Projection onto hyperbolicity cones and beyond: a dual Frank-Wolfe approach

Takayuki Nagano*   Bruno F. Lourenço†   Akiko Takeda‡

September 4, 2024

Abstract

We discuss the problem of projecting a point onto an arbitrary hyperbolicity cone from both theoretical and numerical perspectives. While hyperbolicity cones are furnished with a generalization of the notion of eigenvalues, obtaining closed form expressions for the projection operator as in the case of semidefinite matrices is an elusive endeavour. To address that we propose a Frank-Wolfe method to handle this task and, more generally, strongly convex optimization over closed convex cones. One of our innovations is that the Frank-Wolfe method is actually applied to the dual problem and, by doing so, subproblems can be solved in closed-form using minimum eigenvalue functions and conjugate vectors. To test the validity of our proposed approach, we present numerical experiments where we check the performance of alternative approaches including interior point methods and an earlier accelerated gradient method proposed by Renegar. We also show numerical examples where the hyperbolic polynomial has millions of monomials. Finally, we also discuss the problem of projecting onto p-cones which, although not hyperbolicity cones in general, are still amenable to our techniques.

1 Introduction

Hyperbolicity cones [16, 18, 4, 41] are a far-reaching family of closed convex cones containing all symmetric cones and all polyhedral cones. In particular, the cone of $n \times n$ real symmetric positive semidefinite matrices $\mathcal{S}^n_+$ is a hyperbolicity cone. One distinctive feature of $\mathcal{S}^n_+$ is that the orthogonal projection onto $\mathcal{S}^n_+$ has a well-known expression that can be described in terms of the spectral decomposition of a matrix. More precisely, if X is an $n \times n$ symmetric matrix, $v_1, \ldots v_n$ are n orthonormal eigenvectors of X and $\lambda_1, \ldots, \lambda_n$ are the corresponding eigenvalues, then the projection onto $\mathcal{S}^n_+$ with respect to the Frobenius norm is the result of "zeroing the negative eigenvalues in the spectral decomposition":

$$P_{\mathcal{S}^n_+}(X) = \underset{Y \in \mathcal{S}^n_+}{\arg\min} \|Y - X\|_F = \sum_{i=1}^{n} \max(0, \lambda_i) v_i v_i^\top, \tag{1.1}$$

where $\|\cdot\|_F$ is the Frobenius norm. Analogously, for a hyperbolicity cone there is a natural notion of *eigenvalues* (see Section 2.1) that is strong enough to allow the extension of certain linear algebraic results about symmetric matrices, e.g., [4, 41]. With this in mind, the starting point of this project was the following questions:

---

*Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan.

†Department of Fundamental Statistical Mathematics, Institute of Statistical Mathematics, 10-3 Midori-cho, Tachikawa, Tokyo 190-8562, Japan. This author was supported partly by the JSPS Grant-in-Aid for Early-Career Scientists 23K16844. (bruno@ism.ac.jp)

‡Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan and Center for Advanced Intelligence Project, RIKEN, 1-4-1, Nihonbashi, Chuo-ku, Tokyo 103-0027, Japan. This was author was supported partly by the JSPS Grant-in-Aid for Scientific Research (B) 23H03351 and JST ERATO Grant Number JPM- JER1903. (takeda@mist.i.u-tokyo.ac.jp)

Given a general hyperbolicity cone $\Lambda \subseteq \mathbb{R}^n$ and $x \in \mathbb{R}^n$, how to compute the projection of x onto Λ efficiently, i.e., how to compute $P_\Lambda(x) := \arg\min_{y \in \Lambda} \|y - x\|$? Are there closed form expressions analogous to (1.1) in term of the eigenvalues of x ?

First, we mention briefly why these questions are important. Besides being an interesting geometric question on its own, the usefulness of having a readily available projection operator for some convex set is well-documented in optimization (e.g., see [20]): it is, after all, a basic requirement for the applicability of several algorithms. Not only that, methods such as *cyclic projections* and others (e.g., see [3]) can be used to refine the feasibility properties of a solution obtained by a numerical solver. All of this is, of course, contingent on either having a "reasonable" closed form solution for the projection operator or a fast numerical method.

In the case of a general hyperbolicity cone, the fact that we have a relatively powerful notion of eigenvalues gives some hope of an analogue of (1.1). Unfortunately, even though we have *eigenvalues*, we do not have a suitable generalization of the notion of *spectral decomposition* that is always available for an arbitrary hyperbolicity cone. Nevertheless, in this paper we will present several partial results on this front regarding the computation of *distance functions to hyperbolicity cones*.

As for numerical methods, there are a few challenges one must take care. It goes without saying that a method to compute the projections onto Λ cannot make use of, say, the projection operator $P_\Lambda(\cdot)$ onto Λ, so this excludes several algorithms as potential candidates such as projected gradient-based methods and augmented Lagrangian methods.

In order to overcome this difficulty, we propose a Frank-Wolfe based method for computing the projection operator onto a hyperbolicity cone Λ. However, developing a Frank-Wolfe based approach successfully has its own challenges. For example, the subproblems appearing during the Frank-Wolfe iteration should either have closed form solutions or be efficiently solvable. In addition, it is typically required that the feasible region be compact, which is not true for the problem of projecting a point onto a convex cone.

In this work, we show that it is possible to overcome *all* these difficulties in the case of hyperbolicity cones and we will discuss a *dual* Frank-Wolfe method for solving the projection problem over a hyperbolicity cone and beyond. Our approach is dual in the sense that the Frank-Wolfe algorithm is actually applied to the Fenchel dual of our problem of interest. This is because, surprisingly, solving the problem from the dual side leads to subproblems that have closed form solutions in terms of the underlying hyperbolic polynomial.

Although our focus will be on the hyperbolicity cone case, the method we discuss in this paper is actually capable of solving a larger class of problems as follows:

$$
\begin{aligned}
\min_{x \in \mathbb{R}^n} \quad & f(x) \\
\text{s.t.} \quad & Tx + b \in \mathcal{K}
\end{aligned}
\tag{1.2}
$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a closed proper μ-strongly convex function, $T : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map and $\mathcal{K}$ is a full-dimensional pointed closed convex cone. In this case, we are able to show that the subproblems that appear in the Frank-Wolfe algorithm can be expressed in terms of *generalized eigenvalue functions*. The problem (1.2) contains as a special case the projection problem, since for fixed $x_0 \in \mathbb{R}^n$, we can take $f(x) := \|x - x_0\|^2$, let T be the identity map and $b := 0$.

We now summarize the main contributions of this paper.

- We provide a few theoretical results on the projection operator and the distance function to hyperbolicity cones. In particular, for the so-called *isometric* hyperbolic polynomials, there are formulae analogous to the ones that hold for the positive semidefinite cone, see Propositions 3.4 and 3.6.

- We propose a dual Frank-Wolfe method for solving (1.2), which includes the particular case of projecting onto a hyperbolicity cones. One of our main results is that the solution to the subproblems appearing in our method can be expressed in terms of generalized minimal eigenvalue computations and conjugate vectors, see Theorem 4.3. In the particular case of hyperbolicity cones, we show how conjugate vectors can easily be obtained from the underlying hyperbolic polynomial, see Proposition 4.4. We then provide several convergence results in Section 4.2. We emphasize that since the Frank-Wolfe method is applied to a dual problem of (1.2), it is still necessary to bridge the gap between the dual and primal problems.

With this issue in mind, we provide some convergence results from the primal side, see Theorems 4.6 and 4.7. We also provide a discussion on practical issues one may find when implementing our approach, see Section 4.3.

- We provide an implementation of our algorithm and numerical experiments in Section 5. Taking interior point methods as a baseline, we compare against an earlier algorithm proposed by Renegar for hyperbolicity cones [43]. We also show that our implementation is capable of handling polynomials with millions of monomials, provided that the underlying computational algebra is carefully implemented, see Section 5.1.2. At the end, we also have numerical experiments for non-hyperbolicity cones, see Section 5.2.

1.1 Related works

In this brief subsection, we review some key works on numerical aspects of hyperbolicity cones. Güler wrote a pioneering work on hyperbolic polynomials and interior point methods (IPMs) [18]. Nowadays, there are a few IPM-based generic conic solvers that are capable of handling hyperbolicity cone constraints, such as DDS [25], Hypatia [9] and alfonso [37]. For Hypatia and alfonso, it seems possible, in theory, to use their "generic conic interface" to implement optimization over hyperbolicity cones. DDS, on the other hand, has specific functionalities tailored for hyperbolicity cones.

In any case, the problem of finding the orthogonal projection onto a hyperbolicity cone $\mathcal{K}$ can be naturally formulated as a conic linear program over the direct product between $\mathcal{K}$ and an additional second-order cone constraint. Therefore, finding the projection can be, in theory, done with one of those solvers.

Regarding first-order methods, Renegar proposed an algorithm for conic linear programs over hyperbolicity cones which uses smoothing and accelerating techniques [43].

In Section 5, we present numerical experiments in order to compare the performance of different approaches for particular cases of the problem in (1.2). Although we defer a detailed discussion to Section 5, we will see that our proposed algorithm (which is a first-order method) is quite competitive in comparison with the aforementioned approaches.

1.2 Outline of this work

In Section 2, we recall some necessary definitions from convex analysis, hyperbolic polynomials and the Frank-Wolfe method. In Section 3 we prove a theoretical discussion on the distance function and the projection operator onto an isometric hyperbolicity cone. In Section 4, we propose and analyze a first-order dual algorithm to optimize strongly convex functions over regular cones based on a Frank-Wolfe method. In Section 5, we show the results of numerical experiments. We compare our algorithm with Renegar's method described in [43] and the DDS [25] solver. As our proposed method is also applicable to more general cones, we also include numerical experiments for the problem of projecting onto p-cones and a comparison with Mosek [34]. Section 6 concludes this paper.

2 Preliminaries

We start with notations and basic definitions. Given an element $u \in \mathbb{R}^n$, we will denote its i-th component by u_i. We use $\mathbf{1}_n$ to denote the n-dimensional vector whose components are all equal to 1. We write $\mathbb{R}^n_\downarrow$ for the cone of elements $u \in \mathbb{R}^n$ satisfying $u_1 \geq \cdots \geq u_n$. Let $u \in \mathbb{R}^d$, we denote by $u^\downarrow$ the element in $\mathbb{R}^d_\downarrow$ corresponding to a reordering of the coordinates of u in such a way that

$$u_1^\downarrow \geq \cdots \geq u_d^\downarrow.$$

We write $\mathbb{R}^n_+$ for the nonnegative orthant, i.e., the elements $u \in \mathbb{R}^n$ such that $u_i \geq 0$ for every i.

For a convex subset $S \subseteq \mathbb{R}^n$, we denote its indicator function, recession cone, interior and relative interior by δ_S, 0^+S, $\operatorname{int} S$ and $\operatorname{ri} S$, respectively. Additionally, we suppose $\mathbb{R}^n$ is endowed with an inner product $\langle \cdot, \cdot \rangle$

and a corresponding induced norm $\|\cdot\|$. With that, we denote by $S^\perp$ the set of elements orthogonal to S. For a convex cone $\mathcal{K} \subseteq \mathbb{R}^n$, we define its dual cone as

$$\mathcal{K}^* := \{x \in \mathbb{R}^n \mid \forall y \in \mathcal{K}, \ \langle x, y \rangle \geq 0\}.$$

A cone $\mathcal{K}$ is said to be *pointed* if $\mathcal{K} \cap -\mathcal{K} = \{0\}$ and *full-dimensional* if its interior is non-empty. A full-dimensional pointed cone is said to be *regular*.

Two elements satisfying $x \in \mathcal{K}, y \in \mathcal{K}^*$ and $\langle x, y \rangle = 0$ are said to be *conjugate*. For $x \in \mathcal{K}$, we denote the set of elements conjugate to x by $\mathcal{F}_x^\Delta$ so that

$$\mathcal{F}_x^\Delta := \{y \in \mathcal{K}^* \mid \langle y, x \rangle = 0\} = \mathcal{K}^* \cap \{x\}^\perp. \tag{2.1}$$

The reason for this notation is that if $\mathcal{F}_x$ denotes the unique face of $\mathcal{K}$ satisfying $x \in \mathrm{ri}\,\mathcal{F}_x$, then $\mathcal{F}_x^\Delta$ as defined in (2.1) is exactly the conjugate face to $\mathcal{F}_x$, i.e., $\mathcal{F}_x^\Delta = \mathcal{K}^* \cap \mathcal{F}_x^\perp$ holds. For more details on faces of cones, see [2, 38].

For a closed convex function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$, we denote its conjugate function by f^*, which is defined as

$$f^*(s) := \sup_{x \in \mathrm{dom} f} \{\langle s, x \rangle - f(x)\},$$

where $\mathrm{dom} f = \{x \in \mathbb{R}^n \mid f(x) < \infty\}$.

For a linear map $T : \mathbb{R}^n \to \mathbb{R}^m$, we denote by T^* the adjoint map of T. We denote the operator norm of T by $\|T\|_{\mathrm{op}}$, which is defined as

$$\|T\|_{\mathrm{op}} := \sup_{x \neq 0} \frac{\|Tx\|}{\|x\|},$$

where by a slight abuse of notation we use the same symbol $\|\cdot\|$ to indicate the underlying norm in $\mathbb{R}^n$ and $\mathbb{R}^m$.

A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is called *L-smooth* if ∇f is Lipschitz continuous with constant $L > 0$, i.e.,

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|, \qquad \forall x, y \in \mathbb{R}^n.$$

For a $\mu > 0$, f is called *μ-strongly convex* if for every $\theta \in [0, 1]$ we have

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y) - \frac{\mu}{2}\theta(1 - \theta)\|x - y\|^2, \qquad \forall x, y \in \mathbb{R}^n.$$

Finally, we recall that the conjugate of a μ-strongly convex function is $1/\mu$-smooth, see [21, Theorem 4.2.1].

2.1 Hyperbolic polynomials

Let $p : \mathbb{R}^n \to \mathbb{R}$ be a homogeneous polynomial, we say that *p is hyperbolic along the direction e* if and only if $p(e) \neq 0$ and for every $x \in \mathbb{R}^n$, the one-dimensional polynomial

$$t \mapsto p(x - te)$$

has only real roots. Here we summarize some basic properties of hyperbolic polynomials that will be necessary in the subsequent sections. For more details, see [16, 18, 41].

Suppose that p has degree d. We define the map $\lambda : \mathbb{R}^n \to \mathbb{R}^d_\downarrow$ that maps $x \in \mathbb{R}^n$ to the d roots of the polynomial $p(x - te)$, ordered from largest to smallest. That is, we have

$$p(x - te) = p(e) \prod_{i=1}^{d} (\lambda_i(x) - t)$$

$$\lambda_1(x) \geq \cdots \geq \lambda_d(x).$$

In analogy to classical linear algebra, we will say that $\lambda_1(x), \dots, \lambda_d(x)$ are the *eigenvalues of x*. We also write $\lambda_d(x)$ as $\lambda_{\min}(x)$ to emphasize that $\lambda_d(x)$ is the smallest eigenvalue. Then, the *hyperbolicity cone of p along the direction e* is the closed convex cone $\Lambda(p, e)$ given by

$$\Lambda(p, e) = \{x \in \mathbb{R}^n \mid \lambda_{\min}(x) \geq 0\},$$

see Section 2 in [41]. If p and e are clear from the context, we write $\Lambda(p, e)$ as Λ.

For $x \in \mathbb{R}^n$, we denote by $\text{mult}(x)$ the number of zero eigenvalues of x. That is, $\text{mult}(x)$ is the multiplicity of zero as a root of $t \mapsto p(x - te)$.

A hyperbolic polynomial p is said to be *complete* if and only if

$$\{x \in \mathbb{R}^n \mid \lambda(x) = 0\} = \{x \in \mathbb{R}^n \mid \text{mult}(x) = d\} = \{0\}.$$

This happens if and only if Λ is *pointed*, see [41, Proposition 11].

Derivative relaxations   Let $D_e p$ denote the directional derivative of p along e, so that

$$D_e p(x) = \lim_{t \to 0} \frac{p(x + te) - p(x)}{t}, \quad \forall x \in \mathbb{R}^n.$$

Then, the function $D_e p : \mathbb{R}^n \to \mathbb{R}$ is also a hyperbolic polynomial along e. The hyperbolicity cone associated with $(D_e p, e)$, is called *the derivative cone* (of p along e) and is denoted by Λ'.

We write $D_e^i p$ for the higher order derivatives, so that $D_e^i p(x) = \frac{d^i p(x + te)}{dt^i}\big|_{t=0}$. Then, we define $p^{(0)} := p$, $p^{(1)} := D_e^1 p, \dots, p^{(d)} := D_e^d p$. Taking the derivative repeatedly gives a sequence of hyperbolic polynomials and associated hyperbolicity cones

$$\Lambda \subseteq \Lambda^{(1)} \subseteq \dots \subseteq \Lambda^{(d-1)}$$

Finally, we need the following property of hyperbolicity cones.

Theorem 2.1. *[41, Theorem 12] Let Λ be a hyperbolicity cone. Define $\partial^r \Lambda = \{x \in \Lambda \mid \text{mult}(x) = r\}$. Then, for $r \geq 2$,*

$$\partial^r \Lambda^{(1)} = \partial^{r+1} \Lambda$$

Also,

$$\partial^1 \Lambda^{(1)} \cap \Lambda = \partial^2 \Lambda$$

2.2   Generalized minimum eigenvalue functions

In this subsection we discuss a generalization of the minimum eigenvalue function that is applicable to general regular cones. Let $\mathcal{K} \subseteq \mathbb{R}^n$ be a regular (i.e., full-dimensional and pointed) closed convex cone. Let $e \in \text{int}\,\mathcal{K}$ be fixed, then the *minimum eigenvalue function* $\lambda_{\min} : \mathbb{R}^m \to \mathbb{R}$ *with respect to $\mathcal{K}$ and e* is defined as:

$$\lambda_{\min}(x) := \sup\{t \mid x - te \in \mathcal{K}\}. \tag{2.2}$$

First we observe that $\lambda_{\min}(x) = \inf\{t \mid x - te \notin \mathcal{K}\}$ holds and this was the original definition considered in [42, Section 2], see also [32, Section 2.5] and [23, Section 2]. Given that there is a dependency on e and $\mathcal{K}$, it might be appropriate to use some notation similar to "$\lambda_{\min}^{\mathcal{K}, e}$", but since there will be no ambiguity regarding the chosen $\mathcal{K}$ and e, we will use the simpler $\lambda_{\min}$. We recall the following basic properties of $\lambda_{\min}$.

Proposition 2.2. *Let $\mathcal{K} \subseteq \mathbb{R}^n$ be a regular closed convex cone and $e \in \text{int}\,\mathcal{K}$ be fixed. Let $\lambda_{\min}$ be as in (2.2). For $x \in \mathbb{R}^n$, the following items hold.*

(i) $\lambda_{\min}(x)$ *is finite.*

(ii) $x \in \mathcal{K} \iff \lambda_{\min}(x) \geq 0$.

(iii) $x \in \text{int}\,\mathcal{K} \iff \lambda_{\min}(x) > 0$.

Proof. Since e is an interior point of $\mathcal{K}$, there exists $u > 0$ such that $e + ux \in \mathcal{K}$ and since $\mathcal{K}$ is a cone, $x + e/u \in \mathcal{K}$ holds as well. This shows that $\lambda_{\min}(x) > -\infty$. On the other hand if $\lambda_{\min}(x) = \infty$, then $x - te$ belongs to $\mathcal{K}$ for all t. This would imply that $-e \in \mathcal{K}$ (e.g., see [44, Theorem 8.3]), which would contradict the pointedness of $\mathcal{K}$. This shows that item (i) holds.

For item (ii), if $x \in \mathcal{K}$, then $t = 0$ is a feasible solution to (2.2), so indeed $\lambda_{\min}(x) \geq 0$. Conversely, if $\lambda_{\min}(x) \geq 0$, then for every $k > 0$, there exists t_k satisfying $\lambda_{\min}(x) \geq t_k > \lambda_{\min}(x) - \frac{1}{k} \geq -\frac{1}{k}$ and $x - t_k e \in \mathcal{K}$. Passing to a convergent subsequence if necessary, we may assume that t_k converges to some $\bar{t} \geq 0$. Since $\mathcal{K}$ is closed, we have $x - \bar{t}e \in \mathcal{K}$. This implies that $x = (x - \bar{t}e) + (\bar{t}e) \in \mathcal{K}$.

The proof of item (iii) is similar so we omit it. $\qquad\square$

Now, suppose that $\mathcal{K} = \Lambda(p, e)$ is a regular hyperbolicity cone. In the following proposition we observe that $\lambda_{\min}$ as defined in Section 2.1 and in (2.2) coincide.

Proposition 2.3. *Let $\Lambda(p, e) \subseteq \mathbb{R}^n$ be a regular hyperbolicity cone and suppose that the degree of p is d. Then, for every $x \in \mathbb{R}^n$, $\lambda_d(x) = \sup\{t \mid x - te \in \Lambda(p, e)\}$, where $\lambda_d(x)$ is the smallest root of $t \mapsto p(x - te)$.*

Proof. Let $x \in \mathbb{R}^n$ and suppose that p has degree d. By definition, the eigenvalue map (as in Section 2.1) satisfies

$$\lambda(x - \alpha e) = \lambda(x) - \alpha \mathbf{1}, \qquad \forall \alpha \in \mathbb{R},$$

where $\mathbf{1} \in \mathbb{R}^n$ is the vector where each component is 1. Recalling that $x - te \in \Lambda(p, e)$ if and only if $\lambda(x - te) \geq 0$, we see that the condition "$x - te \in \Lambda(p, e)$" implies the component-wise inequality

$$\lambda(x) \geq t\mathbf{1}.$$

In particular, the maximum value that t can assume under the constraint "$x - te \in \Lambda(p, e)$" is $\lambda_d(x)$, where $\lambda_d(x)$ is the smallest root of $t \mapsto p(x - te)$. Conversely, if $t \coloneqq \lambda_d(x)$, we have $\lambda_d(x - \lambda_d(x)e) = 0$, so $x - \lambda_d(x)e \in \Lambda(p, e)$. That is, $\lambda_d(x)$ is the optimal solution to the maximization problem in (2.2). $\qquad\square$

2.3 The Frank-Wolfe Method

In this subsection, we review some of the basic aspects of the Frank-Wolfe method (FW method) proposed by Frank and Wolfe [13], which is also known as conditional gradient method [10, Chapter 3], see also [14, 7, 40]. Originally Frank and Wolfe proposed the algorithm to optimize a quadratic function over a polyhedral set, but the FW method is applicable to the following more general problem:

$$\min_{x \in C} f(x), \tag{2.3}$$

where C is a convex and compact set in $\mathbb{R}^n$ and f is a differentiable and L-smooth function in C. There are several variants on FW methods [49, 22, 27, 39], but here we only make use of the simplest version, see Algorithm 1, which follows the description in the survey [7].

Algorithm 1: The Frank-Wolfe method [13]

1: Choose a point $x_0 \in C$
2: **for** $k = 0, 1, \ldots$ **do**
3:    If x_k satisfies some stopping criterion, STOP
4:    Compute $s_k \in \arg\min_{x \in C}\langle \nabla f(x_k), x \rangle$
5:    Set $d_k \coloneqq s_k - x_k$
6:    Choose step size $\alpha_k \in (0, 1]$
7:    Set $x_{k+1} \coloneqq x_k + \alpha_k d_k$
8: **end for**

One important aspect of the FW method is that it does not require the projection operator onto C. Instead, we assume the availability of a linear optimization oracle over C which is capable of solving the

subproblem $\min_{x \in C}\langle \nabla f(x_k), x \rangle$ appearing in line 4 of Algorithm 1. A successful application of the FW method thus depends on having a fast way to solve the underlying subproblem. Fortunately there are many such problems, which have contributed for the recent renewed interest in FW methods in optimization, machine learning and even extensions to nonconvex problems, e.g., [24, 33, 1, 50].

Another useful feature of FW methods is that there is an easily computable measure of convergence called the Frank-Wolfe gap (the FW gap). The FW gap at x is denoted by $G(x)$ and is defined as

$$G(x) := \max_{s \in C}\langle -\nabla f(x), s - x \rangle.$$

Let x_{opt} denote an optimal solution to (2.3). Since f is convex, for $x \in C$ we have $\langle -\nabla f(x), x_{\mathrm{opt}} - x \rangle \geq f(x) - f(x_{\mathrm{opt}}) \geq 0$. We also recall that $x \in C$ is optimal if and only if $-\nabla f(x)$ belongs to the normal cone of C at x. Consequently, the FW gap has the following properties for $x \in C$, see also [7, Section 5.1]:

- $G(x)$ is always nonnegative and equal to 0 if and only if x is optimal.

- $G(x) \geq f(x) - f(x_{\mathrm{opt}})$ holds.

- $G(x_k) = \langle -\nabla f(x_k), d_k \rangle$ for the k-th iterate in Algorithm 1.

In particular, under convexity, $G(x)$ is an upper bound to the optimality gap $f(x) - f(x_{\mathrm{opt}})$. Also, $G(x_k)$ can be calculated easily at each iteration. Because of these properties, $G(x_k)$ is often used as a stopping criterion.

We end this section with some known convergence result regarding Algorithm 1.

Theorem 2.4. *If f is convex and step size α_k is given by one of the following rules, the sequence $\{x_k\}$ generated by Algorithm 1 satisfies*

$$f(x_k) - f_{\mathrm{opt}} = O(1/k).$$

- *diminishing step size rule [24, Theorem 1]: $\alpha_k := \frac{2}{k+2}$*

- *exact line search [11, Theorem 3.1]: $\alpha_k := \arg\min_{\alpha \in [0,1]} f(x_k + \alpha d_k)$*

- *Lipschitz constant dependent step size [28, Theorem 6.1]: $\alpha_k := \min\left\{-\frac{\langle \nabla f(x_k), d_k \rangle}{L\|d_k\|^2}, 1\right\}$, where L is a Lipschitz constant of ∇f,*

Finally, we mention in passing that there are also convergence results about the FW gap, e.g., see [24].

3 Projections and distance functions

Let $\Lambda(p, e) \subseteq \mathbb{R}^n$ be a hyperbolicity cone which we will denote simply by Λ. In this section, we present a few theoretical results about the projection operator $P_\Lambda(\cdot)$ and the distance function $\mathrm{dist}\,(\cdot, \Lambda)$, with an important caveat that they are considered with respect a certain norm derived from p as follows

$$P_\Lambda(x) := \arg\min_{y \in \Lambda}\|x - y\|, \qquad \mathrm{dist}\,(x, \Lambda) := \min_{y \in \Lambda}\|x - y\|,$$

where $\|\cdot\|$ and the underlying inner product are given by

$$\|x\| := \sqrt{\lambda_1(x)^2 + \cdots + \lambda_d(x)^2} = \|\lambda(x)\| \tag{3.1}$$

$$\langle x, y \rangle := \frac{1}{4}\|x + y\|^2 - \frac{1}{4}\|x - y\|^2,$$

which is indeed a norm under the assumption that p is complete, see [4, Theorem 4.2]. Here, we abuse the notation slightly so that $\|\lambda(x)\|$ indicates the usual Euclidean norm of $\lambda(x)$ in $\mathbb{R}^d$.

The results in the section are applicable to the so-called *isometric* hyperbolic polynomials which were initially considered in [4] and are defined as follows.

Definition 3.1 (Isometric hyperbolic polynomial, [4, Definition 5.1]). *A hyperbolic polynomial p is isometric if and only if for all $y, z \in \mathbb{R}^n$, there exists $x \in \mathbb{R}^n$ satisfying*

$$\lambda(x) = \lambda(z) \text{ and } \lambda(x + y) = \lambda(x) + \lambda(y).$$

Here are some examples of isometric hyperbolic polynomials and the corresponding inner products and norm.

Example 3.2 (The nonnegative orthant, the semidefinite cone and symmetric cones). *The nonnegative orthant $\mathbb{R}_+^n$ can be realized as a hyperbolicity cone by taking $p : \mathbb{R}^n \to \mathbb{R}$ to be the polynomial $p(x) := \prod_{i=1}^n x_i$ and $e := (1, \ldots, 1)$. With that, p is isometric and induced inner product and norm are the usual Euclidean one.*

Let $\mathcal{S}_n$ be the cone of $n \times n$ real symmetric matrices, and $\mathcal{S}_n^+$ be the cone of $n \times n$ real symmetric positive semidefinite matrices. Then, $\det$ on $\mathcal{S}_n$ is a hyperbolic polynomial w.r.t. the identity matrix I, and $\Lambda(\det, I) = \mathcal{S}_n^+$. With the usual linear algebraic spectral decomposition, one can check that $\det$ is indeed isometric. The norm induced by $\det$ is identical to the Frobenius norm, and the induced inner product is identical to the trace inner product.

More generally, every symmetric cone can be realized as a hyperbolicity cone induced by some isometric hyperbolic polynomial, see [48, Section 2.2] for more details.

3.1 The distance function to a hyperbolicity cone

Let $x \in \mathbb{R}^n$. Because x belongs to Λ if and only if $\lambda(x) \in \mathbb{R}_+^d$, a reasonable guess is that $\mathrm{dist}\,(x, \Lambda)$ satisfies

$$\mathrm{dist}\,(x, \Lambda)^2 = \sum_{i=1}^d \min(\lambda_i(x), 0)^2,$$

which does indeed hold in quite a few cases, for example, when Λ is the cone of $d \times d$ positive semidefinite real matrices. Unfortunately, we will see that proving this expression seems to require extra assumptions on p.

A general formula for $\mathrm{dist}\,(x, \Lambda)$ can be obtained by making use of the following lemma, see also [5, Proposition 5.3].

Lemma 3.3. *If a hyperbolic polynomial is isometric then $\mathrm{dist}\,(u, \lambda(x)) = \mathrm{dist}\,(\lambda^{-1}(u), x)$ holds for all $u \in \lambda(\mathbb{R}^n)$ and all $x \in \mathbb{R}^n$.*

Proof. Let z be such that $\mathrm{dist}\,(\lambda^{-1}(u), x) = \|z - x\|$ and $\lambda(z) = u$. By the isometric property, there exists y such that $\lambda(y) = \lambda(z) = u$ and $\lambda(y + x) = \lambda(y) + \lambda(x)$. Simplifying the equality $\|y + x\|^2 = \|\lambda(y + x)\|^2 = \|\lambda(y) + \lambda(x)\|^2$ leads to

$$\langle x, y \rangle = \langle \lambda(x), \lambda(y) \rangle = \langle \lambda(x), \lambda(z) \rangle. \tag{3.2}$$

Recalling that $\langle z, x \rangle \leq \langle \lambda(z), \lambda(x) \rangle$ holds (see [5, Proposition 4.4]), we have

$$
\begin{aligned}
\mathrm{dist}\,(\lambda^{-1}(u), x)^2 &= \|z - x\|^2 \\
&= \|z\|^2 - 2\langle z, x \rangle + \|x\|^2 \\
&\geq \|z\|^2 - 2\langle \lambda(z), \lambda(x) \rangle + \|x\|^2 \\
&= \|y\|^2 - 2\langle \lambda(y), \lambda(x) \rangle + \|x\|^2,
\end{aligned}
$$

where the last equality follows from $\|y\|^2 = \|\lambda(y)\|^2 = \|\lambda(z)\|^2 = \|z\|^2$ and $\lambda(y) = \lambda(z)$. Then, in view of (3.2), we obtain

$$\mathrm{dist}\,(\lambda^{-1}(u), x)^2 \geq \|x - y\|^2.$$

Since $\lambda(y) = \lambda(z) = u$ holds we have in fact $\mathrm{dist}\,(\lambda^{-1}(u), x)^2 = \|x - y\|^2$. Recalling (3.2), this leads to

$$\mathrm{dist}\,(\lambda^{-1}(u), x)^2 = \|x - y\|^2 = \|\lambda(y)\|^2 - 2\langle \lambda(y), \lambda(x) \rangle + \|\lambda(x)\|^2 = \|\lambda(x) - \lambda(y)\|^2 = \mathrm{dist}\,(u, \lambda(x))^2.$$

$\square$

Proposition 3.4 (The distance to an isometric hyperbolicity cone)**.** *Let $\Lambda = \Lambda(p,e)$ be a complete hyperbolicity cone, p an isometric hyperbolic polynomial and let $x \in \mathbb{R}^n$. We have*

$$\operatorname{dist}(x, \Lambda) = \inf_{u \in \lambda(\Lambda)} \|\lambda(x) - u\|.$$

In particular, if $\lambda(\Lambda) = \mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow$ holds, then $\operatorname{dist}(x, \Lambda)^2 = \sum_{i=1}^d \min(\lambda_i(x), 0)^2$.

Proof. First, we note that that Λ can be written as an union of sets that correspond to elements that have the same eigenvalues, i.e., $\Lambda = \bigcup_{u \in \lambda(\Lambda)} \lambda^{-1}(u)$. With that, we have

$$\begin{aligned}
\operatorname{dist}(x, \Lambda) = \min_{y \in \Lambda} \operatorname{dist}(x, y) &= \min_{u \in \lambda(\Lambda)} \left[\min_{y \in \lambda^{-1}(u)} \operatorname{dist}(y, x) \right] \\
&= \min_{u \in \lambda(\Lambda)} \left[\operatorname{dist}(\lambda^{-1}(u), x) \right] \\
&= \min_{u \in \lambda(\Lambda)} \operatorname{dist}(u, \lambda(x)),
\end{aligned} \tag{3.3}$$

where the last equality follows from Lemma 3.3. This shows the first half of the proposition. If additionally $\lambda(\Lambda) = \mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow$ holds, we have

$$\min_{u \in \lambda(\Lambda)} \operatorname{dist}(u, \lambda(x)) = \min_{u \in \mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow} \|\lambda(x) - u\| \geq \min_{u \in \mathbb{R}^d_+} \|\lambda(x) - u\| = \left\| \lambda(x) - \lambda(x)^+ \right\|, \tag{3.4}$$

where $\lambda(x)^+$ is the projection of $\lambda(x)$ onto $\mathbb{R}^d_+$, which is obtained by zeroing the negative components of $\lambda(x)$. Since $\lambda(x) \in \mathbb{R}^d_\downarrow$ holds, we also have $\lambda(x)^+ \in \mathbb{R}^d_\downarrow$. We conclude that

$$\left\| \lambda(x) - \lambda(x)^+ \right\| \geq \min_{u \in \mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow} \|\lambda(x) - u\|. \tag{3.5}$$

From (3.4) and (3.5) we have $\|\lambda(x) - \lambda(x)^+\| = \min_{u \in \lambda(\Lambda)} \operatorname{dist}(u, \lambda(x))$. This, together with (3.3), leads to $\operatorname{dist}(x, \Lambda) = \|\lambda(x) - \lambda(x)^+\|$ which implies $\operatorname{dist}(x, \Lambda)^2 = \sum_{i=1}^d \min(\lambda_i(x), 0)^2$. □

We now take a look at the requirement that $\lambda(\Lambda)$ coincides with $\mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow$. By construction, $\lambda(\Lambda)$ is always contained in $\mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow$, so the nontrivial part is the opposite containment. The condition $\lambda(\Lambda) = \mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow$ does not hold in general, but it is connected to another condition that was used to prove some of the results of [4] about conjugacy of spectral functions, see [4, Section 5].

Lemma 3.5. *Suppose that $p : \mathbb{R}^n \to \mathbb{R}$ is an isometric hyperbolic polynomial. We have $\lambda(\Lambda) = \mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow$ if and only if $\lambda(\mathbb{R}^n) = \mathbb{R}^d_\downarrow$.*

Proof. Suppose $\lambda(\mathbb{R}^n) = \mathbb{R}^d_\downarrow$. As we have observed, $\lambda(\Lambda) \subseteq \mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow$ always holds, so let us prove the opposite containment. If $u \in \mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow$, since $\lambda(\mathbb{R}^n) = \mathbb{R}^d_\downarrow$, there exists $x \in \mathbb{R}^n$ such that $\lambda(x) = u$. Since $u \in \mathbb{R}^d_+$, such an x must belong to Λ.

Next, suppose that $\lambda(\Lambda) = \mathbb{R}^d_+ \cap \mathbb{R}^d_\downarrow$. Since we always have $\lambda(\mathbb{R}^n) \subseteq \mathbb{R}^d_\downarrow$, let us prove the opposite containment. Let $u \in \mathbb{R}^d_\downarrow$. Then, we can write u as the sum of two elements in $\mathbb{R}^d_\downarrow$ corresponding to the positive and negative components of u. We have

$$u = u^+ + u^-.$$

Since $u^+ \in \mathbb{R}^d_+$ as well, by hypothesis, there exists $y \in \Lambda$ such that $\lambda(y) = u^+$. Similarly, there is $z \in \Lambda$ such that $\lambda(z) = (-u^-)^\downarrow$. We note that since the eigenvalues of $-z$ are the negatives of the eigenvalues of z, we have $\lambda(-z) = u^-$.

Since p is assumed to be isometric, there exists x such that $\lambda(x) = \lambda(-z)$ and $\lambda(x + y) = \lambda(x) + \lambda(y)$. That is, we have

$$\lambda(x + y) = \lambda(x) + \lambda(y) = u^- + u^+ = u.$$

□

9

3.2 Partial results on the projection operator

Having discussed distance functions, we now take a look at how to actually project an arbitrary point onto a hyperbolicity cone. If Λ is the cone of positive semidefinite matrices, computing the projecting is easy: we just compute a spectral decomposition of x and zero negative eigenvalues. However, when Λ is an arbitrary hyperbolicity cone, the analogy to positive semidefinite matrices does not seem to take us very far. A difficulty is that, although we have a generalized notion of eigenvalues, there is no obvious notion of "eigenvectors". Similarly, no obvious notion of spectral decomposition exists.

That said, if each eigenvalue of $\lambda_i(x)$ is simple, i.e., a root of multiplicity one for the polynomial univariate polynomial $t \mapsto p(x - te)$, the projection of x onto Λ can be represented in closed form. We use results of [4] in our proof.

Proposition 3.6. *Let $\Lambda = \Lambda(p, e)$ be a complete hyperbolicity cone and suppose that p is isometric and $\lambda(\mathbb{R}^n) = \mathbb{R}^d_{\downarrow}$ holds. Define $f : \mathbb{R}^d \to \mathbb{R}$ as $f(x) := \frac{1}{2} \sum_{i=1}^d \max\{x_i, 0\}^2$. Then, $f \circ \lambda$ is convex differentiable, and for all $x \in \mathbb{R}^n$,*

$$P_\Lambda(x) = \nabla(f \circ \lambda)(x).$$

Proof. Under the stated assumptions, [4, Theorem 3.9] implies convexity of $f \circ \lambda$ and [4, Theorem 5.5] leads to the following equivalence.

$$y \in \partial(f \circ \lambda)(x) \iff \lambda(y) \in \partial f(\lambda(x)) \text{ and } \langle x, y \rangle = \langle \lambda(x), \lambda(y) \rangle. \tag{3.6}$$

Because $\|\lambda(x)\| = \|x\|$ holds, expanding $\|x - y\|^2$ we also obtain the following equivalence

$$\langle x, y \rangle = \langle \lambda(x), \lambda(y) \rangle \iff \|\lambda(x) - \lambda(y)\| = \|x - y\|. \tag{3.7}$$

By definition, f is differentiable and ∇f is the projection operator onto $\mathbb{R}^n$, i.e.

$$\nabla f = P_{\mathbb{R}^d_+}. \tag{3.8}$$

By (3.7) and (3.8), (3.6) can be rewritten as

$$y \in \partial(f \circ \lambda)(x) \iff \lambda(y) = P_{\mathbb{R}^d_+}(\lambda(x)) \text{ and } \|\lambda(x) - \lambda(y)\| = \|x - y\|. \tag{3.9}$$

Let $y \in \partial(f \circ \lambda)(x)$, then in view of (3.9) we have $\|x - y\|^2 = \sum_{i=1}^d \min(0, \lambda_i(x))^2$. Since $\lambda(y) \in \mathbb{R}^d_+$, we have $y \in \Lambda$, so by Lemma 3.5 and Proposition 3.4, y is precisely the projection of x onto Λ.

Since the projection onto a closed convex set is unique, the subdifferential $\partial(f \circ \lambda)(x)$ is a singleton. This means that $f \circ \lambda$ is differentiable at x and $\nabla(f \circ \lambda)$ is the projection onto Λ. $\square$

Corollary 3.7. *In addition to the assumption in Proposition 3.6, suppose each eigenvalue $\lambda_i(x)$ is simple. Then,*

$$P_\Lambda(x) = \sum_{i=1}^d \max(\lambda_i(x), 0) \frac{1}{p^{(1)}(x - \lambda_i(x)e)} \nabla p(x - \lambda_i(x)e)$$

Proof. If each eigenvalue $\lambda_i(x)$ is simple, then $x \mapsto \lambda_i(x)$ is analytic in a neighbourhood of x, and

$$\nabla \lambda_i(x) = \frac{1}{p^{(1)}(x - \lambda_i(x)e)} \nabla p(x - \lambda_i(x)e) \tag{3.10}$$

holds, see Section 3.1 and Equation (3.5) in [43]. From Proposition 3.6 we have

$$P_\Lambda(x) = \nabla(f \circ \lambda)(x).$$

From the definition of f and (3.10) we have

$$P_\Lambda(x) = \nabla(f \circ \lambda)(x) = \sum_{i=1}^{d} \max(\lambda_i(x), 0) \nabla \lambda_i(x)$$

$$\sum_{i=1}^{d} \max(\lambda_i(x), 0) \frac{1}{p^{(1)}(x - \lambda_i(x)e)} \nabla p(x - \lambda_i(x)e).$$

$\square$

3.3 Limitations and further discussion

The results we proved so far on the distance function and the projection operator are restricted to isometric hyperbolic polynomials and the sharpest results require the provision that $\lambda(\mathbb{R}^n) = \mathbb{R}^d_\downarrow$. Unfortunately there are at least three issues.

   The first is that it is not known how large the class of isometric hyperbolic polynomials is and even hyperbolic polynomials associated to polyhedral cones may fail to be isometric. In a sense, this is not surprising because the same cone can be generated by different hyperbolic polynomials, e.g., $\mathbb{R}^3_+$ also satisfies $\mathbb{R}^3_+ = \Lambda(p, (1, 1, 1))$ for $p(x_1, x_2, x_3) = x_1^2 x_2^2 x_3^2$. An example of non-isometric polynomial is given in [4, Example 5.2], however it is less than ideal because there there are "redundancies" in the description of the underlying cone[1].

   The way to ensure that there are no redundancies is to restrict ourselves to *minimal polynomials*, which we will now recall. For simplicity, assume that Λ is a regular hyperbolicity cone. Although Λ can be generated by different hyperbolic polynomials, there exists a hyperbolic polynomial p of minimal degree that generates Λ with the property that p divides any other hyperbolic polynomial $\hat{p}$ satisfying $\Lambda = \Lambda(\hat{p}, e)$, e.g., see [19, Lemma 2.1] for a more general result or see the discussion in [17, Section 2.2]. We will call such a p a *minimal degree polynomial for* Λ. So the more interesting question is whether *minimal* hyperbolic polynomials can fail to be isometric. Unfortunately, that is indeed possible.

Proposition 3.8 (A non-isometric minimal polynomial associated to a polyhedral cone)**.** *Let $p : \mathbb{R}^3 \to \mathbb{R}$ be the polynomial defined as*

$$p(x) = (x_1 + x_2 + x_3)(x_1 - x_2 + x_3)(2x_1 - x_2 - x_3)(x_1 + 2x_2 - x_3).$$

p is a hyperbolic polynomial along $e = (0, 0, 1)$, and the hyperbolicity cone for (p, e) is the polyhedral cone satisfying $\Lambda(p, e) = \{x \in \mathbb{R}^3 \mid x_1 + x_2 + x_3 \geq 0, x_1 - x_2 + x_3 \geq 0, -2x_1 + x_2 + x_3 \geq 0, -x_1 - 2x_2 + x_3 \geq 0\}$. The polynomial p is minimal for $\Lambda(p, e)$ but p is not isometric.

Proof. The roots of $p(x - te)$ are

$$r_1(x) = x_1 + x_2 + x_3, \qquad r_2(x) = x_1 - x_2 + x_3,$$
$$r_3(x) = -2x_1 + x_2 + x_3, \ r_4(x) = -x_1 - 2x_2 + x_3.$$

In order for x to belong to $\Lambda(p, e)$ all the roots must be nonnegative. This gives the expression for $\Lambda(p, e)$ in the statement of the proposition.

   Next, let q be a minimal hyperbolic polynomial for $\Lambda(p, e)$ so that q divides p and $\Lambda(p, e) = \Lambda(q, e)$ holds. Since p is a product of four degree 1 polynomials, q must be a product of *some* of those four polynomials. Suppose that p is not of minimal degree. Then, q cannot have degree 4, so, up to a constant, it must be a product of *strictly less than* four degree 1 polynomials among the ones that appear in the decomposition of p. Therefore, in order to show that p is of minimal degree, we only need to argue that removing *any* of the polynomials that appear in the decomposition of p will result in a larger cone. We do this case by case.

---

[1]Example 5.2 in [4] corresponds to the restriction of the polynomial $p(x_1, x_2, x_3) = x_1 x_2 x_3$ to a certain two-dimensional space. Every pointed two-dimensional closed convex cone is isomorphic to $\mathbb{R}^2_+$, which only requires a degree 2 hyperbolic polynomial. So, there is indeed a redundancy in the expression provided for Example 5.2 in [4].

If q omits the factor $x_1 + x_2 + x_3$, then $\Lambda(q,e)$ contains $(-1,-1,1) \notin \Lambda(p,e)$. If q omits $x_1 - x_2 + x_3$, then $\Lambda(q,e)$ contains $(-1,1,1) \notin \Lambda(p,e)$. If q omits $2x_1 - x_2 - x_3$, then $\Lambda(q,e)$ contains $(1,-1,1) \notin \Lambda(p,e)$. Finally, if q omits $x_1 + 2x_2 - x_3$, then $\Lambda(q,e)$ contains $(1,1,2) \notin \Lambda(p,e)$.

The conclusion is that in order for q to be minimal it cannot omit any of the degree 1 factors of p, so q must have degree 4 and p is a minimal degree polynomial as well.

Next, we sketch a proof that p is not isometric, see Appendix A for more details. Let $z = (3,1,0)$ and $y = (-1,0,0)$. With that, $\lambda(z) = (4,2,-5,-5)$ holds, so z has an eigenvalue of multiplicity two. This implies that if $\lambda(w) = \lambda(z)$ holds for some w, at least two among the $r_1(w), r_2(w), r_3(w), r_4(w)$ must be the same. By considering all possible cases, we can show the following implication

$$\lambda(w) = \lambda(z) \iff w = z,$$

see Appendix A for more details. Moreover, $\lambda(z + y) \neq \lambda(z) + \lambda(y)$. Thus, p is not isometric. $\qquad\square$

The second issue is that even if a polynomial is isometric, it is not necessarily the case $\lambda(\mathbb{R}^d) = \mathbb{R}^d_\downarrow$ holds. Such an example has already being considered in [4, Section 6], see the "Singular values" part and consider Example 6.1 in view of Theorem 5.4 in [4].

The third and final issue is that even in the cases that the result in this section apply in full, they are only valid to the projection computed with respect the norm induced by the hyperbolic polynomial in (3.1). Naturally, the norm in (3.1) can be different from the usual Euclidean norm in $\mathbb{R}^n$, which may limit the applicability of the results.

In view of these limitations, we believe it is important to also consider numerical algorithms for computing projections onto hyperbolicity cones. This leads us to the next section.

4 A FW algorithm for strongly convex optimization over regular cones

In this section we develop a numerical method that is able to handle strong convex optimization over any regular closed convex cone $\mathcal{K}$. More precisely, we aim to solve the following pair of primal-dual problems:

$$\begin{array}{ll} \min_{x \in \mathbb{R}^n} & f(x) \\ \text{s.t.} & Tx + b \in \mathcal{K} \end{array} \quad \text{(P)} \qquad\qquad \begin{array}{ll} \min_{y \in \mathbb{R}^m} & (f^* \circ T^*)(y) + \langle b, y \rangle \\ \text{s.t.} & y \in \mathcal{K}^*, \end{array} \quad \text{(D)}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a closed proper μ-strongly convex function, f^* is its conjugate, $T : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map, T^* is its adjoint and $b \in \mathbb{R}^m$ is a vector.

Before we move on further, some remarks are in order. We suppose that $\mathbb{R}^m$ is equipped with some arbitrary inner product $\langle \cdot, \cdot \rangle$ for which the corresponding norm is given by $\|\cdot\|$. A similar remark applies to $\mathbb{R}^n$ and while $\mathbb{R}^m$ and $\mathbb{R}^n$ may have different inner products, for simplicity we will use the same symbols $\langle \cdot, \cdot \rangle$, $\|\cdot\|$ to denote the inner product and the norm in both spaces. Furthermore, the conjugate f^*, the adjoint T^* and the dual cone $\mathcal{K}^*$ are, of course, computed with respect to the inner product in the spaces that they are defined. That said, in contrast to Section 3, one important point is that, even if $\mathcal{K}$ is a hyperbolicity cone, we will *not* require that $\|\cdot\|$ be the norm induced by underlying hyperbolic polynomial p.

As mentioned in the Section 1, (P) contains as a particular case the projection problem, since we can take $b = 0$, let T be the identity map and $f(x) := \|x - x_0\|^2$, where x_0 is fixed and $\|\cdot\|$ is the norm induced by some inner product on $\mathbb{R}^n$. However, (P) will also allow us to handle more general quadratic minimization problems, as we shall see in Section 4.4.

The problem (D), although written as a minimization problem, is actually equivalent to the Fenchel dual of (P) and for completeness we show below its derivation. First, we observe that (P) can be expressed as follows using indicator functions:

$$\min_{x \in \mathbb{R}^n} f(x) + \delta_{\mathcal{K}-b}(Tx). \tag{4.1}$$

Then, the Fenchel dual problem of (4.1) is

$$\max_{y \in \mathbb{R}^m} -f^*(T^*y) - \delta^*_{\mathcal{K}-b}(-y), \tag{4.2}$$

e.g., see [44, Corollary 31.2.1]. Here, for all $y \in \mathbb{R}^m$,

$$\delta^*_{\mathcal{K}-b}(y) = \sup_{x \in \mathbb{R}^m} \{\langle x, y \rangle - \delta_{\mathcal{K}-b}(x)\} = \langle -b, y \rangle + \delta_{\mathcal{K}^\circ}(y),$$

where $\mathcal{K}^\circ = -\mathcal{K}^*$ is the polar of $\mathcal{K}$. Hence, $\delta^*_{\mathcal{K}-b}(-y) = \langle b, y \rangle + \delta_{\mathcal{K}^*}(y)$. So, (4.2) is equivalent to (D).

Optimality conditions for (P) **and** (D)   We now briefly discuss the optimality conditions for (P) and (D), which follow from classic convex duality theory. First, we show that there is no duality gap and both problems are attained under mild conditions. Here, we say that a problem is *attained* if there exists a feasible solution whose objective function value is equal to the optimal value of the problem.

Proposition 4.1 (No duality gap). *Let p^* and d^* denote the optimal values of* (P) *and* (D). *The following items hold.*

(i) $p^* + d^* = 0$.

(ii) *If* (P) *is feasible, then p^* and d^* are finite and p^* is attained.*

(iii) *If* (P) *satisfies Slater's condition (i.e., there exists $\bar{x}$ such that $T\bar{x} + b \in \mathrm{int}\,\mathcal{K}$), then p^* and d^* are both finite and attained.*

Proof. Since f is assumed to be strongly convex over $\mathbb{R}^n$, the domain of f^* is $\mathbb{R}^n$, see [21, Theorem 4.2.1]. Because $\mathcal{K}$ is pointed, $\mathcal{K}^\circ$ has an interior point. We note also that $\mathrm{int}\,(\mathcal{K})^\circ = \mathrm{int}\,\mathrm{dom}\,\delta^*_{\mathcal{K}-b}$. Thus any point in $y \in \mathrm{int}\,\mathrm{dom}\,\delta^*_{\mathcal{K}-b}$ is such that

$$T^*y \in \mathrm{ri}\,(\mathrm{dom}\,f^*) = \mathbb{R}^n \tag{4.3}$$

Under (4.3), we can invoke an appropriate version of Fenchel's duality theorem (e.g., [44, Corollary 31.2.1]) to conclude that (P) and (4.2) have the same optimal value and p^* is attained if finite. In particular if (P) is feasible, then, since (4.2) is feasible as well, p^* is attained. Recalling that the optimal value of (4.2) is $-d^*$, we conclude that $p^* + d^* = 0$. This proves items (i) and (ii).

   Furthermore, if (P) satisfies Slater condition, the same Fenchel's duality theorem ensures that d^* is attained, which proves item (iii). □

   From Proposition 4.1, there is no duality gap between (P) and (D) and, as long as Slater's condition is satisfied at (P), both problems have optimal solutions x_{opt} and y_{opt}, respectively. Since f is strongly convex, x_{opt} is unique. In any case, the solutions are related by the formulae:

$$T^*y_{\mathrm{opt}} \in \partial f(x_{\mathrm{opt}}) \tag{4.4}$$
$$Tx_{\mathrm{opt}} \in \partial \delta^*_{\mathcal{K}-b}(-y_{\mathrm{opt}}), \tag{4.5}$$

e.g., see [44, pg. 333 and Theorem 31.3]. Since we assumed that f is a closed proper convex function, (4.4) is equivalent to $x_{\mathrm{opt}} \in \partial f^*(T^*y_{\mathrm{opt}})$, e.g., [44, Theorem 23.5]. Moreover, f^* is differentiable because of the strong convexity of f. In the end, (4.4) is equivalent to

$$x_{\mathrm{opt}} = \nabla f^*(T^*y_{\mathrm{opt}}). \tag{4.6}$$

Similarly, (4.5) is equivalent to $-y_{\mathrm{opt}} \in \partial \delta_{\mathcal{K}-b}(Tx_{\mathrm{opt}})$. An important consequence of (4.6) is that the unique optimal solution of (4.1) can be derived from *any* optimal solution to the dual problem (D).

13

4.1 Overcoming the challenges of constructing a Frank-Wolfe based method

When applying a Frank-Wolfe method there are a few challenges a practitioner must handle in order to obtain an efficient algorithm and to ensure convergence. In this subsection, we discuss these issues one by one. Here, we recall our standing assumption that $\mathcal{K}$ is a regular closed convex cone.

4.1.1 Issue 1: Primal or dual?

The first issue is to decide which side of the problem to solve: (P) or (D). We are mainly interested in (P), but solving (D) would also be enough in view of the fact that an optimal solution to (P) can be obtained via (4.6).

That said, in order for the FW method to be efficient, we need to be efficient in solving the subproblem "$\arg\min_{x \in C} \langle \nabla f(x_k), x \rangle$" that appears in Algorithm 1. So the decision to apply FW to either (P) or (D) depends on which side is more likely to afford an easily solvable subproblem.

After carefully examining this question, it seemed to us that solving (D) is more promising because the FW subproblem in this case can be connected to the generalized minimum eigenvalue problem over a regular cone, as we shall discuss in Section 4.1.3.

Having settled for (D), we discuss in the sequel two outstanding issues.

4.1.2 Issue 2: Compactness of the feasible region

The classical FW method requires the compactness of the feasible region, so it can not be applied directly to (D). However, this can be fixed by adding a constraint that cuts a compact slice of the feasible region of (D) in such a way that at least one optimal solution is inside the slice. In order to do that, we consider the following assumption.

Assumption 1. *We assume that $e \in \operatorname{int} \mathcal{K}$ and $c_D > 0$ are such that there exists at least one optimal solution y_{opt} to (D) satisfying $\langle e, y_{\mathrm{opt}} \rangle \leq c_D$.*

Under Assumption 1 we can show that

$$
\begin{aligned}
\min_{y \in \mathbb{R}^m} \quad & (f^* \circ T^*)(y) + \langle b, y \rangle \\
\text{s.t.} \quad & \langle e, y \rangle \leq c_D \\
& y \in \mathcal{K}^*
\end{aligned}
\tag{4.7}
$$

has a compact feasible region, and, by assumption, (4.7) has at least one optimal solution of (D) among its optimal solutions.

Proposition 4.2. *The non-empty convex set*

$$
\{ y \in \mathcal{K}^* \mid \langle e, y \rangle \leq c_D \}
\tag{4.8}
$$

is compact. Moreover, (4.8) contains at least one optimal solution of (D) under Assumption 1.

Proof. First, since 0 is always contained in (4.8), the set in (4.8) is non-empty. It is also closed and convex because $\mathcal{K}^*$ is a closed convex cone. Next, we prove the boundedness of (4.8) by checking that the recession cone of (4.8) is trivial. Since the set in (4.8) contains zero, its recession cone coincides with the set of elements y such that λy belong to (4.8) for all $\lambda \geq 0$, e.g., see [44, Theorem 8.3]. All such y must then belong to $\mathcal{K}^*$ and satisfy $\langle e, y \rangle \leq 0$. However, since e is an interior point of $\mathcal{K}$, we have $\langle e, y \rangle = 0$ which forces $y = 0$. Finally, (4.8) contains at least one optimal solution of (D) by Assumption 1. $\qquad\square$

In view of Proposition 4.2, at least one of the optimal solutions of (D) can be obtained by solving (4.7). We also note that since the objective functions in (4.7) and (D) are the same, the set of optimal solutions of the former is included in the optimal solution set of the latter. However, (4.7) has a compact feasible region so it is amenable to the classical FW method.

Of course, for a given problem, the crux of the issue is whether we can easily obtain e and c_D as in Assumption 1. As we will show in Section 4.4, c_D can be computed explicitly from the problem data in the case of quadratic optimization, so the assumption of having c_D at hand will not be problematic for our purposes.

4.1.3   Issue 3: How to solve the subproblem exactly?

Our current state of affairs is as follows. Having decided to apply a FW method to the dual side of our problem of interest, we showed that it is enough to solve (4.7), which is a compact version of (D) containing at least one of its optimal solutions. Next we examine whether this choice indeed leads to easy subproblems. Applying Algorithm 1 to the problem (4.7) leads to the following subproblem at each iteration:

$$
\begin{aligned}
\min_{s \in \mathbb{R}^m} \quad & \langle \nabla(f^* \circ T^*)(y_k) + b, s \rangle \\
\text{s.t.} \quad & \langle e, s \rangle \leq c_D \\
& s \in \mathcal{K}^*,
\end{aligned}
\tag{4.9}
$$

where, we use y_k in place of x_k since we are working from the dual side. In what follows, it will be helpful to define

$$
x_k := \nabla f^*(T^* y_k).
$$

With that, we have $\nabla(f^* \circ T^*)(y_k) + b = T\nabla f^*(T^* y_k) + b = Tx_k + b$. Also, we transform the inequality constraint into a equality constraint by using a slack variable α. With that, we arrive at the following subproblem, which is equivalent to (4.9).

$$
\begin{aligned}
\min_{s \in \mathbb{R}^m, \alpha \in \mathbb{R}} \quad & \langle Tx_k + b, s \rangle \\
\text{s.t.} \quad & \langle e, s \rangle + \alpha = c_D \\
& s \in \mathcal{K}^*, \alpha \in \mathbb{R}_+
\end{aligned}
\tag{4.10}
$$

The problem (4.10) is a common conic linear program in primal format. The goal of this subsection is to show that an optimal solution of (4.10) can be written explicitly in terms of the the corresponding generalized eigenvalue function. In order to do so, we consider the dual problem of (4.10).

$$
\begin{aligned}
\max_{t \in \mathbb{R}, z \in \mathbb{R}^m} \quad & c_D t \\
\text{s.t.} \quad & z = (Tx_k + b) - te \\
& z \in \mathcal{K}, t \leq 0.
\end{aligned}
\tag{4.11}
$$

The problem in (4.11) is closely related to the minimum eigenvalue problem in (2.2). As hinted in Section 4.1.1, this is why solving our problem of interest from the dual side makes sense: when doing so, we arrive at a subproblem whose optimal value can be obtained from a minimum eigenvalue computation. We are now positioned to show our main theorem for this subsection.

Theorem 4.3 (Closed-form solution of the FW subproblem)**.** *Consider the primal-dual pair of problems* (4.10) *and* (4.11)*. Then, the following statements hold.*

(i) *Both* (4.10) *and* (4.11) *satisfy Slater's condition. In particular, the optimal values of both problems coincide and are attained.*

(ii) *The optimal solution of* (4.11) *is given by*

$$
\begin{aligned}
t_{\text{opt}} &= \min(0, \lambda_{\min}(Tx_k + b)) \\
z_{\text{opt}} &= Tx_k + b - t_{\text{opt}} e,
\end{aligned}
\tag{4.12}
$$

where $\lambda_{\min}$ is the minimum eigenvalue function along the direction e, as in (2.2)*.*

(iii) If $t_{\mathrm{opt}} = 0$, then $(0, c_D) \in \mathbb{R}^m \times \mathbb{R}$ is an optimal solution to (4.10). Otherwise, if $t_{\mathrm{opt}} < 0$, the optimal solution set of (4.10) with respect to the s variable is

$$\{s \in \mathcal{K}^* \mid \langle e, s \rangle = c_D, \ \langle s, z_{\mathrm{opt}} \rangle = 0\} = \{s \in \mathcal{F}^{\Delta}_{z_{\mathrm{opt}}} \mid \langle e, s \rangle = c_D\},$$

where $\mathcal{F}^{\Delta}_{z_{\mathrm{opt}}}$ is the conjugate face of $\mathcal{K}$ at z_{opt} as in (2.1).

Proof. $\boxed{\text{Item } (i)}$ First we check that (4.10) satisfies Slater's condition. $\mathcal{K}$ is a regular cone by assumption, so $\operatorname{int} \mathcal{K}^*$ is not empty. Let $s \in \operatorname{int} \mathcal{K}^*$. If $\langle e, s \rangle = 0$, then since e is an interior point of $\mathcal{K}$, we have $s = 0$, which is impossible since $0 \in \operatorname{int} \mathcal{K}^*$ implies $\mathcal{K}^* = \mathbb{R}^m$. Therefore, s satisfies $\langle e, s \rangle > 0$. Let

$$\hat{s} := \frac{c_D}{2\langle e, s \rangle} s.$$

Then, $\hat{s} \in \operatorname{int} \mathcal{K}^*$ and $\langle e, \hat{s} \rangle = \frac{c_D}{2}$ hold. Therefore, $(\hat{s}, c_D/2)$ is a strictly feasible solution to (4.10).

Next, we check that (4.11) satisfies Slater's condition. Since $e \in \operatorname{int} \mathcal{K}$, there exists a small $u > 0$ such that $(Tx_k + b)u + e \in \operatorname{int} \mathcal{K}$. Therefore, for $t := -1/u$, we have $Tx_k + b - te \in \operatorname{int} \mathcal{K}$ and $(Tx_k + b - te, t)$ is a strictly feasible solution.

$\boxed{\text{Item } (ii)}$ We divide the proof in two cases.

$\boxed{\text{Case } (a): \ Tx_k + b \in \mathcal{K}}$ In this case, we have $\langle Tx_k + b, s \rangle \geq 0$ for every $s \in \mathcal{K}^*$. Since $(0, c_D)$ is feasible for (4.10), $(0, c_D)$ is an optimal solution of (4.10). Because of item (i) the optimal value of (4.11) must be zero as well, so $t_{\mathrm{opt}} = 0$ which coincides with $\min(0, \lambda_{\min}(Tx_k + b))$.

$\boxed{\text{Case } (b): \ Tx_k + b \notin \mathcal{K}}$ Let t_{opt} denote the optimal solution of (4.11), which exists and is finite because of item (i). By definition of $\lambda_{\min}$ (see (2.2)), we have $t_{\mathrm{opt}} \leq \lambda_{\min}(Tx_k + b)$, since the problem in (4.11) has one additional constraint in comparison to the problem in (2.2). However, since $Tx_k + b \notin \mathcal{K}$, the inequality $\lambda_{\min}(Tx_k + b) < 0$ holds by item (ii) of Proposition 2.2. Therefore, $t := \lambda_{\min}(Tx_k + b)$ and $z := Tx_k + b - te$ is feasible for (4.11), so $t_{\mathrm{opt}} = \lambda_{\min}(Tx_k + b)$. Therefore, in this case too, the formula in (4.12) holds.

$\boxed{\text{Item } (iii)}$ If $t_{\mathrm{opt}} = \min(0, \lambda_{\min}(Tx_k + b)) = 0$, then $\lambda_{\min}(Tx_k + b) \geq 0$ which implies that $Tx_k + b \in \mathcal{K}$, by Proposition 2.2. In this case, we already verified in Case (a) of item (ii) that $(0, c_D)$ is an optimal solution to (4.10). Suppose that $t_{\mathrm{opt}} < 0$. Since both (4.10) and (4.11) satisfy Slater's condition, the following conditions from classical conic linear programming duality theory are necessary and sufficient for optimality

$$\langle z, s \rangle - t\alpha = 0 \tag{4.13}$$
$$\langle e, s \rangle + \alpha = c_D$$
$$(Tx_k + b) - te = z$$
$$z \in \mathcal{K}, s \in \mathcal{K}^*, t \leq 0, \alpha \geq 0.$$

Therefore, if $t_{\mathrm{opt}} < 0$, then complementary slackness (i.e., (4.13)) and $\alpha \geq 0$ implies that the optimal α^* in (4.10) is 0. In particular, the $s \in \mathcal{K}^*$ that are optimal for (4.10) are exactly the ones that satisfy $\langle e, s \rangle = c_D$ and $\langle s, z_{\mathrm{opt}} \rangle = 0$. $\qquad\square$

So far, we have shown that we can obtain the optimal *value* of (4.11) through a minimum eigenvalue computation. However, we still require an optimal *solution* for the case where $t_{\mathrm{opt}} < 0$. Fortunately, from item (iii) of Theorem 4.3, we see that it is enough to find a nonzero $s \in \mathcal{F}^{\Delta}_{z_{\mathrm{opt}}}$ and rescale s so that $\langle e, s \rangle = c_D$ holds. This leads us to our final point in this subsection.

Conjugate vector computations In view of Theorem 4.3, the final piece we need to complete our discussion is a method to find a nonzero vector in $\mathcal{F}^{\Delta}_{z_{\mathrm{opt}}}$. Fortunately, there are many useful cones $\mathcal{K}$ for which $\mathcal{F}^{\Delta}_z$ is completely known, given a particular $z \in \mathcal{K}$. Here we list a few. For symmetric cones (this includes the case of positive semidefinite matrices and second order cones), formulae are given in [12, Theorem 2] and [32, Section 4.1.1]. For p-cones with $p \in (1, \infty)$, see [31, Section 4.1]. Power cones and

16

exponential cones are linearly isomorphic to their dual cones under the Euclidean inner product, so with some adjustments, the formulae discussed in [29, Section 3.1] and [30, Section 4.1], respectively, can also be used to determine conjugate faces for these two cones.

Unfortunately, if $\mathcal{K} = \Lambda(p, e)$ is a general hyperbolicity cone, it seems nontrivial to obtain a formula for $\mathcal{F}_z^\Delta$ given an arbitrary $z \in \Lambda(p, e)$. That said, *a nonzero conjugate vector can be obtained easily by using the following proposition.* Here, we recall that $\mathrm{mult}(z)$ denotes the number of zero eigenvalues of z and $p^{(i)}$ is the i-th directional derivative of p along a fixed hyperbolic direction, see Section 2.1.

Proposition 4.4 (Conjugate vectors in hyperbolicity cones). *Let $\Lambda = \Lambda(p, e) \subseteq \mathbb{R}^m$ be a hyperbolicity cone. Let $z \in \Lambda$ satisfy $\mathrm{mult}(z) \geq 1$ and define $r := \mathrm{mult}(z)$. Then,*

$$\nabla p^{(r-1)}(z) \in \mathcal{F}_z^\Delta \setminus \{0\}.$$

Proof. First, we observe that $z \in \Lambda$ and $\mathrm{mult}(z) \geq 1$ implies that z is in the boundary of Λ which follows, for example, from item (iii) of Proposition 2.2 and Proposition 2.3, see also [41, Section 3].

We first consider the case $r = 1$. We recall that for $y \in \Lambda$, we have $p(y) \geq 0$. In addition, $p(y) = 0$ holds if y is in the boundary of Λ. In particular, we have $p(z + ty) \geq 0$ and $p(z + tz) = 0$ for every $y \in \Lambda$ and $t \geq 0$. Then, taking the derivative with respect to t at $t = 0$, we obtain

$$\langle \nabla p(z), y \rangle \geq 0, \forall y \in \Lambda \quad \text{and} \quad \langle \nabla p(z), z \rangle = 0.$$

This shows that $\nabla p(z) \in \mathcal{F}_z^\Delta$. However, since $\mathrm{mult}(z) = 1$, $\nabla p(z)$ is nonzero (e.g., [41, Lemma 7]). Therefore, $\nabla p^{(r-1)}(x) = \nabla p(x)$ is a non-zero conjugate vector.

Next, we consider the case $r \geq 2$. We recall that

$$\partial^i \Lambda = \{x \in \Lambda \mid \mathrm{mult}(x) = i\}.$$

By definition, $x \in \partial^r \Lambda$. Using Theorem 2.1 repeatedly,

$$\partial^r \Lambda = \partial^{(r-1)} \Lambda^{(1)} = \cdots = \partial^2 \Lambda^{(r-2)} = (\partial^1 \Lambda^{(r-1)}) \cap \Lambda^{(r-2)}$$

holds. Therefore, $x \in (\partial^1 \Lambda^{(r-1)}) \cap \Lambda^{(r-2)}$. Letting $\bar{\Lambda} := \Lambda^{(r-1)}$, the multiplicity of 0 as an eigenvalue of z with respect to $p^{(r-1)}$ is one, so we can apply the previous case to $\bar{\Lambda}$ and $p^{(r-1)}$ to conclude that $\nabla p^{(r-1)}(z) \neq 0$ holds and $\nabla p^{(r-1)}(z)$ belongs to the conjugate face of $\Lambda^{(r-1)}$ at z. However, since $\Lambda \subseteq \Lambda^{(r-1)}$, we have $(\Lambda^{(r-1)})^* \subseteq \Lambda^*$. This implies that $\nabla p^{(r-1)}(z)$ belongs to the conjugate face of Λ at x. $\square$

We are now ready to complete our discussion on the optimal solutions of (4.10) for the case of a hyperbolicity cone. Under the setting of Theorem 4.3, if $t_{\mathrm{opt}} < 0$, we have $\lambda_{\min}(z_{\mathrm{opt}}) = 0$, so $\mathrm{mult}(z_{\mathrm{opt}}) \geq 1$. From Proposition 4.4 and, assuming that Λ is regular, we have that

$$\frac{c_D}{\langle e, \nabla p^{(\mathrm{mult}(z_{\mathrm{opt}})-1)}(z_{\mathrm{opt}}) \rangle} \nabla p^{(\mathrm{mult}(z_{\mathrm{opt}})-1)}(z_{\mathrm{opt}}) \tag{4.14}$$

is included in the optimal solution set, i.e., (4.14) is an optimal solution of (4.7). We recall that

$$\langle e, \nabla p^{(\mathrm{mult}(z_{\mathrm{opt}})-1)}(z_{\mathrm{opt}}) \rangle \neq 0$$

holds because $e \in \mathrm{int}\,\Lambda$ and $\nabla p^{(\mathrm{mult}(z_{\mathrm{opt}})-1)}(z_{\mathrm{opt}}) \neq 0$.

4.2 The proposed method and its convergence analysis

Having solved the issues related to applying Algorithm 1 to our problem of interest, we can now formally state our obtained method. To recap, we wish to solve (P), but we solve instead (4.7) which is a compact version of the dual problem of (P) (Proposition 4.2), under Assumption 1. Then, the idea is to apply a FW method to (4.7). During the iterations, the FW subproblem to be solved is (4.10). However, (4.10) is easier to solve from the dual side (4.11), since it reduces to a minimum eigenvalue computation (Theorem 4.3). Once

17

Algorithm 2: Dual Frank-Wolfe method for solving (P)

1: Choose initial point $y_0 \in \mathcal{K}^*$ satisfying $\langle e, y_0 \rangle \leq c_D$.
2: **for** $k = 0, 1, \ldots$ **do**
3:    $x_k := \nabla f^*(T^* y_k)$
4:    **if** y_k or x_k satisfies stopping criterion **then**
5:     Break
6:    **end if**
7:    **if** $Tx_k + b \in \mathcal{K}$ **then**
8:     $s_k = 0$
9:    **else**
10:    $z_k := Tx_k + b - \lambda_{\min}(Tx_k + b)e$
11:    Let $\hat{s}_k \in \mathcal{F}_{z_k}^{\Delta} \setminus \{0\}$ and let $s_k := \frac{c_D}{\langle e, \hat{s}_k \rangle} \hat{s}_k$. *(For the case of hyperbolicity cones, see Proposition 4.4)*
12:   **end if**
13:   $d_k := s_k - y_k$
14:   Choose $\alpha_k \in (0, 1]$ by an appropriate rule
15:   $y_{k+1} := y_k + \alpha_k d_k$
16: **end for**

---

the optimal value of (4.11) is found, we obtain an optimal solution to the subproblem (4.10) by identifying a conjugate vector which, in the case of hyperbolicity cones, can be done as in (4.14). The resulting procedure is described in Algorithm 2.

In Algorithm 2, any step size rule which guarantees the convergence of the FW method can be used (e.g., rules discussed in Theorem 2.4).

In what follows we will prove some convergence guarantees for Algorithm 2. Although the classical convergence theory of FW methods already provides some guarantees, since we are applying the method to the *dual* side of the problem, it is still necessary to show theoretical guarantees for the primal iterates x_k.

First we recall and introduce some notation. We denote by d^* the optimal value (D). Recall that under Assumption 1, d^* also coincides with the optimal value of (4.7). Also, let Y_{opt} denote the optimal set of (4.7) and let h denote the objective function of (4.7).

The analysis we will conduct in this subsection will be for the most part method agnostic. More precisely, suppose that $\{(x_k, y_k)\} \subseteq \mathbb{R}^n \times \mathbb{R}^n$ is any sequence such that the following assumption is satisfied.

Assumption A. *For every k, y_k is feasible for (4.7), $x_k = \nabla f^*(T^* y_k)$ holds and $h(y_k) \to d^*$ holds*

In particular, under Assumption 1 and under an appropriate step size rule (see Theorem 2.4), the iterates generated by Algorithm 2 satisfy Assumption A, although the feasibility of y_k requires some comments. The initial iterate y_0 is in the feasible region of (4.7) and all the subsequent iterates y_{k+1} are obtained by taking a convex combination between y_k and a direction s_k that is feasible for (4.7) as well. In particular, y_k is always feasible for (4.7) indeed.

We start with the following well-known lemma.

Lemma 4.5. *Under Assumptions 1 and A, the sequence $\{y_k\}$ satisfies*

$$\lim_{k \to \infty} \text{dist}(y_k, Y_{\text{opt}}) = 0 \tag{4.15}$$

Proof. Suppose that (4.15) fails. Then, there exists an $\epsilon > 0$ and a subsequence $\{y_{k_j}\}$ for which

$$\text{dist}(y_{k_j}, Y_{\text{opt}}) \geq \epsilon$$

holds for all j. Since the feasible region of (4.7) is compact (Proposition 4.2) and all the y_k are feasible, there exists a convergent subsequence of $\{y_{k_j}\}$. This would lead to a cluster point $\bar{y}$ that satisfies $\lim_{j \to \infty} h(y_{k_j}) = h(\bar{y}) = d^*$ and $\text{dist}(\bar{y}, Y_{\text{opt}}) \geq \epsilon$, which is a contradiction. $\square$

Although the sequence $\{y_k\}$ is not ensured to converge, its cluster points must all be minimizers of (4.7), by Lemma 4.5. With that, our first result is that the primal iterates x_k indeed converge to the unique optimal solution.

Theorem 4.6 (Convergence of primal iterates)**.** *Under Assumptions 1 and A we have*

$$\lim_{k\to\infty} x_k = x_{\mathrm{opt}},$$

where x_{opt} is the optimal solution of (4.1)*.*

Proof. From (4.6), we have

$$\forall y_{\mathrm{opt}} \in Y_{\mathrm{opt}}, \ x_{\mathrm{opt}} = \nabla f^*(T^* y_{\mathrm{opt}}).$$

Therefore,

$$
\begin{aligned}
\forall y_{\mathrm{opt}} \in Y_{\mathrm{opt}}, \ \|x_k - x_{\mathrm{opt}}\| &= \|x_k - \nabla f^*(T^* y_{\mathrm{opt}})\| \\
&= \|\nabla f^*(T^* y_k) - \nabla f^*(T^* y_{\mathrm{opt}})\|.
\end{aligned}
$$

Taking the infimum with respect to y_{opt} in Y_{opt}, we obtain

$$\|x_k - x_{\mathrm{opt}}\| = \inf_{y_{\mathrm{opt}} \in Y_{\mathrm{opt}}} \|\nabla f^*(T^* y_k) - \nabla f^*(T^* y_{\mathrm{opt}})\|.$$

We recall that ∇f^* is $1/\mu$-Lipschitz continuous because of the μ-strong convexity of f, see [21, Theorem 4.2.1]. Thus,

$$
\begin{aligned}
\|x_k - x_{\mathrm{opt}}\| &= \inf_{y_{\mathrm{opt}} \in Y_{\mathrm{opt}}} \|\nabla f^*(T^* y_k) - \nabla f^*(T^* y_{\mathrm{opt}})\| \\
&\le \inf_{y_{\mathrm{opt}} \in Y_{\mathrm{opt}}} \frac{1}{\mu} \|T^* y_k - T^* y_{\mathrm{opt}}\| \\
&\le \inf_{y_{\mathrm{opt}} \in Y_{\mathrm{opt}}} \frac{\|T^*\|_{\mathrm{op}}}{\mu} \|y_k - y_{\mathrm{opt}}\| \\
&= \frac{\|T^*\|_{\mathrm{op}}}{\mu} \mathrm{dist}\,(y_k, Y_{\mathrm{opt}}).
\end{aligned}
$$

However, under the stated assumptions, Lemma 4.5 ensures that $\lim_{k\to\infty} \mathrm{dist}\,(y_k, Y_{\mathrm{opt}}) = 0$ holds, which leads to $\lim_{k\to\infty} \|x_k - x_{\mathrm{opt}}\| = 0$. □

In spite of the potential lack of convergence of $\{y_k\}$, the primal sequence $\{x_k\}$ *is* ensured to converge by Theorem 4.6. Related to that, under a mild assumption on T, next we prove that the rate of convergence of x_k to x_{opt} is no worse than the square-root of the rate of convergence of $h(y_k)$ to d^*.

Theorem 4.7 (Rate of convergence)**.** *Suppose that Assumptions 1 and A hold, f is μ-strongly convex function and that T^*T is positive definite. Then, for every k we have*

$$\|x_k - x_{\mathrm{opt}}\| \le \sqrt{\frac{2\|T\|_{\mathrm{op}}^2}{\mu \lambda_{\min}(T^*T)}} \sqrt{h(y_k) - d^*},$$

where $\lambda_{\min}(T^*T)$ (> 0) is the minimum eigenvalue of T^*T, x_{opt} is the optimal solution of (4.1) *and d^* is the optimal value of* (D)*.*

Proof. First, we check that ∇h is Lipschitz continuous with constant $\|T\|_{\mathrm{op}}^2/\mu$. Since f is μ-strongly convex, ∇f^* is $1/\mu$-Lipschitz continuous ([21, Theorem 4.2.1]). Thus, for $y, y' \in \mathbb{R}^n$ we have

$$
\begin{aligned}
\|\nabla h(y) - \nabla h(y')\| &= \|T\nabla f^*(T^*y) - T\nabla f^*(T^*y')\| \\
&\leq \|T\|_{\mathrm{op}}\|\nabla f^*(T^*y) - \nabla f^*(T^*y')\| \\
&\leq \|T\|_{\mathrm{op}}\frac{1}{\mu}\|T^*y - T^*y'\| \\
&\leq \|T\|_{\mathrm{op}}\frac{1}{\mu}\|T^*\|_{\mathrm{op}}\|y - y'\| \\
&= \frac{1}{\mu}\|T\|_{\mathrm{op}}^2\|y - y'\|.
\end{aligned}
$$

From the convexity of h and the Lipschitz continuity of ∇h with constant $\|T\|_{\mathrm{op}}^2/\mu$, we obtain

$$
h(y_k) \geq h(y_{\mathrm{opt}}) + \langle y_k - y_{\mathrm{opt}}, \nabla h(y_{\mathrm{opt}})\rangle + \frac{1}{2}\frac{\mu}{\|T\|_{\mathrm{op}}^2}\|\nabla h(y_k) - \nabla h(y_{\mathrm{opt}})\|^2,
$$

see, e.g., [36, Theorem 2.1.5, Equation (2.1.10)]. Recalling that $\nabla h(y_k) = T\nabla f^*(T^*y_k) + b = Tx_k + b$ and readjusting the inequality, we obtain

$$
\|Tx_k - Tx_{\mathrm{opt}}\|^2 \leq \frac{2\|T\|_{\mathrm{op}}^2}{\mu}\left(h(y_k) - h(y_{\mathrm{opt}}) + \langle y_k - y_{\mathrm{opt}}, -\nabla h(y_{\mathrm{opt}})\rangle\right). \tag{4.16}
$$

By Assumption A, y_k is always feasible for (4.7). From the first-order optimality conditions for the problem (4.7) (e.g., [36, Theorem 2.2.9]), we obtain the inequality

$$
\langle y_k - y_{\mathrm{opt}}, -\nabla h(y_{\mathrm{opt}})\rangle \leq 0. \tag{4.17}
$$

Also, from the assumption $T^*T \succ 0$,

$$
\|Tx_k - Tx_{\mathrm{opt}}\|^2 \geq \lambda_{\min}(T^*T)\|x_k - x_{\mathrm{opt}}\|^2. \tag{4.18}
$$

From (4.16), (4.17), and (4.18), we obtain

$$
\lambda_{\min}(T^*T)\|x_k - x_{\mathrm{opt}}\|^2 \leq \frac{2\|T\|_{\mathrm{op}}^2}{\mu}\left(h(y_k) - h(y_{\mathrm{opt}})\right),
$$

which leads to

$$
\|x_k - x_{\mathrm{opt}}\| \leq \sqrt{\frac{2\|T\|_{\mathrm{op}}^2}{\mu\lambda_{\min}(T^*T)}}\sqrt{h(y_k) - h(y_{\mathrm{opt}})}.
$$

$\square$

As a corollary of Theorem 4.7 and of the fact that a convex function is locally Lipschitz continuous over the relative interior of its domain, we can also get a rate of convergence for the primal objective function.

Corollary 4.8 (Convergence of the primal objective function). *Suppose that Assumptions 1 and A hold. Then, there exists a positive constant L such that the output of Algorithm 1 after k iterations satisfies*

$$
f(x_k) - f(x_{\mathrm{opt}}) \leq L\|x_k - x_{\mathrm{opt}}\|.
$$

In particular, under the assumptions of Theorem 4.7 we have

$$
f(x_k) - f(x_{\mathrm{opt}}) \leq L\sqrt{\frac{2\|T\|_{\mathrm{op}}^2}{\mu\lambda_{\min}(T^*T)}}\sqrt{h(y_k) - d^*},
$$

20

Proof. By Assumption A and the compactness of the feasible region of (4.7) (Proposition 4.7), the sequence of iterates $\{y_k\}$ is bounded. By definition $x_k = \nabla f^*(T^* y_k)$ holds and ∇f^* is Lipschitz continuous, because of the strong convexity of f. In particular, $\{x_k\}$ is the image of the bounded set $\{y_k\}$ via a continuous map with closed domain, so $\{x_k\}$ is bounded as well[2] . Although a given x_k might fail to be feasible for (P), it belongs to the domain of f, which is $\mathbb{R}^n$ by assumption. Now, a convex function is Lipschitz continuous relative to any bounded set whose closure is contained in the relative interior of its domain (e.g., [44, Theorem 10.4]). Then, since $\{x_k\}$ is bounded, there exists a constant $L > 0$ such that

$$f(x_k) - f(x_{\text{opt}}) \leq L \, \|x_k - x_{\text{opt}}\|$$

holds for every k, as we wanted to show. The remainder of the corollary follows directly from Theorem 4.7. $\quad\square$

The summary of Theorems 4.6, 4.7 and Corollary 4.8 is the following. Under Assumptions 1 and A, x_k indeed converges to x_{opt}. Furthermore, the convergence rates of x_k to x_{opt} and of $f(x_k)$ to $f(x_{\text{opt}})$ are no worse than the square root of the convergence rate of $h(y_k)$ to d^*, provided that T^*T is positive definite. These results are not specific to Frank-Wolfe methods, since they hold for any approach that generate sequences as in Assumption A and arise as consequences of the relations between a strongly convex optimization problem and its dual.

In the particular case that $\{(x_k, y_k)\}$ are the iterates generated by Algorithm 2, if the step size is chosen as to ensure $h(y_k) - d^* = O(1/k)$ (see Theorem 2.4), we have the following guarantees on the primal iterates and the primal objective function.

$$\lim_{k \to \infty} x_k = x_{\text{opt}}, \tag{4.19}$$

$$\|x_k - x_{\text{opt}}\| = O(1/\sqrt{k}), \tag{4.20}$$

$$f(x_k) - f(x_{\text{opt}}) = O(1/\sqrt{k}), \tag{4.21}$$

with the caveat that (4.20) and (4.21) require the extra assumption that T^*T is positive definite (i.e., T is injective).

Before we move on, we should remark that it was recently shown that the iterates of the Frank-Wolfe method may fail to converge in nontrivial settings, see [6]. We note that there is no contradiction with (4.19), since we proved convergence for the *primal* iterate x_k. We recall that in our approach Frank-Wolfe is applied to the dual problem (4.7) and, indeed, for the corresponding iterates y_k we were not able to say anything more than what is expressed in Lemma 4.5.

4.3 Practical considerations

Having discussed the theoretical properties of Algorithm 2, we now take a look at some implementation issues that may arise.

Choice of stopping criteria There is some level of flexibility regarding the choice of stopping criterion. Typically, a maximum iteration number can be set or, for example, the Frank-Wolfe gap can be used to stop the algorithm when it becomes too small and the x_k iterates are close to being feasible.

Computation of the minimum eigenvalue function and conjugate vectors Recall that the minimum eigenvalue function (see (2.2)) required in Algorithm 2 depends on the (regular) cone $\mathcal{K}$ and the chosen direction $e \in \text{int}\,\mathcal{K}$. If a closed form expression is not readily available, as long as a procedure to decide membership in $\mathcal{K}$ is available, a binary search approach can be used to find $\lambda_{\min}(x)$ for a given x. Obtaining conjugate vectors, however, is more challenging and depends on having a good understanding of the facial structure of $\mathcal{K}$.

---

[2]Of course, the fact that the domain is closed is important, otherwise the image of the interval $(0,1)$ by the function $x \mapsto 1/x$ would be a counter-example.

In the particular case where $\mathcal{K}$ is a hyperbolicity cone, there is more structure one can exploit in order to efficiently compute $\lambda_{\min}$ and conjugate vectors, so let us take a look at this case. Suppose that $\mathcal{K} = \Lambda(p, e)$, where $p : \mathbb{R}^n \to \mathbb{R}$ has degree d. By Proposition 2.3, $\lambda_{\min}$ is the smallest root of the one-dimensional polynomial $t \mapsto p(x - te)$. Or, equivalently, $\lambda_{\min}$ is *minus* the *largest root* of the polynomial $p_x : \mathbb{R} \to \mathbb{R}$ such that $p_x(t) := p(x + te)$. In order to compute the roots of p_x, we first need to determine its coefficients. In theory, this could be done via its Taylor expansion, since

$$p_x(t) = p(x + te) = \sum_{i=1}^{d} \frac{1}{i!} p^{(i)}(x) t^i,$$

where we recall that $p^{(i)} = D_e^i p$. In practice, however, a naive evaluation of $p^{(i)}$ may be computationally prohibitive. To address this issue, we follow Renegar's suggestion in [41, Section 9] to evaluate the terms $p^{(i)}(x)$ using the inverse Fast Fourier Transform as follows.

Theorem 4.9. *[41, Section 9] Let $p : \mathbb{R}^n \to \mathbb{R}$ be a d-degree hyperbolic polynomial whose directional vector is $e \in \mathbb{R}^n$ and ω be a primitive d-th root of unity. Then,*

$$\frac{1}{i!} p^{(i)}(x) = \frac{1}{d} \sum_{j=1}^{d} \omega^{-ij} p(x + \omega^j e) \quad (i = 1, \cdots, d - 1)$$

holds, and hence

$$\nabla p^{(i)}(x) = \frac{i!}{d} \sum_{j=1}^{d} \omega^{-ij} \nabla p(x + \omega^j e) \quad (i = 1, \cdots, d - 1).$$

Remark. *At the end of Section 9 in [41], equations for $p^{(i)}(x)$ and $\nabla p^{(i)}(x)$ are given in which i appears in place of $-i$ in the "ω^{-ij}" term. We believe this is a typo, which can be verified by considering $p(x_1, x_2, x_3) = x_1 x_2 x_3$, $e = (1, 1, 1)$ and computing $p^{(1)}(x_1, x_2, x_3)$ which is $x_1 x_2 + x_1 x_3 + x_2 x_3$. That said, the inverse of the Vandermonde matrix given in Section 9 is correct.*

Once the coefficients of $p_x(t)$ or $p_x(-t)$ are identified, we can numerically obtain its roots which allow us to compute the eigenvalues of x. The computation of the roots of a polynomial is itself a nontrivial problem and there are a few choices on how to handle it. For example, one simple approach is to form the companion matrix and compute its eigenvalues.

There is one extra outstanding issue regarding eigenvalue computations. The terms $p(x + w^j e)$ appearing in Theorem 4.9 may be large even if x itself is not very large[3]. In order to ameliorate possible overflow issues, we may exploit the fact that $\lambda(\alpha x) = \alpha \lambda(x)$ holds for $\alpha > 0$ and scale x by a suitable constant before computing its eigenvalues.

Finally, we note that Theorem 4.9 also provides a formula for the computation of conjugate vectors as in Proposition 4.4.

4.4 Minimizing a positive definite quadratic function

In the previous subsections we discussed Algorithm 2 and its properties. In this subsection, we discuss how Assumption 1 is satisfied for the problem of minimizing a positive definite quadratic function under conic constraints. More precisely, we will check that c_D can be explicitly obtained from the problem data. We consider the following problem

$$
\begin{aligned}
\min_{x \in \mathbb{R}^n} \quad & f(x) = \tfrac{1}{2} \langle x, Qx \rangle + \langle c, x \rangle \\
\text{s.t.} \quad & Tx + b \in \mathcal{K},
\end{aligned}
\tag{4.22}
$$

where Q is a symmetric positive definite matrix (i.e., $Q \succ 0$), $\mathcal{K} \subset \mathbb{R}^m$ is a regular cone and the inner product is the usual Euclidean one. Additionally we assume that there exists $\hat{e}$ such that $T\hat{e} = e \in \operatorname{ri} \mathcal{K}$. We

---

[3]For example, for $p(x) := x_1 \cdots x_n$, $e = (1, \ldots, 1)$ and $x_0 := (2, \ldots, 2)$, we have $p(x_0 + e) = 3^n$.

can construct a feasible solution of (4.22) from $\hat{e}$ as follows. Since $T\hat{e} \in \text{ri}\,\mathcal{K}$, there exists $\epsilon > 0$ such that $T\hat{e} + \epsilon b \in \mathcal{K}$. For this ϵ, $T(\hat{e}/\epsilon) + b \in \mathcal{K}$, so $\hat{e}/\epsilon$ is a feasible solution to (4.22).

The conjugate function of f is

$$f^*(x) = \frac{1}{2}\langle x - c, Q^{-1}(x-c)\rangle,$$

and hence,

$$\nabla f^*(x) = Q^{-1}(x-c).$$

Next, we will show that

$$c_D := \|\hat{e}\| \sqrt{\frac{\left(2f(\hat{e}/\epsilon) + \langle c, Q^{-1}c\rangle\right)}{\lambda_{\min}(Q^{-1})}}$$

satisfies Assumption 1. Let x_{opt} and y_{opt} be optimal solutions of (4.22) and the dual problem of (4.22), respectively. Since $\hat{e}/\epsilon$ is a feasible solution of (4.22),

$$f(\hat{e}/\epsilon) \geq f(x_{\text{opt}})$$
$$= \frac{1}{2}\langle x_{\text{opt}}, Qx_{\text{opt}}\rangle + \langle c, x_{\text{opt}}\rangle$$

From $x_{\text{opt}} = \nabla f^*(T^*y_{\text{opt}})$ (see (4.6)), we have

$$\frac{1}{2}\langle x_{\text{opt}}, Qx_{\text{opt}}\rangle + \langle c, x_{\text{opt}}\rangle = \frac{1}{2}\langle \nabla f^*(T^*y_{\text{opt}}), Q\nabla f^*(T^*y_{\text{opt}})\rangle + \langle c, \nabla f^*(T^*y_{\text{opt}})\rangle$$
$$= \frac{1}{2}\langle Q^{-1}(T^*y_{\text{opt}} - c), T^*y_{\text{opt}} - c\rangle + \langle c, Q^{-1}(T^*y_{\text{opt}} - c)\rangle$$
$$= \frac{1}{2}\langle T^*y_{\text{opt}}, Q^{-1}T^*y_{\text{opt}}\rangle - \frac{1}{2}\langle c, Q^{-1}c\rangle$$

Therefore,

$$\langle T^*y_{\text{opt}}, Q^{-1}T^*y_{\text{opt}}\rangle \leq 2f(\hat{e}/\epsilon) + \langle c, Q^{-1}c\rangle$$

which implies

$$\|T^*y_{\text{opt}}\|^2 \leq \frac{\left(2f(\hat{e}/\epsilon) + \langle c, Q^{-1}c\rangle\right)}{\lambda_{\min}(Q^{-1})},$$

where $\lambda_{\min}(Q^{-1})$ (> 0) is the minimum eigenvalue of Q as a matrix. From this inequality, c_D is derived as follows.

$$\langle e, y_{\text{opt}}\rangle = \langle T\hat{e}, y_{\text{opt}}\rangle$$
$$= \langle \hat{e}, T^*y_{\text{opt}}\rangle$$
$$\leq \|\hat{e}\|\,\|T^*y_{\text{opt}}\|$$
$$\leq \|\hat{e}\| \sqrt{\frac{\left(2f(\hat{e}/\epsilon) + \langle c, Q^{-1}c\rangle\right)}{\lambda_{\min}(Q^{-1})}}.$$

We note that (4.22) contains the particular case of the projection problem. That is, if we wish to project an arbitrary $x_0 \in \mathbb{R}^m$ onto $\mathcal{K}$, we may take $c := -x_0$, let T be the identity map and Q be the identity matrix in (4.22). In this case, it is enough to let $e = \hat{e}$ be any element in interior of $\mathcal{K}$ and let $\epsilon := 1$. With that, c_D simplifies to

$$c_D = \|e\|\,\|e - x_0\|. \tag{4.23}$$

5 Numerical experiments

In order to test our ideas, we wrote a MATLAB implementation of Algorithm 2 and performed some numerical experiments. Naturally, numerical experiments typically involve some sort of comparison. However, there are few methods and solvers that can handle directly the problems we discuss in this paper. The most direct competitor seems to be the accelerated gradient method (AGM) developed by Renegar in [43], which is specific to hyperbolicity cones. The DDS solver, which implements an interior point method, is also capable of handling hyperbolicity cone constraints. With this in mind, our goal in this section is to answer the following questions.

(1) Is our dual Frank-Wolfe method competitive against Renegar's AGM in the case of hyperbolicity cones?

(2) Is our method "competitive" against IPMs?

As we will see, the answer for question (1) is a relatively straightforward "yes" as we found that our proposed method significantly outperforms Renegar's AGM although it should be stressed that Renegar's AGM is applicable to a more general class of problems.

The second question is more delicate. We wrote "competitive" (in quotation marks) because IPMs and first-order methods have different design goals. Generally speaking, first-order methods have a small cost per iteration. They struggle to get accurate solutions but they may be a good choice if the goal is to obtain solutions with low-to-medium accuracy fast. Conversely, IPMs seem to excel at getting accurate solutions, but often have more numerically expensive iterations.

Taking heed of this difference, we designed experiments to check how long does it take for our method to obtain solutions that are "somewhat close" to the solutions obtained by IPMs. This is consistent with the idea that the computation of a projection is often used as a subroutine in another method, so there are cases where getting a less accurate solution fast is more desirable. Taking this into consideration, the answer to question (2) is a qualified yes, as we will see in the results. Roughly speaking, our approach consistently obtains solutions that are within 1% to 5% of the solutions obtained by IPMs in a fraction of the time.

All files can be found in the following link.

https://github.com/bflourenco/dfw_projection

All experiments were done in a PC with a Intel Xeon W-2145 CPU, 128GB RAM and Windows 10 Pro. The code was implemented in Matlab 2022b.

5.1 Projection onto derivative relaxations

In this subsection our paper comes full circle and we address again the problem of projecting a point onto a hyperbolicity cone, this time from a numerical point of view. We focus on hyperbolicity cones for which there are no (known) closed form expressions in terms of the underlying eigenvalues. Perhaps the simplest cones of this type correspond to the derivative relaxations of the nonnegative orthant. We remark that, more generally, derivative relaxations are often used to test ideas in the theory of hyperbolic polynomials and have been extensively studied, e.g., [51, 45, 8, 47, 46, 26].

In particular, the k-th derivative relaxation of $\mathbb{R}^n_+$ (see Section 2.1), denoted by $\mathbb{R}^{n,(k)}_+$ satisfies

$$\mathbb{R}^{n,(k)}_+ = \Lambda(\sigma_{n,n-k}, e),$$

where $e := (1, \dots, 1)$ and $\sigma_{n,k}$ is the k-th elementary symmetric polynomial in n variables which is given by

$$\sigma_{n,k}(x) := \sum_{1 \le i_1 < \cdots < i_k \le n} x_{i_1} \cdots x_{i_k}.$$

In this section, our target problem is

$$\min_{x \in \mathbb{R}^n} \ \frac{1}{2} \|x - c\|^2 \tag{5.1}$$
$$\text{s.t. } x \in \mathbb{R}^{n,(k)}_+$$

where $\mathbb{R}_+^{n,(k)}$ is the k-th derivative cone of $\mathbb{R}_+^n$ along $e = \mathbf{1}_n$ and c is the vector we wish to project onto $\mathbb{R}_+^{n,(k)}$. Since the objective function of (5.1) is a positive definite quadratic function, (5.1) is implementable as discussed in Section 4.4. The derivative relaxations $\mathbb{R}_+^{n,(k)}$ for $0 < k < n-3$, $n \geq 4$ are non-polyhedral and there are no known formulae for their orthogonal projections.

Implementation remarks on Algorithm 2 We implemented Algorithm 2 fairly straightforwardly following the discussion in Section 4. In particular, the constant term in (5.1) is removed and we consider the equivalent problem

$$\min_{x \in \mathbb{R}^n} \quad \frac{1}{2}\|x\|^2 - \langle c, x \rangle \tag{5.2}$$
$$\text{s.t.} \ x \in \mathbb{R}_+^{n,(k)}.$$

The problem (5.2) is regarded as the primal problem (P) and the constant c_D is computed as described in (4.23) with $e = \hat{e} = (1, \ldots, 1)$, $x_0 = c$.

The code for our implementation are in the files `FW_HP_exp.m` and `FW_HP.m`. The former is the one we actually use in the experiments and it returns all the iterates generated by the method and other useful experimental information. It is, however, quite memory intensive, so we also provide the file `FW_HP.m` which only returns the best solution obtained during the course of the algorithm. For users that wish a quick way to compute a projection onto a given hyperbolicity cone, we also provide the file `poly_proj.m` that is a wrapper around `FW_HP.m` specialized for projection computations, see examples in `poly_proj_examples.m`.

We also implemented special functions to handle elementary symmetric polynomials and their gradients. Even for small n, the description of $\sigma_{n,k}$ can be quite large. For example, for $n = 20$, σ_{10} is a sum of 184756 monomials. Rather than store $\sigma_{n,k}$ as a matrix, we use an approach based on a simple divide-and-conquer algorithm to evaluate $\sigma_{n,k}$ and its gradients directly. The corresponding files are `eleSym.m` and `grad_eleSym.m`. In the numerical experiments we compare both the naive approach (i.e., storing the polynomials directly) and the implicit approach tailored for elementary symmetric polynomials.

Experimental and implementation remarks on Renegar's AGM and DDS In order to make use of Renegar's AGM and DDS we considered the following equivalent formulations of (5.1).

$$(5.1) \iff \begin{cases} \min\limits_{x,y} & y \\ \text{s.t.} & y \geq \|x - c\| \\ & x \in \mathbb{R}_+^{n,(k)} \end{cases} \iff \begin{cases} \min\limits_{x,y,z} & y \\ \text{s.t.} & y \geq \|z\| \\ & z = x - c \\ & x \in \mathbb{R}_+^{n,(k)} \end{cases}$$

Given a hyperbolicity cone and an underlying hyperbolic polynomial, Renegar's AGM also requires the computation of $p^{(i)}(x)$, $\nabla p^{(i)}(x)$ ($i = 1, \ldots, d$) and the hyperbolic eigenvalues. In our implementation we took similar precautions as the ones discussed in Section 4.3. Additionally, Renegar's algorithm requires the computation of the following expression which corresponds to the gradient of a smoothed version of the maximum eigenvalue function:

$$\nabla f_\mu(x) = \frac{1}{\sum_j m_j \exp(\lambda_j(x)/\mu)} \sum_j \frac{m_j \exp(\lambda_j(x)/\mu)}{p^{(m_j)}(x - \lambda_j(x)e)} \nabla p^{(m_j-1)}(x - \lambda_j(x)e), \tag{5.3}$$

where $\{\lambda_j(x)\}$ is the set of distinct eigenvalues of x and m_j is the multiplicity of $\lambda_j(x)$ and $\mu\ (> 0)$ is a parameter determining the accuracy of the algorithm, see [43, Proposition 3.3]. The smaller μ is, the smaller the error is guaranteed to be. However, if μ is too small, there may be numerical issues if (5.3) is evaluated naively. To address this problem, we use the idea in [35, Section 5.2] in order to reformulate (5.3) as (5.4).

$$\nabla f_\mu(x) = \frac{1}{\sum_j m_j \exp(\lambda_j(x) - \lambda_{\max}/\mu)} \sum_j \frac{m_j \exp(\lambda_j(x) - \lambda_{\max}/\mu)}{p^{(m_j)}(x - \lambda_j(x)e)} \nabla p^{(m_j-1)}(x - \lambda_j(x)e), \tag{5.4}$$

where $\lambda_{\max} = \max_j\{\lambda_j(x)\}$.

As in the case of our proposed method, we also adjusted the implementation of Renegar's AGM to make it possible to exploit the structure of elementary symmetric polynomials. Finally, we remark that Renegar's main algorithm ("MainAlgo" in [43]) prescribes that two accelerated gradient sub-methods run in parallel and, then, if at given point a certain condition is met *for the iterates of the first sub-method*, *both* sub-methods are stopped, a certain outer update is conducted and the sub-methods are then restarted. Here, in order to simplify the implementation, instead of running the sub-methods in parallel, we perform one iteration of each sub-method and check if the condition for the outer update is satisfied. During the discussion of the results of the numerical experiments we will revisit this issue.

5.1.1   A comparison between Renegar's AGM, Algorithm 2 and DDS

In this series of experiments we proceed as follows. We fix the values of n and k in (5.1) and then we generate 30 normally distributed points in $\mathbb{R}^n$. These are the c's we would like to project onto $\mathbb{R}_+^{n,(k)}$. For each generated point c we check if the minimum eigenvalue of c with respect to $\mathbb{R}_+^{n,(k)}$ and $e = (1, \ldots, 1)$ is greater than -10^{-4}. If this happens, then c is deemed to be too close to the cone, so we discard it and generate a new point.

Once the 30 points are generated, we solve the problem (5.1) with DDS, with our proposed method (see file `FW_HP_exp.m`) and with an implementation of Renegar's AGM (see file `AGM_HP.m`). For Algorithm 2 and Renegar's AGM we also considered variants that use code specialized to elementary symmetric polynomials. So, in total, each of the 30 instances is solved through 5 different methods, which will be, henceforth denoted by "DDS", "FW", "FW EleSym", "AGM" and "AGM EleSym". Here, we recall that FW and AGM correspond to Algorithm 2 and Renegar's accelerated gradient method, respectively. "EleSym" indicates the usage of special methods to handle elementary symmetric polynomials as discussed previously.

We consider the objective function value obtained by DDS as the baseline to which we compare the performance of the other algorithms. The results are described in Tables 1 and 2. We now explain the meaning of the data. For example, consider the first line in Table 1a, so that the "Error" column indicates "10%". Roughly speaking, for this line, we checked how much time does each one of the 4 tested methods need to get a solution that has a value that is within 10% of the function valued obtained by DDS.

More formally, for each instance i, denote by f_{DDS}^i and t_{DDS}^i the objective function value obtained by DDS and the corresponding running time, respectively. Analogously, denote by $f_{\mathrm{FW}}^{i,j}$ the function value obtained by Algorithm 2 for the i-th instance at the j-th iteration. Denote by $t_{\mathrm{FW}}^{i,j}$ the time elapsed after the j-th iteration. For each instance k and a given error tolerance E (e.g., 10%), we checked the amount of time that the code `FW_HP_exp.m` required to reach an iteration j for which

$$f_{\mathrm{FW}}^{i,j} \leq f_{\mathrm{DDS}}^i \times \left(\frac{100 + E}{100}\right), \tag{5.5}$$

holds and the minimal eigenvalue of corresponding primal iterate (the x_k in Algorithm 2) is at least -10^{-8} (i.e., x_k is sufficiently close to being feasible). Then, we record the ratio $\frac{t_{\mathrm{FW}}^{i,j}}{t_{\mathrm{DDS}}^i}$ and in the column "Mean" we register the average of these ratios together with their standard deviation. This average is what we henceforth call the *mean relative time*. In the "S" (for *Success*) column, we indicate the percentage of instances for which the method was able to find a solution that satisfies (5.5) within the feasibility requirements.

For the other methods FW EleSym, AGM and AGM EleSym we proceed similarly. We remark that although each algorithm uses a different equivalent formulation for the problem (5.1), objective function value comparisons are always done with respect to the objective function of (5.1). For each instance, for all methods except DDS, we set the maximum running time to be equal to the time spent by DDS. The rationale is that it does not make sense to run a first order method longer than the time required by an IPM for the same problem.

For example, for the first line in Table 1a (which corresponds to $n = 10$, $k = 1$), the entry "0.39 ± 0.76" under "FW" means that, on average, our proposed method was able to find a solution whose objective value is within 10% of the value found by DDS using 0.39% of the time that DDS needed to find f_{DDS}^i. The

corresponding standard deviation was 0.76. It also succeeds for all 30 instances, i.e., for each one of the instances there was at least one iteration that satisfied (5.5) with $E = 10$. The data in the other columns "FW EleSym", "AGM" and "AGM EleSym" have analogous meaning.

As we go down Table 1a, the mean relative time increases and success rate decreases. As the error E decreases, it becomes harder to approach the values obtained by DDS within the allowed time budget. Still, we believe it is notable that for $E = 1\%$, FW is able to get solutions whose values are within 1% of the value obtained by DDS using less than 1% of DDS's running time. For the FW EleSym code, which is the variant optimized for elementary symmetric polynomials, we were able to get even more mileage with 100% success for $E = 0.5\%$ and mean relative time of less than 2%.

Overall, our impression is that a bottleneck in the 4 methods is the computation of minimum eigenvalues and, in the case of our proposed method, the computation of conjugate vectors too. Both are heavily influenced by the degree of the underlying hyperbolic polynomial. Indeed, the results for FW and FW EleSym for $n = 10$ and $k = 2$ (Table 1b) seem a bit better than the ones for $n = 10$ and $k = 1$ (Table 1a) in the sense that the success rates were higher.

In contrast for $n = 20$ and $k \in \{1, 2\}$, we have polynomials of degrees 19 and 18 respectively. In those cases, the performance of FW, AGM and AGM EleSym plummet and these three methods seem to struggle to even get low accuracy solutions. However, FW EleSym is still competitive and is able to get high success rates up to $E = 0.1$ with reasonable mean relative times.

In Table 2, we have the results for $(n, k) \in \{(30, 27), (40, 37), (50, 47)\}$. In all those cases, the degree of the hyperbolic polynomial is just three. Both FW and FW EleSym had particularly strong performances, which leads further credence to the idea that the degree is an important factor. In the case of FW EleSym, we were able to consistently get within 0.05% or less of the objective value obtained by DDS with just a small fraction of the required running time. For example, for $n = 50, k = 47$ (Table 2c), on average, we needed no more than 0.05% of the running time of DDS in order to get within 0.005% of the objective value. In this case, $\sigma_{n,n-k}$ has 19600 monomials, so using routines specialized to elementary symmetric polynomials leads to a boost in performance. We believe that is why in this set of experiments, FW EleSym was better than FW. Similarly, AGM EleSym had a superior performance when compared to the pure AGM.

In Tables 1 and 2, we configured DDS to run with the default stopping criterion tolerance of 10^{-8}. This means that DDS is actively looking for relatively high-accuracy solutions. One may then reasonably wonder what would happen if we configure DDS to run with a lower accuracy. To address that, we considered the same exact experiment but with the DDS stopping tolerance set to 10^{-3}.

When DDS runs with a lower tolerance there are two opposing effects that appear. On one hand, the ratio of the running times tend to increase, since the denominator (i.e., the DDS running time) decreases as DDS stops earlier. On the other hand, since the solutions obtained by the DDS are less accurate, intuition would suggest that it would be easier to approach the solutions obtained by DDS using a first-order method. This would mean that the numerator of the mean relative times would get smaller. The former effect should lead to "worse" results and the latter effect should lead to "better" (i.e., higher success rates and/or decreased mean relative times) results.

The results are described in Tables 3 and 4. Overall, the results were largely similar to the ones in Tables 1 and 2 and seem to allow for similar conclusions. A notable difference is that, indeed, for some choices of n, k the success rate of FW is higher than in the case where DDS is run with high accuracy. For example, for $n = 10$ and $k = 1$ (Table 3a), we can see that the success rate of FW stays above 60% throughout the 30 instances, although, naturally, the mean relative times increase accordingly. For $n = 20, k = 1$, FW EleSym was able to get 80% success rate up until $E = 0.001\%$ with reasonable mean relative times, see Table 3c. In contrast, in the high-accuracy setting (Table 1c), $E = 0.05\%$ seems to be the best we could obtain with success rate above 80%. In the setting of Table 3, the fact that the solution obtained by DDS are easier to reach seems to be the stronger factor here.

In the case where the polynomials are of smaller degree (Table 4), the fact that DDS stops faster seems to be the preponderant effect, as the success rates for FW and FW EleSym are no longer 100% even at $E = 1\%$. Nevertheless, they still stay above 90% up to $E = 0.5\%$ with quite reasonable mean relative times. Overall, for FW EleSym we still need less than 0.3% of the running time of DDS in order to find solutions whose values are within 0.5% of the objective value found by DDS.

Taking Tables 1–4 in consideration, our conclusion is that: (a) in most cases Algorithm 2 indeed succeeds in getting low-to-medium accuracy solutions in a reasonable time; (b) Algorithm 2 seems to be faster than Renegar's AGM; (c) when the hyperbolic polynomial has many monomials and/or is of higher degree, FW EleSym tends to be significantly better than FW. A caveat is that, as mentioned previously, our implementation of Renegar's AGM is sequential rather than parallel, but even taking into consideration a parallel speed-up factor of 2, our approach still seems to outperform Renegar's AGM by a large margin.

On the other hand, it seems that Algorithm 2 is indeed quite sensitive to the precision of eigenvalue and conjugate vector computations. In particular, for high degree hyperbolic polynomials careful implementation of the eigenvalue computation routines is important.

5.1.2 Hyperbolic polynomials with many monomials

For certain choices of n and k, the corresponding $(n - k)$-th elementary symmetric polynomial has hundreds of thousands of monomials. For example, for $(n, k) = (20, 10)$ and $(n, k) = (30, 15)$, $\sigma_{n,n-k}$ has, respectively, 184756 and 155117520 monomials. In this subsection, we will check that FW EleSym still runs reasonably in those cases and we will also use this opportunity to check the behavior of Algorithm 2 over a single instance.

We note that for $(n, k) = (20, 10)$ and $(n, k) = (30, 15)$, DDS struggles to complete a single iteration. And, from the previous discussion we saw that FW EleSym was significantly faster than either version of Renegar's AGM. So, for this set of experiments we will only focus on FW EleSym and check the behavior of function values and the Frank-Wolfe gap.

For $(n, k) \in \{(20, 10), (30, 15)\}$ we generated 10 random instances using the same procedure as before. Then, for each instance, we ran FW EleSym for 10 seconds and we plotted the Frank-Wolfe gap and the relative objective function value in Figures 1 and 2 using log-log plots. For each instance the relative objective function value was computed as follows: we compute the smallest function value obtained through the 10 seconds among the primal iterates whose minimal eigenvalues were at least -10^{-8}. We call this value $\hat{f}_{\mathrm{opt}}$. Then, denoting the objective function of (P) by f and k-th primal iterate by x_k the relative objective function value at the k-th iteration is

$$\frac{\min_{1 \le i \le k} f(x_i) - \hat{f}_{\mathrm{opt}}}{\hat{f}_{\mathrm{opt}}},$$

with the caveat that "min" is only considered over primal iterates whose minimal eigenvalues were at least -10^{-8}. The goal is to measure empirically how fast the primal objective value is converging. Using $\hat{f}_{\mathrm{opt}}$ may seem odd, but the issue is that we do not know the true optimal values and have no other baselines to compare since we were not able to solve the problem with DDS.

Both Figures 1 and 2 suggest that the Frank-Wolfe gap and the function values are decreasing sublinearly, which is consistent with the convergence results described in Section 4.2. Denoting the common optimal value of (D) and (4.7) by d^*, we recall that the Frank-Wolfe gap at the k-th iteration is an upper bound to the difference $h(y_k) - d^*$, where $h(y_k)$ is the value of the dual objective function at the k-iterate. In view of Theorem 4.7, the square root of the Frank-Wolfe gap times a constant can be used to bound the distance of the primal iterate to the primal optimal solution. So the fact that in both plots the Frank-Wolfe gap is indeed decreasing for all instances, gives us some numerical confidence that Algorithm 2 is indeed approaching the true optimal solution of (P) in spite of the challenging circumstances. This suggests that even if the hyperbolic polynomial has millions of monomials, Algorithm 2 still has a fighting chance provided that the underlying computational algebra for the polynomial is carefully implemented.

5.1.3 Easy projection via `poly_proj.m`

We also provide the file `poly_proj.m` which is a wrapper around `FW_HP.m` specialized on computing projections. It receives three parameters: the point to be projected, the hyperbolic polynomial and a hyperbolicity direction. Optionally, it also receives a parameter that controls the behaviour of the solver. Polynomials can be informed via a matrix format or via an oracle interface where the user provides functions for computing the polynomial and related objects. Additionally, in the case of elementary symmetric polynomials, the

Error	FW Mean	FW S(%)	FW EleSym Mean	FW EleSym S(%)	AGM Mean	AGM S(%)	AGM EleSym Mean	AGM EleSym S(%)
10%	**0.39 ± 0.76**	100.0	0.70 ± 0.97	100.0	49.78 ± 28.06	73.3	93.87 ± 14.26	6.7
5%	**0.40 ± 0.76**	100.0	0.71 ± 0.97	100.0	74.43 ± 22.29	53.3	-	0
1%	**0.51 ± 1.01**	100.0	0.91 ± 1.09	100.0	94.52 ± 0.00	3.3	-	0
0.5%	**0.74 ± 1.26**	100.0	1.36 ± 1.50	100.0	-	0	-	0
0.1%	9.37 ± 7.12	90.0	**22.57 ± 19.05**	100.0	-	0	-	0
0.05%	26.91 ± 13.63	86.7	50.96 ± 29.87	76.7	-	0	-	0
0.01%	62.92 ± 30.91	6.7	91.40 ± 0.00	3.3	-	0	-	0
0.005%	82.37 ± 0.00	3.3	-	0	-	0	-	0

(a) $n = 10$, $k = 1$

Error	FW Mean	FW S(%)	FW EleSym Mean	FW EleSym S(%)	AGM Mean	AGM S(%)	AGM EleSym Mean	AGM EleSym S(%)
10%	**0.30 ± 0.05**	100.0	0.32 ± 0.09	100.0	43.26 ± 21.34	90.0	82.86 ± 9.08	70.0
5%	**0.32 ± 0.04**	100.0	0.36 ± 0.07	100.0	45.47 ± 18.17	46.7	91.00 ± 8.36	30.0
1%	**0.63 ± 0.28**	100.0	0.86 ± 0.42	100.0	62.71 ± 9.24	13.3	-	0
0.5%	**1.50 ± 0.97**	100.0	2.30 ± 1.50	100.0	64.52 ± 10.26	6.7	-	0
0.1%	**17.00 ± 6.88**	100.0	29.51 ± 13.66	100.0	-	0	-	0
0.05%	37.97 ± 13.54	96.7	61.28 ± 21.61	83.3	-	0	-	0
0.01%	71.63 ± 16.48	6.7	95.29 ± 0.00	3.3	-	0	-	0

(b) $n = 10$, $k = 2$

Error	FW Mean	FW S(%)	FW EleSym Mean	FW EleSym S(%)	AGM Mean	AGM S(%)	AGM EleSym Mean	AGM EleSym S(%)
10%	0.34 ± 0.00	3.3	**1.19 ± 0.52**	100.0	-	0	-	0
5%	0.34 ± 0.00	3.3	**1.21 ± 0.52**	100.0	-	0	-	0
1%	0.70 ± 0.00	3.3	**1.71 ± 0.78**	100.0	-	0	-	0
0.5%	-	0	**2.54 ± 1.24**	100.0	-	0	-	0
0.1%	-	0	**9.49 ± 7.33**	100.0	-	0	-	0
0.05%	-	0	31.53 ± 27.68	96.7	-	0	-	0
0.01%	-	0	95.66 ± 0.00	3.3	-	0	-	0

(c) $n = 20$, $k = 1$

Error	FW Mean	FW S(%)	FW EleSym Mean	FW EleSym S(%)	AGM Mean	AGM S(%)	AGM EleSym Mean	AGM EleSym S(%)
10%	0.16 ± 0.04	16.7	**0.22 ± 0.07**	100.0	-	0	-	0
5%	0.20 ± 0.03	13.3	**0.23 ± 0.07**	100.0	-	0	-	0
1%	0.20 ± 0.05	6.7	**0.40 ± 0.18**	100.0	-	0	-	0
0.5%	0.43 ± 0.38	6.7	**0.57 ± 0.27**	100.0	-	0	-	0
0.1%	5.78 ± 0.00	3.3	**6.39 ± 4.62**	100.0	-	0	-	0
0.05%	-	0	**23.44 ± 11.36**	100.0	-	0	-	0
0.01%	-	0	86.89 ± 3.73	6.7	-	0	-	0

(d) $n = 20$, $k = 2$

Table 1: All experiments were done with 30 randomly generated points. The polynomials in the experiments described here have degrees $9, 8, 19, 18$, respectively. For the "Mean" column values closer to 0 are better and indicate the mean *relative* time (in comparison to the running time of DDS) to get a solution whose value is within "Error" of the solution obtained by DDS. A bold entry in a row indicates the method with best mean relative time among the ones that were 100% successful.

	FW		FW EleSym		AGM		AGM EleSym	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	0.82 ± 0.11	100.0	$\mathbf{0.08 \pm 0.01}$	100.0	28.77 ± 11.14	100.0	6.94 ± 2.46	100.0
5%	0.82 ± 0.11	100.0	$\mathbf{0.08 \pm 0.01}$	100.0	34.79 ± 11.45	100.0	8.24 ± 2.52	100.0
1%	0.82 ± 0.11	100.0	$\mathbf{0.08 \pm 0.01}$	100.0	44.67 ± 12.26	93.3	13.56 ± 13.07	100.0
0.5%	0.82 ± 0.11	100.0	$\mathbf{0.08 \pm 0.01}$	100.0	47.22 ± 13.94	93.3	14.07 ± 13.01	100.0
0.1%	0.82 ± 0.11	100.0	$\mathbf{0.08 \pm 0.01}$	100.0	61.88 ± 17.32	63.3	28.66 ± 24.92	96.7
0.05%	1.24 ± 2.29	100.0	$\mathbf{0.28 \pm 1.11}$	100.0	68.68 ± 16.25	50.0	37.52 ± 30.31	90.0
0.01%	11.07 ± 25.34	86.7	12.04 ± 24.17	96.7	-	0	67.89 ± 20.59	46.7
0.005%	12.86 ± 26.33	46.7	19.11 ± 28.52	60.0	-	0	70.41 ± 20.81	30.0
0.001%	16.23 ± 21.75	6.7	37.99 ± 53.47	10.0	-	0	74.61 ± 19.81	20.0

(a) $n = 30$, $k = 27$

	FW		FW EleSym		AGM		AGM EleSym	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	1.01 ± 0.16	100.0	$\mathbf{0.06 \pm 0.01}$	100.0	31.03 ± 5.17	96.7	6.67 ± 11.86	100.0
5%	1.01 ± 0.16	100.0	$\mathbf{0.06 \pm 0.01}$	100.0	36.66 ± 7.43	96.7	7.38 ± 11.74	100.0
1%	1.01 ± 0.16	100.0	$\mathbf{0.06 \pm 0.01}$	100.0	53.32 ± 16.52	93.3	9.97 ± 11.95	100.0
0.5%	1.01 ± 0.16	100.0	$\mathbf{0.06 \pm 0.01}$	100.0	56.12 ± 16.05	90.0	10.42 ± 10.12	96.7
0.1%	1.01 ± 0.16	100.0	$\mathbf{0.06 \pm 0.01}$	100.0	70.36 ± 18.57	73.3	18.72 ± 19.91	96.7
0.05%	1.01 ± 0.16	100.0	$\mathbf{0.06 \pm 0.01}$	100.0	69.93 ± 16.88	40.0	27.55 ± 21.38	96.7
0.01%	1.01 ± 0.16	100.0	$\mathbf{0.06 \pm 0.01}$	100.0	76.53 ± 12.27	10.0	42.10 ± 23.54	90.0
0.005%	5.02 ± 17.12	86.7	$\mathbf{8.38 \pm 21.32}$	100.0	-	0	58.40 ± 21.60	83.3
0.001%	0.99 ± 0.13	30.0	8.03 ± 25.22	33.3	-	0	68.93 ± 21.91	43.3

(b) $n = 40$, $k = 37$

	FW		FW EleSym		AGM		AGM EleSym	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	1.52 ± 0.21	100.0	$\mathbf{0.05 \pm 0.01}$	100.0	47.46 ± 8.61	100.0	4.12 ± 0.74	100.0
5%	1.52 ± 0.21	100.0	$\mathbf{0.05 \pm 0.01}$	100.0	55.60 ± 15.51	93.3	7.75 ± 12.00	100.0
1%	1.52 ± 0.21	100.0	$\mathbf{0.05 \pm 0.01}$	100.0	69.45 ± 18.27	83.3	11.51 ± 14.72	100.0
0.5%	1.52 ± 0.21	100.0	$\mathbf{0.05 \pm 0.01}$	100.0	77.32 ± 15.72	70.0	13.82 ± 15.19	100.0
0.1%	1.52 ± 0.21	100.0	$\mathbf{0.05 \pm 0.01}$	100.0	75.88 ± 20.09	46.7	19.36 ± 19.66	100.0
0.05%	1.52 ± 0.21	100.0	$\mathbf{0.05 \pm 0.01}$	100.0	87.12 ± 23.98	33.3	24.95 ± 21.03	100.0
0.01%	1.52 ± 0.21	100.0	$\mathbf{0.05 \pm 0.01}$	100.0	104.30 ± 0.00	3.3	50.81 ± 22.23	90.0
0.005%	1.52 ± 0.21	100.0	$\mathbf{0.05 \pm 0.01}$	100.0	-	0	57.89 ± 19.14	90.0
0.001%	1.52 ± 0.21	50.0	1.14 ± 4.37	53.3	-	0	68.67 ± 18.28	70.0

(c) $n = 50$, $k = 47$

Table 2: All experiments were done with 30 randomly generated points. The polynomials in the experiments described here all have degree 3.

	FW		FW EleSym		AGM		AGM EleSym	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	**0.45 ± 0.20**	100.0	1.54 ± 1.26	100.0	46.52 ± 7.18	40.0	-	0
5%	**0.46 ± 0.23**	100.0	1.55 ± 1.25	100.0	54.44 ± 6.57	10.0	-	0
1%	**0.60 ± 0.31**	100.0	2.02 ± 1.53	100.0	-	0	-	0
0.5%	**0.89 ± 0.64**	100.0	2.78 ± 2.00	100.0	-	0	-	0
0.1%	10.08 ± 10.14	90.0	**24.80 ± 25.53**	100.0	-	0	-	0
0.05%	20.55 ± 19.45	90.0	32.35 ± 32.92	83.3	-	0	-	0
0.01%	31.44 ± 24.01	66.7	22.76 ± 23.67	43.3	-	0	-	0
0.005%	34.84 ± 26.88	66.7	28.35 ± 28.19	43.3	-	0	-	0
0.001%	34.91 ± 26.51	63.3	32.53 ± 32.91	43.3	-	0	-	0

(a) $n = 10$, $k = 1$, 30 points

	FW		FW EleSym		AGM		AGM EleSym	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	**0.53 ± 0.09**	100.0	1.07 ± 0.70	100.0	60.21 ± 16.23	76.7	-	0
5%	**0.57 ± 0.09**	100.0	1.22 ± 0.78	100.0	68.47 ± 11.59	43.3	-	0
1%	**1.17 ± 0.63**	100.0	2.81 ± 2.10	100.0	91.12 ± 17.68	6.7	-	0
0.5%	**2.68 ± 2.04**	100.0	5.45 ± 3.79	100.0	103.62 ± 0.00	3.3	-	0
0.1%	25.76 ± 15.20	96.7	42.96 ± 20.63	93.3	-	0	-	0
0.05%	43.59 ± 23.51	80.0	60.60 ± 27.33	60.0	-	0	-	0
0.01%	57.97 ± 30.11	43.3	50.11 ± 37.97	16.7	-	0	-	0
0.005%	58.95 ± 33.34	36.7	41.61 ± 35.22	13.3	-	0	-	0
0.001%	53.86 ± 33.49	30.0	45.10 ± 38.23	13.3	-	0	-	0

(b) $n = 10$, $k = 2$

	FW		FW EleSym		AGM		AGM EleSym	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	0.62 ± 0.00	3.3	**1.79 ± 0.70**	100.0	-	0	-	0
5%	0.62 ± 0.00	3.3	**1.81 ± 0.69**	100.0	-	0	-	0
1%	1.26 ± 0.00	3.3	**2.52 ± 0.98**	100.0	-	0	-	0
0.5%	-	0	**3.53 ± 1.59**	100.0	-	0	-	0
0.1%	-	0	**8.59 ± 4.64**	100.0	-	0	-	0
0.05%	-	0	**12.82 ± 11.34**	100.0	-	0	-	0
0.01%	-	0	19.17 ± 19.80	93.3	-	0	-	0
0.005%	-	0	19.20 ± 21.98	90.0	-	0	-	0
0.001%	-	0	19.14 ± 23.07	86.7	-	0	-	0

(c) $n = 20$, $k = 1$

	FW		FW EleSym		AGM		AGM EleSym	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	0.30 ± 0.02	16.7	**0.35 ± 0.09**	100.0	-	0	-	0
5%	0.33 ± 0.07	13.3	**0.38 ± 0.09**	100.0	-	0	-	0
1%	0.32 ± 0.05	6.7	**0.59 ± 0.22**	100.0	-	0	-	0
0.5%	0.65 ± 0.52	6.7	**0.84 ± 0.36**	100.0	-	0	-	0
0.1%	10.14 ± 0.00	3.3	**3.90 ± 5.04**	100.0	-	0	-	0
0.05%	17.08 ± 0.00	3.3	**7.35 ± 9.63**	100.0	-	0	-	0
0.01%	-	0	16.92 ± 18.41	96.7	-	0	-	0
0.005%	-	0	18.95 ± 21.07	96.7	-	0	-	0
0.001%	-	0	21.93 ± 24.73	96.7	-	0	-	0

(d) $n = 20$, $k = 2$

Table 3: The setting is the same as in Table 1 except that DDS is run with 10^{-3} tolerance.

	FW		FW EleSym		AGM		AGM EleSym	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	1.60 ± 0.16	100.0	$\mathbf{0.16 \pm 0.02}$	100.0	53.66 ± 8.51	96.7	13.59 ± 3.54	100.0
5%	1.60 ± 0.16	100.0	$\mathbf{0.16 \pm 0.02}$	100.0	66.23 ± 16.17	96.7	15.76 ± 3.51	96.7
1%	1.61 ± 0.15	96.7	0.16 ± 0.02	96.7	83.69 ± 16.86	76.7	23.15 ± 11.56	96.7
0.5%	1.61 ± 0.15	96.7	0.16 ± 0.02	96.7	82.00 ± 17.17	66.7	25.98 ± 15.76	96.7
0.1%	1.60 ± 0.15	60.0	0.16 ± 0.01	60.0	94.56 ± 8.71	13.3	37.60 ± 20.04	46.7
0.05%	1.59 ± 0.15	40.0	0.16 ± 0.01	40.0	-	0	45.18 ± 25.55	13.3

(a) $n = 30$, $k = 27$

	FW		FW EleSym		AGM		AGM EleSym	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	1.93 ± 0.24	100.0	$\mathbf{0.10 \pm 0.02}$	100.0	58.60 ± 8.38	96.7	8.54 ± 1.17	96.7
5%	1.93 ± 0.24	100.0	$\mathbf{0.10 \pm 0.02}$	100.0	68.14 ± 13.46	93.3	10.00 ± 2.11	96.7
1%	1.94 ± 0.24	96.7	0.11 ± 0.02	96.7	82.65 ± 21.42	56.7	15.12 ± 5.68	96.7
0.5%	1.94 ± 0.24	96.7	0.11 ± 0.02	96.7	79.98 ± 22.22	40.0	21.66 ± 20.04	96.7
0.1%	1.89 ± 0.21	66.7	0.10 ± 0.01	66.7	-	0	40.95 ± 25.76	50.0
0.05%	1.96 ± 0.20	40.0	0.10 ± 0.01	40.0	-	0	47.13 ± 27.91	16.7

(b) $n = 40$, $k = 37$

|	FW		FW EleSym		AGM		AGM EleSym	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	2.69 ± 0.27	100.0	$\mathbf{0.08 \pm 0.01}$	100.0	83.91 ± 13.77	100.0	7.45 ± 1.22	100.0
5%	2.69 ± 0.27	100.0	$\mathbf{0.08 \pm 0.01}$	100.0	87.51 ± 13.34	70.0	10.46 ± 7.67	96.7
1%	2.69 ± 0.27	96.7	0.08 ± 0.01	96.7	89.85 ± 12.53	26.7	17.91 ± 15.46	96.7
0.5%	2.69 ± 0.27	96.7	0.08 ± 0.01	96.7	93.65 ± 6.95	16.7	22.30 ± 20.80	96.7
0.1%	2.64 ± 0.26	73.3	0.08 ± 0.01	73.3	92.77 ± 9.31	10.0	36.95 ± 33.16	70.0
0.05%	2.66 ± 0.25	70.0	0.08 ± 0.01	70.0	-	0	61.83 ± 31.75	53.3

(c) $n = 50$, $k = 47$

Table 4: The setting is the same as in Table 2 except that DDS is run with 10^{-3} tolerance.
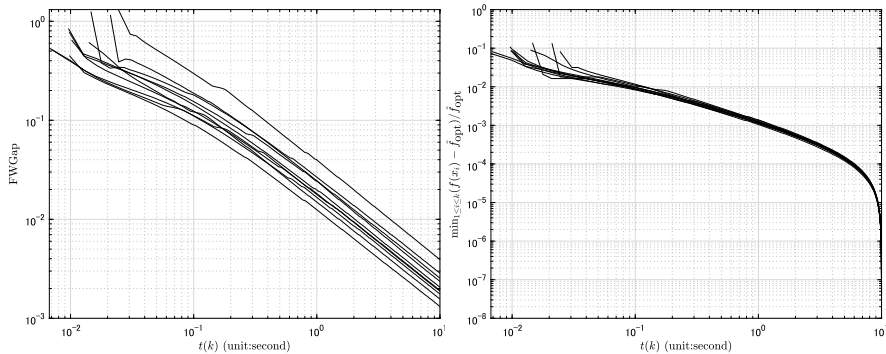
Figure 1: Frank-Wolfe gap and relative function values log-log plots for $n = 20$, $k = 10$, 10 instances. For those values of n and k, the corresponding hyperbolic polynomial has 184756 monomials. For the second figure, since $\hat{f}_{\mathrm{opt}}$ is the best solution obtained during the 10 seconds, it is natural that the relative error computed empirically goes to 0. Still, the fact that the graph is almost a straight line before that suggests that the convergence is indeed sublinear as predicted by (4.21).
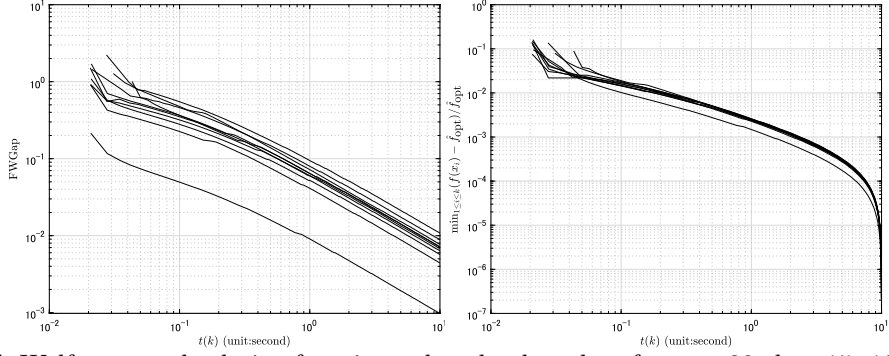
Figure 2: Frank-Wolfe gap and relative function values log-log plots for $n = 30$, $k = 15$, 10 instances. For those values of n and k, the corresponding hyperbolic polynomial has 155117520 monomials. Comments analogous to the ones in Figure 1 apply.

code can be configured to use the routines `eleSym.m` and `grad_eleSym.m`. For some usage examples, see `poly_proj_examples.m`.

5.2 Projection onto p-cones

Algorithm 2 is also applicable in more general settings so in this subsection our goal is to examine its behavior beyond the case of hyperbolicity cones. Here, we consider the problem of projecting a given $c \in \mathbb{R}^{n+1}$ onto a p-cone.

$$\min_{x \in \mathbb{R}^{n+1}} \quad \frac{1}{2}\|x - c\|^2 \tag{5.6}$$
$$\text{s.t.} \quad x \in \mathcal{K}_p^{n+1},$$

where $\mathcal{K}_p^{n+1} = \{(x_0, x_1, \ldots, x_n) \in \mathbb{R}^{n+1} \mid x_0 \geq 0, x_0^p \geq |x_1|^p + \cdots + |x_n|^p\}$ for some $p \in (1, \infty)$.

The problem (5.6) can be solved with both DDS and Mosek [34]. However, Mosek does not handle p-norm constraints directly, so we need to reformulate (5.6) using power cone constraints as follows.

$$x \in \mathcal{K}_p^{n+1} \iff \begin{cases} x_0^p \geq \sum_{i=1}^n |x_i|^p \\ x_0 \geq 0 \end{cases}$$

$$\iff \begin{cases} x_0^p \geq \sum_{i=1}^n y_i x_0^{p-1} \\ y_i x_0^{p-1} \geq |x_i|^p \quad (i = 1, \ldots, n) \\ x_0 \geq 0 \end{cases}$$

$$\iff \begin{cases} x_0 \geq \sum_{i=1}^n y_i \\ y_i^{1/p} x_0^{1-1/p} \geq |x_i| \quad (i = 1, \ldots, n) \\ x_0 \geq 0 \end{cases}$$

Using this transformation and dropping the quadratic objective function to a second-order cone constraint, (5.6) is transformed into the following equivalent problem.

$$\min_{t \in \mathbb{R}, x \in \mathbb{R}^{n+1}} \quad t$$
$$\text{s.t.} \quad t \geq \|x - c\|$$
$$x_0 \geq \sum_{i=1}^n y_i$$
$$x_0^{1-1/p} y_i^{1/p} \geq |x_i| \quad (i = 1, \ldots, n)$$
$$x_0 \geq 0$$

33

In our preliminary tests, Mosek was significantly faster than DDS, so in the following experiments we only compare with Mosek.

As in Section 5.1, our implementation of Algorithm 2 for solving (5.6) is relatively straightforward. For more details, see the file `FW_GCP_exp.m`. The element e is given by $(1, 0, \ldots, 0)$ and, with that, the generalized minimum eigenvalue function is such that

$$\lambda_{\min}(x) = x_0 - \sqrt{\sum_{i=1}^{n} |x_i|^p}.$$

The computation of conjugate vectors is done using the formulae described in [31, Section 4.1] and the constant c_D is computed as in (4.23).

We follow the same experimental procedures as in Section 5.1. We generate c by sampling from the standard normal distribution, discarding points that are too close to the cone and repeating until 30 points were generated.

We tested our implementation of Algorithm 2 with $p \in \{1.1, 1.3, 3, 5\}$ and $n \in \{100, 300, 500, 1000\}$. The results are described in Tables 5 and 6. Table entries have the same meaning as in Tables 1–4. Analogously to Section 5.1.1, our goal was to examine how long does take it take on average to obtain a solution that has a value that is close to the one obtained by Mosek. For example, the entry "2.65 ± 0.93" at the column $p = 3$ at Table 5d means that, on average over 30 points, Algorithm 2 required 2.65% of the time that Mosek needed in order to find a solution whose value is within 0.5% of the optimal value found by Mosek. As before, we only consider iterates that satisfy $\lambda_{\min}(x_k) \geq -10^{-8}$ and all objective function value computations are considered with respect the formulation in (5.6). In all instances, we set the maximum running time to be equal to the time spent by Mosek.

For the experiments in Table 5, Mosek was configured to run with its default accuracy settings. For most values of p and n, Algorithm 2 was able to obtain solutions having objective value between 1% and 0.5% of the value obtained by Mosek in a fraction of the time. The case $p = 1.1$ seems to be the most challenging where except for $n = 100$, Algorithm 2 typically requires on average at least 10% of the running time Mosek to reach solutions with $E = 0.5\%$. The performance for the other p's was better and for, say, $p = 3$, even for $n = 1000$ no more than 3% of the running time of Mosek was required to reach solutions with $E = 0.5\%$.

We also performed experiments where Mosek is configured to run with a lower optimality threshold of 10^{-3}, analogously to Tables 3-4[4]. These experiments are described in Table 6 and the results are largely similar to the ones reported in Table 5.

Again, it should be emphasized that the goal of experimental setting described in Tables 5 and 6 is to understand the trade-off between the accuracy afforded by a second-order approach and the fast iterations of a first-order method for this particular class of problems. In this sense, Algorithm 2 seems to be competitive since it consistently obtain relatively close solutions within a fraction of the time required by Mosek. On the other hand, it struggles to get closer than 0.1% of the objective value obtained by Mosek within the allotted time budget.

6   Conclusion

The initial motivation for this paper was the problem of computing projections onto hyperbolicity cones. We explored this question from both theoretical and numerical perspectives. As seen in Section 3, there are limits to what can be done for general hyperbolicity cones and formulae analogous to the ones for the positive semidefinite cone are only available in certain special cases (Propositions 3.4 and 3.6). In face of these limitations, we also proposed an algorithm based on the classic Frank-Wolfe method for computing projections, see Section 4. In fact, our method can handle more general problems including the case where the underlying cone is not necessarily a hyperbolicity cone.

---

[4]More precisely, the parameter `MSK_DPAR_INTPNT_CO_TOL_MU_RED` of Mosek which controls the relative complementarity gap tolerance is set to 10^{-3}.

	$p = 1.1$		$p = 1.3$		$p = 3$		$p = 5$	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	**6.35 ± 3.83**	100.0	**2.42 ± 0.95**	100.0	**1.32 ± 0.44**	100.0	**1.54 ± 0.53**	100.0
5%	**6.35 ± 3.83**	100.0	**2.42 ± 0.95**	100.0	**1.32 ± 0.44**	100.0	**1.97 ± 0.89**	100.0
1%	**6.35 ± 3.83**	100.0	**9.20 ± 4.26**	100.0	**5.44 ± 3.11**	100.0	**10.67 ± 4.44**	100.0
0.5%	**6.92 ± 4.32**	100.0	28.56 ± 10.52	100.0	16.97 ± 5.88	100.0	24.88 ± 9.84	100.0
0.1%	69.39 ± 18.59	16.7	-	0	89.15 ± 8.92	40.0	77.09 ± 13.37	20.0

(a) $n = 100$

	$p = 1.1$		$p = 1.3$		$p = 3$		$p = 5$	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	**10.03 ± 5.01**	100.0	**2.71 ± 0.76**	100.0	**1.28 ± 0.44**	100.0	**1.70 ± 0.59**	100.0
5%	**10.03 ± 5.01**	100.0	**2.71 ± 0.76**	100.0	**1.28 ± 0.44**	100.0	**1.86 ± 0.75**	100.0
1%	**10.03 ± 5.01**	100.0	**5.07 ± 1.43**	100.0	**3.25 ± 2.70**	100.0	**8.24 ± 5.95**	100.0
0.5%	**10.03 ± 5.01**	100.0	16.28 ± 4.41	100.0	**12.85 ± 7.78**	100.0	22.11 ± 12.55	100.0
0.1%	77.74 ± 12.38	53.3	94.33 ± 0.00	3.3	84.54 ± 8.39	23.3	92.00 ± 5.64	10.0

(b) $n = 300$

	$p = 1.1$		$p = 1.3$		$p = 3$		$p = 5$	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	16.39 ± 5.69	100.0	**3.34 ± 0.99**	100.0	**1.49 ± 0.27**	100.0	**1.73 ± 0.32**	100.0
5%	16.39 ± 5.69	100.0	**3.34 ± 0.99**	100.0	**1.49 ± 0.27**	100.0	**1.73 ± 0.32**	100.0
1%	16.39 ± 5.69	100.0	**5.84 ± 2.04**	100.0	**1.78 ± 0.76**	100.0	**5.34 ± 4.60**	100.0
0.5%	16.39 ± 5.69	100.0	15.11 ± 4.55	100.0	**8.43 ± 5.65**	100.0	16.18 ± 9.24	100.0
0.1%	79.12 ± 17.49	43.3	-	0	86.06 ± 7.92	13.3	95.42 ± 4.93	20.0

(c) $n = 500$

	$p = 1.1$		$p = 1.3$		$p = 3$		$p = 5$	
Error	Mean	S(%)	Mean	S(%)	Mean	S(%)	Mean	S(%)
10%	22.79 ± 6.03	100.0	**4.43 ± 0.99**	100.0	**2.28 ± 0.50**	100.0	**2.59 ± 0.35**	100.0
5%	22.79 ± 6.03	100.0	**4.43 ± 0.99**	100.0	**2.28 ± 0.50**	100.0	**2.59 ± 0.35**	100.0
1%	22.79 ± 6.03	100.0	**9.75 ± 2.11**	100.0	**2.28 ± 0.50**	100.0	**3.01 ± 1.47**	100.0
0.5%	22.79 ± 6.03	100.0	15.33 ± 3.26	100.0	**2.65 ± 0.93**	100.0	**9.10 ± 9.19**	100.0
0.1%	62.71 ± 10.08	96.7	-	0	-	0	96.30 ± 1.76	6.7

(d) $n = 1000$

Table 5: Relative times in comparison with Mosek using default accuracy for $p \in \{1.1, 1.3, 3, 5\}$ and $n \in \{100, 300, 500, 1000\}$. We wrote in bold the entries that correspond to the cases where Algorithm 2 has mean relative time less than 15% and the success rate is 100%.

Error	$p = 1.1$ Mean	S(%)	$p = 1.3$ Mean	S(%)	$p = 3$ Mean	S(%)	$p = 5$ Mean	S(%)
10%	**4.90 ± 2.34**	100.0	**2.41 ± 1.32**	100.0	**1.15 ± 0.44**	100.0	**1.58 ± 0.74**	93.3
5%	**4.90 ± 2.34**	100.0	**2.41 ± 1.32**	100.0	**1.15 ± 0.44**	100.0	**1.96 ± 0.85**	93.3
1%	**4.90 ± 2.34**	100.0	**9.57 ± 5.75**	100.0	**4.97 ± 3.12**	100.0	**11.26 ± 6.07**	93.3
0.5%	**5.21 ± 2.34**	100.0	28.76 ± 15.44	100.0	15.36 ± 6.06	100.0	25.58 ± 13.30	93.3
0.1%	80.01 ± 14.02	46.7	-	0	82.49 ± 12.78	50.0	77.75 ± 20.73	13.3

(a) $n = 100$

Error	$p = 1.1$ Mean	S(%)	$p = 1.3$ Mean	S(%)	$p = 3$ Mean	S(%)	$p = 5$ Mean	S(%)
10%	**10.26 ± 4.37**	100.0	**2.49 ± 0.63**	100.0	**1.19 ± 0.43**	100.0	**1.59 ± 0.37**	100.0
5%	**10.26 ± 4.37**	100.0	**2.49 ± 0.63**	100.0	**1.19 ± 0.43**	100.0	**1.75 ± 0.60**	100.0
1%	**10.26 ± 4.37**	100.0	**4.56 ± 1.11**	100.0	**2.95 ± 2.38**	100.0	**8.24 ± 5.76**	100.0
0.5%	**10.26 ± 4.37**	100.0	15.61 ± 5.20	100.0	**12.09 ± 8.42**	100.0	22.11 ± 11.05	100.0
0.1%	84.19 ± 11.91	50.0	-	0	90.07 ± 5.98	23.3	96.29 ± 4.92	6.7

(b) $n = 300$

Error	$p = 1.1$ Mean	S(%)	$p = 1.3$ Mean	S(%)	$p = 3$ Mean	S(%)	$p = 5$ Mean	S(%)
10%	**14.38 ± 7.55**	100.0	**3.34 ± 1.00**	100.0	**1.56 ± 0.36**	100.0	**1.76 ± 0.39**	100.0
5%	**14.38 ± 7.55**	100.0	**3.34 ± 1.00**	100.0	**1.56 ± 0.36**	100.0	**1.76 ± 0.39**	100.0
1%	**14.38 ± 7.55**	100.0	**5.86 ± 2.03**	100.0	**1.87 ± 0.83**	100.0	**5.32 ± 4.27**	100.0
0.5%	**14.38 ± 7.55**	100.0	15.22 ± 5.81	100.0	**10.02 ± 8.48**	100.0	16.30 ± 8.90	100.0
0.1%	71.77 ± 12.47	50.0	-	0	85.52 ± 5.82	20.0	87.29 ± 7.49	10.0

(c) $n = 500$

Error	$p = 1.1$ Mean	S(%)	$p = 1.3$ Mean	S(%)	$p = 3$ Mean	S(%)	$p = 5$ Mean	S(%)
10%	23.11 ± 5.76	100.0	**4.40 ± 0.72**	100.0	**2.24 ± 0.29**	100.0	**2.64 ± 0.53**	100.0
5%	23.11 ± 5.76	100.0	**4.40 ± 0.72**	100.0	**2.24 ± 0.29**	100.0	**2.64 ± 0.53**	100.0
1%	23.11 ± 5.76	100.0	**9.80 ± 1.89**	100.0	**2.24 ± 0.29**	100.0	**3.00 ± 1.17**	100.0
0.5%	23.11 ± 5.76	100.0	15.55 ± 3.37	100.0	**2.64 ± 0.98**	100.0	**9.32 ± 8.81**	100.0
0.1%	64.34 ± 13.40	100.0	-	0	-	0	90.87 ± 0.61	6.7

(d) $n = 1000$

Table 6: The setting is the same as in Table 5 except that Mosek is run with 10^{-3} complementarity gap tolerance.

As discussed in Section 4, a novel point is that the Frank-Wolfe method is actually applied to the *dual problem*, since this leads to subproblems whose solutions can be expressed in terms of minimum eigenvalues computations and conjugate vectors. In the particular case of hyperbolicity cones, we show how all the necessary objects are computable from the underlying hyperbolic polynomial. Then, in Section 5 we presented some numerical experiments that suggest that our approach has a better performance than an earlier algorithm proposed by Renegar [43]. We also compared against interior point methods. As expected, IPMs excel at getting accurate solutions but we found that our approach was often able to obtain close enough solutions with a fraction of the running time.

Still, there are a few outstanding issues that we believe could be addressed in future works. In particular, there have been many interesting works regarding Frank-Wolfe method and variants, including nonconvex extensions [33, 1, 50]. In particular, one of the most common ways to improve Frank-Wolfe methods is via the so-called *away steps* and it could be interesting to try to port some techniques to our setting by making use of the geometric properties of hyperbolicity cones and their duals. The modifications described in [15], for instance, could be promising, but they seem to require a deep knowledge of the facial structure of the underlying convex set, which may present a challenge for a set obtained by taking a compact slice of the dual of a hyperbolicity cone as we do in our approach.

Finally, a limitation of our approach is the requirement that a constant c_D (see Assumption 1) is known. To conclude this paper, we offer some thoughts on this point.

When c_D in Assumption 1 is unknown Suppose that $\mathcal{K}$ is a regular convex cone and $e \in \text{int}\, \mathcal{K}$ is arbitrary. Suppose also that (D) has an optimal solution. Since $\langle e, y \rangle > 0$ always holds for $y \in \mathcal{K}^* \setminus \{0\}$, for any such problem, there is always some c_D for which Assumption 1 is satisfied. As we saw in Section 4.4, c_D is readily available when minimizing a positive definite quadratic function. But suppose that we have a problem for which c_D is not available.

Then, one can start with any $e \in \text{int}\, \mathcal{K}$, an initial guess for c_D, say, $c_D^0 \coloneqq 1$ and run Algorithm 2 with c_D^0 and e. Let $\bar{x}^0$ and $\bar{y}^0$ denote the output of the algorithm. As remarked in the discussion about the stopping criteria, if c_D^0 is large enough so that Assumption 1 is satisfied, then $\bar{x}^0$ and $\bar{y}^0$ should be close to being zero duality gap pairs of optimal solutions to (P) and (D). On the other hand, if c_D^0 is too small, then either $\bar{x}^0$ is far from being feasible to (P) or the sum of the objective values associated to $\bar{x}^0$ and $\bar{y}^0$ has large absolute value or both phenomena happen at the same time. In that case, we may increase c_D^0 by, say, setting $c_D^1 \coloneqq 2c_D^0$ and try Algorithm 2 again with c_D^1 and e in order to obtain new solutions $\bar{x}^1$ and $\bar{y}^1$.

The summary of discussion is that, in theory, one could handle problems for which c_D is unknown by repeatedly invoking Algorithm 2 with increasingly larger guesses of c_D and stopping when the obtained solutions are sufficiently close to being primal-dual optimal for (P) and (D). This, of course, would require a very careful calibration of the stopping criterion in Algorithm 2. Although we did not explore this possibility in this paper, this might be an interesting future direction to consider.

References

[1] M. V. Balashov, B. T. Polyak, and A. A. Tremba. Gradient projection and conditional gradient methods for constrained nonconvex minimization. *Numerical Functional Analysis and Optimization*, 41(7):822–849, 2020.

[2] G. P. Barker. Theory of cones. *Linear Algebra and its Applications*, 39:263 – 291, 1981.

[3] H. H. Bauschke and J. M. Borwein. On projection algorithms for solving convex feasibility problems. *SIAM Review*, 38(3):367–426, 1996.

[4] H. H. Bauschke, O. Güler, A. S. Lewis, and H. S. Sendov. Hyperbolic polynomials and convex analysis. *Canadian Journal of Mathematics*, 53(3):470–488, 2001.

[5] D. Bini, P. Boito, Y. Eidelman, L. Gemignani, and I. Gohberg. A fast implicit QR eigenvalue algorithm for companion matrices. *Linear Algebra and its Applications*, 432(8):2006–2031, 2010.

[6] J. Bolte, C. W. Combettes, and E. Pauwels. The iterates of the Frank–Wolfe algorithm may not converge. *Mathematics of Operations Research*, Dec. 2023.

[7] I. M. Bomze, F. Rinaldi, and D. Zeffiro. Frank–Wolfe and friends: a journey into projection-free first-order optimization methods. *4OR*, 19(3):313–345, 2021.

[8] P. Brändén. Hyperbolicity cones of elementary symmetric polynomials are spectrahedral. *Optimization Letters*, 8(5):1773–1782, Jun 2014.

[9] C. Coey, L. Kapelevich, and J. P. Vielma. Solving natural conic formulations with hypatia.jl. *INFORMS Journal on Computing*, 34(5):2686–2699, Sept. 2022.

[10] V. F. Demyanov and A. M. Rubinov. *Approximate methods in optimization problems*. Number 32 in Modern analytic and computational methods in science and mathematics. Elsevier Publishing Company, 1970.

[11] J. C. Dunn. Convergence rates for conditional gradient sequences generated by implicit step length rules. *SIAM Journal on Control and Optimization*, 18(5):473–487, 1980.

[12] L. Faybusovich. Jordan-algebraic approach to convexity theorems for quadratic mappings. *SIAM Journal on Optimization*, 17(2):558–576, 2006.

[13] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[14] R. M. Freund and P. Grigas. New analysis and results for the Frank–Wolfe method. *Mathematical Programming*, 155(1–2):199–230, Nov. 2014.

[15] R. M. Freund, P. Grigas, and R. Mazumder. An extended Frank–Wolfe method with "in-face" directions, and its application to low-rank matrix completion. *SIAM Journal on Optimization*, 27(1):319–346, Jan. 2017.

[16] L. Gårding. An inequality for hyperbolic polynomials. *Journal of Mathematics and Mechanics*, pages 957–965, 1959.

[17] J. Gouveia, M. Ito, and B. F. Lourenço. Minimal hyperbolic polynomials and ranks of homogeneous cones. *arXiv preprint*, 2024.

[18] O. Güler. Hyperbolic polynomials and interior point methods for convex programming. *Mathematics of Operations Research*, 22(2):350–377, 1997.

[19] J. W. Helton and V. Vinnikov. Linear matrix inequality representation of sets. *Communications on Pure and Applied Mathematics*, 60(5):654–674, 2007.

[20] D. Henrion and J. Malick. Projection methods for conic feasibility problems: applications to polynomial sum-of-squares decompositions. *Optimization Methods and Software*, 26(1):23–46, 2011.

[21] J. Hiriart-Urruty and C. Lemarechal. *Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2013.

[22] C. A. Holloway. An extension of the Frank and Wolfe method of feasible directions. *Mathematical Programming*, 6(1):14–27, 1974.

[23] M. Ito and B. F. Lourenço. A bound on the Carathéodory number. *Linear Algebra and its Applications*, 532:347 – 363, 2017.

[24] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pages 427–435. PMLR, 2013.

[25] M. Karimi and L. Tunçel. Domain-Driven Solver (DDS) version 2.1: a MATLAB-based software package for convex optimization problems in domain-driven form. *Mathematical Programming Computation*, 16(1):37–92, Oct. 2023.

[26] M. Kummer. Spectral linear matrix inequalities. *Advances in Mathematics*, 384:107749, 2021.

[27] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. *Advances in neural information processing systems*, 28, 2015.

[28] E. S. Levitin and B. T. Polyak. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6(5):1–50, 1966.

[29] Y. Lin, S. B. Lindstrom, B. F. Lourenço, and T. K. Pong. Generalized power cones: Optimal error bounds and automorphisms. *SIAM Journal on Optimization*, 34(2):1316–1340, Apr. 2024.

[30] S. B. Lindstrom, B. F. Lourenço, and T. K. Pong. Error bounds, facial residual functions and applications to the exponential cone. *Mathematical Programming*, 200(1):229–278, Oct. 2022.

[31] S. B. Lindstrom, B. F. Lourenço, and T. K. Pong. Optimal error bounds in the absence of constraint qualifications with applications to the p-cones and beyond. *to appear at Mathematics of Operations Research*, 2024.

[32] B. F. Lourenço. Amenable cones: error bounds without constraint qualifications. *Mathematical Programming*, 186:1–48, 2021.

[33] R. Luss and M. Teboulle. Conditional gradient algorithms for rank-one matrix approximations with a sparsity constraint. *SIAM Review*, 55(1):65–98, 2013.

[34] MOSEK ApS. *MOSEK Modeling Cookbook Release 3.3.0*, 2024.

[35] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[36] Y. Nesterov. *Lectures on Convex Optimization*. Springer Optimization and Its Applications. Springer International Publishing, 2018.

[37] D. Papp and S. Yıldız. alfonso: Matlab package for nonsymmetric conic optimization. *INFORMS Journal on Computing*, 34(1):11–19, 2022.

[38] G. Pataki. The geometry of semidefinite programming. In H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors, *Handbook of semidefinite programming: theory, algorithms, and applications*. Kluwer Academic Publishers, online version at http://www.unc.edu/~pataki/papers/chapter.pdf, 2000.

[39] J. Peña, D. Rodríguez, and N. Soheili. On the von Neumann and Frank–Wolfe algorithms with away steps. *SIAM Journal on Optimization*, 26(1):499–512, 2016.

[40] S. Pokutta. The Frank-Wolfe algorithm: A short introduction. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 126(1):3–35, Dec. 2023.

[41] J. Renegar. Hyperbolic programs, and their derivative relaxations. *Foundations of Computational Mathematics*, 6(1):59–79, Jan. 2006.

[42] J. Renegar. "Efficient" subgradient methods for general convex optimization. *SIAM Journal on Optimization*, 26(4):2649–2676, 2016.

[43] J. Renegar. Accelerated first-order methods for hyperbolic programming. *Mathematical Programming*, 173(1):1–35, 2019.

[44] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1997.

[45] R. Sanyal. On the derivative cones of polyhedral cones. *Advances in Geometry*, 13(2):315–321, 2013.

[46] J. Saunderson. A spectrahedral representation of the first derivative relaxation of the positive semidefinite cone. *Optimization Letters*, 12(7):1475–1486, Oct 2018.

[47] J. Saunderson and P. A. Parrilo. Polynomial-sized semidefinite representations of derivative relaxations of spectrahedral cones. *Mathematical Programming*, 153(2):309–331, Nov 2015.

[48] D. Sun and J. Sun. Löwner's operator and spectral functions in Euclidean Jordan algebras. *Mathematics of Operations Research*, 33(2):421–445, 2008.

[49] P. Wolfe. Convergence theory in nonlinear programming. *Integer and nonlinear programming*, pages 1–36, 1970.

[50] L. Zeng, Y. Zhang, G. Li, and T. K. Pong. Frank-Wolfe-type methods for a class of nonconvex inequality-constrained problems. *Mathematical Programming*, 2024.

[51] Y. Zinchenko. On hyperbolicity cones associated with elementary symmetric polynomials. *Optimization Letters*, 2(3):389–402, Jun 2008.

A A proof that the polynomial in Proposition 3.8 is not isometric

We give a detailed proof that

$$p(x) = (x_1 + x_2 + x_3)(x_1 - x_2 + x_3)(2x_1 - x_2 - x_3)(x_1 + 2x_2 - x_3)$$

is hyperbolic with respect to $e = (0, 0, 1)$, but not an isometric hyperbolic polynomial.

Proof. The roots of $p(x - te)$ are

$$r_1(x) = x_1 + x_2 + x_3, \quad r_2(x) = x_1 - x_2 + x_3,$$
$$r_3(x) = -2x_1 + x_2 + x_3, \ r_4(x) = -x_1 - 2x_2 + x_3,$$

which are all real for $x \in \mathbb{R}^3$. Since $p(e) > 0$, p is hyperbolic along e.

To prove p is not isometric, we show that, for $z = (3, 1, 0)$ and $y = (-1, 0, 0)$, there is no $w \in \mathbb{R}^3$ such that $\lambda(w) = \lambda(z)$ and $\lambda(w + y) = \lambda(w) + \lambda(y)$. First we show that

$$\lambda(w) = \lambda(z) \Rightarrow w = z.$$

Let $w \in \mathbb{R}^3$ be such that $\lambda(w) = \lambda(z)$ holds. We start by observing that $\lambda(z) = (4, 2, -5, -5)$, so z has an eigenvalue of multiplicity two. Therefore, if $\lambda(w) = \lambda(z)$, at least two of $r_1(w), r_2(w), r_3(w), r_4(w)$ must be the same. We consider all possible cases.

(i) $\boxed{r_1(w) = r_2(w).}$ This case happens if and only if $w_2 = 0$. We have the following subcases.

  (a) If $w_1 > 0$, then $r_1(w) = r_2(w) > r_4(w) > r_3(w)$ holds, i.e., the two largest eigenvalues of w are equal. Therefore, $\lambda(w)$ can not be $\lambda(z)$, because the two smallest components of $\lambda(z)$ are equal.

  (b) If $w_1 = 0$, then $r_1(w) = r_2(w) = r_3(w) = r_4(w)$ holds. Similarly, $\lambda(w)$ can not be $\lambda(z)$.

  (c) If $w_1 < 0$, then $r_3(w) > r_4(w) > r_2(w) = r_1(w)$ holds. Therefore,

$$\lambda(w) = \lambda(z) \Rightarrow \begin{cases} r_3(w) = 4 \\ r_4(w) = 2 \\ r_2(w) = r_1(w) = -5 \end{cases}$$

  However, there does not exist w which satisfies these equalities.

(ii) $\boxed{r_1(w) = r_3(w).}$ This case happens if and only if $w_1 = 0$. We have the following subcases.

    (a) If $w_2 > 0$, then $r_1(w) = r_3(w) > r_2(w) > r_4(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

    (b) If $w_2 = 0$, then $r_1(w) = r_2(w) = r_3(w) = r_4(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

    (c) If $w_2 < 0$, then $r_4(w) > r_2(w) > r_3(w) = r_1(w)$ holds. Therefore,

$$\lambda(w) = \lambda(z) \Rightarrow \begin{cases} r_4(w) = 4 \\ r_2(w) = 2 \\ r_1(w) = r_3(w) = -5 \end{cases}$$

However, there does not exist w which satisfies these equalities.

(iii) $\boxed{r_1(w) = r_4(w).}$ This case happens if and only if $2w_1 = -3w_2$. We have the following subcases.

    (a) If $w_1 > 0$, then $r_2(w) > r_1(w) = r_4(w) > r_3(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

    (b) If $w_1 = 0$, then $r_1(w) = r_2(w) = r_3(w) = r_4(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

    (c) If $w_1 < 0$, then $r_3(w) > r_4(w) = r_1(w) > r_2(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

(iv) $\boxed{r_2(w) = r_3(w).}$ This case happens if and only if $3w_1 = 2w_2$. We have the following subcases.

    (a) If $w_1 > 0$, then $r_1(w) > r_2(w) = r_3(w) > r_4(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

    (b) If $w_1 = 0$, then $r_1(w) = r_2(w) = r_3(w) = r_4(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

    (c) If $w_1 < 0$, then $r_4(w) > r_3(w) = r_2(w) > r_1(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

(v) $\boxed{r_2(w) = r_4(w).}$ This case happens if and only if $2w_1 = -w_2$. We have the following subcases.

    (a) If $w_1 > 0$, then $r_2(w) = r_4(w) > r_1(w) > r_3(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

    (b) If $w_1 = 0$, then $r_1(w) = r_2(w) = r_3(w) = r_4(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

    (c) If $w_1 < 0$, then $r_3(w) > r_1(w) > r_2(w) = r_4(w)$ holds. Therefore,

$$\lambda(w) = \lambda(z) \Rightarrow \begin{cases} r_3(w) = 4 \\ r_1(w) = 2 \\ r_2(w) = r_4(w) = -5 \end{cases}$$

However, there does not exist w which satisfies these equalities.

(vi) $\boxed{r_3(w) = r_4(w).}$ This case happens if and only if $w_1 = 3w_2$. We have the following subcases.

    (a) If $w_1 > 0$, then $r_1(w) > r_2(w) > r_3(w) = r_4(w)$ holds. Therefore,

$$\lambda(w) = \lambda(z) \Rightarrow \begin{cases} r_1(w) = 4 \\ r_2(w) = 2 \\ r_3(w) = r_4(w) = -5 \end{cases} \quad \Longleftrightarrow \quad \begin{cases} w_1 = 3 \\ w_2 = 1 \\ w_3 = 0 \end{cases} \quad \Longleftrightarrow \quad w = z$$

    (b) If $w_1 = 0$, then $r_1(w) = r_2(w) = r_3(w) = r_4(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

    (c) If $w_1 < 0$, then $r_4(w) = r_3(w) > r_2(w) > r_1(w)$ holds. Therefore, $\lambda(w)$ can not be $\lambda(z)$.

The summary of all the six cases and subcases above is that the sole possibility for $\lambda(w) = \lambda(z)$ is case (vi).(a) where we have $w = z$. That is,

$$\lambda(w) = \lambda(z) \Rightarrow w = z.$$

Moreover, $(3, 1, -3, -4) = \lambda(z + y)$ is different from $\lambda(z) + \lambda(y) = (6, 3, -6, -6)$. Therefore, p is not isometric. $\qquad\square$