Projection and rescaling algorithm for finding most interior solutions to polyhedral conic systems

Javier Peña* Negar Soheili†

March 19, 2020

Abstract

We propose a simple *projection and rescaling algorithm* that finds *most interior* solutions to the pair of feasibility problems

$$\text{find } x \in L \cap \mathbb{R}^n_+ \quad \text{and} \quad \text{find } \hat{x} \in L^\perp \cap \mathbb{R}^n_+,$$

where L is a linear subspace of $\mathbb{R}^n$ and $L^\perp$ is its orthogonal complement. The algorithm complements a *basic procedure* that involves only projections onto L and $L^\perp$ with a periodic *rescaling step*. The number of rescaling steps and thus overall computational work performed by the algorithm are bounded above in terms of a condition measure of the above pair of problems.

Our algorithm is a natural but significant extension of a previous projection and rescaling algorithm that finds a solution to the problem

$$\text{find } x \in L \cap \mathbb{R}^n_{++}$$

when this problem is feasible. As a byproduct of our new developments, we obtain a sharper analysis of the projection and rescaling algorithm in the latter special case.

Keywords: Projection, rescaling, conic systems, condition measures, duality.

1 Introduction

The projection and rescaling algorithm [18, 19] is a recently developed method to solve the feasibility problem

$$\text{find } x \in L \cap \mathbb{R}^n_{++}, \tag{1}$$

where $L \subseteq \mathbb{R}^n$ is a linear subspace. The gist of this algorithm is to combine two main steps, namely a *basic procedure* and a *rescaling step*. The basic procedure either finds a solution to (1) if this problem is *well-conditioned*, or determines a rescaling step that improves the *conditioning* of problem (1). In the latter case a rescaling step is

*Tepper School of Business, Carnegie Mellon University, USA, jfp@andrew.cmu.edu
†College of Business Administration, University of Illinois at Chicago, USA, nazad@uic.edu

performed and the basic procedure is invoked again. If $L \cap \mathbb{R}^n_{++} \neq \emptyset$ then this kind of iterative basic-procedure and rescaling-step scheme succeeds in finding a solution to (1) because the rescaling step eventually yields a sufficiently well-conditioned problem that the basic procedure can solve [18, Theorem 1]. The conditioning of problem (1) is determined by a suitable measure of the *most interior* points in $L \cap \mathbb{R}^n_{++}$.

The projection and rescaling algorithm [18] was largely inspired by Chubanov's work [5, 6] who developed this algorithm when L is of the form $L = \{x \in \mathbb{R}^n : Ax = 0\}$ for a matrix $A \in \mathbb{R}^{m \times n}$. The projection and rescaling algorithm is in the same spirit as a number of articles based on the principle of enhancing a simple procedure with some sort of periodic reconditioning step [1, 2, 9, 7, 8, 11, 14, 15, 16, 17, 19, 20].

The projection and rescaling algorithm described in [18, Algorithm 1] successfully terminates only when $L \cap \mathbb{R}^n_{++} \neq \emptyset$. As it is described in [18, Algorithm 2], the algorithm has a straightforward extension that terminates with a solution to (1) or to its strict alternative problem

$$\text{find } \hat{x} \in L^\perp \cap \mathbb{R}^n_{++}, \tag{2}$$

where $L^\perp := \{s : \langle s, x \rangle = 0 \text{ for all } x \in L\}$, provided that (1) or (2) is feasible. In other words, the projection and rescaling algorithm [18, Algorithm 2] solves the *full support problem* in the terminology of Dadush, Vegh, and Zambelli [8]. We note that $L^\perp = \{A^{\mathrm{T}} y : y \in \mathbb{R}^m\}$ when $L = \{x \in \mathbb{R}^n : Ax = 0\}$ for $A \in \mathbb{R}^{m \times n}$. In this case the problems (1) and (2) are respectively called *full-support kernel problem* and *full-support image problem* in the terminology of Dadush, Vegh, and Zambelli [8].

The article [19] demonstrates the computational effectiveness of the projection and rescaling algorithm [18, Algorithm 2]. For computational purposes, the implementation in [19] is a tweaked version of [18, Algorithm 2] that applies to the two alternative problems

$$\text{find } x \in L \cap \mathbb{R}^n_+, \quad \text{and} \quad \text{find } \hat{x} \in L^\perp \cap \mathbb{R}^n_+ \tag{3}$$

without any a priori full-support assumptions. The implementation in [19] aims to find *most interior solutions* to the problems in (3), that is, points in the relative interiors of $L \cap \mathbb{R}^n_+$ and $L^\perp \cap \mathbb{R}^n_+$. In the terminology of Dadush, Vegh, and Zambelli [8], the algorithm described in [19, Algorithm 1] aims to find *maximum support* solutions to the above two problems.

The computational results in [19] demonstrate that [19, Algorithm 1] indeed succeeds in solving the maximum support problem. However, the correctness of [19, Algorithm 1] was formally shown only in the special case when either (1) or (2) was feasible. The main goal of this article is to give a formal proof of the correctness of a variant of [19, Algorithm 1] that finds maximum support solutions to the pair of problems in (3) in full generality. Our work is related to the maximum support algorithms described in [8, Section 4]. However there are several major differences. First, our algorithm treats both problems $x \in L \cap \mathbb{R}^n_+$ and $\hat{x} \in L^\perp \cap \mathbb{R}^n_+$ jointly and in completely symmetric fashion. Indeed, a key ingredient of our algorithm is the duality between these two problems. Second, our algorithm is a natural extension of the projection and rescaling algorithm in [18] and inherits most of its simplicity. Third, our results are stated entirely in the real model of computation. Unlike [8], we do not require the subspace L to be of the form $\{x \in \mathbb{R}^n : Ax = 0\}$ for some $A \in \mathbb{Z}^{m \times n}$. Since our results apply to real data, they have no dependence at all on any bit-length encoding of the subspace L. Instead, we show that the running time

of our algorithm depends on suitable *condition measures* $\sigma(L)$ and $\sigma(L^\perp)$ of the relative interiors of $L \cap \mathbb{R}^n_+$ and $L^\perp \cap \mathbb{R}^n_+$ respectively. The condition measures $\sigma(L)$ and $\sigma(L^\perp)$ can be seen as an extension and refinement of the condition measure $\delta(L)$ for $L \cap \mathbb{R}^n_{++}$ proposed in [18]. Fourth, although the analysis of our algorithm depends on the condition measures $\sigma(L)$ and $\sigma(L^\perp)$, the algorithm does not require knowledge of them. In short, our work settles the main open questions stated by Dadush, Vegh, and Zambelli [8, Section 5].

Our approach also yields, as a nice byproduct, a sharper analysis of the original projection and rescaling algorithm [18, Algorithm 1] for the full-support case, that is, when $L \cap \mathbb{R}^n_{++} \neq \emptyset$ or $L^\perp \cap \mathbb{R}^n_{++} \neq \emptyset$. Furthermore, for the full-support case we compare the performance of our projection and rescaling algorithm and its condition-based analysis with the performance of the previous rescaling algorithms and their condition-based analyses described in [1, 8, 9, 17]. The performance of the latter algorithms is stated in terms of a different condition measure $\rho(A)$ for the problems (1) and (2) that depends on a matrix $A \in \mathbb{R}^{m\times n}$ such that $L = \{x \in \mathbb{R}^n : Ax = 0\}$ or equivalently $L^\perp = \{A^{\mathrm{T}}y : y \in \mathbb{R}^m\}$. Thus our comparison concentrates on how the condition measures $\sigma(L)$ and $\sigma(L^\perp)$ used in this paper and the condition measure $\rho(A)$ used in [1, 8, 9, 17] relate to each other. We show that the measures $\sigma(L)$ and $\sigma(L^\perp)$ are less conservative, and possibly far less so, than $\rho(A)$. Consequently, our projection and rescaling algorithm applied to the full-support case has stronger condition-based convergence properties than the previous ones in [1, 8, 9, 17].

The remaining sections of the paper are organized as follows. Section 2 presents our main developments. This section details our approach to finding most interior solutions to (3). The approach hinges on three key ideas. First, we propose an algorithm that finds a point in a set of the form $L \cap \mathbb{R}^n_+$ that may not necessarily have maximum support (see Algorithm 1). Second, by relying on the first algorithm and on a natural duality between the two problems in (3), we propose a second algorithm that finds maximum support solutions to (3) (see Algorithm 2). Third, the analyses and to some extent the design of our algorithms rely on suitable refinements of condition measures previously proposed and used in [18, 22] (see Proposition 1 and Theorem 1). Section 3 shows how the new developments in Section 2 automatically yield a sharpening of the analysis previously performed in [18] for the projection and rescaling algorithm in the full support case. Section 3 also shows that the condition measure $\rho(A)$ used in [1, 8, 9, 17] for other rescaling algorithms is more conservative, and possibly far more so, than $\sigma(L)$ and $\sigma(L^\perp)$. Finally Section 4 details an implementation of the basic procedure, which is a central building block of our projection and rescaling algorithm. We limit our exposition to the most efficient known implementation of the basic procedure, namely a smooth perceptron scheme [18, 19]. The exposition in Section 4 highlights a simple and insightful but somewhat overlooked duality property that underlies the first-order implementations of the basic procedure described in [18, 19].

2 Partial support and maximum support solutions

We develop a two-step approach to finding maximum support solutions to (3). The first step is Algorithm 1 which finds a point $x \in L \cap \mathbb{R}^n_+$ whose support may or may not

be maximum. By leveraging Algorithm 1 and the duality between the two problems in (3), Algorithm 2 finds maximum support solutions $x \in L \cap \mathbb{R}^n_+$ and $\hat{x} \in L^\perp \cap \mathbb{R}^n_+$. Our developments rely on several key constructions and pieces of notation detailed next.

For $x \in \mathbb{R}^n_+$, the *support* of x is the set $\{j \in \{1, \ldots, n\} : x_j > 0\}$. Suppose $L \subseteq \mathbb{R}^n$ is a linear subspace. Let $J(L)$ be the following *maximum support* index set

$$J(L) := \left\{ j \in \{1, \ldots, n\} : x_j > 0 \text{ for some } x \in L \cap \mathbb{R}^n_+ \right\}.$$

In other words, $J(L)$ is the support of any point in the relative interior of $L \cap \mathbb{R}^n_+$ or equivalently the maximum support of all $x \in L \cap \mathbb{R}^n_+$. It is evident that $J(L) = \{1, \ldots, n\} \Leftrightarrow L \cap \mathbb{R}^n_{++} \neq \emptyset$ and $J(L) = \emptyset \Leftrightarrow L \cap \mathbb{R}^n_+ = \{0\} \Leftrightarrow L^\perp \cap \mathbb{R}^n_{++} \neq \emptyset$. As we formalize in the sequel, the difficulty of correctly identifying $J(L)$ and thus that of finding a maximum support solution to $x \in L \cap \mathbb{R}^n_+$ is determined by the following condition measure which is a variant of a condition measure proposed by Ye [22]:

$$\sigma(L) := \min_{j \in J(L)} \max\{x_j : x \in L \cap \mathbb{R}^n_+, \|x\|_\infty \leq 1\}.$$

The construction of $\sigma(L)$ implies that $\sigma(L) \in (0, 1]$ whenever $J(L) \neq \emptyset$. For convenience we let $\sigma(L) := 1$ when $J(L) = \emptyset$.

It is easy to see, via a standard separation argument or theorem of the alternative, that the maximum support index sets $J(L)$ and $J(L^\perp)$ partition $\{1, \ldots, n\}$, that is,

$$J(L) \cap J(L^\perp) = \emptyset \text{ and } J(L) \cup J(L^\perp) = \{1, \ldots, n\}.$$

For $\sigma \in (0, 1]$ let

$$J_\sigma(L) = \{j \in \{1, \ldots, n\} : x_j \geq \sigma \text{ for some } x \in L \cap \mathbb{R}^n_+, \|x\|_\infty \leq 1\}.$$

From the construction of $\sigma(\cdot), J(\cdot)$, and $J_\sigma(\cdot)$ it follows that for $\sigma \in (0, 1]$

$$J_\sigma(L) \subseteq J(L) \text{ and } J_\sigma(L) = J(L) \text{ if and only if } \sigma \leq \sigma(L).$$

For $J \subseteq \{1, \ldots, n\}$ let $\mathbb{R}^J \subseteq \mathbb{R}^n$ denote the subspace $\{x \in \mathbb{R}^n : x_i = 0 \text{ for } i \notin J\}$. For nonempty $J \subseteq \{1, \ldots, n\}$ let $\Delta(J) \subseteq \mathbb{R}^J$ denote the set $\{x \in \mathbb{R}^J : x \geq 0, \|x\|_1 = 1\}$. Given a linear subspace $L \subseteq \mathbb{R}^n$ and $J \subseteq \{1, \ldots, n\}$, let $P_{L|J} : \mathbb{R}^J \to L \cap \mathbb{R}^J$ denote the orthogonal projection from $\mathbb{R}^J$ onto $L \cap \mathbb{R}^J$.

Similar to the projection and rescaling algorithm in [18, 19], Algorithm 1 consists primarily of two main steps, namely a *basic procedure* and a *rescaling phase.* Both of these steps are slight modifications of those in the original projection and rescaling algorithm [19, Algorithm 1]. Our algorithm attempts to find $J(L)$ by gradually identifying and trimming indices presumed not to be in $J(L)$. The trimming decision is based on whether the rescaling matrix has exceeded a certain predefined threshold determined by an educated guess σ of $\sigma(L)$. If the educated guess σ is too large, the algorithm may mistakingly trim indices in $J(L)$. Thus Algorithm 1 is only guaranteed to find a solution $x \in L \cap \mathbb{R}^n_+$ with *partial support,* that is, a solution whose support may not be maximum. When $\sigma \leq \sigma(L)$, Algorithm 1 correctly identifies the maximum support set $J(L)$ (see Proposition 1). Although it is evidently desirable to choose the educated guess σ deliberately small, this comes at a cost as Proposition 1 shows.

Algorithm 1 works as follows. Start with the support set $J = \{1, \ldots, n\}$, rescaling matrix $D = I$, and an educated guess $\sigma > 0$ of $\sigma(L)$. At this initial stage no indices are trimmed and the problem is not rescaled. At each main iteration the basic procedure either finds $u \in \Delta(J)$ such that $(P_{DL|J}u)_J > 0$ or finds $z \in \Delta(J)$ such $\|(P_{DL|J}z)^+\|_1 \leq \frac{1}{2}\|z\|_\infty$. When $(P_{DL|J}u)_J > 0$ the algorithm outputs the point $x := D^{-1}P_{DL|J}u \in L \cap \mathbb{R}^n_+$ with support J. When $\|(P_{DL|J}z)^+\|_1 \leq \frac{1}{2}\|z\|_\infty$ the algorithm updates the rescaling matrix D to improve the conditioning of $DL \cap \mathbb{R}^n_+$ and trims J if some entries in D exceed the threshold $1/\sigma$. The main difference between Algorithm 1 and the projection and rescaling algorithm in [18] is the support set J and its dynamic adjustment after each rescaling step.

Section 4 below describes a possible first-order implementation of the basic procedure that terminates in at most $\mathcal{O}(n^{1.5})$ low-cost iterations. Other first-order implementations are discussed in [18, 19] all of which terminate in at most $\mathcal{O}(n^3)$ low-cost iterations.

The following technical lemma formalizes how the rescaling step improves the conditioning of $L \cap \mathbb{R}^n_+$. To that end, we will rely on one additional piece of notation. Suppose $L \subseteq \mathbb{R}^n$ is a linear subspace and $i \in \{1, \ldots, n\}$. Let

$$\sigma_i(L) := \max\{x_i : x \in L \cap \mathbb{R}^n_+, \|x\|_\infty \leq 1\}.$$

Observe that $J(L) = \{i : \sigma_i(L) > 0\}$ and $\sigma(L) = \min_{i \in J(L)} \sigma_i(L)$. Thus the vector $\begin{bmatrix}\sigma_1(L) & \cdots & \sigma_n(L)\end{bmatrix}$ encodes both $J(L)$ and $\sigma(L)$ but is more informative about the conditioning of $L \cap \mathbb{R}^n_+$ than $J(L)$ and $\sigma(L)$.

We will use the following common notational convention: for $i \in \{1, \ldots, n\}$ let $e_i \in \mathbb{R}^n$ denote the vector whose i-th entry is one and all other entries are zero.

Lemma 1 *Let $L \subseteq \mathbb{R}^n$ be a linear subspace and $P : \mathbb{R}^n \to L$ be the orthogonal projection onto L. Suppose $z \in \mathbb{R}^n_+ \setminus \{0\}$ is such that $\|(Pz)^+\|_1 \leq \frac{1}{2}\|z\|_\infty = z_i$ for some $i \in \{1, \ldots, n\}$. Then*

$$x \in L \cap \mathbb{R}^n_+ \Rightarrow x_i \leq \frac{\|x\|_\infty}{2}. \tag{4}$$

In particular, for $D = I + e_i e_i^{\mathrm{T}}$ the rescaled subspace $DL \subseteq \mathbb{R}^n$ satisfies

$$\sigma_i(DL) = 2\sigma_i(L) \text{ and } \sigma_j(DL) = \sigma_j(L) \text{ for } j \neq i.$$

Proof: If $x \in L \cap \mathbb{R}^n_+$ then

$$0 \leq x_i z_i \leq \langle x, z\rangle = \langle Px, z\rangle = \langle x, Pz\rangle \leq \left\langle x, (Pz)^+\right\rangle \leq \|x\|_\infty \cdot \|(Pz)^+\|_1 \leq \frac{\|x\|_\infty \cdot z_i}{2}.$$

Thus (4) follows.

Next, (4) implies that $\{x \in L \cap \mathbb{R}^n_+ : \|x\|_\infty \leq 1\} = \{x \in L \cap \mathbb{R}^n_+ : \|Dx\|_\infty \leq 1\}$. Therefore for $j = 1, \ldots, n$

$$\begin{aligned}\sigma_j(DL) &= \max\{(Dx)_j : x \in L \cap \mathbb{R}^n_+, \|Dx\|_\infty \leq 1\} \\ &= \max\{(Dx)_j : x \in L \cap \mathbb{R}^n_+, \|x\|_\infty \leq 1\} \\ &= D_{jj} \max\{x_j : x \in L \cap \mathbb{R}^n_+, \|x\|_\infty \leq 1\}\end{aligned}$$

$$= D_{jj}\sigma_j(L).$$

That is, $\sigma_i(DL) = 2\sigma_i(L)$ and $\sigma_j(DL) = \sigma_j(L)$ for $j \neq i$. ■

Algorithm 1 Partial support

1 (**Initialization**)
Let $D := I$, $J := \{1, \ldots, n\}$, and $\sigma > 0$ be an educated guess of $\sigma(L)$.
2 Let $P := P_{DL|J}$
3 (**Basic Procedure**)
Find either $u \in \Delta(J)$ such that $(Pu)_J > 0$ or
$z \in \Delta(J)$ such that $\|(Pz)^+\|_1 \leq \frac{1}{2}\|z\|_\infty$.
4 **If** $(Pu)_J > 0$ **then** HALT and output $x = D^{-1}Pu$ and J
5 **Else (Rescale** L **&** **Trim** J**)**
let $i := \mathsf{argmax}_j\, z_j$ and $D := (I + e_i e_i^{\mathrm{T}})D$
if $D_{ii} > 1/\sigma$ **then** let $J = J \setminus \{i\}$
if $J = \emptyset$ **then** HALT and output $x = 0$ and $J = \emptyset$
Go back to step 2

Proposition 1 *Suppose $\sigma \in (0, 1)$. Then Algorithm 1 finds $x \in L \cap \mathbb{R}^n_+$ such that $x_J > 0$ for some $J_\sigma(L) \subseteq J \subseteq J(L)$ in at most*

$$\sum_{i \in J_\sigma(L)} \lceil \log(1/\sigma_i(L)) \rceil + (n - |J_\sigma(L)|)\, \lceil \log_2(1/\sigma) \rceil \leq n \lceil \log_2(1/\sigma) \rceil \tag{5}$$

rescaling steps. Furthermore $J = J(L)$ if $\sigma \leq \sigma(L)$.

Proof: First, observe that the algorithm must eventually halt since each entry D_{ii} can be rescaled only up to $\lceil \log(1/\sigma) \rceil$ times before i is trimmed from J. Lemma 1 implies that throughout the algorithm

$$\sigma_i(DL) = D_{ii} \cdot \sigma_i(L)$$

for $i = 1, \ldots, n$. In particular, $D_{ii} \leq 1/\sigma_i(L)$ for $i \in J(L)$ because $\sigma_i(DL) \leq 1$. On the other hand, the trimming operation implies that $\log(D_{ii}) \leq \lceil \log(1/\sigma) \rceil$ for $i = 1, \ldots, n$. Hence when the algorithm halts, the total number of rescaling steps that the algorithm has performed is

$$\begin{aligned}
\sum_{i=1}^{n} \log_2(D_{ii}) &= \sum_{i \in J_\sigma(L)} \log_2(D_{ii}) + \sum_{i \notin J_\sigma(L)} \log_2(D_{ii}) \\
&\leq \sum_{i \in J_\sigma(L)} \lceil \log_2(1/\sigma_i(L)) \rceil + (n - |J_\sigma(L)|)\, \lceil \log_2(1/\sigma) \rceil \\
&\leq n \lceil \log_2(1/\sigma) \rceil.
\end{aligned}$$

If the algorithm terminates with $J = \emptyset$ then the output solution $x = 0 \in L \cap \mathbb{R}^n_+$ vacuously satisfies $x_J > 0$. Otherwise the algorithm outputs $x = D^{-1}Pu$ for $Pu \in DL \cap \mathbb{R}^J$ and $(Pu)_J > 0$. Therefore the output solution x satisfies $x = D^{-1}Pu \in L \cap \mathbb{R}^n_+$ and $x_J > 0$.

We next show that upon termination $J_\sigma(L) \subseteq J \subseteq J(L)$. Indeed, since $D_{ii} \leq 1/\sigma_i(L)$ for $i \in J(L)$, we have $D_{ii} \leq 1/\sigma_i(L) \leq 1/\sigma$ for $i \in J_\sigma(L)$ and thus the algorithm never trims any indices in $J_\sigma(L)$. Thus $J_\sigma(L) \subseteq J$ upon termination. On the other hand, upon termination $J \subseteq J(L)$ since the algorithm outputs some $x \in L \cap \mathbb{R}^n_+$ with $x_J > 0$. Finally, if $\sigma \leq \sigma(L)$ then $J(L) = J_\sigma(L) \subseteq J \subseteq J(L)$ and so $J = J(L)$. ■

Since $\sigma(L)$ is typically unknown, the initial guess σ could be larger than $\sigma(L)$. Hence, Algorithm 1 may not identify $J(L)$ and its corresponding maximum support point correctly. To overcome this obstacle, we apply Algorithm 1 to both $L \cap \mathbb{R}^n_+$ and $L^\perp \cap \mathbb{R}^n_+$ simultaneously and rely on the observation formalized in Corollary 1 which provides a natural stopping criterion.

Given a linear subspace $L \subseteq \mathbb{R}^n$ and σ, let $(x, J) := \mathcal{J}(L, \sigma)$ denote the output of Algorithm 1 when called with input (L, σ). The following result readily follows.

Corollary 1 *Let $L \subseteq \mathbb{R}^n$ be a linear subspace and $\sigma > 0$. If $(x, J) = \mathcal{J}(L, \sigma)$ and $(\hat{x}, \hat{J}) = \mathcal{J}(L^\perp, \sigma)$ satisfy $J \cup \hat{J} = \{1, \ldots, n\}$ then $J = J(L), \hat{J} = J(L^\perp)$, and $x, \hat{x}$ are maximum support points in $L \cap \mathbb{R}^n_+$ and $L^\perp \cap \mathbb{R}^n_+$ respectively.*

Proof: Proposition 1 implies that $x \in L \cap \mathbb{R}^n_+$, $\hat{x} \in L^\perp \cap \mathbb{R}^n_+$ with $x_J > 0$, $\hat{x}_{\hat{J}} > 0$ and $J_\sigma(L) \subseteq J \subseteq J(L)$, $J_\sigma(L^\perp) \subseteq \hat{J} \subseteq J(L^\perp)$. Since the index sets $J(L)$ and $J(L^\perp)$ partition $\{1, \ldots, n\}$, the identity $J \cup \hat{J} = \{1, \ldots, n\}$ can only occur when $J = J(L)$ and $\hat{J} = J(L^\perp)$. ■

Corollary 1 naturally suggests the following iterative strategy to find maximum support solutions to (3). Start with an initial guess σ of $\min\{\sigma(L), \sigma(L^\perp)\}$ and let $(x, J) := \mathcal{J}(L, \sigma)$ and $(\hat{x}, \hat{J}) := \mathcal{J}(L^\perp, \sigma)$. If $J \cup \hat{J} = \{1, \ldots, n\}$ then Corollary 1 implies that we found maximum support solutions to (3). Otherwise, reduce σ and repeat. Algorithm 2 formally describes the above strategy. Theorem 1 shows that if σ is reduced by a factor of two each time, then this iterative succeeds after at most $\lceil \log_2(\sigma_0 / \min\{\sigma(L), \sigma(L^\perp)\}) \rceil$ iterations where σ_0 is the initial guess of $\min\{\sigma(L), \sigma(L^\perp)\}$.

Algorithm 2 Maximum support

1 Let $\sigma := \sigma_0$ for an initial educated guess $\sigma_0 > 0$ of $\min\{\sigma(L), \sigma(L^\perp)\}$.
2 Let $(x, J) := \mathcal{J}(L, \sigma)$ and $(\hat{x}, \hat{J}) := \mathcal{J}(L^\perp, \sigma)$
3 **If** $J \cup \hat{J} = \{1, \ldots, n\}$ **then** HALT
4 **Else** (scale down σ)
  let $\sigma := \sigma/2$
  Go back to step 2.

Theorem 1 *Upon termination Algorithm 2 correctly identifies $J = J(L)$, $\hat{J} = J(L^\perp)$ and finds $x \in L \cap \mathbb{R}^n_+$, $\hat{x} \in L^\perp \cap \mathbb{R}^n_+$ with $x_J > 0, \hat{x}_{\hat{J}} > 0$. The algorithm terminates after at most $\lceil \log_2(\sigma_0 / \min\{\sigma(L), \sigma(L^\perp)\}) \rceil$ main iterations. Furthermore, the total number of rescaling steps performed by Algorithm 2 is bounded above by*

$$n \lceil \log_2(\sigma_0 / \min\{\sigma(L), \sigma(L^\perp)\}) \rceil \cdot \left(2 \lceil \log_2(1/\sigma_0) \rceil + \lfloor \log_2(\sigma_0 / \min\{\sigma(L), \sigma(L^\perp)\}) \rfloor \right).$$

In particular, if $\sigma_0 < \min\{\sigma(L), \sigma(L^\perp)\}$ *then the total number of rescaling steps in bounded above by*

$$2n\lceil \log_2(1/\sigma_0)\rceil.$$

On the other hand, if $\sigma_0 \geq \min\{\sigma(L), \sigma(L^\perp)\}$ *then the total number of rescaling steps is bounded above by*

$$2n\lceil \log_2(\sigma_0/\min\{\sigma(L), \sigma(L^\perp)\})\rceil \cdot \lceil \log_2(1/\min\{\sigma(L), \sigma(L^\perp)\})\rceil.$$

Proof: Proposition 1 implies that $J = J(L)$ and $\hat{J} = J(L^\perp)$ and thus $J \cup \hat{J} = \{1, \ldots, n\}$ when $\sigma \leq \min\left\{\sigma(L), \sigma(L^\perp)\right\}$ or possibly sooner. Collorary 1 hence implies that the solutions x and $\hat{x}$ returned by Algorithm 2 are maximum support solutions. Since σ is reduced by a factor of 2 at every iteration starting at σ_0, the number k of main iterations performed by Algorithm 2 is at most $\lceil \log_2\left(\sigma_0/\min\left\{\sigma(L), \sigma(L^\perp)\right\}\right)\rceil$. At each main iteration $i = 1, \ldots, k$ Algorithm 2 calls Algorithm 1 twice with input pairs $(L, \sigma_0/2^{i-1})$ and $(L^\perp, \sigma_0/2^{i-1})$. Proposition 1 implies that each of these calls terminates after at most $n\lceil \log_2(2^{i-1}/\sigma_0)\rceil = n \cdot (\lceil \log_2(1/\sigma_0)\rceil + i - 1)$ rescaling steps. Hence the total number of rescaling steps performed by Algorithm 2 is bounded above by

$$\begin{aligned}
&\sum_{i=1}^{k} 2n \cdot (\lceil \log_2(1/\sigma_0)\rceil + i - 1) \\
&\leq nk(2\lceil \log_2(1/\sigma_0)\rceil + k - 1) \\
&\leq n\lceil \log_2(\sigma_0/\min\{\sigma(L), \sigma(L^\perp)\})\rceil \cdot \left(2\lceil \log_2(1/\sigma_0)\rceil + \lfloor \log_2(\sigma_0/\min\{\sigma(L), \sigma(L^\perp)\})\rfloor\right).
\end{aligned}$$

■

A few comments on the choice of the educated guess σ_0 in Algorithm 2 are in order. Theorem 1 shows that the "ideal" choice is $\sigma_0 = \min\{\sigma(L), \sigma(L^\perp)\}$ since this choice gives the following best upper bound on the total number of rescaling steps

$$2n \cdot \lceil \log_2(1/\min\{\sigma(L), \sigma(L^\perp)\})\rceil.$$

However, this ideal choice requires knowledge of $\sigma(L), \sigma(L^\perp)$ which is unrealistic.

In contrast to the above ideal estimate, the crude estimate $\sigma_0 = 1$ gives the following upper bound that is only a factor of $\log_2(1/\min\{\sigma(L), \sigma(L^\perp)\}$ worse:

$$2n \cdot \lceil \log_2(1/\min\{\sigma(L), \sigma(L^\perp)\})\rceil^2.$$

Finally a lowball estimate $\sigma_0 \ll \min\{\sigma(L), \sigma(L^\perp)\}$ gives the upper bound $2n\lceil \log_2(1/\sigma_0)\rceil$ which is worse than that of the crude estimate $\sigma_0 = 1$ only when σ_0 grossly underestimates $\min\{\sigma(L), \sigma(L^\perp)\}$. The bottom line is that Algorithm 2 has little dependence on the educated guess σ_0 unless σ_0 is unreasonably small.

We conclude this section with a bound on the total number of arithmetic operations required by Algorithm 2. We only give a loose bound since the interesting complexity bounds are already stated in Proposition 1 and Theorem 1. The bounds below can be sharpened via a more detailed and lengthly but not necessarily more insightful accounting of arithmetic operations. In particular, to keep our exposition simple, we state the bounds only in terms of the dimension n of the ambient space and ignore the potentially much lower dimension of L or $L^\perp$.

As we detail in Section 4, the smooth perceptron scheme for the basic procedure is guaranteed to terminate in $\mathcal{O}(n^{1.5})$ iterations. The most costly operation in each iteration of the smooth perceptron is a matrix-vector multiplication involving the projection matrix $P_{DL|J}$, that is, $\mathcal{O}(n^2)$ arithmetic operations. Thus the number of arithmetic operations required by each call to the basic procedure is bounded above by

$$\mathcal{O}(n^{3.5}).$$

The number of arithmetic operations required by the rescaling and trimming step (even if we computed the projection matrix from scratch) is dominated by $\mathcal{O}(n^{3.5})$. Therefore for the crude estimate $\sigma_0 = 1$, the total number of arithmetic operations required by Algorithm 2 is bounded above by

$$\mathcal{O}\left(n^{4.5} \cdot \lceil \log_2(1/\min\{\sigma(L), \sigma(L^\perp)\})\rceil^2\right).$$

3 Full support solutions redux

We next revisit the projection and rescaling algorithm in [18] for the full support problem. We also compare its condition-based performance to that of the methods in [1, 8, 9, 17]. Algorithm 3 describes the projection and rescaling algorithm in [18]. Observe that Algorithm 3 is the same as Algorithm 1 without trimming. Indeed, when $L \cap \mathbb{R}^n_{++} \neq \emptyset$ or equivalently $J(L) = \{1, \ldots, n\}$, Algorithm 3 does exactly the same as Algorithm 1 provided $\sigma < \sigma(L)$.

Algorithm 3 Full support

(**Initialization**)
Let $D := I$
Let $P := P_{DL}$
(**Basic Procedure**)
    Find either $u \in \Delta_{n-1}$ such that $Pu > 0$ or
    $z \in \Delta_{n-1}$ such that $\|(Pz)^+\|_1 \leq \frac{1}{2}\|z\|_\infty$
If $Pu > 0$ **then** HALT and output $x = D^{-1}Pu$
Else (Rescale L**)**
    let $i := \mathsf{argmax}_j\, z_j$ and $D := (I + e_i e_i^{\mathrm{T}})D$
    Go back to step 2

Theorem 1 in [18] shows that when $L \cap \mathbb{R}^n_{++} \neq \emptyset$ Algorithm 3 finds a full support solution in $L \cap \mathbb{R}^n_{++}$ in at most $\log_2(1/\delta(L))$ rescaling iterations where $\delta(L)$ is the following measure of the most interior solution to $L \cap \mathbb{R}^n_{++}$:

$$\delta(L) = \max\left\{\prod_{j=1}^n x_j : x \in L \cap \mathbb{R}^n_{++}, \|x\|_\infty \leq 1\right\},$$

It is easy to see that $\prod_{j=1}^n \sigma_j(L) \geq \delta(L)$. Proposition 2 shows the iteration bound

$\log_2(1/\delta(L))$ can be sharpened to

$$\sum_{j=1}^{n} \log_2(1/\sigma_j(L)) = \log_2 \left(\prod_{j=1}^{n} 1/\sigma_j(L) \right)$$

modulo some rounding.

Proposition 2 *If $L \cap \mathbb{R}^n_{++} \neq \emptyset$ then Algorithm 3 finds $x \in L \cap \mathbb{R}^n_{++}$ in at most*

$$\sum_{j=1}^{n} \lceil \log_2(1/\sigma_j(L)) \rceil \leq n \lceil \log(1/\sigma(L)) \rceil \tag{6}$$

rescaling steps.

Proof: This readily follows from Proposition 1 since Algorithm 3 is identical to Algorithm 1 applied to $\sigma < \sigma(L)$. Indeed, for this choice of σ we have $J_\sigma(L) = J(L) = \{1, \ldots, n\}$ and thus Algorithm 1 does not trim any indices and the first expression in (5) yields precisely the first expression in (6). ■

It is evident that Algorithm 3 terminates only when $L \cap \mathbb{R}^n_{++} \neq \emptyset$. Proceeding exactly as in [18, Algorithm 2], we can apply Algorithm 3 in parallel so that it terminates with either a solution in $L \cap \mathbb{R}^n_{++}$ or in $L^\perp \cap \mathbb{R}^n_{++}$ as long as one of them is nonempty in a number of rescaling iterations either bounded above by (6) when $L \cap \mathbb{R}^n_{++} \neq \emptyset$ or bounded above by

$$\sum_{j=1}^{n} \lceil \log_2(1/\sigma_j(L^\perp)) \rceil \leq n \lceil \log(1/\sigma(L^\perp)) \rceil$$

when $L^\perp \cap \mathbb{R}^n_{++} \neq \emptyset$.

It is natural to ask how our projection and rescaling algorithm and its condition-based analysis compares with other rescaling algorithms and their condition-based analyses such as those described in [1, 8, 9, 17]. The condition-based analyses in all of these previous articles applies only to the full support case and are stated in terms of a different condition measure $\rho(A)$ where $A \in \mathbb{R}^{m \times n}$ is such that $L = \{x \in \mathbb{R}^n : Ax = 0\}$ or equivalently $L^\perp = \{A^{\mathrm{T}} y : y \in \mathbb{R}^m\}$. Proposition 3 below shows the relationship between the condition measures $\sigma(L), \sigma(L^\perp)$ and $\rho(A)$. We should note that the bounds in Proposition 3 are in the same spirit and similar to some results in [10].

Proposition 3 shows that the condition measure $\rho(A)$ is more conservative, and possibly far more so, than $\sigma(L)$ and $\sigma(L^\perp)$. In particular, any algorithm whose condition-based analysis is stated in terms of $\sigma(L)$ is automatically stronger, and possibly vastly so, than any other algorithm whose condition-based analysis is stated in terms of $\rho(A)$ as far as the dependence on the condition measure goes. Said differently, the projection and rescaling algorithm for (3) described in this paper as well as its predecessor [18, Algorithm 2] applied to the full-support case have stronger condition-based convergence properties than those in [1, 8, 9, 17].

The condition measure $\rho(A)$ is defined as follows. Suppose $A \in \mathbb{R}^{m \times n}$ is a full row-rank matrix whose columns have Euclidean norm equal to one. That is,

$$A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} \in \mathbb{R}^{m \times n} \text{ with } \|a_i\|_2 = 1 \text{ for } i = 1, \ldots, n.$$

The condition measure $\rho(A)$ is defined as follows

$$\rho(A) := \max_{\|y\|_2=1} \min_{i=1,\ldots,n} \langle a_i, y\rangle .$$

The condition measure $\rho(A)$ has an interesting history in optimization as discussed in [3, 4, 9, 10, 12, 21]. Among other features, it has the following nice geometric interpretation. When $\rho(A) > 0$, the quantity $\rho(A)$ can be interpreted as a measure of *thickness* of the cone $K := \{y \in \mathbb{R}^m : A^{\mathrm{T}} y \leq 0\} \subseteq \mathbb{R}^m$. Indeed, in this case $\rho(A)$ is the radius of the largest ball center at a point of Euclidean norm one and contained in K. Furthermore, $\rho(A) > 0$ if and only if $0 \notin \{Ax : x \in \Delta_{n-1}\}$ and $\rho(A) < 0$ if and only if $0 \in \mathrm{int}(\{Ax : x \in \Delta_{n-1}\})$, where $\Delta_{n-1} := \{x \in \mathbb{R}^n_+ : \|x\|_1 = 1\}$. We note that the set $\{Ax : x \in \Delta_{n-1}\}$ is precisely the convex hall of the columns of A. Regardless of the sign of $\rho(A)$ its absolute value $|\rho(A)|$ is precisely the distance from 0 to the boundary of $\{Ax : x \in \Delta_{n-1}\}$.

Let $L = \{x \in \mathbb{R}^n : Ax = 0\}$ or equivalently $L^\perp = \{A^{\mathrm{T}} y : y \in \mathbb{R}^m\}$. It is easy to see that $\rho(A) > 0 \Leftrightarrow L^\perp \cap \mathbb{R}^n_{++} \neq \emptyset$ and $\rho(A) < 0 \Leftrightarrow L \cap \mathbb{R}^n_{++} \neq \emptyset$. The following result refines these equivalences in terms of the condition measures $\sigma(L), \sigma(L^\perp)$.

Proposition 3 *Suppose $A \in \mathbb{R}^{m\times n}$ is a full row-rank matrix whose columns have Euclidean norm equal to one. Let $L = \{x \in \mathbb{R}^n : Ax = 0\}$ or equivalently $L^\perp = \{A^{\mathrm{T}} y : y \in \mathbb{R}^m\}$.*

(a) *If $\rho(A) > 0$ then $\rho(A) \leq \sigma(L^\perp)$. Furthermore, $\sigma(L^\perp)$ can be arbitrarily larger than $\rho(A)$.*

(b) *If $\rho(A) < 0$ then $|\rho(A)| \leq \sigma(L)$. Furthermore, $\sigma(L)$ can be arbitrarily larger than $|\rho(A)|$.*

Proof:

(a) Let $\bar{y} \in \mathbb{R}^m$ be such that $\|\bar{y}\|_2 = 1$ and $\rho(A) = \min_{i=1,\ldots,n} \langle a_i, \bar{y}\rangle > 0$. Then $\bar{x} := A^{\mathrm{T}} \bar{y} \in L^\perp$ and for each $i = 1, \ldots, n$ we have $\bar{x}_i = \langle a_i, \bar{y}\rangle \geq \rho(A) > 0$ and $\bar{x}_i \leq \|a_i\|_2 \cdot \|\bar{y}\|_2 \leq 1$. In other words, $\bar{x} \in L \cap \mathbb{R}^n_{++}$, $\|\bar{x}\|_\infty \leq 1$, and $\bar{x}_i \geq \rho(A)$ for each $i = 1, \ldots, n$. Thus $\sigma(L^\perp) \geq \rho(A)$. The following example shows that $\sigma(L^\perp)$ can be arbitrarily larger than $\rho(A)$. Let

$$A = \frac{1}{\sqrt{1+\epsilon^2}} \begin{bmatrix} 1 & 1 & -1 & -1 \\ \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix}$$

where $0 < \epsilon < 1$. It is easy to see that $\rho(A) = \epsilon/\sqrt{1+\epsilon^2}$ and $\sigma(L^\perp) = 1$.

(b) In this case we have $\min_{i=1,\ldots,n} \langle a_i, y\rangle \leq \rho(A) < 0$ for all $y \in \mathbb{R}^m$ with $\|y\|_2 = 1$. Thus for all $v \in \mathbb{R}^m$ with $\|v\|_2 \leq |\rho(A)|$ we have

$$\min_{y\in\mathbb{R}^m\setminus\{0\}} \min_{x\in\Delta_{n-1}} \langle Ax + v, y\rangle \leq 0.$$

It thus follows, via a standard separation argument, that $v \in \{Ax : x \in \Delta_{n-1}\}$ for all $v \in \mathbb{R}^m$ with $\|v\|_2 \leq |\rho(A)|$. In particular, for each $i = 1, \ldots, n$ there exists $\bar{x} \in \Delta_{n-1}$ such that $A\bar{x} = -|\rho(A)|a_i$. Thus $\hat{x} := (\bar{x} + |\rho(A)|e_i)/(1 + |\rho(A)|)$ satisfies

$$A\hat{x} = 0,\ \hat{x} \in \Delta_{n-1},\ \hat{x}_i \geq |\rho(A)|.$$

Consequently $\hat{x} \in L \cap \mathbb{R}^n_{++}$, $\|\hat{x}\|_\infty \leq 1$, and $\hat{x}_i \geq |\rho(A)|$. Therefore $\sigma(L) \geq |\rho(A)|$. The following example shows that $\sigma(L)$ can be arbitrarily larger than $|\rho(A)|$. Let

$$A = \frac{1}{\sqrt{1+\epsilon^2}} \begin{bmatrix} 1 & 1 & -1 & -1 \\ \epsilon & -\epsilon & \epsilon & -\epsilon \end{bmatrix}$$

where $0 < \epsilon < 1$. It is easy to see that $\rho(A) = -\epsilon/\sqrt{1+\epsilon^2}$ and $\sigma(L) = 1$.

4 Basic procedure

This concluding section describes an implementation of the basic procedure, which is a key component of Algorithm 1. To simplify notation, we describe the basic procedure for the case when $J = \{1, \ldots, n\}$. The extension to any $J \subseteq \{1, \ldots, n\}$ is completely straightforward. Suppose $P \in \mathbb{R}^{n \times n}$ is the projection onto a linear subspace $L \subseteq \mathbb{R}^n$. The goal of the basic procedure is to find $z \in \Delta_{n-1} := \{x \in \mathbb{R}^n : x \geq 0, \|x\|_1 = 1\}$ such that either $Pz > 0$ or $\|(Pz)^+\|_1 \leq \frac{1}{2}\|z\|_\infty$. To that end, consider the problem

$$\min_{z \in \Delta_{n-1}} \frac{1}{2}\|Pz\|_2^2 \tag{7}$$

and its dual

$$\max_{y \in \mathbb{R}^n} \left\{ -\frac{1}{2}\|y\|_2^2 + \min_{z \in \Delta_{n-1}} \langle Py, z \rangle \right\} \Leftrightarrow \max_{u \in \Delta_{n-1}} \left\{ -\frac{1}{2}\|Pu\|_2^2 + \min_{z \in \Delta_{n-1}} \langle Pu, z \rangle \right\}.$$

The articles [18, 19] describe several first-order schemes for (7) that achieve the goal of the basic procedure. All of these algorithms generate sequences $z_k, u_k \in \Delta_{n-1}$ satisfying

$$\frac{1}{2}\|Pz_k\|_2^2 + \frac{1}{2}\|Pu_k\|_2^2 - \min_{z \in \Delta_{n-1}} \langle Pu_k, z \rangle \leq \mu_k \tag{8}$$

for $\mu_k \to 0$. The above property of first-order schemes is not explicitly stated in [18, 19] but it can be easily inferred as shown in the recent paper [13].

From (8) it follows that as long as $Pu_k \not> 0$ we must have $\frac{1}{2}\|Pz_k\|_2^2 \leq \mu_k$. The latter in turn implies that

$$\|(Pz_k)_+\|_1 \leq \sqrt{n}\|Pz_k\|_2 \leq \sqrt{2n\mu_k} \leq n\sqrt{2n\mu_k}\|z\|_\infty$$

and thus the basic procedure terminates when $\mu_k \leq \frac{1}{8n^3}$. Algorithm 4 describes the *smooth perceptron* basic procedure which generates iterates $u_k, z_k \in \Delta_{n-1}$ satisfying (8) with $\mu_k = \frac{8}{(k+1)^2}$ and thus is guaranteed to terminate in at most $k = \mathcal{O}(n^{1.5})$ iterations. This is both theoretically and computationally the fastest of the first-order schemes for the basic procedure proposed in [18, 19]. Algorithm 4 relies on the mapping $u_\mu : \mathbb{R}^n \to \Delta_{n-1}$ defined as follows. Let $\bar{u} \in \Delta_{n-1}$ be fixed and $\mu > 0$. Let

$$u_\mu(v) := \operatorname*{argmin}_{u \in \Delta_{n-1}} \left\{ \langle u, v \rangle + \frac{\mu}{2}\|u - \bar{u}\|_2^2 \right\}.$$

Algorithm 4 Smooth Perceptron Scheme

let $u_0 := \bar{u}$; $\mu_0 = 2$; $z_0 := u_{\mu_0}(Pu_0)$; and $k := 0$
while $Pu_k \not> 0$ and $\|(Pz_k)^+\|_1 > \epsilon\|z_k\|_\infty$ **do**
  $\theta_k := \frac{2}{k+3}$
  $u_{k+1} := (1-\theta_k)(u_k + \theta_k z_k) + \theta_k^2 u_{\mu_k}(Pu_k)$
  $\mu_{k+1} := (1-\theta_k)\mu_k$
  $z_{k+1} := (1-\theta_k)z_k + \theta_k u_{\mu_{k+1}}(Pu_{k+1})$
  $k := k+1$
end while

References

[1] A. Belloni, R. Freund, and S. Vempala. An efficient rescaled perceptron algorithm for conic systems. *Math. of Oper. Res.*, 34(3):621–641, 2009.

[2] U. Betke. Relaxation, new combinatorial and polynomial algorithms for the linear feasibility problem. *Discrete & Computational Geometry*, 32:317–338, 2004.

[3] P. Bürgisser and F. Cucker. *Condition*. Springer Berlin Heidelberg, 2013.

[4] D. Cheung and F. Cucker. A new condition number for linear programming. *Math. Prog.*, 91(2):163–174, 2001.

[5] S. Chubanov. A strongly polynomial algorithm for linear systems having a binary solution. *Math. Program.*, 134:533–570, 2012.

[6] S. Chubanov. A polynomial projection algorithm for linear feasibility problems. *Math. Program.*, 153:687–713, 2015.

[7] D. Dadush, L. A Végh, and G. Zambelli. Rescaled coordinate descent methods for linear programming. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 26–37. Springer, 2016.

[8] D. Dadush, L. A Végh, and G. Zambelli. Rescaling algorithms for linear conic feasibility. *To Appear in Math. of Oper. Res.*, 2020.

[9] J. Dunagan and S. Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. *Math. Program.*, 114(1):101–114, 2006.

[10] M. Epelman and R. M. Freund. Condition number complexity of an elementary algorithm for computing a reliable solution of a conic linear system. *Math. Program.*, 88(3):451–485, 2000.

[11] R. Freund, R. Roundy, and M. Todd. Identifying the set of always-active constraints in a system of linear inequalities by a single linear program. *Working Paper, Massachusetts Institute of Technology, Alfred P. Sloan School of Management*, 1985.

[12] J. Goffin. The relaxation method for solving systems of linear inequalities. *Math. Oper. Res.*, 5:388–414, 1980.

[13] D. Gutman and J. Peña. Perturbed Fenchel duality and first-order methods. *arXiv preprint arXiv:1812.10198*, 2018.

[14] R. Hoberg and T. Rothvoss. An improved deterministic rescaling for linear programming algorithms. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 267–278. Springer, 2017.

[15] T. Kitahara and T. Tsuchiya. An extension of Chubanov's polynomial-time linear programming algorithm to second-order cone programming. *Optimization Methods and Software*, 33(1):1–25, 2018.

[16] B. Lourenço, T. Kitahara, M. Muramatsu, and T. Tsuchiya. An extension of Chubanov's algorithm to symmetric cones. *Math. Program.*, pages 1–33, 2016.

[17] J. Peña and N. Soheili. A deterministic rescaled perceptron algorithm. *Math. Program.*, 155:497–510, 2016.

[18] J. Peña and N. Soheili. Solving conic systems via projection and rescaling. *Math. Program.*, 166:87–111, 2017.

[19] J. Peña and N. Soheili. Computational performance of a projection and rescaling algorithm. *Optimization Methods and Software*, pages 1–18, 2019.

[20] C. Roos. An improved version of Chubanov's method for solving a homogeneous feasibility problem. *Optimization Methods and Software*, 33:26–44, 2018.

[21] N. Soheili and J. Peña. A smooth perceptron algorithm. *SIAM J. on Optim.*, 22(2):728–737, 2012.

[22] Y. Ye. Toward probabilistic analysis of interior-point algorithms for linear programming. *Math. of Oper. Res.*, 19:38–52, 1994.