This algorithm was recently proposed by Aharoni, Censor, and Jiang in [1]. We denote their algorithm as ACJ. ACJ builds on the earlier HLWB algorithm. (The letters in HLWB signify relevant works by Halpern [15], by Lions [20], by Wittmann [23], and by Bauschke [2]; the name HLWB was coined by Censor in [11] )\ ACJ can be viewed as an alternating version of HLWB to find a solution of \( (1) \). Here is the description of ACJ. First, we fix a sequence \((\lambda_k)_{k \in \mathbb{N}}\) of positive real numbers such that
\[
\lambda_k \to 0, \quad \sum_{k \in \mathbb{N}} \lambda_k = \infty, \quad \sum_{k \in \mathbb{N}} |\lambda_k - \lambda_{k+m}| < \infty
\]
and also an increasing (not necessarily strictly though) sequence of natural numbers $(n_k)_{k \in \mathbb{N}}$ such that

$$n_k \to \infty \quad \text{and} \quad \sup_{k_0 \in \mathbb{N}} \sum_{k > k_0} \prod_{n > n_k} (1 - \lambda_n) < \infty. \quad (8)$$

For instance, (7)–(8) hold when $(\forall k \in \mathbb{N}) \lambda_k = \frac{1}{k + 1}$ and $n_k = \lfloor 1.1^k \rfloor$ (see [1, page 512]).

Next, given our sequence of sets $(A_i)_{i \in \mathbb{N}}$ and $n \in \mathbb{N}$, we define the operator

$$Q_{A,n} : Y \times Y \to Y : (w,w') \mapsto w_n, \quad (9)$$

where $w_0 = w'$ and $w_n$ is computed iteratively via

$$(\forall i \in \{0,1,\ldots,n-1\}) \quad w_{i+1} = \lambda_{i+1} w + (1 - \lambda_{i+1}) P_{A_{i+1}}(w_i). \quad (10)$$

The operator $Q_{B,n}$, for $(B_i)_{i \in \mathbb{N}}$ and $n \in \mathbb{N}$, is defined analogously. Finally, we initialize $(x_0, y_0) \in X \times X$, and iteratively update via

$$(\forall k \in \mathbb{N}) \quad (x_{k+1}, y_{k+1}) := \begin{cases} (Q_{B,n_k}(y_k, y'_k), y_k), & \text{if } k \text{ is odd;} \\ (x_k, Q_{A,n_k}(x_k, x'_k)), & \text{if } k \text{ is even}, \end{cases} \quad (11)$$

and where $(x'_k, y'_k)_{k \in \mathbb{N}}$ in $X \times X$ is a bounded sequence that can either be fixed beforehand, e.g., $(x'_k, y'_k) = (y_0, x_0)$, or dynamically updated using, e.g., $(x'_0, y'_0) = (y_0, x_0)$ and $(x'_k, y'_k) = (y_{k-1}, x_{k-1})$ for $k \in \{1, 2, \ldots\}$.

The main result of [1] yields the convergence of $(x_k, y_k)_{k \in \mathbb{N}}$ to a solution of (1) provided that $C \neq \emptyset$ and each $A_i$ and $B_i$ is a halfspace. We refer the reader to (2) for the formula for the projection onto a halfspace which is required by ACJ.

**Remark 3.1.** Note that ACJ takes into account the order of the sets while the problem (1) does not. It is a nice feature of ACJ that it works throughout in the “small” space $X \times X$. On the other hand, we are not aware of any extension of ACJ to the case when the sets underlying the intersections are not halfspaces. This is an interesting topic for further research.

### 3.2 Douglas–Rachford Algorithm

This algorithm, abbreviated as DR, can be traced back to the paper by Douglas and Rachford [14] although its relevance to optimization was brought to light later in the seminal paper by Lions and Mercier [21]. DR can deal with problems of the form (5), and it implicitly operates in the space $X^{m+1}$. First, set $f_0(x, y) := \alpha \|x - y\|^p$ with $\alpha > 0$
and $p \geq 1$, as well as $f_1 := \iota_{C_1}, \ldots, f_m := \iota_{C_m}$ and $I_0 := \{0\} \cup I$. Second, fix a parameter $0 < \lambda < 2$ (the default being $\lambda = 1$).

Now initialize $z_0 := (z_{0,0}, z_{0,1}, \ldots, z_{0,m}) = ((x_{0,0}, y_{0,0}), (x_{0,1}, y_{0,1}), \ldots, (x_{0,m}, y_{0,m})) \in X^{m+1}$. Given $z_k = (z_{k,0}, z_{k,1}, \ldots, z_{k,m}) \in X^{m+1}$, set

$$
\bar{z}_k := \frac{1}{m+1} \sum_{i \in I_0} z_{k,i} \quad (12a)
$$

$$(\forall i \in I_0) \ x_{k,i} := \text{Prox}_{f_i}(2\bar{z}_k - z_{k,i}) \quad (12b)
$$

$$(\forall i \in I_0) \ z_{k+1,i} := z_{k,i} + \lambda (x_{k,i} - \bar{z}_k) \quad (12c)
$$

to obtain the update $z_{k+1} := (z_{k+1,0}, z_{k+1,1}, \ldots, z_{k+1,m})$.

If $i \in I$, then the prox operators corresponding to $f_i$ is simply the projector $P_{C_i}$. In particular, if each $C_i = A_i \times B_i$ is the Cartesian product of two halfspaces, then we may utilize (2) twice to compute $P_{C_i} = \text{Prox}_{f_i}$. The prox operator $\text{Prox}_{f_0}$ will be computed in closed form for $p \in \{1, 2\}$ in Section 4.1 below. When $p = 1$, which produced better numerical results, then $f_0(x, y) = \alpha \|x - y\|$ and

$$
\text{Prox}_{f_0}(x, y) = (x, y) - \frac{1}{\max \{2, \|x - y\|/\alpha\}}(x - y, y - x). \quad (13)
$$

It is well known (see, e.g., [5, Proposition 28.7]) that the sequence $(\bar{z}_k)_{k \in \mathbb{N}}$ will converge to a solution of (5), i.e., of (1).

**Remark 3.2.** The DR approach does not care about the order of the sets presented — unlike, ACJ! A downside is that it operates in the larger space $X^{m+1}$ which can become an issue if $m$ is large. On the positive side, if $C_1 \cap \cdots \cap C_m = \emptyset$, then $(\bar{z}_k)_{k \in \mathbb{N}}$ will converge to a minimizer of $f_0$ over the set of least-squares solutions (see [7, Corollary 6.8] for further information). Finally, it does not require the constraint sets $C_i$ to be Cartesian product of halfspaces.

### 3.3 Dual-Based Proximal Method

We largely follow Beck’s [8, Section 12.4.2] (see also [9] and [13] for further background material) but slightly modify the algorithms presented there to give two additional methods for solving (1). We will work with the form given in (5) where $h(x, y)$ needs to be $\varepsilon$-strongly convex, for some $\varepsilon > 0$, which precludes using $\alpha \|x - y\|^p$ directly. However, below we will add $\varepsilon \frac{1}{2}(\|x\|^2 + \|y\|^2)$ to this last function to obtain the required $\varepsilon$-strong convexity. We point out that by adding this energy term and solving the corresponding new perturbed optimization problem, the solution obtained does not solve the original problem exactly.
The first method considered is the Dual Proximal Gradient method, which — following [8] — we abbreviated as DPG. Because the algorithm requires strong convexity of the objective function, we consider
\[
f_0(x, y) := \alpha \frac{1}{2} \|x - y\|^2 + \epsilon \frac{1}{2} \left(\|x\|^2 + \|y\|^2\right) \quad \text{with } \alpha > 0 \text{ and } \epsilon > 0.
\] (14)
We also set \( f_1 := \iota_{C_1}, \ldots, f_m := \iota_{C_m} \). Second, fix a parameter \( L \geq m/\epsilon \).

Now initialize \( z_0 := (z_{0,1}, \ldots, z_{0,m}) = ((x_{0,1}, y_{0,1}), \ldots, (x_{0,m}, y_{0,m})) \in X^m \), and update it using
\[
s_k := \sum_{i \in I} z_{k,i} \quad \text{(15a)}
\]
\[
x_k := \arg\max_{w \in X} \left[ \langle w, s_k \rangle - f_0(w) \right] \quad \text{(15b)}
\]
\[
(\forall i \in I) \quad z_{k+1,i} := z_{k,i} - \frac{1}{L} x_k + \frac{1}{L} P_{f_i}(x_k - L z_{k,i}) \quad \text{(15c)}
\]
to obtain \( z_{k+1} := (z_{k+1,1}, \ldots, z_{k+1,m}) \). This is the primal representation of DPG, see [8, page 356], which is most convenient in our setting. Once again, the prox operators corresponding to \( f_i \) for \( i \in I \) are just the projectors \( P_{C_i} \). If the sets \( C_i \) are Cartesian products of halfspaces, we may use (2) to compute \( P_{C_i} \). The closed form for the argmax operator in (15b) is given by
\[
x_k = \frac{1}{(2\alpha + \epsilon)\epsilon} \left( (\alpha + \epsilon)u_k + \alpha v_k, (\alpha + \epsilon)v_k + \alpha u_k \right), \quad \text{where } s_k = (u_k, v_k).
\] (16)
This formula will be proved in Section 4.2 below. For sufficiently small \( \epsilon > 0 \), the primal sequence \( (x_k)_{k \in \mathbb{N}} \) approximates a solution of (5) and hence of (1) provided that the relative interiors of the sets \( C_i \) form a nonempty intersection (see [8, page 362]). Note that we do not expect that the primal sequence converges to an exact solution of (1) because the objective function \( f_0 \) in (14) is not identical to the one required to tackle (1).

An accelerated version of DPG, known as Fast Dual Proximal Gradient or simply FDPG, applies a FISTA-type acceleration (see [8, Section 12.3] and [9].) Here is how FDPG proceeds: Starting with \( z_0 \) as before, initialize \( w_0 := z_0, t_0 := 1 \), and update via
\[
s_{k} := \sum_{i \in I} w_{k,i} \quad \text{(17a)}
\]
\[
u_{k} := \arg\max_{v \in X} \left[ \langle v, s_{k} \rangle - f_{0}(v) \right] \quad \text{(17b)}
\]
\[
(\forall i \in I) \quad z_{k+1,i} := w_{k,i} - \frac{1}{L} u_k + \frac{1}{L} P_{f_i}(u_k - L w_{k,i}) \quad \text{(17c)}
\]
\[
t_{k+1} := \frac{1 + \sqrt{1 + 4t_{k}^2}}{2} \quad \text{(17d)}
\]
\[(\forall i \in I) \quad w_{k+1,i} := z_{k,i+1} + \left(\frac{t_k - 1}{t_{k+1}}\right) (z_{k,i+1} - z_k) \]  \hspace{1cm} (17e)

to get the primal sequence of interest

\[x_{k+1} := \arg\max_{v \in X} \left[ \langle v, s_{k+1} \rangle - f_0(v) \right], \quad \text{where} \quad s_{k+1} := \sum_{i \in I} z_{k+1,i}. \]  \hspace{1cm} (18)

Again, for sufficiently small \(\varepsilon > 0\), the sequence \((x_k)_{k \in \mathbb{N}}\) approximates a solution “close” to that of (5) — but not exactly — and (as a consequence of (1)) provided that the relative interior of \(C\) is not empty.

**Remark 3.3.** Note that although a smaller \(\varepsilon\) ensures a solution that is closer to that of the original problem (1), it also increases the lower bound for \(L\), which in turn reduces the step size for each iteration as seen in (15c) and (17c), and so the speed of convergence reduces as well. These algorithms are not affected by the order of the sets presented, but like DR, they operate in a “large” space (here \(X^m\)). This may become a problem when \(m\) is large.

### 3.4 Proximal Distance Algorithm

The Proximal Distance Algorithm, or PDA for short, was first introduced by Lange and Keys [19]. It is motivated by the framework of MM algorithms, where MM stands for majorize/minimize or for minorize/maximize depending on the underlying problem. This framework was pioneered by Lange; see, e.g., his book [18] on this topic. It can be interpreted as a prox-gradient method applied to the function \(\frac{1}{\rho} h + \frac{1}{m} \sum_{i \in I} \frac{1}{2} d_{C_i}^2\), where the penalty parameter is in theory driven to \(+\infty\) (see [18, Section 5.5] for a gentle introduction). The parameter \(\rho\) has to be carefully driven to infinity. We will apply PDA to the problem formulation given by (5). Set \(h(x, y) := \alpha \|x - y\|\), which is \(\sqrt{2} \alpha\)-Lipschitz by Proposition 4.2, for \(\alpha > 0\). Also, write \(z = (x, y)\). The PDA with starting point \(z_0 \in X\) generates a sequence \((z_k)_{k \in \mathbb{N}}\) via

\[z_{k+1} := \text{Prox}_{\rho_k^{-1} h} \left( \sum_{i=1}^m \frac{1}{m} P_{C_i} z_k \right), \]  \hspace{1cm} (19)

where

\[\text{Prox}_{\rho_k^{-1} h}(x, y) = (x, y) - \frac{1}{\max \{2, \rho_k \|x - y\|/\alpha\}} (x - y, y - x) \]  \hspace{1cm} (20)

by (36) and where \((\rho_k)_{k \in \mathbb{N}}\) is a sequence of positive (and “sufficiently large”) parameters. If the sets \(C_i\) are Cartesian products of halfspaces, we may use (2) to compute \(P_{C_i}\).
Under suitable choices of the parameter sequences, the sequence $(z_k)_{k \in \mathbb{N}}$ approximates a solution of (5). Lange and Keys recommend $\rho_k = \min\{(1.2)^k \rho_0, \rho_{\max}\}$ but other choices may yield better performance (see [19, Sections 4 and 5] and [16] for details). Keys, Zhou, and Lange also point out a Nesterov-style accelerated version of PDA (accPDA for short), which proceeds as follows:

$$w_k := z_k + \frac{k-1}{k+2}(z_k - z_{k-1}),$$

(21a)

$$z_{k+1} := \text{Prox}_{\rho_k^{-1}} \left( \frac{1}{m} \sum_{i=1}^{m} P_{C_i} w_k \right).$$

(21b)

See [16, Algorithm 1 and Section 3] for further information. Note that because $\rho_k \leq \rho_{\max} < +\infty$, both PDA and accPDA find a solution of the penalized but not of the original problem.

### 3.5 Stochastic Subgradient Descent

The roots of stochastic gradient descent can be traced back to two key papers from the early 1950s co-authored by Robbins and Monro [22] and by Kiefer and Wolfowitz [17]; see also [10] for a recent survey. The method has since been generalized to many different settings. We follow largely the presentation in [8]. Set $f_0(x, y) := \alpha \|x - y\|$ and $(\forall i \in I)$ $f_i := Ld_{C_i}$, where $L > 0$. Then $f_0$ is $\sqrt{2}\alpha$-Lipschitz and (by (35) below)

$$f_0'(z) = f_0'(x, y) = \alpha (\text{sign}(x - y), -\text{sign}(x - y)),$$

(22)

where “sign” is defined in (25). The other functions $f_i$ are $L$-Lipschitz. Moreover, for $i \in I$, we have

$$f_i'(z) = L \text{sign}(z - P_{C_i} z) \in \partial f_i(z)$$

(23)

by, e.g., [5, Example 16.62]. If the sets $C_i$ are Cartesian products of halfspaces, we may use (2) to compute $P_{C_i}$. Now Stochastic Subgradient Descent, which we abbreviate as SSD (see [8, Section 8.3] for further information), applied to (6) generates a sequence via

$$z_{k+1} := z_k - \eta_k f_i'(z_k),$$

(24)

where $(\eta_k)_{k \in \mathbb{N}}$ is a sequence of positive parameters (typically constant or a constant divided by $\sqrt{k+1}$) and where $f_i'(z_k) \in \partial f_i(z_k)$ where $i_k$ is uniformly sampled from $I_0 := \{0\} \cup I$.

Under appropriate conditions, the sequence generated by SSD approximates a minimizer of the function $\alpha \|x - y\| + L \sum_{i \in I} d_{C_i}$. Note that for large $L$, the distance functions converge pointwise to the corresponding indicator functions, but they are different for fixed $L$. In this sense, SSD finds a perturbed but not exact solution of the original problem.
4 Useful operators

In this section, we collect formulae for operators that are used later in our numerical experiments.

4.1 Prox and (sub)differential operators

Denote the standard unit ball by $B := \{ y \in Y \mid \|y\| \leq 1 \}$. It will be convenient to define the generalized signum functions on $Y$ via

$$\text{sign}(x) := \begin{cases} x/\|x\|, & \text{if } x \neq 0; \\ 0, & \text{if } x = 0 \end{cases} \quad \text{and} \quad \text{Sign}(x) := \begin{cases} \{x/\|x\|\}, & \text{if } x \neq 0; \\ B, & \text{if } x = 0. \end{cases} \quad (25)$$

By [5, Example 16.32], we have

$$\forall x \in Y \quad \text{sign}(x) \in \text{Sign}(x) = \partial \| \cdot \| (x). \quad (26)$$

**Proposition 4.1.** Let $\alpha > 0$. Then the prox operator of the function

$$h: Y \times Y \to \mathbb{R}: (x, y) \mapsto \alpha \frac{1}{2}\|x - y\|^2. \quad (27)$$

is given by

$$\text{Prox}_h: (x, y) \mapsto \frac{1}{2\alpha + 1}((1 + \alpha)x + ay, ax + (1 + \alpha)y). \quad (28)$$

Moreover, $\nabla h: (x, y) \mapsto (a(x - y), a(y - x))$ is $(1 + 2\alpha)$-Lipschitz continuous.

**Proof.** Set $z = (x, y) \in Y \times Y$ and $B: Y \times Y \to Y: (x, y) \mapsto \sqrt{\alpha}(x - y)$. Then

$$h(z) = \frac{1}{2}\|Bz\|^2 = \frac{1}{2} \langle Bz, Bz \rangle = \frac{1}{2} \langle B^*Bz, z \rangle \quad (29)$$

and thus $\nabla h = B^*B$. It follows that

$$\text{Prox}_h = (\text{Id} + B^*B)^{-1}. \quad (30)$$

Write $B$ in block matrix form, $B = \sqrt{\alpha} [\text{Id}, -\text{Id}]$. Then $B^* = \sqrt{\alpha} [\text{Id}, -\text{Id}]^\top$,

$$\nabla h = B^*B = \alpha \begin{bmatrix} \text{Id} & -\text{Id} \\ -\text{Id} & \text{Id} \end{bmatrix}, \quad (31)$$

and

$$\text{Id} + \nabla h = \text{Id} + B^*B = \begin{bmatrix} (1 + \alpha)\text{Id} & -\alpha\text{Id} \\ -\alpha\text{Id} & (1 + \alpha)\text{Id} \end{bmatrix}. \quad (32)$$
The largest eigenvalue of the very last matrix is $1 + 2\alpha$ which implies that $1 + 2\alpha$ is the sharp Lipschitz constant of $\nabla h$. Finally,

$$\text{Prox}_h = (\text{Id} + \nabla h)^{-1} = \left( \text{Id} + B^* B \right)^{-1} = \frac{1}{2\alpha + 1} \begin{bmatrix} (1 + \alpha) \text{Id} & \alpha \text{Id} \\ \alpha \text{Id} & (1 + \alpha) \text{Id} \end{bmatrix}$$ \hspace{1cm} (33)

and the result follows.

\[\square\]

**Proposition 4.2.** Let $\alpha > 0$. The function

$$h: Y \times Y \to \mathbb{R}: (x, y) \mapsto \alpha \|x - y\|,$$  

is $\alpha \sqrt{2}$-Lipschitz and a convenient selection of $\partial h$ is given by

$$(x, y) \mapsto \alpha \left( \frac{\text{sign}(x - y)}{\|x - y\|}, \frac{\alpha(y - x)}{\|y - x\|} \right), \quad \text{if } x \neq y; \hspace{1cm} (35a)$$

$$= \begin{cases} (\alpha(x - y) / \|x - y\|, \alpha(y - x) / \|y - x\|), & \text{if } x \neq y; \\ (0, 0), & \text{if } x = y \end{cases} \hspace{1cm} (35b)$$

$$\in \{ \alpha(s, -s) \mid s \in \text{Sign}(x - y) \}$$

$$= \partial h(x, y). \hspace{1cm} (35c)$$

The prox operator of $h$ is given by

$$\text{Prox}_h: (x, y) \mapsto (x, y) - \frac{1}{\max\{2, \|x - y\|/\alpha\}}(x - y, y - x). \hspace{1cm} (36)$$

**Proof.** Set $A = [\text{Id}, -\text{Id}], z = (x, y)$, and $f(z) = \|Az\|$. Then $Az = x - y$ and $h = \alpha f$. Furthermore, $\|Az\|^2 = \|x - y\|^2 = \|x\|^2 + \|y\|^2 - 2\langle x, y \rangle \leq 2\|x\|^2 + 2\|y\|^2 = 2(\|x\|^2 + \|y\|^2) = 2\|z\|^2$ which shows that $f$ is $\sqrt{2}$-Lipschitz and therefore $h$ is $\alpha \sqrt{2}$-Lipschitz. The subdifferential formula follows using [5, Proposition 16.6(i) and Example 16.32]. Note that, for $\beta \geq 0$,

$$A^T = \begin{bmatrix} \text{Id} \\ -\text{Id} \end{bmatrix}, \quad AA^T = 2\text{Id}, \quad (AA^T + \beta \text{Id})^{-1} = \frac{1}{2 + \beta} \text{Id}$$ \hspace{1cm} (37)

and

$$\text{Id} - A^T (AA^T + \beta \text{Id})^{-1} A = \begin{bmatrix} \text{Id} & 0 \\ 0 & \text{Id} \end{bmatrix} - \begin{bmatrix} \text{Id} \\ -\text{Id} \end{bmatrix} \frac{1}{2 + \beta} \begin{bmatrix} \text{Id} & -\text{Id} \\ -\text{Id} & \text{Id} \end{bmatrix}$$ \hspace{1cm} (38a)

$$= \begin{bmatrix} \text{Id} & 0 \\ 0 & \text{Id} \end{bmatrix} - \frac{1}{2 + \beta} \begin{bmatrix} \text{Id} & -\text{Id} \\ -\text{Id} & \text{Id} \end{bmatrix}$$ \hspace{1cm} (38b)

$$= \frac{1}{2 + \beta} \begin{bmatrix} 1 + \beta & 1 \\ 1 & 1 + \beta \end{bmatrix}. \hspace{1cm} (38c)$$
We now discuss cases.

Case 1: \(|x - y| \leq 2\alpha\).

In view of (37) (with \(\beta = 0\)), this is equivalent to \(\|(AA^\top)^{-1}Az\| \leq \alpha\). Using [8, Lemma 6.68] and (38) (with \(\beta = 0\)), we obtain (switching back to row vector notation for convenience)

\[
\text{Prox}_h(x, y) = \frac{1}{2}(x + y, x + y) = (x, y) - \frac{1}{2}(x - y, y - x).
\]  

(39)

Case 2: \(|x - y| > 2\alpha\).

In view of (37) (with \(\beta = 0\)), this is equivalent to \(\|(AA^\top)^{-1}Az\| > \alpha\). Using again (37) (with general \(\beta \geq 0\)), we set and obtain

\[
g(\beta) = \|(AA^\top + \beta \text{Id})^{-1}Az\|^2 - \alpha^2
\]

\[= \frac{1}{(2 + \beta)^2}\|x - y\|^2 - \alpha^2.\]  

(40a)

(40b)

Because \(\alpha > 0\) and \(\beta \geq 0\), we obtain the equivalences: \(g(\beta) = 0 \iff \|x - y\| = \alpha(2 + \beta) \iff \beta = \|x - y\|/\alpha - 2\). Set

\[
\beta^*: = \frac{\|x - y\|}{\alpha} - 2 > 0
\]

(41)

so that \(g(\beta^*) = 0\). Moreover, \(2 + \beta^* = \|x - y\|/\alpha\) and \(1 + \beta^* = (\|x - y\| - \alpha)/\alpha\). Hence (38) (with \(\beta = \beta^*\)) turns into

\[
\text{Id} - A^\top(AA^\top + \beta^* \text{Id})^{-1}A = \frac{1}{2 + \beta^*} \begin{bmatrix} 1 + \beta^* & 1 \\ 1 & 1 + \beta^* \end{bmatrix}
\]

\[= \frac{\alpha}{\|x - y\|} \begin{bmatrix} \|x - y| - \alpha \\ \alpha \end{bmatrix} \begin{bmatrix} 1 \\ \|x - y\| - \alpha \end{bmatrix}
\]

\[= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \frac{\alpha}{\|x - y\|} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.
\]

(42a)

(42b)

(42c)

Using [8, Lemma 6.68] and (42), we obtain (switching back to row vector notation for convenience)

\[
\text{Prox}_h(x, y) = (x, y) - \frac{\alpha}{\|x - y\|}(x - y, y - x).
\]  

(43)

Finally, the formula given in (36) follows by combining (39) and (43).

\[\blacksquare\]

4.2 Argmax operator

Consider, for \(\alpha > 0\) and \(\varepsilon > 0\),

\[
f_0(x, y) := \alpha \frac{1}{2}\|x - y\|^2 + \varepsilon \frac{1}{2}(\|x\|^2 + \|y\|^2),
\]

(44)
which is a perturbation of $\alpha \frac{1}{2} \|x - y\|^2$ that is $\epsilon$-strongly convex.

Given $(u, v) \in X$, the dual-based proximal methods of Section 3.3 require from us to find the unique maximizer of

\[
(x, y) \mapsto \langle (u, v), (x, y) \rangle - f_0(x, y)
\]

\[
= \langle u, x \rangle + \langle v, y \rangle - \alpha \frac{1}{2} \|x - y\|^2 - \epsilon \frac{1}{2} (\|x\|^2 + \|y\|^2).
\]

We now derive an explicit formula for this maximizer:

**Proposition 4.3.** Given $\alpha > 0$, $\epsilon > 0$, and $(u, v) \in X$, the unique maximizer of (45) is

\[
(x, y) = \frac{1}{(2\alpha + \epsilon)e} \left( (\alpha + \epsilon)u + \alpha v, (\alpha + \epsilon)v + \alpha u \right).
\]

**Proof.** Because $f_0$ is strongly convex, we employ standard convex calculus to find the maximizer by finding the zero of the gradient of the function in (45). That is, we need to solve

\[
(0, 0) = (u - \alpha x + \alpha y - \epsilon x, v - \alpha y + \alpha x - \epsilon y);
\]

or equivalently (switching to more formal column vector notation),

\[
\begin{bmatrix}
\alpha + \epsilon & -\alpha \\
-\alpha & \alpha + \epsilon
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
= \begin{bmatrix}
u \\
v
\end{bmatrix}.
\]

Because

\[
\begin{bmatrix}
\alpha + \epsilon & -\alpha \\
-\alpha & \alpha + \epsilon
\end{bmatrix}^{-1} = \frac{1}{(2\alpha + \epsilon)e} \begin{bmatrix}
\alpha + \epsilon & \alpha \\
\alpha & \alpha + \epsilon
\end{bmatrix}
\]

we obtain the announced formula.

\[\Box\]

## 5 Numerical experiments

We start by discussing metrics — in the sense of “standard of measurement” not in the sense of topology — to evaluate the quality of the the iterates for the different algorithms proposed in Section 3. To make the measurement uniform over the different algorithms, we use this metric once for each prox or proj evaluation. For the same purpose, we consider $A_i$ or $B_i$ as unit inputs in these operators. So, for example, as DR projects to all $A_i$ and $B_i$ along with a prox evaluation, we get a total of $2m + 1$ operations. Since it is not possible to obtain the final output of a given iteration before computing all the proxes, we
repeat the final output given by the iteration $2m + 1$ times. Again for the sake of uniformity, we only repeat the final output of each iteration, regardless of whether intermediate updates can be calculated. We shall consider two cases: in the first, true solutions are known (and assumed to be unique) while in the second, they aren’t. The former case allows us to inspect the progress of the iterates towards the solution, while in the latter case a metric is needed to gauge the performance of the algorithms. The latter scenario is the one most realistic for applications.

5.1 Two examples

The convergence plots for the algorithms are straightforward when the solution is known. Assuming the solution is unique and denoted by $(\bar{x}, \bar{y})$, we use the metric

$$ (x, y) \mapsto \| (x, y) - (\bar{x}, \bar{y}) \| $$

(50)
applied to the appropriate iterates of the algorithms.

The distance to the solution \((50)\) is evaluated once for each projection or prox operator evaluation. For example, because each iteration of \(\text{DR}\) (see Section 3.2) uses \(2m + 1\) prox evaluation (see the first paragraph of Section 5), one \(\text{DR}\) step invokes \(2m + 1\) “updates”. This approach ensures that the evaluation has some uniformity/fairness over the different algorithms.

We begin with an example where we know the solution. This (small-scale) example is motivated by the one provided by Aharoni et al. in [1, Section 5].

**Example 5.1.** Consider the subsets \(A\) and \(B\) of \(Y = \mathbb{R}^2\), defined by the two systems of \(m = 4\) linear inequalities

\[
\begin{bmatrix}
4 & 3 & -17 \\
1 & 0 & 4 \\
1 & 1 & 11 \\
0 & 1 & 5
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
1
\end{bmatrix}
\leq
\begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
5 & -4 & -30 \\
1 & -2 & 0 \\
-1 & -4 & 24 \\
-2 & -1 & 13
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
1
\end{bmatrix}
\leq
\begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix}, \quad (51)
\]
respectively. The corresponding problem (1) possesses the unique solution
\[(−6, −5), (4, 5)\],
which is also visualized in Fig. 1.

We ran the algorithms from Section 3, and also accelerated versions when available. The algorithms were run for a total 1800 Prox or Projection operations per algorithm. The accelerated versions performed clearly better than the original versions in this case. Therefore, for the clarity of the exposition, we do not report the DPG and PDA results. We used the following parameters and also report to how many iterations in each algorithm this corresponded:

**ACJ**: 55 iterations, with \(λ_k = 1/(k + 1), n_k = [1.1^k]\), and
\[(x'_k, y'_k) = \begin{cases} (y_0, x_0), & \text{if } k = 0; \\ (y_{k-1}, x_{k-1}), & \text{otherwise} \end{cases} \text{(see Section 3.1).}\]

**DR**: 200 iterations, with \(p = 1, α = 5, \text{and } λ = 1\) (see Section 3.2).

**FDPG**: 200 iterations, with \(α = 1, L = 16, \text{and } ε = 1/4\) (see Section 3.3).

**accPDA**: 200 iterations, with \(α = 1, ρ_0 = 1 \text{ and } ρ_{\text{max}} = 100000\) (see Section 3.4).

**SSD**: 1800 iterations, with \(α = 1, L = 1, \text{and } η_k = 1/√{k+1}\) (see Section 3.5).

In each case, we use the starting point \(x_0 = y_0 = (8, −13)\). The distance of the iterates from the solution, calculated using (50), results in the convergence plot shown in Fig. 2, where the grey-dotted lines marks intervals of 50 iterations of DR. From the plot, we see that accPDA appears to perform better than DR for the first 50 iterations, but then DR slowly but steadily begins to produce the most accurate solution. We note that since FDPG is solving for a strong convex version of the objective function \(∥x − y∥\), it converges to a solution that is not the same as our original problem.

Note that ACJ and SSD do not perform as well as the other algorithms in Example 5.1; however, when \(m\) becomes larger, they become much more competitive as the following example illustrates:

**Example 5.2.** Let \(n ∈ \{1, 2, \ldots\}\), and consider the subsets \(A\) and \(B\) of \(Y = \mathbb{R}^n\), defined by the two system of \(m = n\) linear inequalities
\[
\begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n
\end{bmatrix} \geq \begin{bmatrix}
  5 \\
  \vdots \\
  5
\end{bmatrix} \quad \text{and} \quad \begin{bmatrix}
  x_1 \\
  \vdots \\
  x_n
\end{bmatrix} \leq \begin{bmatrix}
  -5 \\
  \vdots \\
  -5
\end{bmatrix},
\]
respectively. Clearly, the unique solution to the corresponding problem (1) is \((\bar{x}, \bar{y}) ∈ Y × Y\), where \(\bar{x} = (5, \ldots, 5)\) and \(\bar{y} = (-5, \ldots, -5)\). This time we run the algorithms for around \(10^8\) prox evaluations with the starting point \(x_0 = y_0 = (0, \ldots, 0)\). It turns out that
Figure 3: Example 5.2 for which $n = m = 1000$

In this case, PDA outperforms accPDA, so we omit the results of the latter. We only use the error metric once after every 50000 evaluations. We used the following parameters and also report to how many iterations in each algorithm this corresponded:

**ACJ:** 169 iterations, with $\lambda_k = 1/(k+1)$, $n_k = \lceil 1.1^k \rceil$, and

$$
(x'_k, y'_k) = \begin{cases} 
(y_0, x_0), & \text{if } k = 0; \\
(y_{k-1}, x_{k-1}), & \text{otherwise}
\end{cases}
$$

(see Section 3.1).

**DR:** 50000 iterations, with $p = 1$, $\alpha = 5$, and $\lambda = 1$ (see Section 3.2).

**FDPG:** 50000 iterations, with $\alpha = 1$, $L = 10000$, and $\varepsilon = 1/10$ (see Section 3.3).

**PDA:** 50000 iterations, with $\alpha = 1$, $\rho_0 = 1$ and $\rho_{max} = 10^5$ (see Section 3.4).

**SSD:** $\approx 10^8$ iterations, with $\alpha = 1$, $L = 10$, and $\eta_k = 1/\sqrt{k+1}$ (see Section 3.5).

For $n = 1000 = m$ and the error metric given by (50), we obtain the plot shown in Fig. 3. Note that ACJ and SSD operate in $X = Y \times Y = \mathbb{R}^{2000}$ while, for instance, DR operates in the much bigger space $X^{m+1} = \mathbb{R}^{2001000}$! The plot makes it clear that in this situation ACJ and SSD fare much better than in Example 5.1.
We note that given enough iterations, DR again trumps the other algorithms, but the initial descent is very slow. In fact, the other algorithms perform much better in the beginning than DR. This suggests an interesting topic for further research: one could consider a hybrid approach, where one uses an algorithm such as SSD, ACJ, PDA or FDPG, and then switches over to DR. Note that only ACJ and DR are known to converge to a solution of the original — the algorithms SSD, PDA, and FDPG solve perturbed versions and thus can play a role in quickly getting “close” to nearby points in a preprocessing capacity.

5.2 What to do in the absence of known solutions

In general, one has no access to true solutions, so it becomes necessary to measure performance by a metric different from (50). We propose the measure

\[ D_\delta(x, y) := \|x - y\| + \frac{1}{\delta} \sum_{i \in I} d_{C_i}(x, y), \]

Figure 4: Using (54) to measure performance of the algorithms
where $\delta > 0$. Because the problem asks to find a point in $A \times B$, feasibility is of greater importance than minimizing $\|x - y\|$; thus, a smaller value of $\delta$ is desirable to stress feasibility. (One could also envision a “dynamic” metric, where $\delta \to 0$ as the number of iterations increases, but we have not tested this.) Revisiting the problem considered in Example 5.1, we show in Fig. 4 the convergence plot using the parameter $\delta = 1/10$. This time, the horizontal axis was taken with a log scaling to increase readability of the resulting graph. The behaviour of the algorithms in the plot on the right, which includes the feasibility conditions along with the distance between $x$ and $y$, resembles the one seen in Fig. 2.

5.3 Combining constraints

![Figure 5: Comparing variants of DR on Example 5.1 with paired projections](image)

In some cases, it is possible to combine constraints and still be able to compute the projection onto the intersection. For instance, if all sets $A_i$ are halfspaces, then any two sets $A_i$ and $A_j$ may be combined and the projection onto the intersection is explicitly available using, e.g., [5, Proposition 2.22–2.24].
Revising Example 5.1 in this light, we ran DR with the paired projection (PP), with choosing mostly non-adjacent halfspaces (labelled as DR+PP in Fig. 5), and also with choosing explicitly adjacent halfspaces (labelled as DR+PP adjacent in Fig. 5) along with the original version of DR (labelled as DR in Fig. 5). To compare this fairly, we count one “paired” projection (onto the intersection of two halfspaces) as being equivalent to two regular projections. The convergence plot shown in Fig. 5 illustrates that adding the paired projections significantly improves the performance significantly, even more so when the halfspaces are adjacent. The well known and characteristic “rippling” seen in typical DR curves is heavily damped in the last case.

Figure 6: Comparing ACJ on Example 5.1 with paired projections

Using this technique on ACJ, we note that the paired-projection variants do not improve the performance significantly; see Fig. 6. On the other hand, the approach of the iterates to the true solution looks far less scattered as can be seen in Fig. 7.

In higher dimensions, further investigations are needed to determine the “best” way to pair up halfspaces as “adjacent halfspaces”.

20
6 Conclusion

We revisited the recent study by Aharoni, Censor, and Jiang on finding best approximation pairs of two polyhedra. The framework we proposed works for two sets that are themselves finite intersections of closed convex sets with “simple” projections. Several algorithms were proposed and the required prox operators were computed. Our numerical experiments suggested that other algorithms deserve serious consideration.

Acknowledgments

The authors thank the editor and the reviewers for helpful suggestions and constructive feedback which helped us to improve the presentation of the results, and Patrick Combettes for pointing out the relevant reference [13]. HHB and XW are supported by the Natural Sciences and Engineering Research Council of Canada.
References

[1] R. Aharoni, Y. Censor, Z. Jiang, Finding a best approximation pair of points for two polyhedra, *Computational Optimization and Applications* 71 (2018), 509–523. https://doi.org/10.1007/s10589-018-0021-3

[2] H.H. Bauschke, The approximation of fixed points of compositions of nonexpansive mappings in Hilbert space, *Journal of Mathematical Analysis and Applications* 202 (1996), 150–159. https://doi.org/10.1006/jmaa.1996.0308

[3] H.H. Bauschke and J.M. Borwein, On the convergence of von Neumann’s alternating projection algorithm for two sets, *Set-Valued Analysis* 1 (1993), 185–212. https://doi.org/10.1007/BF01027691

[4] H.H. Bauschke and J.M. Borwein, Dykstra’s alternating projection algorithm for two sets, *Journal of Approximation Theory* 79 (1994), 418–443. https://doi.org/10.1006/jath.1994.1136

[5] H.H. Bauschke and P.L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, second edition, Springer, 2017. https://doi.org/10.1007/978-3-319-48311-5

[6] H.H. Bauschke, P.L. Combettes, and D.R. Luke, Finding best approximation pairs relative to two closed convex sets in Hilbert spaces, *Journal of Approximation Theory* 127 (2004), 178–192. https://doi.org/10.1016/j.jat.2004.02.006

[7] H.H. Bauschke and W.M. Moursi, On the behavior of the Douglas-Rachford algorithm for minimizing a convex function subject to a linear constraint, *SIAM Journal on Optimization* 30 (2020), 2559–2576. https://doi.org/10.1137/19M1281538

[8] A. Beck, *First-Order Methods in Optimization*, SIAM 2017. https://doi.org/10.1137/1.9781611974997

[9] A. Beck and M. Teboulle, A fast dual proximal-gradient algorithm for convex minimization and applications, *Operations Research Letters* 42 (2014), 1–6. https://doi.org/10.1016/j.orl.2013.10.007

[10] L. Bottou, F.E. Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, *SIAM Review* 60 (2018), 223–311. https://doi.org/10.1137/16M1080173

[11] Y. Censor, Computational acceleration of projection algorithms for the linear best approximation problem, *Linear Algebra and its Applications* 416 (2006), 111–123. https://doi.org/10.1016/j.laa.2005.10.006
[12] W. Cheney and A.A. Goldstein, Proximity maps for convex sets, *Proceedings of the AMS* 10 (1959), 448–450. https://doi.org/10.2307/2032864

[13] P.L. Combettes, Đ. Dùng, and B.C. Vư, Dualization of signal recovery problems, *Set-Valued and Variational Analysis* 18 (2010), 373–404. https://doi.org/10.1007/s11228-010-0147-7

[14] J. Douglas and H.H. Rachford, On the numerical solution of heat conduction problems in two and three space variables, *Transactions of the AMS* 82 (1956), 421–439. https://doi.org/10.1090/S0002-9947-1956-0084194-4

[15] B. Halpern, Fixed points of nonexpanding maps, *Bulletin of the AMS* 73 (1967), 957–961. https://doi.org/10.1090/S0002-9904-1967-11864-0

[16] K.L. Keys, H. Zhou, and K. Lange, Proximal distance algorithms: theory and practice, *Journal of Machine Learning Research* 20 (2019), 1–38. https://jmlr.csail.mit.edu/papers/v20/17-687.html

[17] J. Kiefer and J. Wolfowitz, Stochastic estimation of the maximum of a regression function, *Annals of Mathematical Statistics* 23(3) (1952), 462–466. https://doi.org/10.1214/aoms/1177729392

[18] K. Lange, *MM Optimization Algorithms*, SIAM 2016. https://doi.org/10.1137/1.9781611974409

[19] K. Lange and K.L. Keys, The proximal distance algorithm, in *Proceedings of the 2014 International Congress of Mathematicians*, pages 96–116, Seoul: Kyung Moon, 4. See also https://arxiv.org/abs/1507.07598

[20] P.-L. Lions, Approximation de points fixes de contractions, *Comptes Rendus des Séances de l’Académie des Sciences Séries A (Sciences Mathématiques)) et B (Sciences Physiques)* 284(21), (June 6, 1977), 1357–1359. https://gallica.bnf.fr/ark:/12148/bpt6k5731057m

[21] P.-L. Lions and B. Mercier, Splitting algorithms for the sum of two nonlinear operators, *SIAM Journal on Numerical Analysis* 16 (1979), 964–979. https://doi.org/10.1137/0716071

[22] H. Robbins and S. Monro, A stochastic approximation method, *Annals of Mathematical Statistics* 22(3) (1951), 400–407. https://doi.org/10.1214/aoms/1177729586

[23] R. Wittmann, Approximation of fixed points of nonexpansive mappings, *Archiv der Mathematik* 58 (1992), 486–491. https://doi.org/10.1007/BF01190119