A Cutting-Plane, Alternating Projections Algorithm for Conic Optimization Problems

Akshay Agrawal akshayka@cs.stanford.edu
Department of Computer Science
Stanford University
Stanford, CA 94305, USA

Advised by: Stephen Boyd

Abstract

We introduce a hybrid projection-localization method for solving large convex cone programs. The method interleaves a series of projections onto localization sets with the classical alternating projections method for convex feasibility problems; the problem is made amenable to the method by reducing it to a convex feasibility problem composed of a subspace and a cone via its homogeneous self-dual embedding.

At each step, the only requirement on the localization set, besides convexity, is that it contain the intersection of the subspace and the cone that define the feasibility problem. The key task, then, is to instantiate localization sets that are both informative and easy to project on — there is a trade-off between these two desiderata.

Our primary contributions are three-fold. First, we present the projection-localization algorithm family and prove that it is convergent. Second, we empirically evaluate the algorithm obtained when the localization sets are polyhedrons defined by cutting planes obtained from projecting upon the subspace and the convex cone. Our findings are promising, but not yet entirely satisfying: our method significantly outperforms alternating projections on complex problems, where the cone is a cartesian product of multiple cones, but either matches or underperforms the alternating directions method of multipliers. For very simple problems, we demonstrate empirically that this method is, at least in appearance, quadratically convergent. Finally, not to be overlooked, our third contribution is the publication of software[1] that enables clients to rapidly prototype and evaluate projection algorithms for convex feasibility and conic problems.

Keywords: Optimization, Cone programming, Cutting planes, Alternating projections

1. Introduction

Because they are robust, easy-to-implement, and scalable, projection methods, which involve a series of iterations in which one or more projections onto convex sets are computed, are an appealing choice of algorithm to solve convex problems. In particular, projection methods are applied to convex feasibility problems, where the inputs are two convex sets with non-empty intersection, and the output is any point in their intersection. These

1. `https://github.com/akshayka/projection-methods/`.

methods are practical, for there exists a link between convex feasibility problems and cone programs: any cone program for which strong duality holds can be summarized by its KKT system, and finding a point that satisfies the KKT system can be reduced to a convex feasibility problem.

Cone programs have emerged as the de-facto intermediate representation for convex programs — most convex programs encountered in practice can be canonicalized to cone programs. For example, CVXPY, a domain specific language for convex optimization, compiles its inputs to cone programs (Diamond and Boyd, 2016). Indeed, many statistical problems, including classical convex model fitting problems like training an ℓ_1-regularized logistic regression model, can be reduced to convex feasibility problems by the canonicalization procedure described by Chu et al. (2013) and carried out by CVXPY.

O'Donoghue et al. (2016) were the first to apply a projection method to the homogeneous, self-dual embeddings of the conic primal-dual pair. Their solver, called *SCS* (splitting conic solver), uses the alternating direction method of multipliers (ADMM) to solve the embedded problem. ADMM has become eminent among projection methods for its agreeablility to distributed computation (Boyd et al., 2011). SCS, as such, scales to large problem instances (see Boyd et al., 2011, for a survey of ADMM); however, it is not suitable for cases in which high accuracy is desired, as ADMM exhibits slow tail convergence (He and Yuan, 2012).

ADMM is not a new method: it can be traced back to the 1950s (Gabay and Mercier, 1976). The history of projection methods reaches still further back. In the 1933, John von Neumann introduced the now-classical method of alternating projections method for finding a point in the intersection of two subspaces; the algorithm, as the name implies, simply projects alternatingly onto the two subspaces and converges to a fixed point that lies within the intersection (Von Neumann, 1950). Years later, Cheney and Goldstein (1959) proved that the method remains viable when the subspaces are replaced with convex sets. A rich literature on projection methods emerged in the intervening decades, which is surveyed by, for example, Bauschke and Borwein (1996), among others. And while von Neumann's method remains a classic historical example, it has not enjoyed the resurgence in popularity that ADMM has seen, precisely because its convergence in practice is often quite slow (indeed, as shown in Boyd, 2003, alternating projections is but a subgradient method in disguise).

The authors of SCS did not benchmark ADMM against other projection methods. This fact, along with the demonstrated effectiveness of classical projection methods like ADMM and the elegant simplicity of von Neumann's alternating projections method, furnishes the motivation for our research: how might the alternating projections method compare to ADMM when applied to the homogeneous self-dual embedding of a cone program, and might there be a systematic way of accelerating the alternating projections method to render it competitive to ADMM?

In this paper, we present first steps towards answering these motivating questions. In particular, we present a family of projection algorithms for solving convex feasibility problems and thus convex cone programs. We instantiate a method from our family that scales to large problems and involves but first-order-like computational effort. Following in the footsteps of O'Donoghue et al. (2016), we apply it to the homogeneous self-dual embedding of the primal-dual pair. The embedding, in turn, furnishes primal or dual certificates of infeasibility when needed.

The remainder of this paper is structured as follows. In section 2, we review the convex feasibility problem and state it in full generality, as this is the problem to which our methods will be applied; in section 3, we motivate the convex feasibiltiy problem by reviewing cone programs and reviewing a reduction from cone programs to convex feasibility problems via the homogeneous self-dual embedding; in section 4, we present our family of algorithms and prove that the algorithms described by the family are convergent; in section 5, we instantiate a particular algorithm, which we dub the cutting-plane, alternating projections method, from our family; in section 6, we present numerical experiments that benchmark our algorithm against competitors; in section 7, we outline future work; and in section 8, we provide closing remarks.

2. The Convex Feasibility Problem

We will concern ourselves with solving convex feasibility problems for the rest of this paper. The inputs to a convex feasibility problem are two convex sets, $C_1 \subseteq \mathbb{R}^n$ and $C_2 \subseteq \mathbb{R}^n$, and the goal is to find any point that lies in the intersection of C_1 and C_2[2]. A standard form for subc a problem is:

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & x \in C_1 \cap C_2. \end{array} \tag{2.1}$$

The problem of finding a point in the intersection of an arbitrary number of sets $S_1, S_2, \ldots, S_N$ can be reduced to (2.1) by taking $C_1 := S_1 \times S_2 \times \cdots \times S_N$, $C_2 := \{x \mid x_1 = x_2 = \cdots = x_N\}$, where x is a block vector composed of the vectors $x_1, x_2, \cdots, x_N$.

2.1 Algorithms

Many projection methods exist that target the convex feasibility problem, and moreover these methods can be generalized using the theory of monotone operators (Ryu and Boyd, 2016). In this section, however, we review with a narrow focus the two algorithms that are relevant to our particular study — von Neumann's method of alternating projections and ADMM. In describing these algorithms, we use the setting of (2.1) and assume that $C_1 \cap C_2 \neq \emptyset$.

2. The theory of projection methods is often set against the backdrop of Hilbert spaces, but we stick to Euclidean space to simplify the exposition.

2.1.1 The Method of Alternating Projections

The method of alternating projections is the simplest of projection methods. It can be described with a single update equation:

$$x_{k+1} := \Pi_{C_2}\left(\Pi_{C_1}(x_k)\right), \tag{2.2}$$

where Π_C denotes the orthogonal projection operator for the set C and x_0 is an arbitrary initial iterate. Cheney and Goldstein (1959) proved that, so long as the sets C_1 and C_2 are closed, convex, and non-empty, then the sequence $\{x_k\}$ converges to a point in $C_1 \cap C_2$; Bauschke and Borwein (1993) showed that the convergence is linear if the sets satisfy certain regularity conditions. A key step in proving convergence of alternating projections is to show that the sequence $\{x_k\}$ is *Fejér monotone* with respect to $C_1 \cap C_2$, which means that,

$$(\forall \overline{x} \in C_1 \cap C_2)(\forall k \in \mathbb{N}) \quad \|x_{k+1} - \overline{x}\|_2 \leq \|x_k - \overline{x}\|_2.$$

Fejér monotonicity holds for the sequence generated by the method of alternating projections because orthogonal projections decrease distances to points that live in the projected-upon (convex) set. The idea of Fejér monotonicity is key, and it is what will let us build upon the alternating projections method to yield new algorithms in section 4.

2.1.2 The Alternating Directions Method of Multipliers

The alternating directions method of multipliers, or ADMM, is an operator splitting method that can be applied to problems of the form

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & x = z. \end{array} \tag{2.3}$$

The algorithm is

$$x_{k+1} = \operatorname{argmin}_x(f(x) + (\rho/2)\|x - z_k - \lambda_k\|_2^2) \tag{2.4}$$

$$z_{k+1} = \operatorname{argmin}_z(g(z) + (\rho/2)\|x_{k+1} - z - \lambda_k\|_2^2) \tag{2.5}$$

$$\lambda_{k+1} = \lambda_k - x_{k+1} + z_{k+1}, \tag{2.6}$$

and it converges (roughly speaking) whenever f and g are closed, proper, and convex (see Boyd et al., 2011, for an additional mild assumption required for convergence). Letting I_C be the indicator function of the set C and taking $f(x) = I_{C_1}(x)$ and $g(z) = I_{C_2}(z)$ yields a problem equivalent to (2.1), giving $x_{k+1} = \Pi_{C_1}(z_k - \lambda_k)$, $z_{k+1} = \Pi_{C_2}(x_{k+1} + \lambda_k)$. It is in this sense that ADMM can be seen as a projection method for the convex feasbility problem.

3. Reducing Cone Programs to Convex Feasibility Problems

A cone program is an optimization problem of the form

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax + s = b \\ & (x, s) \in \mathbb{R}^n \times \mathcal{K}, \end{array} \tag{3.1}$$

where $\mathcal{K}$ is a non-empty, closed, convex cone. A dual of (3.1) can be written as

$$\begin{array}{ll} \text{maximize} & -b^T y \\ \text{subject to} & -A^T y + r = c \\ & (r, y) \in \{0\}^n \times \mathcal{K}^*, \end{array} \tag{3.2}$$

where $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$. If strong duality obtains for (3.1), then the KKT conditions

$$Ax^\star + s^\star = b, \quad s^\star \in \mathcal{K}, \quad A^T y^\star + c = r^\star, \quad r^\star = 0, \quad y^\star \in \mathcal{K}^*, \quad c^T x + b^T y = 0. \tag{3.3}$$

are both necessary and sufficient for optimality. Because the problem of finding a point that satisfies the KKT conditions above can be framed as a convex feasibility problem with an affine set as one input and a cartesian product of cones as the second input, we say that cone programs *reduce* to convex feasibility programs.

3.1 Homogeneous Self-Dual Embeddings of Cone Programs

The KKT system (3.3) can be augmented in such a way that solving the augmentation will yield information about the feasibility or lack thereof of the corresponding cone program. One such embedding is the homogeneous self-dual embedding (Ye et al., 1994); SCS is simply an application of ADMM to 3.4. The details of the embedding are not particularly relevant to our discussion, so we omit them here (see O'Donoghue et al., 2016). Suffice it to say that the embedding introduces two nonnegative variables, κ and τ, whose values at a solution are used to either (1) recover an optimal solution to (3.1), (2) produce a certificate of infeasibility, or (3) if a nonzero solution does not exist, then return an indeterminate status. In particular, the embedding produces the convex feasibility problem

$$\begin{array}{ll} \text{find} & (u, v) \\ \text{subject to} & Qu = v = 0 \\ & (u, v) \in \mathcal{C} \times \mathcal{C}^* \end{array} , \tag{3.4}$$

where

$$u = \begin{bmatrix} x \\ y \\ \tau \end{bmatrix}, \quad v = \begin{bmatrix} r \\ s \\ \kappa \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & A^T & c \\ -A & 0 & b \\ -c^T & -b^T & 0 \end{bmatrix},$$

and $\mathcal{C} = \mathbb{R}^n \times \mathcal{K}^* \times \mathbb{R}_+$. Note how closely (3.4) resembles (3.3). Note, moreover, that (3.4) can be formulated as a convex feasibility problem of form (2.1) where the variable $x := (u, v)$, $C_1 := \mathcal{C} \times \mathcal{C}^*$, and C_2 is the subspace $\{x \mid [Q, -I]x = 0\}$.

The problem (3.4) always admits 0 as a trivial solution, so care must be taken to avoid it. The upshot is that if a solution is found with $\tau > 0$ and $\kappa = 0$, then $(x/\tau, y/\tau, s/\tau)$ is a solution to the primal-dual pair (3.1) and (3.2). To simplify the discussion going fowrard, we will assume that the primal-dual pairs in question are feasible.

4. A Family of Projection-Localization Algorithms

In this section, we present our first contribution: a family of projection algorithms for solving the convex feasibility problem (2.1). The algorithms can be thought of as hybrid localization/alternating-projections procedures; in each iteration, we perform a round of alternating projections and then project the iterate onto a localization set that is known to contain $C_1 \cap C_2$. Indeed, each localization set is perhaps best thought of as a collection of true statements that each $\overline{x} \in C_1 \cap C_2$ must satisfy.

In the following two subsections, we describe the family of algorithms in full and prove that it converges to a point $x^\star \in C_1 \cap C_2$

4.1 Description of the Projection-Localization Algorithm Family

To motivate our algorithm family, let us revisit the alternating projections method. We can decompose the update (2.2) into the two steps

$$y_k = \Pi_{C_1}(x_k), \quad x_{k+1} = \Pi_{C_2}(y_k); \tag{4.1}$$

we refer to these two steps together as one *iteration* of alternating projections. As mentioned in section (2.1.1), the only property needed for this algorithm to converge to a fixed point is Fejér monotonicty of $\{x_k\}$. The insight is this: instead of returning directly to $y_{k+1} \in C_1$ from x_{k+1}, we can instead move first to any point $C_1 \cap C_2$ that is closer to $C_1 \cap C_2$ than x_{k+1} is, in the Euclidean sense. To describe the algorithm family precisely, let $\mathcal{Q}_k$ be any convex set such that $\mathcal{Q}_k \supseteq \mathcal{C}_1 \cap \mathcal{C}_2$. Note that, unlike traditional localization techniques, the localization sets $\mathcal{Q}_k$ need not be bounded nor decreasing in volume. An algorithm belonging to our family is any algorithm that performs the updates

$$y_k = \Pi_{C_1}(x_k), \quad z_k = \Pi_{C_2}(y_k), \quad x_{k+1} = \Pi_{Q_k}(z_k) \tag{4.2}$$

during its k-th iteration. Note that though (4.2) says nothing of techniques like over-projection and heavy-ball momentum, algorithms that use these acceleration techniques but otherwise resemble (4.2) should be considered morally members of our family.

4.2 Proof of convergence.

Theorem 1 *Let C_1, C_2 be non-empty closed convex sets with $C_1 \cap C_2 \neq \emptyset$. Then the sequences x_k, y_k, and z_k yielded by iteration of (4.2) converge in norm to a point $x^\star \in C_1 \cap C_2$.*

Proof As stated in section (2.1.1), the key step is to verify that the sequence of iterates produced by (4.2) is Fejér monotone with respect to $C_1 \cap C_2$. After establishing this, we need only show that a subsequence of the iterates converges (in norm) to a point in $C_1 \cap C_2$: because a Fejér monotone sequence is decreasing and bounded, this second result will prove that our sequence converges in norm to a point in $C_1 \cap C_2$. The proof we provide is modeled after the proof provided by Boyd and Dattorro (2003), though these authors do not set their proof in the language of Fejér monotonicity.

We begin by showing that, for any $\overline{x} \in C_1 \cap C_2$,

$$\|x_{k+1} - \overline{x}\|^2 \leq \|z_k - \overline{x}\|^2 - \|z_k - x_{k+1}\|^2, \tag{4.3}$$

where the norm is the Euclidean norm. This is true because

$$\begin{aligned}\|z_k - \overline{x}\|^2 &= \|z_k - x_{k+1} + x_{k+1} - \overline{x}\|^2 \\ &= \|z_k - x_{k+1}\|^2 + \|x_{k+1} - \overline{x}\|^2 - 2\langle z_k - x_{k+1}, \overline{x} - x_{k+1}\rangle \\ &\geq \|z_k - x_{k+1}\|^2 + \|x_{k+1} - \overline{x}\|^2,\end{aligned}$$

where the inequality follows because x_{k+1} is the projection of z_k onto $\mathcal{Q}_k$ and $\overline{x} \in \mathcal{Q}_k$, since $\mathcal{Q}_k \supseteq C_1 \cap C_2$ by construction, so by convexity of $\mathcal{Q}_k$ we have that $\langle z_k - x_{k+1}, \overline{x} - x_{k+1}\rangle \leq 0$. By similar argumentation, the following two statements can also be shown to be true:

$$\|y_k - \overline{x}\|^2 \leq \|x_k - \overline{x}\|^2 - \|x_k - y_k\|^2 \tag{4.4}$$

$$\|z_k - \overline{x}\|^2 \leq \|y_k - \overline{x}\|^2 - \|y_k - z_k\|^2. \tag{4.5}$$

We can use (4.3), (4.4), and (4.5) to conclude that that the sequence of iterates is Fejér monotone with respect to $C_1 \cap C_2$; that is,

$$\|x_0 - \overline{x}\|,\ \|y_0 - \overline{x}\|,\ \|z_0 - \overline{x}\|,\ \|x_1 - \overline{x}\|,\ \cdots \tag{4.6}$$

is decreasing and bounded, and therefore convergent.

From (4.6) we can observe that $\|y_k - \overline{x}\| \leq \|x_0 - \overline{x}\|$, so the sequence y_k is bounded. As such, it has a convergent subsequence with some limit point $x^\star \in C_1$ (since $y_k \in C_1$ and C_1 closed by assumption). All that remains to show is that $x^\star$ lies in C_2 as well.

The Fejér monotonicity of the iterate sequence, together with (4.5), implies that y_k converges to z_k in norm. This in turn implies that $x^\star \in C_2$, as $z_k = \Pi_{C_2}(y_k)$ and C_2 is closed. Taking $\overline{x}$ to be $x^\star$, the fact that a subsequence of (4.6) converges to 0 (which is true because a subsequence of y_k converges to $x^\star$) implies that the entire sequence converges to 0, and in particular that x_k, y_k, and z_k converge in norm to $x^\star \in C_1 \cap C_2$. ■

5. Cutting-Plane, Alternating Projections for Convex Feasibility

Any policy for generating the localization sets $\mathcal{Q}_k$ induces an algorithm that belongs to the projection-localization family described in section 4. In this section, we present a policy that uses *cutting planes* obtained by projecting on C_1 and C_2 to construct the $\mathcal{Q}_k$. We begin with a brief review of cutting-plane methods. We then present our cutting-plane policy and contextualize it in the setting of the homogeneous self-dual embedding (3.4).

5.1 Cutting Planes

Boyd and Vandenberghe (2007) provide a primer on cutting-plane methods; we highlight the bits relevant to our discussion here.

Cutting-plane methods are used to find a point in some convex set C. In this setting, access to the convex set is mediated by an *oracle.* The oracle supports a single operation: the query operation. When we query the oracle at a point z, it first checks whether z is in our targeted convex set. If z is indeed in C, then it tells us as much; otherwise, the oracle returns a *cutting-plane* — that is, a hyperplane that separates the query point from C. In particular, it returns to us an $a \neq 0$ and b such that

$$a^T x \leq b \; \forall x \in C, \quad a^T z \geq b.$$

The nomenclature is due to the observation that a cutting plane eliminates the halfspace $\{x \mid a^T x > b\}$ from our search for a point in C, and the algorithmic idea is that by accumulating cutting planes, we can form a piecemeal outer approximation of the target set C. There are a number of ways to exploit cutting planes to solve convex optimization problems, but we will only focus on our particular application, as outlined in the following subsection.

5.2 Constructing the Localization Sets via Cutting Planes

Recall that for a convex set C,

$$\langle x_0 - \Pi_C(x_0), x - \Pi_C(x_0) \rangle \leq 0, \; \forall x \in C.$$

This simple fact provides us with a convenient way to obtain true statements (i.e., cutting planes that take the form of supporting hyperplanes) about our convex set $C_1 \cap C_2$. In each iteration of (4.2), we obtain at least two cutting planes: one from projecting onto C_1 and one from projecting onto C_2. Because every point in C_1 belongs to the appropriate halfspace defined by its cutting plane and similarly for C_2, every point in $C_1 \cap C_2$ must belong to the intersection of the two halfspaces corresponding to the cutting planes.

To be more concrete, let O_{C_1} be a cutting-plane oracle for C_1 and O_{C_2} be a cutting-plane oracle for C_2. When an oracle O_C is queried at a point x_0, it returns with exactly two pieces of information:

$$\begin{aligned} &\tilde{x} := \Pi_C(x_0), \\ &\mathcal{H} := \{x \mid \langle x_0 - \tilde{x}, x - \tilde{x} \rangle \leq 0\} \supseteq C, \end{aligned}$$

where $\tilde{x}$ is the projection of x_0 onto C and $\mathcal{H}$ is a hafspace containing C and defined by the cutting plane obtained from the projection. Equipped with oracles O_{C_1} and O_{C_2}, we can describe a cutting-plane instantiation of the projection-localization family, which we call the cutting-plane alternating projections method. The method, in its simplest form, is an iteration

$$\begin{aligned} (y_k, \mathcal{H}_{y_k}) &= O_{C_1}(x_k), \\ (z_k, \mathcal{H}_{z_k}) &= O_{C_2}(y_k), \\ \mathcal{Q}_k &:= \mathcal{Q}_{k-1} \cap \mathcal{H}_{y_k} \cap \mathcal{H}_{z_k} \\ x_{k+1} &= \Pi_{Q_k}(z_k), \end{aligned} \tag{5.1}$$

where $\mathcal{Q}_0 = \mathbb{R}^n$.

There is a trade-off between the number of cutting planes that describe each $\mathcal{Q}_k$ (i.e., the descriptiveness of the outer approximations) and the cost of computing each iteration. The more descriptive the outer approximations, the more progress each iteration will make but the more expensive each projection onto the $\mathcal{Q}_k$ will be. For this method to be practical when applied to large problems, the projections onto $\mathcal{Q}_k$ must be inexpensive relative to the projections onto C_1 and C_2. So, in practice, we might impose a cap on the maximum number of cutting planes that define $\mathcal{Q}_k$, and we might also experiment with different policies for pruning cutting planes. Boyd and Vandenberghe (2007) propose heuristics for quantifying the informativeness of a cutting plane and for pruning redundant ones, and many of these heuristics may prove relevant here. A thorough analysis of pruning policies as they pertain to our method is a topic for future work.

5.2.1 Application to the Homogeneous, Self-Dual Embedding

In the homogeneous self-dual embedding (3.4), our set C_1 is a cartesian product of cones and our set C_2 is a subspace. We can specialize the application of our cutting-plane method to this embedding, in the following two senses. First, the projection onto C_1 yields not a single halfspace but in fact a halfspace for *each* of the cones that compose the cartesian product C_1. Indeed, the cone $\mathcal{K}$ that is present in the provenance of the embedding (i.e., in problem (3.1) may itself be a a cartesian product of cones. And second, the projection onto the subspace C_2 yields not a halfspace but a hyperplane that all points in C_2 must satisfy; this holds because of the orthogonal complementarity properties of subspaces (i.e., $\langle x_0 - \Pi_{C_2}(x_0), x - \Pi_{C_2}(x_0)\rangle = 0$, for all $x \in C_2$). We can use these two facts in order to generate more precise outer approximations $\mathcal{Q}_k$ of $C_1 \cap C_2$ at each step of the iteration (5.1).

6. Software and Numerical Experiments

In order to empirically benchmark our cutting-plane alternating projections method, we implemented a python library that allows for rapidly prototyping projection methods and evaluating them on a suite of cone programs that are reduced to covnex feasibility form. We used our library to evaluate the cutting-plane alternating projections algorithm on a number of problems. Our initial findings suggest that the cutting-plane alternating projections method significantly outperforms classicial alternating projections; however, it roughly matches the performance of and in some cases underperforms ADMM/SCS. An interesting result in and of itself is that Dykstra's algorith, which was not surveyed in this paper due to space constraints, more or less tracks the performance of alternating projections and thus underperforms ADMM.

6.1 A Library for Rapidly Prototyping Projection Methods

An alpha version source code is freely available on Github[3]; even in its early stages, the library should prove useful for other researchers. The projection methods library is implemented in Python and adheres to an object-oriented framework that closely matches the oracle formulation presented in section 5.2. Clients can choose to solve abritrary convex feasibility problems, specified using an atomic set of convex sets and a cartesian product of sets, as well as homogeneous self-dual embeddings. Currently, only the non-negative, zero, free, and second-order cones are implemented. Our library does interface with CVXPY, so users can describe arbitrary convex sets if they so wish, at the cost of incurring the overhead that CVXPY brings to the table.

For a research library designed for prototyping, it is reasonably efficient. When solving homogeneous self-dual embeddings, the majority of time is spent factorizing the KKT matrix of the projection onto the affine set; this factorization occurs exactly once. Projections onto the aforementioned cones are computed analytically. For reference, our library takes on the order of seconds (less than 10) to run 300 iterations of von Nuemann alternating projections on a cone program with variable size 1000 and 10^4 non-zeros in the data matrix when executing on an Ubuntu virtual machine (vagrant on VirtualBox) attached to a 2012 MacBook Pro.

6.2 Numerical Experiments

We evaluated the cutting-plane alternating projections method on a number of randomly generated feasible cone programs (i.e., problems of the form 3.1). Our algorithm underperforms ADMM/SCS when applied to linear programs and composite cone programs, but exhibits quadratic-like convergence when applied to problems where the cone $\mathcal{K}$ is simply one of the zero cone, the non-negative cone, or the second order cone. Our extremely fast convergence in the latter experiments is the empirical counterpart of the result proved by Ali et al. (2017) — these authors present a quasi-Newton ADMM method and show that it converges quadratically for these simple cone programs.

When evaluating our algorithms, we defined the residual at each step as the sum of the ℓ_2 distances of each iterate x_k to the sets C_1 and C_2. Additionally, we reported the relative primal error, defined as $|c^T x - p^\star|/|p^\star|$, for the cone program (3.1) that was the provenance of the feasibility problem. In our plots, "dyk" refers to Dykstra's algorithm, "altp" to von Neumann's alternating projections algorithm, "scs“ to SCS/ADMM, "apop_alt" to the cutting-plane alternating projections method, and "apop_alt" to the cutting-plane averaged projections method, where averaged projections is the well-known, classical variant of alternating projections (it is alternating projections applied to the cartesian product of the sets

3. `https://github.com/akshayka/projection-methods/`

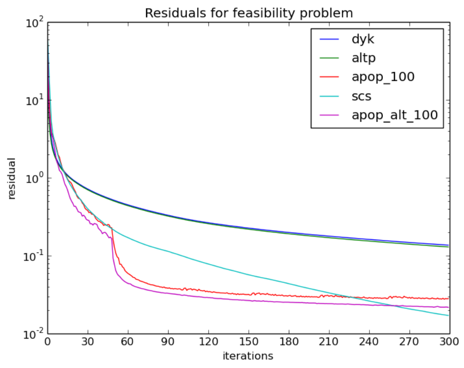


Figure 1: Random linear program. Note that SCS outperforms our cutting-plane method (apop_alt), and our method in turn outperforms alternating projections.

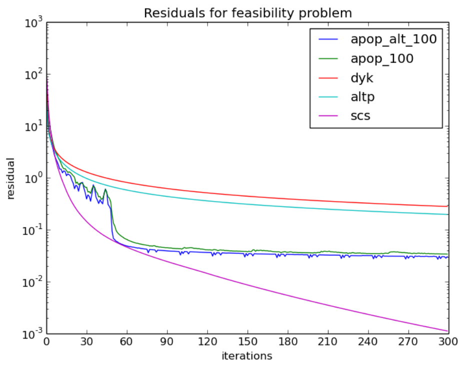


Figure 2: Random cone program, $\mathcal{K}$ the cartesian product of the zero, non-negative, and two second-order cones, each of size 500, and with a data matrix holding 10^4 random Gaussian entries.

C_1 and C_2 and the average set). In all runs of the cutting-plane methods, we evicted the least recently added hyperplanes and halfspaces after the number of cutting planes exceeded some threshold.

6.2.1 Linear Programs

Our first type of problem is a simple linear program subject to equality constraints $Gx - h = 0$ and $x \geq 0$. We solved exactly the same problem instance solved by Ali et al. (2017), ensuring that the generated problem was feasible. We took G of size 300 by 600, fully dense, as did Ali et al. (2017), and canonicalized the problem to a cone program in the obvious way using the zero and the non-negative cones. Our results are plotted in Figure 1; note that we limited the cutting-plane methods to hold at most 50 halfspaces and 50 hyperplanes. The cutting-plane alternating projection method achieved a relative primal error of 3.3×10^{-4}, while SCS achieved an error of 1.2×10^{-5}. Alternating projections fared much worse, achieving an error of 1.4×10^{-2}. The sharp drop in the residuals of the cutting-plane method starting at iteration 50 is curious, and it begs further investigation into pruning policies and more generally into the management of cutting planes. Also note that the cutting-plane methods initially outpace SCS, only to flatten out.

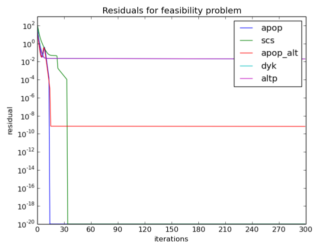
6.2.2 Cartesian Product Cone Programs

The second problem on which we profiled our algorithms was a random cone program where the cone $\mathcal{K}$ was the cartesian product of the zero, non-negative, and two second-order cones, each of size 500. The variable x was of dimension 1000, data matrix $A \in \mathbb{R}^{2000 \times 1000}$ was randomly filled with 10^4 Gaussian entries. We followed exactly the same procedure outlined by O'Donoghue et al. (2016) in order to ensure that the generated cone program was feasible.

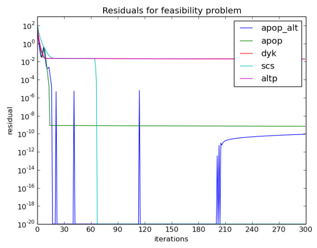
Our results are plotted in Figure 2. The relative primal errors for the alternating cutting-plane method, SCS, and alternating projections were 7.2×10^{-3}, 1.8×10^{-6}, and 9.9×10^{-2}.

6.2.3 Single-Cone Programs

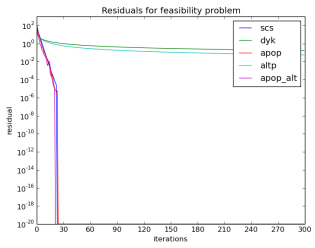
Finally, the last type of problem involved problems where the cone was simple either the zero cone, the non-negative cone, or the second-order cone. Both the cutting-plane methods and SCS solve these simple problems more or less exactly (up to numerical fluctuations), as shown in Figure 3, while classical alternating projections struggles. In this light, the quadratic convergence result by Ali et al. (2017) is perhaps not that interesting.



Random zero cone.



Random non-negative cone.



Random second-order cone.

Figure 3: Our algorithm exhibits quadratic-like convergence on simple problems.

7. Future Work

Our numerical experiments show that while cutting-plane method certainly accelerates alternating projections, it does not in its current form outperform ADMM. There are two complementary lines of future work that we are pursuing in an attempt to increase the performance of our algorithm. The first line of work involves experimenting with pruning policies, caps on the size of the $\mathcal{Q}_k$, over-projections, and heavy-ball momentum in an attempt to accelerate the cutting-plane alternating projections method in its current form. The second line of work is to use the cutting-plane method to augment ADMM itself. Such an augmentation must be done with care to ensure that it does not break the convergence properties of ADMM, nor interfere with properties that make ADMM faster than alternating projections to begin with. Both lines of research are being actively investigated.

8. Conclusion

In this paper, we presented a family of hybrid localization-projection methods that can be viewed as a generalization of von Neumann's method of alternating projections. We then instantiated a particular algorithm from our family that constructs localization sets using cutting planes, and showed empircally that this algorithm significantly accelerates von

Neumann's classical alternating projections method but underperforms ADMM, at least when applied to the homogeneous self-dual embedding of a conic primal-dual pair. That such a simple technique accelerated such a simple method is nonetheless promising, and this result motivates further study into augmenting projection algorithms with cutting-planes.

Acknowledgments

I would like to acknowledge Stanford professors Stephen Boyd, for guiding the development of this investigation, and John Duchi, for teaching an engaging course on statistical learning theory for which this report was prepared.

References

Alnur Ali, Eric Wong, and J Zico Kolter. A semismooth newton method for fast, generic convex programming. *arXiv preprint arXiv:1705.00772*, 2017.

Heinz H Bauschke and Jonathan M Borwein. On the convergence of von neumann's alternating projection algorithm for two sets. *Set-Valued Analysis*, 1(2):185–212, 1993.

Heinz H Bauschke and Jonathan M Borwein. On projection algorithms for solving convex feasibility problems. *SIAM review*, 38(3):367–426, 1996.

Stephen Boyd. Subgradient methods. 2003. Lecture notes on convex optimization, from the Stanford course EE 364B.

Stephen Boyd and Jon Dattorro. Alternating projections. 2003. Lecture notes on convex optimization, from the Stanford course EE 364B.

Stephen Boyd and Lieven Vandenberghe. Localization and cutting-plane methods. 2007.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Ward Cheney and Allen A Goldstein. Proximity maps for convex sets. *Proceedings of the American Mathematical Society*, 10(3):448–450, 1959.

Eric Chu, Neal Parikh, Alexander Domahidi, and Stephen Boyd. Code generation for embedded second-order cone programming. In *Control Conference (ECC), 2013 European*, pages 1547–1552. IEEE, 2013.

Steven Diamond and Stephen Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

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

Bingsheng He and Xiaoming Yuan. On the o(1/n) convergence rate of the douglas–rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

Ernest K Ryu and Stephen Boyd. Primer on monotone operator methods. 2016.

John Von Neumann. *Functional Operators (AM-22), Vol. II. The Geometry of Orthogonal Spaces.* Princeton University Press, 1950. Reprint of lecture notes originally compiled in 1933.

Yinyu Ye, Michael J Todd, and Shinji Mizuno. An o($\sqrt{nL}$)-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19(1):53–67, 1994.