UNIVERSITY OF CALIFORNIA
Los Angeles

First-Order Methods for Self-Dual Linear Programs

by

Justin Shao

2024

ABSTRACT OF THE THESIS

First-Order Methods for Self-Dual Linear Programs

by

Justin Shao
Master of Science in Electrical and Computer Engineering
University of California, Los Angeles, 2024
Professor Lieven Vandenberghe, Chair

Recently, there has been increasing interest in first-order methods for conic linear programming, an extension of linear programming in which scalar inequalities are replaced with vector inequalities defined relative to convex cones. Conic programming is an important class of optimization problems with a wide range of applications including in operations research, finance, and engineering. There is a rich history of extensive research into methods for solving them efficiently and accurately.

First-order methods show potential to improve the efficiency of conic programming solvers since they circumvent the need to factor matrices or solve linear systems, operations which can be especially computationally expensive when modeling complex problems or problems with large data sets. In this work, we examine two first-order methods for solving conic linear programs using a self-dual embedding whose solution yields a primal-dual optimal solution or a certificate of primal or dual infeasibility. These methods require no feasible starting points, and the most expensive operations are matrix-vector multiplications or projections onto cones which makes it easy to customize the methods to exploit problem structure.

We compare two methods, the primal-dual hybrid gradient method and the extragradient

method, with a suite of practical improvements. We test these methods on linear programs and other non-polyhedral conic linear programs. The numerical experiments indicate that these methods solve the self-dual embedding efficiently and accurately.

Vwani Roychowdhury

Lin Yang

Lieven Vandenberghe, Committee Chair

University of California, Los Angeles

2024

For my beloved family and friends.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This thesis is made possible by the support and teachings from many people I had the honor of studying and interacting with during my time as a student at UCLA. Firstly, I would like to extend sincerest thanks to Professor Lieven Vandenberghe for his advisorship. Learning in his courses since I was an undergraduate student and working on this thesis under his guidance has been a fantastic and motivating educational experience, for which I will forever be grateful.

I would also like to thank my committee members Professor Vwani Roychowdhury and Professor Lin Yang, whose courses have been a big inspiration for my continued studies in computation and applied mathematics. Learning from these excellent teachers and scientists has kept me excited and passionate about my current and future studies. My experiences with all of my committee members inspires me to strive to be an expert creator of knowledge.

Furthermore, I am thankful for my friends and community who have all been great sources of support, energy, and vitality. Without this fellowship with my friends and classmates I would not have been able to produce my best efforts.

Finally, I am grateful for my family who have continually supported me through my educational journey and have been a persistent source of encouragement and strength.

VITA

2022 B.S. (Eletrical Engineering) and Minor (Mathematics), University of California, Los Angeles.

CHAPTER 1

Introduction

Linear programming is an important class of convex optimization problems that can be used to solve a wide range of problems in applied mathematics, operations research, and engineering. There has long been focus on developing algorithms for solving linear programs; in 1947, Dantzig developed the simplex algorithm for solving linear programs as described in [Dan63], which was one of the earliest works in numerical optimization. The simplex method finds an optimal vector by traversing the vertices of the feasible polyhedron. While the method is still popularly used and generally performs well, Klee and Minty in [KM72] show that the simplex algorithm runs in exponential worst-case complexity by constructing a simple example linear program and showing that the simplex algorithm visits every vertex of its feasible polyhedron. Yudin and Nemirovskii in [YN77, YN83] study the informational complexity of convex optimization and introduce the ellipsoid method; Khachiyan applied their findings to linear programming in [Kha79] and proved that the worst-case complexity of the ellipsoid method for linear programming is polynomial. Though the ellipsoid method was not competitive in practice with the simplex algorithm, it inspired further research into methods besides the simplex method.

The development of interior-point methods, methods that iterate within the feasible region, for linear programs was an important milestone in the field of linear programming algorithms, and these methods are still widely used today. Karmarkar introduced a new interior-point method for linear programming in [Kar84], which used projective transformations and solved linear programs in polynomial complexity. This successful method inspired

many following works leading to new interior-point methods and a renewed interest in related approaches. These notably included the affine scaling method [Bar86, VMF86, Dik67] and the barrier method [Ren88].

In 1994, Nesterov and Nemirovski in [NN94] analyzed interior-point methods for convex programming, including semidefinite programming and second-order cone programming. In this work, they apply analysis and methods of linear programs to non-polyhedral conic optimization problems. Primal-dual interior-point methods are used in many commercial solvers including Mosek, SeDuMi, and SDPT3 [ApS19, Stu99, TTT99].

A conic linear program is a convex optimization problem with linear objective function and conic inequality constraints. It can be viewed as a generalization of linear programming, and many of the results and algorithms for linear programming can be naturally extended to this framework. Conic linear programming is widely studied, and it can model important cases of optimization problems including semidefinite programming and second-order cone programming. It is also the basis of many modeling packages including CVX, CVXPY, PICOS, YALMIP, and CVXOPT [GB14, GB08, DB16, SS22, L04, ADV13].

Although surprisingly many convex optimization problems can be modeled as semidefinite programs and second-order cone programs, there are optimization problems that cannot be modeled with these tools, such as problems with power or exponential constraints, or that incur a heavy overhead cost when reformulated in the form of a more standard problem. As a result, there has been growing interest in interior-point methods for non-symmetric cones, for example in [SY15], [ADV10], [Ser15], and in implementations of power cones and exponential cones for Mosek and Clarabel [GC21].

There is also growing interest is in the development of methods besides interior-point methods. Interior-point methods are generally able to solve problems in few iterations, but the computational per-iteration cost is high, which creates difficulties in scaling. One celebrated approach for semidefinite programming is the Burer-Montiero approach, introduced in [BM03], which represents a positive semidefinite matrix in factored form and optimizes

over the factors. This makes the problem non-convex in general but reduces the dimension of the problem.

Another direction of growing interest is in first-order methods for conic linear programs, the most expensive operations of which are matrix-vector products and projections onto cones. These operations scale much better to larger problems than matrix factorizations, which are used in simplex and interior-point methods. For instance, Zheng et al. in [ZFP20] employ chordal decomposition to first reformulate a semidefinite program to one with smaller semidefinite constraints then use the alternating direction method of multipliers, a first-order method, to solve the reformulated problem. In this work, we examine two first-order primal-dual methods, the primal-dual hybrid gradient method and the extragradient method, applied to a self-dual embedding.

In this thesis, we first review conic linear programming in Chapter 2. Then, we describe the primal-dual hybrid gradient method and the extragradient method in Chapter 3 and Chapter 4 respectively. In Chapter 5, we describe some numerical experiments involving these methods, and finally we conclude in Chapter 6 discussions on these methods and some directions for further research.

CHAPTER 2

Conic Linear Programming

In this chapter, we review conic linear programming and the necessary background and notations for this work. In Section 1, we review the background of conic linear programming, including the standard cases of linear programming, second-order cone programming, and semidefinite programming and their applications. We review duality in Section 2 and note some important results and their consequences. Then, in Section 3 we review self-dual embeddings and how this reformulation can be used to develop efficient solvers. Finally, in Section 4 we give a brief overview of algorithms that are commonly used to solve conic linear programs.

2.1 Conic Linear Programming

Conic linear programming is a general and important form that can be used to model many convex optimization problems. It is an extension of linear programming and the foundation of many solvers like YALMIP, CVX, and CVXPY [L04, GB14, GB08, DB16].

Firstly, a *conic linear program* is an optimization problem with a linear objective function and inequality constraints defined with respect to a cone. For this work, we assume that the objective function is the standard inner product, and the linear operators defining the constraints are left multiplications by matrices and therefore have adjoint operators left

multiplication by the transposes of the matrices. So, a conic linear program is written as

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & Gx + s = h \\ & s \in \mathcal{K} \end{aligned} \tag{2.1}$$

where $\mathcal{K}$ is a proper convex cone. A *cone* is a nonempty set $\mathcal{K}$ such that

$$x \in \mathcal{K} \Rightarrow tx \in \mathcal{K} \quad \forall t \geqslant 0$$

and a *proper cone* is a convex cone that is closed, has nonempty interior, and does not contain any lines. For a proper cone $\mathcal{K}$, the generalized inequality $\preceq_{\mathcal{K}}$ with respect to $\mathcal{K}$ is defined

$$x \preceq_{\mathcal{K}} y \iff y - x \in \mathcal{K},$$

and the inequality holds strictly as

$$x \prec_{\mathcal{K}} y \iff y - x \in \operatorname{int}(\mathcal{K}).$$

For this work, we write m_1 and m_2 for the respective number of rows in A and G. Note that $Gx + s = h$ and $s \in \mathcal{K}$ are equivalently $Gx \preceq_{\mathcal{K}} h$.

Conic linear programming is a richly studied topic, and we refer to [BV04, BN01] for conic linear programming and convex optimization in general, and to [Roc70] for background on cones and convexity.

2.1.1 Linear Programming

Linear programming (LP) is an extremely important optimization problem that can find application in every discipline in engineering and modeling. It is also important to study because theory and algorithms can often be appropriately generalized to conic linear programming.

A linear program has the form

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & Gx \leqslant h \end{aligned} \tag{2.2}$$

where the inequality $\leqslant$ is element-wise comparison and is equivalent to $\preceq_{\mathbb{R}_+^{m_2}}$.

As a result of the wide applicability of linear programming, it has a long history of being extensively studied. In the 1940's, Dantzig introduced the simplex method for linear programming [Dan63]. Since then, there have been numerous advancements in the development of LP solvers. Khachiyan in [Kha79] first showed with the ellipsoid method, earlier introduced in [YN77, YN83], that linear programs could be theoretically solved in polynomial time, and in 1984 Karmarkar developed an interior-point method with polynomial complexity for LPs [Kar84] that performs well in practice and inspired more research on efficient algorithms for solving LPs. Later, there has been more research on improving the efficiency of these algorithms and applying them to large-scale programs.

Integer linear programming is a well studied discipline of optimization related to linear programming, where the optimization variable of an LP is constrained further to be integral, that is, x in (2.2) is additionally constrained to $x \in \mathbb{Z}^n$. This can be used to model discrete optimization problems like in combinatorics and graph theoretic optimization problems, and efficient LP solvers are required for solving integer linear programs. We direct to [Sch98] for reference on integer linear programming.

2.1.2 Second-Order Cone Programming

Another important class of conic linear programs are second-order cone programs (SOCPs). An SOCP is a program of the form

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & \|B_{k0}x + d_{k0}\|_2 \leqslant B_{k1}x + d_{k1}, \quad k = 1, \ldots, r \end{aligned}$$

where $\|\cdot\|_2$ is the Euclidean norm.

Using the *second-order cone*, defined as

$$\mathcal{Q}^p = \{(x, y) \in \mathbb{R}^{p-1} \times \mathbb{R} \mid \|x\|_2 \leqslant y\},$$

the SOCP can be expressed as a conic linear program of the form (2.1) by choosing

$$G = \begin{bmatrix} -B_{10} \\ -B_{11} \\ \vdots \\ -B_{r0} \\ -B_{r1} \end{bmatrix}, \qquad h = \begin{bmatrix} d_{10} \\ d_{11} \\ \vdots \\ d_{r0} \\ d_{r1} \end{bmatrix}, \qquad \mathcal{K} = \mathcal{Q}^{m_{10}} \times \cdots \times \mathcal{Q}^{m_{r0}}$$

where $m_{k0} - 1$ is the number of rows in B_{k0} for $k = 1, \ldots, r$.

SOCPs have been extensively studied and have wide applications including modeling norm minimization problems, approximation problems, robust optimization, portfolio optimization, and other nonlinear optimization problems. For more on applications of second-order cone programs, see [LVB98, AG03]. One example in robust linear programming is the problem

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & g_k^T x \leqslant h_k, \quad k = 1, \ldots, m \end{aligned}$$

where for each k, g_k is uncertain or variable, but known to be within an ellipsoid

$$g_k \in \{\bar{g}_k + P_k u \mid \|u\|_2 \leqslant 1\}.$$

So, each constraint is $\bar{g}_k^T x + u^T P_k^T x \leqslant h_k$ for all $\|u\|_2 \leqslant 1$, and equivalently

$$\bar{g}_k^T x + \sup_{\|u\|_2 \leqslant 1} \left(u^T P_k^T x\right) \leqslant h_k.$$

Since $\sup_{\|u\|_2 \leqslant 1} \left(u^T P_k^T x\right) = \left\|P_k^T x\right\|_2$, the robust linear program can be written as the SOCP

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & \left\|P_k^T x\right\|_2 \leqslant h_k - \bar{g}_k^T x, \bar{k} = 1, \ldots, m \end{aligned}$$

Another example of how an SOCP can model a nonlinear optimization problem is in maximizing the geometric mean of nonnegative affine functions:

$$\begin{aligned} \text{maximize} \quad & \prod_{k=1}^{p} \left(a_k^T x + b_k\right)^{1/p} \\ \text{subject to} \quad & a_k^T x + b_k \geqslant 0, \quad k = 1, \ldots, p. \end{aligned}$$

First, consider the case where p is a power of two, and write $p = 2^K$. Then, the problem can be equivalently written

$$\begin{aligned} \text{maximize} \quad & v_1 v_2 \cdots v_p \\ \text{subject to} \quad & 0 \leqslant v_k = a_k^T x + b_k, \quad k = 1, \ldots, p \end{aligned}$$

which can then be written as

$$\begin{aligned} \text{maximize} \quad & (t_1)_1 (t_1)_2 \cdots (t_1)_{p/2} \\ \text{subject to} \quad & 0 \leqslant v_k = a_k^T x + b_k, \quad k = 1, \ldots, p \\ & (t_1)_1^2 \leqslant v_1 v_2 \\ & (t_1)_2^2 \leqslant v_3 v_4 \\ & \vdots \\ & (t_1)_{p/2}^2 \leqslant v_{p-1} v_p \\ & t_1 \geqslant 0. \end{aligned}$$

Repeating this procedure of rewriting the problem, each time reducing the number of factors in the objective function by half and adding the appropriate constraints, the problem can

be expressed as

$$
\begin{aligned}
\text{maximize} \quad & t_K \\
\text{subject to} \quad & 0 \leqslant v_k = a_k^T x + b_k, \quad k = 1, \ldots, p \\
& t_K^2 \leqslant (t_{K-1})_1 (t_{K-1})_2, \; t_K \geqslant 0 \\
& (t_{K-1})_1^2 \leqslant (t_{K-2})_1 (t_{K-2})_2, \; (t_{K-1})_2^2 \leqslant (t_{K-2})_3 (t_{K-2})_4, \; t_{K-1} \geqslant 0 \\
& \vdots \\
& (t_1)_1^2 \leqslant v_1 v_2 \\
& (t_1)_2^2 \leqslant v_3 v_4 \\
& \vdots \\
& (t_1)_{p/2}^2 \leqslant v_{p-1} v_p \\
& t_1 \geqslant 0.
\end{aligned}
$$

Since hyperbolic constraints of the form

$$u^T u \leqslant yz, \qquad y \geqslant 0, \qquad z \geqslant 0$$

are equivalently

$$\left\| \begin{bmatrix} 2u \\ y - z \end{bmatrix} \right\|_2 \leqslant y + z, \qquad y \geqslant 0, \qquad z \geqslant 0$$

the previous formulation of the geometric mean problem can be written as a second-order cone program.

In the case where p is not a power of 2, then we can first define add additional variables $a_k = 0$ and $b_k = 1$ for $k = p+1, \ldots, 2^K$ where 2^K is the smallest power of 2 greater than p, then apply the same procedure. We note that while this reformulation conveniently yields a well-studied form, the additional variables incur additional computational cost for a solver.

2.1.3 Semidefinite Programming

A semidefinite program (SDP) is a program of the form

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} && x_1 A_{k1} + x_2 A_{k2} + \cdots + x_n A_{kn} \preceq_{\mathbb{S}_+^{p_k}} B_k, \quad k = 1, \ldots, r \end{aligned}$$

where $\mathbb{S}_+^n$ is the cone of positive semidefinite $n \times n$ matrices, and $A_{ki},\ B_k \in \mathbb{S}_+^{p_k}$ for $i = 1, \ldots, n$ and $k = 1, \ldots, r$. To write this program in the form of (2.1), we first define a vectorization operation for symmetric matrices $\text{vec} : \mathbb{S}^p \to \mathbb{R}^{\frac{p(p+1)}{2}}$ as

$$\text{vec}(M) = \left(M_{11}, \sqrt{2}M_{21}, \sqrt{2}M_{31}, \ldots, \sqrt{2}M_{p1}, M_{22}, \sqrt{2}M_{32}, \ldots, M_{p2}, \ldots, M_{pp}\right)$$

and the cone of vectorized positive semidefinite matrices

$$\mathcal{S}^p = \{\text{vec}(M) \mid M \in \mathbb{S}_+^p\}.$$

Then, the SDP can be written in the form of (2.1) as

$$G = \begin{bmatrix} \text{vec}(A_{11}) & \text{vec}(A_{12}) & \cdots & \text{vec}(A_{1n}) \\ \text{vec}(A_{21}) & \text{vec}(A_{22}) & \cdots & \text{vec}(A_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ \text{vec}(A_{r1}) & \text{vec}(A_{r2}) & \cdots & \text{vec}(A_{rn}) \end{bmatrix}, \quad h = \begin{bmatrix} \text{vec}(B_1) \\ \text{vec}(B_2) \\ \vdots \\ \text{vec}(B_r) \end{bmatrix}, \quad \mathcal{K} = \mathcal{S}^{p_1} \times \cdots \times \mathcal{S}^{p_r}.$$

The factor of $\sqrt{2}$ is added so that the standard inner product of the vectorized matrices coincides with the standard inner product of matrices, that is

$$\text{tr}(AB) = \text{vec}(A)^T \text{vec}(B)$$

for symmetric A and B.

Semidefinite programming has many applications in controls, experiment design, combinatorial optimization, and others in engineering.

In combinatorial optimization, the maximum cut problem can be analyzed with semidefinite programming, as described in [GW95]. For a graph $G = (V, E)$ with weighted edges,

denoted w_{ij} for $(i,j) \in E$, the objective of the maximum cut problem is to find a set of vertices $S \subseteq V$ that maximizes the sum of weights of edges with one endpoint in S and the other in $V \backslash S$. An upper bound can be formulated as follows: for each vertex i, we introduce an indicator variable y_i with $y_i = 1$ if $i \in S$ and -1 otherwise; the weight of the cut is

$$\frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - y_i y_j).$$

We consider the matrix Y with $Y_{ij} = y_i y_j$ for all $i, j \in V$. Y is such that $\operatorname{rank}(Y) = 1$ and $Y_{ii} = 1$ for all $i \in V$. Relaxing the constraints gives the semidefinite program

$$\begin{array}{ll} \text{maximize} & \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - Y_{ij}) \\ \text{subject to} & Y_{ii} = 1, \quad i \in V \\ & Y \succeq 0. \end{array}$$

For more materials on combinatorial optimization and semidefinite programming, see [Lov03, Ali95, GW95].

Semidefinite programming can be used to solve optimization problems involving eigenvalues. Consider for example the problem of minimizing the sum of the r largest eigenvalues of $A(x) = A_0 + A_1 x_1 + \cdots + A_n x_n$ with $A_1, \ldots, A_n$ symmetric. This problem can be solved with the semidefinite program

$$\begin{array}{ll} \text{minimize} & rt + \operatorname{tr}(X) \\ \text{subject to} & tI + X - A(x) \succeq 0 \\ & X \succeq 0 \end{array}$$

of variables $x \in \mathbb{R}^n$, $t \in \mathbb{R}$, and X symmetric. We refer to [VB96, VB99] for additional notes on semidefinite programming and some of its applications.

The class of problems that can be modeled as semidefinite programs is surprisingly large, but these reformulations often require many additional variables. One such possible refor-

mulation is for the problem of minimizing trace norm:

$$\text{minimize} \quad \|A_1 x_1 + \cdots + A_n x_n - B\|_*$$

where $\|\cdot\|_*$ is the *trace norm*, also known as the *nuclear norm*, which is the sum of singular values. This problem can be reformulated into a semidefinite program of the form

$$\begin{aligned} \text{minimize} \quad & \left(\operatorname{tr}(V_1) + \operatorname{tr}(V_2)\right)/2 \\ \text{subject to} \quad & \begin{bmatrix} V_1 & A(x) - B \\ (A(x) - B)^T & V_2 \end{bmatrix} \succeq 0 \end{aligned}$$

where V_1 and V_2 are additional symmetric matrix variables and the operator A is defined $A(x) = A_1 x_1 + \cdots + A_n x_n$. We note that the addition of V_1 and V_2 introduces a considerable overhead.

Trace norm minimization is useful because the trace norm is the convex envelope of rank in the unit operator norm ball, meaning that it is the best pointwise approximation for rank in the ball. This makes minimization of trace norm a useful heuristic for finding low rank solutions, similarly to how l_1 norm minimization serves as a heuristic for finding sparse solutions. In fact, Recht and others in [RFP10] show that under certain conditions, a minimum rank solution can be exactly recovered by solving the convex trace norm minimization problem. We direct to [RFP10] for reference on trace norm minimization and to [Faz02] for a report on the optimization problem and some of its applications.

We refer to [BN01, VB96] for further reading on LP, SOCP, and SDP theory.

2.1.4 Nonsymmetric Cone Programming

A *symmetric cone* is a cone that is self-dual and homogeneous, meaning that its automorphism group has a subgroup that acts transitively on its interior. For practical purposes, the important symmetric cones are the nonnegative orthant, the second-order cone, the positive semidefinite cone, and Cartesian products of the three previous. Primal-dual interior-point

methods for linear programs extend very naturally to conic LPs with symmetric cone constraints. Symmetric cones have special algebraic properties, and we direct to [FK94] for reference on background of symmetric cones. In [NT97], Nesterov and Todd introduce self-scaled barriers as an extension of the logarithmic barrier, and they find that the cones that admit self-scaled barriers are exactly the symmetric cones, and they describe symmetric primal-dual interior-point methods for conic LPs on symmetric cones.

A *nonsymmetric* cone is a cone that is not symmetric. There is growing interest in the study of conic linear programs with nonsymmetric cones; see [SY15, ADV10, Ser15, Nes06, DA22] for examples of works investigating nonsymmetric conic optimization.

An important example of a nonsymmetric cone is the *exponential cone*

$$\mathcal{K}_{\text{exp}} = \text{cl}\left(\left\{(x, y, z) \in \mathbb{R}^3 \mid ye^{y/x} \leqslant z,\ y > 0\right\}\right).$$

Using the exponential cone, many convex functions can be modeled as conic linear programs, including logarithms, entropy, and exponential functions, which cannot be solved with SOCPs or SDPs.

An important family of problems that can be modeled with exponential cones is *geometric programming*, which considers problems of the form

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} && \log\left(\sum_{i=1}^{n_k} \exp\left(a_{ki}^T x + b_{ki}\right)\right) \leqslant 0, \quad k = 1, \ldots, r. \end{aligned}$$

Zener studied geometric programs in the 60's [Zen61], and they can be applied to many engineering problems including design of circuits and power systems [BKV07, Ben68] and regional analysis. As Dinkel and others write in [DKW77], an entropy maximization problem

in urban and regional analysis,

$$\begin{aligned}
\text{maximize} \quad & -\sum_{(i,j)} \left(t_{ij} \ln\left(K_{ij} t_{ij}\right) - t_{ij}\right) \\
\text{subject to} \quad & \sum_{i=1}^{n} t_{ij} = D_j, \quad j = 1, \ldots, n \\
& \sum_{j=1}^{n} t_{ij} = O_i, \quad i = n+1, \ldots, 2n \\
& \sum_{(i,j)} c_{ij} t_{ij} = C \\
& t_{ij} \geqslant 0, \quad \text{for all } (i,j),
\end{aligned}$$

can be expressed as a geometric program. In the above, t_{ij} is flow from node i to j with associated cost weight c_{ij}, D_j is flow ending at node j, O_i is flow originating from node i, and K_{ij} is the cost of flow from i to j. Assuming that $\sum_j D_j = \sum_i O_i$, we write T to denote this quantity. Using the change of variables $t_{ij} = \delta_{ij} T$, the objective function S can be rewritten as

$$-\sum_{(i,j)} \left(t_{ij} \ln\left(K_{ij} t_{ij}\right) - t_{ij}\right) = T \left(\sum_{(i,j)} \left(\ln\left(T K_{ij} \delta_{ij}\right)^{-\delta_{ij}} + \delta_{ij} \right) \right)$$

and the objective is equivalent to maximizing $\exp(S/T)$, or equivalently

$$\exp(S/T) = \prod_{(i,j)} \left(\frac{e}{T K_{ij} \delta_{ij}} \right)^{\delta_{ij}}.$$

Combining all the above, the problem can be equivalently written as

$$\begin{aligned}
\text{maximize} \quad & \prod_{(i,j)} \left(\frac{e}{TK_{ij}\delta_{ij}} \right)^{\delta_{ij}} \\
\text{subject to} \quad & \sum_i T\delta_{ij} - D_j = 0, \quad j = 1, \ldots, n \\
& \sum_j T\delta_{ij} - O_i = 0, \quad i = n+1, \ldots, 2n \\
& \sum_{(i,j)} c_{ij} T\delta_{ij} - C = 0 \\
& \delta_{ij} \geqslant 0, \quad \text{for all } (i,j) \\
& \sum_{(i,j)} \delta_{ij} = 1
\end{aligned}$$

where the last constraint follows from $\sum_j D_j = \sum_i O_i = T$. This formulation of the entropy maximization problem is a geometric program.

A geometric program can be expressed as conic linear program using the exponential cone as

$$\begin{aligned}
\text{minimize} \quad & c^T x \\
\text{subject to} \quad & \begin{bmatrix} a_{ki}^T x + b_{ki} \\ 1 \\ z_{ki} \end{bmatrix} \in \mathcal{K}_{\exp}, \quad i = 1, \ldots, n_k, \quad k = 1, \ldots, r \\
& \sum_{i=1}^{n_k} z_{ki} \leqslant 1, \quad k = 1, \ldots, r.
\end{aligned}$$

The three-dimensional *power cone* defined as

$$\mathcal{Q}_\alpha = \left\{ (x, y, z) \in \mathbb{R}^3 \;\middle|\; x^\alpha y^{1-\alpha} \geqslant |z| \right\}$$

in [Nes06] for some $\alpha \in (0, 1)$ can model power constraints, for example $|x|^p \leqslant t$ is equivalent to $(t, 1, x) \in \mathcal{Q}_{1/p}$. This is further generalized in [Cha09] to

$$\mathcal{K}_\alpha = \left\{ (x, z) \in \mathbb{R}^n_+ \times \mathbb{R} \;\middle|\; \prod_{i=1}^n x_i^{\alpha_i} \geqslant |z| \right\}$$

with $\alpha \in \mathbb{R}^n_+$ and $\sum_{i=1}^n \alpha_i = 1$. Solvers like Mosek and YALMIP include power cone modeling capabilities, and there is recent interest in developing methods for solving conic linear programs with constraints of these cones.

The trace norm minimization problem presented previously as a semidefinite program can be more simply written as a conic linear program using the trace norm cone, which we define as

$$\mathcal{K}^* = \{(\text{vec}(X), t) \in \mathbb{R}^{m \times n} \times \mathbb{R} \mid \|X\|_* \leqslant t\}.$$

The problem is equivalent to

$$\begin{aligned} &\text{minimize} \quad t \\ &\text{subject to} \begin{bmatrix} \text{vec}(A(x) - B) \\ t \end{bmatrix} \in \mathcal{K}^*. \end{aligned}$$

This formulation may be favorable for computation compared to the SDP formulation because it avoids the increase in number of variables and increased dimension from the Schur complement system.

2.1.5 Modeling Using Conic Linear Programming

Using thoughtfully constructed cones, any convex optimization problem can be formulated as a conic linear program. For a convex set C, the inverse image of C under perspective is

$$P^{-1}(C) = \{(x, y) \in \mathbb{R}^n \times \mathbb{R} \mid y > 0,\ x/y \in C\}.$$

If C additionally has nonempty interior and contains no lines, then $\text{cl}\,(P^{-1}(C))$ is a proper cone. Then the constraint $x \in C$ can equivalently be written as the conic inequality $(x, 1) \in P^{-1}(C)$. For a convex function f, the epigraph of the perspective of f is

$$\text{epi}\,(P_f) = \{(x, y, z) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \mid y > 0,\ yf(x/y) \leqslant z\}.$$

A convex constraint $f(x) \leqslant t$ can be equivalently written as $(x, 1, t) \in \text{epi}\,(P_f)$. Combining the construction of cones from convex sets and cones from convex functions, every convex

optimization problem can be modeled as a conic linear program. While these constructions show that conic linear programming is universal, in practice optimization problems are typically reformulated into programs with more standard cones such as the symmetric cones, the exponential cone, or the power cone.

2.2 Duality

For a cone (and in fact for any set) $\mathcal{K}$, the *dual cone* of $\mathcal{K}$ is defined as

$$\mathcal{K}^* = \{s \mid s^T x \geqslant 0 \quad \forall x \in \mathcal{K}\}.$$

The dual of the conic linear program (2.1) is

$$\begin{array}{ll} \text{maximize} & -b^T y - h^T z \\ \text{subject to} & c + A^T y + G^T z = 0 \\ & z \in \mathcal{K}^*, \end{array} \tag{2.3}$$

and from this related problem we can derive optimality and infeasibility conditions for the primal problem (2.1).

2.2.1 Optimality Conditions

Let $p^\star$ be the optimal value of the primal program (2.1) and $d^\star$ the optimal value of the dual program (2.3). Weak duality always holds, that is, $d^\star \leqslant p^\star$. Strong duality, the condition where $d^\star = p^\star$ holds under stricter conditions. A commonly noted sufficient condition is Slater's condition, which holds when the primal problem is strictly feasible, that is, when there exists a point x that satisfies $Gx \prec h$.

When strong duality holds, the Karush–Kuhn–Tucker (KKT) conditions are sufficient and necessary for optimality. $(x^\star, s^\star, y^\star, z^\star)$ satisfies the KKT conditions when

- Primal feasibility: $Ax^\star = b,\ Gx^\star + s^\star = h,\ s^\star \succeq_{\mathcal{K}} 0$

- Dual feasibility: $A^T y^\star + G^T z^\star + c = 0, z^\star \succeq_{\mathcal{K}^*} 0$
- Complementary slackness: $(z^\star)^T s^\star = 0$.

2.2.2 Theorems of Alternatives

For the constraints of the conic linear program (2.1), the following theorem of alternatives is useful:

Theorem 1 *At most one of the following is feasible:*

- $Ax = b$, $Gx \preceq h$
- $A^T y = 0$, $G^T z = 0$, $z \succeq_* 0$, $b^T y + h^T z < 0$

and at most one of the following is feasible:

- $Ax = 0$, $Gx \preceq h$, $c^T x < 0$
- $A^T y + G^T z + c = 0$, $z \succeq_* 0$.

The system of alternatives, related closely to Farkas' lemma for linear systems, is useful since the alternatives can be used to prove that there does not exist some x with $Ax = b$ and $Gx \preceq h$ or that there does not exist some (y, z) with $A^T y + G^T z + c = 0$, $z \succeq_* 0$, certifying primal or dual infeasibility respectively.

We direct to [Nem04, BN01] as reference materials for theory of optimality, feasibility, and duality in convex optimization.

2.3 Self-Dual Embedding

For a conic linear program, a *self-dual embedding* is a larger program that embeds the original primal and dual programs. Forming the embedding is an elegant and successful approach

to handling initialization and infeasibility detection. The self-dual embedding is primal and dual feasible with known feasible points and known optimal value of zero. So, solving the embedding does not require a phase I procedure or an infeasibility detection procedure, and the self-dual property simplifies termination criteria for a solver. From the solution to a self-dual embedding we can usually recover an optimal vector or a certificate of infeasibility.

2.3.1 Homogeneous Self-Dual Embedding

Goldman and Tucker in [GT56] consider dual linear programs of the form

$$\begin{array}{lll} \text{maximize} & c^T x & \qquad \text{minimize} \quad b^T u \\ \text{subject to} & Ax \leqslant b & \qquad \text{subject to} \quad A^T u \geqslant c \\ & x \geqslant 0 & \qquad\qquad\quad u \geqslant 0 \end{array}$$

and introduce the self-dual feasibility problem

$$\begin{array}{ll} \text{find} & (u, x, t) \\ \text{subject to} & \begin{bmatrix} 0 & -A & b \\ A^T & 0 & -c \\ -b^T & c^T & 0 \end{bmatrix} \begin{bmatrix} u \\ x \\ t \end{bmatrix} \geqslant 0 \\ & (u, x, t) \geqslant 0 \end{array}$$

that encodes the KKT conditions. By a result Tucker states in [Tuc56], the above system has a nonzero solution (u_0, x_0, t_0) such that

$$\begin{bmatrix} 0 & -A & b \\ A^T & 0 & -c \\ -b^T & c^T & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ x_0 \\ t_0 \end{bmatrix} + \begin{bmatrix} u_0 \\ x_0 \\ t_0 \end{bmatrix} > 0.$$

If $t_0 > 0$, then u_0/t_0 and x_0/t_0 are dual and primal optimal vectors respectively. In the case where $t_0 = 0$, either the primal or dual program has no feasible vector, and if a program has

a feasible vector its set of feasible vectors is unbounded; additionally, neither program has an optimal vector.

Recall the conic linear program of form (2.1):

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & Gx \preceq_{\mathcal{K}} h. \end{aligned}$$

The embedding of Goldman and Tucker can be extended to problems of this form:

$$\begin{aligned} \text{minimize} \quad & 0 \\ \text{subject to} \quad & \begin{bmatrix} 0 \\ 0 \\ s \\ \kappa \end{bmatrix} = \begin{bmatrix} 0 & A^T & G^T & c \\ -A & 0 & 0 & b \\ -G & 0 & 0 & h \\ -c^T & -b^T & -h^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix} \\ & (s, \kappa, z, \tau) \succeq 0 \end{aligned} \tag{2.4}$$

where $(s, \kappa, z, \tau) \succeq 0$ is shorthand for $s \succeq_{\mathcal{K}} 0,\ \kappa \geqslant 0,\ z \succeq_{\mathcal{K}^*} 0,\ \tau \geqslant 0$. This embedding is studied by Ye, Todd, and Mizuno in [YTM94], where they detail interior-point algorithms for solving it. O'Donoghue et. al. in [OCP16] analyze solving the same self-dual formulation using the alternating direction method of multipliers, and they implement this method in Splitting Conic Solver (SCS) [OCP23].

This embedding is feasible since $(s, \kappa, x, y, z, \tau) = 0$ is a feasible point. Since the objective function is not dependent on the variables, any feasible point is optimal. The embedding is called homogeneous since for any feasible $(s, \kappa, x, y, z, \tau)$, $(ts, t\kappa, tx, ty, tz, t\tau)$ for any $t \geqslant 0$ is also feasible.

The equality constraint implies that for any feasible $(s, \kappa, x, y, z, \tau)$,

$$\begin{aligned} s^T z + \kappa\tau &= \begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix}^T \begin{bmatrix} 0 \\ 0 \\ s \\ \kappa \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix}^T \begin{bmatrix} 0 & A^T & G^T & c \\ -A & 0 & 0 & b \\ -G & 0 & 0 & h \\ -c^T & -b^T & -h^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix} \\ &= -\left(\begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix}^T \begin{bmatrix} 0 & A^T & G^T & c \\ -A & 0 & 0 & b \\ -G & 0 & 0 & h \\ -c^T & -b^T & -h^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix} \right)^T \\ &= -(s^T z + \kappa\tau) \end{aligned}$$

and so

$$s^T z + \kappa\tau = 0$$

at every feasible point, and so there is no strictly feasible point.

Suppose $(s, \kappa, x, y, z, \tau)$ is a solution to the homogeneous self-dual embedding with either κ or τ nonzero. Since $(s, \kappa, z, \tau) \succeq 0$ and $s^T z + \kappa\tau = 0$, then exactly one of κ and τ is positive and the other is zero.

- In the case $\tau > 0$, $\kappa = 0$, $(\hat{x}, \hat{y}, \hat{z}) = (x, y, z)/\tau$ and $\hat{s} = s/\tau$ are such that

  $$\begin{bmatrix} 0 \\ 0 \\ \hat{s} \end{bmatrix} = \begin{bmatrix} 0 & A^T & G^T \\ -A & 0 & 0 \\ -G & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} + \begin{bmatrix} c \\ b \\ h \end{bmatrix}, \qquad (\hat{s}, \hat{z}) \succeq 0, \qquad \hat{z}^T \hat{s} = 0$$

  so $\hat{x}$ and $(\hat{y}, \hat{z})$ and primal and dual optimal.

- In the case $\tau = 0$, $\kappa > 0$, $c^T x + b^T y + h^T z < 0$ so $c^T x$ is negative or $b^T y + h^T z$ is negative. If $c^T x < 0$ then the dual is infeasible since

  $$Gx + s = 0, \qquad Ax = 0, \qquad s \succeq 0, \qquad c^T x < 0.$$

If $b^T y + h^T z < 0$ then the primal is infeasible since

$$A^T y + G^T z = 0, \qquad z \succeq 0, \qquad b^T y + h^T z < 0.$$

The algorithm described in [OCP16] chooses the initial point specifically to avoid convergence to the meaningless $(s, \kappa, x, y, z, \tau) = 0$ solution.

If the found solution is such that $\tau = \kappa = 0$, then neither an optimal solution nor certificate of infeasibility for the original problem can be found.

2.3.2 Extended Self-Dual Embedding

Another self-dual linear program that encodes the KKT conditions can be defined as

$$\begin{aligned} \text{minimize} \quad & (m+1)\theta \\ \text{subject to} \quad & \begin{bmatrix} 0 \\ 0 \\ s \\ \kappa \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & A^T & G^T & c & q_x \\ -A & 0 & 0 & b & q_y \\ -G & 0 & 0 & h & q_z \\ -c^T & -b^T & -h^T & 0 & q_\tau \\ -q_x^T & -q_y^T & -q_z^T & -q_\tau & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ m+1 \end{bmatrix} \\ & (s, \kappa, z, \tau) \succeq 0 \end{aligned} \tag{2.5}$$

which has the advantage of being strictly feasible, unlike the homogeneous self-dual embedding (2.4). We choose m as the degree of the cone $\mathcal{K}$, which is n for $\mathcal{K} = \mathbb{R}^n_+$, p for $\mathcal{K} = \mathcal{S}^p$, and 1 for $\mathcal{K}$ a norm cone.

The parameters (q_x, q_y, q_z, q_τ) are chosen to be

$$\begin{bmatrix} q_x \\ q_y \\ q_z \\ q_\tau \end{bmatrix} = \frac{m+1}{s_0^T z_0 + 1} \left(\begin{bmatrix} 0 \\ 0 \\ s_0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 & A^T & G^T & c \\ -A & 0 & 0 & b \\ -G & 0 & 0 & h \\ -c^T & -b^T & -h^T & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \right) \tag{2.6}$$

with x_0, s_0, y_0, z_0 chosen arbitrarily with $s_0 \succ_{\mathcal{K}} 0$ and $z_0 \succ_{\mathcal{K}^*} 0$.

The extended self-dual embedding program is always strictly feasible with

$$(s, \kappa, x, y, z, \tau, \theta) = \left(s_0, 1, x_0, y_0, z_0, 1, \frac{s_0^T z_0 + 1}{m + 1}\right)$$

a strictly feasible point.

Similarly to the case of (2.4), the conic linear program (2.5) is self-dual and its solution can give optimal vectors of certificates of infeasibility for the original program if $\tau > 0$ and $\kappa = 0$ or $\tau = 0$ and $\kappa > 0$ respectively.

2.3.3 $\tau = \kappa = 0$ Pathology

A solution to the homogeneous self-dual embedding (2.4) or extended self-dual embedding (2.5) with $\tau = \kappa = 0$ is undesirable because it gives no information about optimal solutions or feasibility of the original problem. Permenter et al. in [PFA17] describe an algorithm using *facial reduction* that can be approximately implemented that can derive an objective vector, improving ray, or certificate of infeasibility when solutions to the homogeneous self-dual embedding with $\tau = \kappa = 0$ are encountered. They also note that if $\tau = \kappa = 0$ for all solutions of the embedding, then it indicates that the primal or dual problem is ill-posed, or poorly conditioned as described in [Ren95].

Facial reduction for conic programming was first introduced by Borwein and Wolkowicz in [BW81]. The procedure is used to replace the cone of the program with a new nonempty cone of smaller dimension that is a subset of the original cone and still contains the optimal vector. We refer to the work [DW17] as a survey and reference for theory and recent work in facial reduction.

For dual conic linear programs of the form

$$\begin{array}{rlrl} \text{minimize} & \langle c, x\rangle & \text{maximize} & \langle b, y\rangle \\ \text{subject to} & Ax = b & \text{subject to} & c - A^*y = s \\ & x \in \mathcal{C} & & s \in \mathcal{C}^* \end{array}$$

the facial reduction procedure can be used to replace $\mathcal{C}$ with a smaller cone that is a subset of $\mathcal{C}$. Permenter and others in [PFA17] show that when a strict solution to (2.4) neither provides an optimal solution nor a certificate of infeasibility, it provides a *facial reduction certificate*. s is a facial reduction certificate for the primal program if $s^\perp$ contains the set $\{x \mid Ax = b\}$ and $\mathcal{C} \cap s^\perp \subset \mathcal{C}$ strictly. Here, $(\cdot)^\perp$ is the set of all orthogonal vectors. Analogously, x is a facial reduction certificate for the dual program if $x^\perp$ contains $\{c - A^*y \mid y \in \mathbb{R}^m\}$ and $\mathcal{C}^* \cap x^\perp \subset \mathcal{C}^*$.

[PFA17] describe that s is a facial reduction certificate for the primal program if there exists y with

$$\langle b, y\rangle = 0, \qquad A^*y + s = 0, \qquad s \in \mathcal{C}^* \backslash \mathcal{C}^\perp,$$

and x is a facial reduction certificate for the dual program if

$$\langle c, x\rangle = 0, \qquad Ax = 0, \qquad x \in \mathcal{C} \backslash (\mathcal{C}^*)^\perp.$$

Using facial reduction, Permenter and others describe an algorithm that yields a meaningful solution to a self-dual embedding even in the case where $\tau = \kappa = 0$. Firstly, write $\mathbf{H}(\mathcal{C})$ to be the set of solutions (x, s, y, τ, κ) to the homogeneous self-dual system

$$\begin{aligned} Ax - b\tau &= 0 \\ -A^*y - s + \tau c &= 0 \\ \langle b, y\rangle - \langle c, x\rangle - \kappa &= 0 \\ (x, s, \tau, \kappa) &\succeq_{\mathcal{K}} 0. \end{aligned}$$

where $\mathcal{K} = \mathcal{C} \times \mathcal{C}^* \times \mathbb{R}_+ \times \mathbb{R}_+$ The algorithm is

---

$\mathcal{C} \leftarrow \mathcal{K}$
repeat
    Find (x, s, y, τ, κ) in the relative interior of $\mathbf{H}(\mathcal{C})$
    if $\tau + \kappa > 0$ **then**
        return (x, s, y, τ, κ)
    else
        if $s \notin \mathcal{C}^\perp$ **then**
            $\mathcal{C} \leftarrow \mathcal{C} \cap s^\perp$
        else
            $\mathcal{C} \leftarrow \left(\mathcal{C}^* \cap x^\perp\right)^*$
        end if
    end if
until algorithm returns

The algorithm returns, in finitely many iterations, a solution (x, s, y, τ, κ) that satisfies $\tau > 0$ or $\kappa > 0$. The update of replacing the cone $\mathcal{C}$ with $\mathcal{C} \cap s^\perp$ replaces the conic program to one with a reduced cone that still contains the optimal vector.

2.4 Algorithms

2.4.1 Interior-Point Methods

Interior-point methods have been a popular method of solving conic linear programs, and is used in solvers including Mosek, SeDuMi, and SDPT3 [ApS19, Stu99, TTT99]. There has been extensive research on these methods since the 1990's, and they have been successful in solving medium-scale problems efficiently. These methods repeatedly solve a linearization of the KKT equations of an approximate problem where the inequality constraints are replaced

instead with a barrier function. While these methods generally are robust and converge in few iterations, there is difficulty in scaling these methods for larger problems since each iteration solves a linear system. The cost of solving the linear system is especially expensive since there is often a need to embed the problem into some larger problem and the linear system is dense or has dense factorizations in general, so the cost per iteration grows quickly.

We refer to [NN94, Wri97] for further readings on interior-point methods.

2.4.2 First-Order Methods

Recent focus has been on *first-order methods*, which do not require solving a linear system. Since the computational cost is dominated by matrix-vector multiplications and projection or proximal operations, which are frequently efficient to compute, these methods are promising for large-scale conic linear programs, where factoring a data matrix would be very costly.

Firstly, for convex f, the *proximal operator* prox_f is defined

$$\text{prox}_f(x) = \underset{y}{\text{argmin}} \left(f(y) + \frac{1}{2} \|y - x\|_2^2 \right)$$

and the *convex conjugate* f^* is defined

$$f^*(y) = \sup_{x \in \text{dom}\, f} \left(y^T x - f(x) \right).$$

We also note that proximal operators extend projections. The proximal operator of the indicator function of a nonempty set C,

$$\delta_C(x) = \begin{cases} 0, & x \in C \\ \infty, & x \notin C \end{cases}$$

is

$$\text{prox}_{\delta_C}(x) = \underset{y}{\text{argmin}} \left(\delta_C(y) + \frac{1}{2} \|y - x\|_2^2 \right) = \underset{y \in C}{\text{argmin}} \|x - y\|_2$$

which is identically the Euclidean projection onto C, which we denote with Π_C.

The *Douglas-Rachford* algorithm, first introduced in 1956 in [DR56] for numerically solving the heat equation, solves problems of the form

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

with f and g convex. The algorithm updates as

$$\begin{aligned} x_{k+1} &= \text{prox}_f(y_k) \\ y_{k+1} &= y_k + \text{prox}_g(2x_{k+1} - y_k) - x_{k+1} \end{aligned}$$

with any y_0. If g additionally has conjugate g^* with easily computable proximal operator, the updates can be simplified using Moreau decomposition and change of variables $z = x - y$ to

$$\begin{aligned} x_{k+1} &= \text{prox}_f(x_k - z_k) \\ z_{k+1} &= \text{prox}_{g^*}(z_k + 2x_{k+1} - x_k) \end{aligned}$$

with any x_0, z_0.

The *alternating direction method of multipliers* (ADMM) is a choice first-order method, equivalent to applying the Douglas-Rachford method to the dual problem as shown in [Gab83]. ADMM updates as

$$\begin{aligned} x_{k+1} &= \text{prox}_f(z_k + \lambda_k) \\ z_{k+1} &= \text{prox}_g(\lambda_k - x_{k+1}) \\ \lambda_{k+1} &= \lambda_k - x_{k+1} + z_{k+1} \end{aligned}$$

We refer to [BPC11] for a survey on ADMM and its applications and to [ZFP20] and [OCP16, OCP23] as some examples modern solvers that use ADMM.

In this work, we examine two first-order methods: the *primal-dual hybrid gradient* and *extragradient* methods; we defer to chapters 3 and 4 respectively for details on these methods.

CHAPTER 3

Primal-Dual Hybrid Gradient Method

The primal-dual hybrid gradient (PDHG) method was first introduced by Zhu and Chan in [ZC08], where they present a primal-dual descent algorithm and apply it to total variation minimization for image denoising tasks. Esser, Zhang, and Chan in [EZC10] generalize the algorithm to a broader class of convex optimization problems and survey related methods. Chambolle and Pock in [CP11] also study the algorithm and analyze the rate of convergence. Pock and others in [PCB09] use PDHG to minimize the Mumford-Shah functional for image segmentation tasks. We refer to [CP16] for further readings on optimization algorithms and their applications in imaging.

For a minimization problem

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

with equivalent saddle-point formulation

$$\min_x \max_y \left(\langle Ax, y \rangle + f(x) - g^*(y) \right)$$

with f and g^* proper, convex, lower semicontinuous functions, the PDHG algorithm is given as

$$\begin{aligned} y_{k+1} &= \operatorname{prox}_{\sigma g^*} \left(y_k + \sigma A \bar{x}_k \right) \\ x_{k+1} &= \operatorname{prox}_{\tau f} \left(x_k - \tau A^* y_{k+1} \right) \\ \bar{x}_{k+1} &= x_{k+1} + \theta \left(x_{k+1} - x_k \right) \end{aligned} \tag{3.1}$$

with primal and dual step sizes $\tau, \sigma > 0$ satisfying $\sigma\tau \left\| A \right\|_2^2 \leqslant 1$, $\theta \in [0, 1]$ and initial points x_0, y_0, and $\bar{x}_0 = x_0$. With $\theta = 0$ the method is equivalent to the Arrow-Hurwicz algorithm, and

with $\theta = 1$ and $A = I$ the method is the Douglas-Rachford splitting algorithm. Chambolle and Pock in [CP11] and [CP15] show that the method converges with rate $O(1/N)$.

3.1 PDHG for Conic Linear Programming

Now, consider the application of PDHG to the conic linear program (2.1). For ease of notation, we write

$$B = \begin{bmatrix} A \\ G \end{bmatrix}, \quad q = \begin{bmatrix} b \\ h \end{bmatrix}.$$

The conic linear program (2.1) can be written as

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

with

$$f(x) = c^T x, \qquad g(y) = \delta_{\{0\}^{m_1} \times (-\mathcal{K})}(y - q).$$

The conjugate of g is

$$g^*(y) = q^T y + \delta_{\mathbb{R}^{m_1} \times \mathcal{K}^*}(y).$$

Recalling that the proximal operator of the indicator function of a set is the Euclidean projection onto that set, the updates in PDHG (3.1) simplify to

$$\begin{aligned} y_{k+1} &= \Pi_{\mathbb{R}^{m_1} \times \mathcal{K}^*} \left(y_k - \sigma \left(q - B\bar{x}_k \right) \right) \\ x_{k+1} &= x_k - \tau \left(c - B^T y_{k+1} \right) \\ \bar{x}_{k+1} &= x_{k+1} + \theta(x_{k+1} - x_k). \end{aligned}$$

Here, we again write Π_S to denote the Euclidean projection operator

$$\Pi_S(x) = \operatorname*{argmin}_{s \in S} \|x - s\|_2.$$

For the case where the conic linear program is written to have constraints $Ax = b$, $Gx \preceq_{\mathcal{K}} h$, $x \in C$ for some convex C, then the primal iteration can be expressed

$$x_{k+1} = \Pi_C \left(x_k - \tau \left(c - B^T y_{k+1} \right) \right)$$

with $f(x) = c^T x + \delta_C(x)$, which may be useful for programs where the primal variable is constrained to a convex cone or other convex set.

PDHG is promising for large scale conic linear programming because of its low per-iteration complexity; each iteration of PDHG consists of evaluating matrix-vector products and projections onto cones. Unlike matrix factorization or solution of linear systems, these operations scale well to larger dimensions.

3.2 PDHG for Linear Programming

Applegate et al. in [ADH21] introduce PDLP (PDHG for LP), a method for solving linear programs using PDHG equipped with a suite of practical enhancements. These include adaptive step size updates, adaptive restarts, and dynamic primal weight updates. While first-order methods frequently quickly converge to an accurate solution, progress towards a precise solution often slows. Applegate and others show that equipped with the algorithmic enhancements, PDLP is able to efficiently find precise solutions to linear programs.

For a linear program of the form

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & Gx \leqslant h \\ & l \leqslant x \leqslant u \end{aligned}$$

with $l \in (\mathbb{R} \cup \{-\infty\})^n$ and $u \in (\mathbb{R} \cup \{\infty\})^n$, the PDHG iterates with $\theta = 1$ can be written as

$$\begin{aligned} x_{k+1} &= \Pi_X \left(x_k - \tau \left(c - B^T y_k \right) \right) \\ y_{k+1} &= \Pi_Y \left(y_k - \sigma \left(q - B(2x_{k+1} - x_k) \right) \right) \end{aligned}$$

with $X = \{x \in \mathbb{R}^n \mid l \leqslant x \leqslant u\}$ and $Y = \mathbb{R}^{m_1} \times \mathbb{R}^{m_2}_+$. PDLP parameterizes the primal step size and dual step size as $\tau = \eta/\omega$ and $\sigma = \eta\omega$ with *primal weight* $\omega > 0$ and *step size* $\eta > 0$.

Then, the method is known to converge with $\eta \leqslant 1/\|B\|_2$. For $\omega > 0$, the weighted norm of primal-dual pair (x_k, y_k) is defined as

$$\|(x_k, y_k)\|_\omega = \sqrt{\omega\|x_k\|_2^2 + \frac{\|y_k\|_2^2}{\omega}}.$$

The algorithm is described as

$n \leftarrow 0$, $k \leftarrow 0$, $\eta \leftarrow 1/\|K\|_\infty$, $\omega_0 \leftarrow \texttt{InitializePrimalWeight}(c, q)$
repeat
  $t \leftarrow 0$
  repeat
    $z^{n,t+1}, \eta^{n,t+1}, \hat{\eta}^{n,t+1} \leftarrow \texttt{AdaptiveStepOfPDHG}(z^{n,t}, \omega_n, \hat{\eta}^{n,t}, k)$
    $\hat{z}^{n,t+1} \leftarrow \frac{1}{\sum_{i=1}^{t+1}\eta^{n,i}} \sum_{i=1}^{t+1} \eta^{n,i} z^{n,i}$
    $z_c^{n,t+1} \leftarrow \texttt{GetRestartCandidate}(z^{n,t+1}, \bar{z}^{n,t+1}, z^{n,0})$
    $t \leftarrow t + 1$, $k \leftarrow k + 1$
  until restart conditions or termination conditions met
  $z^{n+1,0} \leftarrow z_c^{n,t}$, $n \leftarrow n + 1$
  $\omega_n \leftarrow \texttt{PrimalWeightUpdate}(z^{n,0}, z^{n-1,0}, \omega_{n-1})$
until termination criteria hold
return $z^{n,0}$.

In the above, t counts the number of iterations of the inner loop, which is the number of iterations since the last restart; k counts the total number of iterations, and n counts the number of times the inner loop is run, which counts the number of restarts. z denotes the primal-dual iterate (x, y). In the rest of this section, we summarize the practical algorithmic improvements implemented in PDLP.

3.2.1 Adaptive Step Size Choice

The adaptive step size update is implemented in `AdaptiveStepOfPDHG` by repeatedly performing the proximal operations and accepting iterates x_{k+1} and y_{k+1} when

$$\eta \leqslant \frac{\|(x_{k+1} - x_k, y_{k+1} - y_k)\|_\omega^2}{2(y_{k+1} - y_k)^T B(x_{k+1} - x_k)}$$

and decreasing η otherwise. This stepping method finds a step size small enough to guarantee convergence, and it is more liberal than setting $\eta = 1/\|B\|_2$ and additionally does not require computing $\|B\|_2$.

3.2.2 Adaptive Restarts

In each outer iteration of PDLP, the algorithm is restarted according to the *normalized duality gap* as introduced in [AHL22], defined as

$$\rho_r^n(x, y) = \frac{1}{r} \max_{\substack{(\hat{x},\hat{y}) \in X \times Y \\ \|(\hat{x},\hat{y}) - (x,y)\|_{\omega_n} \leqslant r}} \left(\mathcal{L}(x, \hat{y}) - \mathcal{L}(\hat{x}, y)\right),$$

which is always finite. Also defined is $\mu_n(z, z_{\text{ref}}) = \rho^n_{\|z - z_{\text{ref}}\|_{\omega_n}}(z)$ for some reference point z_{ref}. For parameters $\beta_{\text{sufficient}} \in (0, 1)$, $\beta_{\text{necessary}} \in (0, \beta_{\text{sufficient}})$, and $\beta_{\text{artificial}} \in (0, 1)$, the algorithm is restarted if one of the conditions is met:

- $\mu_n(z_c^{n,t+1}, z^{n,0}) \leqslant \beta_{\text{sufficient}} \mu_n(z^{n,0}, z^{n-1,0})$
- $\mu_n(z_c^{n,t+1}, z^{n,0}) \leqslant \beta_{\text{necessary}} \mu_n(z^{n,0}, z^{n-1,0})$ and $\mu_n(z_c^{n,t+1}, z^{n,0}) > \mu_n(z_c^{n,t}, z^{n,0})$
- $t \geqslant \beta_{\text{artificial}} k$

where n counts the iterations of the outer loop, t counts the iterations of the inner loop, and k is the total number of iterations; $z^{n,t}$ is the iterate after the tth iteration of the inner loop, of the nth iteration of the outer loop. $z_c^{n,t+1}$ is the restart candidate after updating $z^{n,t}$, and

it is chosen by `GetRestartCandidate` as

$$z_c^{n,t+1} = \begin{cases} z^{n,t+1}, & \mu_n(z^{n,t+1}, z^{n,0}) < \mu_n(\bar{z}^{n,t+1}, z^{n,0}) \\ \bar{z}^{n,t+1}, & \text{otherwise} \end{cases}$$

where $\bar{z}^{n,t+1}$ is the average iterate in the nth outer loop after t iterations of the inner loop, weighted by the step sizes of the updates, that is

$$\bar{z}^{n,t+1} = \frac{\sum_{i=1}^{t+1} \eta^{n,i} z^{n,1}}{\sum_{i=1}^{t+1} \eta^{n,i}}.$$

In other words, the restart candidate is selected as the current iterate or the average iterate over the inner loop, whichever results in a lower normalized duality gap with the iterate at the beginning of the inner loop as the reference point.

3.2.3 Primal Weight Updates

The primal weight is initialized in `InitializePrimalWeight` to be

$$\omega_0 = \begin{cases} \frac{\|c\|_2}{\|q\|_2}, & \|c\|_2 > 0 \text{ and } \|q\|_2 > 0 \\ 1, & \text{otherwise} \end{cases}.$$

The primal weight is updated so that the distance to optimal solutions in the primal and dual tend to equality, that is, $\|(x^{n,t} - x^\star, 0)\|_{\omega_n} = \|(0, y^{n,t} - y^\star)\|_{\omega_n}$, which would be at

$$\omega_n \|x^{n,t} - x^\star\|_2^2 = \frac{1}{\omega_n} \|y^{n,t} - y^\star\|_2^2.$$

Since $x^\star$ and $y^\star$ are unknown, `PrimalWeightUpdate` instead estimates

$$\omega_n = \|y^{n,t} - y^\star\|_2 / \|x^{n,t} - x^\star\|_2$$

as $\|y^{n,0} - y^{n-1,0}\|_2 / \|x^{n,0} - x^{n-1,0}\|_2$.

3.3 Infeasibility Detection

Applegate et al. in [ADL21] Theorem 4 describe the behavior of PDHG for linear programs and how the iterates can yield a certificate of infeasibility. They consider a linear program in standard form

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geqslant 0 \end{aligned}$$

with operator T the PDHG update, $T(x_k, y_k) = (x_{k+1}, y_{k+1})$, with $\sigma\tau\|A\|_2^2 < 1$. They show that T is *firmly nonexpansive*

$$\|T(z_1) - T(z_2)\|_M^2 \leqslant \|z_1 - z_2\|_M^2 - \|(T-I)(z_1) - (T-I)(z_2)\|_M^2 \quad \forall z_1, z_2 \in \mathbb{R}^{n+m}$$

with respect to weighted norm $\|\cdot\|_M$ with M defined as

$$M = \begin{bmatrix} \frac{1}{\tau} I_n & -A^T \\ -A & \frac{1}{\sigma} I_{m_1} \end{bmatrix}.$$

Then, one of the following holds for the fixed-point iteration of T:

- If the primal and dual programs are feasible, then (x_k, y_k) converges to optimal $(x^\star, y^\star)$ and $(T-I)(x^\star, y^\star) = 0$.
- If the primal and dual programs are infeasible, then the *infimal displacement vector* $(v_x, v_y) = \operatorname{argmin}_{z \in \mathrm{cl}(\mathrm{range}(T-I))} \frac{1}{2}\|z\|_M^2$ gives certificates of infeasibility v_x and v_y for the dual and primal programs respectively.
- If the primal is infeasible and the dual is feasible, then y_k diverges to infinity and x_k converges to $x^\star$, v_y is a certificate of primal infeasibility, and there exists some $y^\star$ such that $(v_x, v_y) = (T-I)(x^\star, y^\star)$.

- If the dual is infeasible and the primal is feasible, then the analogous conclusion to the previous holds with the primal and dual components swapped.

For this work, we solve the extended self-dual embedding, which is strictly feasible, circumventing the need for an infeasibility detection method like that described above. In our alternate approach to detecting infeasibility, we solve the extended-self dual embedding and if the solution is such that $\kappa > 0$, we conclude the problem is infeasible and recover a certificate of primal or dual infeasibility.

3.4 PDHG for Extended Self-Dual Embedding

In our implementation, we consider PDHG for conic linear programs, augmented with the same practical improvements that Applegate and others describe in [ADH21]. We apply this algorithm to the extended-self dual embedding (2.5). Recall the embedding

$$\begin{aligned}
&\text{minimize} \quad (m+1)\theta \\
&\text{subject to} \quad \begin{bmatrix} 0 \\ 0 \\ s \\ \kappa \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & A^T & G^T & c & q_x \\ -A & 0 & 0 & b & q_y \\ -G & 0 & 0 & h & q_z \\ -c^T & -b^T & -h^T & 0 & q_\tau \\ -q_x^T & -q_y^T & -q_z^T & -q_\tau & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \\ \theta \end{bmatrix} + \begin{bmatrix} `0 \\ 0 \\ 0 \\ 0 \\ m+1 \end{bmatrix} \\
&\qquad (s, \kappa, z, \tau) \geq 0.
\end{aligned}$$

Now writing $c = (0, \ldots, 0, m+1)$ and Q the skew-symmetric constraint matrix, the updates can be written as

$$\begin{aligned}
v_{k+1} &= \Pi_{\mathcal{C}} \left(v_k - \sigma \left(c + Q \bar{u}_k \right) \right) \\
u_{k+1} &= \Pi_{\mathcal{C}} \left(x_k - \tau \left(c + Q^T v_{k+1} \right) \right) \\
\bar{u}_{k+1} &= u_{k+1} + \theta (u_{k+1} - u_k)
\end{aligned}$$

with u being the primal variable and v being the dual variable. $\mathcal{C}$ is $\mathbb{R}^n \times \mathbb{R}^{m_1} \times \mathcal{K}^* \times \mathbb{R}_+ \times \mathbb{R}$, so the projection onto $\mathcal{C}$ projects the z component to the dual of $\mathcal{K}$ and the τ component to the nonnegative numbers. In other words,

$$\Pi_{\mathcal{C}}(x, y, z, \tau, \theta) = \left(x, y, \Pi_{\mathcal{K}^*}(z), \tau^+, \theta\right).$$

CHAPTER 4

Extragradient Method for Self-Dual Programs

Consider a pair of primal and dual optimization problems

$$\text{minimize} \quad f(x) + g(Ax) \qquad \text{maximize} \quad -g^*(z) - f^*(-A^*z)$$

with f and g closed convex functions. We compare extensions of the *extragradient method* [Kor77], described in [AT05, JN12, Nem04, Tse08], for solving the saddle point formulation

$$\min_x \max_y \left(\langle Ax, y\rangle + f(x) - g^*(y)\right).$$

This saddle point problem is also formulated as a *variational inequality*, that is, the problem of finding $(x^\star, y^\star)$ such that

$$f(x) - f(x^\star) + \langle x - x^\star, A^*y^\star\rangle \geqslant 0, \quad \forall x, \qquad -g^*(y) + g^*(y^\star) + \langle y - y^\star, Ax^\star\rangle \leqslant 0, \quad \forall y.$$

The iterates of the extragradient method are given as:

$$\begin{aligned} \bar{x}_k &= \text{prox}_{t_k f}(x_k - t_k A^* z_k) \\ \bar{z}_k &= \text{prox}_{t_k g^*}(z_k + t_k A x_k) \\ x_{k+1} &= \text{prox}_{t_k f}(x_k - t_k A^* \bar{z}_k) \\ z_{k+1} &= \text{prox}_{t_k g^*}(z_k + t_k A \bar{x}_k) \end{aligned} \tag{4.1}$$

4.1 Extragradient Method for Self-Dual Problems

For a self-dual program, with $f^* = g$ and $A = -A^*$, the primal and dual programs are equivalent:

$$\text{minimize} \quad f(x) + f^*(Ax) \qquad\qquad \text{maximize} \quad -f(z) - f^*(Az). \tag{4.2}$$

Assuming that the problem is strictly feasible, strong duality holds so $p^\star = d^\star$. Additionally, if $p^\star > -\infty$, the dual optimum is attained. Since a feasible point $\hat{x}$ is also dual feasible, the optimal value is bounded below and so $p^\star = d^\star = 0$. The primal-dual optimality conditions

$$0 \in \begin{bmatrix} 0 & A^* \\ -A & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} \partial f(x) \\ \partial g^*(z) \end{bmatrix}$$

reduce to $0 \in -Ax + \partial f(x)$ for the self-dual program.

The associated convex-concave function

$$\mathcal{L}(x, z) = f(x) - g^*(z) + \langle z, Ax \rangle$$

for the self-dual function can be formulated as

$$\mathcal{L}(x, z) = f(x) - f(z) + \langle z, Ax \rangle$$

and so $\mathcal{L}(x, z) = -\mathcal{L}(z, x)$ and $\mathcal{L}(x, x) = 0$.

We apply the extragradient method (4.1) to the self-dual program (4.2), choosing some initialization $x_0 = z_0$. Then, $\bar{x}_k = \bar{z}_k$ and $x_k = z_k$ for all k, so the four updates simplify to two:

$$\bar{x}_k = \text{prox}_{t_k f}(x_k + t_k A x_k)$$
$$x_{k+1} = \text{prox}_{t_k f}(x_k + t_k A \bar{x}_k).$$

The step size selection method by Tseng [Tse08] is to compute $\bar{x}_k$ and x_{k+1} given some positive t_k and accept the iterates if

$$f(x_{k+1}) - \langle A\bar{x}_k, x_{k+1} \rangle + \frac{1}{t_k}\left(\frac{\|x_{k+1} - x_k\|_2^2}{2}\right) \geqslant f(\bar{x}_k)$$

holds. Otherwise, we decrease t_k and recompute the iterates.

4.2 Extragradient Method for Extended Self-Dual Embedding

Recall the self-dual embedding (2.5)

$$\begin{array}{ll} \text{minimize} & (m+1)\theta \\ \text{subject to} & \begin{bmatrix} 0 \\ 0 \\ s \\ \kappa \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & A^T & G^T & c & q_x \\ -A & 0 & 0 & b & q_y \\ -G & 0 & 0 & h & q_z \\ -c^T & -b^T & -h^T & 0 & q_\tau \\ -q_x^T & -q_y^T & -q_z^T & -q_\tau & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ m+1 \end{bmatrix} \\ & (s, \kappa, z, \tau) \geq 0. \end{array} \tag{4.3}$$

The program can be written in self-dual composite form as

$$\text{minimize} \quad f(x, y, z, \tau, \theta) + f^* \left(\begin{bmatrix} 0 & A^T & G^T & c & q_x \\ -A & 0 & 0 & b & q_y \\ -G & 0 & 0 & h & q_z \\ -c^T & -b^T & -h^T & 0 & q_\tau \\ -q_x^T & -q_y^T & -q_z^T & -q_\tau & 0 \end{bmatrix}^T \begin{bmatrix} x \\ y \\ z \\ \tau \\ \theta \end{bmatrix} \right) \tag{4.4}$$

with

$$f(x, y, z, \tau, \theta) = \begin{cases} (m+1)\theta & (z, \tau) \geq 0 \\ \infty & \text{otherwise.} \end{cases}$$

For ease of notation, write $v = (x, y, z, \tau, \theta)$ the optimization variable, $q = (0, \ldots, 0, m+1)$, and Q for the skew-symmetric matrix in the equality constraint. The extragradient algorithm updates can be written as

$$\bar{v}_k = \text{prox}_{t_k f}(v_k - t_k Q v_k)$$

$$v_{k+1} = \text{prox}_{t_k f}(v_k - t_k Q \bar{v}_k)$$

which are

$$\bar{v}_k = \underset{\bar{v}}{\operatorname{argmin}} \left(t_k f(\bar{v}) + \frac{1}{2} \left\| \bar{v} - (v_k - t_k Q v_k) \right\|_2^2 \right)$$
$$v_{k+1} = \underset{v}{\operatorname{argmin}} \left(t_k f(v) + \frac{1}{2} \left\| v - (v_k - t_k Q \bar{v}_k) \right\|_2^2 \right).$$

Solving the minimization problem in the updates gives

$$\bar{v}_k = \Pi_{\mathcal{C}} \left(v_k - t_k Q v_k - t_k q \right)$$
$$v_{k+1} = \Pi_{\mathcal{C}} \left(v_k - t_k Q \bar{v}_k - t_k q \right).$$

Similarly to the PDHG updates, $\mathcal{C} = \mathbb{R}^n \times \mathbb{R}^{m_1} \times \mathcal{K}^* \times \mathbb{R}_+ \times \mathbb{R}$, so

$$\Pi_{\mathcal{C}}(x, y, z, \tau, \theta) = \left(x, y, \Pi_{\mathcal{K}^*}(z), \tau^+, \theta \right).$$

The extragradient method applied to the self-dual embedding is promising for large-scale conic linear programs because, as a first-order method, the most computationally expensive operations of this algorithm are multiplications by Q and projections onto cones, which are relatively inexpensive operations.

CHAPTER 5

Numerical Results

In this chapter, we describe numerical experiments to illustrate the performance of PDHG and the extragradient method. We solve linear programs and non-polyhedral conic linear programs and provide empirical evidence that the algorithms converge and find a correct solution. All experiments are run on a machine with a 4 GHz eight-core AMD FX-8370 with 16 GB of memory.

5.1 Description of Experiments

For these experiments, we use the open-source Julia [BEK17] implementation of PDLP from [ADH21], which we have edited to support non-polyhedral conic linear programs. We also provide an implementation of a modified extragradient method, which solves self-dual programs without performing the redundant dual iterates and supports non-polyhedral conic linear programs.

For these experiments, we solve a conic linear program using the extended self-dual formulation and record the total time spent on the computational operations in the algorithm and the number of *KKT passes*, where each KKT pass is left-multiplication of a vector by (A, G) and one by $(A, G)^T$.

5.1.1 Initialization

For forming the embedding, we choose x_0 and y_0 to be zero vectors. To choose $s_0 \succ_{\mathcal{K}} 0$, we use the following choices:

- For $\mathcal{K} = \mathbb{R}^n_+$ we choose $s_0 = \mathbf{1}$.
- For $\mathcal{K} = \mathcal{S}^n$, we choose $s_0 = \operatorname{vec}(I)$.
- For $\mathcal{K}$ a norm cone, including second-order cones, the trace norm cone, and the operator norm cone, we choose $s_0 = (0, 0, \ldots, 0, 1)$, the vectorization of $(\mathbf{0}, 1)$.

We choose $z_0 \succ_{\mathcal{K}^*} 0$ using the same scheme. With these choices of x_0, s_0, y_0, z_0, we form the extended self-dual embedding by computing (q_x, q_y, q_z, q_τ) according to (2.6) and form the extended self-dual embedding (2.5).

For choosing an initial point, we use the heuristic presented in [Meh92]. Inspired by this, for a conic linear program of the form (2.1), the first iterate is chosen according to the least-norm problem

$$\begin{array}{ll} \text{minimize} & \|x\|_2^2 + \|s\|_2^2 \\ \text{subject to} & Ax = b \\ & Gx + s = h \end{array}$$

and the least-squares problem

$$\text{minimize} \quad \left\|A^T y + G^T z - c\right\|_2^2 + \frac{1}{2}\|z\|_2^2.$$

We choose initial point $(x, y, z, \tau, \theta) = (\tilde{x}, \tilde{y}, \tilde{z}, 1, (\tilde{s}^T \tilde{z} + 1)/(m + 1))$ where $(\tilde{x}, \tilde{s})$ and $(\tilde{y}, \tilde{z})$ are the optimal solutions to the above programs.

We use the above simple methods for choosing strictly feasible points used in forming the embedding and for initializing PDHG and the extragradient method, though we note that there are many other appropriate choices.

5.1.2 Termination Conditions

Since the extended self-dual embedding is self-dual and strictly feasible, there is a feasible vector that attains the optimum. So, we choose termination criteria to be zero duality gap and feasibility, within some small positive tolerance ϵ. For computing distance to feasibility, we compute the residuals

$$r_{\mathrm{e}} = \begin{bmatrix} A^T y + G^T z + \tau c + \theta q_x \\ -Ax + \tau b + \theta q_y \\ -q_x^T x - q_y^T y - q_z^T z - \tau q_\tau \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ m+1 \end{bmatrix}$$

$$r_{\mathrm{i}} = \left(I - \Pi_{\mathcal{K}\times\mathbb{R}_+}\right)\left(\begin{bmatrix} -Gx + \tau h + \theta q_z \\ -c^T x - b^T y - h^T z + \theta q_\tau \end{bmatrix}\right).$$

We note that the constraint $(z, \tau) \geq 0$ is always satisfied since each iteration of both PDHG and the extragradient method projects (z, τ) onto $\mathcal{K}^* \times \mathbb{R}_+$. We implement termination criteria as

$$\begin{aligned} \theta &\leqslant \epsilon && \text{small duality gap} \\ \|(\, r_{\mathrm{e}}, r_{\mathrm{i}}\,)\| &\leqslant (m+1)\epsilon && \text{small distance to feasibility} \end{aligned}$$

and the factor of $(m+1)$ for the feasibility condition is to make the tolerance relative to the scale of the problem. For these experiments, we choose $\epsilon = 1\mathrm{e}{-6}$, and we note that there are other approaches to implementing these termination criteria.

5.2 Convergence Curves

In this section, we show convergence curves to illustrate the performance of PDHG and the extragradient method on the extended self-dual embedding. We run the algorithms on the instance `25fv47` from the Netlib data set, which we describe again in a later section. The problem `25fv47` is a feasible linear program with 1571 variables and 821 constraints. Additionally, the problem constrains the variables to be nonnegative, which we reform as

$-Ix \leqslant 0$. Figure 5.1 and Figure 5.2 depict the absolute objective value and relative residuals respectively of the extragradient method iterates, as the iteration number increases.

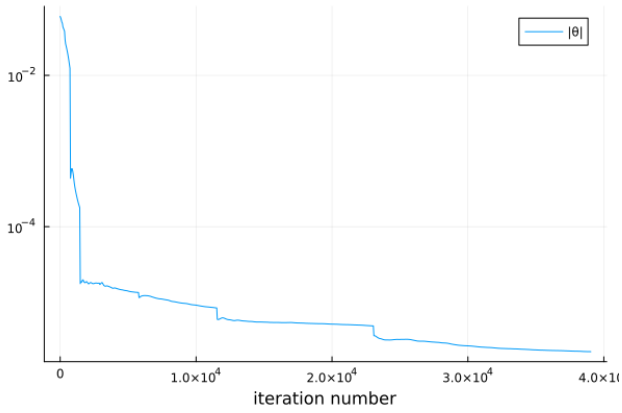

Figure 5.1: Magnitude of primal objective $|\theta|$ of extragradient iterates plotted against iteration number.

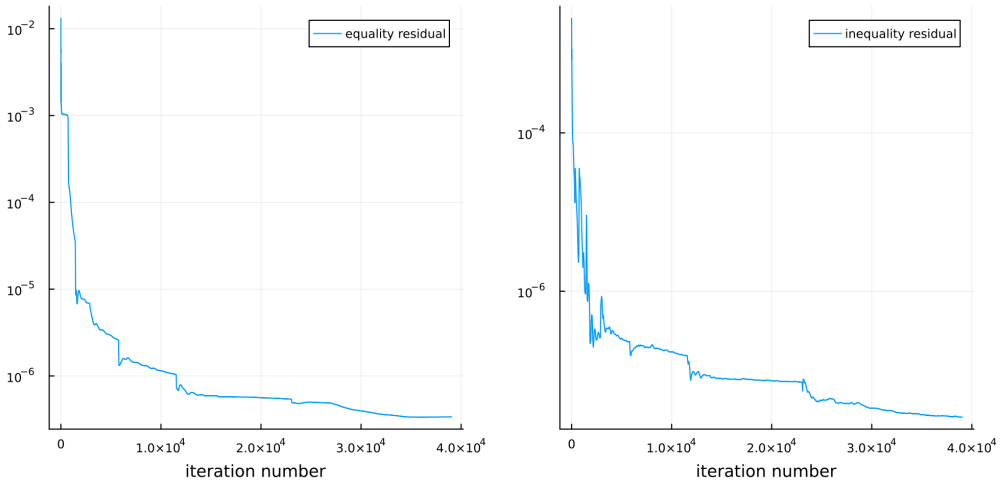

Figure 5.2: Relative residuals $\|r_{\mathrm{e}}\|_\infty/(m+1)$ and $\|r_{\mathrm{i}}\|_\infty/(m+1)$ of extragradient iterates plotted against iteration number.

Similarly, we create the same plots with the PDHG iterates; since PDHG does not perform the same updates for the primal and dual vectors, we record the same metrics for both the

primal and dual iterates in Figure 5.4 and Figure 5.3.

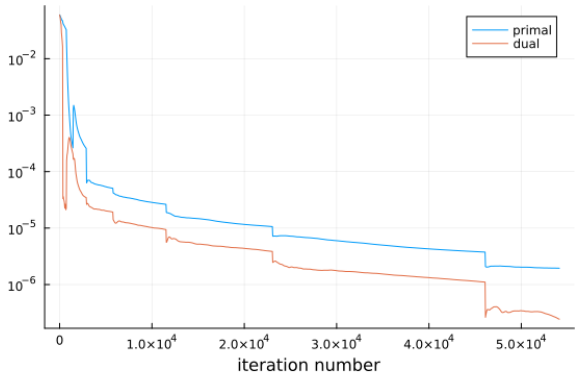

Figure 5.3: Magnitudes of primal and dual objectives $|\theta|$ of PDHG iterates plotted against iteration number.

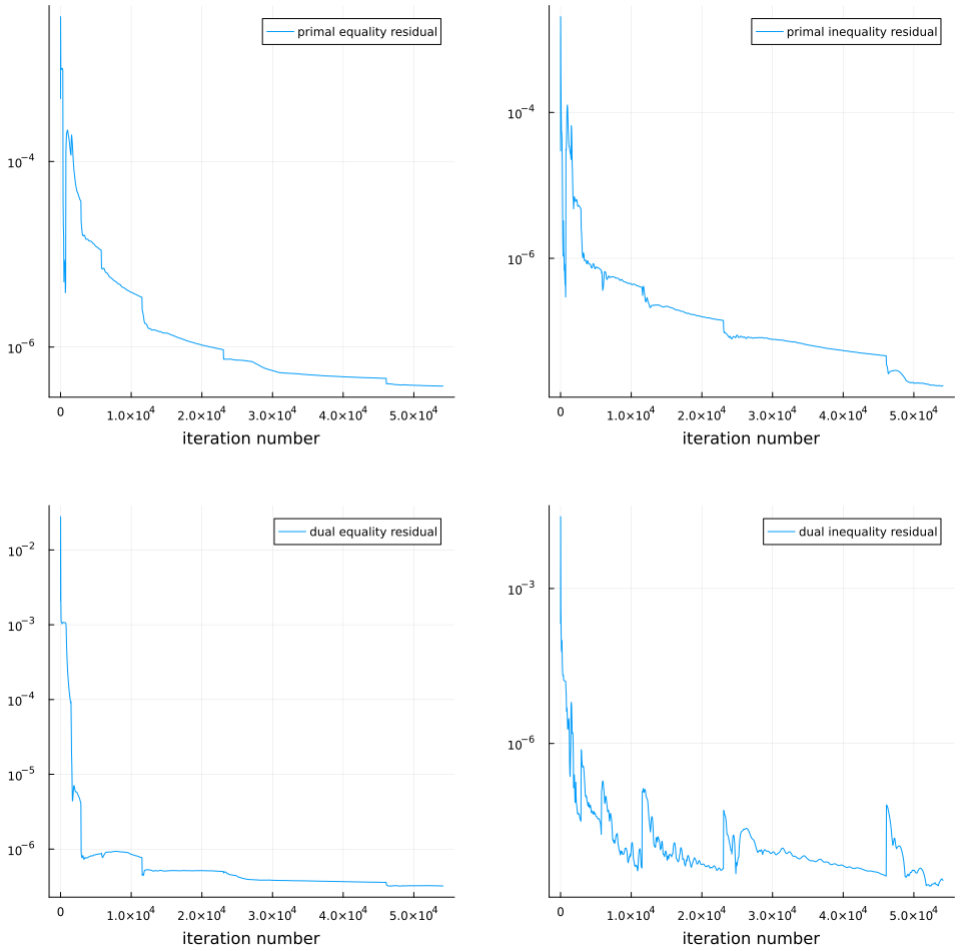

Figure 5.4: Relative residuals $\|r_{\mathrm{e}}\|_\infty/(m+1)$ and $\|r_{\mathrm{i}}\|_\infty/(m+1)$ of the primal (top) and dual (bottom) PDHG iterates plotted against iteration number.

From the previous figures, we observe that the algorithms make fast progress towards a solution in the early iterations, but progress slows. This is indicated by the curves trending towards zero less steeply at larger iteration numbers. This phenomenon is expected of first-order methods. We also note that the curves are not monotonic and that many of the discontinuities correspond to the algorithm triggering a restart.

5.3 Linear Programs

In this section, we measure the performance of PDHG and the extragradient method on linear programs. In addition to the experimental setup described in the previous section, we also then preprocess the problems by applying 10 iterations of Ruiz diagonal rescaling [Rui01] as also used in [ADH21].

5.3.1 Netlib

For these experiments, we run the algorithms on the Netlib problems [Gay85], a collection of linear programming problems historically used for benchmarking solvers. These problems are imported in the form

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & Gx \geqslant h \\ & l \leqslant x \leqslant u. \end{aligned}$$

We convert this formulation into that of (2.2) by writing the box constraints as

$$\begin{bmatrix} -I \\ I \end{bmatrix} x \leqslant \begin{bmatrix} -l \\ u \end{bmatrix}$$

and removing rows where $(-l, u)$ is ∞ to form the constraint

$$\tilde{G}x \leqslant \tilde{h}.$$

Then we reform the problem as

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & \begin{bmatrix} -G \\ \tilde{G} \end{bmatrix} x \leqslant \begin{bmatrix} -h \\ \tilde{h} \end{bmatrix}. \end{aligned}$$

5.3.1.1 Netlib Feasible Problems

This data set of 99 feasible linear programs includes problems across different scales from problems with 32 to 22275 variables and 25 to 16676 constraints, making it a suitable data set for testing algorithms on small- to medium-scale problems.

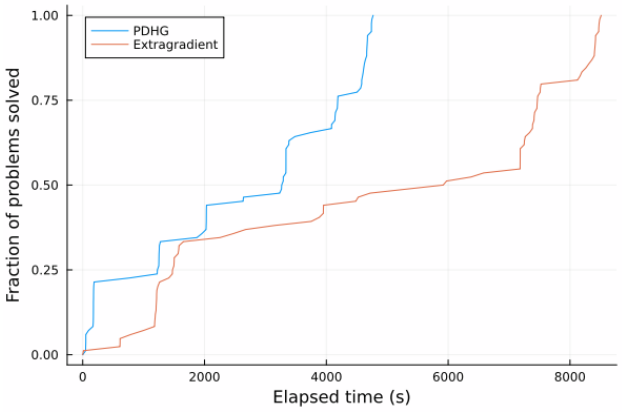

Figure 5.5: Fraction of problems solved plotted against the total elapsed time.

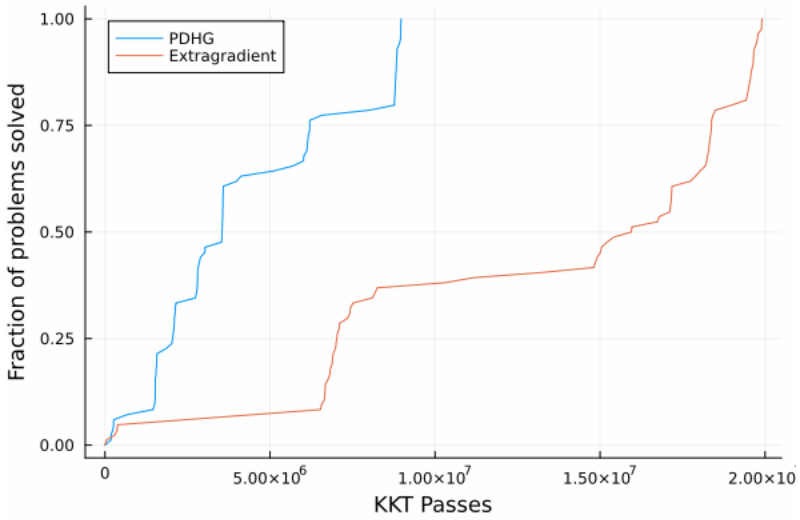

Figure 5.6: Fraction of problems solved plotted against the total number of KKT passes.

In Figure 5.5 and Figure 5.6, we show the fraction of the 99 Netlib problems solved

plotted against the elapsed time and against the total number of KKT passes respectively. Both plots indicate that the performance of PDHG is generally better than that of the extragradient method, but there are problems on which extragradient outperforms PDHG.

In Table 5.1, we report the optimal objective value for some of the Netlib problems where PDHG and extragradient method converge to meaningful solution. In the second column is the optimal value reported on Netlib.

instance	optimal value	PDHG	extragradient
25fv47	5501.8459	5509.8109	5970.9263
afiro	-464.7531	-465.3581	-464.9837
bandm	-158.6280	-158.2968	-157.3001
bore3d	1373.0803	1400.5497	1402.3580
cycle	-5.2263	-2.7768	-3.2487
d2q06c	1.2278e5	1.2385e5	2.0025e5
d6cube	315.4917	315.4913	322.3287
degen3	-987.2940	-987.0684	-987.1569
e226	-18.7519	-18.5221	-18.6577
recipe	-266.6160	-257.7636	-266.3522

Table 5.1: Optimal objective values found for a subset of the Netlib problems.

The results of these experiments indicate that the first-order methods applied to the extended self-dual embedding are suitable for problems on a similar scale to the Netlib problems, and they converge to a correct solution. However, on 48 of the problems, either PDHG or the extragradient method converge to a solution with $\tau = \kappa = 0$, where the methods fail to get a solution, reflecting a weakness of the self-dual embedding methods. Additionally, for some problems the relative discrepancy between the actual optimal value and the ones returned by the first-order methods is large, even with a moderately tight tolerance of 1e$-$6, reflecting another weakness, loss of precision when recovering a solution

to the original problem from the embedding solution.

5.3.2 Netlib Infeasible Problems

Netlib also contains a set of 29 infeasible linear programs. The performance of the methods on this set of problems are depicted in Figure 5.7 and Figure 5.8.

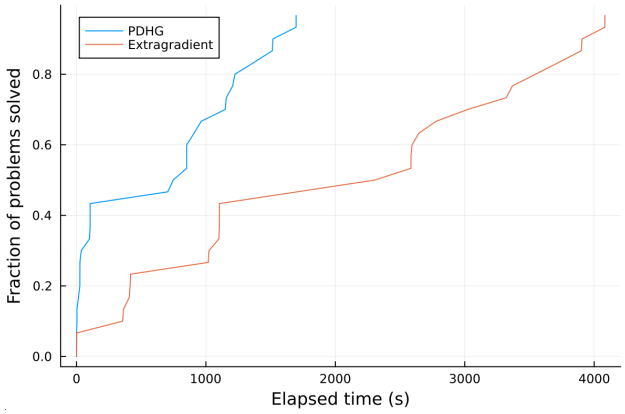

Figure 5.7: Fraction of problems solved plotted against the total elapsed time.

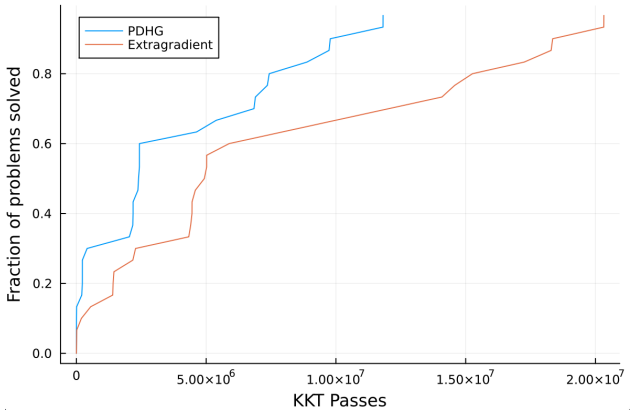

Figure 5.8: Fraction of problems solved plotted against the total number of KKT passes

Similarly to the Netlib feasible problems, these results provide evidence that PDHG outperforms the extragradient method in general. In 12 of the problems, both methods

converged to a solution with $\tau = \kappa = 0$; for problems on which a meaningful solution with $\tau + \kappa > 0$ is found, the first-order methods correctly conclude that the problems are infeasible.

From the results of the experiments on the Netlib feasible and infeasible problems, we observe that PDHG generally solves the problems faster than the extragradient method. However, for a significant proportion of the problems, both algorithms converge to a meaningless solution.

We also run a similar experiment on the LP benchmark described by Hans Mittelmann at [Mit]. These problems are very large compared to the Netlib ones, and for some problems, the first-order methods would take excessively long to converge to a solution for the self-dual embedding.

5.3.3 Randomly Generated Problems

In this section, we test the methods on randomly generated problems to analyze how the size of a problem affects the convergence time of our methods. We randomly generate feasible linear programs by first generating random $A \in \mathbb{R}^{m \times n}$ and $G \in \mathbb{R}^{m \times n}$ with density 0.01, $x_0 \in \mathbb{R}^n$, and $s_0 \in \mathbb{R}^m_+$. Next, we choose b and h as

$$b = Ax_0, \qquad h = Gx_0 + s$$

so x_0 is a feasible point. The results of some random instances are shown in Table 5.2. In this table, the problem dimensions (n, m) represent an LP with n variables, m equality constraints, and m inequality constraints.

problem dimensions	PDHG	extragradient
(500, 250)	0.194	0.283
(500, 500)	0.405	0.461
(500, 750)	5.669	4.038
(500, 1000)	8.920	16.421
(1000, 250)	0.099	0.161
(1000, 500)	0.310	0.598
(1000, 750)	6.965	10.518
(1000, 1000)	92.507	181.684

Table 5.2: CPU times in seconds of our methods solving randomly generated LPs.

The results of this experiment indicate that PDHG and the extragradient method are able to adequately find a solution for smaller problems, but for larger problems the increase in problem dimensions from forming the embedding causes difficulty for the first-order methods. Consistent with the performance of these methods on the Netlib problems, PDHG generally performs better than the extragradient method, though the difference in performance, especially on smaller problems, seems smaller.

5.4 Trace Norm Minimization

For this experiment, we test our first-order methods on a non-polyhedral conic linear program by considering a trace norm minimization problem

$$\text{minimize} \quad \left\| \sum_{k=1}^{n} x_k A_k - B \right\|_*$$

with conic linear programming formulation

$$\begin{aligned} &\text{minimize} \quad t \\ &\text{subject to} \quad \begin{bmatrix} \text{vec}(A_1) & \text{vec}(A_2) & \cdots & \text{vec}(A_n) & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ t \end{bmatrix} \succeq_{\mathcal{K}^*} \begin{bmatrix} \text{vec}(B) \\ 0 \end{bmatrix}. \end{aligned}$$

We note that the projection onto the trace norm cone can be efficiently computed as

$$\Pi_{\mathcal{K}^*}(\text{vec}(X), s) = \begin{cases} (\text{vec}(X), s), & s \geqslant \|X\|_* \\ (\text{vec}(U\text{diag}(\hat{\sigma}_1, \ldots, \hat{\sigma}_r)V^T), s + \lambda^\star), & \text{otherwise} \end{cases}$$

where X has singular value decomposition

$$X = U\text{diag}(\sigma_1, \ldots, \sigma_r)V^T,$$

$\hat{\sigma}_k$ is the soft threshold of σ_k for each $k = 1, \ldots, r$,

$$\hat{\sigma}_k = \begin{cases} \sigma_k - \lambda^\star, & \sigma_k \geqslant \lambda^\star \\ 0, & -\lambda^\star \leqslant \sigma_k \leqslant \lambda^\star \\ \sigma_k + \lambda^\star, & \sigma_k \leqslant -\lambda^\star, \end{cases}$$

and $\lambda^\star$ is a positive root of the piecewise linear function

$$\sum_{k=1}^{r} \max\{0, \sigma_k - \lambda\} - \lambda - s.$$

This result follows from Theorem 6.36 in [Bec17] and the proximal operator of $\|\cdot\|_*$. Since the methods require projection onto the dual cone, we also note that from Moreau's decomposition identity, the projection onto the operator norm cone

$$\mathcal{K} = \{(\text{vec}(X), s) \mid s \geqslant \|X\|\}$$

can be found as

$$\Pi_{\mathcal{K}}(x, s) = (x, s) + \Pi_{\mathcal{K}^*}(-x, -s).$$

In this experiment, we randomly generate square matrices $A_1, A_2, \ldots, A_n, B$ with density of 0.05 and solve the extended self-dual embedding. We compare the elapsed time with that of CVX. The CPU times are in Table 5.3, and we also report the objective value found in Table 5.4. Here, the problem dimensions (n, m) reflect a problem with n variables and matrices $A_1, \ldots, A_n, B \in \mathbb{R}^{m \times m}$.

problem dimensions	CVX	PDHG	extragradient
(100, 50)	18.4688	30.2940	36.5160
(100, 100)	532.8281	166.7670	125.0410
(100, 150)	2046.4000	744.7190	641.0940

Table 5.3: CPU times in seconds of CVX, PDHG, and the extragradient method for randomly generated trace norm minimization problems.

problem dimensions	CVX	PDHG	extragradient
(100, 50)	61.1516	61.1516	61.1516
(100, 100)	181.4902	181.4902	181.4906
(100, 150)	320.3810	320.3815	320.3825

Table 5.4: Objective values found by CVX, PDHG, and the extragradient method for randomly generated trace norm minimization problems.

We observe that the results of these experiments provide evidence that the complexity of solving the extended self-dual embedding with first-order methods scales better than that of forming and solving the Schur complement system for trace norm minimization.

CHAPTER 6

Conclusions and Future Work

In conclusion, in this work we implemented two first-order methods for solving conic linear programs by solving a self-dual embedding. We empirically show through numerical experiments that these methods perform well for conic linear programs. Though not competitive with more mature methods for very large problems, for small- and medium-scale problems, solving the extended self-dual embedding with PDHG or the extragradient method is a successful approach for finding an optimal vector or certificate of primal or dual infeasibility without requiring an infeasibility detection method.

Several possible future improvements are relevant. Firstly, one of the next steps for our methods would be implementing the facial reduction algorithm so a solution can be found in the pathological $\tau = \kappa = 0$ case. Improvements can also come from further exploiting the symmetric structure of the self-dual problem. When solved with a first-order method, the self-dual embedding is beneficial because it includes infeasibility detection capacity and has a known optimal value, and its self-dual structure makes the extragradient method seem especially suitable, but more improvements are required for this method to be scalable to very large problems. Another area of improvement could be the development of techniques to reduce the impact of significantly increasing the problem size and dimension when forming the self-dual embedding.

Algorithmic improvements can also be found from fine-tuning the large number of parameters and choices for our methods. For instance, in this work we form the embedding by choosing x_0, y_0, s_0, and z_0 according to a simple scheme, but the embedding can be formed

with any choice of x_0, y_0, $s_0 \succeq_{\mathcal{K}} 0$, and $z_0 \succeq_{\mathcal{K}^*} 0$; there may be possible improvements in choosing these vectors more thoughtfully with consideration for algorithmic performance. We also initialize the algorithms using a simple heuristic from linear programming; there may be better initialization approaches customized for non-polyhedral cones or that are tailored to the structure and symmetry of the extended self-dual embedding. Additionally, in our implementation we use a simple stopping condition, and it may be beneficial to implement other techniques commonly used in solvers like considering a weighted sum of $\|r_{\mathrm{e}}\|$ and $\|r_{\mathrm{i}}\|$ instead of simply $\|(r_{\mathrm{e}}, r_{\mathrm{i}})\|$, associating different tolerances for each component, and using a combination of absolute and relative tolerances. It may also be useful to include more information from the original primal and dual programs in addition to information from the embedding. Furthermore, there are many options for line searches and step size policies when finding an acceptable iterate, and there may be one better suited for solving the self-dual embedding.

Another future research direction is with replacing the proximal operators in the PDHG updates (3.1) and in the extragradient updates (4.1) with the *Bregman proximal operator*, which generalizes the proximal operator. The Bregman proximal operator of f maps x to the minimizer of $f(x) + d(x, y)$ where $d(x, y) = h(x) - \langle \nabla h(y), x \rangle$ for some convex h. We direct to [CZ97] for reference on Bregman divergences and point to [Teb92, CFI07] as examples of research on Bregman methods for optimization. We note that choosing $h = \|\cdot\|_2^2/2$ yields the standard proximal operator. This generalization allows for more flexibility in designing methods that could better fit a problem's geometry or structure.

REFERENCES

[ADH21] David Applegate, Mateo Diaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O' Donoghue, and Warren Schudy. "Practical Large-Scale Linear Programming using Primal-Dual Hybrid Gradient." In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pp. 20243–20257. Curran Associates, Inc., 2021.

[ADL21] David Applegate, Mateo Díaz, Haihao Lu, and Miles Lubin. "Infeasibility detection with primal-dual hybrid gradient for large-scale linear programming." *arXiv preprint arXiv:2102.04592*, 2021.

[ADV10] Martin S. Andersen, Joachim Dahl, and Lieven Vandenberghe. "Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones." *Mathematical Programming Computation*, **2**(3):167–201, 2010.

[ADV13] Martin S. Andersen, Joachim Dahl, and Lieven Vandenberghe. "CVXOPT: A Python package for convex optimization.", 2013.

[AG03] Farid Alizadeh and Donald Goldfarb. "Second-order cone programming." *Mathematical programming*, **95**(1):3–51, 2003.

[AHL22] David Applegate, Oliver Hinder, Haihao Lu, and Miles Lubin. "Faster first-order primal-dual methods for linear programming using restarts and sharpness." *Mathematical Programming*, **201**(1–2):133—184, October 2022.

[Ali95] Farid Alizadeh. "Interior point methods in semidefinite programming with applications to combinatorial optimization." *SIAM journal on Optimization*, **5**(1):13–51, 1995.

[ApS19] MOSEK ApS. *Semidefinite optimization*, 2019.

[AT05] Alfred Auslender and Marc Teboulle. "Interior projection-like methods for monotone variational inequalities." *Mathematical Programming*, **104**(1):39–68, Sep 2005.

[Bar86] Earl R. Barnes. "A variation on Karmarkar's algorithm for solving linear programming problems." *Mathematical programming*, **36**:174–182, 1986.

[Bec17] Amir Beck. *First-Order Methods in Optimization*. SIAM, 2017.

[BEK17] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. "Julia: A fresh approach to numerical computing." *SIAM Review*, **59**(1):65–98, 2017.

[Ben68] Adi Ben-Israel. "Geometric Programming–Theory and Application (R. J. Duffin, E. L. Peterson and C. Zener)." *SIAM Review*, **10**(2):235–236, 1968.

[BKV07] Stephen P. Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. "A tutorial on geometric programming." *Optimization and engineering*, **8**:67–127, 2007.

[BM03] Samuel Burer and Renato D. C. Monteiro. "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization." *Mathematical Programming*, **95**(2):329–357, 2003.

[BN01] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[BPC11] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers." *Foundations and Trends® in Machine Learning*, **3**(1):1–122, 2011.

[BV04] Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[BW81] Jon M. Borwein and Henry Wolkowicz. "Facial reduction for a cone-convex programming problem." *Journal of the Australian Mathematical Society*, **30**(3):369–380, 1981.

[CFI07] João Xavier Cruz Neto, Orizon Pereira Ferreira, Alfredo N. Iusem, and Renato D. C. Monteiro. "Dual convergence of the proximal point method with Bregman distances for linear programming." *Optimization Methods and Software*, **22**(2):339–360, 2007.

[Cha09] Robert Chares. *Cones and interior-point algorithms for structured convex optimization involving powers and exponentials*. PhD thesis, Université catholique de Louvain, 2009.

[CP11] Antonin Chambolle and Thomas 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.

[CP15] Antonin Chambolle and Thomas Pock. "On the ergodic convergence rates of a first-order primal–dual algorithm." *Mathematical Programming*, **159**:253–287, 05 2015.

[CP16] Antonin Chambolle and Thomas Pock. "An introduction to continuous optimization for imaging." *Acta Numerica*, **25**:161–319, 2016.

[CZ97] Yair Censor and Stavros Andrea Zenios. *Parallel optimization: Theory, algorithms, and applications*. Oxford University Press, USA, 1997.

[DA22] Joachim Dahl and Erling D. Andersen. "A primal-dual interior-point algorithm for nonsymmetric exponential-cone optimization." *Mathematical Programming*, **194**(1):341–370, 2022.

[Dan63] George Bernard Dantzig. *Linear programming and extensions*. Princeton University Press, 1963.

[DB16] Steven Diamond and Stephen P. Boyd. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization." *Journal of Machine Learning Research*, 2016. To appear.

[Dik67] Ilya I. Dikin. "Iterative solution of problems of linear and quadratic programming." *Doklady Akademii Nauk SSSR*, **174**:747–748, 1967.

[DKW77] J. J. Dinkel, G. A. Kochenberger, and S-N Wong. "Entropy Maximization and Geometric Programming." *Environment and Planning A: Economy and Space*, **9**(4):419–427, 1977.

[DR56] Jim Douglas and Henry H. Rachford. "On the numerical solution of heat conduction problems in two and three space variables." *Transactions of the American mathematical Society*, **82**(2):421–439, 1956.

[DW17] Dmitriy Drusvyatskiy and Henry Wolkowicz. "The Many Faces of Degeneracy in Conic Optimization." *Foundations and Trends® in Optimization*, **3**(2):77—-170, Dec 2017.

[EZC10] Ernie Esser, Xiaoqun Zhang, and Tony F. Chan. "A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science." *SIAM Journal on Imaging Sciences*, **3**(4):1015–1046, 2010.

[Faz02] Maryam Fazel. *Matrix rank minimization with applications*. PhD thesis, Stanford University, 2002.

[FK94] Jacques Faraut and Ádám Korányi. *Analysis on symmetric cones*. Oxford university press, 1994.

[Gab83] Daniel Gabay. "Applications of the Method of Multipliers to Variational Inequalities." In *Studies in Mathematics and Its Applications*, volume 15, pp. 299–331. Elsevier, 1983.

[Gay85] David M. Gay. "Electronic mail distribution of linear programming test problems." *Mathematical Programming Society COAL Newsletter*, **13**:10–12, 1985.

[GB08] Michael Grant and Stephen P. Boyd. "Graph implementations for nonsmooth convex programs." In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pp. 95–110. Springer-Verlag Limited, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[GB14] Michael Grant and Stephen P. Boyd. "CVX: Matlab Software for Disciplined Convex Programming, version 2.1." https://cvxr.com/cvx, March 2014.

[GC21] Paul Goulart and Yuwen Chen. "Clarabel: A Library for Optimization and Control." https://clarabel.org/stable/, 2021.

[GT56] Alan J. Goldman and Albert William Tucker. "Theory of Linear Programming." In Harold William Kuhn and Albert William Tucker, editors, *Linear Inequalities and Related Systems (AM-38)*, volume 38 of *Annals of Mathematics Studies*, pp. 53–97. Princeton University Press, 1956.

[GW95] Michel X. Goemans and David P. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming." *Journal of the ACM (JACM)*, **42**(6):1115–1145, 1995.

[JN12] Anatoli Juditsky and Arkadi Nemirovski. "First-Order Methods for Nonsmooth Convex Large-Scale Optimization, II: Utilizing Problem's Structure." In Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors, *Optimization for machine learning*, chapter 6, pp. 149–183. MIT Press, 2012.

[Kar84] Narendra Karmarkar. "A new polynomial-time algorithm for linear programming." In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pp. 302—-311, New York, NY, USA, 1984. Association for Computing Machinery.

[Kha79] Leonid Genrikhovich Khachiyan. "A polynomial algorithm in linear programming." In *Doklady Akademii Nauk*, volume 244, pp. 1093–1096. Russian Academy of Sciences, 1979.

[KM72] Victor Klee and George J. Minty. "How good is the simplex algorithm?" *Inequalities*, **3**(3):159–175, 1972.

[Kor77] Galina M. Korpelevich. "The extragradient method for finding saddle points and other problems." *Matekon*, **13**(4):35–49, 1977.

[L04] Johan Löfberg. "YALMIP : A Toolbox for Modeling and Optimization in MATLAB." In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[Lov03] László Lovász. "Semidefinite programs and combinatorial optimization." In *Recent advances in algorithms and combinatorics*, pp. 137–194. Springer, 2003.

[LVB98] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen P. Boyd, and Hervé Lebret. "Applications of second-order cone programming." *Linear algebra and its applications*, **284**(1-3):193–228, 1998.

[Meh92] Sanjay Mehrotra. "On the Implementation of a Primal-Dual Interior Point Method." *SIAM Journal on Optimization*, **2**(4):575–601, 1992.

[Mit] Hans D. Mittelmann. "Decison Tree for Optimization Software." https://plato.asu.edu/guide.html.

[Nem04] Arkadi Nemirovski. "Prox-Method with Rate of Convergence O(1/t) for Variational Inequalities with Lipschitz Continuous Monotone Operators and Smooth Convex-Concave Saddle Point Problems." *SIAM Journal on Optimization*, **15**(1):229–251, 2004.

[Nes06] Yurii Nesterov. "Towards nonsymmetric conic optimization." *Optimization Methods and Software*, 2006.

[NN94] Yurii Nesterov and Arkadi Nemirovski. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[NT97] Yurii Nesterov and Michael J. Todd. "Self-scaled barriers and interior-point methods for convex programming." *Mathematics of Operations research*, **22**(1):1–42, 1997.

[OCP16] 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, June 2016.

[OCP23] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. "SCS: Splitting Conic Solver, version 3.2.4." https://github.com/cvxgrp/scs, November 2023.

[PCB09] Thomas Pock, Daniel Cremers, Horst Bischof, and Antonin Chambolle. "An algorithm for minimizing the Mumford-Shah functional." In *2009 IEEE 12th International Conference on Computer Vision*, pp. 1133–1140, 2009.

[PFA17] Frank Permenter, Henrik A. Friberg, and Erling D. Andersen. "Solving Conic Optimization Problems via Self-Dual Embedding and Facial Reduction: A Unified Approach." *SIAM Journal on Optimization*, **27**(3):1257–1282, 2017.

[Ren88] James Renegar. "A polynomial-time algorithm, based on Newton's method, for linear programming." *Mathematical Programming*, **40**(1):59–93, 1988.

[Ren95] James Renegar. "Incorporating Condition Measures into the Complexity Theory of Linear Programming." *SIAM Journal on Optimization*, **5**(3):506–524, 1995.

[RFP10] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization." *SIAM Review*, **52**(3):471–501, 2010.

[Roc70] Ralph Tyrell Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, 1970.

[Rui01] Daniel Ruiz. "A scaling algorithm to equilibrate both rows and columns norms in matrices." Technical report, CM-P00040415, 2001.

[Sch98] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[Ser15] Santiago Akle Serrano. *Algorithms for unsymmetric cone optimization and an implementation for problems with the exponential cone*. PhD thesis, Stanford University, 2015.

[SS22] Guillaume Sagnol and Maximilian Stahlberg. "PICOS: A Python interface to conic optimization solvers." *Journal of Open Source Software*, **7**(70):3915, February 2022.

[Stu99] Jos F. Sturm. "Using SeDuMi 1.02, A MATLAB toolbox for optimization over symmetric cones." *Optimization Methods and Software*, **11**(1-4):625–653, 1999.

[SY15] Anders Skajaa and Yinyu Ye. "A homogeneous interior-point algorithm for non-symmetric convex conic optimization." *Mathematical Programming*, **150**(2):391–422, 2015.

[Teb92] Marc Teboulle. "Entropic proximal mappings with applications to nonlinear programming." *Mathematics of Operations Research*, **17**(3):670–690, 1992.

[Tse08] Paul Tseng. "On Accelerated Proximal Gradient Methods for Convex Optimization.", 2008.

[TTT99] Kim-Chuan Toh, Michael Todd, and Reha Tütüncü. "SDPT3—A Matlab software package for semidefinite programming, version 2.1." *Optimization Methods and Software*, **11**:545–581, 10 1999.

[Tuc56] Albert William Tucker. "Dual Systems of Homogeneous Linear Relations." In Harold William Kuhn and Albert William Tucker, editors, *Linear Inequalities and Related Systems (AM-38)*, volume 38 of *Annals of Mathematics Studies*, pp. 3–18. Princeton University Press, 1956.

[VB96] Lieven Vandenberghe and Stephen P. Boyd. "Semidefinite Programming." *SIAM Review*, **38**(1):49–95, 1996.

[VB99] Lieven Vandenberghe and Stephen P. Boyd. "Applications of semidefinite programming." *Applied Numerical Mathematics*, **29**(3):283–299, 1999.

[VMF86] Robert J. Vanderbei, Marc S. Meketon, and Barry A. Freedman. "A modification of Karmarkar's linear programming algorithm." *Algorithmica*, **1**:395–407, 1986.

[Wri97] Stephen J. Wright. *Primal-dual interior-point methods*. SIAM, 1997.

[YN77] David B. Yudin and Arkadi S. Nemirovskii. "Informational complexity and efficient methods for the solution of convex extremal problems." *Matekon*, **13**(3):25–45, 1977.

[YN83] David B. Yudin and Arkadi S. Nemirovskii. *Problem complexity and method efficiency in optimization*. John Wiley & Sons, 1983.

[YTM94] 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.

[ZC08] Mingqiang Zhu and Tony Chan. "An efficient primal-dual hybrid gradient algorithm for total variation image restoration." *UCLA CAM Report*, **34**(2), 2008.

[Zen61] Clarence Zener. "A mathematical aid in optimizing engineering designs." *Proceedings of the National Academy of Sciences*, **47**(4):537–539, 1961.

[ZFP20] Yang Zheng, Giovanni Fantuzzi, Antonis Papachristodoulou, Paul Goulart, and Andrew Wynn. "Chordal decomposition in operator-splitting methods for sparse semidefinite programs." *Mathematical Programming*, **180**(1):489–532, 2020.