This article was downloaded by: [192.16.191.136] On: 26 November 2020, At: 02:50
Publisher: Institute for Operations Research and the Management Sciences (INFORMS)
INFORMS is located in Maryland, USA

Mathematics of Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Rescaling Algorithms for Linear Conic Feasibility

Daniel Dadush, László A. Végh, Giacomo Zambelli

To cite this article:
Daniel Dadush, László A. Végh, Giacomo Zambelli (2020) Rescaling Algorithms for Linear Conic Feasibility. Mathematics of Operations Research 45(2):732-754. https://doi.org/10.1287/moor.2019.1011

Full terms and conditions of use: https://pubsonline.informs.org/Publications/Librarians-Portal/PubsOnLine-Terms-and-Conditions

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2020, INFORMS

Please scroll down for article—it is on subsequent pages

With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes.
For more information on INFORMS, its publications, membership, or meetings visit http://www.informs.org

http://pubsonline.informs.org/journal/moor

MATHEMATICS OF OPERATIONS RESEARCH
Vol. 45, No. 2, May 2020, pp. 732–754
ISSN 0364-765X (print), ISSN 1526-5471 (online)

Rescaling Algorithms for Linear Conic Feasibility

Daniel Dadush,[a] László A. Végh,[b] Giacomo Zambelli[b]

[a] Centrum Wiskunde & Informatica, 1098 XG Amsterdam, Netherlands; [b] Department of Mathematics, London School of Economics and Political Science, London WC2A 2AE, United Kingdom

Contact: dadush@cwi.nl, https://orcid.org/0000-0001-5577-5012 (DD); l.vegh@lse.ac.uk, http://orcid.org/0000-0003-1152-200X (LAV); g.zambelli@lse.ac.uk, http://orcid.org/0000-0003-3147-3938 (GZ)

Received: December 17, 2017
Revised: September 20, 2018
Accepted: April 4, 2019
Published Online in Articles in Advance: January 23, 2020

MSC2010 Subject Classification: Primary: 90C05
OR/MS Subject Classification: Primary: Programming/linear/algorithms/theory

https://doi.org/10.1287/moor.2019.1011

Copyright: © 2020 INFORMS

Abstract. We propose simple polynomial-time algorithms for two linear conic feasibility problems. For a matrix $A \in \mathbb{R}^{m\times n}$, the *kernel problem* requires a positive vector in the kernel of A, and the *image problem* requires a positive vector in the image of $A^\top$. Both algorithms iterate between simple first-order steps and rescaling steps. These rescalings improve natural geometric potentials. If Goffin's condition measure ρ_A is negative, then the kernel problem is feasible, and the worst-case complexity of the kernel algorithm is $O((m^3n + mn^2)\log|\rho_A|^{-1})$; if $\rho_A > 0$, then the image problem is feasible, and the image algorithm runs in time $O(m^2n^2 \log \rho_A^{-1})$. We also extend the image algorithm to the oracle setting. We address the degenerate case $\rho_A = 0$ by extending our algorithms to find maximum support nonnegative vectors in the kernel of A and in the image of $A^\top$. In this case, the running time bounds are expressed in the bit-size model of computation: for an input matrix A with integer entries and total encoding length L, the maximum support kernel algorithm runs in time $O((m^3n + mn^2)L)$, whereas the maximum support image algorithm runs in time $O(m^2n^2L)$. The standard linear programming feasibility problem can be easily reduced to either maximum support problems, yielding polynomial-time algorithms for linear programming.

Funding: D. Dadush was supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek [Veni Grant 639.071.510]. L. A. Végh was supported by the Engineering and Physical Sciences Research Council [First Grant EP/M02797X/1] and by the European Research Council.

Keywords: linear programming • rescaling • condition measure

1. Introduction

We consider two fundamental linear conic feasibility problems for an $m \times n$ matrix A. In the *kernel problem*, the goal is to find a positive vector in $\ker(A)$, whereas in the *image problem*, the goal is to find a positive vector in $\operatorname{im}(A^\top)$. These can be formulated by the following feasibility problems:

$$\begin{aligned} Ax &= 0 \\ x &> 0 \end{aligned} \qquad (K_{++}) \qquad\qquad A^\top y > 0 \qquad (I_{++})$$

We present simple polynomial-time algorithms for the kernel problem (K_{++}) and the image problem (I_{++}). Both algorithms combine a first-order method with a geometric rescaling, which improves natural volumetric potentials.

The algorithms we propose fit into a line of research developed over the past 15 years (Basu [2], Belloni et al. [3], Betke [4], Chubanov [7, 8, 9], Dunagan and Vempala [15], Li et al. [22], Peña and Soheili [27, 28], Roos [30], Végh and Zambelli [35]). These are polynomial algorithms for linear programming (LP) that combine simple iterative updates, such as variants of the perceptron (Rosenblatt [31]) or of the relaxation method (Agmon [1], Motzkin and Schoenberg [23]), with some form of geometric rescaling.

Problems (K_{++}) and (I_{++}) have the following natural geometric interpretations. Let $a_1, \dots, a_n$ denote the columns of the matrix A. A feasible solution to (K_{++}) means that 0 is in the relative interior of the convex hull of the columns a_i, whereas a feasible solution to (I_{++}) gives a hyperplane that strictly separates 0 from the convex hull. We measure the running time of algorithms for (K_{++}) and (I_{++}) in terms of Goffin's [18] condition measure ρ_A, where $|\rho_A|$ is the distance of 0 from the relative boundary of the convex hull of the vectors $a_i/\|a_i\|$, $i \in [n]$. If $\rho_A < 0$, then (K_{++}) is feasible; if $\rho_A > 0$, then (I_{++}) is feasible.

In case $\rho_A = 0$, 0 falls on the relative boundary of the convex hull of the a_i's, and both problems (K_{++}) and (I_{++}) are infeasible. We extend our kernel and image algorithms to finding *maximum support nonnegative vectors* in $\ker(A)$ and in $\operatorname{im}(A^\top)$, respectively. Geometrically, these amount to identifying the face of the convex hull that contains 0 in its relative interior. By strong duality, the two maximum supports are complementary to each other. The maximum support problems provide fine-grained structural information on LP and are crucial for exact LP algorithms (see, e.g., Vavasis and Ye [34]). With either the maximum support kernel or maximum support image algorithm, one can solve a general LP feasibility problem of the form $Ax \leq b$ via simple homogenization. Although LP feasibility (and thus LP optimization) can also be reduced either to (K_{++}) or to (I_{++}) via standard perturbation methods (see, e.g., Schrijver [32]), this is not desirable for numerical stability. Furthermore, we recall that the reduction from LP optimization to feasibility creates degenerate (i.e., non-full-dimensional) systems by construction, and hence in this sense, most "interesting" LP feasibility problems are degenerate.

1.1. Previous Work

We give a brief overview of geometric rescaling algorithms that combine first-order iterations and rescalings. The first such algorithms were given by Betke [4] and Dunagan and Vempala [15]. Both papers address the problem (I_{++}). The deterministic algorithm of Betke [4] combines a variant of Wolfe's [36] algorithm with a rank 1 update to the matrix A. Progress is measured by showing that the spherical volume of the cone $A^\top y \geq 0$ increases by a constant factor at each rescaling. This approach was further improved by Peña and Soheili [27] using different first-order methods, in particular, a smoothed perceptron algorithm (Nesterov [25], Soheili and Peña [33]). Dunagan and Vempala [15] give a randomized algorithm, combining two different first-order methods. They also use a rank 1 update, but a different one from Betke [4], and can show progress directly in terms of Goffin's [18] condition measure ρ_A. The problem (I_{++}) has been also studied in the more general oracle setting (Belloni et al. [3], Chubanov [10], Peña and Soheili [27, 28]).

For (K_{++}), as well as for the maximum support version, a rescaling algorithm was given by Chubanov [9] (see also Li et al. [22], Roos [30]). A main iteration of the algorithm concludes that in the system $Ax = 0, 0 \leq x \leq \vec{e}$, one can identify at least one index i such that $x_i \leq 1/2$ must hold for every solution. Hence, the rescaling multiplies A from the right-hand side by a diagonal matrix. (This is in contrast to the above-mentioned algorithms, where rescaling multiplies the matrix A from the left-hand side.) The first-order iterations are von Neumann steps on the system defined by the projection matrix.

The algorithm by Chubanov [9] builds on previous work by Chubanov [6, 7] on binary integer programs and linear feasibility (see also Basu [2]). A more efficient variant of this algorithm was given by Végh and Zambelli [35]. These algorithms use a similar rescaling, but for a nonhomogeneous linear system, and the first-order iterations are variants of the relaxation method.

1.2. Our Contributions

We introduce new algorithms for (K_{++}) and (I_{++}), as well as for their maximum support versions, and improve on the previous best geometric rescaling algorithm running time bounds in each of the settings. For the *kernel problem*, that is, if $\rho_A < 0$, we present a simple new algorithm whose running time analysis is based on a new volumetric potential that serves as a proxy for $|\rho_A|$. In contrast, Chubanov [9], in essence, reduces the problem to the image setting. Using a direct volumetric argument enables us to obtain a better running time in $O\big((m^3n + mn^2)\log|\rho_A|^{-1}\big)$ arithmetic operations.

For the *image problem*, that is, if $\rho_A > 0$, our new algorithm is an enhanced version of Betke's [4] and Peña and Soheili's [27] algorithms. In contrast to rank 1 updates used in these papers, we use higher-rank updates. The running time of our algorithm is $O(m^2n^2\log\rho_A^{-1})$. This can be improved to $O\Big(m^3n\sqrt{\log n}\cdot\log\rho_A^{-1}\Big)$ using smoothing techniques (Nesterov [25], Soheili and Peña [33], Yu et al. [38]). We also present an extension of our algorithm for the case where the interior of a cone Σ expressed by a separation oracle; the algorithm requires $O(m^3\log\rho_\Sigma^{-1})$ oracle calls and $O(m^5\log\rho_\Sigma^{-1})$ arithmetic operations, where ρ_Σ is the cone width. This oracle variant was used by Dadush et al. [12] to develop new polynomial and strongly polynomial algorithms for submodular function minimization.

We can obtain algorithms for the *maximum support versions* in both settings by repeatedly applying the full support algorithm and observing the rescaled length of the column vectors. We show that if a column vector becomes too long in the kernel setting (or too short in the image setting) after a number of rescalings, then it cannot be contained in the maximum support. Thus, we can remove such vectors and restart the algorithm. For the maximum support image problem, we obtain the first rescaling algorithm.

These algorithms offer a particularly simple approach for degenerate problems, even though these typically require substantial additional effort compared with the full-dimensional setting. For example, in the ellipsoid method, the simultaneous Diophantine approximation technique is used (Grötschel et al. [19]). For interior point methods, degeneracy must be dealt with both in the initialization phase, to set up a full-dimensional auxiliary system, and in the termination phase, to round to the optimal face.

The full support algorithms can be implemented in the *real model of computation* (Blum et al. [5]), and the algorithms do not require an estimation of the value of $|\rho_A|$. In contrast, for the maximum support algorithms, we need bounds on the bit complexity of the input to determine the threshold for removing columns. Thus, we assume that the input matrix A is an integer of total encoding length L. In this setting, $|\rho_A| \geq 2^{-O(L)}$ whenever $\rho_A \neq 0$. This provides running time bounds $O\left((m^3n + mn^2) \cdot L\right)$ for the full support kernel algorithm and $O\left(m^3 n\sqrt{\log n} \cdot L\right)$ for the full support image algorithm in the bit-complexity model. For the maximum support variants, the above-described column elimination method requires n calls to the full support algorithm in the kernel setting and m calls in the image setting. In Appendix A, we present improved versions for the maximum support variants of both the kernel and image problems that can be implemented in the same asymptotic complexity as the respective full support variants. A summary of running times is given in Table 1.

The full support kernel algorithm was first presented in the conference paper by Dadush et al. [11]. The image algorithm and the maximum support variants for both the kernel and dual problems are new in this paper. The full support image algorithm was also independently obtained by Hoberg and Rothvoß [20].

The rest of this paper is structured as follows. Section 1.3 introduces notation and important concepts. Section 1.4 briefly surveys relevant first-order methods. In Sections 2 and 3, we give the algorithms for the full support problems (K_{++}) and (I_{++}), respectively. These are extended in Section 4 to the maximum support cases. Variants with improved running times are given in Appendix A.

1.3. Notation and Preliminaries

For a natural number n, let $[n] = \{1, 2, \ldots, n\}$. For a subset $X \subseteq [n]$, we let $A_X \in \mathbb{R}^{m\times|X|}$ denote the submatrix formed by the columns of A indexed by X. For any nonzero vector $v \in \mathbb{R}^m$, we denote by $\hat{v}$ the normal vector in the direction of v; that is,

$$\hat{v} \overset{\text{def}}{=} \frac{v}{\|v\|}.$$

By convention, we also define $\hat{0} = 0$. We let $\hat{A} \overset{\text{def}}{=} [\hat{a}_1, \ldots, \hat{a}_n]$. Note that given $v, w \in \mathbb{R}^m \setminus \{0\}$, $\hat{v}^\top \hat{w}$ is the cosine of the angle between them. Given a vector $x \in \mathbb{R}^n$, the *support of* x is the subset of $[n]$ defined by $\operatorname{supp}(x) \overset{\text{def}}{=} \{i \in [n] : x_i \neq 0\}$.

Let $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n : x \geq 0\}$ and $\mathbb{R}^n_{++} = \{x \in \mathbb{R}^n : x > 0\}$ denote the set of nonnegative and positive vectors in $\mathbb{R}^n$, respectively. For any set $H \subseteq \mathbb{R}^n$, we let $H_+ \overset{\text{def}}{=} H \cap \mathbb{R}^n_+$ and $H_{++} \overset{\text{def}}{=} H \cap \mathbb{R}^n_{++}$. These notations will be used in particular for the kernel and image spaces

$$\ker(A) \overset{\text{def}}{=} \{x \in \mathbb{R}^n : Ax = 0\}, \quad \operatorname{im}\left(A^\top\right) \overset{\text{def}}{=} \left\{A^\top y : y \in \mathbb{R}^m\right\}.$$

Clearly, $\operatorname{im}(A^\top) = \ker(A)^\perp$. Using this notation, (K_{++}) is the problem of finding a point in $\ker(A)_{++}$, and (I_{++}) amounts to finding a point in $\operatorname{im}(A^\top)_{++}$. By strong duality, (K_{++}) is feasible if and only if $\operatorname{im}(A^\top)_+ = \{0\}$; that is,

$$A^\top y \geq 0, \tag{I}$$

Table 1. Running times of geometric rescaling algorithms. In the oracle setting, SO is the complexity of a separation oracle call.

Kernel problem	
Full support	Maximum support
$O(n^{18+3\varepsilon} \cdot L^{12+2\varepsilon})$ (Basu [2], Chubanov [6])	$O(n^4 \cdot L)$ (Chubanov [9])
$O([n^5/\log n] \cdot L)$ (Végh and Zambelli [35])	$O((m^3n + mn^2) \cdot L)$ (this paper)
$O(n^4 \cdot L)$ (Chubanov [9])	
$O\left((m^3n + mn^2) \cdot \log|\rho_A|^{-1}\right)$ (this paper)	

Image problem		
Full support	Full support oracle model	Maximum support
$O(m^3n^3 \cdot \log \rho_A^{-1})$ (Betke [4])	$O((\text{SO} \cdot m^5 + m^6)\rho_\Sigma^{-1})$ (Peña and Soheili [27])	$O\left(m^3 n\sqrt{\log n} \cdot L\right)$ (this paper)
$O(m^4 n \log m \cdot \rho_A^{-1})$ (Dunagan and Vempala [15])	$O((\text{SO} \cdot m^4 + m^6)\rho_\Sigma^{-1})$ (Chubanov [10])	
$O\left(m^{2.5}n^2\sqrt{\log n} \cdot \log \rho_A^{-1}\right)$ (Peña and Soheili [27])	$O((\text{SO} \cdot m^3 + m^5)\rho_\Sigma^{-1})$ (this paper)	
$O\left(m^3 n\sqrt{\log n} \cdot \log \rho_A^{-1}\right)$ (this paper)		

has no solution other than $y \in \ker(A^\top)$. Similarly, (I_{++}) is feasible if and only if $\ker(A)_+ = \{0\}$; that is,

$$\begin{aligned} Ax &= 0, \\ x &\geq 0 \end{aligned} \tag{K}$$

has no solution other than $x = 0$. Let us define

$$\Sigma_A \overset{\text{def}}{=} \left\{ y \in \mathbb{R}^m : A^\top y \geq 0 \right\}$$

as the set of solutions to (I).

Let I_d denote the d-dimensional identity matrix. Let $\vec{e}_j$ denote the jth unit vector, and let $\vec{e}$ denote the all-ones vector of appropriate dimension (depending on the context). We denote by $\mathbb{B}^d$ the unit ball centered at the origin in $\mathbb{R}^d$, and let $\nu_d = \operatorname{vol}(\mathbb{B}^d)$.

Given any set C contained in $\mathbb{R}^d$, we denote by $\operatorname{span}(C)$ the linear subspace of $\mathbb{R}^d$ spanned by the elements of C. If $C \subseteq \mathbb{R}^d$ has affine dimension r, we denote by $\operatorname{vol}_r(C)$ the r-dimensional volume of C.

1.3.1. Projection Matrices. For any matrix $A \in \mathbb{R}^{m \times n}$, we denote by Π_A^I the *orthogonal projection matrix* to $\operatorname{im}(A^\top)$ and by Π_A^K the orthogonal projection matrix to $\ker(A)$. We recall that

$$\Pi_A^I = A^\top \left(AA^\top \right)^+ A, \quad \Pi_A^K = I_n - A^\top \left(AA^\top \right)^+ A,$$

where $(\cdot)^+$ denotes the Moore–Penrose pseudoinverse. Note that in order to compute Π_A^I and Π_A^K, we do not need to compute the pseudoinverse of $AA^\top$; instead, if we let B be a matrix comprised of $\operatorname{rk}(A)$ many linearly independent rows of A, then $\Pi_A^I = B^\top (BB^\top)^{-1} B$, which can be computed in $O(n^2 m)$ arithmetic operations.

1.3.2. Scalar Products. We will often need to use scalar products and norms other than the Euclidean ones. Let $\mathbb{S}_+^d$ and $\mathbb{S}_{++}^d$ be the sets of symmetric $d \times d$ positive semidefinite and positive definite matrices, respectively. Given $Q \in \mathbb{S}_{++}^d$, we denote by $Q^{\frac{1}{2}}$ the *square root* of Q, that is, the unique matrix in $\mathbb{S}_{++}^d$ such that $Q = Q^{\frac{1}{2}} Q^{\frac{1}{2}}$, and by $Q^{-\frac{1}{2}}$ the inverse of $Q^{\frac{1}{2}}$. For $Q \in \mathbb{S}_{++}^d$ and two vectors $v, w \in \mathbb{R}^d$, we let

$$\langle v, w \rangle_Q \overset{\text{def}}{=} v^\top Q w, \quad \|v\|_Q \overset{\text{def}}{=} \sqrt{\langle v, v \rangle_Q}.$$

These define a scalar product and a norm over $\mathbb{R}^d$. We use $\|\cdot\|_1$ for the L^1-norm and $\|\cdot\|_2$ for the Euclidean norm. When there is no risk of confusion, we will simply write $\|\cdot\|$ for $\|\cdot\|_2$. Furthermore, for any $Q \in \mathbb{S}_{++}^d$, we define the ellipsoid

$$E(Q) \overset{\text{def}}{=} \left\{ z \in \mathbb{R}^d : \|z\|_Q \leq 1 \right\},$$

and we recall that $E(Q) = Q^{-\frac{1}{2}} \mathbb{B}^d$ and $\operatorname{vol}(E(Q)) = \det(Q)^{-\frac{1}{2}} \nu_d$.

We will use the following simple properties of matrix traces.

Lemma 1. *For any $X \in \mathbb{S}_+^d$,*

a. $\det(I_d + X) \geq 1 + \operatorname{tr}(X)$.
b. $\det(X)^{1/m} \leq \operatorname{tr}(X)/m$.

Proof. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$ denote the eigenvalues of X. (a) Then $\det(I_d + X) = \prod_{i=1}^d (1 + \lambda_i) \geq 1 + \sum_{i=1}^d \lambda_i = 1 + \operatorname{tr}(X)$, where the equality is by the nonnegativity of the λ_i values. (b) By the inequality of arithmetic and geometric means, $\det(X)^{1/m} = (\prod_{i=1}^d \lambda_i)^{1/m} \leq \sum_{i=1}^d \lambda_i / m = \operatorname{tr}(X)/m$. □

1.3.3. The Goffin Measure. The running times of our full support algorithms will depend on the following condition measure introduced by Goffin [18]. Given $A \in \mathbb{R}^{m \times n}$, we define

$$\rho_A \overset{\text{def}}{=} \max_{y \in \operatorname{im}(A) \setminus \{0\}} \min_{j \in [n]} \hat{a}_j^\top \hat{y}. \tag{1}$$

We remark that, in the literature, A is typically assumed to have full row rank (i.e., $\operatorname{rk}(A) = m$), in which case y in the preceding definition ranges over all of $\mathbb{R}^m$. However, in some parts of this paper, it will be convenient not to make such an assumption. The following lemma summarizes well-known properties of ρ_A. The proof will be given in Appendix B for completeness. For the matrix A, we let $\operatorname{conv}(A)$ denote the convex hull of the column vectors of A.

Lemma 2. For any $A \in \mathbb{R}^{m\times n}$, $|\rho_A|$ *equals the distance of* 0 *from the relative boundary of* $\mathrm{conv}(\hat{A})$. *Furthermore,*

a. $\rho_A < 0$ *if and only if* 0 *is in the relative interior of* $\mathrm{conv}(A)$.

b. $\rho_A > 0$ *if and only if* 0 *is outside* $\mathrm{conv}(A)$. *In this case, the Goffin measure* ρ_A *equals the width of the image cone* Σ_A, *that is, the radius of the largest ball in* $\mathbb{R}^m$ *centered on the surface of the unit sphere and inscribed in* Σ_A.

The following quantities will be needed for bit complexity estimations. Assume that the $m \times n$ matrix A has integer entries and encoding size L. Letting $\mathcal{B} = \{B \subseteq [n] : |B| = \mathrm{rk}(A_B)\}$, we define

$$\Delta_A = \max_{B\in\mathcal{B}} \prod_{j\in B} \|a_j\| \quad \text{and} \quad \theta_A = \frac{1}{m^2 \Delta_A{}^2}. \tag{2}$$

Lemma 3. *Let* $A \in \mathbb{Z}^{m\times n}$ *of total encoding length L. If* $\rho_A \neq 0$, *then* $|\rho_A| \geq \theta_A \geq 2^{-4L}$.

The proof can be found in Appendix B. Let us note that Δ_A and θ_A can be efficiently computed. Indeed, the value of Δ_A is the maximum weight base of a linear matroid and can be computed by the greedy algorithm in $O(m^2 n + n \log n)$ arithmetic operations because this amounts to sorting the columns of A according to their length and then applying Gaussian elimination (which requires $O(m^2 n)$ operations).

1.4. First-Order Algorithms

Various first-order methods are known for finding nonzero solutions to (*K*) or to (*I*). Most algorithms assume either the feasibility of (K_{++}) or the feasibility of (I_{++}). We outline the common update steps of these algorithms.

At every iteration, maintain a nonnegative, nonzero vector $x \in \mathbb{R}^n$, and let $y = Ax$. If $y = 0$, then x is a nonzero point in $\ker(A)_+$. If $A^\top y > 0$, then $A^\top y \in \mathrm{im}(A)_{++}$. Otherwise, choose an index $k \in [n]$ such that $a_k^\top y \leq 0$, and update x and y as follows:

$$y' := \alpha y + \beta \hat{a}_k, \quad x' := \alpha x + \frac{\beta}{\|a_k\|}\vec{e}_k, \tag{3}$$

where $\alpha, \beta > 0$ depend on the specific algorithm. Below we discuss various possible update choices.

1.4.1. Von Neumann Algorithm.

The vector y is maintained throughout as a convex combination of $\hat{a}_1, \ldots, \hat{a}_n$. The parameters $\alpha, \beta > 0$ are chosen so that $\alpha + \beta = 1$ and $\|y'\|$ is the smallest possible; that is, y' is the point of minimum norm on the line segment joining y and $\hat{a}_k$. If we denote by y^t the vector at iteration t and initialize $y^1 = \hat{a}_k$ for an arbitrary $k \in [n]$, a simple argument shows that $\|y^t\| \leq 1/\sqrt{t}$ (see Dantzig [14]). Epelman and Freund [16] showed that if 0 is contained in the interior of the convex hull, that is, $\rho_A < 0$, then $\|y^t\|$ decreases by a factor of $\sqrt{1-\rho_A^2}$ in every iteration.

If 0 is outside the convex hull, that is, $\rho_A > 0$, then the algorithm terminates after at most $1/\rho_A^2$ iterations. A recent result by Peña et al. [29] gives a variant of the algorithm with a provable convergence guarantee in the case $\rho_A = 0$, that is, if 0 is on the boundary of the convex hull.

Among the preceding geometric rescaling algorithms, variants of the von Neumann algorithm have been used by Betke [4] for the case $\rho_A > 0$ and by Chubanov [9] for the case $\rho_A < 0$. We will use this method in our image algorithm, that is, for $\rho_A > 0$.

We note that the von Neumann algorithm is the same as the Frank and Wolfe [17] conditional gradient descent method with optimal step size for the quadratic program $\min \|Ax\|^2$ subject to (s.t.) $\vec{e}^\top x = 1, x \geq 0$.

1.4.2. Perceptron Algorithm.

The perceptron algorithm chooses $\alpha = \beta = 1$ at every iteration. If $\rho_A > 0$, then, similarly to the von Neumann algorithm, the perceptron algorithm terminates with a solution to the system (I_{++}) after at most $1/\rho_A^2$ iterations (see Novikoff [26]). The perceptron and von Neumann algorithms can be interpreted as duals of each other (see Li and Terlaky [21]).

Soheili and Peña [33] gave a smoothed variant of the perceptron update that guarantees termination in time $O(\sqrt{\log n}/|\rho_A|)$ iterations. This was used in the polynomial-time rescaling algorithm (Peña and Soheili [27]) for $\rho_A > 0$. The same iteration bound $O(\sqrt{\log n}/|\rho_A|)$ was achieved by the Mirror Prox method of Yu et al. [38] by adapting previous work of Nemirovski [24].

With a normalization to $\vec{e}^\top x = 1$, the perceptron algorithm implements the Frank–Wolfe algorithm for the same system, $\min \|Ax\|^2$ s.t. $\vec{e}^\top x = 1, x \geq 0$, with step length $1/(k+1)$ at iteration k. An alternative view is to interpret the perceptron algorithm as a coordinate descent method for the system $\min \|Ax\|^2$ s.t. $x \geq \vec{e}$, with fixed step length 1 at every iteration.

1.4.3. Dunagan and Vempala (DV) Algorithm. The first-order method used in Dunagan and Vempala [15] chooses $\alpha = 1$ and $\beta = -(\hat{a}_k^\top y)$. The choice of β is the one that makes $\|y'\|$ the smallest possible when $\alpha = 1$. It follows that $\|y'\|^2 = \|y\|^2 + 2\beta(\hat{a}_k^\top y) + \beta^2\|\hat{a}_k\|^2 = \|y\|^2 - 2(\hat{a}_k^\top y)^2 + (\hat{a}_k^\top y)^2 = \|y\|^2(1-(\hat{a}_k^\top \hat{y})^2)$, and hence

$$\|y'\| = \|y\|\sqrt{1-(\hat{a}_k^\top \hat{y})^2}. \tag{4}$$

In particular, the norm of y' decreases at every iteration, and the larger the angle between a_k and y, the larger is the decrease. If $\rho_A < 0$, then $|\hat{a}_k^\top \hat{y}| \geq |\rho_A|$; therefore, this guarantees a decrease in the norm of at least $\sqrt{1-\rho_A^2}$.

This is a coordinate descent for the system $\min \|Ax\|^2$ s.t. $x \geq \vec{e}$ but with the optimal step length. One can also interpret it as the Frank–Wolfe algorithm with the optimal step length for the same system.[1]

Whereas Dunagan and Vempala [15] used these updates for $\rho_A > 0$, we will use them in our kernel algorithm for $\rho_A < 0$.

2. The Full Support Kernel Algorithm

The full support kernel algorithm (Algorithm 1) solves the full support problem (K_{++}); that is, it finds a point in $\ker(A)_{++}$, assuming that $\rho_A < 0$ or, equivalently, $\ker(A)_{++} \neq \emptyset$. We assume that the columns of input matrix A are normalized; that is, $A = \hat{A}$. However, this property does not hold throughout the algorithm because the rescalings will modify the length of the columns. We use the parameter

$$\varepsilon \overset{\text{def}}{=} \frac{1}{11m}. \tag{5}$$

Algorithm 1 (Full Support Kernel Algorithm)

```
Input: A matrix $A \in \mathbb{R}^{m\times n}$ such that $\rho_A < 0$ and $\|a_j\| = 1$ for all $j \in [n]$
Output: A solution to the system $(K_{++})$
1: Compute $\Pi := \Pi_A^K = I_n - A^\top(AA^\top)^+ A$
2: Set $x_j := 1$ for all $j \in [n]$ and $y := Ax$
3: while $\Pi x \not> 0$ do
4:     Let $k := \arg\min_{j\in[n]} \hat{a}_j^\top \hat{y}$
5:     if $\hat{a}_k^\top \hat{y} < -\varepsilon$ then
6:         update $x := x - \frac{a_k^\top y}{\|a_k\|^2}\vec{e}_k$, $y := y - (\hat{a}_k^\top y)\hat{a}_k$
7:     else
8:         rescale $A := (I_m + \hat{y}\hat{y}^\top)A$, $y := 2y$
   return $\bar{x} = \Pi x$ as a feasible solution to $(K_{++})$
```

We use DV updates as the first-order method. We maintain a vector $x \in \mathbb{R}^n$, initialized as $x = \vec{e}$; the coordinates x_i never decrease during the algorithm. We also maintain $y = Ax$, and a main quantity of interest is the norm $\|y\|^2$. In each iteration of the algorithm, we check whether $\bar{x} = \Pi x$, the projection of x onto $\ker(A)$, is strictly positive. If this happens, then $\bar{x}$ is returned as a feasible solution to (K_{++}).

Every iteration performs either a DV update to x (line 6) or a rescaling of the matrix A (line 8). Because DV updates are performed only for $k \in [n]$ satisfying $\hat{a}_k^\top \hat{y} < -\varepsilon$, (4) ensures a substantial decrease in the norm, namely

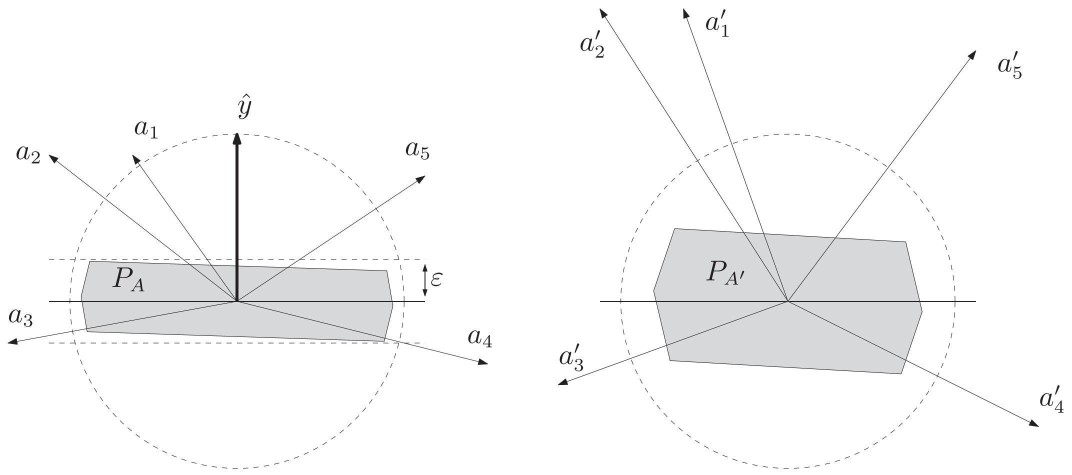
$$\|y'\| \leq \|y\|\sqrt{1-\varepsilon^2}. \tag{6}$$

By contrast, rescalings are performed if $\hat{a}_j^\top \hat{y} \geq -\varepsilon$ for all $j \in [n]$, which implies that $|\rho_A| < \varepsilon$. In this case, we show a substantial improvement in a volumetric potential. Let us define the polytope P_A as

$$P_A \overset{\text{def}}{=} \operatorname{conv}(\hat{A}) \cap (-\operatorname{conv}(\hat{A})). \tag{7}$$

The volume of P_A will be used as a proxy to $|\rho_A|$. Indeed, from Lemma 2, $|\rho_A|$ is the radius of the largest ball around the origin inscribed in $\operatorname{conv}(\hat{A})$, and this ball must be contained in P_A.

Figure 1. Effect of rescaling. The dashed circles represent the points of norm 1. The shaded areas are P_A and $P_{A'}$.



The condition $\hat{a}_j^\top \hat{y} \geq -\varepsilon$ for all $j \in [n]$ implies that P_A is contained in a "narrow strip" of width 2ε, namely $P_A \subseteq \{z \in \mathbb{R}^m : -\varepsilon \leq \hat{y}^\top z \leq \varepsilon\}$. If we replace A with the matrix $A' := (I + \hat{y}\hat{y}^\top)A$, then Lemma 5 implies that $\mathrm{vol}(P_{A'}) \geq {}^3\!/\!{}_2\, \mathrm{vol}(P_A)$. Geometrically, A' is obtained by applying to the columns of A the linear transformation that "stretches" them by a factor of 2 in the direction of $\hat{y}$ (see Figure 1).

Thus, at every iteration, we either have a substantial decrease in the length of the current y, or we have a constant factor increase in the volume of P_A.

The volume of P_A is bounded by the volume of the unit ball in $\mathbb{R}^m$ and initially contains a ball of radius $|\rho_A|$ around the origin. Consequently, the number of rescalings cannot exceed $m \log_{3/2} |\rho_A|^{-1}$.

The norm $\|y\|$ changes as follows. In every iteration where the DV update is applied, the norm of $\|y\|$ decreases by a factor of $\sqrt{1-\varepsilon^2}$ according to (6). At every rescaling, the norm of $\|y\|$ increases by a factor of 2. Lemma 6 shows that once $\|y\| < |\rho_A|$ for the an initial value of $|\rho_A|$, the algorithm terminates with $\bar{x} = \Pi x > 0$. We will prove the following running time bounds.

Theorem 1. *For any input matrix $A \in \mathbb{R}^{m\times n}$ such that $\rho_A < 0$ and $\|a_j\| = 1$ for all $j \in [n]$, Algorithm 1 finds a feasible solution of (K_{++}) in $O(m^2 \log n + m^3 \log |\rho_A|^{-1})$ DV updates. The number of arithmetic operations is $O(m^2 n \log n + (m^3 n + mn^2) \log |\rho_A|^{-1})$.*

Using Lemma 3, we obtain a running time bound in terms of bit complexity.

Corollary 1. *Let A be an $m \times n$ matrix with integer entries and encoding size L. If $\rho_A < 0$, then Algorithm 1 applied to $\hat{A}$ finds a feasible solution of (K_{++}) in $O\big((m^3 n + mn^2)L\big)$ arithmetic operations.*

2.1. Coordinate Descent with Finite Convergence

Before proceeding to the proof of Theorem 1, let us consider a modification of Algorithm 1 without any rescaling. That is, at every iteration we perform a DV update (even if $\hat{a}_j^\top \hat{y} \geq -\varepsilon$ for all $j \in [n]$) until $\Pi_A^K x > 0$. We claim that if $\rho_A < 0$, then the total number of DV steps is bounded by $O(\log(n/|\rho_A|)/\rho_A^2)$.

This is in contrast to the von Neumann algorithm, which does not have finite convergence for $\rho_A < 0$; this aspect is discussed by Li and Terlaky [21]. Dantzig [13] proposed a finitely converging variant of the von Neumann algorithm, but this involves running the algorithm $m+1$ times, as well as an explicit lower bound on the parameter $|\rho_A|$. Our algorithm does not incur a running time increase compared with the original variant and does not require such a bound.

Let us now verify the running time bound of our variant. Again, let us assume that $\|a_j\| = 1$ for all $j \in [n]$ for the input. It follows by (6) that the norm $\|y\|$ decreases by at least a factor of $\sqrt{1-\rho_A^2}$ in every DV update. Initially, $\|y\| \leq n$, and as shown in Lemma 6, the algorithm terminates with a solution $\Pi_A^K x > 0$ as soon as $\|y\| < |\rho_A|$. This yields the bound $O(\log(n/|\rho_A|)/\rho_A^2)$ on the number of DV steps.

2.2. Analysis

We will use the following technical lemma.

Lemma 4. *Let $X \in \mathbb{R}$ be a random variable supported on the interval $[-\varepsilon, \eta]$, where $0 \le \varepsilon \le \eta$, satisfying $\mathbb{E}[X] = \mu$. Then, for any $c \ge 0$, we have that $\mathbb{E}[\sqrt{1+cX^2}] \le \sqrt{1 + c\eta(\varepsilon + |\mu|)}$.*

Proof. Let $l(x) = \frac{\eta - x}{\eta+\varepsilon}\sqrt{1+c\varepsilon^2} + \frac{x+\varepsilon}{\eta+\varepsilon}\sqrt{1+c\eta^2}$ denote the unique affine interpolation of $\sqrt{1+cx^2}$ through the points $\{-\varepsilon, \eta\}$. By the convexity of $\sqrt{1+cx^2}$, we have that $l(x) \ge \sqrt{1+cx^2}$ for all $x \in [-\varepsilon, \eta]$. It follows that $\mathbb{E}[\sqrt{1+cX^2}] \le \mathbb{E}[l(X)] = l(\mathbb{E}[X]) = l(\mu)$, where the first equality holds because l is affine. From here, we get that

$$\begin{aligned} l(\mu) &= \frac{\eta-\mu}{\eta+\varepsilon}\sqrt{1+c\varepsilon^2} + \frac{\mu+\varepsilon}{\eta+\varepsilon}\sqrt{1+c\eta^2} \\ &\le \sqrt{1 + c\left(\frac{\eta-\mu}{\eta+\varepsilon}\varepsilon^2 + \frac{\mu+\varepsilon}{\eta+\varepsilon}\eta^2\right)} \quad \text{(by concavity of } \sqrt{x}) \\ &= \sqrt{1 + c(\eta\varepsilon + (\eta-\varepsilon)\mu)} \le \sqrt{1 + c\eta(\varepsilon + |\mu|)} \quad (\text{because } \varepsilon \le \eta), \end{aligned}$$

as needed. □

The crucial part of the analysis is to bound the volume increase of P_A at every rescaling iteration.

Lemma 5. *Let $A \in \mathbb{R}^{m\times n}$, and let $r = \mathrm{rk}(A)$. For some $0 < \varepsilon \le 1/(11r)$, let $v \in \mathbb{R}^m$, $\|v\| = 1$, such that $\hat{a}_j^\top v \ge -\varepsilon$ for all $j \in [n]$. Let $T = (I + vv^\top)$, and let $A' = TA$. Then*

a. *$TP_A \subseteq (1 + 3\varepsilon)P_{A'}$.*

b. *If $v \in \mathrm{im}(A)$, then $\mathrm{vol}_r(P_{A'}) \ge 3/2\, \mathrm{vol}_r(P_A)$.*

Proof. (a) The statement is trivial if $P_A = \emptyset$; thus, we assume that $P_A \ne \emptyset$. Consider an arbitrary point $z \in P_A$. By symmetry, it suffices to show that $Tz \in (1 + 3\varepsilon)\mathrm{conv}(\hat{A}')$. By definition, there exists $\lambda \in \mathbb{R}^n_+$ such that $\sum_{j=1}^n \lambda_j = 1$ and $z = \sum_{j=1}^n \lambda_j \hat{a}_j$. Note that

$$Tz = \sum_{j=1}^n \lambda_j T\hat{a}_j = \sum_{j=1}^n (\lambda_j \|T\hat{a}_j\|)\hat{a}'_j = \sum_{j=1}^n \lambda_j \sqrt{1 + 3(v^\top \hat{a}_j)^2}\, \hat{a}'_j.$$

Because $P_{A'} \ne \emptyset$, it follows that $0 \in \mathrm{conv}(\hat{A}')$; thus, it suffices to show that $\sum_{j=1}^n \lambda_j \sqrt{1 + 3(v^\top \hat{a}_j)^2} \le 1 + 3\varepsilon$. This expression is of the form $\mathbb{E}[\sqrt{1+3X^2}]$, where X is a random variable supported on $[-\varepsilon, 1]$, and $|\mathbb{E}[X]| = |\sum_{j=1}^n \lambda_j v^\top \hat{a}_j| = |v^\top z|$. Note that $|v^\top z| \le \varepsilon$ because both z and $-z$ are in P_A. Hence, by Lemma 4,

$$\sum_{j=1}^n \lambda_j \sqrt{1 + 3(v^\top \hat{a}_j)^2} \le \sqrt{1 + 3(2\varepsilon)} \le 1 + 3\varepsilon.$$

(b) Note that $\mathrm{vol}_r(TP_A) = 2\mathrm{vol}_r(P_A)$ as $\det(T) = 2$. Thus, we obtain $\mathrm{vol}_r(P_{A'}) \ge 2\mathrm{vol}_r(P_A)/(1+3\varepsilon)^r \ge 3/2\, \mathrm{vol}_r(P_A)$ because $(1+3\varepsilon)^r \le (1 + 3/(11m))^r \le e^{3/11} \le 4/3$. □

Lemma 6. *Let $A \in \mathbb{R}^{m\times n}$ with $\|a_j\| = 1$ for all $j \in [n]$. Given $x \in \mathbb{R}^n$ such that $x \ge \vec{e}$, if $\|Ax\| < |\rho_{(A,-A)}|$, then $\Pi_A^K x > 0$. In particular, if $\rho_A < 0$, then $\Pi_A^K x > 0$ whenever $\|Ax\| < |\rho_A|$.*

Proof. Let $\Pi := \Pi_A^K$, and define $\delta \overset{\text{def}}{=} \min_{j\in[n]} \|(AA^\top)^+ a_j\|^{-1}$. Observe that if $\|Ax\| < \delta$, then $\Pi x > 0$. Indeed, for $j \in [n]$, $(\Pi x)_j = x_j - a_j^\top (AA^\top)^+ y \ge 1 - \|(AA^\top)^+ a_j\|\, \|Ax\| > 1 - \delta^{-1}\delta = 0$, as required.

Thus, it suffices to show that $\delta \ge |\rho_{(A,-A)}|$. Let $k := \arg\max_{j\in[n]} \|(AA^\top)^+ a_j\|$, define $z := (AA^\top)^+ a_k$, and note that $\|z\| = 1/\delta$. Note that $\rho_{(A,-A)} < 0$; thus,

$$|\rho_{(A,-A)}| = -\rho_{(A,-A)} = \min_{y\in \mathrm{im}(A)\setminus\{0\}} \max_{j\in[n]} |a_j^\top \hat{y}| \le \max_{i\in[n]} |a_i^\top \hat{z}| = \delta \max_{j\in[n]} |\Pi_{jk}| \le \delta,$$

where the last inequality follows from the fact that $|\Pi_{ij}| \le 1$ for all $i, j \in [n]$. The last part of the statement follows from the fact that $|\rho_A| \le |\rho_{(A,-A)}|$. □

Lemma 7. *Let $v \in \mathbb{R}^m$, $\|v\| = 1$. For any $y, \bar{y} \in \mathbb{R}^m$ such that $y = (I + vv^\top)\bar{y}$, we have $\|\bar{y}\| \le \|y\|$.*

Proof. We have $\|y\|^2 = \|\bar{y} + (v^\top\bar{y})v\|^2 = \|\bar{y}\|^2 + 2(v^\top\bar{y})^2 + (v^\top\bar{y})^2\|v\|^2 \ge \|\bar{y}\|^2$. □

Proof of Theorem 1. We use $\bar{A}$ for the input matrix and A for the current matrix during the algorithm so that, after k rescalings, $A = (I + v_1v_1^\top)\cdots(I + v_kv_k^\top)\bar{A}$ for some vectors $v_1, \dots, v_k \in \mathrm{im}(A)$ with $\|v_i\| = 1$ for $i \in [n]$.

Let $\rho := |\rho_{\bar{A}}|$ and $\Pi := \Pi_{\bar{A}}^K$. Note that $\ker(A) = \ker(\bar{A})$, and hence, $\Pi_A^K = \Pi$ throughout the algorithm. The matrix Π needs to be computed only once, requiring $O(n^2m)$ arithmetic operations, which is clearly dominated by the stated running time of the algorithm.

Let x and $y = Ax$ be the vectors computed in every iteration, and define $\bar{y} := \bar{A}x$ and $\bar{x} = \Pi x$. Lemma 7 implies that $\|\bar{y}\| \le \|y\|$; thus, it follows from Lemma 6 that $\bar{x} > 0$ whenever $\|y\| < \rho$. This shows that the algorithm terminates with the $\bar{x}$ to (K_{++}) as soon as $\|y\| < \rho$.

As discussed previously, Lemma 5 implies that the number K of rescalings cannot exceed $m\log_{3/2}|\rho|^{-1}$. At every rescaling, $\|y\|$ increases by a factor of 2. In every iteration where the DV update is applied, $\|y\|$ decreases by a factor of $\sqrt{1-\varepsilon^2}$ according to (6). Initially, $y = \bar{A}\vec{e}$; therefore, $\|y\| \le n$ because all columns of A have unit norm. This shows that the number of DV iterations is bounded by κ, where κ is the smallest integer such that $n^2(1-\varepsilon^2)^\kappa 4^K < \rho^2$. Taking the logarithm on both sides and using the fact that $\log(1-\varepsilon^2) < -\varepsilon^2$, it follows that $\kappa \in O(m^2(\log n + K + \log|\rho|^{-1})) = O(m^2\log n + m^3\log|\rho|^{-1})$.

We can implement every DV update in $O(n)$ time, at a cost of an $O(n^2)$ time preprocessing at every rescaling, as explained next. After every rescaling, we compute the matrix $F := A^\top A$ and the norms of the columns of A. Computing the norms requires $O(nm)$ processing time. The matrix F is updated as $F := A^\top(I+\hat{y}\hat{y}^\top)^2A = AA^\top + 3(A^\top\hat{y}\hat{y}^\top A) = F + 3zz^\top/\|y\|^2$, which requires $O(n^2)$ processing time.

Furthermore, at every DV update, we maintain the vectors $z = A^\top y$ and $\bar{x} = \Pi x$. Using the vector z, we can compute $\arg\min_{j\in[n]} \hat{a}_j^\top\hat{y} = \arg\min_{j\in[n]} z_j/\|a_j\|$ in time $O(n)$ at any DV update. We also need to recompute y, z, and $\bar{x}$. Using $F = [f_1, \dots, f_n]$, these can be obtained as $y := y - (\hat{a}_k^\top y)\hat{a}_k$, $z := z - f_k(\hat{a}_k^\top y)/\|a_k\|$, and $\bar{x} := \bar{x} - \Pi_k(\hat{a}_k^\top y)/\|a_k\|$, where Π_k denotes the kth column of Π. These updates altogether take $O(n)$ processing time.

Therefore, the number of arithmetic operations is $O(n)$ times the number of DV updates plus $O(n^2)$ times the number of rescalings. The overall running time estimate follows. □

3. The Full Support Image Algorithm

The image algorithm maintains a positive definite matrix Q, initialized as $Q = I_m$. We use the von Neumann algorithm (Algorithm 2) as the first-order method, with the scalar product $\langle\cdot,\cdot\rangle_Q$. Within $O(m^2)$ iterations, the von Neumann algorithm obtains a vector $y \in \mathrm{conv}(a_1/\|a_1\|_Q, \dots, a_n/\|a_n\|_Q)$ with $\|y\|_Q \le \varepsilon$. Then we update the matrix Q using the coefficients of the convex combination.

Algorithm 2 is same as von Neumann's algorithm as described by Dantzig [14], with the standard scalar product replaced by $\langle\cdot,\cdot\rangle_Q$ for a matrix $Q \in \mathbb{S}_{++}^m$ and using the normalized columns $a_i/\|a_i\|_Q$. We remark that running the algorithm with $\langle\cdot,\cdot\rangle_Q$ is the same as running it for the standard scalar product for the unit vectors $Q^{1/2}a_i/\|Q^{1/2}a_i\|_2$.

Algorithm 2 (The von Neumann Algorithm)

```
Input: A matrix $A \in \mathbb{R}^{m\times n}$, a positive definite matrix $Q \in \mathbb{R}^{m\times m}$, and an $\varepsilon > 0$
Output: Vectors $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ such that $y = \sum_{i=1}^n x_i a_i/\|a_i\|_Q$, $\vec{e}^\top x = 1$, $x \ge 0$, and either $A^\top Qy > 0$ or $\|y\|_Q \le \varepsilon$
1: Set $x := \vec{e}_1$, $y := a_1/\|a_1\|_Q$
2: while $\|y\|_Q > \varepsilon$ do
3:    if $\langle a_i, y\rangle_Q > 0$ for all $i \in [n]$ then return $x$ and $y$ satisfying $A^\top Qy > 0$
4:       Terminate
5:    else Select $k \in [n]$ such that $\langle a_k, y\rangle_Q \le 0$
6:       Let $\lambda := \dfrac{\langle y - a_k/\|a_k\|_Q, y\rangle_Q}{\|y - a_k/\|a_k\|_Q\|_Q^2}$
7:       update $x := (1-\lambda)x + \lambda\vec{e}_k$,   $y := (1-\lambda)y + \lambda\dfrac{a_k}{\|a_k\|_Q}$
8: return the vectors $(x, y)$
```

Lemma 8. *For a given $\varepsilon > 0$, the von Neumann algorithm terminates in at most $\lceil 1/\varepsilon^2\rceil$ updates. Each iteration requires $O(n)$ arithmetic operations, provided that the matrix $A^\top QA$ has been precomputed*

Proof. The $\lceil 1/\varepsilon^2\rceil$ bound on the number of iterations is due to Dantzig [14]. If we maintain the vector $z := A^\top Qy$, then checking whether $\langle a_i, y\rangle_Q > 0$ for all $i \in [n]$ amounts to checking whether $z > 0$, which can be performed in time

$O(n)$. Recomputing $x' := (1-\lambda)x + \lambda\vec{e}_k$ and $y' := (1-\lambda)y + \lambda a_k/\|a_k\|_Q$ requires time $O(n)$. Recomputing $z' := A^\top Qy$ requires computing $z' := (1-\lambda)z + \lambda A^\top Q a_k/\|a_k\|_Q$, which can also be done in time $O(n)$ provided that $A^\top QA$ has been precomputed. □

Algorithm 3 shows the full support image algorithm. We set the same $\varepsilon = \frac{1}{11m}$ as in the kernel algorithm. Without loss of generality, we can assume that the matrix A has full row rank; that is, $\mathrm{im}(A) = \mathbb{R}^m$.

Algorithm 3 (Full Support Image Algorithm)
Input: A matrix $A \in \mathbb{R}^{m\times n}$ such that $\mathrm{rk}(A) = m$ and (I_{++}) is feasible
Output: A feasible solution to (I_{++})

```
1: Set $Q := I_m$, $R := I_m$; call von Neumann$(A, Q, \varepsilon)$ to obtain $(x, y)$
2: while $A^\top Qy \not> 0$ do
3:     rescale
           $R := \frac{1}{1+\varepsilon}\left(R + \sum_{i=1}^n \frac{x_i}{\|a_i\|_Q^2} a_i a_i^\top\right); \quad Q := R^{-1}.$
4:     Call von Neumann$(A, Q, \varepsilon)$ to obtain $(x, y)$
   return The feasible solution $\bar{y} := Qy$ to $(I_{++})$
```

Theorem 2. *For any input matrix $A \in \mathbb{R}^{m\times n}$ such that $\mathrm{rk}(A) = m$ and (I_{++}) is feasible, Algorithm 3 finds a feasible solution to (I_{++}) by performing $O\left(m^3 \log \rho_A^{-1}\right)$ von Neumann iterations. The total number of arithmetic operations is $O\left(m^2 n^2 \log \rho_A^{-1}\right)$.*

This will be proved in Section 3.1. Using Lemma 3, we obtain the running time in terms of the encoding length L.

Corollary 2. *Let $A \in \mathbb{Z}^{m\times n}$ be an integer matrix of encoding size L. If $\mathrm{rk}(A) = m$ and (I_{++}) is feasible, then Algorithm 3 finds a feasible solution of (I_{++}) in $O\left(m^2 n^2 L\right)$ arithmetic operations.*

In this framework, the only important property of the von Neumann algorithm is that it delivers a vector $y \in \mathrm{conv}(a_1/\|a_1\|_Q, \ldots, a_n/\|a_n\|_Q)$ with $\|y\| \le O(1/m)$ in time polynomial in m and n. This can be also achieved using other first-order methods, such as the perceptron algorithm, the DV updates, or Wolfe's [36] nearest-point algorithm. The best running times can be obtained using the smoothed perceptron algorithm of Soheili and Peña [33] or the Mirror Prox for feasibility problems by Yu et al. [38].

Theorem 3. *For any input matrix $A \in \mathbb{R}^{m\times n}$ such that $\mathrm{rk}(A) = m$ and (I_{++}) is feasible, Algorithm 3 with the smoothed perceptron of Soheili and Peña [33] or the Mirror Prox method of Yu et al. [38] finds a feasible solution to (I_{++}) by performing $O\left(m^2\sqrt{\log n} \cdot \log \rho_A^{-1}\right)$ iterations. The number of arithmetic operations is $O\left(m^3 n\sqrt{\log n} \cdot \log \rho_A^{-1}\right)$. If $A \in \mathbb{Z}^{m\times n}$ is an integer with encoding length L, then the running time is $O\left(m^3 n\sqrt{\log n} \cdot L\right)$.*

The main difference between Algorithm 3 and the algorithms by Betke [4] and Peña and Soheili [27] is the use of a multirank rescaling, as opposed to rank 1 updates. The multirank rescaling allows for a factor of n improvement in the overall number of iterations. Although we use a similar volumetric potential, the multirank update guarantees a constant factor decrease in potential (Lemma 11) whenever in the algorithm $\|y\|_Q \in O(1/m)$, whereas the rank 1 update provides the same guarantee only when $\|y\|_Q \in O(1/(m\sqrt{n}))$.

3.1. Analysis

It is easy to see that the matrix R remains positive semidefinite throughout the algorithm and admits the following decomposition.

Lemma 9. *At any stage of the algorithm, we can write the matrix R in the form*

$$R = \alpha I_m + \sum_{i=1}^n \gamma_i \hat{a}_i \hat{a}_i^\top,$$

where $\alpha = 1/(1+\varepsilon)^t$ for the total number of rescalings t performed thus far, and $\gamma_i \ge 0$. The trace is $\mathrm{tr}(R) = \alpha m + \sum_{i=1}^n \gamma_i$.

Recall that we denote by $\Sigma_A = \{y \in \mathbb{R}^m : A^\top y \ge 0\}$ the image cone. Let us define the set

$$F_A = \Sigma_A \cap \mathbb{B}^m. \tag{8}$$

The ellipsoid $E(R) = \{z \in \mathbb{R}^m : \|z\|_R^2 \le 1\}$ plays a key role in the analysis because of the following properties.

Lemma 10. *Throughout Algorithm 3, $F_A \subseteq E(R)$ holds.*

Proof. The proof is by induction on the number of rescalings. At initialization, $F_A \subseteq E(I_m) = \mathbb{B}^m$ is trivial. Assume that $F_A \subseteq E(R)$, and we rescale R to R'. We show that $F_A \subseteq E(R')$. Consider an arbitrary point $z \in F_A$; then $a_i^\top z \ge 0$ for all $i \in [n]$ and, by the induction hypothesis, $\|z\|_R^2 \le 1$ because $z \in E(R)$.

For the vector x returned by the von Neumann algorithm, the algorithm sets

$$R' = \frac{1}{1+\varepsilon}\left(R + \sum_{i=1}^n \frac{x_i}{\|a_i\|_Q^2} a_i a_i^\top\right).$$

Recall that in the algorithm, the vector $y = \sum_{i=1}^n x_i \frac{a_i}{\|a_i\|_Q}$ satisfies $\|y\|_Q \le \varepsilon$. By the Cauchy–Schwartz inequality, we have $y^\top z = y^\top Q^{1/2} Q^{-1/2} z \le \|y\|_Q \|z\|_R \le \varepsilon$ and, similarly, $a_i^\top z \le \|a_i\|_Q \|z\|_R \le \|a_i\|_Q$ for every $i \in [m]$. We then have

$$\|z\|_{R'}^2 = \frac{1}{1+\varepsilon} z^\top \left(R + \sum_{i=1}^n \frac{x_i}{\|a_i\|_Q^2} a_i a_i^\top\right) z = \frac{1}{1+\varepsilon}\left(\|z\|_R^2 + \sum_{i=1}^n x_i \left(\frac{a_i^\top z}{\|a_i\|_Q}\right)^2\right) \le \frac{1}{1+\varepsilon}\left(1 + \sum_{i=1}^n x_i \frac{a_i^\top z}{\|a_i\|_Q}\right) = \frac{1 + y^\top z}{1+\varepsilon} \le 1,$$

where the first inequality follows from the fact that $\|z\|_R \le 1$, $x \ge 0$, and $0 \le a_i^\top z \le \|a_i\|_Q$ for all $i \in [n]$, whereas the second follows from $y^\top z \le \varepsilon$. Consequently, $z \in E(R')$, completing the proof. □

Lemma 11. *At every rescaling,* $\det(R)$ *increases by a factor of at least* 16/9.

Proof. Let R and R' denote the matrix before and after the rescaling. Let $X = \sum_{i=1}^n x_i a_i a_i^\top / \|a_i\|_Q^2$; hence, $R' = (R+X)/(1+\varepsilon)$. The ratio of the two determinants is

$$\frac{\det(R')}{\det(R)} = \frac{\det(R+X)}{(1+\varepsilon)^m \det(R)} = \frac{\det\left(I_m + R^{-1/2} X R^{-1/2}\right)}{(1+\varepsilon)^m}.$$

Now $R^{-1/2} = Q^{1/2}$, and $Q^{1/2} X Q^{1/2}$ is a positive semidefinite matrix. The determinant can be lower bounded using Lemma 1(a) and the linearity of the trace:

$$\frac{\det(R')}{\det(R)} \ge \frac{1 + \operatorname{tr}\left(Q^{1/2} X Q^{1/2}\right)}{(1+\varepsilon)^m} = \left(1 + \sum_{i=1}^n \frac{x_i}{\|a_i\|_Q^2} \operatorname{tr}\left(Q^{1/2} a_i a_i^\top Q^{1/2}\right)\right) \Big/ (1+\varepsilon)^m.$$

Finally, $\operatorname{tr}(Q^{1/2} a_i a_i^\top Q^{1/2}) = \operatorname{tr}(a_i^\top Q a_i) = \|a_i\|_Q^2$. Therefore, we conclude that

$$\frac{\det(R')}{\det(R)} \ge \frac{1 + \sum_{i=1}^n x_i}{(1+\varepsilon)^m} = \frac{2}{(1+\varepsilon)^m}.$$

The claim follows using that $\varepsilon = \frac{1}{11m}$. □

We now present the proofs of Theorems 2 and 3 based on these lemmas.

Proof of Theorem 2. By Lemma 2, Σ_A contains a ball B of radius ρ_A centered on the surface of the unit sphere. Consequently, $B \subseteq \Sigma_A \cap (1+\rho_A)\mathbb{B}^m = (1+\rho_A)F_A$. In particular, F_A contains a ball of radius $\rho_A/(1+\rho_A)$; therefore, $\operatorname{vol}(F_A) \ge (\rho_A/(1+\rho_A))^m \operatorname{vol}(\mathbb{B}^m) \ge (\rho_A/2)^m \operatorname{vol}(\mathbb{B}^m)$. By contrast, because $\operatorname{vol}(E(R)) = \det(R)^{-1/2} \operatorname{vol}(\mathbb{B}^m)$, Lemma 11 implies that $\operatorname{vol}(E(R))$ decreases at least by a factor of 2/3 at every rescaling. Lemma 10 ensures that $\operatorname{vol}(E(R)) \ge \operatorname{vol}(F_A)$. Consequently, the total number of rescalings during the entire course of the algorithm provides the bound $O(m \log \rho_A^{-1})$.

By Lemma 8, the von Neumann algorithm performs $O(m^2)$ iterations between two consecutive rescalings, where each von Neumann iteration can be implemented in time $O(n)$ assuming that we compute the matrix $A^\top QA$ at the beginning and after every rescaling. Thus, the total number of arithmetic operations required by the von Neumann iterations between two rescalings is $O(m^2 n)$. To compute $A^\top QA$, provided that we have computed Q, requires time $O(n^2 m)$. Updating the matrix R requires time $O(m^2 n)$ because we need time $O(m^2)$ to compute each of the n terms $x_i a_i a_i^\top / \|a_i\|_Q^2$, $i \in [n]$. The inverse Q of R can be computed in time $O(m^3)$. Hence, the overall number of arithmetic operations needed between two rescalings is $O(n^2 m)$. This gives an overall complexity of $O(n^2 m^2 \log \rho_A^{-1})$ arithmetic operations. □

We note that the higher running time compared with Algorithm 1 is due to the time required to update $A^\top QA$. This has to be recomputed from scratch, whereas the corresponding update to $A^\top A$ in the kernel case was done in $O(n^2)$ because a rank 1 rescaling was used.

Proof of Theorem 3. Both the smoothed perceptron of Soheili and Peña [33] and the Mirror Prox method of Yu et al. [38] terminate in $O(\sqrt{\log n}/\varepsilon)$ iterations with output $x \in \mathbb{R}^n_+$ and $y \in \mathbb{R}^m$ such that $\|x\|_1 = 1$, $y = \sum_{i=1}^n x_i a_i / \|a_i\|_Q$, and either $A^\top Qy > 0$ or $\|y\|_Q \le \varepsilon$. For both methods, each iteration requires $O(mn)$ arithmetic operations. As before, R and Q can be recomputed in time $O(m^2 n)$. Thus, the overall number of operations required between rescalings is $O(m^2 n\sqrt{\log n})$. As before, the total number of rescalings is $O(m \log \rho_A^{-1})$. These together give the claimed bound. □

3.2. Oracle Model for Strict Conic Feasibility

Observe that Algorithm 3 does not require explicit knowledge of the matrix A. In particular, Algorithm 3 can be easily adapted to an oracle model. Here the purpose is to find a point in the interior of a full-dimensional cone defined as $\Sigma = \{y \in \mathbb{R}^m : a_i^\top y \ge 0 \; \forall a_i \in I\}$, where I is a set (possibly infinite) indexing vectors $a_i \in \mathbb{R}^m$, $i \in I$. We assume that we have access to a *strict separation oracle* (SO), where for each $v \in \mathbb{R}^m$ the call SO(v) returns "YES" if $v \in \text{int}(\Sigma)$ (i.e., if $a_i^\top v > 0$ for all $i \in I$), or it returns a_k for some $k \in I$ such that $a_k^\top v \le 0$.

Below we present Algorithm 5 to determine a point in the interior of Σ. The algorithm is nearly identical to Algorithm 3, and it uses an oracle version of the von Neumann algorithm (Algorithm 4). The running time is expressed in terms of the Goffin measure ρ_Σ of a full-dimensional cone Σ, which is the radius of the largest ball contained in Σ centered on the surface of the unit sphere:

$$\rho_\Sigma \overset{\text{def}}{=} \sup\{r : \mathbb{B}^m(p, r) \subseteq \Sigma \; \exists p \in \mathbb{R}^m \text{ s.t. } \|p\| = 1\}.$$

Algorithm 4 (Oracle von Neumann Algorithm)
Input: A positive definite matrix $Q \in \mathbb{R}^{m\times m}$ and an $\varepsilon > 0$
Output: Vectors $\{a_i : i \in N\}$ for $N \subseteq I$, $x \in \mathbb{R}^N_+$, y such that $y = \sum_{i\in N} x_i a_i/\|a_i\|_Q$, $\sum_{i\in N} x_i = 1$, and either $Qy \in \text{int}(\Sigma)$ or $\|y\|_Q \le \varepsilon$

```
1: Call SO(0) to obtain $a_k$. Set $N := \{k\}$, $x_k := 1$, $y := a_k/\|a_k\|_Q$
2: while $\|y\|_Q > \varepsilon$ do
3:    if SO($Qy$) returns "YES," then return $\{a_i : i \in N\}$, $x$, $y$
4:       Terminate
5:    else let $a_k$, $k \in I$, be the output of SO($Qy$)
6:       Let $\lambda := \frac{\langle y - a_k/\|a_k\|_Q, y\rangle_Q}{\|y - a_k/\|a_k\|_Q\|_Q^2}$
7:       update $x_i := (1-\lambda)x_i$ for all $i \in N\setminus\{k\}$, $x_k := (1-\lambda)x_k + \lambda$.
8:       $y := (1-\lambda)y + \lambda \frac{a_k}{\|a_k\|_Q}$, $N := N \cup \{k\}$
9: return $\{a_i : i \in N\}$, $x$, $y$
```

Note that in line 7 of Algorithm 4, for notational convenience we consider x_k to be 0 if $k \notin N$.

Algorithm 5 (Strict Conic Feasibility Algorithm)
Output: A point in the interior of a full-dimensional cone Σ given via a separation oracle SO

```
1: Set $Q := I_m$, $R := I_m$; call ORACLE VON NEUMANN($Q, \varepsilon$) to obtain $\{a_i : i \in N\}$, $x \in \mathbb{R}^N$, $y \in \mathbb{R}^m$
2: while $Qy \notin \text{int}(\Sigma)$ do
3:    rescale
          $$R := \tfrac{1}{1+\varepsilon}\left(R + \sum_{i\in N} \frac{x_i}{\|a_i\|_Q^2} a_i a_i^\top\right); \quad Q := R^{-1}.$$
4:    Call ORACLE VON NEUMANN($Q, \varepsilon$) to obtain $\{a_i : i \in N\}$, $x$, $y$
5: return $\bar{y} := Qy$
```

Theorem 4. *For any full-dimensional cone Σ expressed by a separation oracle, Algorithm 5 finds a point in* int(Σ) *by performing $O(m^3 \log \rho_\Sigma^{-1})$ von Neumann iterations. The total number of oracle calls is $O(m^3 \log \rho_\Sigma^{-1})$, whereas the total number of arithmetic operations is $O(m^5 \log \rho_\Sigma^{-1})$.*

Proof. The analysis is nearly identical to that of Theorem 2. As before, Algorithm 4 terminates in at most $\lceil 1/\varepsilon \rceil \in O(m^2)$ iterations, and the number of rescalings in Algorithm 5 is $O\big(m \log \rho_\Sigma^{-1}\big)$. Thus, we perform $O\big(m^3 \log \rho_\Sigma^{-1}\big)$ von Neumann iterations, each requiring one oracle call. For the number of arithmetic operations, we observe first that the set N computed by Algorithm 4 has size at most $\lceil 1/\varepsilon \rceil \in O(m^2)$ because $|N|$ increases by at most one at every iteration. In every von Neumann iteration, recomputing x requires $O(|N|) = O(m^2)$ arithmetic operations, recomputing y requires $O(m)$ operations, and recomputing Qy requires $O(m^2)$ operations; thus, overall, these computations require $O\big(m^5 \log \rho_\Sigma^{-1}\big)$ arithmetic operations. Recomputing R during rescalings requires $O(m^2|N|) = O(m^4)$ operations, whereas recomputing $Q = R^{-1}$ requires $O(m^3)$ operations, for a total of $O\big(m^5 \log \rho_\Sigma^{-1}\big)$ arithmetic operations over all rescalings. □

4. Maximum Support Algorithms

In the case $\rho_A = 0$, the volumetric arguments of the previous sections fail: both sets P_A and F_A are lower dimensional and therefore have volume 0. In what follows, we show how both algorithms naturally extend to this scenario.

Given any linear subspace H, we denote by $\text{supp}(H_+)$ the *maximum support* of H_+, that is, the unique inclusion-wise maximal element of the family $\{\text{supp}(x) : x \in H_+\}$. Note that because H_+ is closed under summation, it follows that $\text{supp}(H_+) = \{i \in [n] : \exists x \in H_+\ x_i > 0\}$. We let

$$S_A^* \overset{\text{def}}{=} \text{supp}\big(\ker(A)_+\big), \quad T_A^* \overset{\text{def}}{=} \text{supp}\big(\text{im}(A^\top)_+\big). \tag{9}$$

When clear from the context, we will use the simpler notation S^* and T^*. Because $\ker(A)$ and $\text{im}(A^\top)$ are orthogonal to each other, it is immediate that $S^* \cap T^* = \emptyset$. Furthermore, the strong duality theorem implies that $S^* \cup T^* = [n]$.

The maximum support kernel algorithm (Section 4.1) finds a solution x to (K) with $\text{supp}(x) = S^*$, and the maximum support image algorithm (Section 4.2) finds a solution y to (I) with $\text{supp}(A^\top y) = T^*$. In this section, we show that these algorithms can be directly obtained using the full support algorithms by repeatedly removing vectors from the support based on their lengths after a sequence of rescalings. With this direct implementation, however, the maximum support algorithm runs the corresponding full support algorithm n times in the kernel case and m times in the image case, leading to an increase in running time.

With small modifications, both maximum support algorithms can be implemented in essentially the same asymptotic running time as their full support counterparts. We defer these variants to Appendix A because the amortized analyses are somewhat technical. Still, they offer some interesting insights for degenerate linear programs. In particular, they show how to bound the degradation of an ellipsoidal outer approximation of the feasible region when moving to a lower-dimensional space, which we believe to be of independent interest.

We will need to argue about lower-dimensional ellipsoids and their volumes. Let $Q \in \mathbb{S}_{++}^d$. For a linear subspace $H \subseteq \mathbb{R}^d$, we let $E_H(Q) \overset{\text{def}}{=} E(Q) \cap H$. Furthermore, we define the projected determinant of Q on H as

$$\det_H(Q) \overset{\text{def}}{=} \det\big(W^\top Q W\big),$$

where W is any matrix whose columns form an orthonormal basis of H. Note that the definition is independent of the choice of the basis W. Indeed, if H has dimension r, then

$$\text{vol}_r(E_H(Q)) = \frac{\nu_r}{\sqrt{\det_H(Q)}}. \tag{10}$$

4.1. The Maximum Support Kernel Algorithm

We start with an easy observation; the proof is deferred to Appendix B.

Lemma 12. *Let $A \in \mathbb{R}^{m\times n}$ and $S^* = S_A^*$. Then $\text{span}(P_A) = \text{im}(A_{S^*})$, and $P_A = P_{A_{S^*}}$.*

In this section, we observe that if Algorithm 1 is applied to a matrix A with $\rho_A \geq 0$ (i.e., (K_{++}) is infeasible), then, after a certain number of iterations, based on the encoding size of A, we can establish a column of A that cannot be contained in S_A^*. This is based on the observation contained in the next lemma that columns in S_A^* need to remain "short" throughout the execution of Algorithm 1.

Lemma 13. *Let $A \in \mathbb{R}^{m\times n}$ such that $\|a_i\| = 1$ for all $i \in [n]$. Let $H = \mathrm{im}(A)$. After t rescalings in Algorithm 1 with A as input, let A' be the current matrix. Let $M \in \mathbb{R}^{m\times m}$ be the matrix obtained by combining all t rescalings so that $A' = MA$, and define $Q = (1+3\varepsilon)^{-2t}M^{\mathsf{T}}M$. The following hold:*

a. $P_A \subseteq E(Q)$.
b. $\|a_i\|_Q \le |\rho_{A_{S^*}}|^{-1}$ *for every* $i \in S^*$.
c. *If* $\mu := \max_{i\in[n]} \|a_i\|_Q$, *then* $(\mu^{-1}\cdot|\rho_{(A,-A)}|)\mathbb{B}^m \cap H \subseteq E_H(Q)$.
d. *At every rescaling, the relative volume of* $E_H(Q)$ *decreases by a factor of at least* $2/3$.

Proof. (a) At initialization, $Q = I_m$, so $P_A \subseteq E(Q) = \mathbb{B}^m$. After t rescalings, by Lemma 5 applied t times, $MP_A \subseteq (1+3\varepsilon)^t P_{A'}$. Recall that, by definition, $P_{A'} \subseteq \mathbb{B}^m$; thus, $P_A \subseteq (1+3\varepsilon)^t M^{-1}\mathbb{B}^m = E(Q)$.

(b) Lemma 12 shows that $P_A = P_{A_{S^*}}$. Hence, by Lemma 2 and the fact that $\|a_i\|_2 = 1$ for all $i \in [n]$, it follows that $|\rho_{A_{S^*}}|a_i \in P_A$ for all $i \in S^*$. From (a), $|\rho_{A_{S^*}}|a_i \in E(Q)$, which implies that $|\rho_{A_{S^*}}|\|a_i\|_Q \le 1$.

(c) By definition, $\mu^{-1}a_j \in E(Q)$; therefore, $E_H(Q)$ contains $\mu^{-1}\cdot \mathrm{conv}(A,-A)$. Note that $\rho_{(A,-A)} < 0$; hence, Lemma 2 implies that $|\rho_{(A,-A)}|\mathbb{B}^m \cap H \subseteq \mathrm{conv}(A,-A)$. This implies that $(\mu^{-1}\cdot\rho_{(A,-A)})\mathbb{B}^m \cap H \subseteq E_H(Q)$ as needed.

(d) Let $r = \mathrm{rk}(A)$. Rescaling corresponds to replacing M by $M' = (I_m + \hat{y}\hat{y}^\top)M$, where $y = A'x$ is the current point computed by the algorithm. Accordingly, matrix Q is replaced by $Q' = (1+3\varepsilon)^{-2}M^{\mathsf{T}}M$. Because $y \in H$, the function $z \to (I_m + \hat{y}\hat{y}^\top)z$ is the identity over $H^\perp$, and it is an automorphism over H. This implies that $E_H(Q') = (1+3\varepsilon)(I_m + \hat{y}\hat{y}^\top)^{-1}E_H(Q)$ and that $\mathrm{vol}_r(E_H(Q')) = (1+3\varepsilon)^r \det(I_m + \hat{y}\hat{y}^\top)^{-1}\mathrm{vol}_r(E_H(Q))$. The statement follows from the fact that $(1+3\varepsilon)^r \le 4/3$ and $\det(I_m + \hat{y}\hat{y}^\top) = 2$. □

4.1.1. Basic Maximum Support Kernel Algorithm. Based on the preceding lemma, we can immediately describe an algorithm for the maximum support kernel problem for a matrix $A \in \mathbb{Z}^{m\times n}$ of encoding size L. Let $\theta := \theta_A$ as defined in (2), recalling that $\theta \ge 2^{-4L}$. Let $S^* := S^*_A$. Observe that by Lemma 3 we have $|\rho_{A_{S^*}}| \ge \theta$ if $S^* \ne \emptyset$ and $|\rho_{(A_S,-A_S)}| \ge \theta$ for any $\emptyset \ne S \subseteq [n]$ (indeed, note that $\Delta_A \ge \Delta_{A_S} = \Delta_{(A_S,-A_S)}$ by definition).

We start with initial guess $S = [n]$ for the support S^*. To get a maximum support solution, we will iteratively run a slightly modified version of Algorithm 1 on the matrix $\hat{A}_S$, which will return either a full support solution $\hat{A}_S x = 0$, $x > 0$, or an index $k \in S\setminus S^*$. In the former case, we return x with zeros on the components in $[n]\setminus S$ as a maximum support solution. In the latter case, we replace $S := S\setminus\{k\}$ and rerun the modified Algorithm 1 on $\hat{A}_S$. We continue this process until either $S = \emptyset$ or a solution is found.

For the modified Algorithm 1 on $\hat{A}_S$, the only change is a step that recognizes when an index in S is not in the support S^*. For this purpose, we maintain the vector lengths $\|\hat{a}_i\|_Q$ as in Lemma 13 applied to $\hat{A}_S$. After each rescaling, which updates Q, we simply check whether there is an index $k \in S$ such that $\|\hat{a}_k\|_Q > \theta^{-1}$ (here $\hat{a}_k$ refers the normalized column of the original matrix A), and if so, we return it as an index not in S^*. Note that this assertion is justified by Lemma 13(b). Let us note that if A' is the current matrix in Algorithm 1 on $\hat{A}_S$ after t rescalings, then $\|\hat{a}_i\|_Q = \|a'_i\|/(1+3\varepsilon)^t$. Therefore, we do not need to maintain the matrix Q explicitly.

We now bound the running time of the modified Algorithm 1 at every call. Let $S \supseteq S^*$ be the current support, $H = \mathrm{im}(\hat{A}_S)$, and $r := \mathrm{rk}(A_S) \le m$. Note that as long as we have not identified a column to remove, which would end the current call on $\hat{A}_S$, by part (c) of Lemma 13 we have that $\theta^2\mathbb{B}^m \cap H \subseteq E_H(Q)$; hence, $\mathrm{vol}_r(E_H(Q)) \ge \theta^{2r}\nu_r$. Because initially $\mathrm{vol}(E_H(Q))_r = \nu_r$, by part (d) of Lemma 13 we conclude that the number K of rescalings is bounded by $O(\log(\theta^{2r}))$; hence, $K \in O(mL)$ (because $r \le m$). By Lemma 6, the call to Algorithm 1 terminates as soon as the current vector y has norm less than $\theta \le |\rho_{(A_S,-A_S)}|$; hence, the number of DV iterations can be bounded exactly as in the proof of Theorem 1 by $O(m^2(n + K + \log(\theta^{-1}))) = O(m^3L)$. The proof of Theorem 1 also shows that each DV update can be performed in time $O(n)$ and that each rescaling can be computed in time $O(n^2)$. Hence, each call to Algorithm 1 requires $O((m^3n + mn^2)L)$ arithmetic operations, so the overall number of operations to compute a maximum support solution is $O((m^3n^2 + mn^3)L)$.

4.2. The Maximum Support Image Algorithm

In this section, we show that if Algorithm 3 is applied to a matrix A with $\rho_A \le 0$ (i.e., (I_{++}) is infeasible), then, after a certain number of iterations, based on the encoding size of A, we can pinpoint an index $k \in [n]\setminus T^*_A$.

The following will be a key concept in the analysis. Given a convex set $X \subset \mathbb{R}^d$ and a vector $a \in \mathbb{R}^d$, we define the *width of X along a* as

$$\mathrm{width}_X(a) \overset{\mathrm{def}}{=} \max\{a^{\mathsf{T}}z : z \in X\}. \qquad (11)$$

Lemma 14. *Given $R \in \mathbb{S}^d_{++}$, let $E := E(R)$. For any $a \in \mathbb{R}^d$,* $\mathrm{width}_E(a) = \|a\|_{R^{-1}}$.

Proof. For every $z \in E$, $a^\top z = a^\top R^{-1/2} R^{1/2} z \le \|a\|_{R^{-1}} \|z\|_R \le \|a\|_{R^{-1}}$, where the first inequality follows from the Cauchy–Schwarz inequality and the second from $z \in E$. By contrast, if we define $z = R^{-1}a/\|a\|_{R^{-1}}$, it follows that $z \in E$ and $a^\top z = \|a\|_{R^{-1}}$. □

Let us introduce

$$\omega_A \overset{\mathrm{def}}{=} \min_{i \in T_A^*} \mathrm{width}_{F_A}(\hat{a}_i). \tag{12}$$

The quantity ω_A is related to ρ_A, as illustrated by the next claim, whose straightforward proof can be found in Appendix B. We recall that θ_A was defined in (2).

Claim 1. *If $T_A^* = [n]$, then $\omega_A \ge \rho_A$. If $T_A^* \ne \emptyset$ and A has integer entries, then $\omega_A \ge \theta_A$.*

The next lemma provides the main tools to detect columns $k \notin T_A^*$ in the full support image algorithm. We recall that F_A is as defined in (8).

Lemma 15. *Let $R \in \mathbb{S}^m_{++}$ such that $F_A \subseteq E(R)$, and let $Q = R^{-1}$. For $k \in [n]$, if $\|\hat{a}_k\|_Q < \omega_A$, then $k \notin T_A^*$.*

Proof. From Lemma 14 and $F_A \subseteq E(R)$, we have $\|\hat{a}_k\|_Q = \mathrm{width}_{E(R)}(\hat{a}_k) \ge \mathrm{width}_{F_A}(\hat{a}_k) \ge \omega_A$. □

Lemma 11 shows that in Algorithm 3, $\det(R)$ increases at least by a factor 16/9 in every rescaling. The following lemma bounds $\min_{k\in[n]} \|\hat{a}_k\|_Q$ in terms of $\det(R)$.

Lemma 16. *At any stage of Algorithm 3 applied to $A \in \mathbb{R}^{m\times n}$, if $\det(R) > 1$, then there exists $k \in [n]$ such that $\|\hat{a}_k\|_Q \le (\det(R)^{1/m} - 1)^{-1/2}$.*

Proof. Let $k = \arg\min_{i\in T} \|\hat{a}_i\|_Q$. Let us use the decomposition of R as in Lemma 9. Then

$$\|\hat{a}_k\|_Q^2 \sum_{i=1}^n \gamma_i \le \sum_{i=1}^n \gamma_i \|\hat{a}_i\|_Q^2 = \sum_{i=1}^n \gamma_i \left(\hat{a}_i^\top Q \hat{a}_i\right) = \mathrm{tr}\left(Q \sum_{i=1}^n \gamma_i \hat{a}_i \hat{a}_i^\top\right) = \mathrm{tr}(Q(R - \alpha I_m)) = \mathrm{tr}(I_m - \alpha Q) = m - \alpha \mathrm{tr}(Q) < m. \tag{13}$$

The third equality used the decomposition of R, the fourth used $QR = I_m$, and the final inequality holds because Q is positive definite.

The fact that $\mathrm{tr}(R) = \alpha m + \sum_{i=1}^n \gamma_i \le m + \sum_{i=1}^n \gamma_i$ and Lemma 1(b) imply that $\sum_{i=1}^n \gamma_i \ge \mathrm{tr}(R) - m \ge m(\det(R)^{1/m} - 1)$. Note that the latter term is positive because $\det(R) > 1$; therefore, the statement follows from (13). □

4.2.1. Basic Maximum Support Image Algorithm. In light of the preceding lemmas, we can extend the full support image algorithm (Algorithm 3) to the maximum support case for a matrix $A \in \mathbb{Z}^{m\times n}$ of encoding size L as follows. Let us assume that $rk(A) = m$. We again use θ_A as in (2) and observe that by Claim 1, $\omega_A \ge \theta_A$ whenever $T^* \ne \emptyset$. We run Algorithm 3 until either we can find y such that $A^\top Q y > 0$ or we find an index k such that $\|\hat{a}_k\|_Q < \theta_A$.

Lemmas 11 and 16 guarantee that either outcome is reached within $O(mL)$ rescalings. In the first case, we terminate with the maximum support solution Qy. Lemma 15 guarantees that in the second case, $k \notin T^*$.

Once an index $k \notin T^*$ is identified, we must have $a_k^\top y = 0$ for every solution y to (*I*). Hence, the necessary update is to project the columns of A onto the subspace $a_k^\perp$. Formally, we compute an orthonormal basis $W \in \mathbb{R}^{m\times(m-1)}$ of $a_k^\perp$ and replace the matrix A by A' obtained from $W^\top A$ by removing the zero columns. Then we recursively apply the same algorithm to A' instead of A. Assume that we obtain y' as the output from the recursive call such that $A'^\top y$ is a maximum support vector in $\mathrm{im}(A'^\top)_+$. Then we output the vector $y = Wy'$ for the original matrix A.

To verify the correctness of this recursive call, we need to show that the maximum support solutions to A and A' are in one-to-one correspondence. Furthermore, we need to provide a lower bound on $\omega_{A'}$ in terms of L. We show that $\omega_{A'} \ge \omega_A$, and therefore, $\theta := \theta_A$ remains a valid lower bound. These claims are formally verified in Lemma 17.

To estimate the running time of the algorithm, we recall that a new column outside T^* can be identified within $O(mL)$ rescalings, and there are at most m recursive calls because every call decreases the rank of the matrix A. As in the full support algorithm, we can implement the iterations between two rescalings in $O(n^2 m)$ arithmetic operations. Furthermore, we need to compute orthonormal bases at every recursive call, which can be done in time $O(r^2)$ for the current rank r. Thus, we obtain a total running time $O(n^2 m^3 L)$.

Lemma 17. *Let $A \in \mathbb{R}^{m\times n}$, and let $H \subseteq \mathbb{R}^m$ be an r-dimensional subspace such that $\Sigma_A \subseteq H$. Let $U \in \mathbb{R}^{m\times r}$ be an orthornormal basis of H, and let A′ be the matrix obtained from $U^\top A$ after removing all 0 columns. The following hold:*

a. $F_A = UF_{A'}$.
b. *For* $v \in \mathbb{R}^m$, $w = U^\top v$, *we have* $\text{width}_{F_A}(v) = \text{width}_{F_{A'}}(w)$ *and* $\text{width}_{F_A}(\hat{v}) \le \text{width}_{F_{A'}}(\hat{w})$.
c. $\omega_A \le \omega_{A'}$.

Proof. (a) Take $x \in UF_{A'}$ and $y \in F_{A'}$ such that $x = Uy$. First, $\|x\| = \|Uy\| = \|y\| \le 1$ because $y \in F_{A'}$. For $i \in [n]$, note that if $U^\top a_i = 0$, then $a_i^\top x = (U^\top a_i)^\top y = 0$, and if not, a_i appears as a column of A', and hence $a_i^\top x = (U^\top a_i)y \ge 0$ because $y \in F_{A'}$. Thus, $x \in F_A$. For $x \in F_A$, because $F_A \subseteq H$, we can write $x = Uy$ for $y \in \mathbb{R}^r$. Applying the previous argument in reverse, we conclude that $y \in F_{A'}$, and hence $x \in UF_{A'}$. Thus, $F_A = UF_{A'}$ as needed.

(b) The equality follows directly from part (a) because

$$\text{width}_{F_A}(v) = \text{width}_{UF_{A'}}(v) = \text{width}_{F_{A'}}\left(U^\top v\right) = \text{width}_{F_{A'}}(w).$$

We prove the inequality. By positive homogeneity of width, if either v or w equals 0, we clearly have $0 = \text{width}_{F_A}(\hat{v}) = \text{width}_{F_{A'}}(\hat{w})$, and the statement follows. So we may assume that both $v, w \neq 0$. Because U is orthonormal, we see that $0 < \|w\| = \|U^\top v\| \le \|v\|$. Because $0 \in F_A$, by homogeneity, we have that

$$0 \le \text{width}_{F_A}(\hat{v}) = \frac{1}{\|v\|}\text{width}_{F_A}(v) = \frac{1}{\|v\|}\text{width}_{F_{A'}}(w) \le \frac{1}{\|w\|}\text{width}_{F_{A'}}(w) = \text{width}_{F_{A'}}(\hat{w}),$$

as needed.

(c) First, note that the set $T^*_{A'}$ comprises the indices of the columns $U^\top a_i$ for which $i \in T^*_A$; that is, $\text{width}_{F_{A'}}(a_i) > 0$. The inequality follows from the last statement in part (b). □

5. Conclusions

We have given polynomial-time algorithms for the full support and maximum support versions of the kernel and image problems. These methods give new insights on how to leverage the underlying geometry of linear (and, more generally, conic) programs.

There is an important conceptual difference between the full support and maximum support variants. The running times of the full support kernel and image algorithms depend on $\log|\rho_A|^{-1}$. However, the algorithms do not require explicit knowledge of ρ_A; this parameter shows up only in the running time analysis. These algorithms can be implemented in the real model of computation.

By contrast, the maximum support variants rely on bit complexity estimations. The algorithms require an integer input matrix and use θ_A, computed from the Hadamard bound, as a threshold for removing columns from the support. Given the duality between maximum support versions of (K_{++}) and (I_{++}), the most natural goal would be to find a *complementary pair* of maximum support solutions to (K) and (I) because such solutions are self-certifying (i.e., each would certify that the other is indeed a maximum support solution). Developing a rescaling algorithm that solves this problem directly using natural geometric potentials, as opposed to the bit complexity arguments presented earlier, is an interesting open problem.

We note that the interior point methods of Vavasis and Ye [34] and Ye [37] provides a complementary pair in the real model of computation, based on certain condition measures (one of them being related to our ω_A). However, these condition measures do not improve over the course of the algorithm. Our goal would be to find an algorithm that finds a rescaling of the problem that simultaneously approximates both kernel and image geometries.

Acknowledgments

This paper is partly based on the paper, "Rescaled Coordinate Descent Methods for Linear Programming," presented at the 18th Conference on Integer Programming and Combinatorial Optimization (Dadush et al. [11]).

Appendix A. Faster Algorithms for the Maximum Support Problems

This appendix exhibits improved versions of the maximum support kernel and image algorithms described in Section 4. The key idea to the amortized analyses is bounding the possible increase in the volume of the ellipsoidal approximation when moving to a lower-dimensional subspace. The following lemma will be useful in computing the projected determinant.

Lemma A.1. *Consider a matrix $R \in \mathbb{S}^d_{++}$. For a vector $a \in \mathbb{R}^d$, $\|a\| = 1$, let $H = \{x : a^\top x = 0\}$. Then* $\det_H(R) = \det(R)\|a\|^2_{R^{-1}}$.

Proof. Let $W \in \mathbb{R}^{d\times(d-1)}$ be a matrix whose columns form an orthonormal basis of H. Because $(W|a)$ is an orthonormal basis of $\mathbb{R}^d$, we have

$$\det(R) = \det\left(\begin{pmatrix} W^\top \\ a^\top \end{pmatrix} R(W|a)\right) = \det\begin{pmatrix} W^\top RW & W^\top Ra \\ a^\top RW & \|a\|^2_R \end{pmatrix} = \det\left(W^\top RW\right)\left(\|a\|^2_R - a^\top RW\left(W^\top RW\right)^{-1}W^\top Ra\right),$$

where the last equality follows from the determinant identity for the Schur complement. Observe that $\|a\|_R^2 - a^\top RW(W^\top RW)^{-1}W^\top Ra = \|q\|^2$, where q is the orthogonal projection of the vector $v := R^{\frac{1}{2}}a$ onto the orthogonal complement of the hyperplane $R^{\frac{1}{2}}H$. The orthogonal complement of this hyperplane is the line generated by $p := R^{-\frac{1}{2}}a$. Thus, $\|q\| = \hat{p}^\top v = (a^\top R^{-\frac{1}{2}}R^{\frac{1}{2}}a)/\|R^{-\frac{1}{2}}a\| = 1/\|a\|_{R^{-1}}$ because $\|a\| = 1$. Therefore, $\det_H(R) = \det(W^\top RW) = \det(R)\|a\|_{R^{-1}}^2$, as required. □

Lemma A.2. *Let $E \subset \mathbb{R}^d$ be an ellipsoid and H an r-dimensional subspace of $\mathbb{R}^d$. Given $a \in H$, $\|a\| = 1$, let $H' = \{x \in H : a^\top x = 0\}$. Then*

$$\mathrm{vol}_{r-1}(E_{H'}) = \frac{\mathrm{vol}_r(E_H)}{\mathrm{width}_E(a)} \cdot \frac{\nu_{r-1}}{\nu_r}.$$

Proof. We can assume that $H = \mathbb{R}^r$, so $E = E_H$. Let $R \in \mathbb{S}_{++}^r$ such that $E = E(R)$. The volume of E can be written as $\mathrm{vol}_r(E) = \nu_r/\sqrt{\det(R)}$, and using (10), we get $\mathrm{vol}_{r-1}(E_{H'}) = \nu_{r-1}/\sqrt{\det_{H'}(R)}$. The statement follows from Lemmas A.1 and 14. □

A.1. Amortized Maximum Support Kernel Algorithm

To describe the algorithm, it is more convenient to work with the scalar products defined by Q than with the rescaled matrix $A' = MA$. Consider a vector $y' = A'x = MAx$ in any iteration of Algorithm 1, and let $y = Ax$. Note that $y' = My$, so when computing $\hat{a}_j'^\top \hat{y}'$, we have $\hat{a}_j'^\top \hat{y}' = \left\langle \frac{a_j}{\|a_j\|_Q}, \frac{y}{\|y\|_Q} \right\rangle_Q$. Rescaling in Algorithm 1 replaces A' by $(I_m + \hat{y}'\hat{y}'^\top)A'$; therefore, the corresponding update of the scalar product consists of replacing M with $(I_m + \hat{y}'\hat{y}'^\top)M$ and recomputing Q. Noting that $\|y'\|_2 = \|My\|_2 = (1+3\varepsilon)^t\|y\|_Q$, the update can be written in terms of Q and y as

$$Q' = \frac{1}{(1+3\varepsilon)^{2(t+1)}} M^\top \left(I_m + \frac{y'y'^\top}{\|y'\|_2^2}\right)\left(I_m + \frac{y'y'^\top}{\|y'\|_2^2}\right) M = \frac{1}{(1+3\varepsilon)^2}\left(Q + \frac{3Qyy^\top Q}{\|y\|_Q^2}\right). \tag{A.1}$$

We define the procedure Rescale(Q, y), which, given Q and y, replaces Q with the matrix Q' defined in (A.1).

In this section, we show that we can improve the running time estimate of the basic maximum support kernel algorithm (Section 4.1) by a factor n by adopting two ideas:

1. Instead of removing a column a_k, $k \in T^*$, from A every time we identify one, we maintain as before a set $S \subseteq [n]$ with the property that $S^* \subseteq S$, as well as a set $T \subseteq S$ of indices that we have determined not to belong to S^* (i.e., $T \cap S^* = \emptyset$ throughout the algorithm). Whenever we conclude that $i \notin S^*$ for an index i based on Lemma 13(b), we add i to T. Columns indexed by T are removed from S only when doing so decreases the rank of the matrix A_S.

2. After removing columns from A, instead of restarting from $Q = I_m$, we restart from the same Q we had at the last iteration. If in a given iteration we remove columns from S, thus obtaining a set $S' \subsetneq S$, it may happen that the relative volume of $E(Q) \cap \mathrm{im}(A_{S'})$ is larger than the relative volume of $E(Q) \cap \mathrm{im}(A_S)$, but Lemma A.3 ensures that the increase in volume is not too large.

Algorithm A.1 (Maximum Support Kernel Algorithm)

```
Input: A matrix $A \in \mathbb{Z}^{m\times n}$
Output: A maximum support solution to the system $(K)$
 1: Compute $\theta := \theta_A$ as in (2)
 2: Compute $\Pi := \Pi_{\hat{A}}^K$
 3: Set $x_j := 1$ for all $j \in [n]$, and $y := \hat{A}x$
 4: Set $S := [n]$, $T := \emptyset$, $Q := I_m$
 5: while $(S \neq \emptyset)$ and $(\Pi x \not> 0)$ do
 6:     Let $k := \arg\min_{i\in S} \langle a_i, y\rangle_Q / \|a_i\|_Q$
 7:     if $\langle a_k, y\rangle_Q < -\varepsilon\|a_k\|_Q\|y\|_Q$, then
 8:         update $x := x - \frac{\langle a_k, y\rangle_Q}{\|a_k\|_Q^2}\vec{e}_k$, $y := y - \frac{\langle a_k, y\rangle_Q}{\|a_k\|_Q^2} a_k$
 9:     else Rescale$(Q, y)$
10:         $T := T \cup \{k \in S\backslash T : \|\hat{a}_k\|_Q > \theta^{-1}\}$
11:         if $\mathrm{rk}(A_{S\backslash T}) < \mathrm{rk}(A_S)$, then Remove$(T)$
12: if $\Pi x > 0$, then
13:     Define $\bar{x}_i \in \mathbb{R}^n$ by $\bar{x}_i := (\Pi x)_i/\|a_i\|_2$ if $i \in S$, $\bar{x}_i := 0$ if $i \notin S$ return $\bar{x}$
14: if $S = \emptyset$, then return $\bar{x} = 0$
```

Algorithm A.2 (Column Deletion)

```
1: procedure Remove$(T)$
2:     $S := S \setminus T$, $T := \emptyset$
3:     Reset $x_j := 1$ for all $j \in S$, $y := \hat{A}_S x$
4:     Recompute $\Pi := \Pi_{\hat{A}_S}^K$
```

Algorithm A.1 terminates either with a solution $\bar{x} \in \ker(A)_+$ with $\mathrm{supp}(\bar{x}) = S$, in which case we may conclude $S = S^*$, or when $S = \emptyset$ is reached, in which case we may conclude that $\bar{x} = 0$ is a maximum support solution.

Theorem A.1. *Let $A \in \mathbb{Z}^{m\times n}$. Algorithm A.1 finds a solution of $Ax = 0$, $x \geq 0$ of maximum support in $O\big((m^3n + mn^2)L\big)$ arithmetic operations.*

The proof of Theorem A.1 requires the following lemma, which gives a bound on the volume increase of the relevant ellipsoid at a column removal step.

Lemma A.3. *Consider a stage of Algorithm A.1 in which* Remove*(T) is called. Let $r = \mathrm{rk}(A_S)$, $r' = \mathrm{rk}(A_{S\setminus T})$, and let $E := E_{\mathrm{im}(A_S)}(Q)$ and $E' := E_{\mathrm{im}(A_{S\setminus T})}(Q)$, where S, T are as in line 11. Then*

$$\frac{\mathrm{vol}_{r'}(E')}{\mathrm{vol}_r(E)} \leq \frac{\nu_{r'}}{\nu_r}\left(\frac{2}{\theta^2}\right)^{r-r'}.$$

Proof. Let T' denote the state of T in the previous iteration, that is, before the update at line 10. Because Remove was not called in the previous iteration, we have that $r = \mathrm{rk}(A_S) = \mathrm{rk}(A_{S\setminus T'}) > \mathrm{rk}(A_{S\setminus T}) = r'$. Because rank can only decrease by one after the removal of a column, we can construct a sequence of sets $S \setminus T := S_{r'} \subset S_{r'+1} \subset \cdots \subset S_r := S \setminus T'$ such that $\mathrm{rk}(A_{S_k}) = k$ for $k \in \{r', \ldots, r\}$. To prove the desired statement, it suffices to show that for $k \in \{r'+1, \ldots, r\}$,

$$\frac{\mathrm{vol}_{k-1}(E_{k-1})}{\mathrm{vol}_k(E_k)} \leq \frac{\nu_{k-1}}{\nu_k}\left(\frac{2}{\theta^2}\right),$$

where $E_l := E_{\mathrm{im}(A_{S_l})}(Q)$, $l \in \{r', \ldots, r\}$. This follows by induction, recalling that $E_{r'} = E'$ and $E = E_r$ (because $\mathrm{im}(A_{S_r}) = \mathrm{im}(A_{S\setminus T'}) = \mathrm{im}(A_S)$).

Let $k \in \{r'+1, \ldots, r\}$, $H_{k-1} := \mathrm{im}(A_{S_{k-1}})$, and $H_k := \mathrm{im}(A_{S_k})$. Let $v \in H_k$ be the vector orthogonal to H_{k-1} such that $\|v\|_2 = 1$. By Lemma A.2,

$$\mathrm{vol}_{k-1}(E_{k-1}) = \frac{\mathrm{vol}_k(E_k)}{\mathrm{width}_{E_k}(v)} \cdot \frac{\nu_{k-1}}{\nu_k},$$

and hence it suffices to show that $\mathrm{width}_{E_k}(a) \geq \frac{\theta^2}{2}$.

Because at every rescaling the Q-norm of the columns of $\hat{A}_{S\setminus T'}$ increases by at most a factor of 2, and because in the previous iteration the columns had Q-norm at most θ^{-1} (otherwise they would have been added to T'), their Q-norm during the current iteration is at most $2\theta^{-1}$. In particular, because $S_k \subseteq S \setminus T'$, we have $\|\hat{a}_i\|_Q \leq 2\theta^{-1}$ for all $i \in S_k$. From here, we have that

$$\mathrm{width}_{E_k}(v) = \max\{v^\top z : \|z\|_Q \leq 1,\, z \in H_k\} \geq \max_{i\in S_k} \frac{|v^\top \hat{a}_i|}{\|\hat{a}_i\|_Q} \geq \frac{\theta}{2} \min_{y\in H_k\setminus\{0\}} \max_{i\in S_k} |\hat{y}^\top \hat{a}_i| = \frac{\theta}{2}\left|\rho_{(A_{S'}, -A_{S'})}\right| \geq \frac{\theta^2}{2},$$

where the last inequality follows from $|\rho_{(A_{S'},-A_{S'})}| \geq \theta$. □

Proof of Theorem A.1. If the algorithm terminates with $\Pi x > 0$, then it correctly outputs a solution to (K_{++}). Next, we observe that throughout the algorithm, $S \supseteq S^*$, which implies that the solution returned at the end is always a maximum support solution. To prove this, we need to show only that $T \subseteq T^*$ throughout. New elements are added to T in line 10 that are in T^* by Lemma 13 and the fact that $\rho_{A_{S^*}} \geq \theta_A$ if $S \neq \emptyset$.

We need to argue that the algorithm terminates in the claimed number of iterations. Recall that by Lemma 12, we have $P_A = P_{A_S} = P_{A_{S^*}}$ throughout the algorithm because $S^* \subseteq S$.

A *round* of the algorithm consists of the iterations that take place between two consecutive calls of Remove. Because Remove(T) is called only when $\mathrm{rk}(A_{S\setminus T}) < \mathrm{rk}(A_S)$, the number of rounds is at most $\mathrm{rk}(A) \leq m$. We want to bound the total number of rescalings performed by the algorithm.

Claim A.1. *The total number K of rescalings throughout the algorithm is $O(m\log(\theta^{-1}))$.*

Proof. In any given round, let $E := E_{\mathrm{im}(A_S)}(Q)$ and $r = \mathrm{rk}(A_S)$. We first show that at every rescaling within the round, except for the last, the invariant

$$\mathrm{vol}_r(E) \geq \nu_r\theta^{2r} \tag{A.2}$$

is maintained. Indeed, by Lemma 13(a), $P_A \subseteq E$ throughout. Because $\|\hat{a}_j\|_Q \leq \theta^{-1}$ for all $j \in S\setminus T$, it follows that $E \supseteq \theta\mathrm{conv}(\hat{A}_{S\setminus T}, -\hat{A}_{S\setminus T})$. Because at every rescaling except for the last one of the round we have $\mathrm{rk}(A_{S\setminus T}) = r$, it follows by Lemma 2 that $\mathrm{conv}(\hat{A}_{S\setminus T}, -\hat{A}_{S\setminus T})$ contains an r-dimensional ball of radius $|\rho_{(A_{S\setminus T}, -A_{S\setminus T})}| \geq \theta$. This implies (A.2).

At the first iteration, $Q = I_m$, $S = \emptyset$, and $E = \mathbb{B}_m \cap \mathrm{im}(A)$; therefore, initially, $\mathrm{vol}_r(E) \leq \nu_r$. By Lemma 13(c), at every rescaling in which we do not remove any column, $\mathrm{vol}_r(E)$ decreases by at least $2/3$; Lemma A.3 bounds the increase in $\mathrm{vol}_r(E)$ at column removals. Combined with the lower bound (A.2), we obtain that the total number of rescalings is at most m plus the smallest number K satisfying

$$\left(\frac{2}{3}\right)^K \cdot \left(\frac{2}{\theta^2}\right)^m < \theta^{2m}.$$

The claimed bound on K follows. □

By Lemma 6, the algorithm is guaranteed to terminate when $\|y\|_2 < |\rho_{(A_S,-A_S)}|$, so, in particular, $\|y\|_2 \geq \theta$ throughout the algorithm because $\theta \leq |\rho_{(A_S,-A_S)}|$. By Lemma 7, after t rescalings, $\|y\|_2 \leq \|y\|_Q(1+3\varepsilon)^t$. Hence, the algorithm is guaranteed to terminate if $\|y\|_Q \leq \theta/(1+3\varepsilon)^K$.

At the beginning of each round, we reinitialize x so that $x_j = 1$ for all $j \in S$. In particular, $y = \hat{A}_S x$ satisfies $\|y\|_Q \leq |S|\theta^{-1} \leq n\theta^{-1}$ because $\|\hat{a}_j\|_Q \leq \theta^{-1}$ for all $j \in S$. At every rescaling within the same round, $\|y\|_Q$ increases by $2/(1+3\varepsilon)$, and in every DV update, it decreases by at least a factor of $\sqrt{1-\varepsilon^2}$. Let R be the number of rounds, and let $K_1, \dots, K_R$ be the number of rescalings within rounds $1, \dots, R$.

It follows that, at the ith round, the number of DV updates is at most the smallest number κ_i such that

$$n\theta^{-1}(1-\varepsilon^2)^{\kappa_i/2}2^{K_i} < \theta/(1+3\varepsilon)^K.$$

Taking the logarithm on both sides and recalling that $\log(1-\varepsilon^2) < -\varepsilon^2$ and $\log(1+3\varepsilon) \geq 3\varepsilon$, it follows from our choice of ε that $\kappa_i \in O(m^2)K_i + O(m)K$. Because $K = K_1 + \cdots + K_R$ and $R \leq m$, this implies that the total number of DV updates is $O(m^2)K$. Using Claim A.1, the total number of DV updates is $O(m^3 \log(\theta^{-1}))$. As explained in the Proof of Theorem 1, we can perform each DV update in $O(n)$ arithmetic operations, provided that at each rescaling we recompute $F = A_S^\top Q A_S$, which, as we showed, can be done in $O(n^2)$ arithmetic operations. Observe also that $\|a_j\|_Q^2$ is the jth diagonal entry of F. Because $\theta^{-1} \leq 2^{4L}$ by Lemma 3, the total number of arithmetic operations performed for DV updates and rescalings is within the stated bound.

Every time a new column is added to T at line 10, we need to then test at line 11 whether $\text{rk}(A_{S\setminus T}) < \text{rk}(A_S)$. This can be done in $O(m^2 n)$ operations via Gaussian elimination. Because new columns are added to T at most n times, and because $n \leq L$, the total number of arithmetic operations required to test rank is $O(m^2 nL)$, which is within the stated running time bound.

Finally, at the beginning of each round, we need to recompute the projection matrix. Computing each projection matrix requires time $O(n^2 m)$, and the total number of rounds is at most m. Because $n \leq L$, the total number of arithmetic operations performed to recompute the projection matrices is $O(m^2 nL)$, which is within the stated bound. □

A.2. Amortized Maximum Support Image Algorithm

Analogously to the kernel setting, we now improve the running time of the basic maximum support image algorithm (Section 4.2) by a factor of m. The maximum support image algorithm (Algorithm A.3) maintains a set T of indices with the property $T^* \subseteq T$. The set T is initialized as $T = [n]$, and we remove an index a_k once we conclude that $k \notin T^*$. The algorithm terminates with a solution $\bar{y}$ such that $a_k^\top \bar{y} > 0$ for all $i \in T$ and $a_k^\top \bar{y} = 0$ for all $i \notin T$, verifying $T = T^*$ at termination. We maintain r as the number of rows of A throughout the algorithm. As in the full support case, we assume that initially the matrix has full row rank; this will be preserved throughout the reduction steps.

Algorithm A.3 (Maximum Support Image Algorithm)

Input: A matrix $A \in \mathbb{R}^{m\times n}$ with $\text{rk}(A) = m$

Output: A solution $\bar{y} \in \mathbb{R}^m$ to (*I*) satisfying the maximum number of strict inequalities

```
1: Compute $\theta := \theta_A$ as in (2)
2: Set $Q := I_m$, $R := I_m$, $U := I_m$, $T := [n]$, $r := m$
3: **while** $T \neq \emptyset$ **do**
4:     Call von Neumann$(A, Q, \varepsilon)$ to obtain $(x, y)$
5:     **if** $A^\top Q y > 0$, **then return** $\bar{y} = UQy$ **Terminate**
6:     **else** rescale

$$R := \frac{1}{1+\varepsilon}\left(R + \sum_{i\in T} \frac{x_i}{\|a_i\|_Q^2} a_i a_i^\top\right); \quad Q := R^{-1}.$$

7:     **while** $\exists k \in T$ such that $(\|\hat{a}_k\|_Q < \theta)$ **do**
8:         Remove$(k)$
     **return** $\bar{y} = 0$
```

Algorithm A.4 (Column Deletion)

```
1: procedure Remove($k$)
2:     Select $W \in \mathbb{R}^{r\times(r-1)}$ whose columns form an orthonormal basis of $a_k^\perp$
3:     Set $A := W^\top A$, delete all 0 columns, and remove the corresponding indices from $T$
4:     Set $R := W^\top RW$, $U := UW$, and $r := r-1$; recompute $Q = R^{-1}$
```

Theorem A.2. *Let the matrix $A \in \mathbb{Z}^{m\times n}$ have $\mathrm{rk}(A) = m$ and encoding length L. Algorithm A.3 finds a maximum support solution to $A^\top y \geq 0$ in $O\big(m^2n^2 \cdot L\big)$ arithmetic operations. Using the smoothed perceptron of Soheili and Peña [33] or the Mirror Prox method of Yu et al. [38] instead of the von Neumann algorithm requires $O(m^3n\sqrt{\log n} \cdot L)$ arithmetic operations.*

We need the following stronger version of Lemma 9, with explicit bounds on the coefficients. Note that the dimension m is replaced by the actual dimension r and the set of columns $[n]$ by T.

Lemma A.4. *At any stage of the algorithm, the matrix R is positive definite and can be written in the form*

$$R = \alpha I_r + \sum_{i\in T}\gamma_i\hat{a}_i\hat{a}_i^\top,$$

where $\gamma_i \leq 2/\theta^2$, for all $i \in T$; $\alpha = 1/(1+\varepsilon)^t$ for the total number of rescalings t performed thus far; and $\gamma_i \geq 0$. The trace is $\mathrm{tr}(R) = \alpha r + \sum_{i\in T}\gamma_i$. Furthermore, for any $v \in \mathbb{R}^r$ with $\|v\| = 1$, we have $\|v\|_Q \geq \theta/\sqrt{2(n+1)}$.

Proof. Clearly, any matrix of this form is positive definite. The proof is by induction. The formula and bound are valid at initialization when $R = I_m$ and $\gamma_i = 0$ for all $i \in [n]$. Let $R = \alpha I_r + \sum_{i\in T}\gamma_i\hat{a}_i\hat{a}_i^\top$ denote the current decomposition, where $\gamma_i \leq 2/\theta^2$. We show that the required form and bounds hold for the next update.

Assume that we rescale in the current iteration. Let R and Q denote the matrices before, and R' and Q' after, the rescaling. For $i \in [n]$, using Lemma 14, we see that

$$\|\hat{a}_i\|_Q^2 = \mathrm{width}^2_{E(R)}(\hat{a}_i) = \max\left\{(\hat{a}_i{}^\top x)^2 : \alpha\|x\|^2 + \sum_{j\in T}\gamma_j\left(\hat{a}_j^\top x\right)^2 \leq 1, x \in \mathbb{R}^r\right\} \leq \frac{1}{\gamma_i}.$$

Now let x be the convex combination returned by the von Neumann algorithm in line 4. By the rescaling formula in line 6, the matrix R is updated to R', satisfying

$$R' = \frac{1}{1+\epsilon}\left(R + \sum_{i\in T}\frac{x_i}{\|a_i\|_Q^2}a_ia_i^\top\right) = \frac{1}{1+\epsilon}\left(\alpha I_r + \sum_{i\in T}\left(\gamma_i + \frac{x_i}{\|\hat{a}_i\|_Q^2}\right)\hat{a}_i\hat{a}_i^\top\right).$$

Hence, recalling that $\|\hat{a}_i\|_Q \geq \theta$ for every $i \in T$ at the beginning of every iteration, each γ_i is updated to γ_i', satisfying

$$\gamma_i' = \frac{1}{1+\epsilon}\left(\gamma_i + \frac{x_i}{\|\hat{a}_i\|_Q^2}\right) \leq \frac{2}{\|\hat{a}_i\|_Q^2} \leq \frac{2}{\theta^2}.$$

Consider now a step where some columns are eliminated. Then the matrices A and R are updated to A' and R', where A' is obtained by removing the zero columns from $W^\top A$ and $R' = W^\top RW$. We denote by $T' \subseteq T$ the index set of columns of A'. Thus,

$$R' = \alpha W^\top W + \sum_{i\in T'}\gamma_iW^\top\hat{a}_i\hat{a}_i^\top W = \alpha I_{r-1} + \sum_{i\in T'}\gamma_i\|W^\top\hat{a}_i\|^2\frac{a_i'a_i'^\top}{\|a_i'\|^2},$$

where the last equality follows from $W^\top W = I_{r-1}$ and the fact that $W^\top a_i = 0$ for all $i \in T\setminus T'$. Setting $\alpha' = \alpha$ and $\gamma_i' = \gamma_i\|W^\top\hat{a}_i\|^2$ for $i \in T'$ gives the desired decomposition of R'. Next, because $\|W^\top\hat{a}_i\| \leq \|\hat{a}_i\| \leq 1$, we get that $\gamma_i' \leq \gamma_i \leq 2/\theta^2$, for all $i \in T'$.

We now prove the last part, lower bounding $\|v\|_Q$ for any unit vector $v \in \mathbb{R}^r$. First, for any $x \in \mathbb{R}^r$, the Cauchy–Schwarz inequality gives

$$x^\top Rx = \alpha\|x\|^2 + \sum_{i\in T}\gamma_i(\hat{a}_i{}^\top x)^2 \leq \left(\alpha + \sum_{i\in T}\gamma_i\right)\|x\|^2,$$

and hence $E(R)$ contains a Euclidean ball of radius at least $1/\sqrt{\alpha + \sum_{i\in T}\gamma_i}$. Therefore, for any unit vector $v \in \mathbb{R}^r$, using Lemma 14, we get

$$\|v\|_Q = \max\{v^\top x : x \in E(R)\} \geq \frac{1}{\sqrt{\alpha + \sum_{i\in T}\gamma_i}} \geq \frac{1}{\sqrt{1 + 2|T|/\theta^2}} \geq \frac{\theta}{\sqrt{2(n+1)}},$$

as needed. □

The next lemma gives a lower bound on the decrease in $\det(R)$ for column removal steps.

Lemma A.5. *Assume that at a given iteration, $F_A \subseteq E(R)$, and consider an index $k \in T \setminus T^*$. Let $W \in \mathbb{R}^{r\times(r-1)}$ be a matrix whose columns form an orthonormal basis of $a_k^\perp$. Let A' be the matrix obtained by removing all zero columns from $W^\top A$, and let $R' = W^\top RW$. Then $F_{A'} \subseteq E(R')$ and $\det(R') \ge \det(R)\theta^2/(2(n+1))$.*

Proof. The first part of the statement follows from the fact that $F_A \subseteq E_{a_k^\perp}(R)$. Furthermore, Lemma 17(a) implies that $F_{A'} = W^\top W F_{A'} = W^\top F_A$. Thus, $F_{A'} = W^\top F_A \subseteq W^\top E_{a_k^\perp}(R) = W^\top W E(R') = E(R')$. For the second part, note that $\det(R') = \det_{a_k^\perp}(R) = \det(R)\|\hat{a}_k\|_Q^2$ using Lemma A.1. To obtain the desired bound, we use the estimate $\|\hat{a}_k\|_Q^2 \ge \theta^2/(2(n+1))$ from Lemma A.4, which holds because $\hat{a}_k$ is a unit vector. □

We now prove Theorem A.2 based on these lemmas and the results proved in Section 3.1.

Proof of Theorem A.2. We first argue the correctness of the algorithm. Let A be the input matrix, and let A' be the current matrix during any stage of the algorithm. For the current matrix R, Lemmas 10 and A.5 ensure that $F_{A'} \subseteq E(R)$. Therefore, Lemma 15 implies that $T \supseteq T^*$ throughout the algorithm, so Σ_A is contained in the subspace $H := \{x \in \mathbb{R}^m : a_i^\top x = 0\ \forall i \in [n]\setminus T\}$. By construction, the columns of U are an orthonormal basis of H, and A' is obtained from the matrix $U^\top A$ by removing the 0 columns.

We next show that the solution $\bar{y}$ returned by the algorithm is a solution to (I) satisfying the maximum number of strict inequalities. If $T = \emptyset$ at termination, then $\bar{y} = 0$ is indeed a maximum support solution. Assume that the algorithm terminated at line 5 with $\bar{y} = UQy$. Then, for every $k \in T$, we have $a_k^\top \bar{y} = a_k^\top UQy = (a'_k)^\top Qy > 0$, whereas for every $k \notin T$, we have $a_k^\top \bar{y} = a_k^\top UQy = 0$, because $U^\top a_k = 0$ by construction.

We now prove that the algorithm terminates in the claimed number of iterations. Lemma 16 remains valid; the proof uses Lemma A.4 in place of Lemma 9. By Lemmas 16 and 17(c), whenever $\det(R) > (1+\theta^{-2})^m$, we can find $k \in T$ such that $\|a\|_Q < \theta$; thus, we remove at least one column at line 8. The potential $\det(R)$ is initially 1; by Lemma 11, it increases by at least a factor of 16/9 at every rescaling, and by Lemma A.5, it decreases by at most a factor of $\frac{\theta^2}{2(n+1)}$ after elimination of a column. Because $\mathrm{rk}(A)$ decreases by 1 every time we remove a column, the algorithm performs at most m column removals. Consequently, within $O(m\log(n\theta^{-1})) = O(mL)$ rescalings, all columns outside T^* are removed, and the algorithm terminates.

As in the Proof of Theorem 2, the iterations between two rescalings can be implemented in time $O(n^2m)$, whereas recomputing R and Q when rescaling requires $O(m^2n)$ operations. This contributes $O(m^2n^2L)$ to the overall running time. When removing a column, computing W requires computing an orthonormal basis of a_k in $\mathbb{R}^r$, which can be done by closed-form formula in $O(r^2)$ arithmetic operations; computing WA and WRW requires $O(m^2n)$ and $O(m^3)$, respectively; recomputing the inverse Q or R requires $O(m^3)$ operations. Hence, the total number of arithmetic operations needed for the $O(m)$ column removals is $O(m^3n)$. This implies the stated running time bound.

From the preceding, following the Proof of Theorem 3, we obtain the running time bound for the smoothed perceptron of Soheili and Peña [33] or the Mirror Prox method of Yu et al. [38]. □

Appendix B. Missing Proofs

Proof of Lemma 2. Note that $\rho_A = \tau_{\hat{A}}$ as defined in Lemma B.1, which shows that $|\rho_A|$ is the distance of 0 from the relative boundary of $\mathrm{conv}(A)$. (a) By Lemma B.1(b), $\rho_A < 0$ if and only if 0 is in the relative interior of $\mathrm{conv}(A)$, which is the case if and only if there exists $x > 0$ such that $Ax = 0$.

(b) For any $\bar{y} \in \Sigma_A$, $\|\bar{y}\| = 1$, the distance between $\bar{y}$ and the hyperplane $\{y : a_j^\top y = 0\}$ $(j \in [n])$ is $\hat{a}_j^\top \bar{y}$; therefore, $\min_{j\in[n]} \hat{a}_j^\top \bar{y}$ is the distance of $\bar{y}$ from the boundary of Σ_A, that is, the radius of the largest ball centered at $\bar{y}$ and contained in Σ_A. The statement then follows from the definition of ρ_A. □

Lemma B.1. *Let $A \in \mathbb{R}^{m\times n}$. Let p be a point of minimum norm in the relative boundary of $\mathrm{conv}(A)$. Define*

$$\tau_A \overset{\text{def}}{=} \max_{y\in \mathrm{im}(A)\setminus\{0\}} \min_{z\in\mathrm{conv}(A)} z^\top \hat{y}.$$

a. *If $0 \notin \mathrm{conv}(A)$, then $\|p\| = \tau_A = \min_{j\in[n]} a_j^\top \hat{p}$.*
b. *If 0 is in the relative interior of $\mathrm{conv}(A)$, then p is in the relative interior of some facet of $\mathrm{conv}(A)$, and $\|p\| = -\tau_A = \max_{j\in[n]} a_j^\top \hat{p}$.*

Proof. (a) Assume that $0 \notin \mathrm{conv}(A)$. Then p is a point of minimum norm in $\mathrm{conv}(A)$. It follows that $p^\top z \ge \|p\|^2$ for every $z \in \mathrm{conv}(A)$, implying that $\|p\| \le \tau_A$. We now show that $\tau_A \le \|p\|$. If not, then there exists $y \in \mathrm{im}(A)$ such that $\|y\| = 1$ and $\min_{j\in[n]} a_j^\top y > \|p\|$. In particular, this implies that every point in $\mathrm{conv}(A)$ has distance greater than $\|p\|$ from the origin, contradicting our choice of $p \in \mathrm{conv}(A)$.

(b) Assume that 0 is in the relative interior of $\mathrm{conv}(A)$. By our choice of p, for any $y \in \mathrm{im}(A)$, $\|y\| = 1$, we have $z := -\|p\|y \in \mathrm{conv}(A)$ and $z^\top y = -\|p\|$, which implies that $\tau_A \le -\|p\|$. For the other direction, consider any facet F of $\mathrm{conv}(A)$ containing p, let H be the affine hyperplane of $\mathrm{im}(A)$ containing F, and let q be the minimum norm point in H. Because $p \in F \subseteq H$, by definition, $\|q\| \le \|p\|$. Because F is a facet, $q^\top z \le \|q\|^2$ is a defining inequality for F (i.e., it is verified by all $z \in \mathrm{conv}(A)$ and satisfied as equality by all $z \in F$). If we let $y = -q$, this shows that $\tau_A \ge \min_{z\in\mathrm{conv}(A)} \hat{y}z \ge -\|q\| \ge -\|p\|$. This shows that $p = q$ and $\tau_A = -\|p\|$. In particular, F must be the only facet containing p; therefore, p is in the relative interior of F. □

Claim B.1. *Let $A \in \mathbb{Z}^{m\times n}$. If $T_A^* \neq \emptyset$, then*

$$\max_{y\in\Sigma_A\setminus\{0\}} \min_{j\in T^*} a_j^\top \hat{y} \geq \frac{1}{m^2\Delta_A}.$$

Proof. Let $S^* := S_A^*$ and $T^* := T_A^*$. Let $(y^*, s^*) \in \mathbb{R}^{2m}$ be an optimal basic solution of the following linear program:

$$\begin{aligned} \min \quad & \vec{e}^\top s \\ A_{T^*}^\top y &\geq \vec{e}, \\ A_{S^*}^\top y &= 0, \\ s - y &\geq 0, \\ s + y &\geq 0. \end{aligned} \tag{B.1}$$

This linear program is feasible by the definition of T^*, and the optimal value equals $\|y^*\|_1$. Note that for every square submatrix A' of A, $|\det(A')| \leq \Delta_{A'} \leq \Delta_A$, where the first inequality follows from Hadamard's bound and the second from the fact that the entries of A are integer. From this fact, a straightforward application of Cramer's rule implies that $s_j^* \leq m\Delta_A$ for all $j \in [n]$, so $\|y^*\|_1 \leq m^2\Delta_A$. Because, by construction, $y^* \in \Sigma_A\setminus\{0\}$, the statement follows from the fact that

$$\min_{j\in T^*} a_j^\top \frac{y^*}{\|y^*\|_2} \geq \frac{1}{\|y^*\|_2} \geq \frac{1}{\|y^*\|_1} \geq \frac{1}{m^2\Delta_A}. \quad \square$$

Proof of Lemma 3. Let $\alpha := \max_{i\in T^*} \|a_i\|$. Note that $|\rho_A| \geq |\tau_A|/\alpha$, where τ_A is defined as in Lemma B.1. Because $\alpha \leq \Delta_A$, it suffices to show that $|\tau_A| \geq 1/(m^2\Delta_A)$.

If $0 \notin \operatorname{conv}(A)$, then $T^* = [n]$, and we observe that $\tau_A = \max_{y\in\Sigma_A\setminus\{0\}} \min_{j\in[n]} a_j^\top \hat{y} \geq \frac{1}{m^2\Delta_A}$, where the inequality follows from Claim B.1. Assume now that 0 is in the relative interior of $\operatorname{conv}(A)$. Let p be a point of minimum norm in the relative boundary of $\operatorname{conv}(A)$. According to Lemma B.1, $|\tau_A| = \|p\|$, and p is contained in the relative interior of a facet F of $\operatorname{conv}(A)$. Let A' be the submatrix of A comprised of the columns that are contained in F. In particular, $\operatorname{conv}(A') = F$; therefore, $0 \notin \operatorname{conv}(A')$, which implies that $\tau_{A'} > 0$. By the previous argument, $\tau_{A'} \geq 1/(m^2\Delta_{A'}) \geq 1/(m^2\Delta_A)$. Because p is the point of minimum norm in F, it follows from Lemma B.1 that $\tau_{A'} = \|p\|$. It follows that $|\tau_A| = \tau_{A'} \geq 1/(m^2\Delta_{A'})$.

Finally, because $\Delta_A \leq 2^L$ (see, e.g., Grötschel et al. [19, lemma 1.3.3]) and $m \leq L$, it follows that $1/(m^2\Delta_A^2) \geq 2^{-4L}$. $\square$

Proof of Lemma 12. We first show that $P_A = P_{A_{S^*}}$. The inclusion $P_{A_{S^*}} \subseteq P_A$ is obvious. For the reverse inclusion, consider $y \in P_A$, and let $x, z \in \mathbb{R}_+^n$ such that $\vec{e}^\top x = \vec{e}^\top z = 1$ and $y = \hat{A}x = -\hat{A}z$. Then $\hat{A}(x+z) = 0$, $x + z \geq 0$, which implies $x_i = z_i = 0$ for all $i \in [n]\setminus S^*$, which shows that $y \in P_{A_{S^*}}$.

We show $\operatorname{span}(P_A) = \operatorname{im}(A_{S^*})$. It suffices to show that $\operatorname{span}(P_{A_{S^*}}) = \operatorname{im}(A_{S^*})$ because $P_A = P_{A_{S^*}}$. The inclusion $\operatorname{span}(P_{A_{S^*}}) \subseteq \operatorname{im}(A_{S^*})$ is obvious. For the reverse inclusion, it suffices to show that for every $i \in S$, there exists $\alpha \neq 0$ such that $\alpha a_i \in P_{A_{S^*}}$. Consider $\lambda \in \mathbb{R}_{++}^{|S^*|}$ such that $\hat{A}_{S^*}\lambda = 0$, and assume without loss of generality that $\sum_{j\in S^*\setminus\{i\}} \lambda_j = 1$. Then $-\lambda_i\hat{a}_i = \sum_{j\in S^*\setminus\{i\}} \lambda_j\hat{a}_j$, which implies that $-\lambda_i\hat{a}_i \in P_{A_{S^*}}$. $\square$

Proof of Claim 1. First, observe that

$$\omega_A = \min_{j\in T^*} \max_{y\in\Sigma_A\setminus\{0\}} \hat{a}_j^\top \hat{y} \geq \max_{y\in\Sigma_A\setminus\{0\}} \min_{j\in T^*} \hat{a}_j^\top \hat{y} \overset{\text{def}}{=} \eta_A.$$

Note that if $T^* = [n]$, then $\eta_A = \rho_A$, which proves the first part of the statement. For the second part of the statement, assume that A has integer entries and that $T^* \neq \emptyset$. Letting $\alpha := \max_{i\in T^*} \|a_i\|$, we have

$$\eta_A \geq \alpha^{-1} \max_{y\in\Sigma_A\setminus\{0\}} \min_{j\in T^*} a_j^\top \hat{y} \geq \frac{1}{m^2\Delta_A^2} = \theta_A,$$

where the last inequality follows from $\alpha \leq \Delta_A$ and Claim B.1. $\square$

Endnote

[1] The Frank–Wolfe method is originally described for a compact set, but the set here is unbounded. Nevertheless, one can easily modify the method by moving along an unbounded recession direction.

References

[1] Agmon S (1954) The relaxation method for linear inequalities. *Canadian J. Math.* 6(3):382–392.
[2] Basu A, Loera JAD, Junod M (2013) On Chubanov's method for linear programming. *INFORMS J. Comput.* 26(2):336–350.
[3] Belloni A, Freund RM, Vempala S (2009) An efficient rescaled perceptron algorithm for conic systems. *Math. Oper. Res.* 34(3):621–641.
[4] Betke U (2004) Relaxation, new combinatorial and polynomial algorithms for the linear feasibility problem. *Discrete Comput. Geometry* 32(3): 317–338.
[5] Blum L, Shub M, Smale S (1989) On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc.* 21(1):1–46.
[6] Chubanov S (2011) A polynomial relaxation-type algorithm for linear programming. Working paper, Bosch Center for Artificial Intelligence, Siegen, Germany.
[7] Chubanov S (2012) A strongly polynomial algorithm for linear systems having a binary solution. *Math. Programming* 134(2):533–570.
[8] Chubanov S (2015) A polynomial algorithm for linear optimization which is strongly polynomial under certain conditions on optimal solutions. Working paper, Bosch Center for Artificial Intelligence, Siegen, Germany.
[9] Chubanov S (2015) A polynomial projection algorithm for linear feasibility problems. *Math. Programming* 153(2):687–713.
[10] Chubanov S (2017) A polynomial algorithm for linear feasibility problems given by separation oracles. Working paper, Bosch Center for Artificial Intelligence, Siegen, Germany.
[11] Dadush D, Végh LA, Zambelli G (2016) Rescaled coordinate descent methods for linear programming. Louveaux Q, Skutella M, eds. *Proc. 18th Internat. Conf. Integer Programming Combin. Optim.* (Springer-Verlag, Berlin), 26–37.
[12] Dadush D, Végh LA, Zambelli G (2018) Geometric rescaling algorithms for submodular function minimization. Czumaj A, ed. *Proc. 29th Annual ACM-SIAM Sympos. Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia), 832–848.
[13] Dantzig GB (1991) Converting a converging algorithm into a polynomially bounded algorithm. Technical report, Stanford University, Stanford, CA.
[14] Dantzig GB (1992) An ε-precise feasible solution to a linear program with a convexity constraint in $1/\varepsilon^2$ iterations independent of problem size. Technical Report 92-5, Stanford University, Stanford, CA.
[15] Dunagan J, Vempala S (2008) A simple polynomial-time rescaling algorithm for solving linear programs. *Math. Programming* 114(1): 101–114.
[16] Epelman M, Freund RM (2000) Condition number complexity of an elementary algorithm for computing a reliable solution of a conic linear system. *Math. Programming* 88(3):451–485.
[17] Frank M, Wolfe P (1956) An algorithm for quadratic programming. *Naval Res. Logist. Quart.* 3(1–2):95–110.
[18] Goffin J (1980) The relaxation method for solving systems of linear inequalities. *Math. Oper. Res.* 5(3):388–414.
[19] Grötschel M, Lovász L, Schrijver A (2012) *Geometric Algorithms and Combinatorial Optimization*, Algorithms and Combinatorics, vol. 2 (Springer-Verlag, Berlin Heidelberg).
[20] Hoberg R, Rothvoss T (2017) An improved deterministic rescaling for linear programming algorithms. Eisenbrand F, Koenemann J, eds. *Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science, vol. 10328 (Springer, Cham, Switzerland), 267–278.
[21] Li D, Terlaky T (2013) The duality between the perceptron algorithm and the von Neumann algorithm. Zuluaga L, Terlaky T, eds. *Modeling and Optimization: Theory and Applications*, Springer Proceedings in Mathematics & Statistics, vol. 62 (Springer, New York), 113–136.
[22] Li D, Roos C, Terlaky T (2015) A polynomial column-wise rescaling von Neumann algorithm. Working paper, Capital One, Plano, TX.
[23] Motzkin T, Schoenberg I (1954) The relaxation method for linear inequalities. *Canadian J. Math.* 6(3):393–404.
[24] Nemirovski A (2004) Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM J. Optim.* 15(1):229–251.
[25] Nesterov Y (2005) Smooth minimization of non-smooth functions. *Math. Programming* 103(1):127–152.
[26] Novikoff AB (1962) On convergence proofs for perceptrons. *Proc. Sympos. Math. Theory Automata* (Polytechnic Institute of Brooklyn, New York), 615–622.
[27] Peña J, Soheili N (2016) A deterministic rescaled perceptron algorithm. *Math. Programming* 155(1–2):497–510.
[28] Peña J, Soheili N (2017) Solving conic systems via projection and rescaling. *Math. Programming* 166(1–2):87–111.
[29] Peña J, Rodriguez D, Soheili N (2016) On the von Neumann and Frank–Wolfe algorithms with away steps. *SIAM J. Optim.* 26(1):499–512.
[30] Roos K (2015) On Chubanov's method for solving a homogeneous inequality system. Al-Baali M, Grandinetti L, Purnama A, eds. *Numerical Analysis and Optimization*, Springer Proceedings in Mathematics & Statistics, vol. 134 (Springer, Cham, Switzerland), 319–338.
[31] Rosenblatt F (1958) The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.* 65(6):386–408.
[32] Schrijver A (1998) *Theory of Linear and Integer Programming* (John Wiley & Sons, New York).
[33] Soheili N, Peña J (2012) A smooth perceptron algorithm. *SIAM J. Optim.* 22(2):728–737.
[34] Vavasis SA, Ye Y (1995) Condition numbers for polyhedra with real number data. *Oper. Res. Lett.* 17(5):209–214.
[35] Végh LA, Zambelli G (2014) A polynomial projection-type algorithm for linear programming. *Oper. Res. Lett.* 42(1):91–96.
[36] Wolfe P (1976) Finding the nearest point in a polytope. *Math. Programming* 11(1):128–149.
[37] Ye Y (1994) Toward probabilistic analysis of interior-point algorithms for linear programming. *Math. Oper. Res.* 19(1):38–52.
[38] Yu AW, Kılınç-Karzan F, Carbonell JG (2014) Saddle points and accelerated perceptron algorithms. *Proc. Machine Learn. Res.* 32(2): 1827–1835.