Projection Algorithms for Convex and Combinatorial Optimization

By

JAMIE HADDOCK

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Professor Jesús A. De Loera, Chair
University of California, Davis

Professor Albert Fannjiang
University of California, Davis

Professor Roland W. Freund
University of California, Davis

Professor Deanna Needell
University of California, Los Angeles

Committee in Charge

2018

To my family, whose examples have taught me strength and perseverence.

Table of Contents

List of Tables

List of Figures

Abstract

This thesis studies projection algorithms in optimization which have frequent applications in data science (e.g., image processing). Contributions in this thesis include proposing and analyzing novel iterative projection methods for the linear feasibility problem, and providing a worst-case analysis of Wolfe's combinatorial method for the minimum norm point problem. We further strengthen results connecting the linear feasibility and minimum norm point problems. In an appendix, we include MATLAB code for all methods described in this thesis.

Chapter 2 deals with both new and classical iterative projection methods for linear feasibility problems. We provide an accelerated convergence analysis of Motzkin's method on systems of linear equations which is governed by the *dynamic range* of the residual of the system. We give a probabilistic analysis of new randomized Kaczmarz methods for detecting corruption in systems of linear equations where the number of corruptions is small compared to the number of rows of the system. Finally, we propose a new, generalized family of algorithms that includes Motzkin's methods and the randomized Kaczmarz methods, the *Sampling Kaczmarz-Motzkin* methods. We provide an analysis of their convergence, prove they detect feasibility of the linear feasibility problems they solve, and show that these methods even terminate in a finite number of steps in special cases. We include ample experimental evidence comparing these methods to competing methods.

Chapter 3 studies Wolfe's methods for the minimum norm point problem. The complexity of Philip Wolfe's method for the minimum Euclidean-norm point problem over a convex polytope has remained unknown since he proposed the method in 1974. The method is important because it is used as a subroutine for one of the most practical algorithms for submodular function minimization (a topic with remarkable applications in machine learning). We discuss Wolfe's methods in detail, including discussing variations of the original method, and present several interesting examples of

polytopes on which Wolfe's methods exhibit differing behavior. We present the first example that Wolfe's method can take exponential time.

Chapter 4 presents results regarding the complexity of the linear feasibility and minimum norm point problems, and connects the two problems. We discuss the complexity of the minimum norm vertex problem over convex polytopes, a problem which is related to the minimum norm point problem but illustrates the potential difference in complexity between seemingly similar computational problems. Finally, we demonstrate that the linear feasibility problem reduces to the minimum norm point problem in strongly-polynomial time. We conclude the thesis with some final remarks and discussion of future work.

Acknowledgments

I have found support and encouragement from many people, far too many to list, over the course of my education and wish to take this opportunity to thank them.

First and foremost, I am extremely grateful to my advisor, Professor Jesús A. De Loera. Jesús's passion for mathematics is inspiring and infectious, and his guidance has been integral to my success in this endeavor. I look up to him as a gifted teacher, a brilliant researcher, and as a person who cares for and invests in his students, collaborators, and the greater culture of mathematics. He leaves each of his students impossibly large shoes to fill.

Prof. Albert Fannjiang has been encouraging and a great source of advice regarding my research and teaching experiences. I have appreciated his insightful comments on my research, which have led me to gain a better understanding and to identify interesting directions forward.

Prof. Roland W. Freund has challenged me to improve as a mathematician and to invest additional effort in my endeavors. His Numerical Analysis course was one of the most demanding I have taken, but also one of my favorites and one of the most influential for my research.

Prof. Deanna Needell has been a great role model, collaborator, and mentor throughout the time I have known her. Her guidance has already been integral to my career, and her patience and encouragement an inspiration to continue on in academia. I am grateful and extremely excited for the opportunity to continue under her mentorship in the coming years.

I am grateful to Prof. Dan Gusfield and Prof. Michael P. Friedlander, who served on my qualifying committee and whose courses were great sources of motivation for my work. Prof. Luis Rademacher has been a supportive collaborator, from whom I have learned a lot in only a little time.

I am additionally grateful to the UC Davis Mathematics Department as a whole; I have been blessed to work in a department with a wonderfully open culture which encourages students to grow as teachers and researchers. Prof. Becca Thomases and Prof. Tim Lewis contribute to that culture, and were highly influential in my pedagogical development. Prof. Dan Romik has been an incredible source of kindness, and I am grateful for his generosity to me and his great efforts within our department. Finally, I am grateful to our spectacular staff who are sources of assistance, encouragement, and friendship; I am especially indebted to Tina and Sarah.

I am grateful for my encouraging experience at Gonzaga University and for those who make the GU Honors Program possible, especially Father Tim Clancy. Prof. Paul De Palma gave me my first research opportunity, and inspired my interest in computer science. Prof. Kathie A. Yerion challenged me (and all her students); her courses were tough but the skills they developed continue to serve me well. I am grateful to the entire GU Mathematics Department; the faculty go above and beyond to educate their students. Prof. Thomas McKenzie provided the inspiration and guidance that led me to graduate school in mathematics; his mentorship is something I aspire to pay forward throughout my career.

I could not have completed my degree without the support of my friends. Amanda shows me how it is done, keeps me goofy, and is always there to celebrate or support me. Anna helped me get through the last year; I have been so grateful for her friendship, our collaboration, and all the fun. Kelly sprinkles my life with excitement; I am inspired by her vivacity. Tiyana has been generous and a truly loyal friend; I am grateful that she befriended me and that we've grown so close over the past five years. I am also thankful for Ben, Jordan, Emily, Adam, Michael, Lauren, Andrew, Kassi, Lily, and Rohit; I've been blessed to have their friendship.

Most importantly, I thank my family (especially Mom, Dad, Terry, Jenna, Linkin, Hudson, Brendan, Annie, Katie, and Alaia) for their unwavering support and love. My sister, Katie, is my hero, role model, and best friend; I wouldn't be who I am without her example of strength, vitality, and fun. Her confidence in my success has helped me to raise above my own doubts. My mom, Carol, is an endless source of selfless support. She has imparted on me a strong value for the people in my life and I hope to live up to her caring example. My dad, Jim, has had a vast impact on my

decision to teach; I hope to be half of the teacher he is. His belief in me has led me to continue to challenge myself. They inspire me to try harder at everything I do, but especially to try harder to help others.

Finally, I gratefully acknowledge support from the GAANN fellowship, the UC Davis Mathematics fellowship, the William K. Schwarze scholarship, the UC Davis Dissertation Year fellowship, NSA grant H98230-15-1-0226, NSF grant DMS-1522158, and NSF grant DMS-1440140 while I was in residence at the Mathematical Sciences Research Institute in Berkeley, California. I also thank the AMS, SIAM, UC Davis Mathematics, the Insitute for Mathematics and Its Applications, the Berlin Mathematical School, and the Hausdorff Research Institute for Mathematics for travel support.

CHAPTER 1

Introduction

Broadly speaking, the mathematical concept of projection deals with operators which minimize distances or norms. This broad class of operators encompasses many algorithms used in data science and optimization. This thesis deals with such projection algorithms and their applications. This chapter gives a summary of the entire thesis.

1.1. Background

Throughout this thesis, we study operators, $\mathcal{P}_K$, which map from a normed linear space, X, into a subset, K, of X and minimize a norm on X. We call these set-valued functions *projections onto K in $\|\cdot\|$*. The study of such operators is of interest in the case of infinite-dimensional Hilbert spaces (e.g., [**KKM09**, **KR12**]), although this will not be the focus of this thesis. Throughout this thesis, we discuss only operators which act on the normed linear space $\mathbb{R}^n$ and in later sections will further focus on those which minimize the Euclidean-norm. We begin with some common definitions and basic results on projections, convex analysis in $\mathbb{R}^n$, and optimization that we will refer to for the following sections and chapters; see [**Sch86**, **BV04**, **Lay82**, **Zie12**, **Bar02**, **HN01**, **CLO07**] for resources.

Definition 1.1.1. *A* projection onto *or* into K *is an operator $\mathcal{P}_K : \mathbb{R}^n \to K \subseteq \mathbb{R}^n$ which is defined as $\mathcal{P}_K(\mathbf{x}) = \mathbf{argmin}_{\mathbf{y}\in K}\|\mathbf{y} - \mathbf{x}\|$ where $\|\cdot\|$ denotes a norm on $\mathbb{R}^n$.*

Some authors define a projection as a linear map on a linear space, $\mathcal{P} : X \to X$, which satisfies $\mathcal{P}^2 = \mathcal{P}$ [**HN01**]. Note that we have relaxed the requirement that $\mathcal{P}_K$ be a linear operator. See Figure 1.1 for an example of the nonlinear behavior of norm minimization.

If the norm is an ℓ^p norm, we denote the projection operator as the ℓ^p-projection onto K. Geometrically, the ℓ^p-projection of the point $\mathbf{x}$ onto K may be interpreted as the first point(s) of intersection

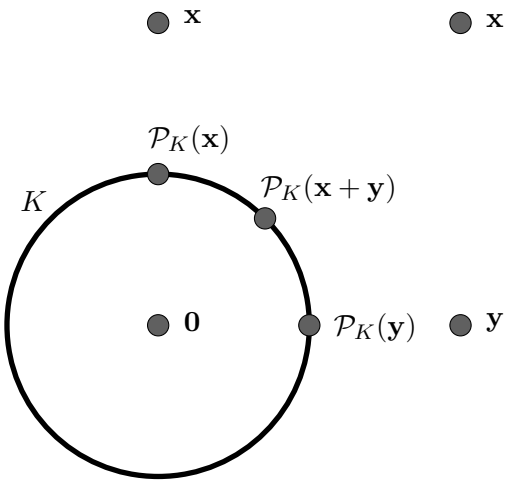

FIGURE 1.1. An example of the nonlinear behavior of ℓ^2-projection onto a sphere, $\mathcal{P}_K$. Here $\mathbf{x} = (0, 2)$, $\mathbf{y} = (2, 0)$, and K is the unit circle. Note that $\mathcal{P}_K(\mathbf{x} + \mathbf{y}) \neq \mathcal{P}_K(\mathbf{x}) + \mathcal{P}_K(\mathbf{y})$.

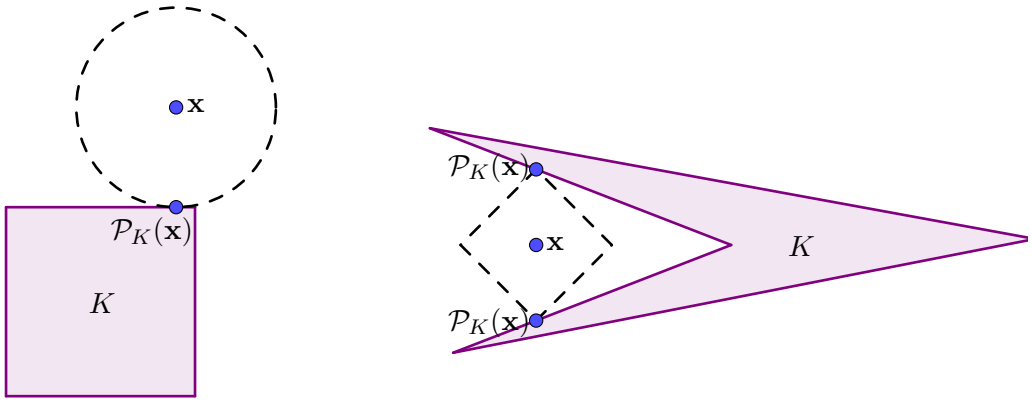

FIGURE 1.2. Left: ℓ^2-projection of $\mathbf{x}$ onto K; right: ℓ^1-projection of $\mathbf{x}$ onto K.

of the α-scaling of the ℓ^p-norm ball centered at $\mathbf{x}$, $\{\mathbf{y} : \|\mathbf{y} - \mathbf{x}\|_p = \alpha\}$, with the set K as α increases. Examples of this intuition are included in Figure 1.2.

As Figure 1.2 suggests, the projection onto K may contain multiple points if the set K is non-convex. The contrapositive is true if the norm is defined by the inner product on $\mathbb{R}^n$ (i.e., the ℓ^2-norm); if K is convex then $\mathcal{P}_K(\mathbf{x})$ for $\|\cdot\|_2$ is unique if nonempty.

Note that our definition of projection satisfies $\mathcal{P}_K^2 = \mathcal{P}_K$, since if $\mathcal{P}_K(\mathbf{x})$ is nonempty then $\mathcal{P}_K(\mathbf{x}) \subset K$ so $\mathcal{P}_K(\mathcal{P}_K(\mathbf{x})) = \mathcal{P}_K(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$. This simple fact about projections demonstrates one connection between fixed point theory and optimization. Many of the iterative methods used in optimization define iterates via a projection operator precisely because the underlying problem is solved by a fixed point, $\mathcal{P}_K(\mathbf{y}) = \mathbf{y}$; see [**Aub93**, **Tod13**, **RT17**, **MGC11**, **HYZ08**] and references therein for resources. Each of the algorithms analyzed in this thesis may be seen as fixed point operators. Each of the methods discussed in Chapter 2 may be considered instances of the following simple fixed-point operator which computes a point, $\mathbf{x}$, in the intersection of closed, convex sets, $K_i \neq \emptyset$ for $i = 1, 2, ..., m$; that is, $\mathbf{x} \in \cap_{i=1}^m K_i$. Define $I_{\mathbf{x}}$ to be the indices of sets in which $\mathbf{x}$ does not lie, $I_{\mathbf{x}} := \{j : \mathbf{x} \notin K_j\} \subseteq \{1, 2, ..., m\}$, and let $\mathcal{P}_{K_i}(\mathbf{x})$ be the ℓ^2-projection operator onto K_i. Define the fixed-point operator

$$T(\mathbf{x}) = \begin{cases} \mathcal{P}_{K_i}(\mathbf{x}) \text{ where } i \in I_{\mathbf{x}} & \text{if } I_{\mathbf{x}} \neq \emptyset \\ \mathbf{x} & \text{otherwise.} \end{cases} \tag{1.1}$$

The following proposition demonstrates that fixed points of the operator T are points in the feasible region, $\mathbf{x} \in \cap_{i=1}^m K_i$.

Proposition 1.1.2. *A point is feasible, $\mathbf{x} \in \cap_{i=1}^m K_i$, if and only if it is a fixed point of T (defined in (1.1)), $\mathbf{x} = T(\mathbf{x})$.*

PROOF. Note that $\mathbf{x} \in \cap_{i=1}^m K_i$ if and only if $I_{\mathbf{x}} = \emptyset$. Finally, $I_{\mathbf{x}} = \emptyset$ if and only if $T(\mathbf{x}) = \mathbf{x}$. □

The concept of projection is well-defined for all nonempty sets K and norms. However, the sets K onto which projection in a given norm may be explicitly computed are far more restricted. Even for simple sets, projection in norms (other than the ℓ^2-norm) are rarely computable by a closed formula. Additionally, ℓ^p-norm minimization over even simple sets is NP-hard for $0 \leq p < 1$ (see [**GJY11**] and the references therein), while for $p \geq 1$, the convexity of the norm allows for polynomial time approximation (or exact computation for $p = 1$ and $p = 2$). Throughout the rest of this thesis, $\|\cdot\|$ denotes the ℓ^2-norm and other norms will be explicitly noted.

We will denote the Euclidean distance of a point $\mathbf{x}$ to a set K to be

$$d(\mathbf{x}, K) := \inf_{\mathbf{y} \in K} \|x - y\|$$

where inf denotes the infimum. If the set K is empty, we consider $d(\mathbf{x}, K) = \infty$. Note that when the projection of $\mathbf{x}$ onto K exists, it achieves this minimal distance to $\mathbf{x}$. While the concept of projection is well-defined, it is also possible for the projection onto a set K to be empty (as no point in K achieves the infimal distance to $\mathbf{x}$); consider any nonempty, open set K and $\mathbf{x} \notin K$ and note that for any $\mathbf{y} \in K$ there exists $\mathbf{z} \in K$ so that $\|\mathbf{z} - \mathbf{x}\| < \|\mathbf{y} - \mathbf{x}\|$. It is likewise possible for $d(\mathbf{x}, K) = 0$ with $\mathbf{x} \notin K$; consider the previous situation with $\mathbf{x} \in \overline{K}$ where $\overline{K}$ denotes the closure of K and note that $d(\mathbf{x}, K) = 0$. Additionally, we consider the distance between two sets to be the infimal distance between points of each set,

$$d(S, K) := \inf_{\mathbf{x} \in S} d(\mathbf{x}, K)$$

where if S is empty this distance is defined to be $d(S, K) = \infty$. Note that it is possible for two nonempty, closed, convex sets S, K with $S \cap K = \emptyset$ to have $d(S, K) = 0$, but if at least one is compact then this distance must be nonzero.

We begin with one of the simplest examples of projection, the ℓ^2-projection onto an affine subspace of dimension $n - 1$. An affine subspace of dimension $n - 1$ is defined by an equation $\mathbf{a}_i^T \mathbf{x} = b_i$ and is a *hyperplane*, $H_{\mathbf{a}_i, b_i} := \{\mathbf{x} : \mathbf{a}_i^T \mathbf{x} = b_i\}$. The projection of an arbitrary point $\mathbf{x}$ onto $H_{\mathbf{a}_i, b_i}$ is easily computable, given by the closed formula

$$\mathcal{P}_{H_{\mathbf{a}_i, b_i}}(\mathbf{x}) = \mathbf{x} + \frac{b_i - \mathbf{a}_i^T \mathbf{x}}{\|\mathbf{a}_i\|^2} \mathbf{a}_i. \tag{1.2}$$

Note that this computation requires only $\mathcal{O}(n)$ algebraic operations. The projection moves x parallel to the vector normal to the hyperplane, $\mathbf{a}_i$, with displacement given by the norm of the second term of the right hand side of (1.2). See Figure 1.3 for an example of a projection of $\mathbf{x}$ into the hyperplane $H_{\mathbf{a}_i, b_i}$. One important fact about this computation is that if all data $(\mathbf{x}, \mathbf{a}_i, b_i)$ are rational then $\mathcal{P}_{H_{\mathbf{a}_i, b_i}}(\mathbf{x})$ will be rational.

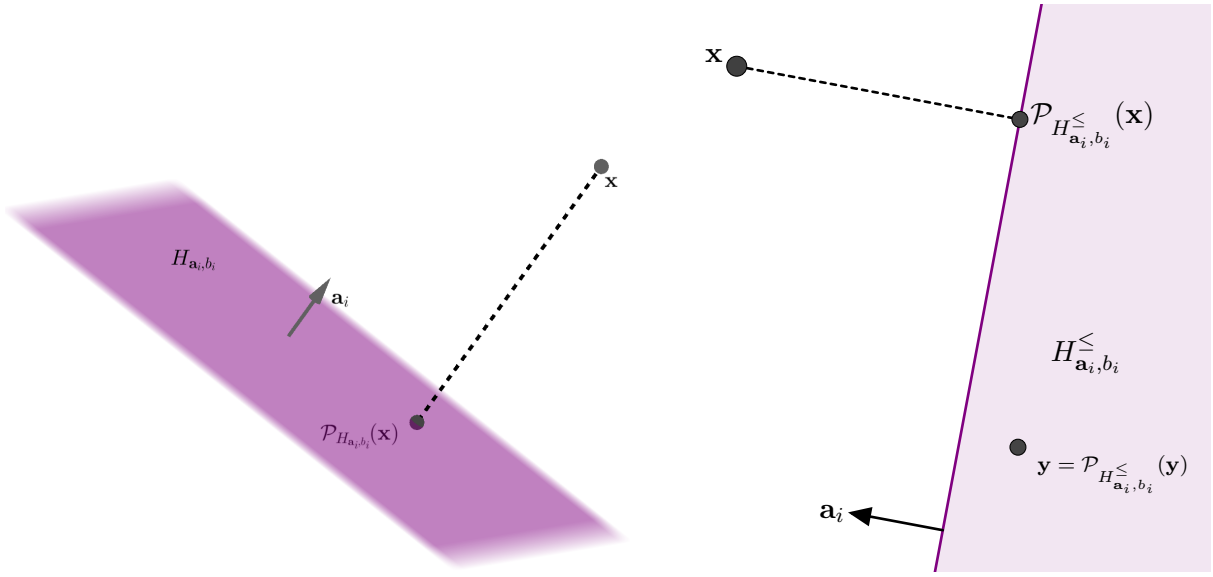

FIGURE 1.3. Left: a visualization of ℓ^2-projection into a hyperplane in three dimensions; Right: a visualization of ℓ^2-projection into a halfspace in two dimensions.

This formula generalizes to ℓ^2-projection onto a *halfspace*, the solution set of a linear inequality, $H^{\leq}_{\mathbf{a}_i,b_i} := \{\mathbf{x} : \mathbf{a}_i^T\mathbf{x} \leq b_i\}$. The ℓ^2-projection of an arbitrary point $\mathbf{x}$ into $H^{\leq}_{\mathbf{a}_i,b_i}$ is given by

$$\mathcal{P}_{H^{\leq}_{\mathbf{a}_i,b_i}}(\mathbf{x}) = \mathbf{x} - \frac{(\mathbf{a}_i^T\mathbf{x} - b_i)^+}{\|\mathbf{a}_i\|^2}\mathbf{a}_i \tag{1.3}$$

where $(\alpha)^+ = \max(0, \alpha)$. Again, this computation requires $\mathcal{O}(n)$ algebraic operations. The projection moves $\mathbf{x}$ parallel to the vector normal to boundary hyperplane $H_{\mathbf{a}_i,b_i}$, $\mathbf{a}_i$, with displacement given by the norm of the second term of the right hand side of (1.3). See Figure 1.3 for an example of projections of $\mathbf{x}$ and $\mathbf{y}$ into the halfspace $H^{\leq}_{\mathbf{a}_i,b_i}$. Again, we point out that if $\mathbf{x}$, $\mathbf{a}_i$ and b_i are rational, then $\mathcal{P}_{H^{\leq}_{\mathbf{a}_i,b_i}}(\mathbf{x})$ is rational.

Another simple case for projection computation is that of ℓ^2-projection onto a *sphere* of radius α about $\mathbf{y}$, $S(\mathbf{y}, \alpha) := \{\mathbf{x} : \|\mathbf{x} - \mathbf{y}\| = \alpha\}$. The projection of $\mathbf{x} \neq \mathbf{y}$ onto the sphere of radius α around $\mathbf{y}$ is given by

$$\mathcal{P}_{S(\mathbf{y},\alpha)}(\mathbf{x}) := \mathbf{y} + \frac{\alpha(\mathbf{x} - \mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|}. \tag{1.4}$$

Figure 1.4 is a visualization of an ℓ^2-projection onto a sphere in three dimensions. Note that this computation requires $\mathcal{O}(n)$ algebraic operations. However, note that the presence of $\|\mathbf{x} - \mathbf{y}\|$ in

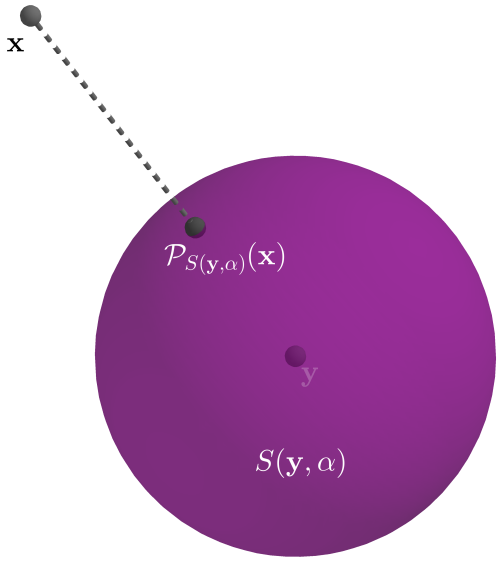

FIGURE 1.4. The ℓ^2-projection of $\mathbf{x}$ onto the sphere of radius α around $\mathbf{y}$ in three dimensions.

the formula for $\mathcal{P}_{S(\mathbf{y},\alpha)}(\mathbf{x})$ means that even if $\mathbf{y}$, $\mathbf{x}$ and α are rational, $\mathcal{P}_{S(\mathbf{y},\alpha)}(\mathbf{x})$ may be irrational; consider the projection of $(1,1)$ onto the unit circle in $\mathbb{R}^2$.

As both the hyperplane and halfspace are convex, the ℓ^2-projection of any point onto either is unique. However, note that the sphere $S(\mathbf{y},\alpha)$ is not convex and that there is precisely one point where $\mathcal{P}_{S(\mathbf{y},\alpha)}(\mathbf{x})$ is non-unique, the center of the sphere. The ℓ^2-projection of $\mathbf{y}$ onto the sphere is non-unique, $\mathcal{P}_{S(\mathbf{y},\alpha)}(\mathbf{y}) = S(\mathbf{y},\alpha)$. Not only is this ℓ^2-projection non-unique, it contains infinitely many points. If the non-convex set K has boundary defined by finitely many linear equations (hyperplanes) the ℓ^2-projection of a point may be non-unique but may contain only finitely many points. Meanwhile, if the boundary of non-convex sets are described by polynomials of higher degree, the projection may be infinite. Figure 1.5 contains an example set X where the projection of the point $\mathbf{x}$, $\mathcal{P}_X(\mathbf{x})$, is infinite.

Note that each of the sets described above is the solution set to a polynomial equation or inequality in n variables; in the case of the hyperplane and halfspace, the polynomial is a linear equation, while in the case of the sphere, the polynomial is a quadratic polynomial. The set of solutions to a system of polynomial equations in n variables is known as a *real algebraic variety*. The set X in Figure 1.5 is a real algebraic variety (solution set of $x^2 + y^2 = z$ in $\mathbb{R}^3$) and the circle $\mathcal{P}_X(\mathbf{x})$ is

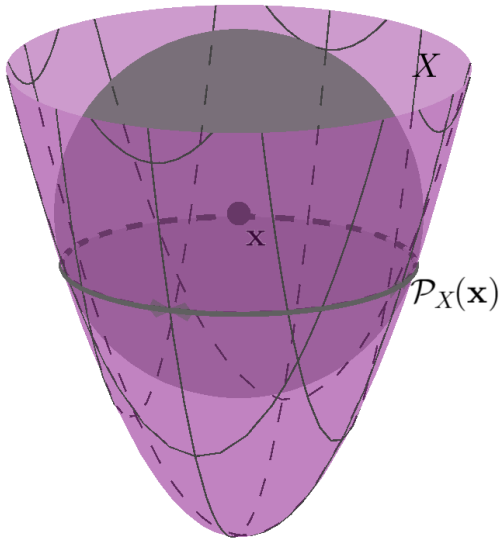

FIGURE 1.5. An example of a variety, X, and point $\mathbf{x}$ where the ℓ^2-projection of $\mathbf{x}$ onto X is infinite.

the projection of $\mathbf{x}$ onto X. Computing the ℓ^2-projection onto a real algebraic variety is in general NP-hard; see [**DHO$^+$16**] and references therein. Hyperplanes, halfspaces, and spheres are the only sets we will discuss which have a closed formula for ℓ^2-projection. We now move on to sets which are more complex than these three, but still less general than arbitrary real algebraic varieties.

Each of the problems and algorithms described in this thesis require projection onto polyhedra. A *polyhedron* (plural *polyhedra*) is the set of solutions to a system of finitely many linear inequalities, $P_{A,\mathbf{b}} := \{\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$ where $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$. If $\mathbf{x}$ solves the system of linear inequalities, $A\mathbf{x} \leq \mathbf{b}$, then it must solve each of the inequalities defined by the ith row of the matrix A and the corresponding entry of $\mathbf{b}$, $\mathbf{a}_i^T\mathbf{x} \leq b_i$. Here we denote the row vectors of A as $\mathbf{a}_i^T$ as we prefer to think of $\mathbf{a}_i$ as defining the normal of the hyperplane $H_{\mathbf{a}_i,b_i}$. As $\mathbf{x}$ satisfies $\mathbf{a}_i^T\mathbf{x} \leq b_i$, it resides in $H^{\leq}_{\mathbf{a}_i,b_i}$. Thus, the polyhedron is the intersection of the halfspaces defined by the rows of A and corresponding entries of $\mathbf{b}$, $P_{A,\mathbf{b}} = \cap_{i=1}^m H^{\leq}_{\mathbf{a}_i,b_i}$. See Figure 1.6 for two examples of polyhedra.

Recall the *affine hull* of a set of points is the set of linear combinations of the points where the coefficients sum to one,

$$\text{aff}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m) := \{\mathbf{x} : \mathbf{x} = \sum_{i=1}^m \lambda_i \mathbf{p}_i, \sum_{i=1}^m \lambda_i = 1\}.$$

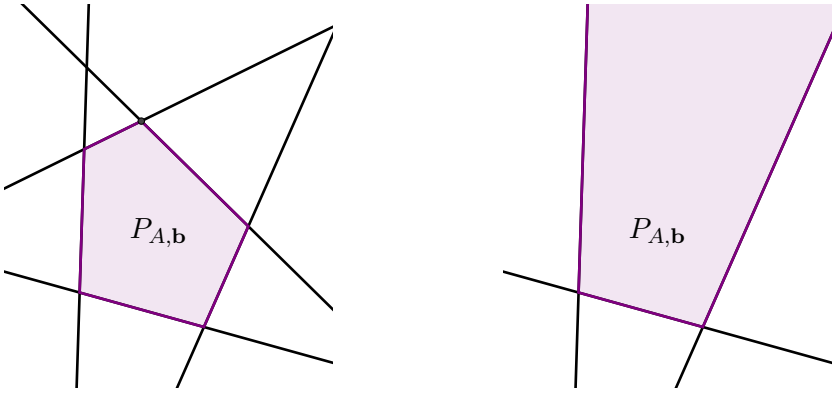

FIGURE 1.6. Two examples of polyhedra in two dimensions. The lines represent the boundary hyperplanes of the halfspaces which define the polyhedra.

Similarly, the *convex hull* of a set of points is the set of linear combinations of the points where the coefficients are nonnegative and sum to one,

$$\text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m) := \{\mathbf{x} : \mathbf{x} = \sum_{i=1}^{m} \lambda_i \mathbf{p}_i, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0\}.$$

The *cone* of a set of points is the set of nonnegative linear combinations of the points,

$$\text{cone}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m) := \{\mathbf{x} : \mathbf{x} = \sum_{i=1}^{m} \lambda_i \mathbf{p}_i, \lambda_i \geq 0\}.$$

See Figure 1.7 for a visual example of these objects for five points. Recall that the *relative interior* of a set S, $\mathbf{relint}(S)$, is the interior of S relative to $\text{aff}(S)$.

The boundary of a polyhedron is made up of polyhedra of smaller dimension called *faces*. A face is a set of points in the polyhedron which maximize a given linear objective. A *facet* is a face of maximal dimension on the boundary of the polyhedron. In Figure 1.6 and Figure 1.8, the facets of the polyhedra are the line segments on the boundary. If the polyhedron is full-dimensional ($\text{aff}(P_{A,\mathbf{b}}) = \mathbb{R}^n$), then the facets lie on the boundary hyperplanes of the halfspaces defining the polyhedron. The linear objective which is maximized on these facets are the normals to the hyperplanes on which they lie. More linear objectives are maximized on faces of lower dimension. The set of linear objectives which are maximized on faces of lower dimension are precisely the *normal cones* of the faces, the cones of the normals of the facets in which they are contained. See

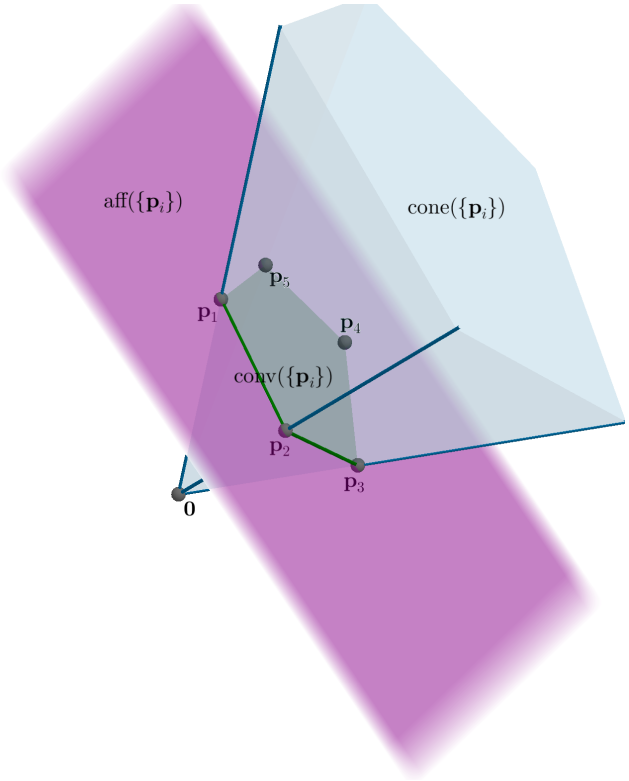

FIGURE 1.7. A visualization of the convex hull, affine hull, and cone of a set of five points.

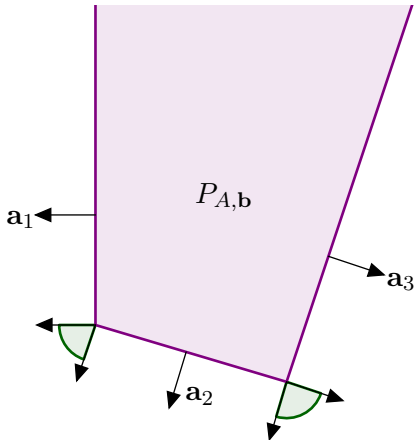

FIGURE 1.8. A polyhedron with the normal cones for each of its faces. Each cone contains all linear objective functions which are maximized on that face.

Figure 1.8 for a visualization. The faces of dimension one are called *edges* and the faces of dimension zero are called *vertices*.

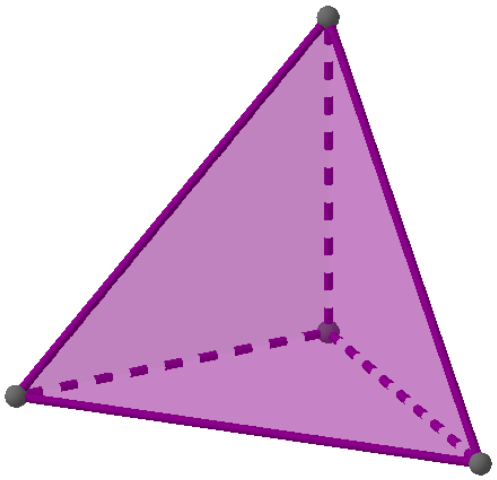

FIGURE 1.9. A 3-simplex in three dimensions.

As Figure 1.6 suggests, polyhedra may be bounded or unbounded. The famed Weyl-Minkowski Theorem gives an additional mode for understanding bounded polyhedra. A bounded polyhedron is both the intersection of finitely many halfspaces and the convex hull of finitely many points, and is known as a *polytope*; see [**Zie12**] and references therein. The left image in Figure 1.6 is a polytope. The halfspaces, if not redundant (unnecessary for defining the polytope), define the hyperplanes which contain the facets of a full-dimensional polytope. Meanwhile, the extremal (unique maximum of linear objective) points are vertices of the polytope. Polytopes which are given as the set of solutions to a system of linear inequalities are called *H-polytopes*. Polytopes which are given as the convex hull of a set of points are called *V-polytopes*. The complexity of testing equivalence of a V-polytope and an H-polytope is not understood fully understood; see [**FO85**] for additional information.

Recall that a *d-simplex* is the convex hull of any $d+1$ affinely independent points in $\mathbb{R}^n$ where $n \geq d$. A d-simplex has $d+1$ facets and thus may also be defined as the intersection of $d+1$ halfspaces in $\mathbb{R}^d$. Simplices are especially interesting in the theory of complexity of optimization as they have the same number of vertices and facets, two objects often manipulated in optimization algorithms. See Figure 1.9 for an example in three dimensions.

Note that the projection of a set S is defined as the set of all projections of points in the set; $\mathcal{P}_K(S) := \{\mathcal{P}_K(\mathbf{x}) : \mathbf{x} \in S\}$. We will mainly be interested in projections of polyhedra into affine and

linear subspaces. The ℓ^2-projection of a polyhedron into an affine subspace remains a polyhedron; see [**Zie12**] for references.

For convenience, we now set some notation that we will maintain throughout the rest of this thesis. As above, vectors will be denoted in boldface (e.g., $\mathbf{a}$), matrices will be capitalized (e.g., A), and scalars will be lower case English (usually subscripted) and Greek letters (e.g., b_i and α). The origin will be denoted $\mathbf{0} \in \mathbb{R}^n$ and we will denote the all-ones vector as $\mathbb{1} = (1, 1, ..., 1)^T \in \mathbb{R}^n$. We use $\mathbf{a}_i^T \in \mathbb{R}^n$ to represent the ith row of $A \in \mathbb{R}^{m \times n}$ and $\mathbf{e}_i \in \mathbb{R}^k$ to represent the ith coordinate vector in k-dimensional space. We will define the positive entries of a vector, $\mathbf{v}^+$, to be defined entry-wise as $(\mathbf{v}^+)_i := (v_i)^+$. We define the absolute value of a vector $|\mathbf{v}|$ to be defined entry-wise as $|\mathbf{v}|_i := |v_i|$.

In this thesis, polyhedra will be given in two forms, either as the intersection of finitely many halfspaces (H-polyhedra) or as the convex hull of finitely many points (V-polytopes). In halfspace form, we will refer to the given set of inequalities whose solution set forms $P_{A,\mathbf{b}}$ as $A\mathbf{x} \leq \mathbf{b}$ where $A \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$, unless otherwise noted. We will label polyhedra of this form by $P_{A,\mathbf{b}}$. Likewise, in vertex form, we will refer to the given points whose convex hull forms P as $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m \in \mathbb{R}^n$, unless otherwise noted. We will label polytopes of this form by P. For such sets of points, we will be concerned with the projection of the origin onto P which is the point of minimum norm in the convex hull,

$$\mathcal{P}_P(\mathbf{0}) = \mathbf{argmin}_{\mathbf{x} \in \text{conv}(\{\mathbf{p}_i\}_{i=1}^m)} \|\mathbf{x}\|,$$

and which we will occasionally call the *convex minimizer*. Computing the convex minimizer of a set of points is the minimum norm point problem discussed in Section 1.3. We will also be interested in the *affine minimizer*,

$$\mathcal{P}_{\text{aff}(\{\mathbf{p}_i\})}(\mathbf{0}) = \mathbf{argmin}_{\mathbf{x} \in \text{aff}(\{\mathbf{p}_i\}_{i=1}^m)} \|\mathbf{x}\|,$$

which is the projection of the origin (and the point of minimum norm) on the affine hull. Computing the affine minimizer of a set of points only requires solving a system of linear equations; see Chapter 4, Lemma 4.1.1 for details.

Many problems in optimization have polyhedral feasible regions. The *feasible region* of an optimization problem is the set of points over which the objective function is optimized. Many results in the theory of complexity of optimization require the size of data describing the polyhedron. We recall the *binary encoding length* of a rational matrix, $A = (a_{ij}/\alpha_{ij})_{i=1,j=1}^{m,n}$ where $a_{ij}, \alpha_{ij} \in \mathbb{Z}$ and $\alpha_{ij} \neq 0$ is

$$\sigma_A := 2mn + \sum_{i,j} \lceil \log_2(|a_{ij}| + 1) \rceil + \lceil \log_2(|\alpha_{ij}| + 1) \rceil,$$

and the length of the binary encoding of a rational vector $\mathbf{b} = (b_i/\beta_i)_{i=1}^{m}$ where $b_i, \beta_i \in \mathbb{Z}$ and $\beta_i \neq 0$, is

$$\sigma_{\mathbf{b}} := 2m + \sum_{i} \lceil \log_2(|b_{ij}| + 1) \rceil + \lceil \log_2(|\beta_{ij}| + 1) \rceil,$$

as in [**Sch86**, **GLS88**]. The binary encoding size of a system of inequalities and the corresponding H-polyhedron is $\sigma_{A,\mathbf{b}} := \sigma_A + \sigma_{\mathbf{b}}$. The binary encoding size of a V-polytope given as the convex hull of the points $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\}$ is $\sigma_P := \sum_{i=1}^{m} \sigma_{\mathbf{p}_i}$.

Note that the polyhedra and polytopes considered in this thesis reside in ambient space $\mathbb{R}^n$ (if the dimension is fixed), unless otherwise noted. As we deal with lifts of polyhedra in Chapters 3 and 4, we will sometimes deal with polyhedra in $\mathbb{R}^d$ (this should indicate that the dimension may be changing). Occasionally, we will be concerned with polyhedra which are given as the intersection of finitely many hyperplanes, and we give them as the solution set for the system of linear equations $A\mathbf{x} = \mathbf{b}$ where $A \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ are given data. We will label polyhedra of this form by $P^{=}_{A,\mathbf{b}}$. As above, projection operators will be denoted by $\mathcal{P}_K$ and minimize the Euclidean distance, unless otherwise noted.

Our theoretical results will often deal with the matrices defining the polyhedra in question, $A \in \mathbb{R}^{m \times n}$. Let $[m] = \{1, 2, ..., m\}$ and let $[A]$ refer to the set of indices of the rows of matrix A (i.e., for $A \in \mathbb{R}^{m \times n}$, $[A] = [m]$). For $I \subset [A]$, we let A_I denote the submatrix of A of rows indexed by elements of I. For $D \subset [A]$, we let $A_{D^C} := A_{[A]-D}$ denote the submatrix of A whose rows are indexed by the complement of D. Often our theoretical results will deal with the *residual* of an LF or a system of linear equations, which is $(A\mathbf{x} - \mathbf{b})^+$ or $A\mathbf{x} - \mathbf{b}$, respectively.

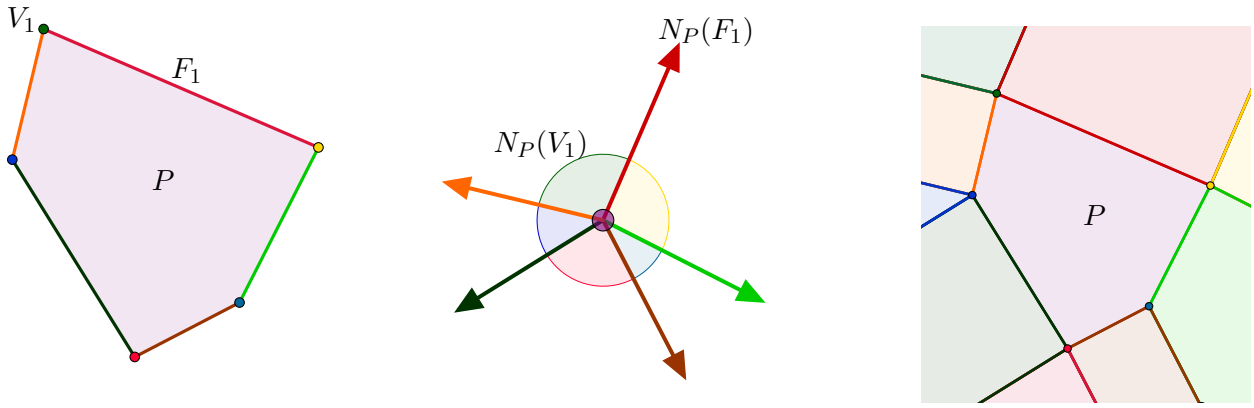

FIGURE 1.10. Left: A polytope P; middle: the normal fan of P; right: the normal manifold of P.

We will often consider polyhedra defined by $A, \mathbf{b}$ where A is a random matrix. We call A a *Gaussian matrix* if the entries of A are iid Gaussian random variables ($a_{ij} \sim \mathcal{N}(0,1)$). We also discuss other matrices where $a_{ij} \sim \mathcal{N}(\mu, \sigma^2)$ for different mean and variance values, but we will define these in the appropriate section. We also discuss *normalized* matrices by which we mean matrices that have been row-normalized so that $\|\mathbf{a}_i\| = 1$ for $i = 1, ..., m$. A normalized Gaussian matrix is a matrix that was of Gaussian construction (entries according to the standard normal distribution) before being row-normalized.

One nice property of V- and H-polyhedra is that $d(\mathbf{x}, P) = 0$ if and only if $\mathbf{x} \in P$ even if the polyhedron is unbounded. Note that the projection of any point onto a nonempty polyhedron is well-defined and simple to describe; it is the projection of that point onto the nearest face of the polyhedron; in particular, $d(\mathbf{x}, P) = \min_{F \text{ face of } P} d(\mathbf{x}, F)$. The ℓ^2-projection of any point onto any polyhedron is unique. Note that the projection may lay on any face of P. In fact, the *normal fan* of a polytope P partitions $\mathbb{R}^n$ into sets of points which project onto each face; see [**Zie12**] and references therein. The normal fan is the collection of the normal cones of the faces of the polytope. For every face F of P, the set of points whose projection lies in relint(F) is relint$(F) + N_P(F)$ where $N_P(F)$ is the normal cone of P on the face F. This partition of $\mathbb{R}^n$ is often denoted the *normal manifold*. See Figure 1.10 for an example.

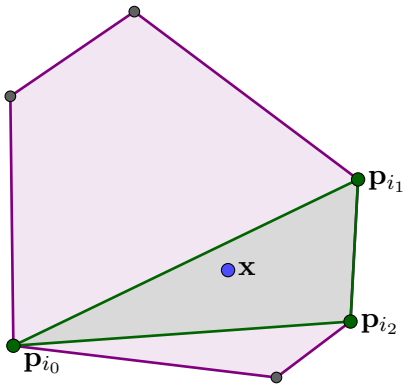

FIGURE 1.11. An example of Carathéodory's theorem applied to a set of points.

Another nice property of polytopes is that the points in them may be expressed as a convex combination of no more than $n+1$ points, even if the polytope itself is the convex hull of $m \gg n+1$ points. This fact is *Carathéodory's theorem*. See Figure 1.11 for an example.

Theorem 1.1.3 (Carathéodory's theorem)**.** *The point* $\mathbf{x} \in \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m)$ *for* $\mathbf{p}_i \in \mathbb{R}^n$ *if and only if* $\mathbf{x} \in \text{conv}(\mathbf{p}_{i_0}, \mathbf{p}_{i_1}, ..., \mathbf{p}_{i_n})$ *for some subset of* $n+1$ *points* $\{\mathbf{p}_{i_0}, \mathbf{p}_{i_1}, ..., \mathbf{p}_{i_n}\} \subseteq \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\}$.

This fact translates into potential algorithmic efficiency in polytope membership questions. If one knows which vertices, $\mathbf{p}_{i_k}$, express the point in question, one may discard the extra vertices allowing for a simpler computation. This fact also allows for *triangulation* of polytopes, which is an entire body of theory and algorithmic questions we will not explore in this thesis; see [**DLRS10**] for more information.

As mentioned above, we are motivated to study projections because problems and algorithms formulated as projections onto polyhedra are ubiquitous throughout optimization, analysis, machine learning, statistics, and data and computer science. Examples of the exciting areas where projections arise include compressed sensing, linear programming, regression, and combinatorial optimization; we will see examples from these areas in Sections 1.2 and 1.3. The vast mathematical theory regarding projections have enabled the development of provably efficient methods for problems spanning these fields. We now discuss the two motivating uses of projections in this thesis; *iterative projection methods*, methods which iteratively project onto polyhedra to solve an optimization

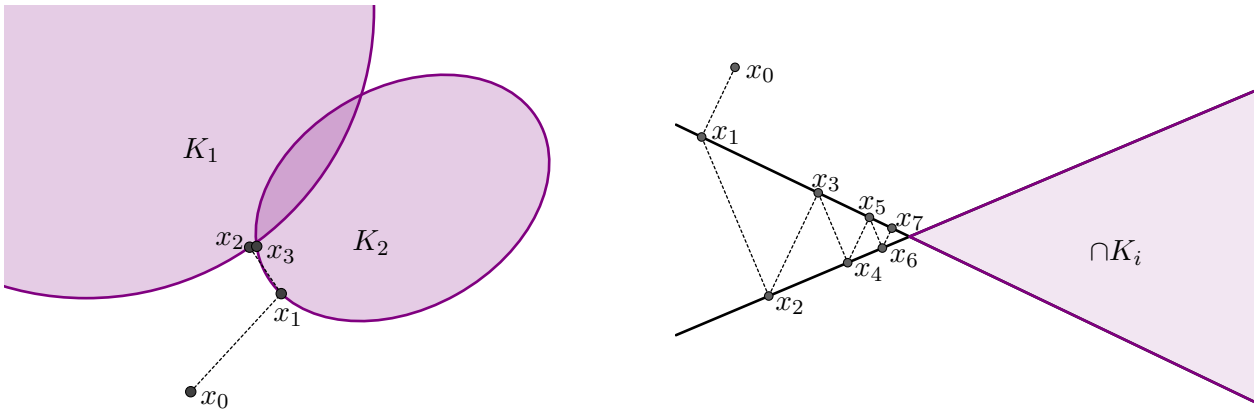

FIGURE 1.12. Examples of iterative projection methods which approximately compute a point in $\cap K_i$.

problem, and *projection problems*, optimization problems which are naturally formulated as the problem of computing a projection onto a polyhedron.

Most *optimization problems*, $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$ subject to $g(\mathbf{x}) \leq \mathbf{0}, h(\mathbf{x}) = \mathbf{0}$, are solved or approximately solved with *iterative methods* of the form $\mathbf{x}_{k+1} = \mathbf{x}_k + \mu_k T(\mathbf{x}_k)$. Here, $T(\mathbf{x}_k)$ is an *improving direction* that decreases the objective function, $f(\mathbf{x})$, while (possibly) preserving feasibility, $g(\mathbf{x}) \leq \mathbf{0}, h(\mathbf{x}) = \mathbf{0}$. The steepest (or gradient) descent method, the stochastic gradient method, Newton's method, the Simplex method, and interior-point methods all fall into this category of optimization procedures; see [**BV04**, **Sch86**, **SNW12**] for more information. In many cases, the optimization problem may be reformulated as an equivalent *feasibility problem*, find $\mathbf{x} \in \cap_{i=1}^n K_i$ where K_i are (possibly convex) sets encoding the original problem. In this case, the step $\mu_k T(\mathbf{x}_k)$, projects $\mathbf{x}_k$ iteratively into a (convex) set, K_i, and is given by $\mathcal{P}_{K_i}(\mathbf{x}_k)$. These types of methods are known as *iterative projection methods*. See Figure 1.12 for an example.

Furthermore, many problems in data science and machine learning are expressly formulated as *projection problems*, $\min \|\mathbf{x}\|_a$ subject to $g(\mathbf{x}) \leq \mathbf{0}, h(\mathbf{x}) = \mathbf{0}$. This problem is to compute the projection of $\mathbf{0}$ in norm $\|\cdot\|_a$ onto the feasible region, $\{\mathbf{x}|g(\mathbf{x}) \leq \mathbf{0}, h(\mathbf{x}) = \mathbf{0}\}$. See Figure 1.13 for two examples.

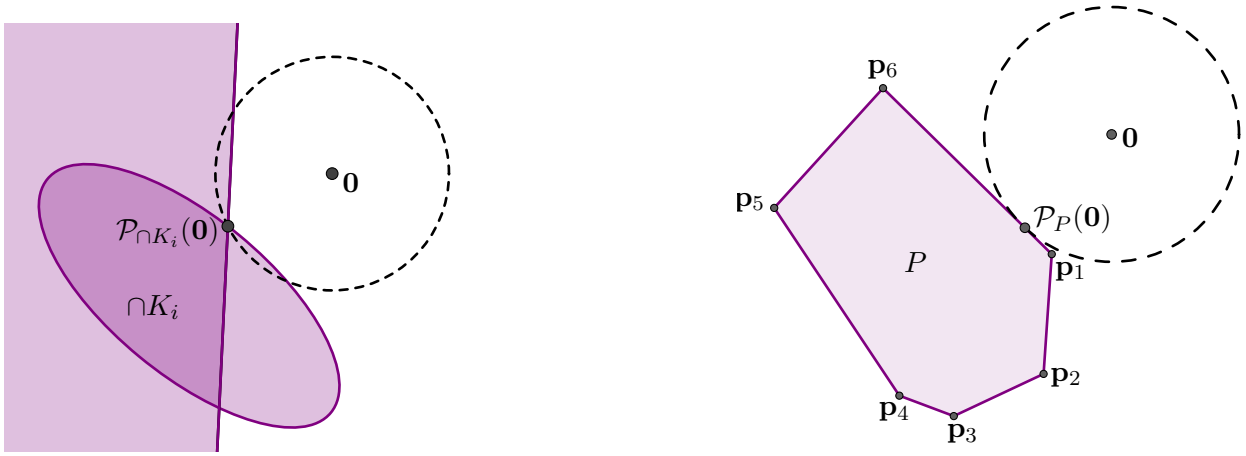

FIGURE 1.13. Examples of projection problems which compute the minimum norm point in the feasible region.

1.2. Linear Feasibility (LF)

The *linear feasibility problem* (LF) is the problem of computing a point in a given polyhedron, $P_{A,\mathbf{b}} = \{\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$. In this thesis, we consider polyhedra defined by rational data, as we consider LF a computational problem and wish to develop efficient algorithms for solving this and related problems. This problem arises in many areas of optimization and machine learning. Perhaps the most frequent application of LF is linear programming, however it also arises in machine learning as a classification problem known as the support vector machine; we discuss several of these applications in Section 1.2.1. We give a rigorous definition of this computational problem below.

Definition 1.2.1. *Consider the following computational problem:*

- **LF:** *Given a rational matrix A and a rational vector $\mathbf{b}$, if $P_{A,\mathbf{b}} := \{\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$ is nonempty, output a rational $\mathbf{x} \in P_{A,\mathbf{b}}$, otherwise output NO.*

We will refer to LF problems with an empty feasible polyhedron, $P_{A,\mathbf{b}}$, as *infeasible*. Methods for solving this problem must be able to detect infeasibility.

For computational reasons, it is useful to note that the vertices of a polyhedron, $P_{A,\mathbf{b}}$, defined by rational $A, \mathbf{b}$ are rational. We include a proof of this fact in Chapter 4, Corollary 4.1.3. Note that an LF may be formulated as a linear program. Linear programming is known to be polynomial

time solvable via interior-point methods [**Kar84**] or the ellipsoid method [**Kha79**]. These methods compute an approximation to a solution ($\mathbf{y}$ within ϵ distance of the solution $\mathbf{x}$) and then rationality of the solution allows for rounding of the approximation. The complexity of these methods depend upon the required approximation accuracy, which depends upon the binary encoding size of the problem. The simplex method [**Dan48**] is a popular combinatorial solver for linear programs. While exponential in the worst-case [**KM72**], it has been shown to perform well in several *average-case* analyses [**Bor82**, **ST04**, **Ver09**]. Linear programming (and LF) can be solved in strongly-polynomial time for some special cases [**Tar86**].

1.2.1. Examples of LF Applications. The wide applicability of the LF problem cannot be overstated. Much of the success of the technical age is due to the success of algorithms developed for solving problems of this type. The problems that can be formulated as an LF appear in many areas of optimization, statistics, computer science, and engineering. This section presents brief discussions of a few problems of this type and related problems. We distinguish between problems that may be formulated exactly as an LF, and problems that are similar in nature by marking these as *related problems*.

1.2.1.1. *Linear Programming.* A *linear program* (LP) is the problem of maximizing (minimizing) a linear objective functional constrained by a system of linear inequalities,

$$\max(\min)\ \mathbf{c}^T\mathbf{x} \text{ s.t. } A\mathbf{x} \leq \mathbf{b}.$$

The idea of linear programming goes back to letters of Fourier from 1826 and 1827, though the creation of this vastly important discipline may be attributed to Dantzig, Kantorovich, Koopmans, and von Neumann; see [**Sch86**] and references therein. The quick recognition of its importance was, in part, due to the fact that it became quite useful following World War II, as many problems of resource allocation were able to be formulated as LPs and solved using the early-developed optimization algorithms.

One should note that there are numerous equivalent formulations of an LP; see [**Sch86**, Section 7.4] for a discussion. As mentioned previously, however, one great strength of the LP problem is that it may be equivalently reformulated as an LF problem. For clarity, we provide a rigorous definition

of LP below and in Lemma 4.2.3 provide the result that LP and LF are strongly-polynomial time equivalent problems. The proof of this lemma is not new and may be found in [**Sch86**, Section 7.4]; however, for completeness we include this proof in Chapter 4.

Definition 1.2.2. *Consider the following computational problems:*

- LP**:** *Given a rational matrix A, a rational column vector $\mathbf{b}$, and a rational row vector $\mathbf{c}^T$, output rational $\mathbf{x} \in \operatorname{argmax}\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$ if $\max\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$ is finite, otherwise output INFEASIBLE if $P_{A,\mathbf{b}}$ is empty and else output INFINITE.*
- LFE**:** *Given a rational matrix A and a rational vector $\mathbf{b}$, if $P := \{\mathbf{x} : A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ is nonempty, output a rational $\mathbf{x} \in P$, otherwise output NO.*

Lemma 1.2.3. *LP is strongly-polynomial time equivalent to LFE.*

The proof of this lemma is included in Chapter 4, Section 4.2.2. As stated in Definition 1.2.2, the LFE problem is that of finding a point which satisfies the system of equations $A\mathbf{x} = \mathbf{b}$ and the inequalities $\mathbf{x} \geq 0$. However, it is trivial to transform this into an equivalent LF.

The geometric interpretation of an LP problem is computing the point in the feasible polyhedron, $P_{A,\mathbf{b}}$, furthest in the direction of the objective functional, $\mathbf{c}$; see Figure 1.14 for an example. As seen in Definition 1.2.2 and Figure 1.14, there are three outcomes; the LP can be finite, infeasible, or infinite.

While LP is polynomially solvable, an important open question is whether there exists a strongly-polynomial time method for linear programming [**Sma00**]. Informally speaking, a strongly-polynomial time method for linear programming is one whose number of arithmetic operations depends on only the dimensions of the problem, m and n, and not on the binary encoding size of the input data, σ_A, $\sigma_{\mathbf{b}}$ or $\sigma_{\mathbf{c}}$. The simplex method, a combinatorial, finite method, is a natural candidate for a strongly-polynomial time algorithm, but all well-known pivot rules have been shown to have exponential behavior; see [**AZ99**] and references therein. In Section 1.4, we show that a strongly-polynomial time algorithm for the minimum norm point problem (discussed in Section 1.3) would provide a strongly-polynomial time algorithm for LP. It has recently been shown that standard

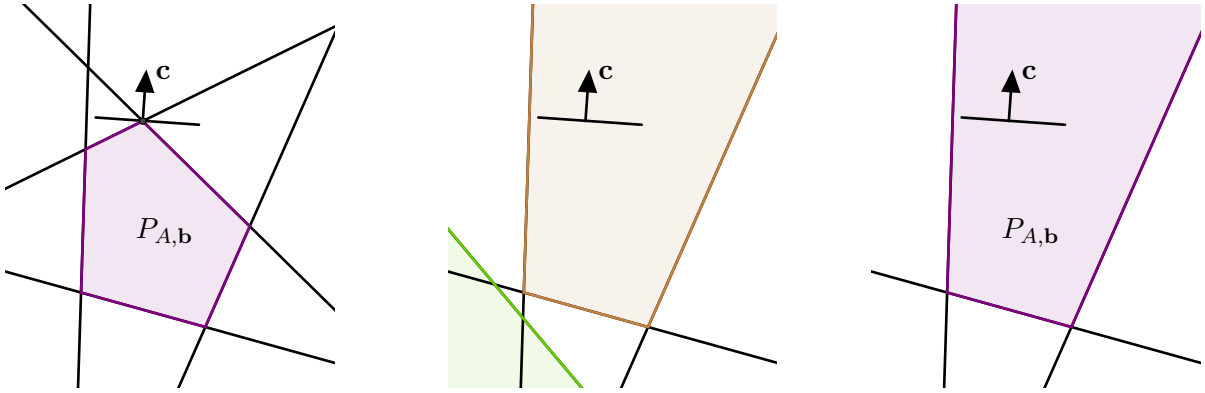

FIGURE 1.14. Left: a finite-valued linear program with feasible polyhedron, $P_{A,\mathbf{b}}$, and linear objective, $\mathbf{c}$; middle: an infeasible linear program with the green halfspace incompatible with the brown polyhedron defined by the other halfspaces; right: an infinite-valued linear program with feasible polyhedron, $P_{A,\mathbf{b}}$, and linear objective, $\mathbf{c}$.

primal-dual log-barrier interior-point methods for linear programming are not strongly-polynomial time algorithms [**ABGJ18**], which only increases the interest in combinatorial methods for linear programming and the minimum norm point problem.

1.2.1.2. *Systems of Linear Equations.* One of the most common and elementary problems in data science is that of solving a system of linear equations, $A\mathbf{x} = \mathbf{b}$ where $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$. The list of exciting and complex areas in which this type of problem arises is far too long to list, but includes linear inverse problems, regression, a subroutine of the simplex method for linear programming, a subroutine of Newton's method for general convex optimization, and a subroutine of methods for tensor decomposition; see [**Sch86**, **BV04**, **BBK17**] for more information. This problem, finding $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}$, may be trivially reformulated as an LF which will be feasible if and only if the original system of equations is feasible. This reformulation may be geometrically interpreted as replacing each of the m hyperplanes defined by the rows of the system of equations with two halfspaces whose intersection is exactly this hyperplane. Solving the system of equations finds a point in the intersection of the m hyperplanes defined by rows $\mathbf{a}_i^T\mathbf{x} = b_i$, while solving the system of $2m$ inequalities finds a point in the intersection of the $2m$ halfspaces defined by $\mathbf{a}_i^T\mathbf{x} \leq b_i$ and $-\mathbf{a}_i^T\mathbf{x} \leq -b_i$. See Figure 1.15 for an example of this transformation.

If the rank of the matrix A is at least n (full-rank) and the problem is feasible then the aforementioned LF problem will have a unique solution. See the left image of Figure 1.15 for an example of this

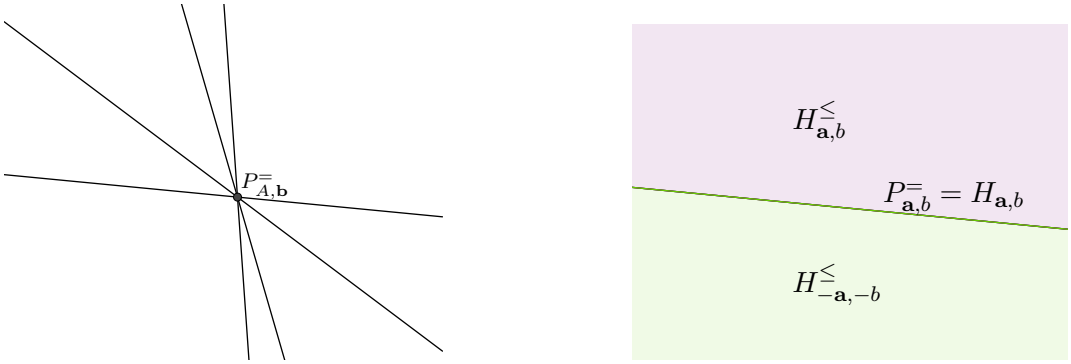

FIGURE 1.15. Left: an example of a polyhedron given as the solution set of a system of equations; right: a geometric view of an example of transformation of $A\mathbf{x} = \mathbf{b}$ to equivalent system of inequalities.

situation. Throughout the rest of this manuscript, we will be concerned primarily with the situation when the matrix is full-rank and the system is overdetermined, $m \gg n$. However, in most applications where systems of linear equations arise, the system will be noisy or corrupted. By *noisy*, we mean the situation in which any, and most likely many, of the entries of A and $\mathbf{b}$ have been altered by noise (which we generally view as numbers of a small random magnitude). By *corrupted*, we mean the situation in which few of the entries of A and the entries of the right hand side vector $\mathbf{b}$ have been altered by corruption (which we generally view as numbers of arbitrary magnitude). In Section 1.2.1.4, we discuss a problem related to 'solving' a corrupted system of equations, and in Section 1.3.1.4, we discuss a problem related to 'solving' a noisy system of equations. We define which solution is desired in these sections.

1.2.1.3. *Linear Support Vector Machine.* Within machine learning, supervised binary classification is formulated as the *support vector machine* (SVM) problem, whose feasibility form is an LF problem. The SVM problem is, given binary classified training data, $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where $\mathbf{x}_i \in \mathbb{R}^{n-1}$ and

$$y_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \in \text{ class 1} \\ -1 & \text{if } \mathbf{x}_i \in \text{ class 2,} \end{cases}$$

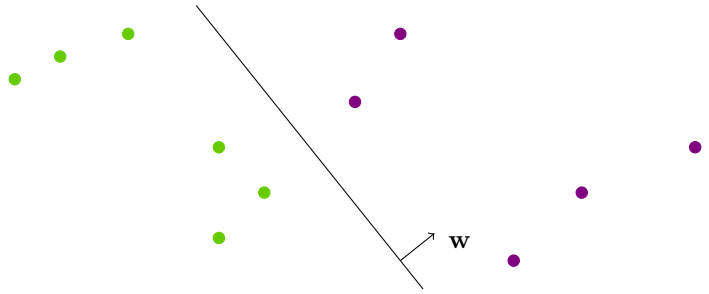

FIGURE 1.16. A linearly separable SVM problem with linear classifier.

to find a linear classifier, $F(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i - \beta$ so that $y_iF(\mathbf{x}_i) \geq 1$ for all $i = 1, ..., m$. This leads to a system of linear inequalities $\tilde{X}\tilde{\mathbf{w}} \leq -\mathbb{1}$ where

$$\tilde{X} = \begin{bmatrix} -y_1\mathbf{x}_1^T & y_1 \\ \vdots & \\ -y_m\mathbf{x}_m^T & y_m \end{bmatrix} \in \mathbb{R}^{m\times n} \text{ and } \tilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ \beta \end{bmatrix} \in \mathbb{R}^n.$$

See Figure 1.16 for a visual representation of this separation problem.

Geometrically, this problem is to find a hyperplane which separates these two classes of points. This supervised machine learning model computes such a hyperplane, known as the *linear classifier*, which can then be used to predict to which class a new, unlabeled data point belongs. For an example of the use of such a model, see Section 2.4.2. The study of SVM is extensive; see [**CE14**] for a reference.

One important note about SVM is that this feasibility formulation is only feasible (a separating hyperplane given by $\tilde{\mathbf{w}}$ exists) if the data points are linearly separable, which is not often the case in applications. In order to find a classifying hyperplane which separates many of the data points, *soft-margin SVM* uses the hinge-loss objective, $\max(0, 1 - y_iF(\mathbf{x}_i))$. Note that this objective function gives positive penalty only to those data points which do not satisfy $y_iF(\mathbf{x}_i) \geq 1$. Full soft-margin SVM usually minimizes the sum of the hinge-loss objectives with a regularizer on the normal of the hyperplane, $\lambda\|\mathbf{w}\|^2$, which encourages the maximization of the distance of the classifying hyperplane from the nearest correctly classified data points in each class.

If we instead consider minimizing the objective function

$$\sum_{i=1}^{m} \max(0, 1 - y_i F(\mathbf{x}_i)),$$

we are maximizing the sum of $y_i F(\mathbf{x}_i)$ for those data points, $\mathbf{x}_i$, which have been misclassified. One should note that this optimization problem is a convex problem which is easily solvable. This problem is to minimize the positive residual entries of an LF, which is polynomially solvable with interior-point methods for linear programming. The problem of instead minimizing the number of misclassified data points is to find the largest feasible subproblem of an LF, which is known as MAX-FS. We discuss this related problem next.

1.2.1.4. *Related problem: MAX-FS.* As one can imagine, many LF problems encountered in applications are infeasible. There are many useful reformulations of infeasible LF problems (e.g., to minimize a norm of the positive entries of the residual, $(A\mathbf{x} - \mathbf{b})^+$). The *maximal feasible subsystem* problem, MAX-FS, is to find the largest feasible subsystem in an infeasible LF, that is to compute

$$\min \|(A\mathbf{x} - \mathbf{b})^+\|_0$$

where $\|\cdot\|_0$ denotes the ℓ^0 cardinality function. This problem is NP-hard due to the presence of the ℓ^0 cardinality function, whereas minimizing $\|(A\mathbf{x}-\mathbf{b})^+\|_p$ for $p \geq 1$ is polynomially approximable due to the convexity of the norm, and polynomially solvable if $p = 1, 2$. As well as being NP-hard, MAX-FS is notoriously difficult to approximate, with the existence of a polynomial time approximation scheme being equivalent to $P = NP$ [**AK95**]. This problem is one of the foci of infeasibility analysis, the study of changes necessary to make an infeasible system of linear constraints feasible; see [**MKC00**] for more information.

Instances of MAX-FS arise when solving inconsistent systems of linear equations as mentioned in Section 1.2.1.2, and in the context of SVM on data that is not linearly-separable as mentioned in Section 1.2.1.3. Additionally, it appears in applications like logic programming, error detection and correction in telecommunications, and infeasible linear programming models. However, due to the difficulty of this problem, it is more common to solve an easier problem like least-squares (see Section

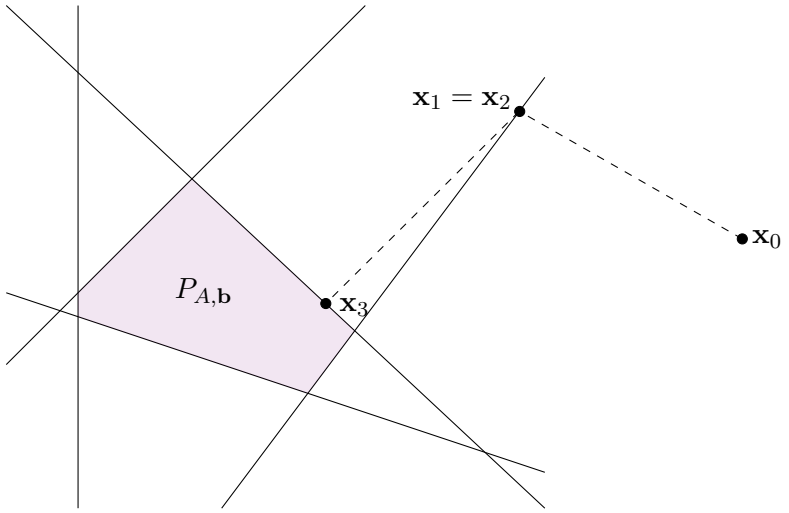

FIGURE 1.17. An example of the iterates of an iterative projection method on an LF.

1.3.1.4). In Section 2.2.2, we propose and analyze a randomized method for solving this problem when the feasible subsystem is large relative to the number of the infeasible constraints.

1.2.2. Iterative Projection Methods for LF. A common and classical approach to solving LF problems is via iterative projection methods, also known as *relaxation methods* or *Bregman projection methods* [**MS54**, **Agm54**, **Bre67**]. Beginning with an arbitrary guess, $\mathbf{x}_0$, the method iteratively projects into a halfspace in order to satisfy a linear inequality; i.e., choose i_k and define

$$\mathbf{x}_k = \mathcal{P}_{H^{\leq}_{\mathbf{a}_{i_k}, b_{i_k}}}(\mathbf{x}_{k-1}) = \mathbf{x}_{k-1} - \frac{(\mathbf{a}_{i_k}^T \mathbf{x}_{k-1} - b_{i_k})^+}{\|\mathbf{a}_{i_k}\|^2} \mathbf{a}_{i_k} \tag{1.5}$$

which is the orthogonal projection of $\mathbf{x}_{k-1}$ into the halfspace, $H^{\leq}_{\mathbf{a}_{i_k}, b_{i_k}}$, defined by the i_kth constraint [**Sch86**]. There are many variants of this basic method which vary in their row selection rule (i.e., how to choose i_k) and are repeatedly reinvented and analyzed; see [**Cen81**] and references therein. See Figure 1.17 for an example of iterates defined by an iterative projection method on an LF. These methods may all be considered special cases of the fixed point operator defined in Equation (1.1), T, with $K_i := H^{\leq}_{\mathbf{a}_i, b_i}$ for $i \in [m]$.

Note that if $P_{A,\mathbf{b}}$ is nonempty and the row selection strategy selects all constraints infinitely many times (at least almost surely) then the iterative projection method iterates will converge to a feasible point [**Bre67**, **MS54**, **Agm54**]. It is possible that no iterate is itself feasible, but the iterates will continually move closer to the feasible region. Much analysis in this area seeks to bound the *convergence rate*, i.e., how quickly the iterates approach a feasible point. Given a bound on the convergence, $d(\mathbf{x}_k, P_{A,\mathbf{b}}) = \mathcal{O}(f(k))$ where f is an easily invertible function that strictly decreases with k, for any ϵ_K one can compute K so that after K iterations, $\min_{\mathbf{y}\in P_{A,\mathbf{b}}} \|\mathbf{x}_K - \mathbf{y}\| \leq \epsilon_K$. This allows for the development of a *stopping criterion*, a rule for deciding when to cease a non-finite iterative method.

Note that the distance between the iterates,

$$\|\mathbf{x}_{k+1} - \mathbf{x}_k\| = \frac{(\mathbf{a}_{i_k}^T \mathbf{x}_k - b_{i_k})^+}{\|\mathbf{a}_{i_k}\|},$$

depends upon the magnitude of the residual. In order to use the known distance $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|$ to bound the unknown distance

$$\min_{\mathbf{y}\in P_{A,\mathbf{b}}} \|\mathbf{x}_{k+1} - \mathbf{y}\|,$$

we must have a bound for how the residual and distance are related. This relationship is governed by the *Hoffman constants* investigated first by Agmon [**Agm54**] and then later by Hoffman [**Hof52**]. If the system of inequalities $A\mathbf{x} \leq \mathbf{b}$ is feasible, i.e., $P_{A,\mathbf{b}} \neq \emptyset$, then there exist Hoffman constants L_∞ and L_2 so that

$$d(\mathbf{x}, P_{A,\mathbf{b}}) \leq L_\infty \|(A\mathbf{x} - \mathbf{b})^+\|_\infty \quad \text{and} \quad d(\mathbf{x}, P_{A,\mathbf{b}}) \leq L_2 \|(A\mathbf{x} - \mathbf{b})^+\|_2$$

for all $\mathbf{x} \in \mathbb{R}^n$. The constants satisfy $L_\infty \leq \sqrt{m} L_2$. These constants are related to the singular values of the matrix A. We will denote the minimum singular value of A as $\sigma_{\min}(A)$.

The Hoffman constants are, in general, not simple to compute. However, we include several extremely simple examples in which they may be computed. First, if the system is orthogonal with $A = I$, then $L_2 = 1$ and $L_\infty = \sqrt{n}$. In the case of highly redundant systems, if every row of A and entry of $\mathbf{b}$ is identical, then $L_2 = 1/\sqrt{m}$ and $L_\infty = 1$. When the system of inequalities, $A\mathbf{x} \leq \mathbf{b}$, defines a consistent system of equations, $\tilde{A}\mathbf{x} = \tilde{\mathbf{b}}$, with full column-rank matrix, $\tilde{A}$, then the Hoffman

constant, L_2, is simply the norm of the left inverse which is equal to the reciprocal of the smallest singular value of A, $\|\tilde{A}^{-1}\| = \frac{1}{\sigma_{\min}(A)}$.

1.2.2.1. *Motzkin's Method.* A simple and classical projection method is *Motkzin's method* (MM). MM is an iterative projection method where the inequality defining the halfspace onto which the iterate $\mathbf{x}_k$ is projected is chosen greedily, $i_k \in \text{argmax}(\mathbf{a}_{i_k}^T \mathbf{x}_k - b_{i_k})^+$ [**MS54**]. This method will be discussed in detail in Section 2.1. With the Hoffman constants, one can prove convergence rate results like the following (a spin-off of Theorem 3 of [**Agm54**] which is easily proven in the style of [**LL10**]):

Proposition 1.2.4. *Consider a normalized system with $\|\mathbf{a}_i\| = 1$ for all $i = 1, ..., m$. If the feasible region $P_{A,\mathbf{b}}$ is nonempty then Motzkin's method converges linearly:*

$$d(\mathbf{x}_k, P_{A,\mathbf{b}})^2 \leq \left(1 - \frac{1}{L_\infty^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2 \leq \left(1 - \frac{1}{mL_2^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2.$$

We include a proof of this result in Section 2.1.1. Note that this result deterministically guarantees that the distance between the iterate and the nonempty polyhedral feasible region decreases linearly. If the desired accuracy between the iterate, $\mathbf{x}_k$, and the actual solution, $\mathbf{x}$, is ϵ, then one need only run

$$k \geq \frac{log\epsilon - 2\log d(\mathbf{x}_0, P_{A,\mathbf{b}})}{\log(1 - 1/L_\infty^2)} \text{ iterations.}$$

Note that it is possible to put an upper bound on $d(\mathbf{x}_0, P_{A,\mathbf{b}})$ in terms of $\|\mathbf{x}_0\|$ and the binary encoding size of A and $\mathbf{b}$, $\sigma_{A,\mathbf{b}}$, meaning that Motzkin's method may be used to deterministically find an ϵ-approximation to a feasible $\mathbf{x}$ if one exists.

In Section 2.1.2, we present a novel, accelerated convergence rate for Motzkin's method on systems of linear equations. We additionally present heuristics quantifying the increased convergence rate for Motzkin's method on Gaussian systems of linear equations.

1.2.2.2. *Randomized Kaczmarz.* When the projection halfspace is chosen at random rather than greedily, the iterative projection method is known as the *randomized Kaczmarz* (RK) method [**Kac37**]. This method will be discussed in detail in Section 2.2. Strohmer and Vershynin [**SV09**] provided an elegant convergence analysis of the randomized Kaczmarz method for consistent

equations. Later, Leventhal and Lewis [**LL10**] extended the probabilistic analysis from systems of equations to systems of linear inequalities. They focused on giving bounds on the convergence rate that take into account the numerical conditions of the system of inequalities captured by the Hoffman constant, L_2.

Theorem 1.2.5 ([**LL10**]). *If the feasible region, $P_{A,\mathbf{b}}$, is nonempty then the Randomized Kaczmarz method converges linearly in expectation:*

$$\mathbb{E}[d(\mathbf{x}_k, P_{A,\mathbf{b}})^2] \leq \left(1 - \frac{1}{\|A\|_F^2 L_2^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2.$$

We include a proof of this result in Section 2.2.1. Note the similarities between Proposition 1.2.4 and Theorem 1.2.5; the convergence rate constants are identical for normalized systems ($\|A\|_F^2 = m$). Additionally, note that as RK is a randomized algorithm, one cannot hope for a deterministic guarantee as in Proposition 1.2.4. However, again one may design stopping criterion as in the previous section and they will hold in expectation.

In Section 2.2.2, we present methods which use RK iterations to detect corrupted equations, and present theoretical guarantees on the probability that these methods discard all corrupted equations in a system. The problem these methods solve are related to the problems described in Sections 1.2.1.2 and 1.2.1.4.

1.2.2.3. *Sampling Kaczmarz-Motzkin.* We generalized MM and RK to construct the family of *Sampling Kaczmarz-Motzkin* (SKM) methods, hybrid randomized and greedy methods which select a sample of β rows of A uniformly at random and project onto the halfspace defined by the most violated inequality among the sample [**DLHN17**]. The SKM method with $\beta = m$ recovers Motzkin's method, while the SKM method with $\beta = 1$ recovers the randomized Kaczmarz method. This method will be discussed in greater detail in Section 2.3. We now state our first main result.

Theorem 1.2.6. *Let A be normalized so $\|\mathbf{a}_i\|^2 = 1$ for all rows $i = 1, ..., m$. If the feasible region $P_{A,\mathbf{b}}$ is nonempty then the SKM method with samples of size β converges at least linearly in expectation and the bound on the rate depends on the number of satisfied constraints in the system $A\mathbf{x} \leq \mathbf{b}$. More precisely, let s_{k-1} be the number of satisfied constraints after iteration $k-1$ and*

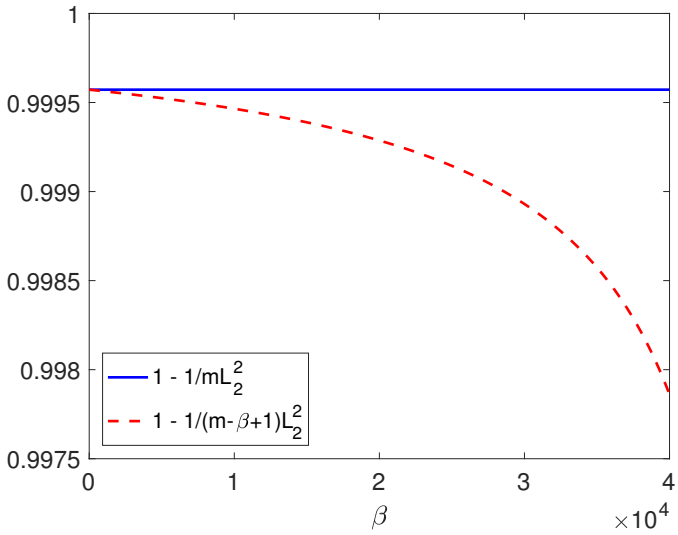

FIGURE 1.18. Comparisons of convergence constant, $\left(1 - \frac{1}{(m-\beta+1)L_2^2}\right)$, for various number of satisfied constraints and sample size of SKM, β.

$V_{k-1} = \max\{m - s_{k-1}, m - \beta + 1\}$*; then, in the* k*th iteration,*

$$\mathbb{E}[d(\mathbf{x}_k, P_{A,\mathbf{b}})^2] \leq \left(1 - \frac{1}{V_{k-1}L_2^2}\right) d(\mathbf{x}_{k-1}, P_{A,\mathbf{b}})^2 \leq \left(1 - \frac{1}{mL_2^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2.$$

We provide a proof of this result in Section 2.3.1. Note that this result guarantees the convergence rate (in expectation) of MM and RK, but potentially offers an increased convergence rate. If at least $\beta - 1$ of the constraints are satisfied in the kth iteration, then the expected ratio of the iterate distances to the feasible region is significantly smaller than guaranteed by Proposition 1.2.4 and Theorem 1.2.5,

$$\frac{d(\mathbf{x}_k, P_{A,\mathbf{b}})^2}{d(\mathbf{x}_{k-1}, P_{A,\mathbf{b}})^2} \leq \left(1 - \frac{1}{(m-\beta+1)L_2^2}\right).$$

In Figure 1.18, we compare these values for various choices of β. In this simulation, L_2 is given by $1/\sigma_{\min}(A)$ for a normalized Gaussian matrix $A \in \mathbb{R}^{50000\times 100}$ (entries are iid $a_{ij} \sim \mathcal{N}(0,1)$ before normalization). Note that as many constraints are satisfied, SKM guarantees an accelerated convergence rate and that many iterations with many satisfied constraints improves the convergence guarantee drastically.

Our second main theoretical result notes that, for rational data, one can provide a *certificate of feasibility* after finitely many iterations of SKM. This is an extension of the results by Telgen [**Tel82**] who also noted the connection between iterative projection techniques and the ellipsoid method. Denote the *maximum violation* of a point $\mathbf{x} \in \mathbb{R}^n$ as $\theta(\mathbf{x}) = \max\{0, \max_{i\in[m]}\{\mathbf{a}_i^T\mathbf{x} - b_i\}\}$.

Telgen's proof of the finiteness of Motzkin's method makes use of the following lemma (which is also key in demonstrating that Khachiyan's ellipsoid algorithm is finite and polynomial time [**Kha79**]):

Lemma 1.2.7. *If the polyhedron, $P_{A,\mathbf{b}}$, defined by the rational system, $A\mathbf{x} \leq \mathbf{b}$, is empty, then for all $\mathbf{x} \in \mathbb{R}^n$, the maximum violation satisfies $\theta(\mathbf{x}) \geq 2^{-\sigma_{A,\mathbf{b}}+1}$.*

Thus, to detect feasibility of the rational system $A\mathbf{x} \leq \mathbf{b}$, we need only find a point, $\mathbf{x}_k$, with $\theta(\mathbf{x}_k) < 2^{-\sigma_{A,\mathbf{b}}+1}$; such a point will be called a *certificate of feasibility.*

In the following theorem, we demonstrate that we expect to find a certificate of feasibility when the system is feasible, and that if we do not find a certificate after finitely many iterations, we can put a lower bound on the probability that the system is infeasible. Furthermore, if the system is feasible, we can bound the probability of finding a certificate of feasibility.

Theorem 1.2.8. *Suppose $A, \mathbf{b}$ are rational matrices and that we run an SKM method on the normalized system $\tilde{A}\mathbf{x} \leq \tilde{\mathbf{b}}$ $\left(\text{where } \tilde{\mathbf{a}}_i = \frac{1}{\|\mathbf{a}_i\|}\mathbf{a}_i \text{ and } \tilde{b}_i = \frac{1}{\|\mathbf{a}_i\|}b_i\right)$ with $\mathbf{x}_0 = \mathbf{0}$. Suppose the number of iterations k satisfies*

$$k > \frac{4\sigma_{A,\mathbf{b}} - 4 - \log n + 2\log\left(\max_{j\in[m]}\|\mathbf{a}_j\|\right)}{\log\left(\frac{mL_2^2}{mL_2^2-1}\right)}.$$

If the system $A\mathbf{x} \leq \mathbf{b}$ is feasible, the probability that the iterate, $\mathbf{x}_k$, is not a certificate of feasibility is at most

$$\frac{\max\|\mathbf{a}_j\|\ 2^{2\sigma_{A,\mathbf{b}}-2}}{n^{1/2}}\left(1 - \frac{1}{mL_2^2}\right)^{k/2},$$

which decreases with k.

We provide a proof of this result in Section 2.3.2. This result provides a number of iterations after which we expect that the SKM iterate will be a certificate of feasibility for a feasible LF. Thus, if one runs more than this number of iterations and the iterate is not a certificate, our bound provides confidence that the system is infeasible.

1.3. Minimum Norm Point (MNP)

The fundamental algorithmic problem we consider here is: given a convex polytope, $P \subset \mathbb{R}^n$, to find the point $\mathbf{x} \in P$ of minimum Euclidean-norm, i.e., the *closest point to the origin* or what we call its *minimum norm point* for short. This problem may be interpreted as computing the projection of the origin, $\mathbf{0}$, onto P. Note that solving the minimum norm point problem solves ℓ^2-projection of an arbitrary point to a polytope; $\mathbf{argmin}_{\mathbf{x}\in P}\|\mathbf{x}-\mathbf{a}\|$ is the same as $\mathbf{a}+\mathbf{argmin}_{\mathbf{y}\in P-\mathbf{a}}\|\mathbf{y}\|$.

We focus on the situation in which P is presented as a V-polytope, although there are many applications in which the polytope could be presented in hyperplane representation; we will discuss several of these related problems in Subsection 1.3.1.

Additionally, we focus on the situation in which projection is in the Euclidean-norm over the polytope. However, there are applications in which an alternate norm or weighted norm could be of interest; we discuss several of these related problems in Subsection 1.3.1. Again, we provide a rigorous definition of our specific computational problem below.

Definition 1.3.1. *Consider the following computational problem:*

- MNP: *Given rational points* $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m \in \mathbb{R}^n$ *defining* $P = \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m)$, *output rational* $p = \mathbf{argmin}_{\mathbf{q}\in P}\|\mathbf{q}\|^2$.

For computational reasons, it is useful to note that if the vertices of the polytope are rational then the minimum norm point will be rational. We include a proof of this fact in Chapter 4, Corollary 4.1.2. Since the Euclidean-norm is a convex quadratic form, the minimum norm point problem is a convex quadratic optimization problem. Indeed, it is well-known that a convex quadratic programming problem can be approximately solved in polynomial time; that is, some point, $\mathbf{y}$, within distance ε of the desired minimizing point, $\mathbf{x}$, may be found in time polynomial with respect to $\log \frac{1}{\varepsilon}$. This can

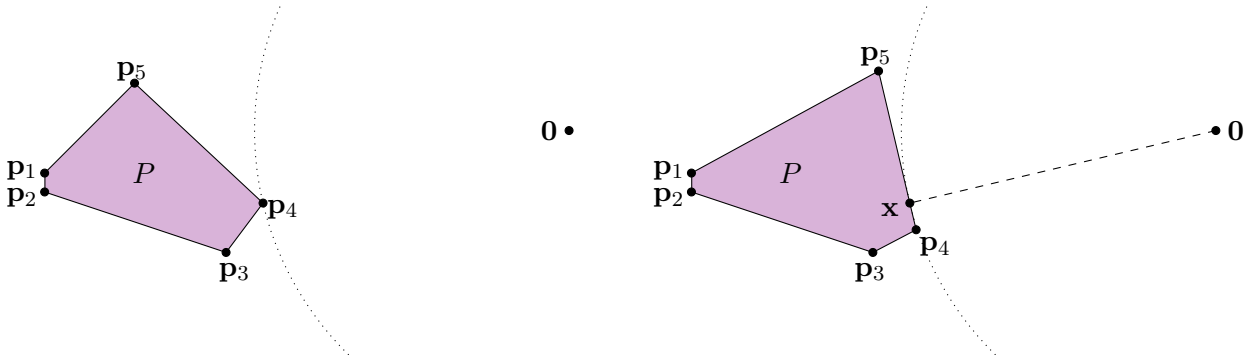

FIGURE 1.19. Left: MNP problem in which the minimum norm point is on a vertex; right: MNP problem in which the minimum norm point is on a facet.

be computed with several iterative (convergent) algorithms, such as the ellipsoid method [**KTK80**] and interior-point method techniques [**BV04**]. Each of these are methods whose complexity depends upon the desired accuracy. However, an approximate numerical solution is inconvenient when the application requires more information; e.g., if we require to know the face that contains the minimum norm point. Furthermore, Freund [**Fre87**] showed that not all convex quadratic programs are equivalent to norm minimization problems. Thus, in general, we face a specialized situation which may allow for faster finite and combinatorial algorithms. The minimum norm problem can indeed be solved in strongly-polynomial time for some polytopes; most notably in network-flow and transportation polytopes; see [**CR16**, **BK80**, **Vég16**], and references therein.

1.3.1. Examples of MNP Applications. As previously discussed, many problems in data science may be formulated as projection (MNP) problems. Many of the exciting, recent advances in machine learning rely upon formulation as convex optimization problems, and use efficient convex and quadratic optimization methods. This section presents brief discussions of a few problems which may be formulated as MNPs or related problems. We distinguish between problems which may be formulated or solved via an MNP formulation exactly as described in Definition 1.3.1, and problems which make use of an alternate norm or a different polytope description by marking these as *related problems*.

1.3.1.1. *Related Problem: Compressed Sensing. Compressed sensing* is a technique that arises in signal processing. The problem is to design a measurement matrix A where $A \in \mathbb{R}^{p \times n}$ with $p < n$,

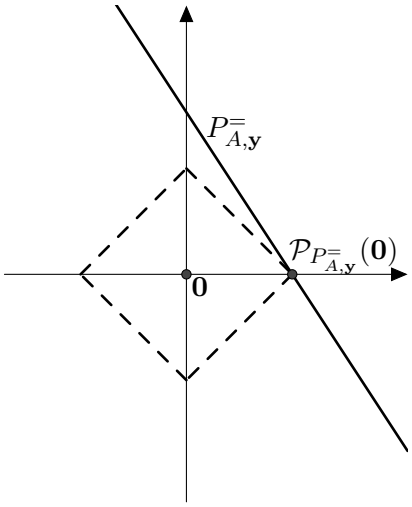

FIGURE 1.20. A visualization of the compressed sensing recovery technique. Note that the ℓ^1-projection of the origin onto $P^{=}_{A,\mathbf{y}}$ recovers its sparsest point.

such that one may recover any signal $\mathbf{x}$, given a smaller set of sample measurements $\mathbf{y} = A\mathbf{x}$. As such a system of equations is underdetermined ($P^{=}_{A,\mathbf{y}}$ is an affine subspace of dimension at least $n-p$), this may not be possible. However, in the case when $\mathbf{x}$ is sparse (as is true in many real-world applications), the sparsity may be exploited to recover $\mathbf{x}$ with accuracy. That is, the sparse signal $\mathbf{x}$ is the unique solution to

$$\text{(P0):}\ \ \min \|\mathbf{x}\|_0 \text{ s.t. } A\mathbf{x} = \mathbf{y},$$

for matrices A satisfying certain properties; see [**BDE09**] and references therein. However, computing the minimizer of (P0) is NP-hard [**Nat95**], so this formulation does not offer an immediate algorithmic advantage. Fortunately, the solution to (P0) coincides with the solution to the ℓ^1-projection problem,

$$\text{(P1):}\ \ \min \|\mathbf{x}\|_1 \text{ s.t. } A\mathbf{x} = \mathbf{y},$$

with a measurement matrix, A, that satisfies an appropriate *restricted isometry property* among other sufficient conditions; see [**CT08**] and references therein. See Figure 1.20 for a visualization of this recovery technique.

One beauty of compressed sensing is that the recovery scheme is naturally algorithmic. First, (P1) may be formulated as an LP, so any LP method is available for these problems; see [**BV04**] and references therein. Besides linear programming techniques, there are also highly successful greedy

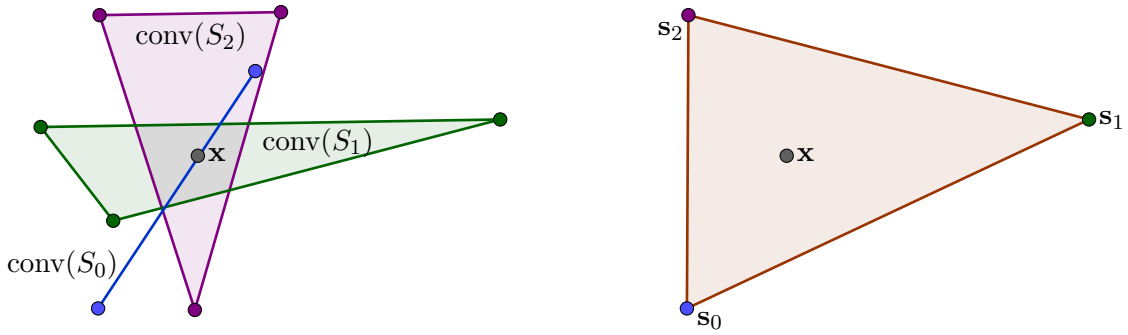

FIGURE 1.21. Left: an example of color sets S_0, S_1, S_2 whose convex hulls contains $\mathbf{x}$; right: an example of a colorful set $\{\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2\}$ whose convex hull contains $\mathbf{x}$.

techniques for signal recovery; see e.g., [**TG07**, **NV09**, **NT09**]. Additionally, there has been much work in the area of ℓ^1-minimization techniques as it has long been known that this technique often recovers sparse solutions; see [**Tib96**], for example.

1.3.1.2. *Colorful Linear Programming.* In 1982, Bárány proved a generalization of Carathéodory's theorem known as the *Colorful Carathéodory's theorem*, which we state below [**Bár82**].

Theorem 1.3.2 (Colorful Carathéodory's theorem)**.** *If* $\mathbf{x} \in \text{conv}(S_i)$ *for* $i = 0, 1, ..., n$ *where* S_i *is a set of points in* $\mathbb{R}^n$*, then there exists a set of* $n+1$ *points each from one of the* $n+1$ *sets* S_i*,* $T = \{\mathbf{s}_0, \mathbf{s}_1, ..., \mathbf{s}_n\}$ *such that* $\mathbf{s}_i \in S_i$*, with* $\mathbf{x} \in \text{conv}(T)$*.*

The sets $S_0, S_1, ..., S_n$ should be considered color classes (with each vertex in the set labelled by the same color). In this setting, the result reveals that there is a colorful set whose convex hull contains $\mathbf{x}$. See Figure 1.21 for an example.

In [**BO97**], Bárány and Onn defined and studied the associated computational problem; given sets of rational points, $S_0, S_1, ..., S_n$, decide if there exists a colorful set, $\{\mathbf{s}_0, \mathbf{s}_1, ..., \mathbf{s}_n\}$, whose convex hull contains $\mathbf{x} \in \mathbb{Q}^n$, and if so, find it. They termed this problem the *Colorful Linear Programming* problem. In this same work, they show that this problem is strongly NP-complete. Much work has been done since then to better understand this problem and generalizations; see [**DHST08**, **ST08**, **DSX11**, **MD13**, **MS16**] and references therein for more information.

However, Bárány and Onn presented an algorithm for *approximating* a solution to this problem which makes use of ℓ^2-projection onto a V-simplex as a subroutine. Without loss of generality, one may assume that $\mathbf{x} = \mathbf{0}$, and define an ϵ-approximation to be a colorful set of points so that the minimum norm point in their convex hull has norm no more than ϵ. Their algorithm proceeds as follows.

Initialize with an arbitrary colorful set, $T_1 = \{\mathbf{s}_0, \mathbf{s}_1, ..., \mathbf{s}_n\}$. Iterate by checking if the minimum norm point in T_k, $\mathbf{x}_k$, has norm less than ϵ; if so, stop and return $\mathbf{x}_k$. If not, then the minimum norm point must lie on a face of the convex hull of T_k and there must be at least one $\mathbf{s}_i$ we can discard so that $\mathbf{x}_k \in \text{conv}(T_k \setminus \{\mathbf{s}_i\})$. Then choose $\mathbf{s} \in S_i$ which minimizes $\mathbf{s}^T\mathbf{x}_k$ and update $T_{k+1} = (T_k \setminus \{\mathbf{s}_i\}) \cup \{\mathbf{s}\}$. Repeat.

This method is closely related to one of von Neumann for linear programming [**Dan92**] as well as to Wolfe's method which is discussed in Section 1.3.2. Note that the main computational cost of the method is the MNP computation in each iteration.

1.3.1.3. *Related Problem: Linear Support Vector Machine.* The support vector machine problem is a supervised learning problem which computes a linear classifier for binary-labeled training data to be used for predicting labels of unlabeled data. This problem and a simple formulation for linearly separable data is presented in Section 1.2.1.3. However, in most applications, one seeks not only a linear classifier that separates the given data, but the linear classifier which provides the largest margin of separation; see Figure 1.22 for an example. This problem is known as *hard-margin SVM* and is formulated as a weighted minimum norm problem over a polyhedron. The problem is to compute the linear classifier given by $\mathbf{w}$ and β which separates the given binary classified data, $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, and minimizes $\|\mathbf{w}\|_2^2$; see [**BV04**] and references therein. This is a weighted minimum norm problem since we are minimizing the norm of a projection of the polyhedron. Using the notation introduced in Section 1.2.1.3, the problem is

$$\min \|[\mathbb{1}^T\ 0]\tilde{\mathbf{w}}\|_2^2 \text{ subject to } \tilde{X}\tilde{\mathbf{w}} \leq -\mathbb{1}.$$

Thus, the problem is to compute the minimum norm point in the projection of the feasible polyhedron into the subspace spanned by the first $n-1$ coordinate vectors.

FIGURE 1.22. A linearly separable SVM problem with two possible linear classifiers.

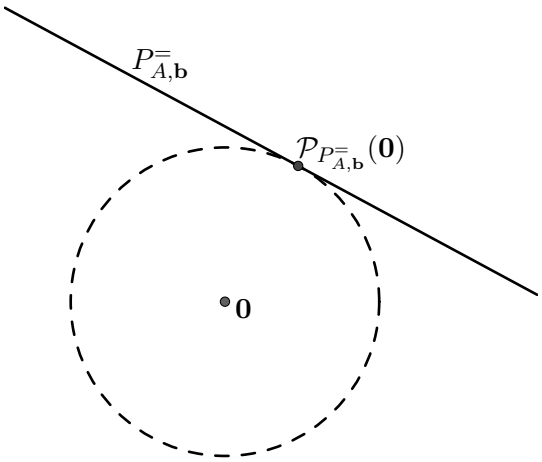

FIGURE 1.23. An example of the least-norm formulation of $A\mathbf{x} = \mathbf{b}$ for an underdetermined system of equations.

1.3.1.4. *Related Problem: Systems of Linear Equations.* As discussed in Section 1.2.1.2, the problem of solving systems of linear equations, $A\mathbf{x} = \mathbf{b}$, arises in nearly every field of science and engineering. There are two natural reformulations of systems of linear equations as minimum norm problems, the least-norm and least-squares formulations.

The *least-norm* formulation computes the point of minimum norm in the feasible polyhedron; $\min \|x\|^2$ subject to $A\mathbf{x} = \mathbf{b}$. Here the feasible polyhedron is an affine subspace of dimension $n - \text{rank}(A)$. Note that this formulation applies only to feasible systems of linear equations, and is most interesting for underdetermined systems. The solution to this problem is also easily computable as it is given by the closed formula $\mathbf{x}^* = A^T(AA^T)^{-1}\mathbf{b}$; see [**BV04**] and references therein. We also include a proof of this well-known result in Chapter 4. See Figure 1.23 for an example of this formulation.

The *least-squares* formulation computes the point which minimizes the norm of the residual, $\|A\mathbf{x}-\mathbf{b}\|$; see [**BV04**] and references therein. This formulation can be viewed as a minimum norm problem on a projection of the feasible region of a lift of the original set of hyperplanes. First, we lift the original set of hyperplanes into higher dimension by introducing slack variables, $\mathbf{y} = A\mathbf{x} - \mathbf{b}$; this allows for nonempty intersection of the higher dimensional affine sets, given by

$$\{\mathbf{z} : \begin{bmatrix} A & -I \end{bmatrix} \mathbf{z} = \mathbf{b}\} \text{ where } \mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}.$$

Now, the problem is reduced to computing a minimum norm problem over a projection of this polyhedron onto the span of the last m coordinate vectors. Concretely, the least-squares problem $\min \|A\mathbf{x} - \mathbf{b}\|^2$ may be equivalently formulated as

$$\min \| \begin{bmatrix} \mathbf{0}^T & \mathbb{1}^T \end{bmatrix} \mathbf{z}\|^2 \text{ subject to } \begin{bmatrix} A & -I \end{bmatrix} \mathbf{z} = \mathbf{b}.$$

1.3.1.5. *Submodular Function Minimization.* Many problems in machine learning (graph-cut problems, such as clustering or image segmentation, and set-covering problems, such as image denoising or sparsity-inducing norms) lead to *submodular* minimization problems, $\min_{A \in 2^V} F(A)$. These are discrete optimization problems defined over subsets of a ground set V with a submodular set function F. A submodular set-function is a function $F : 2^V \to \mathbb{R}$ which satisfies the diminishing-returns property; for all $A \subset B \subset V$ and for all $k \in V \setminus B$,

$$F(A \cup \{k\}) - F(A) \geq F(B \cup \{k\}) - F(B).$$

These functions naturally occur in many applications like economics (e.g., costs of items) and machine learning where they model notions of similarity or diversity. It is easy to see that the *graph-cut* function, which maps subsets of vertices to the weight of the corresponding cut, is submodular. See Figure 1.24 for an example. Many problems in machine learning may be formulated as graph-cut problems. Clustering data points may be formulated as graph-cut problems where the vertices of the graph are the data points and the edges between them are weighted with their distance or a similarity metric. Similarly, image segmentation may be formulated as graph-cut problems

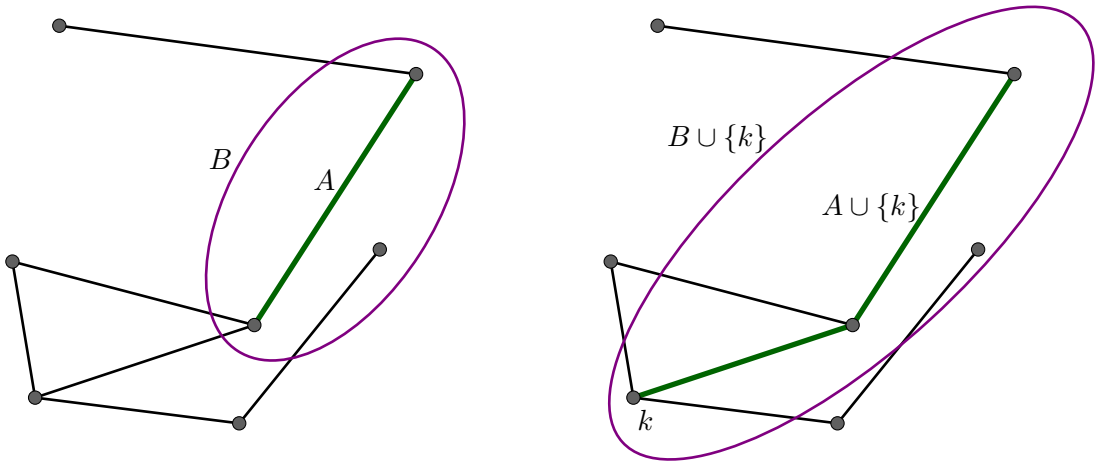

FIGURE 1.24. An example of the submodularity of a graph-cut function. Note that $F(A \cup \{k\}) - F(A) = 4 - 3 \geq F(B \cup \{k\}) - F(B) = 5 - 4$ where $F(S)$ is the cardinality of the cut between vertex set $S \subset V$ and $V \setminus S$ (number of edges).

where the vertices of the graph are pixels of an image and the edges between neighboring pixels are weighted with a pixel similarity-score.

The well-studied Lovász extension of the submodular minimization problem leads to solving the equivalent problem $\min_{\mathbf{y} \in B_F} \|\mathbf{y}\|^2$ where B_F is the base polytope for the submodular function F [**B**$^+$**13**]. Thus, this problem is solved by the ℓ^2-projection of the origin onto the base polytope. Indeed, one method which is currently considered an important practical algorithm for submodular minimization is the Fujishige-Wolfe algorithm [**FI11**, **FHI06**, **CJK14**]. This method uses Wolfe's combinatorial method for projection onto a polytope as a subroutine in solving submodular minimization. We discuss Wolfe's method in the next subsection, and in greater detail in Chapter 3.

1.3.2. Wolfe's Methods for MNP. Wolfe's method is a combinatorial active-set method for solving the minimum norm point problem over a polytope, $P = \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m) \subset \mathbb{R}^n$, introduced by P. Wolfe in [**Wol76**]. The method iteratively solves the MNP problem exactly over a sequence of subsets of no more than $n + 1$ affinely independent points from $\mathbf{p}_1, ..., \mathbf{p}_m$ and checks to see if the

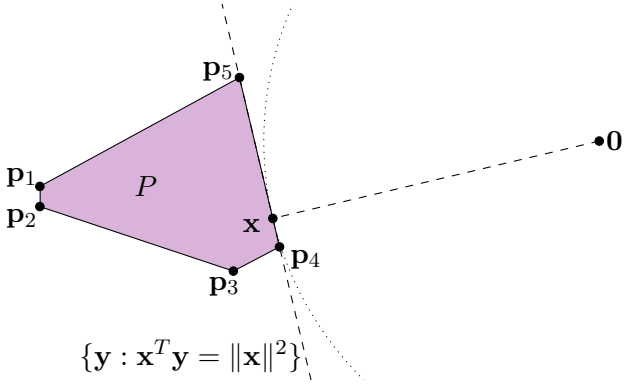

FIGURE 1.25. A visualization of Wolfe's criterion. Note that $\mathbf{p}_i^T\mathbf{x} \geq \|\mathbf{x}\|^2$ for all $i = 1, ..., 5$, so $\mathbf{x}$ is the MNP in P.

solution to the subproblem is a solution to the problem over P using the following lemma due to Wolfe.

Lemma 1.3.3 (Wolfe's criterion [**Wol76**])**.** *Let $P = \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m) \subset \mathbb{R}^n$, then $\mathbf{x} \in P$ is the minimum norm point in P if and only if*

$$\mathbf{x}^T\mathbf{p}_j \geq \|\mathbf{x}\|_2^2 \quad \textit{for all} \quad j \in [m].$$

Note that this tells us that if there exists a point $\mathbf{p}_j$ so that $\mathbf{x}^T\mathbf{p}_j < \|\mathbf{x}\|_2^2$ then $\mathbf{x}$ is not the minimum norm point in P. Geometrically, this means that if the hyperplane $\{\mathbf{y} : \mathbf{x}^T\mathbf{y} = \|\mathbf{x}\|^2\}$ does not weakly-separate all $\mathbf{p}_j$ from the origin, then $\mathbf{x}$ is not the MNP. We say that $\mathbf{p}_j$ violates Wolfe's criterion and adding this point to the current subset of points will decrease the norm of the minimum norm point of the current subproblem. See Figure 1.25 for a visualization.

It should be observed that just as Wolfe's criterion is a rule to decide optimality for the convex minimizer, one has a very similar rule for deciding optimality over the affine hull. We state the result and prove this in Chapter 3 since we do not know of a reference.

Lemma 1.3.4 (Optimality condition for affine minimizer)**.** *Let $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\} \subset \mathbb{R}^n$, then $\mathbf{x} \in \text{aff}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m)$ is the affine minimizer if and only if we have $\mathbf{p}_i^T\mathbf{x} = \|\mathbf{x}\|^2$ for all $i \in [m]$.*

The iteration of Wolfe's method may be informally described as follows: beginning with a set $C \subset \{\mathbf{p}_i\}_{i=1}^m$ and $\mathbf{x}$ the minimum norm point in $\text{conv}(C)$, check if $\mathbf{x}$ is the minimum norm point in

P using Lemma 1.3.3. If not, add $\mathbf{p}_j$ which violates the optimality condition to the set C. Find the minimum norm point in the affine hull of the updated C and check if it is in the convex hull of C. If not, discard old points of C until the affine minimizer of C is the convex minimizer of C. Repeat. This method is described in far greater detail in Chapter 3.

Note that when one adds points to or removes points from the set C, there may be a choice of which point to add or remove. We call the rules which decide which point to add *insertion* rules and will discuss these further in Chapter 3. Here we mention only one insertion rule, the *minnorm rule*, which from the points available to add dictates that one should choose the point of minimum norm. In Chapter 3, we additionally demonstrate that the choice of insertion rule affects the behavior of Wolfe's method and present an example on which two insertion rules have different behavior. We will discuss further in Chapter 3 the fact that *deletion* rules have less effect on the behavior of the algorithm. However, for the purpose of our main results, we follow the general principle that the same deletion rule is used throughout the course of the method. We will refer to Wolfe's method with a specific insertion rule as *Wolfe's algorithm* (e.g., Wolfe's algorithm with the *minnorm* insertion rule).

The complexity of Wolfe's method was not understood since its inception in 1974. Our main result in Chapter 3 gives the first example that Wolfe's method displays exponential behavior. This is akin to the well-known Klee-Minty examples showing exponential behavior for the simplex method [**KM72**]. Prior work by [**CJK14**] showed that after t iterations, Wolfe's method returns an $\mathcal{O}(1/t)$-approximate solution to the minimum norm point on any polytope, while work by [**LJJ15**] showed that t iterations suffices for an $\mathcal{O}(e^{-\rho t})$-approximate solution, where ρ is an eccentricity parameter of the polytope. Both of these provide a pseudo-polynomial complexity bound for Wolfe's method. Our result presents a family of polytopes on which the number of iterations of Wolfe's algorithm with the *minnorm* insertion rule grows exponentially in the dimension and number of input points.

Theorem 1.3.5. *There exists a family of polytopes given as the convex hull of the point sets $P(n)$ such that the execution of Wolfe's algorithm with the* minnorm *point insertion rule on input $P(n)$ where $n = 2k - 1$ has number of iterations at least $5 \cdot 2^{k-1} - 4$.*

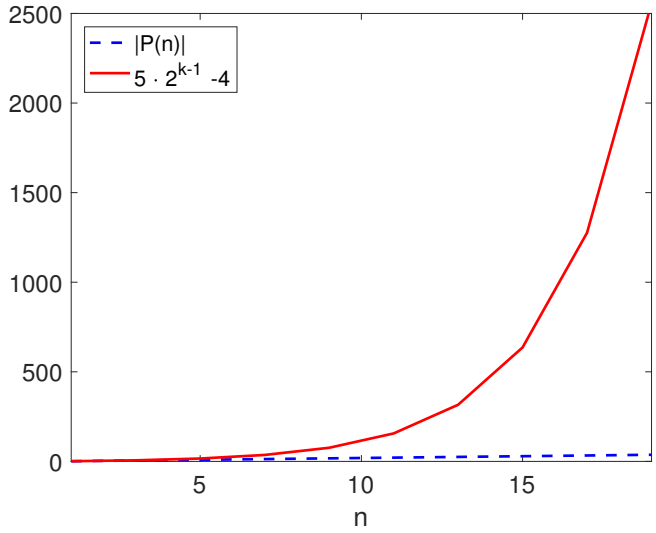

FIGURE 1.26. A plot of the size of $P(n)$ and the iteration lower bound for various dimension n. Note that the size of $P(n)$ grows linearly in n, while the lower bound grows exponentially in n.

We give a detailed description of the point sets $P(n)$ and prove this result in Chapter 3. These sets are of size $2n-1$ and are defined for $n = 2k+1$ where $k \geq 0$. See Figure 1.26 for a plot of the size of $P(n)$ and the number of iterations in the lower bound for various values of n (the dimension).

1.4. How are LF and MNP Related?

It is natural to ask: does this thesis study two unrelated problems whose only connection is that each may be solved by projection-based methods? The answer is resoundingly negative; here (and in more detail in Chapter 4) we illustrate the deep connection between LF and MNP. We reduce linear programming to the minimum norm point problem over a simplex via a series of strongly-polynomial time reductions in Chapter 4. Recall that in Section 1.2.1.1, we presented that LF and LP are strongly-polynomial time equivalent. Definitions for LP were given in Section 1.2.1.1. Here, we give a definition for the problem of distance to a V-simplex (DVS). See [**Sch86**, **GLS88**, **Sch03**] for detailed discussions of strongly-polynomial time reductions.

Definition 1.4.1. *Consider the following computational problem:*

- **DVS:** *Given $d \leq n+1$ affinely independent rational points $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_d \in \mathbb{R}^n$ defining $(d-1)$-dimensional simplex $P = \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_d)$, output $d(\mathbf{0}, P)^2$.*

The main result in Chapter 4 reduces linear programming to finding the minimum norm point in a (vertex-representation) simplex, as finding the minimum norm point also determines the distance $d(\mathbf{0}, P)^2$. It was previously known that LP reduces to MNP over a polytope in weakly-polynomial time [**FHI06**], but our result is stronger as it is a novel strongly-polynomial time reduction to MNP over a simplex.

Theorem 1.4.2. *LP reduces to DVS in strongly-polynomial time.*

The proof of this result may be found in Chapter 4. It is proved via a string of strongly-polynomial time reductions. Most of the reductions are classically known, but we highlight the last reduction as new. This result ties the minimum norm point problem intimately to the question of strong-polynomiality of linear programming, which is one of Stephen Smale's eighteen 21st century problems identified in 1999 [**Sma00**].

CHAPTER 2

Iterative Projection Methods for Linear Feasibility

In this chapter we discuss three methods for solving LF problems; Motzkin's method (MM) and the randomized Kaczmarz method (RK) are classical methods, and we present a generalization of the MM and RK methods which we name the *Sampling Kaczmarz-Motzkin* methods (SKM). We present simple proofs of some slight generalizations of previously known results for MM and RK. In Section 2.1.2, we demonstrate the acceleration of Motzkin's method on a system of linear equations in the presence of a simple condition on the residual. We go on to show that systems of linear equations defined by Gaussian matrices satisfy this residual condition. In Section 2.2.2, we describe methods which use RK to identify and discard corruptions within a system of linear equations. We present probabilistic guarantees that these methods identify all corrupted equations. Our main results in this chapter present the convergence rate of the SKM methods and prove that these methods detect and certify feasibility with high probability (we bound the probability that SKM iterates are not certificates of feasibility).

2.1. Motzkin's Method

The first method for linear feasibility we will discuss is *Motzkin's method*. This method has been re-discovered several times; e.g., the famous 1958 *perceptron* algorithm [**Ros58**] can be thought of as a member of this family of methods. The first analysis of this type of method appeared a few years earlier in 1954, within the work of Agmon [**Agm54**], and Motzkin and Schoenberg [**MS54**]. More recently, Motzkin's method has been referred to as the Kaczmarz method with the "most violated constraint control" or the "maximal-residual control" [**Cen81**, **NSV**$^+$**16**, **PP15**].

This method can be described as follows: starting from any initial point $\mathbf{x}_0$, a sequence of points is generated. If the current point $\mathbf{x}_i$ is feasible we stop, else there must be a constraint $\mathbf{a}^T\mathbf{x} \leq \mathbf{b}$

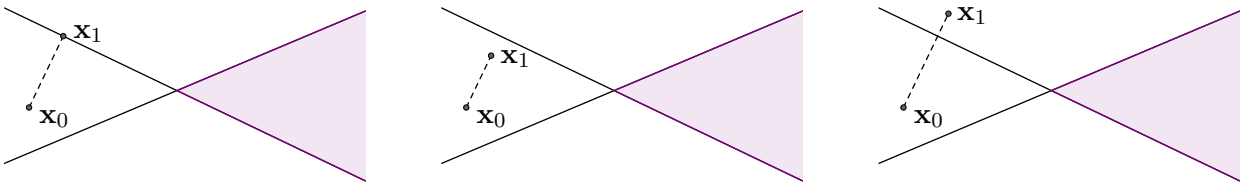

FIGURE 2.1. Three projections with $\lambda = 1$, $\lambda < 1$ and $\lambda > 1$.

that is *most violated*. The constraint defines a halfspace $H^{\leq}_{\mathbf{a},b}$. Let $\mathcal{P}_{H^{\leq}_{\mathbf{a},b}}(\mathbf{x}_i)$ be the orthogonal (ℓ^2) projection of $\mathbf{x}_i$ onto the halfspace $H^{\leq}_{\mathbf{a},b}$, and choose a number λ ($\lambda \in (0, 2]$). The new point $\mathbf{x}_{i+1}$ is given by

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda(\mathcal{P}_{H^{\leq}_{\mathbf{a},b}}(\mathbf{x}_i) - \mathbf{x}_i).$$

Note that the description of Motzkin's method in Section 1.2.2.1 presented the method with *projection parameter*, $\lambda = 1$. Here we consider the more general family of methods which allow for *under* ($\lambda < 1$) and *over-projection* ($\lambda > 1$). Figure 2.1 displays the iteration visually. We present pseudo-code for Motzkin's method in Method 2.1. MATLAB code for this algorithm may be found in Appendix A.

Method 2.1 Motzkin's method

procedure MOTZKIN($A, \mathbf{b}, \mathbf{x}_0, \lambda, k$)
  for $j = 1, 2, ..., k$ **do**
    $\mathbf{x}_j = \mathbf{x}_{j-1} - \lambda \frac{(\mathbf{a}_{i_j}^T \mathbf{x}_{j-1} - b_{i_j})^+}{\|\mathbf{a}_{i_j}\|^2} \mathbf{a}_{i_j}$ where $i_j \in \underset{i \in [m]}{\operatorname{argmax}} \ \mathbf{a}_i^T \mathbf{x}_{j-1} - b_i$.
  end for
  return $\mathbf{x}_k$
end procedure

For clarity, we include an example of several iterations of Motzkin's method on an `LF` in Figure 2.2. Each image illustrates the projection $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ in a dashed line. The lines surrounding the polyhedral feasible region, $P_{A,\mathbf{b}}$ represent the boundary hyperplanes of the halfspaces defining $P_{A,\mathbf{b}}$. The lines are dotted if the constraint defining the corresponding halfspace is satisfied by the iterate $\mathbf{x}_{k-1}$, and solid if the constraint defining the corresponding halfspace is violated by the iterate $\mathbf{x}_{k-1}$. The line representing the hyperplane bounding the halfspace defined by the selected constraint, $\mathbf{a}_{i_k}^T \mathbf{x} \leq b_{i_k}$ is shown in bold.

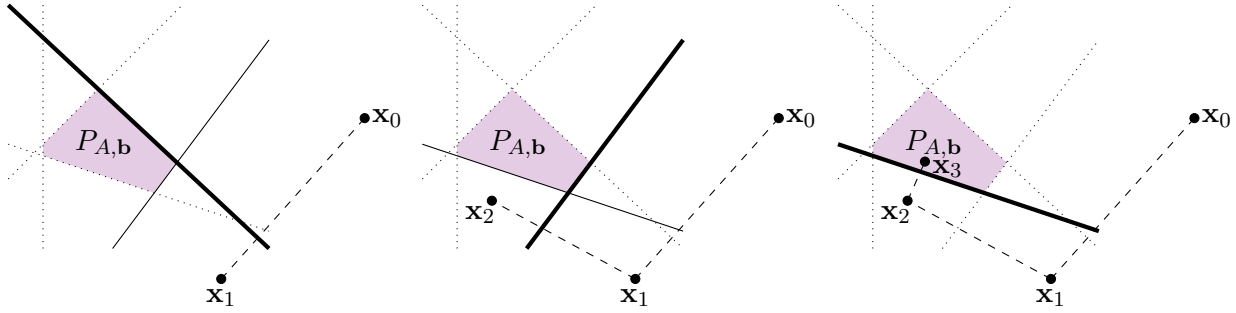

FIGURE 2.2. An example of the iterates formed by Motzkin's method with $\lambda > 1$ on a feasible `LF`.

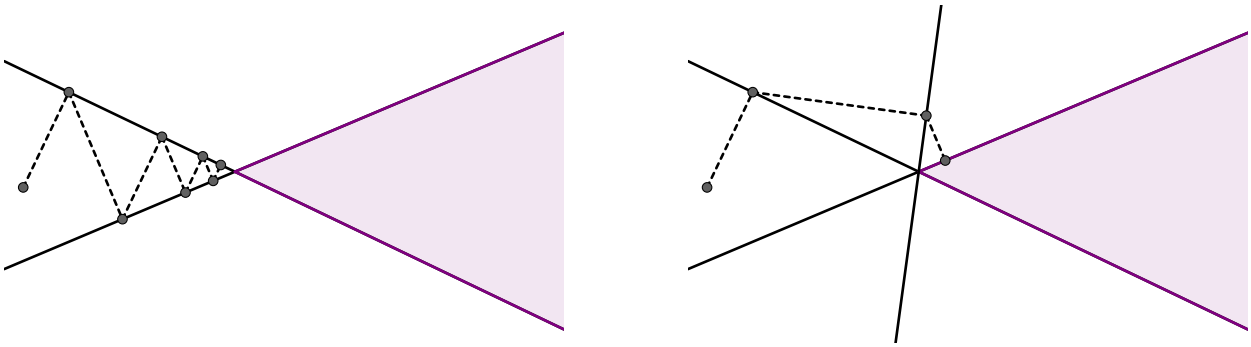

FIGURE 2.3. Left: Projecting onto original hyperplanes. Right: Projecting onto an induced hyperplane (like those in Chubanov's method).

A bad feature of Motzkin's method is that when the system, $A\mathbf{x} \leq \mathbf{b}$, is infeasible, it cannot terminate, as there will always be a violated inequality. In the 1980's, this method was revisited with interest because it can be seen as a member of the same family of algorithms as the ellipsoid method; see [**AH05**, **Bet04**, **Gof80**, **Tel82**] and references therein. One can show that the Motzkin's method is finite in all cases when using rational data, in that it can be modified to detect infeasible systems. In some special cases the method gives a polynomial time algorithm (e.g., for totally unimodular matrices [**MTA81**]), but there are also examples of exponential running times (see [**Gof82**, **Tel82**]). In late 2010, Chubanov [**Chu12**] announced a modification of the traditional method which gives a *strongly-polynomial* time algorithm in some situations [**BDJ14**, **VZ14**]. Unlike [**Agm54**, **MS54**], who only projected onto the original hyperplanes that describe the polyhedron, $P_{A,\mathbf{b}}$, Chubanov [**Chu12**] projects onto new, auxiliary inequalities which are linear combinations of the input. See Figure 2.3 for an example of this process.

2.1.1. Convergence Rate. The rate of convergence of Motzkin's method depends not only on λ, but also on the *Hoffman constants* which were first discussed in Section 1.2.2. In this section, we prove the following generalization of Proposition 1.2.4 which demonstrates that Motzkin's method with projection parameter $\lambda \neq 2$ converges linearly. The first linear convergence rate result for Motzkin's method is due to Agmon [**Agm54**], but we include this simple generalization and its proof for completeness.

Proposition 2.1.1. *Consider a normalized system with $\|\mathbf{a}_i\| = 1$ for all $i = 1, ..., m$. If the feasible region $P_{A,\mathbf{b}}$ is nonempty, then Motzkin's method (Method 2.1) converges linearly:*

$$d(\mathbf{x}_k, P_{A,\mathbf{b}})^2 \leq \left(1 - \frac{2\lambda - \lambda^2}{L_\infty^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2 \leq \left(1 - \frac{2\lambda - \lambda^2}{mL_2^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2.$$

PROOF. As the system has been normalized, note that the method defines $\mathbf{x}_{j+1} = \mathbf{x}_j - \lambda(\mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*})^+\mathbf{a}_{i^*}$ where $i^* \in \underset{i\in[m]}{\operatorname{argmax}} \quad \mathbf{a}_i^T\mathbf{x}_j - b_i$, we have

$$\begin{aligned}
d(\mathbf{x}_{j+1}, P_{A,\mathbf{b}})^2 &= \|\mathbf{x}_{j+1} - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_{j+1})\|^2 \\
&\leq \|\mathbf{x}_{j+1} - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)\|^2 \\
&= \|\mathbf{x}_j - \lambda(\mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*})^+\mathbf{a}_{i^*} - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)\|^2 \\
&= \|\mathbf{x}_j - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)\|^2 + \lambda^2((\mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*})^+)^2\|\mathbf{a}_{i^*}\|^2 - 2\lambda(\mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*})^+\mathbf{a}_{i^*}^T(\mathbf{x}_j - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)),
\end{aligned}$$

where $\mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x})$ denotes the ℓ^2-projection of $\mathbf{x}$ onto $P_{A,\mathbf{b}}$. Since $\mathbf{a}_{i^*}^T(\mathbf{x}_j - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)) \geq \mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*}$ and $\|\mathbf{a}_{i^*}\|^2 = 1$ we have that

$$\begin{aligned}
d(\mathbf{x}_{j+1}, P_{A,\mathbf{b}})^2 &\leq d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 + \lambda^2((\mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*})^+)^2\|\mathbf{a}_{i^*}\|^2 - 2\lambda(\mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*})^+(\mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*})^+ \\
&= d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 - (2\lambda - \lambda^2)((\mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*})^+)^2 \\
&= d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 - (2\lambda - \lambda^2)\|(A\mathbf{x}_j - \mathbf{b})^+\|_\infty^2 \\
&\leq \left(1 - \frac{2\lambda - \lambda^2}{L_\infty^2}\right) d(\mathbf{x}_j, P_{A,\mathbf{b}})^2,
\end{aligned}$$

where the last inequality follows from the definition of the Hoffman constant. With induction, we find that

$$d(\mathbf{x}_k, P_{A,\mathbf{b}})^2 \leq \left(1 - \frac{2\lambda - \lambda^2}{L_\infty^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2.$$

Note the fact that $L_\infty \leq \sqrt{m}L_2$ so we have

$$d(\mathbf{x}_k, P_{A,\mathbf{b}})^2 \leq \left(1 - \frac{2\lambda - \lambda^2}{L_\infty^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2 \leq \left(1 - \frac{2\lambda - \lambda^2}{mL_2^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2.$$

□

Note that when $L_\infty << \sqrt{m}L_2$ this outer bound can be unnecessarily pessimistic in that the appearance of $\sqrt{m}$ is tight only for $\|(A\mathbf{x}-\mathbf{b})^+\|^2 = m\|(A\mathbf{x}-\mathbf{b})^+\|_\infty^2$. In Section 2.1.2, we avoid this potential pessimism by introducing a parameter which bounds the difference between the ℓ^2 and ℓ^∞-norm of the residual which we denote the *dynamic range*. We show that Motzkin's method for systems of linear equations features an initially accelerated convergence rate when the residual has a large dynamic range. We provide bounds for the iterate error which depend on the dynamic range of the residual. These bounds can potentially be used when designing stopping criteria or hybrid approaches. Next, for a concrete example we show that Gaussian systems of linear equations have large dynamic range and provide bounds on this value. We extend this to a corollary which shows that the initial convergence rate is highly accelerated and our theoretical bound closely matches experimental evidence.

2.1.2. Acceleration of Motzkin's Method. In this section, we present the potential acceleration of Motzkin's method on a system of potentially inconsistent linear equations, $A\mathbf{x} = \mathbf{b}$ [**HN18a**]. Throughout this section, we consider Motzkin's method with projection parameter $\lambda = 1$ which is appropriately modified to solve systems of linear equations. For clarity, we include pseudo-code in Method 2.2. MATLAB code for this algorithm may be found in Appendix A.

The advantage of Motzkin's method is that by greedily selecting the most violated constraint, the method makes large moves at each iteration, thereby accelerating convergence. One drawback of course, is that it is computationally expensive to compute which constraint is most violated. For this reason, in Section 2.3, we propose a hybrid batched variant of the method that randomly selects a

Method 2.2 Motzkin's method for $A\mathbf{x} = \mathbf{b}$

1: **procedure** MOTZKINLS($A, \mathbf{b}, \mathbf{x}_0, k$)
2: **for** $j = 1, 2, ..., k$ **do**
3: $\mathbf{x}_j = \mathbf{x}_{j-1} - \frac{\mathbf{a}_{i_j}^T \mathbf{x}_{j-1} - b_{i_j}}{\|\mathbf{a}_{i_j}\|^2} \mathbf{a}_{i_j}$ where $i_j \in \underset{i \in [m]}{\operatorname{argmax}} \ |\mathbf{a}_i^T \mathbf{x}_{j-1} - b_i|$.
4: **end for**
5: **return** $\mathbf{x}_k$
6: **end procedure**

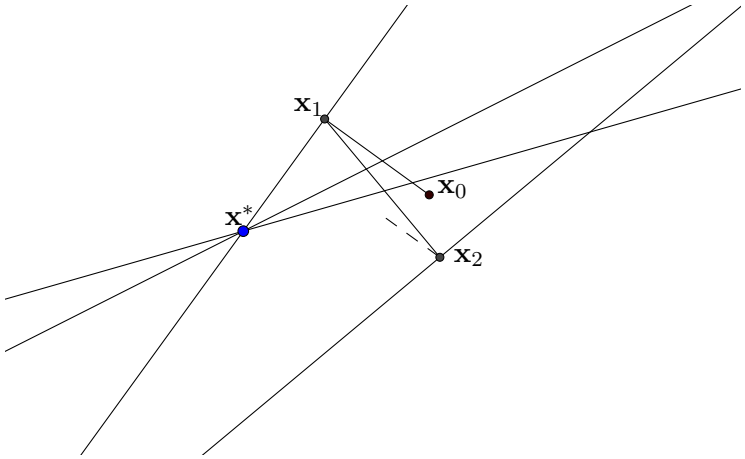

FIGURE 2.4. An example of a series of projections using the Motzkin approach on an inconsistent system. Lines represent the hyperplanes consisting of sets $\{\mathbf{x} : \mathbf{a}_i^T \mathbf{x} = b_i\}$ for rows $\mathbf{a}_i^T$ of A, and $\mathbf{x}^*$ denotes the desired solution.

batch of rows and then computes the most violated from that batch. When the system is inconsistent, however, there is an additional drawback to Motzkin's method because projecting onto the most violated constraint need not move the iterate closer to the desired solution, as already mentioned and shown in Figure 2.4. Our first lemma provides a rule for deciding if a greedy projection offers desirable improvement which depends upon the error of the system of equations, $\mathbf{e} := A\mathbf{x} - \mathbf{b}$.

Lemma 2.1.2. *Let* $\mathbf{x}$ *denote the desired solution of the system given by matrix* A *and right hand side* $\mathbf{b}$ *with* $\|\mathbf{a}_i\| = 1$ *for all* $i = 1, ..., m$. *If* $\mathbf{e} = A\mathbf{x} - \mathbf{b}$ *and* $\|A\mathbf{x}_k - \mathbf{b}\|_\infty > 4\|\mathbf{e}\|_\infty$ *then the next iterate,* $\mathbf{x}_{k+1}$ *defined by Motzkin's method with* $\lambda = 1$ *(Method 2.2) satisfies*

$$\|\mathbf{x}_{k+1} - \mathbf{x}\|^2 \leq \|\mathbf{x}_k - \mathbf{x}\|^2 - \frac{1}{2}\|A\mathbf{x}_k - \mathbf{b}\|_\infty^2.$$

PROOF. By definition of $\mathbf{x}_{k+1}$, we have

$$\begin{aligned}
\|\mathbf{x}_{k+1}-\mathbf{x}\|^2 &= \|\mathbf{x}_k-\mathbf{x}\|^2 - 2(\mathbf{a}_{i^*}^T\mathbf{x}_k - b_{i^*})(\mathbf{a}_{i^*}^T\mathbf{x}_k - b_{i^*} - e_{i^*}) + (\mathbf{a}_{i^*}^T\mathbf{x}_k - b_{i^*})^2 \\
&= \|\mathbf{x}_k-\mathbf{x}\|^2 - (\mathbf{a}_{i^*}^T\mathbf{x}_k - b_{i^*})^2 + 2(\mathbf{a}_{i^*}^T\mathbf{x}_k - b_{i^*})e_{i^*} \\
&\leq \|\mathbf{x}_k-\mathbf{x}\|^2 - (\mathbf{a}_{i^*}^T\mathbf{x}_k - b_{i^*})^2 + 2|\mathbf{a}_{i^*}^T\mathbf{x}_k - b_{i^*}| \cdot |e_{i^*}| \\
&= \|\mathbf{x}_k-\mathbf{x}\|^2 - \|A\mathbf{x}_k-\mathbf{b}\|_\infty^2 + 2\|A\mathbf{x}_k-\mathbf{b}\|_\infty|e_{i^*}| \\
&\leq \|\mathbf{x}_k-\mathbf{x}\|^2 - \|A\mathbf{x}_k-\mathbf{b}\|_\infty^2 + 2\|A\mathbf{x}_k-\mathbf{b}\|_\infty\|\mathbf{e}\|_\infty \qquad (2.1) \\
&\leq \|\mathbf{x}_k-\mathbf{x}\|^2 - \frac{1}{2}\|A\mathbf{x}_k-\mathbf{b}\|_\infty^2.
\end{aligned}$$

□

Note that this tells us that, while our residual is still large relative to the error, Motzkin's method can offer good progress in each iteration. Also, this progress is better than the expected progress offered by Randomized Kaczmarz (RK) when the residual has good dynamic range, in particular when:

$$\frac{1}{2}\|A\mathbf{x}_k-\mathbf{b}\|_\infty^2 > \frac{1}{m}\|A\mathbf{x}_k-\mathbf{b}\|^2.$$

We can use Lemma 2.1.2 to easily obtain the following corollary.

Corollary 2.1.3. *Let* $\mathbf{x}$ *denote the desired solution of the system given by matrix* A *and right hand side* $\mathbf{b}$ *with* $\|\mathbf{a}_i\| = 1$ *for all* $i = 1, ..., m$ *and write* $\mathbf{e} = A\mathbf{x} - \mathbf{b}$ *as the error term. For any given iteration* k*, the iterate defined by Motzkin's method with* $\lambda = 1$ *(Method 2.2) satisfies either (i) or both (ii) and (iii), where*

$$\begin{aligned}
&\textit{(i)} \quad \|\mathbf{x}_{k+1}-\mathbf{x}\|^2 \leq \|\mathbf{x}_k-\mathbf{x}\|^2 - \frac{1}{2}\|A\mathbf{x}_k-\mathbf{b}\|_\infty^2, \\
&\textit{(ii)} \quad \|\mathbf{x}_k-\mathbf{x}\|^2 \leq 17m\sigma_{\min}^{-2}(A)\|\mathbf{e}\|_\infty^2, \\
&\textit{(iii)} \quad \|\mathbf{x}_{k+1}-\mathbf{x}\|^2 \leq \left(17m\sigma_{\min}^{-2}(A) + 8\right)\|\mathbf{e}\|_\infty^2.
\end{aligned}$$

In addition, if the method is run with stopping criterion $\|A\mathbf{x}_k - \mathbf{b}\|_\infty \leq 4\|\mathbf{e}\|_\infty$*, then the method exhibits the (possibly highly accelerated) convergence rate*

$$\|\mathbf{x}_T - \mathbf{x}\|^2 \leq \prod_{k=0}^{T-1} \left(1 - \frac{\sigma_{\min}^2(A)}{4\gamma_k}\right) \cdot \|\mathbf{x}_0 - \mathbf{x}\|^2 + 2m\sigma_{\min}^{-2}(A)\|\mathbf{e}\|_\infty^2, \tag{2.2}$$

$$\leq \left(1 - \frac{\sigma_{\min}^2(A)}{4m}\right)^T \|\mathbf{x}_0 - \mathbf{x}\|^2 + 2m\sigma_{\min}^{-2}(A)\|\mathbf{e}\|_\infty^2, \tag{2.3}$$

with final error satisfying (ii). Here γ_k *bounds the dynamic range of the* k*th residual,* $\gamma_k := \frac{\|A\mathbf{x}_k - A\mathbf{x}\|^2}{\|A\mathbf{x}_k - A\mathbf{x}\|_\infty^2}$.

Proof. We consider two cases, depending on whether $\|A\mathbf{x}_k - \mathbf{b}\|_\infty > 4\|\mathbf{e}\|_\infty$ or $\|A\mathbf{x}_k - \mathbf{b}\|_\infty \leq 4\|\mathbf{e}\|_\infty$. If the former holds, then (i) is valid by Lemma 2.1.2. If instead the latter holds, then we first obtain (ii) by the simple argument

$$\begin{aligned}
\|\mathbf{x}_k - \mathbf{x}\|^2 &\leq \sigma_{\min}^{-2}(A)\|A\mathbf{x}_k - A\mathbf{x}\|^2 \\
&\leq \sigma_{\min}^{-2}(A)m\|A\mathbf{x}_k - A\mathbf{x}\|_\infty^2 \\
&\leq \sigma_{\min}^{-2}(A)m\left(\|A\mathbf{x}_k - \mathbf{b}\|_\infty^2 + \|\mathbf{e}\|_\infty^2\right) \\
&\leq \sigma_{\min}^{-2}(A)m\left(16\|\mathbf{e}\|_\infty^2 + \|\mathbf{e}\|_\infty^2\right) \\
&= 17m\sigma_{\min}^{-2}(A)\|\mathbf{e}\|_\infty^2.
\end{aligned}$$

To obtain (iii) still in this latter case, we continue from (2.1) showing

$$\begin{aligned}
\|\mathbf{x}_{k+1} - \mathbf{x}\|^2 &\leq \|\mathbf{x}_k - \mathbf{x}\|^2 - \|A\mathbf{x}_k - \mathbf{b}\|_\infty^2 + 2\|A\mathbf{x}_k - \mathbf{b}\|_\infty\|\mathbf{e}\|_\infty \\
&\leq 17m\sigma_{\min}^{-2}(A)\|\mathbf{e}\|_\infty^2 - \|A\mathbf{x}_k - \mathbf{b}\|_\infty^2 + 2\|A\mathbf{x}_k - \mathbf{b}\|_\infty\|\mathbf{e}\|_\infty \\
&\leq 17m\sigma_{\min}^{-2}(A)\|\mathbf{e}\|_\infty^2 + 2\|A\mathbf{x}_k - \mathbf{b}\|_\infty\|\mathbf{e}\|_\infty \\
&\leq 17m\sigma_{\min}^{-2}(A)\|\mathbf{e}\|_\infty^2 + 8\|\mathbf{e}\|_\infty^2 \\
&= \left(17m\sigma_{\min}^{-2}(A) + 8\right)\|\mathbf{e}\|_\infty^2.
\end{aligned}$$

To prove (2.2) and (2.3), we first note that by choice of stopping criterion, (i) holds for all $0 \leq k \leq T$. Thus for all such k, we have

$$\begin{aligned}
\|\mathbf{x}_k - \mathbf{x}\|^2 &\leq \|\mathbf{x}_{k-1} - \mathbf{x}\|^2 - \frac{1}{2}\|A\mathbf{x}_{k-1} - \mathbf{b}\|_\infty^2 \\
&= \|\mathbf{x}_{k-1} - \mathbf{x}\|^2 - \frac{1}{2}\|(A\mathbf{x}_{k-1} - A\mathbf{x}) - \mathbf{e}\|_\infty^2 \\
&\leq \|\mathbf{x}_{k-1} - \mathbf{x}\|^2 - \frac{1}{4}\|A\mathbf{x}_{k-1} - A\mathbf{x}\|_\infty^2 + \frac{1}{2}\|\mathbf{e}\|_\infty^2 \qquad (2.4) \\
&= \|\mathbf{x}_{k-1} - \mathbf{x}\|^2 - \frac{1}{4\gamma_{k-1}}\|A\mathbf{x}_{k-1} - A\mathbf{x}\|^2 + \frac{1}{2}\|\mathbf{e}\|_\infty^2 \\
&\leq \|\mathbf{x}_{k-1} - \mathbf{x}\|^2 - \frac{\sigma_{\min}^2(A)}{4\gamma_{k-1}}\|\mathbf{x}_{k-1} - \mathbf{x}\|^2 + \frac{1}{2}\|\mathbf{e}\|_\infty^2 \\
&= \left(1 - \frac{\sigma_{\min}^2(A)}{4\gamma_{k-1}}\right)\|\mathbf{x}_{k-1} - \mathbf{x}\|^2 + \frac{1}{2}\|\mathbf{e}\|_\infty^2, \qquad (2.5)
\end{aligned}$$

where the first line follows from (i), the third from Jensen's inequality, and the fifth from the facts that $\|\mathbf{x}\|^2 \leq \|A^{-1}\|^2\|A\mathbf{x}\|^2$ and $\|A^{-1}\| = 1/\sigma_{\min}(A)$.

Iterating the relation given by (2.5) recursively yields (we use the convention that an empty sum or product equates to one)

$$\begin{aligned}
\|\mathbf{x}_T - \mathbf{x}\|^2 &\leq \prod_{k=0}^{T-1}\left(1 - \frac{\sigma_{\min}^2(A)}{4\gamma_k}\right)\cdot\|\mathbf{x}_0 - \mathbf{x}\|^2 + \sum_{j=0}^{T-1}\prod_{k=0}^{j-1}\left(1 - \frac{\sigma_{\min}^2(A)}{\gamma_k}\right)\frac{1}{2}\|\mathbf{e}\|_\infty^2 \\
&\leq \prod_{k=0}^{T-1}\left(1 - \frac{\sigma_{\min}^2(A)}{4\gamma_k}\right)\cdot\|\mathbf{x}_0 - \mathbf{x}\|^2 + \sum_{j=0}^{T-1}\left(1 - \frac{\sigma_{\min}^2(A)}{4m}\right)^j\frac{1}{2}\|\mathbf{e}\|_\infty^2 \\
&\leq \prod_{k=0}^{T-1}\left(1 - \frac{\sigma_{\min}^2(A)}{4\gamma_k}\right)\cdot\|\mathbf{x}_0 - \mathbf{x}\|^2 + 2m\sigma_{\min}^{-2}(A)\|\mathbf{e}\|_\infty^2 \\
&\leq \left(1 - \frac{\sigma_{\min}^2(A)}{4m}\right)^T\|\mathbf{x}_0 - \mathbf{x}\|^2 + 2m\sigma_{\min}^{-2}(A)\|\mathbf{e}\|_\infty^2,
\end{aligned}$$

where the second and fourth inequalities follow from the simple bound $\gamma_k \leq m$ and the third by bounding above by the infinite sum. The last two inequalities complete the proof of (2.2) and (2.3). □

2.1.2.1. *Experimental Results.* We note that the convergence rate given by (2.2) yields a significant improvement over that of RK when the dynamic range of many residuals is large, i.e., when

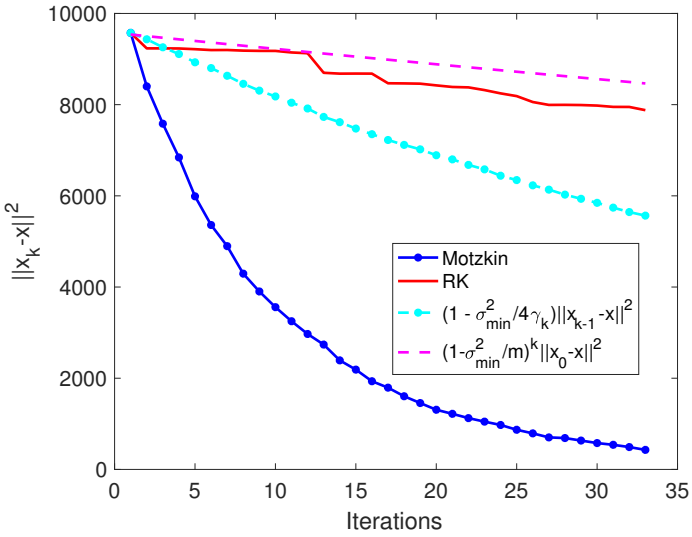

FIGURE 2.5. Convergence of Motzkin's method and RK on correlated system with corresponding theoretical bounds.

$\gamma_k \ll m$ for many iterations k. In Figure 2.5, we present the convergence of Motzkin and RK on a random system defined by matrix A with $a_{ij} \sim \mathcal{N}(1, 0.5)$, and the corresponding theoretical bounds. Figure 2.6 presents plots providing the convergence of Motzkin and RK, and the corresponding theoretical bounds on systems of equations defined by problems from the *Netlib* linear programming benchmark set [**Net**]. These problems contain naturally underdetermined systems, which we transform into overdetermined, inconsistent systems with nearly the same least-squares solution. We transform the problem, originally given by the underdetermined systems of equations $A\mathbf{x} = \mathbf{b}$ by adding equations to form

$$\begin{bmatrix} A \\ I \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{b} \\ \mathbf{x}_{LS} + \epsilon \end{bmatrix},$$

where $\mathbf{x}_{LS}$ is the least-norm solution of $A\mathbf{x} = \mathbf{b}$ and ϵ is a Gaussian vector with small variance. Each problem has very small error which is distributed relatively uniformly, thus there are many iterations in which the theoretical bounds hold. These plots are only for the iterations before the stopping criterion is met.

This acceleration is in force until the stopping criterion given in the corollary. This bound therefore, can be used to design such stopping criteria; one could design an approach for example that utilizes

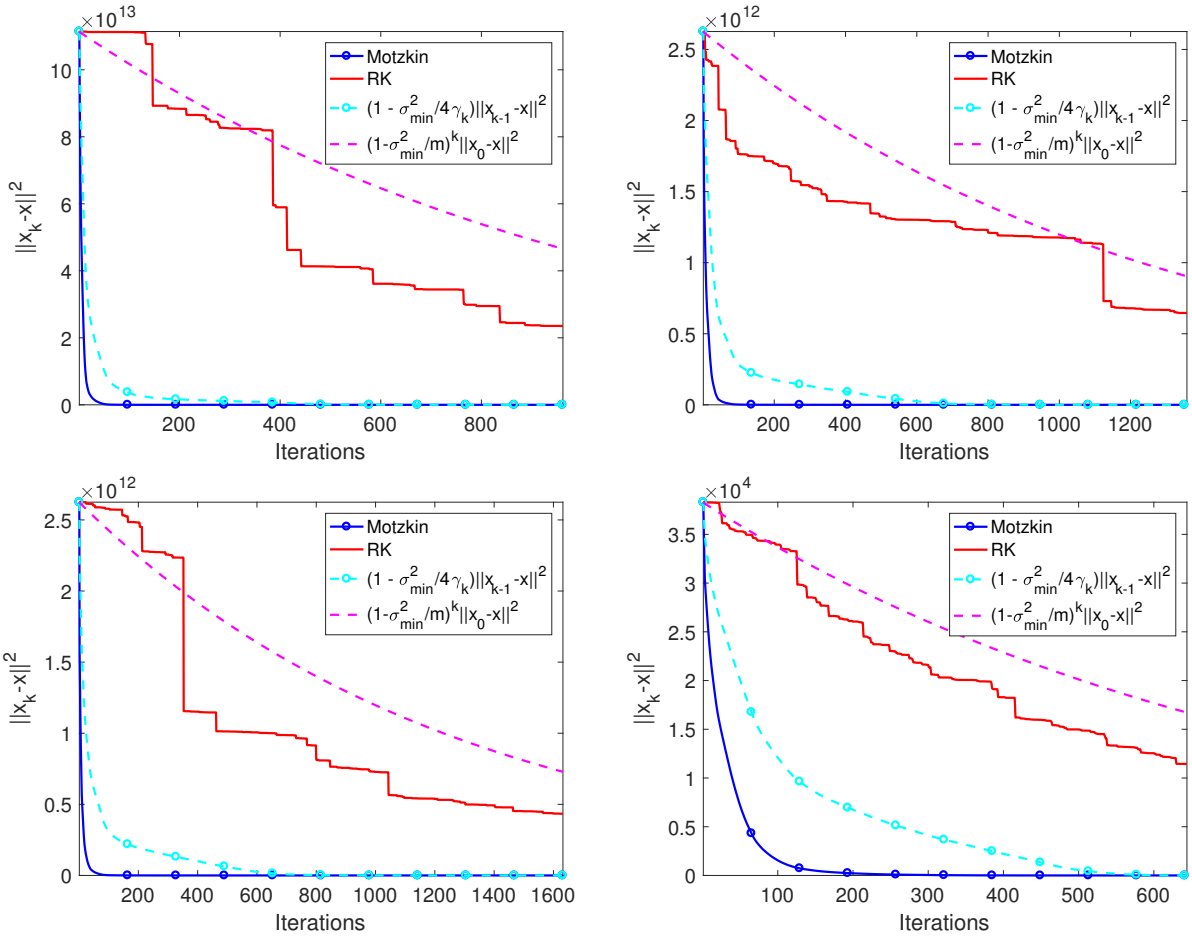

FIGURE 2.6. Convergence of Motzkin's method and RK, and corresponding theoretical bounds for *Netlib* linear programming problems. Upper left: `agg`; upper right: `agg2`; lower left: `agg3`; lower right: `bandm`.

Motzkin's method until reaching this threshold, and then switching to the traditional RK selection strategy to reduce the convergence horizon. In Figure 2.7, we see that Motzkin outperforms RK for the initial iterations (while $\|A\mathbf{x}_k - \mathbf{b}\|_\infty \gg \|\mathbf{e}\|_\infty$) on a system with Gaussian noise. However, for a system with sparse, large magnitude error, Motzkin does not perform as well in the long run, as it suffers from a worse *convergence horizon* than RK.

To capitalize on this accelerated convergence, one needs knowledge of an upper bound $\|\mathbf{e}\|_\infty \leq \beta$, in which case the stopping criterion of $\|A\mathbf{x}_k - \mathbf{b}\|_\infty \leq 4\beta$ guarantees the accelerated convergence of (2.2) and a final error of $\|\mathbf{x}_k - \mathbf{x}\|^2 \leq 25m\sigma_{\min}^{-2}(A)\beta^2$. Indeed, one quickly verifies that when

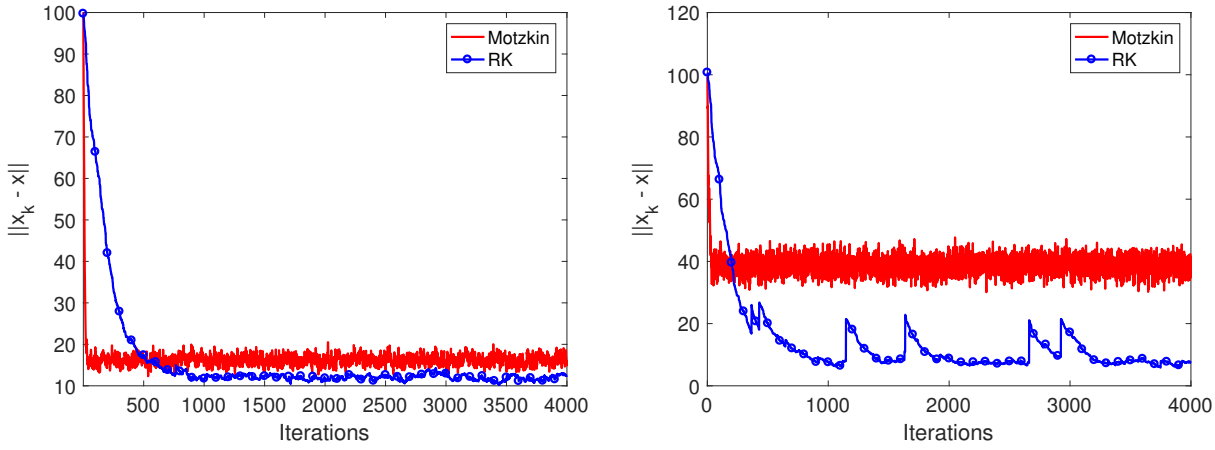

FIGURE 2.7. Left: Motzkin's method vs. RK distance from least-squares solution for a Gaussian system with Gaussian noise. Right: Motzkin's method vs. RK distance from least-squares solution for a Gaussian system with sparse, 'spiky' noise.

$\|A\mathbf{x}_k - \mathbf{b}\|_\infty \leq 4\beta$, we have

$$\begin{aligned}
\|\mathbf{x}_k - \mathbf{x}\| &\leq \sigma_{\min}^{-1}(A)\|A\mathbf{x}_k - A\mathbf{x}\| \\
&\leq \sqrt{m}\sigma_{\min}^{-1}(A)\|A\mathbf{x}_k - A\mathbf{x}\|_\infty \\
&\leq \sqrt{m}\sigma_{\min}^{-1}(A)\left(\|A\mathbf{x}_k - \mathbf{b}\|_\infty + \|\mathbf{e}\|_\infty\right) \\
&\leq \sqrt{m}\sigma_{\min}^{-1}(A)\left(4\beta + \beta\right).
\end{aligned}$$

Since the acceleration of the method occurs when many of the terms γ_k are small, we plot an example in Figure 2.8. As expected, many terms are bounded away from m. We will analyze this in the Gaussian case further below.

We also only expect this acceleration to be present while the condition of Lemma 2.1.2 is in force (i.e., prior to the stopping condition given in the corollary). Once the condition of Lemma 2.1.2 is no longer satisfied, selecting greedily chooses those entries of the residual which have large contribution from the error, moving the estimation far from the desired solution. While the difference between greedy selection and randomized selection is not so drastic for Gaussian noise, it will be drastically different for sparse error. We include an example system in Figure 2.10 to assist with intuition. Again, one could of course implement the Kaczmarz approach after an initial use of Motzkin's as a

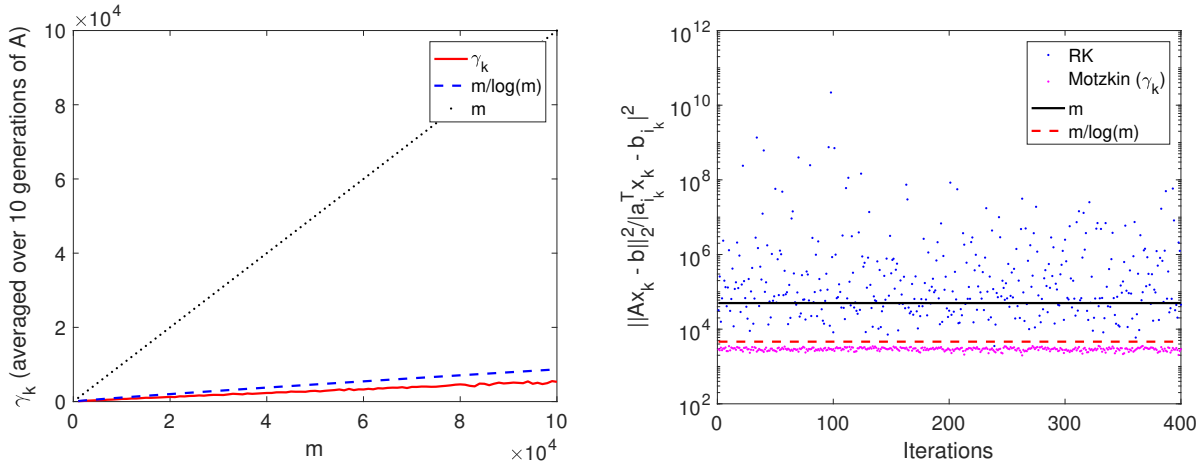

FIGURE 2.8. Left: Average γ_k values for choice of row dimension, m, of normalized Gaussian $A \in \mathbb{R}^{m\times 100}$. Right: Example γ_k values for iterations of Motzkin's method and the corresponding ratio for RK; $A \in \mathbb{R}^{50000\times 100}$ Gaussian. The index i_k chosen in the kth iteration. Horizontal lines denote the values m and $m/\log(m)$.

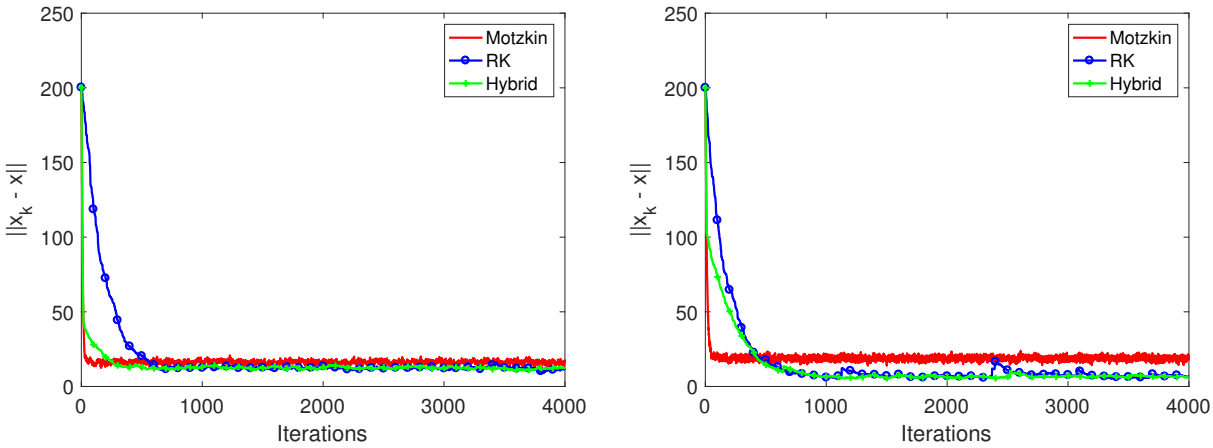

FIGURE 2.9. Left: Motzkin's method, RK, and hybrid distance from least-squares solution for a Gaussian system with Gaussian noise. Right: Motzkin's method, RK, and hybrid distance from least-squares solution for a Gaussian system with sparse, 'spiky' noise.

strategy to gain acceleration without sacrificing convergence horizon. In Figure 2.9, we present the convergence of Motzkin's method, the RK method, and a hybrid method which consists of Motzkin iterations until $\|A\mathbf{x}_k - \mathbf{b}\|_\infty \leq 4\|\mathbf{e}\|_\infty$, followed by RK iterations. Again we include results on both a system with Gaussian error and a system with a sparse, 'spiky' error.

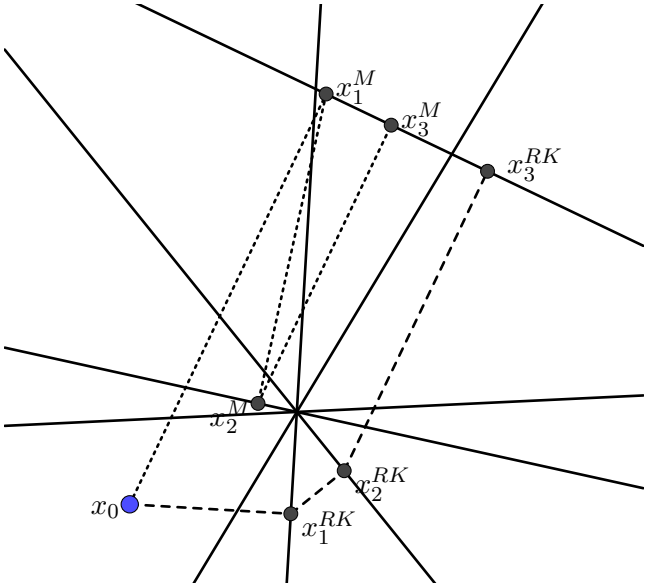

FIGURE 2.10. An example of three iterations of Motzkin's method ($\mathbf{x}_k^M$) and three iterations of RK ($\mathbf{x}_k^{RK}$) on a Gaussian system with sparse, 'spiky' error. More of the RK iterations are near the least squares solution while Motzkin consistently selects the corrupted equation.

2.1.2.2. *Heuristics for Gaussians.* Here, we study heuristics for our convergence results for the Gaussian matrix case. Note that our results hold for matrices with normalized rows. For simplicity however, we will consider an $m \times n$ matrix whose entries are i.i.d. Gaussian with mean 0 and variance $1/n$. We will then assume we are in the asymptotic regime where this distribution approximates a Gaussian matrix with normalized unit-norm rows. This can be readily verified by observing that the distribution of $\mathbf{a}_i$ is rotationally invariant and thus $\left(\mathbf{a}_i^T \frac{\mathbf{x}}{\|\mathbf{x}\|}\right)^2$ has the same distribution as $\left(\mathbf{a}_i^T \mathbf{e}_1\right)^2$, where $\mathbf{e}_1$ is the first coordinate vector. Thus it has the same distribution as the ratio of chi-square random variables $g_1^2 / \sum_{i=1}^n g_i^2$, for i.i.d. standard normal g_i. One then applies Slutsky's theorem to obtain the asymptotic result. To that end, we assume m and n both grow linearly with respect to one another, and that they are both substantially large.

Define I_k to be the rows of A that are independent from $\mathbf{x}_k$ and note that $I_k \subseteq I_{k-1} \subseteq ... \subseteq I_1 \subseteq I_0 = [m]$. Fix iteration k and define $m' = m - |I_k|$. Note that $m - k \leq m' \leq m$ is the dimension of the sub-matrix whose rows are independent of the iterates up to iteration k. Throughout this section $\mathbb{P}$ and $\mathbb{E}$ refer to probability and expectation taken with respect to the random and unsampled portion of the matrix A, A_{I_k}, which has m' rows.

Our first lemma gives a bound on the expected dynamic range for a Gaussian matrix.

Lemma 2.1.4. *If $A \in \mathbb{R}^{m\times n}$ is a Gaussian matrix with $a_{ij} \sim \mathcal{N}(0, 1/n)$ and $\mathbf{x}$ is independent of at least m' rows of A (e.g., constructed via k iterations of Motzkin's method) then*

$$\frac{\mathbb{E}\|A\mathbf{x}\|^2}{\mathbb{E}\|A\mathbf{x}\|_\infty^2} \lesssim \frac{m' + \sum_{i\notin I_k} \|\mathbf{a}_i\|^2}{\log(m')}.$$

Proof. First note that

$$\begin{aligned}
\mathbb{E}(\sum_{i=1}^{m}(\mathbf{a}_i^T\mathbf{x})^2) &= \sum_{i=1}^{m}\mathbb{E}(\mathbf{a}_i^T\mathbf{x})^2 \\
&\le \sum_{i=1}^{m}\mathbb{E}(\|\mathbf{a}_i\|^2\|\mathbf{x}\|^2) && \text{by Cauchy-Schwarz} \\
&\le \sum_{i\in I_k}\mathbb{E}(\|\mathbf{a}_i\|^2\|\mathbf{x}\|^2) + \sum_{i\notin I_k}\|\mathbf{a}_i\|^2\|\mathbf{x}\|^2 \\
&= (m' + \sum_{i\notin I_k}\|\mathbf{a}_i\|^2)\|\mathbf{x}\|^2.
\end{aligned}$$

Next, note that if $\mathbf{a}_i$ and $\mathbf{x}$ are independent then $\mathbf{a}_i^T\mathbf{x} \sim \mathcal{N}(0, \|\mathbf{x}\|^2)$. Then

$$\begin{aligned}
\mathbb{E}(\max_{i\in[m]}(\mathbf{a}_i^T\mathbf{x})^2) &\ge \mathbb{E}(\max_{i\in I_k}(\mathbf{a}_i^T\mathbf{x})^2) \\
&\ge \mathbb{E}(\max_{i\in I_k}\mathbf{a}_i^T\mathbf{x})^2 \\
&\ge (\mathbb{E}\max_{i\in I_k}\mathbf{a}_i^T\mathbf{x})^2 && \text{by Jensen's inequality} \\
&\ge c\|\mathbf{x}\|^2\log(m'),
\end{aligned}$$

as it is commonly known that $\mathbb{E}(\max_{i\in[N]} X_i) \ge c\sigma\sqrt{logN}$ for $X_i \sim \mathcal{N}(0, \sigma^2)$. Thus, we have

$$\mathbb{E}\|A\mathbf{x}\|_\infty^2 \ge c\|\mathbf{x}\|^2\log(m') \ge c\frac{\log(m')}{m' + \sum_{i\notin I_k}\|\mathbf{a}_i\|^2}\mathbb{E}\|A\mathbf{x}\|^2.$$

□

We can use this lemma along with our main result to obtain the following.

Corollary 2.1.5. *Let $A \in \mathbb{R}^{m\times n}$ be a normalized Gaussian matrix as described previously, $\mathbf{x}$ denote the desired solution of the system given by matrix A and right hand side $\mathbf{b}$, write $\mathbf{e} = A\mathbf{x} - \mathbf{b}$ as the error term and assume $\mathbf{x}_0$ is chosen so that $\mathbf{x}_0 - \mathbf{x}$ is independent of the rows of A, $\mathbf{a}_i^T$. If Motzkin's method with $\lambda = 1$ (Method 2.2) is run with stopping criterion $\|A\mathbf{x}_k - \mathbf{b}\|_\infty \leq 4\|\mathbf{e}\|_\infty$, in expectation the method exhibits the accelerated convergence rate*

$$\mathbb{E}\|\mathbf{x}_{k+1} - \mathbf{x}\|^2 \lesssim \mathbb{E}\left[\left(1 - \frac{\log(m')\sigma_{\min}^2(A)}{4m}\right)\|\mathbf{x}_k - \mathbf{x}\|^2 + \frac{1}{2}\|\mathbf{e}\|_\infty^2\right]. \tag{2.6}$$

Proof. Beginning from line (2.4) of the proof of Corollary 2.1.3 and taking expectation of both sides, we have

$$\begin{aligned}
\mathbb{E}\|\mathbf{x}_{k+1} - \mathbf{x}\|^2 &\leq \mathbb{E}\|\mathbf{x}_k - \mathbf{x}\|^2 - \frac{1}{4}\mathbb{E}\|A(\mathbf{x}_k - \mathbf{x})\|_\infty^2 + \frac{1}{2}\mathbb{E}\|\mathbf{e}\|_\infty^2 \\
&\lesssim \mathbb{E}\|\mathbf{x}_k - \mathbf{x}\|^2 - \frac{\log(m')}{4(m' + \sum_{i\notin I_k}\|\mathbf{a}_i\|^2)}\mathbb{E}\|A(\mathbf{x}_k - \mathbf{x})\|^2 + \frac{1}{2}\mathbb{E}\|\mathbf{e}\|_\infty^2 \\
&= \mathbb{E}\left[\|\mathbf{x}_k - \mathbf{x}\|^2 - \frac{\log(m')}{4m}\|A\mathbf{x}_k - A\mathbf{x}\|^2 + \frac{1}{2}\|\mathbf{e}\|_\infty^2\right] \\
&\leq \mathbb{E}\left[\left(1 - \frac{\log(m')\sigma_{\min}^2(A)}{4m}\right)\|\mathbf{x}_k - \mathbf{x}\|^2 + \frac{1}{2}\|\mathbf{e}\|_\infty^2\right],
\end{aligned}$$

where the second inequality follows from Lemma 2.1.4 and the fourth from properties of singular values. □

This corollary implies a logarithmic improvement in the convergence rate, at least initially. Of course, we conjecture that the $\log(m')$ term in (2.6) is an artifact of the proof and could actually be replaced with $\log(m)$. This is supported by the experiments shown in Figures 2.8 and 2.11. Furthermore, Corollary 5.35 of [**Ver12**] provides a lower bound for the size of the smallest singular value of A with high probability, $\mathbb{P}\left(\sigma_{\min}(A) \leq \sqrt{m/n} - 1 - t/\sqrt{n}\right) \leq 2e^{-t^2/2}$. That is, asymptotically $\sigma_{\min}(A)$ is tightly centered around $\sqrt{m/n} - 1$.

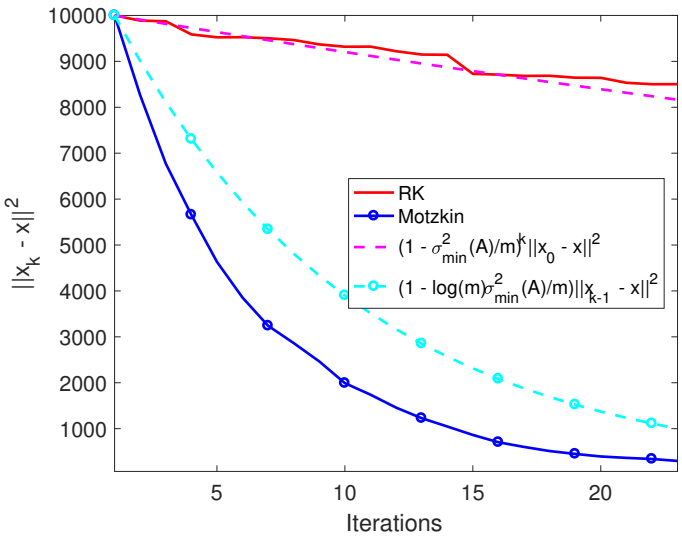

FIGURE 2.11. Convergence of Motzkin's method and RK on Gaussian system with corresponding theoretical bounds.

2.2. Randomized Kaczmarz Method

The second method for linear feasibility we will discuss is the Kaczmarz method [**Kac37**, **GBH70**] which is one of the most popular solvers of overdetermined systems of linear equations due to its speed and simplicity. Just like Motzkin's, it is an iterative method which consists of a series of alternating orthogonal projections onto the halfspaces defined by the system of inequalities. The original Kaczmarz method simply cycles through the inequalities sequentially, so its convergence rate depends on the order of the rows. One way to overcome this is to use the inequalities in a *random order*, rather than sequentially [**HS78**, **HM93**, **Nat01**]. More precisely, we begin with $A\mathbf{x} \leq \mathbf{b}$, a linear system of inequalities where A is an $m \times n$ matrix with rows $\mathbf{a}_i^T$ and $\mathbf{x}_0$ an initial guess. For $k = 0, 1, 2, ...$ one defines

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \lambda \frac{(\mathbf{a}_i^T \mathbf{x}_k - b_i)^+}{\|\mathbf{a}_i\|^2} \mathbf{a}_i,$$

where i is chosen from $\{1, 2, ..., m\}$ at random, say with probability proportional to $\|\mathbf{a}_i\|^2$, and $\lambda \in (0, 2]$. Thus, $\mathbf{x}_k$ lies on the ray which begins at $\mathbf{x}_{k-1}$ and extends through $\mathcal{P}_{H_{\mathbf{a}_i, b_i}}(\mathbf{x}_{k-1})$. Pseudo-code for the randomized Kaczmarz method (as stated in [**SV09**]) is provided in Method 2.3. Note that the description of RK in Section 1.2.2.2 presented the method with *projection parameter*,

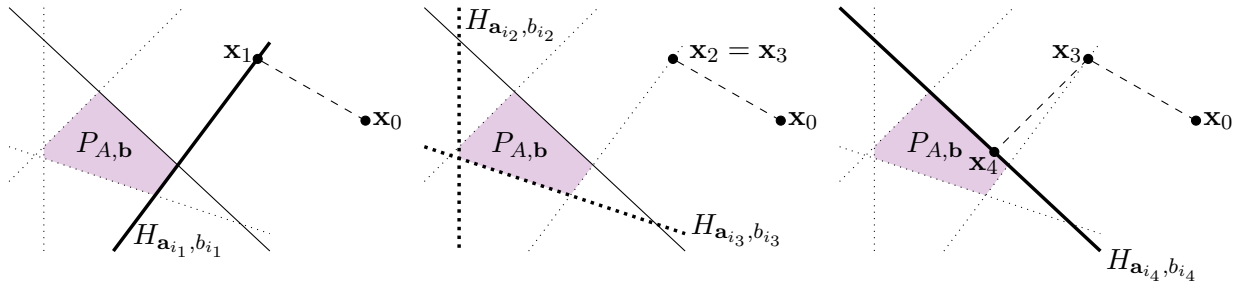

FIGURE 2.12. An example of the iterates formed by the randomized Kaczmarz method with $\lambda = 1$ on a feasible LF. Note that the middle image represents two iterations of RK where the randomly selected constraints were satisfied by the previous iterate.

$\lambda = 1$. Here we consider the more general family of methods which allow for under ($\lambda < 1$) and over-projection ($\lambda > 1$). MATLAB code for this algorithm may be found in Appendix A.

Method 2.3 Randomized Kaczmarz method

1: **procedure** RK($A, \mathbf{b}, \mathbf{x}_0, \lambda, k$)
2: **for** $j = 1, 2, ..., k$ **do**
3: $\mathbf{x}_j = \mathbf{x}_{j-1} - \lambda \frac{(\mathbf{a}_{i_j}^T \mathbf{x}_{j-1} - b_{i_j})^+}{\|\mathbf{a}_{i_j}\|^2} \mathbf{a}_{i_j}$ where $i_j = t \in [m]$ with probability proportional to $\|\mathbf{a}_t\|^2$.
4: **end for**
5: **return** $\mathbf{x}_k$
6: **end procedure**

For clarity, we include an example of several iterations of the randomized Kaczmarz method on an LF in Figure 2.12. Each image illustrates the projection $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ in a dashed line. The lines surrounding the polyhedral feasible region, $P_{A,\mathbf{b}}$ represent the boundary hyperplanes of the halfspaces defining $P_{A,\mathbf{b}}$. The lines are dotted if the constraint defining the corresponding halfspace is satisfied by the iterate $\mathbf{x}_{k-1}$, and solid if the constraint defining the corresponding halfspace is violated by the iterate $\mathbf{x}_{k-1}$. The line representing the hyperplane bounding the halfspace defined by the randomly selected constraint, $\mathbf{a}_{i_k}^T \mathbf{x} \leq b_{i_k}$ is shown in bold and labeled $H_{\mathbf{a}_{i_k},b_{i_k}}$.

The work of Strohmer and Vershynin [**SV09**] sparked a new interest in the Kaczmarz approach and there have been many recent developments in the method and its analysis. Needell [**Nee10**] extended this work to the case of inconsistent systems of equations, showing exponential convergence down to some fixed *convergence horizon*, see also [**WAL15**]. In order to break this convergence

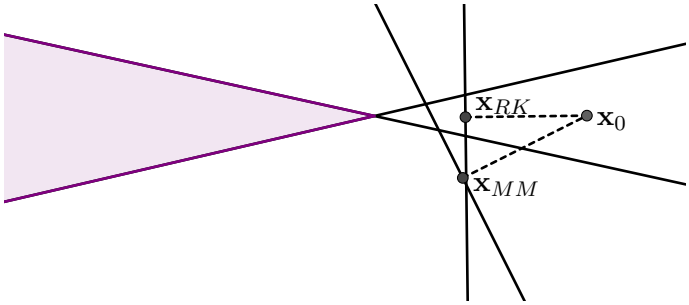

FIGURE 2.13. An example of the iterates defined by RK, $\mathbf{x}_{RK}$, and Motzkin's method, $\mathbf{x}_{MM}$. Note that $\mathbf{x}_{RK}$ is nearer to the feasible region than $\mathbf{x}_{MM}$, despite being constructed via a random selection.

horizon, one needs to modify the Kaczmarz method since by design it projects exactly onto a given hyperplane. Zouzias and Freris [**ZF13**] analyzed an extended randomized Kaczmarz method which incorporates an additional projection step to reduce the size of the residual. This was extended to the block case in [**NZZ15**]. The relation of these approaches to coordinate descent and gradient descent methods has also been recently studied, see e.g., [**GO12**,**Dum14**,**NSW14**,**OZ15**,**MNR15**, **HNR15**, **OZ15**, **GR15**].

Other variations to the Kaczmarz method include block methods [**Elf80**, **EHL81**, **NW13**, **NT13**, **BN**, **XZ02**] which have been shown to offer acceleration for certain systems of equations with fast-multipliers. Other acceleration and convergence schemes focus on sampling selections [**AWL14**, **EN11**, **NSW14**, **OZ17**], projection parameters [**WM67**, **CEG83**, **Tan71**, **HN90**], adding row directions [**PPKR12**], parallelized implementations [**LWS14**, **ADG14**], structure exploiting approaches [**LW15**, **LMY16**], and the use of preconditioning [**GPS16**]. Some other references on recent work include [**CP12**, **RT12**].

Now, one could hope that the distance remaining to the polyhedral feasible region is always decreased more by the deterministic, greedy strategy of Motzkin than by the randomized selection of the randomized Kaczmarz method, however this is not always the case. Consider the example in Figure 2.13.

2.2.1. Convergence Rate. Strohmer and Vershynin [**SV09**] provided an elegant convergence analysis of the randomized Kaczmarz method for consistent equations. Later, Leventhal and

Lewis [**LL10**] extended the probabilistic analysis from systems of equations to systems of linear inequalities. They focused on giving bounds on the convergence rate that take into account the numerical conditions captured by the Hoffman constants L_∞ and L_2. If one additionally makes use of a projection parameter, $\lambda \neq 1$, you can easily extend the convergence rate in [**LL10**] to account for this:

Proposition 2.2.1. *If the feasible region, $P_{A,\mathbf{b}}$, is nonempty then the Randomized Kaczmarz method (Method 2.3) with projection parameter λ converges linearly in expectation:*

$$\mathbb{E}[d(x_k, P_{A,\mathbf{b}})^2] \leq \left(1 - \frac{2\lambda - \lambda^2}{\|A\|_F^2 L_2^2}\right)^k d(x_0, P_{A,\mathbf{b}})^2.$$

PROOF. By the definition of the iterate $\mathbf{x}_{j+1}$, we have

$$\begin{aligned} d(\mathbf{x}_{j+1}, P_{A,\mathbf{b}})^2 &= \|\mathbf{x}_{j+1} - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_{j+1})\|^2 \leq \|\mathbf{x}_{j+1} - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)\|^2 \\ &= \left\| \mathbf{x}_j - \lambda \frac{(\mathbf{a}_{i_{j+1}}^T \mathbf{x}_j - b_{i_{j+1}})^+}{\|\mathbf{a}_{i_{j+1}}\|^2} \mathbf{a}_{i_{j+1}} - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j) \right\|^2 \\ &= \|\mathbf{x}_j - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)\|^2 + \lambda^2 \frac{((\mathbf{a}_{i_{j+1}}^T \mathbf{x}_j - b_{i_{j+1}})^+)^2}{\|\mathbf{a}_{i_{j+1}}\|^4} \|\mathbf{a}_{i_{j+1}}\|^2 \\ &\qquad - 2\lambda \frac{(\mathbf{a}_{i_{j+1}}^T \mathbf{x}_j - b_{i_{j+1}})^+}{\|\mathbf{a}_{i_{j+1}}\|^2} \mathbf{a}_{i_{j+1}}^T (\mathbf{x}_j - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)), \end{aligned}$$

where $\mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x})$ denotes the ℓ^2-projection of $\mathbf{x}$ onto $P_{A,\mathbf{b}}$. Since $\mathbf{a}_{i_{j+1}}^T (\mathbf{x}_j - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)) \geq \mathbf{a}_{i_{j+1}}^T \mathbf{x}_j - b_{i_{j+1}}$ we have that

$$\begin{aligned} d(\mathbf{x}_{j+1}, P_{A,\mathbf{b}})^2 &\leq d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 + \lambda^2 \frac{((\mathbf{a}_{i_{j+1}}^T \mathbf{x}_j - b_{i_{j+1}})^+)^2}{\|\mathbf{a}_{i_{j+1}}\|^2} - 2\lambda \frac{((\mathbf{a}_{i_{j+1}}^T \mathbf{x}_j - b_{i_{j+1}})^+)^2}{\|\mathbf{a}_{i_{j+1}}\|^2} \\ &= d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 - (2\lambda - \lambda^2) \frac{((\mathbf{a}_{i_{j+1}}^T \mathbf{x}_j - b_{i_{j+1}})^+)^2}{\|\mathbf{a}_{i_{j+1}}\|^2}. \end{aligned}$$

Now, note that the i_{j+1}st constraint is chosen with probability $\|\mathbf{a}_{i_{j+1}}\|^2 / \|A\|_F^2$, so taking expectation of both sides with respect to the choice of the i_{j+1}st constraint (and conditioning upon all previous

row selections) yields

$$\begin{aligned}\mathbb{E}_{j+1}[d(\mathbf{x}_{j+1}, P_{A,\mathbf{b}})^2] &\leq \sum_{i=1}^{m} \frac{\|\mathbf{a}_i\|^2}{\|A\|_F^2} d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 - (2\lambda - \lambda^2) \sum_{i=1}^{m} \frac{\|\mathbf{a}_i\|^2}{\|A\|_F^2} \frac{((\mathbf{a}_i^T \mathbf{x}_j - b_i)^+)^2}{\|\mathbf{a}_i\|^2} \\ &= d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 - (2\lambda - \lambda^2) \frac{\|(A\mathbf{x}_j - \mathbf{b})^+\|^2}{\|A\|_F^2} \\ &\leq d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 - (2\lambda - \lambda^2) \frac{d(\mathbf{x}_j, P_{A,\mathbf{b}})^2}{\|A\|_F^2 L_2^2},\end{aligned}$$

where the last inequality follows from the definition of the Hoffman constant, L_2. Iterating this inequality yields

$$\mathbb{E}[d(\mathbf{x}_k, P_{A,\mathbf{b}})^2] \leq \left(1 - \frac{2\lambda - \lambda^2}{\|A\|_F^2 L_2^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2,$$

where $\mathbb{E}$ is with respect to all k random selections of constraints. □

Note the similarities between Propositions 2.1.1 and 2.2.1: the convergence rate constants are identical for normalized systems ($\|A\|_F^2 = m$).

2.2.2. A Kaczmarz-Type Approach for Corruption. We consider solving highly overdetermined, large-scale systems of linear equations represented by a matrix $A \in \mathbb{R}^{m \times n}$ and vector $\mathbf{b} \in \mathbb{R}^m$ with $m \gg n$ [**HN17**, **HN18b**]. Note that the system may not have a solution, so one may seek the least squares solution $\mathbf{x}_{\text{LS}}$; many efficient solvers have been developed that converge to such a solution. An alternative setting is one where there is a solution $\mathbf{x}^*$ (which we refer to as the *pseudo-solution*) to our desired system $A\mathbf{x} = \mathbf{b}^*$, but rather than observing $\mathbf{b}^*$ we only have access to a corrupted version, $\mathbf{b}$, where $\mathbf{b} = \mathbf{b}^* + \mathbf{b}_C$. When the number of non-zero entries in $\mathbf{b}_C$, denoted $s := \|\mathbf{b}_C\|_0$, is small relative to m, one may still hope to recover the "true" solution $\mathbf{x}^*$. For simplicity, we define some general notation to be used throughout this subsection. Let $I \subset [m]$ be the set of indices of inconsistent equations, i.e., $\text{supp}(\mathbf{b}_C) = I$ and $s = |I| \ll m$. We refer to the amount of corruption in each index of I by $\epsilon_i \in \mathbb{R}$, so $\mathbf{b}_C = \sum_{i \in I} \epsilon_i \mathbf{e}_i$. We let ϵ^* be the smallest magnitude of the corruption vector, $\epsilon^* := \min_{i \in I} |\epsilon_i|$. We will also use A_* to refer to the matrix A without the rows indexed by I, $A_* = A_{I^C}$, and likewise for $\mathbf{b}_*$. Note then that $\mathbf{b}_* := \mathbf{b}_{I^C} = \mathbf{b}^*_{I^C}$. Formally, given matrix A and right hand side vector $\mathbf{b}$ corrupted as described

above, we are searching for $\mathbf{x}^*$ given by:

$$(\mathbf{b}_C, \mathbf{x}^*) = \mathbf{argmin}_{\mathbf{b}_C, \mathbf{x}} \|\mathbf{b}_C\|_0 \quad \text{such that} \quad A\mathbf{x} = \mathbf{b} - \mathbf{b}_C. \tag{2.7}$$

Note that if for all corrupted equations, $i \in I$, we have $\mathbf{a}_i$ is not parallel to any affine subspace of dimension at least one defined by more than $m - 2s - 1$ equations of the system $A\mathbf{x} = \mathbf{b}$, then the solution to (2.7) coincides with the pseudo-solution. In particular, for $\mathbf{a}_i$ in general position, this holds. For convenience, we assume throughout this subsection that $\|\mathbf{a}_i\| = 1$ for $i \in [m]$.

It is important to point out that solving for the pseudo-solution of systems $A\mathbf{x} = \mathbf{b} = \mathbf{b}^* + \mathbf{b}_C$ where s is small relative to m is a special case of the problem `MAX-FS`. This type of sparse corruption models many applications, ranging from medical imaging to sensor networks and error correcting codes. For example, a small number of sensors may malfunction, resulting in large catastrophic reporting errors in the vector $\mathbf{b}$; since the reporting errors themselves may be arbitrarily large, the least squares solution is far from the desired solution, but since the number of such reporting errors is small, we may still hope to recover the true solution to the uncorrupted system. We emphasize that such a pseudo-solution $\mathbf{x}^*$ may be very far from the least squares solution $\mathbf{x}_{\text{LS}}$ when the entries in $\mathbf{b}_C$ are large, even when there are only a few non-zero corruptions; see Figure 2.14 for a visual. Similar types of sparse errors may also appear in medical imaging from artifacts or system malfunctions, or in error correcting codes from transmission errors. Indeed, the problem of so-called *sparse recovery* is well-studied in the approximation and compressed sensing literature [**FR13**, **EK12**]. However, in this subsection, we are concerned with the setting where the system is highly *overdetermined*, the errors in $\mathbf{b}$ are sparse and large, and the system may be so large-scale that it cannot be fully loaded into memory. This latter property has sparked a recent resurgence of work in the area of iterative solvers that do not need access to the entire system at once [**GHJ75**, **HLL78**, **Nat01**, **SV09**]. Our work is motivated by such iterative methods and will make use of the randomized Kaczmarz method whose pseudo-code appears in Method 2.4. MATLAB code for this algorithm may be found in Appendix A.

It is known that the randomized Kaczmarz method converges for systems $A\mathbf{x} = \mathbf{b}$ corrupted by noise with an error threshold dependent on A and the noise. In [**Nee10**] it was shown that this

Method 2.4 Randomized Kaczmarz method for $A\mathbf{x} = \mathbf{b}$

procedure RKLS($A, \mathbf{b}, \mathbf{x}_0, k$)
  for $j = 1, 2, ..., k$ **do**
    $\mathbf{x}_j = \mathbf{x}_{j-1} - \frac{\mathbf{a}_{i_j}^T \mathbf{x}_{j-1} - b_{i_j}}{\|\mathbf{a}_{i_j}\|^2} \mathbf{a}_{i_j}$ where $i_j = t \in [m]$ with probability proportional to $\|\mathbf{a}_t\|^2$.
  end for
  return $\mathbf{x}_k$
end procedure

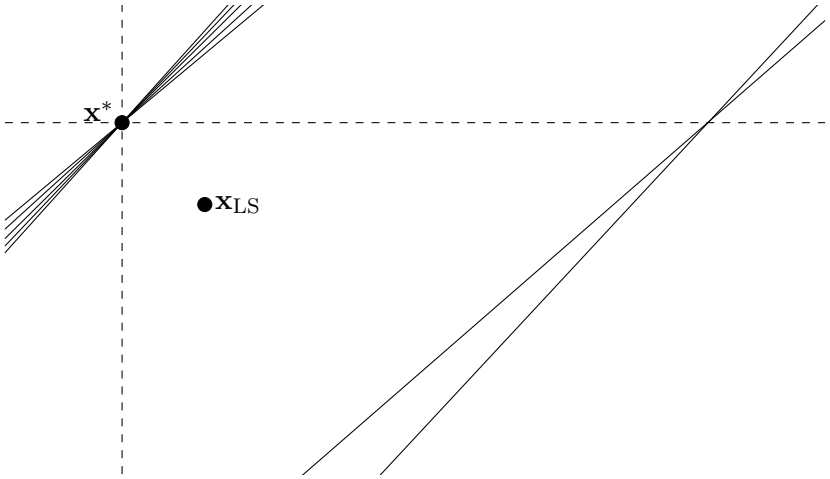

FIGURE 2.14. A system for which the pseudo-solution $\mathbf{x}^\star$ is very far from the least squares solution $\mathbf{x}_{\text{LS}}$. Lines represent the hyperplanes consisting of all systems $\{\mathbf{x} : \mathbf{a}_i^T \mathbf{x} = b_i\}$ for rows $\mathbf{a}_i^T$ of A.

method has iterates that satisfy:

$$\|\mathbf{x}_k - \mathbf{x}_{\text{LS}}\|^2 \leq \left(1 - \frac{\sigma_{\min}^2(A)}{\|A\|_F^2}\right)^k \|\mathbf{x}_0 - \mathbf{x}_{\text{LS}}\|^2 + \frac{\|A\|_F^2}{\sigma_{\min}^2(A)} \|\mathbf{e}\|_\infty^2, \tag{2.8}$$

where $\sigma_{\min}(A)$ denotes the minimum singular value of A, $\|A\|_F$ its Frobenius norm, $\mathbf{x}_{\text{LS}}$ the least squares solution and $\mathbf{e} = \mathbf{b} - A\mathbf{x}_{\text{LS}}$ denotes the error term (also known as the residual). There are variants of this method that converge to the least squares solution [**CEG83**, **ZF13**], however these typically either require operations on the columns or unknown relaxation parameters. However, we are now interested in using randomized Kaczmarz for infeasible systems in which the least-squares solution is unsatisfactory because it is far from satisfying most of the equations (e.g., the noise is sparse and large). The intuition behind our proposed approaches is simple. Since the number of corruptions is small, most iterates of an RK approach will be close to the pseudo-solution, since

it is rare to project onto a corrupted hyperplane. Therefore, if we run the RK method several times, or for several *windows* of time, most of the iterates upon which we halt will be close to the pseudo-solution. Such iterates will also have the property that the largest components of the absolute value of their residual, $|A\mathbf{x}_k - \mathbf{b}|$, will correspond to the large corrupted entries. We can thus utilize this knowledge to gradually detect the corruptions, remove them from the system, and solve for the desired pseudo-solution.

Our proposed methods can thus be described as follows. Each method consists of W windows of k RK iterations beginning with $\mathbf{x}_0 = \mathbf{0}$. In each window, we collect the d indices of the largest magnitude residual entries and after all windows, we solve the system without the rows of A indexed by these collected indices (there may be as many as dW rows removed). The methods differ in two ways. First, we can choose to remove d rows within each window (resulting in Method 2.5 below), or simply collect these indices and remove all collected rows after the W windows (resulting in Methods 2.6 and 2.7 below). Second, when waiting to remove the rows until after W windows, we may simply select the d largest residual entries in each window (Method 2.6), or we may require that the selected indices are always unique (so exactly dW rows are removed), resulting in Method 2.7. The values W, k and d are all parameters of the methods. We give theoretical results for various natural choices of these parameters. MATLAB code for each of these algorithms may be found in Appendix A.

Method 2.5 Windowed Kaczmarz with Removal

procedure WKwR($A, \mathbf{b}, k, W, d$)
  $B = A$, $\mathbf{c} = \mathbf{b}$
  for $i = 1, 2, ...W$ **do**
    $\mathbf{x}_k^i = RKLS(B, \mathbf{c}, \mathbf{0}, k)$
    $D = \text{argmax}_{D\subset[B],|D|=d} \sum_{j\in D} |B\mathbf{x}_k^i - \mathbf{c}|_j$.
    $B = B_{D^C}$, $\mathbf{c} = \mathbf{c}_{D^C}$
  end for
  return $\mathbf{x}$, where $B\mathbf{x} = \mathbf{c}$
end procedure

Method 2.6 Windowed Kaczmarz without Removal

procedure WKwoR($A, \mathbf{b}, k, W, d$)
  $S = \emptyset$
  for $i = 1, 2, ...W$ **do**
    $\mathbf{x}_k^i = RKLS(A, \mathbf{b}, \mathbf{0}, k)$
    $D = \text{argmax}_{D \subset [A], |D|=d} \sum_{j \in D} |A\mathbf{x}_k^i - \mathbf{b}|_j$.
    $S = S \cup D$
  end for
  return $\mathbf{x}$, where $A_{S^C}\mathbf{x} = \mathbf{b}_{S^C}$
end procedure

Method 2.7 Windowed Kaczmarz without Removal with Unique Selection

procedure WKwoRUS($A, \mathbf{b}, k, W, d$)
  $S = \emptyset$
  for $i = 1, 2, ...W$ **do**
    $\mathbf{x}_k^i = RKLS(A, \mathbf{b}, \mathbf{0}, k)$
    $D = \text{argmax}_{D \subset [A]-S, |D|=d} \sum_{j \in D} |A\mathbf{x}_k^i - \mathbf{b}|_j$.
    $S = S \cup D$
  end for
  return $\mathbf{x}$, where $A_{S^C}\mathbf{x} = \mathbf{b}_{S^C}$
end procedure

2.2.2.1. *Main Results.* Our theoretical results provide a lower bound for the probability of successfully removing all corrupted equations after performing Method 2.6 or Method 2.7 with natural values for k, d and W. Lemma 2.2.2 shows that there is a *detection horizon* around the pseudo-solution, so that if $\|\mathbf{x} - \mathbf{x}^*\|$ is sufficiently small, the largest residual entries (of $|A\mathbf{x} - \mathbf{b}|$) correspond exactly to the corrupted equations and we may distinguish these equations from the consistent system. Lemma 2.2.3 gives a value of k so that after k iterations of randomized Kaczmarz, one can give a nonzero lower bound on the probability that the current iterate is within the detection horizon. Theorems 2.2.4 and 2.2.5 then give lower bounds on the probability of successfully detecting all corrupted equations in one out of all W windows for Methods 2.6 and 2.7, respectively.

Lemma 2.2.2. *If $\|\mathbf{x}-\mathbf{x}^*\| < \frac{1}{2}\epsilon^*$ we have that the $d \leq s$ indices of largest magnitude residual entries are contained in I; that is for*

$$D = \underset{D\subset[A],|D|=d}{\operatorname{argmax}} \sum_{i\in D} |A\mathbf{x}-\mathbf{b}|_i$$

we have $D \subset I$.

PROOF. Suppose $\|\mathbf{x}-\mathbf{x}^*\| < \frac{1}{2}\epsilon^*$. Note that for $\|\mathbf{a}_i\| = 1$, we have

$$|r_i| = |A\mathbf{x}-\mathbf{b}|_i = |\mathbf{a}_i^T\mathbf{x}-b_i| = \frac{|\mathbf{a}_i^T\mathbf{x}-b_i|}{\|\mathbf{a}_i\|} = d(\mathbf{x},H_i)$$

where $d(\mathbf{x},H)$ is the Euclidean distance of $\mathbf{x}$ to the set H and $H_i = \{\mathbf{x} : \mathbf{a}_i^T\mathbf{x} = b_i\}$ is the hyperplane defined by the ith equation. Next, note that

$$d(\mathbf{x}^*,H_i) = |\mathbf{a}_i^T\mathbf{x}^*-b_i| = |b_i^*-b_i| = \begin{cases} |\epsilon_i| & i\in I \\ 0 & i\notin I \end{cases}.$$

Now, consider r_i for $i \in I$. Denoting by P_H the orthogonal projectiong onto H, note that

$$\begin{aligned} |r_i| = d(\mathbf{x},H_i) &= \|P_{H_i}(\mathbf{x})-\mathbf{x}\| = \|P_{H_i}(\mathbf{x})-\mathbf{x}^*-(\mathbf{x}-\mathbf{x}^*)\| \\ &\geq |\|P_{H_i}(\mathbf{x})-\mathbf{x}^*\| - \|\mathbf{x}-\mathbf{x}^*\|| \\ &\geq d(\mathbf{x}^*,H_i) - \|\mathbf{x}-\mathbf{x}^*\| > \frac{1}{2}\epsilon^*, \end{aligned}$$

where the first inequality follows from the triangle inequality and the second from the fact that $\|P_{H_i}(\mathbf{x})-\mathbf{x}^*\| \geq d(\mathbf{x}^*,H_i) = |\epsilon_i| \geq \epsilon^* > \|\mathbf{x}-\mathbf{x}^*\|$.

For $i \notin I$, since $\mathbf{x}^* \in H_i$,

$$|r_i| = d(\mathbf{x},H_i) \leq \|\mathbf{x}-\mathbf{x}^*\| < \frac{1}{2}\epsilon^*.$$

To summarize,

$$|r_i| = |\mathbf{a}_i^T\mathbf{x}_k - b_i| \begin{cases} < \frac{1}{2}\epsilon^* & \text{for } i\notin I \\ > \frac{1}{2}\epsilon^* & \text{for } i\in I \end{cases}.$$

Thus, if we consider the above, $D = \underset{D\subset[A],|D|=d}{\operatorname{argmax}} \sum_{i\in D} |A\mathbf{x}-\mathbf{b}|_i$ is clearly a subset of I for $d \leq s$. $\square$

Lemma 2.2.3. *Let $0 < \delta < 1$. Define*

$$k^* = \max\left(0, \left\lceil \frac{\log\left(\frac{\delta(\epsilon^*)^2}{4\|\mathbf{x}^*\|^2}\right)}{\log\left(1-\frac{\sigma_{\min}^2(A_*)}{m-s}\right)} \right\rceil\right).$$

Then in window i of Method 2.6 or Method 2.7, the iterate produced by the RK iterations, $\mathbf{x}_{k^*}^i$ satisfies

$$\mathbb{P}\Big[\|\mathbf{x}_{k^*}^i - \mathbf{x}^*\| \leq \frac{1}{2}\epsilon^*\Big] \geq (1-\delta)\Big(\frac{m-s}{m}\Big)^{k^*}. \tag{2.9}$$

Proof. Let E be the event that $i_1, i_2, ..., i_{k^*} \notin I$ for all index selections in window W. Note that

$$\mathbb{P}(E) \geq \left(\frac{m-s}{m}\right)^{k^*}$$

since there are $m-s$ consistent equations and the equations are being selected uniformly at random (the rows of A have unit norm).

Now, note that if one conditions upon E and looks at the expected value of $\|\mathbf{x}_{k^*} - \mathbf{x}^*\|^2$, this will be the same value as the expectation of $\|\mathbf{x}_{k^*} - \mathbf{x}^*\|^2$ if $\mathbf{x}_{k^*}$ is created with randomized Kaczmarz run on $A_*, \mathbf{b}_*$; we denote this expectation as $\mathbb{E}_{A_*,\mathbf{b}_*}[\|\mathbf{x}_{k^*} - \mathbf{x}^*\|^2]$. Applying [**SV09**, Theorem 2] (a special case of Proposition 2.2.1), we see that

$$\begin{aligned}
\mathbb{E}[\|\mathbf{x}_{k^*} - \mathbf{x}^*\|^2 | E] &= \mathbb{E}_{A_*,\mathbf{b}_*}[\|\mathbf{x}_{k^*} - \mathbf{x}^*\|^2] \\
&\leq \left(1 - \frac{\sigma_{\min}^2(A_*)}{m-s}\right)^{k^*} \|\mathbf{x}_0 - \mathbf{x}^*\|^2 \\
&= \left(1 - \frac{\sigma_{\min}^2(A_*)}{m-s}\right)^{k^*} \|\mathbf{x}^*\|^2.
\end{aligned}$$

Now, since $k^* \geq \frac{\log\left(\frac{\delta(\epsilon^*)^2}{4\|\mathbf{x}^*\|^2}\right)}{\log\left(1-\frac{\sigma_{\min}^2(A_*)}{(m-s)}\right)}$, we have $\left(1 - \frac{\sigma_{\min}^2(A_*)}{m-s}\right)^{k^*} \leq \frac{\delta(\epsilon^*)^2}{4\|\mathbf{x}^*\|^2}$ and so

$$\mathbb{E}[\|\mathbf{x}_{k^*} - \mathbf{x}^*\|^2 | E] \leq \frac{\delta}{4}(\epsilon^*)^2.$$

Applying the conditional Markov inequality, we have

$$\begin{aligned}\mathbb{P}[\|\mathbf{x}_{k^*} - \mathbf{x}^*\|^2 > \frac{1}{4}(\epsilon^*)^2 | E] &\leq \frac{\mathbb{E}[\|\mathbf{x}_{k^*} - \mathbf{x}^*\|^2 | E]}{\frac{1}{4}(\epsilon^*)^2} \\ &\leq \frac{\frac{\delta}{4}(\epsilon^*)^2}{\frac{1}{4}(\epsilon^*)^2} = \delta\end{aligned}$$

Thus, $\mathbb{P}[\|\mathbf{x}_{k^*} - \mathbf{x}^*\|^2 \leq \frac{1}{4}(\epsilon^*)^2 | E] \geq 1 - \delta$ so

$$\mathbb{P}[\|\mathbf{x}_{k^*} - \mathbf{x}^*\| \leq \frac{1}{2}\epsilon^*] \geq (1-\delta)\left(\frac{m-s}{m}\right)^{k^*}.$$

□

First, note that we must restrict k^* to be nonnegative; since $\log\left(1 - \frac{\sigma_{\min}^2(A_*)}{m-s}\right)$ is negative, if $\log\left(\frac{\delta(\epsilon^*)^2}{4\|\mathbf{x}^*\|^2}\right)$ is positive, we must define $k^* = 0$. However, this corresponds to the situation in which $\epsilon^* > 2\|\mathbf{x}^*\|$ and the initial iterate $\mathbf{x}_0 = \mathbf{0}$ is within the detection horizon. Additionally, note that k^* depends upon δ, so one is not able to make this probability as large as one likes. As δ decreases, k^* increases, so the right hand side of (2.9) is bounded away from 1. In Figure 2.15, we plot k^* and $(1-\delta)\left(\frac{m-s}{m}\right)^{k^*}$ for Gaussian systems with various number of corruptions. In the plots, we see that the value of δ which maximizes this probability depends upon s. Determining this maximizing δ was not computable in closed form. Additionally, we point out that the empirical behavior of the method does not appear to depend upon δ; we believe this is an artifact of our proof.

Theorem 2.2.4. *Let $0 < \delta < 1$. Suppose $d \geq s$, $W \leq \lfloor \frac{m-n}{d} \rfloor$ and k^* is as given in Lemma 2.2.3. Then Method 2.6 on $A, \mathbf{b}$ will detect the corrupted equations ($I \subset S$) and the remaining equations given by $A_{[m]-S}, \mathbf{b}_{[m]-S}$ will have solution $\mathbf{x}^*$ with probability at least*

$$1 - \left[1 - (1-\delta)\left(\frac{m-s}{m}\right)^{k^*}\right]^W.$$

Proof. Since $d \geq s$, we need only have one 'successful' window where $\|\mathbf{x}_{k^*} - \mathbf{x}^*\| < \frac{1}{2}\epsilon^*$ in order to guarantee detection of all of the corrupted equations, by Lemma 2.2.2. Note that all of the windows are independent from each other in Method 2.6 and Lemma 2.2.3 yields that the probability that $\|\mathbf{x}_{k^*} - \mathbf{x}^*\| < \frac{1}{2}\epsilon^*$ is at least $p := (1-\delta)\left(\frac{m-s}{m}\right)^{k^*}$. Thus, we may bound the probability of

success by that of a binomial distribution with parameters W and p. Thus, success happens with probability at least

$$1 - \left[1 - (1-\delta)\Big(\frac{m-s}{m}\Big)^{k^*}\right]^W.$$

□

In Figure 2.15, we plot $1 - \left[1 - (1-\delta)\Big(\frac{m-s}{m}\Big)^{k^*}\right]^W$ for corrupted Gaussian systems and choices of δ. Here $W = \lfloor (m-n)/d \rfloor$ and $d = s$. Again, we reiterate that we believe the dependence upon δ is an artifact of the proof of Lemma 2.2.3. Substituting e.g., $\delta = 0.5$ in probability bounds gives a value not far from its maximum for all systems we studied; see Figures 2.15 and 2.18.

Theorem 2.2.5. *Let $0 < \delta < 1$. Suppose $d \geq 1$, $W \leq \lfloor \frac{m-n}{d} \rfloor$ and k^* is as given in Lemma 2.2.3. Then Method 2.7 on $A, \mathbf{b}$ will detect the corrupted equations ($I \subset S$) and the remaining equations given by $A_{[m]-S}, \mathbf{b}_{[m]-S}$ will have solution $\mathbf{x}^*$ with probability at least*

$$1 - \sum_{j=0}^{\lceil s/d \rceil - 1} \binom{W}{j} p^j (1-p)^{W-j}$$

where $p = (1-\delta)\Big(\frac{m-s}{m}\Big)^{k^*}$.

Proof. Since $d \geq 1$ and we are selecting unique indices in each iteration of Method 2.7, we need to have $\lceil s/d \rceil$ 'successful' windows where $\|\mathbf{x}_{k^*} - \mathbf{x}^*\| < \frac{1}{2}\epsilon^*$ in order to guarantee detection of all of the corrupted equations, by Lemma 2.2.2. Since all of the windows of RK iterations are independent from each other in Method 2.7 and by Lemma 2.2.3 the probability that $\|\mathbf{x}_{k^*} - \mathbf{x}^*\| < \frac{1}{2}\epsilon^*$ is at least $p := (1-\delta)\Big(\frac{m-s}{m}\Big)^{k^*}$, we may bound the probability of success by that of a cumulative binomial distribution with parameters W and p and we calculate the probability that the number of successes, $j \geq \lceil s/d \rceil$. Thus, success happens with probability defined by the probabilities that less than $\lceil s/d \rceil$ windows are successful. The probability of success is bounded below by

$$1 - \sum_{j=0}^{\lceil s/d \rceil - 1} \binom{W}{j} p^j (1-p)^{W-j}.$$

□

In Figure **??**, we plot $1 - \sum_{j=0}^{\lceil s/d \rceil - 1} \binom{W}{j} p^j (1-p)^{W-j}$ for corrupted Gaussian systems and choices of δ. Here $W = 2$, $d = \lceil s/2 \rceil$, and k^* is as given in Lemma 2.2.3. We believe that the dependence upon δ is an artifact of our proof. Evidence suggesting this is seen in the middle and right plots of Figure **??**, as the empirical behavior of Method 2.7 does not appear to depend upon δ.

These bounds on the probability of successfully detecting all corrupted equations in one window, while provable and nonzero, are pessimistic and do not resemble the experimental rate of success for any systems we studied; see Figures 2.15 and 2.18. A tighter bound on the rate of convergence for particular systems could provide a tighter lower bound on this probability.

2.2.2.2. *Experimental Results.* We are only able to prove theoretical results when the windows of Methods 2.6 and 2.7 are independent and for the specified values of k^* and d. However, in practice, these methods perform well for different values of k and d, and Method 2.5 can be quite successful. In this section, we present experimental results demonstrating the performance of these methods, for various choices of d and k, on Gaussian, correlated, and real systems.

Random Data Experiments. The plots in Figures 2.15 and 2.16 are all for Method 2.6 on a 50000×100 Gaussian system defined by A with $a_{ij} \sim \mathcal{N}(0, 1)$, then normalized. The system is corrupted in randomly selected right-hand side entries with random integers in $[1, 5]$ so that $\epsilon^* = 1$. For these plots and experiments, $d = s$. The upper left image of Figure 2.15 plots the k^* values defined in Lemma 2.2.3 for this system, and the upper middle image plots the theoretically guaranteed probability of selecting all s corrupted equations in a single window. The upper right image of Figure 2.15 plots the theoretically guaranteed probability of selecting all s corrupted equations in one window out of the $W = \lfloor \frac{m-n}{s} \rfloor$, while the lower left image plots the ratio of successful trials, in which all s corrupted equations were selected in one window of the W, out of 100 trials. The lower right plot of Figure 2.15 plots how this ratio changes as the number of RK iterations, k, in each window varies. Finally, Figure 2.16 plots the ratio of successful trials, in which all s corrupted equations were selected after all $W = \lfloor \frac{m-n}{s} \rfloor$ windows, out of 100 trials as one varies δ (left) and k (right). We note that the lower bounds on the probability of successfully detecting all corrupted equations in one window are quite pessimistic; experimentally (in the lower left plot) we see that Method 2.6 is able to detect all corruption for much larger numbers of corrupted equations, s,

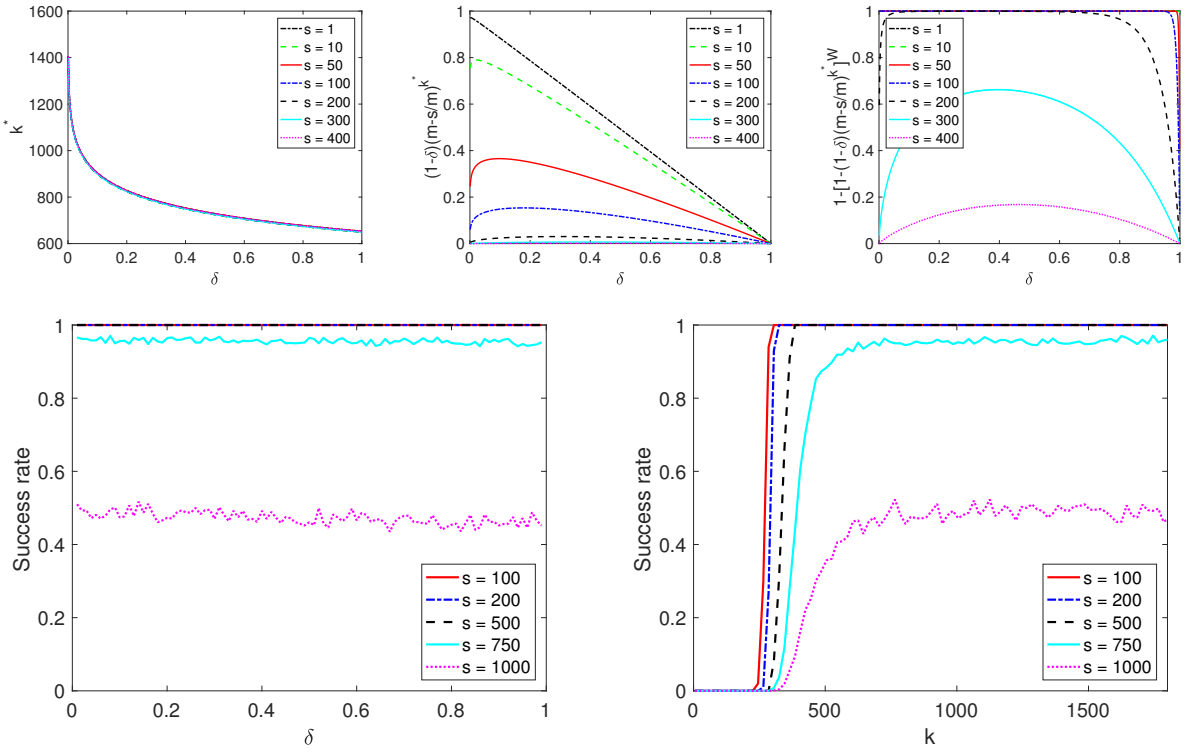

FIGURE 2.15. Plots for Method 2.6 on 50000×100 Gaussian system (normalized). Upper left: k^*; upper middle: lower bound on probability of detecting all corruption in single window; upper right: lower bound on probability of detecting all corruption in one out of all windows; lower: average ratio of detecting all corruption in one out of all windows for choice of δ (left) or k (right).

than predicted theoretically (in the upper right plot). Additionally, we note that experimentally, successfully detecting the corrupted equations does not appear to depend upon δ. For all $0 < \delta < 1$, the k^* value defined in Lemma 2.2.3 appears to be large enough to guarantee convergence within the detection horizon.

In Figure **??**, we briefly explore the theoretical guarantees for Method 2.7 given in Theorem 2.2.5, and the empirical behavior of this method. These plots are for a 50000×100 Gaussian system (normalized) with various number, s, of corrupted equations. We randomly sample s entries of the right hand side vector b and corrupt them by adding 1, so $\epsilon^* = 1$. The plot on the left of Figure **??** plots the lower bound on the probability of selecting all corrupted equations given in Theorem 2.2.5 for $W = 2$, $d = \lceil s/2 \rceil$, and k^* as given in Lemma 2.2.3. Meanwhile in the middle and right plots of Figure **??**, we plot the average fraction of corrupted equations recorded for Method 2.7 with

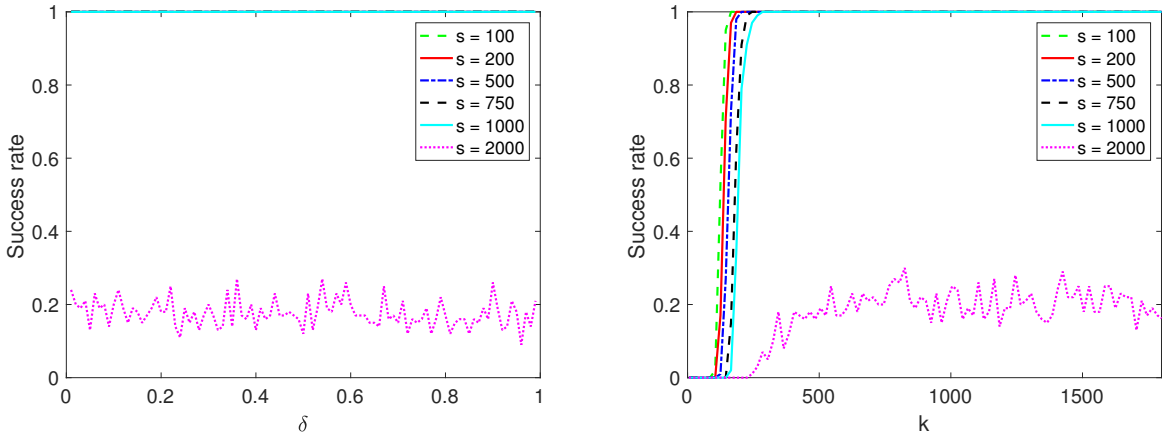

FIGURE 2.16. Plots for Method 2.6 on 50000×100 Gaussian system (normalized) with various number of corrupted equations, s. Left: Experimental ratio of successfully detecting all s corrupted equations after all $W = \lfloor \frac{m-n}{s} \rfloor$ windows for choice of δ; right: Experimental ratio of successfully detecting all s corrupted equations after all $W = \lfloor \frac{m-n}{s} \rfloor$ windows for choice of k (number of RK iterations per window).

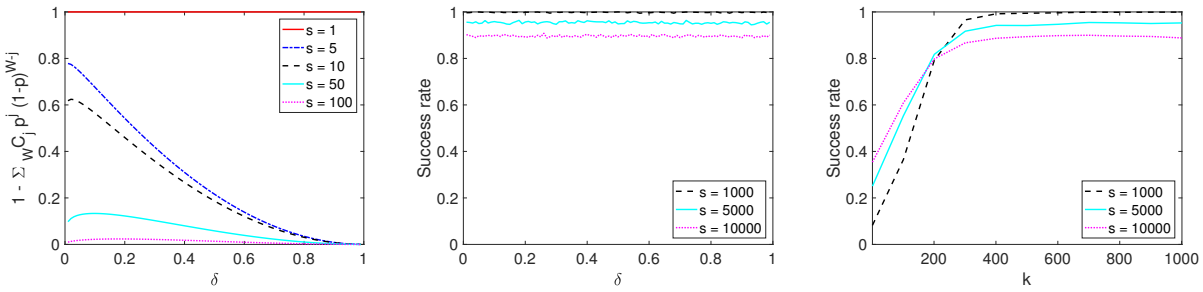

FIGURE 2.17. Plots for Method 2.7 on 50000 × 100 Gaussian system (normalized). Left: Bound given in Theorem 2.2.5; middle and right: Average fraction of corrupted equations detected after $W = \lceil s/d \rceil$ windows recording $d = \lceil s/10 \rceil$ equations per window varying δ (middle) and k (right).

$W = \lceil s/d \rceil$ and $d = \lceil s/10 \rceil$ for 100 trials. The middle plot has k^* (as given in Lemma 2.2.3) RK iterations per window for varying δ, while the right plot has varying k (number of RK iterations per window). Note that this experiment is different from the others we present in this section as we display the average fraction of corrupted equations recorded over 100 trials, rather than the fraction of trials which detected all corrupted equations. We note that the theoretical bound plotted on the left of Figure **??** is even more pessimistic than that of Method 2.6, but meanwhile the empirical performance of Method 2.7 plotted in the middle and right of Figure **??** is even better

than that seen for Method 2.6. For this reason, we do not plot these bounds (Theorem 2.2.5) or the performance of Method 2.7 for additional systems as we expect the results to trend similarly for other systems.

The figures for Method 2.6 mentioned above are recreated for a system whose rows are more correlated ($A \in \mathbb{R}^{50000\times 100}$ with $a_{ij} \sim \mathcal{N}(1, 0.5)$ then normalized) in Figures 2.18 and 2.19. The system is corrupted in randomly selected right-hand side entries with random integers in $[1, 5]$ so that $\epsilon^* = 1$. For these plots and experiments, $d = s$. The upper left image of Figure 2.18 plots the k^* values defined in Lemma 2.2.3 for this system, and the upper middle image plots the theoretically guaranteed probability of selecting all s corrupted equations in a single window of Method 2.6. The upper right image of Figure 2.18 plots the theoretically guaranteed probability of selecting all s corrupted equations in one window out of the $W = \lfloor \frac{m-n}{s} \rfloor$, while the lower left image plots the ratio of successful trials, in which all s corrupted equations were selected in one window of the W, out of 100 trials. The lower right plot of Figure 2.18 plots how this ratio changes as the number of RK iterations, k, in each window varies. Finally, Figure 2.19 plots the ratio of successful trials, in which all s corrupted equations were selected after all W windows, out of 100 trials as one varies δ (left) and k (right). We note that the discrepancy between the lower bound on the probability of successfully detecting all corrupted equations in one window (upper right plot) and the experimental rate of detecting all corruption (lower left plot) is even larger than in the case of Gaussian systems.

First, note that k^*, as given in Lemma 2.2.3, depends very weakly upon s. In the upper left plots of Figures 2.15 and 2.18, the values of k^* plotted are very slightly different for different values of s (the line thickness makes these distinct lines appear as one). Note that the definition of k^* (the theoretically required number of RK iterations to reach the detection horizon) is defined by the theoretical convergence rate which can be quite pessimistic. As has been seen in the lower right plots of Figures 2.15 and 2.18, and in the right plots of Figures 2.16 and 2.19, detection can be successful with a significantly smaller choice of k. Note that in Figure 2.15, the theoretically required k^* value is between 600 and 1400 but $k > 500$ seems to perform well. Likewise, in Figure 2.18, the theoretically required k^* value is between 3000 and 8000 but $k > 500$ seems to perform

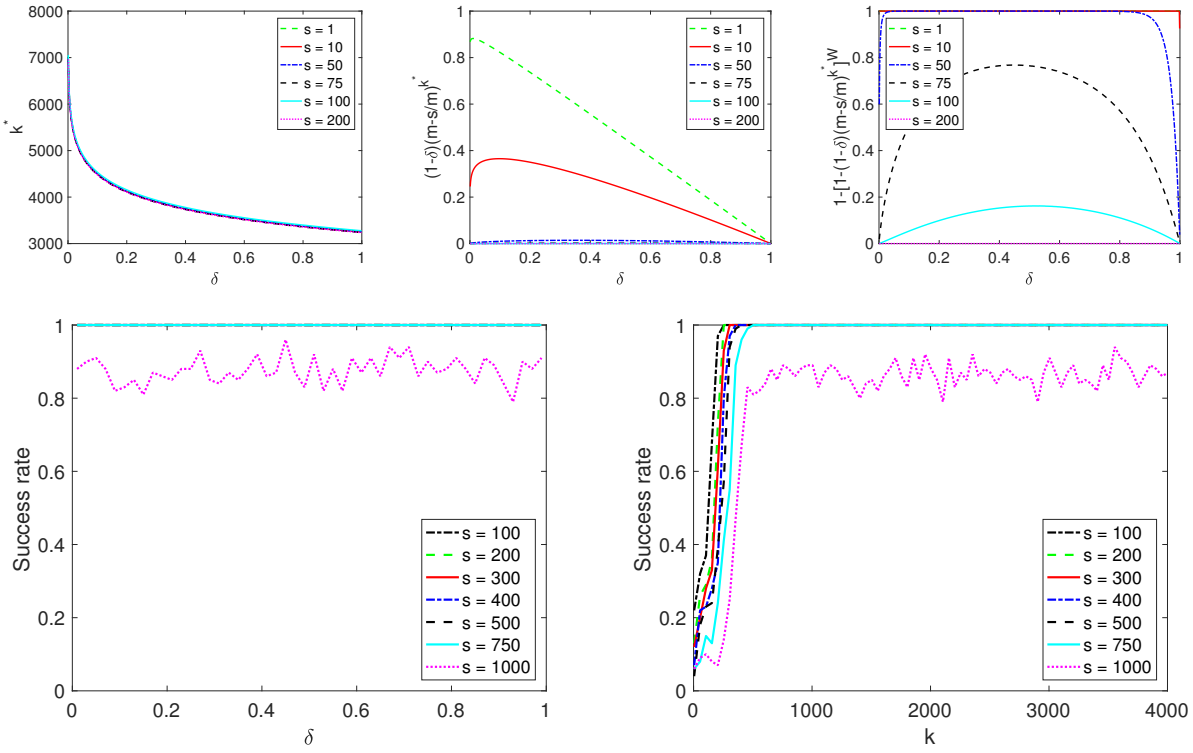

FIGURE 2.18. Plots for Method 2.6 on 50000×100 correlated system (normalized). Upper left: k^*; upper middle: lower bound on probability of detecting all corruption in single window; upper right: lower bound on probability of detecting all corruption in one out of all windows; lower: average ratio of detecting all corruption in one out of all windows for choice of δ (left) or k (right).

well. It is unsurprising that this bound is even more pessimistic for the correlated system, as the conditioning of a correlated system causes the RK convergence guarantee to be quite poor, while experimentally we see a much faster rate of convergence.

Implementation considerations. There are several options for d, some more practically feasible than others. Our theoretical results are probabilistic guarantees for Method 2.6 with $d \geq s$, which of course cannot be known in practice, as well as for Method 2.7 with $d \geq 1$, which is practical, but the method is more expensive computationally. In practice, one could choose d as the user estimate for s.

The choice of d and W are complementary in that increasing d decreases W (since one may have less windows if in each window more equations are selected). In selecting d and W, we wish to

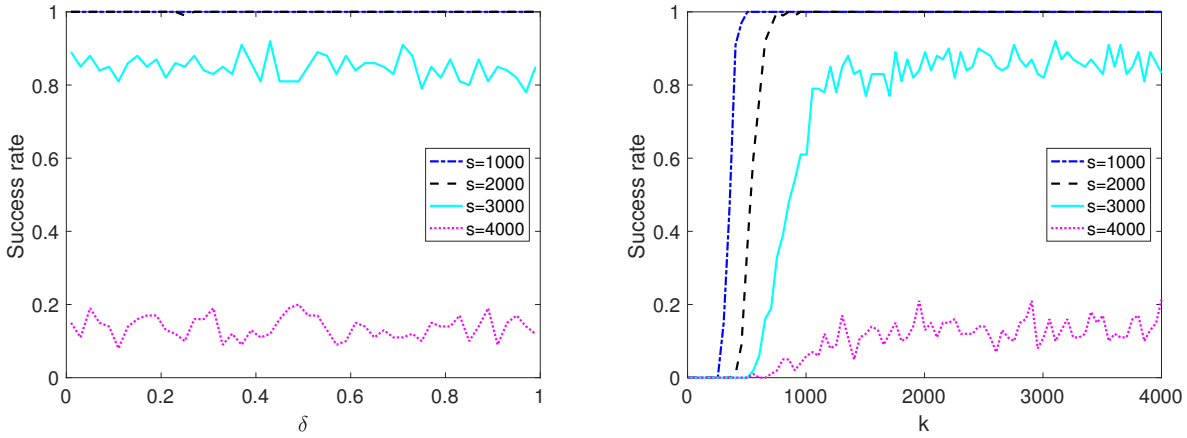

FIGURE 2.19. Plots for Method 2.6 on 50000×100 correlated system (normalized) with various number of corrupted equations, s. Left: Experimental ratio of successfully detecting all s corrupted equations after all $W = \lfloor \frac{m-n}{s} \rfloor$ windows for choice of δ; right: Experimental ratio of successfully detecting all s corrupted equations after all $W = \lfloor \frac{m-n}{s} \rfloor$ windows for choice of k (number of RK iterations per window).

balance the desire to increase d in order to record all of the corrupted equations when we have a successful window with the fact that as d grows, we can have less windows. We never discard or record more than $m-n$ of the constraints, as at the end of any method, we wish to have a full rank linear system of equations remaining whose solution is $\mathbf{x}^*$, the pseudo-solution. Thus, for any d we may not run more than $W = \lfloor \frac{m-n}{d} \rfloor$ windows. However, in practice, this choice of W may be larger than is necessary. This is explored in Figures 2.20 and 2.21. In the experiment producing Figure 2.20, we ran $W = \lfloor \frac{m-n}{d} \rfloor$ windows of Method 2.6 with k^* (defined in Lemma 2.2.3) RK iterations selecting d equations each window, and record the ratio of successful trials, which selected all s corrupted equations after all windows, out of 100 trials. The figure on the left shows the results for a Gaussian system, while the figure on the right shows the results for a correlated system. In the experiment producing Figure 2.21, we ran $W \leq \lfloor \frac{m-n}{s} \rfloor$ windows of Method 2.6 with k^* (defined in Lemma 2.2.3) RK iterations selecting s equations each window, and record the ratio of successful trials, which selected all s corrupted equations after all windows, out of 100 trials. The figure on the left is for a Gaussian system, while the figure on the right is for a correlated system. Both are corrupted with random integers in $[1, 5]$ in randomly selected entries of $\mathbf{b}$, so $\epsilon^* = 1$.

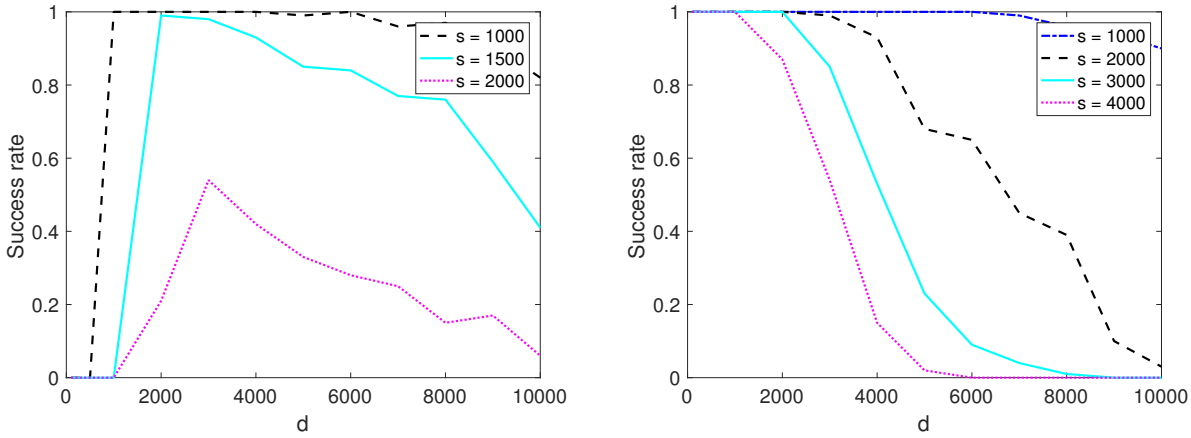

FIGURE 2.20. Plots for Method 2.6 for varying d. Average ratio of detecting all corruption after all windows on 50000×100 Gaussian system (left) or 50000×100 correlated system (right).

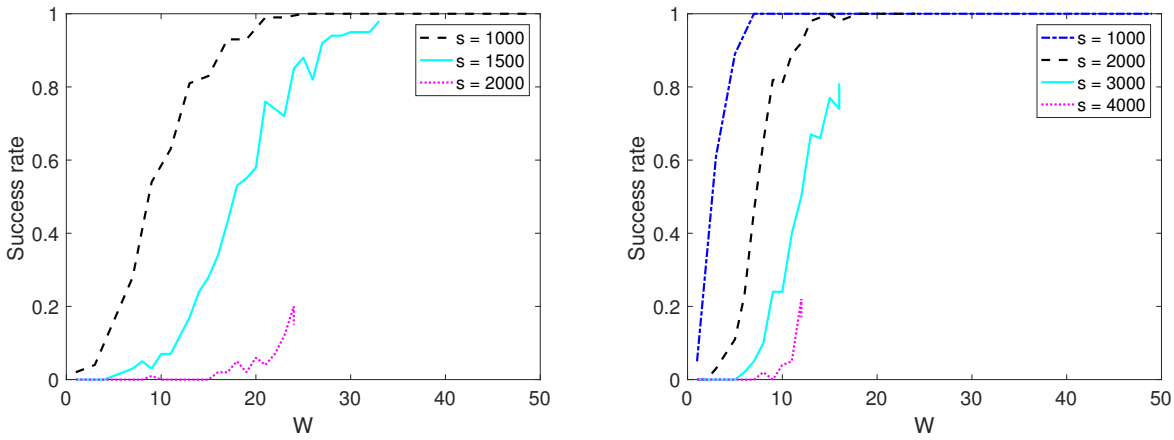

FIGURE 2.21. Plots for Method 2.6 for varying W. Average ratio of detecting all corruption after all windows on 50000×100 Gaussian system (left) or 50000×100 correlated system (right).

Method 2.5, despite not having independent windows, performs well in practice as is seen in Figure 2.22. In this experiment, we perform $W = \lfloor \frac{m-n}{s} \rfloor$ windows of Method 2.5 with k RK iterations, removing s equations each window. The plot shows the ratio of successful trials, in which all s corrupted equations are removed after all W windows, out of 100 trials. The method is run on a Gaussian system which is corrupted by random integers in $[1, 5]$ in randomly selected entries of $\mathbf{b}$, so $\epsilon^* = 1$.

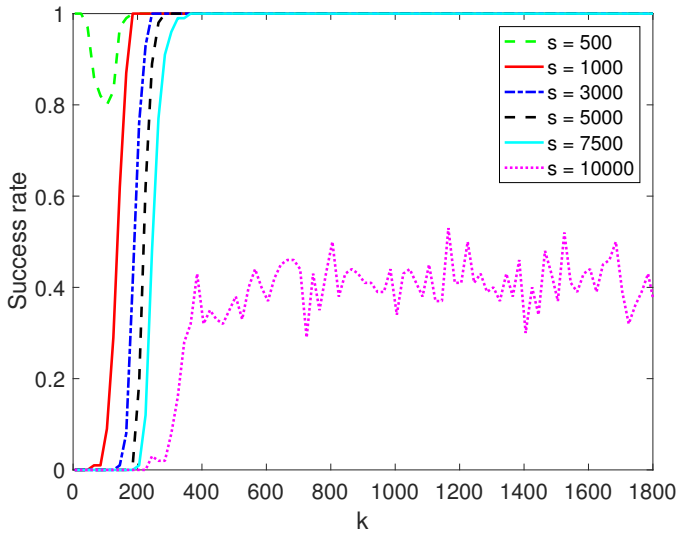

FIGURE 2.22. Experimental ratio of success of removing (Method 2.5) all s corrupted equations after all $W = \lfloor \frac{m-n}{s} \rfloor$ windows for 50000×100 Gaussian system with s corrupted equations and choice of k.

Real Data Experiments. We additionally test the methods on real data. Our first experiments are on tomography problems, generated using the MATLAB Regularization Toolbox by P.C. Hansen (`http://www.imm.dtu.dk/~pcha/Regutools/`) [**Han07**]. In particular we present a 2D tomography problem $A\mathbf{x} = \mathbf{b}$ for an $m \times n$ matrix with $m = fN^2$ and $n = N^2$. Here A corresponds to the absorption along a random line through an $N \times N$ grid. In our experiments we set $N = 20$ and the oversampling factor $f = 3$. This yielded a matrix A with condition number $\kappa(A) = 2.08$. As the resulting system was consistent, we randomly sampled $s = 100$ constraints uniformly from among the rows of A and corrupted the right-hand side vector $\mathbf{b}$ by adding 1 in these entries, so $\epsilon^* = 1$. This corrupted system has $k^* = 66334$ (as given in Lemma 2.2.3). Figure 2.23 contains the average fraction of the $s = 100$ corrupted equations detected or removed for Methods 2.6 (left) and 2.5 (right) after all $W = \lfloor \frac{m-n}{d} \rfloor$ windows for various values of k (RK iterations per window) for 100 trials.

We also generated corrupted data sets using the Wisconsin (Diagnostic) Breast Cancer data set, which includes data points whose features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass and describe characteristics of the cell nuclei present in the image [**Lic13**]. This collection of data points forms our matrix $A \in \mathbb{R}^{699\times 10}$, we construct $\mathbf{b}$ to form

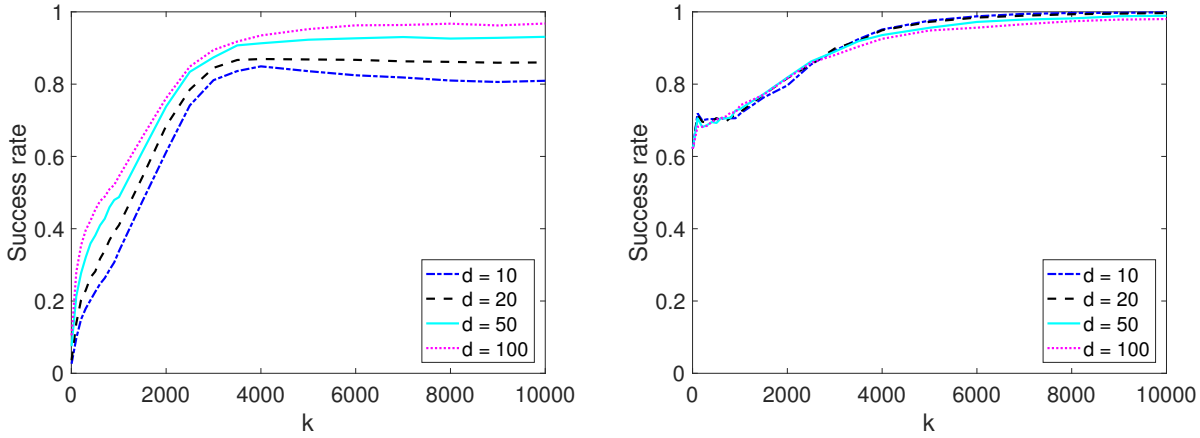

FIGURE 2.23. Plots for 1200×400 tomography system with $s = 100$ corrupted equations. Left: average fraction of corrupted equations detected in 100 trials after all $W = \lfloor \frac{m-n}{d} \rfloor$ windows of Method 2.6; right: average fraction of corrupted equations removed in 100 trials after all $W = \lfloor \frac{m-n}{d} \rfloor$ windows of Method 2.5.

a consistent system, and then corrupt a random selection of 100 entries of the right-hand side by adding 1, so $\epsilon^* = 1$. This corrupted system has $k^* = 3432$ (as given in Lemma 2.2.3). Figure 2.24 contains the average fraction of the $s = 100$ corrupted equations detected or removed for Methods 2.6 (left) and 2.5 (right) after all $W = \lfloor \frac{m-n}{d} \rfloor$ windows for various values of k (RK iterations per window) for 100 trials.

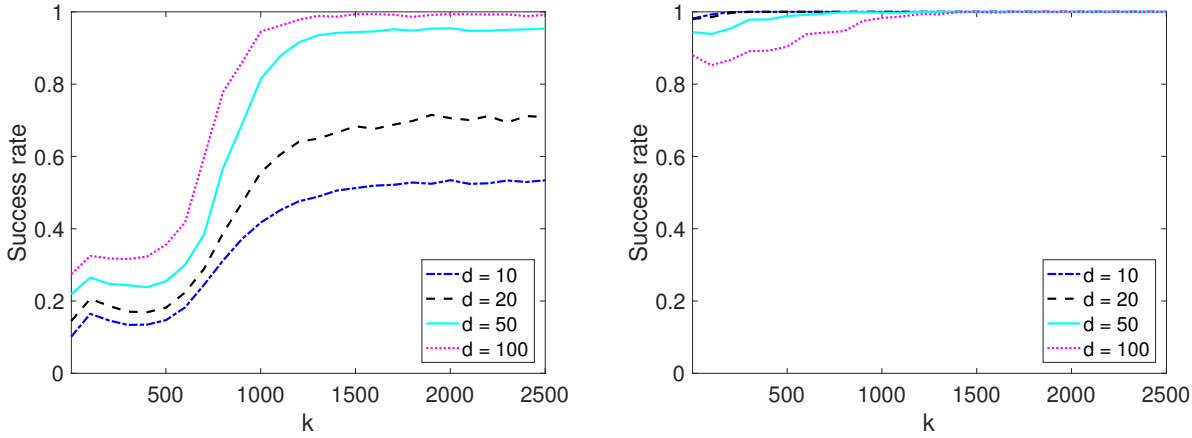

FIGURE 2.24. Plots for 699×10 system defined by Wisconsin (Diagnostic) Breast Cancer data set with $s = 100$ corrupted equations. Left: average fraction of corrupted equations detected in 100 trials after all $W = \lfloor \frac{m-n}{d} \rfloor$ windows of Method 2.6; right: average fraction of corrupted equations removed in 100 trials after all $W = \lfloor \frac{m-n}{d} \rfloor$ windows of Method 2.5.

2.3. Sampling Kaczmarz-Motzkin Method

Despite the similarity between the Kaczmarz and Motzkin methods (the difference only being in the row selection criterion), work on these approaches has remained mostly disjoint. One main contribution of this thesis is a family of methods, the *Sampling Kaczmarz-Motzkin* (SKM) methods, which are intended to balance the pros and cons of these two related methods [**DLHN17**]. Namely, Motzkin's method forms iterates whose distance to the polyhedral solution space are monotonically decreasing; however, the time required to choose the most violated hyperplane in each iteration is costly. Conversely, the Randomized Kaczmarz method has a very inexpensive cost per iteration; however, the method has slow convergence when many of the constraints are satisfied. Our methods still have a probabilistic choice, like in randomized Kaczmarz, but make strong use of the maximum violation criterion within this random sample of the constraints. Our method is easily seen to interpolate between what was proposed in [**LL10**] and in [**MS54**]. Note that the description of SKM in Section 1.2.2.3 presented the method with *projection parameter*, $\lambda = 1$. Here we consider the more general family of methods which allow for under ($\lambda < 1$) and over-projection ($\lambda > 1$). Pseudo-code for this algorithm is presented in Method 2.8 and MATLAB code for this algorithm may be found in Appendix A.

Method 2.8 Sampling Kaczmarz-Motzkin method

procedure SKM($A, \mathbf{b}, \mathbf{x}_0, \lambda, \beta, k$)
  for $j = 1, 2, ..., k$ **do**
    Choose a random sample of β constraints, τ_j, uniformly from among the rows of A, $[m]$.
    $\mathbf{x}_j = \mathbf{x}_{j-1} - \lambda \frac{(\mathbf{a}_{i_j}^T \mathbf{x}_{j-1} - b_{i_j})^+}{\|\mathbf{a}_{i_j}\|^2} \mathbf{a}_{i_j}$ where $i_j \in \underset{i \in \tau_j}{\operatorname{argmax}} \ \mathbf{a}_i^T \mathbf{x}_{j-1} - b_i$.
  end for
  return $\mathbf{x}_k$
end procedure

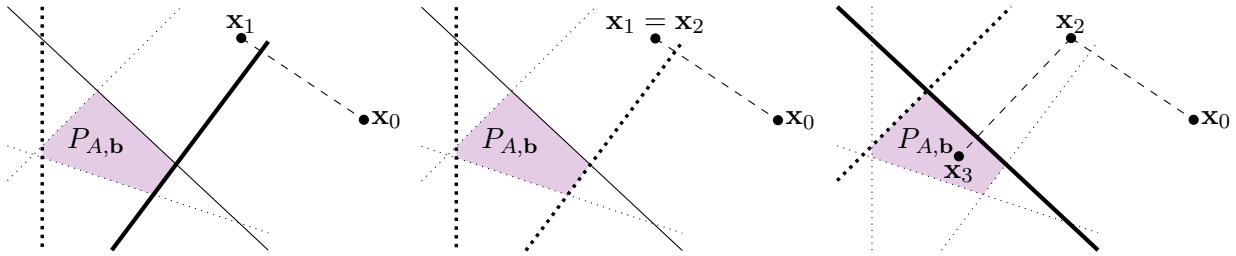

FIGURE 2.25. An example of the iterates formed by the SKM method with $\lambda > 1$ on a feasible LF.

For clarity, we include an example of several iterations of the SKM method on an LF in Figure 2.25. Each image illustrates the projection $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ in a dashed line. The lines surrounding the polyhedral feasible region, $P_{A,\mathbf{b}}$ represent the boundary hyperplanes of the halfspaces defining $P_{A,\mathbf{b}}$. The lines are dotted if the constraint defining the corresponding halfspace is satisfied by the iterate $\mathbf{x}_{k-1}$, and solid if the constraint defining the corresponding halfspace is violated by the iterate $\mathbf{x}_{k-1}$. The lines representing the hyperplanes bounding the halfspaces defined by the randomly selected constraints, $\mathbf{a}_i^T \mathbf{x} \leq b_i$ for $i \in \tau_k$ are shown in bold. The constraint selected from among this random sample (if any are violated) is shown in extra bold.

Remark: the SKM method with $\beta = m$ recovers the Motzkin methods, while the SKM method with $\beta = 1$ gives the randomized Kaczmarz methods.

2.3.1. Convergence Rate. We now state our first main result. We show that the SKM methods satisfy a linear rate of convergence and with a potential acceleration depending upon the number of satisfied constraints.

Theorem 2.3.1. *Consider a normalized system with $\|\mathbf{a}_i\|^2 = 1$ for all $i \in [m]$. If the feasible region, $P_{A,\mathbf{b}}$, is nonempty then the SKM method (Method 2.8) with samples of size β and projection parameter λ converges at least linearly in expectation and the bound on the rate depends on the number of satisfied constraints in the system $A\mathbf{x} \leq \mathbf{b}$. More precisely, let s_{k-1} be the number of satisfied constraints after iteration $k-1$ and $V_{k-1} = \max\{m - s_{k-1}, m - \beta + 1\}$; then, in the kth iteration,*

$$\mathbb{E}[d(\mathbf{x}_k, P_{A,\mathbf{b}})^2] \leq \left(1 - \frac{2\lambda - \lambda^2}{V_{k-1}L_2^2}\right) d(\mathbf{x}_{k-1}, P_{A,\mathbf{b}})^2 \leq \left(1 - \frac{2\lambda - \lambda^2}{mL_2^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2.$$

We show that the SKM methods enjoy a linear rate of convergence. We begin with a simple useful observation.

Lemma 2.3.2. *Suppose $\{a_i\}_{i=1}^n, \{b_i\}_{i=1}^n$ are real sequences so that $a_{i+1} \geq a_i \geq 0$ and $b_{i+1} \geq b_i \geq 0$. Then*

$$\sum_{i=1}^n a_i b_i \geq \sum_{i=1}^n \bar{a} b_i, \text{ where } \bar{a} \text{ is the average } \bar{a} = \frac{1}{n}\sum_{i=1}^n a_i.$$

Proof. Note that $\sum_{i=1}^n a_i b_i = \sum_{i=1}^n \bar{a} b_i + \sum_{i=1}^n (a_i - \bar{a}) b_i$, so we need only show that $\sum_{i=1}^n (a_i - \bar{a}) b_i \geq 0$, which is equivalent to $\sum_{i=1}^n (n a_i - \sum_{j=1}^n a_j) b_i \geq 0$, so we define the coefficients $c_i := n a_i - \sum_{j=1}^n a_j$. Now, since $\{a_i\}_{i=1}^n$ is increasing, there is some $1 < k < n$ so that $c_k \leq 0$ and $c_{k+1} \geq 0$ and the c_i are increasing. Since $\{b_i\}_{i=1}^n$ is non-negative and non-decreasing we have

$$\sum_{i=1}^n c_i b_i = \sum_{i=1}^k c_i b_i + \sum_{i=k+1}^n c_i b_i \geq \sum_{i=1}^k c_i b_k + \sum_{i=k+1}^n c_i b_k = b_k \sum_{i=1}^n c_i = 0.$$

Thus, we have $\sum_{i=1}^n a_i b_i = \sum_{i=1}^n \bar{a} b_i + \sum_{i=1}^n (a_i - \bar{a}) b_i \geq \sum_{i=1}^n \bar{a} b_i$. □

Proof. (of Theorem 2.3.1) Write s_j for the number of zero entries in the residual $(A\mathbf{x}_j - \mathbf{b})^+$, which correspond to satisfied constraints. Define $V_j := \max\{m - s_j, m - \beta + 1\}$. Recalling that the method defines $\mathbf{x}_{j+1} = \mathbf{x}_j - \lambda (A\mathbf{x}_j - \mathbf{b})^+_{i^*} \mathbf{a}_{i^*}$ where $i^* = \underset{i \in \tau_{j+1}}{\operatorname{argmax}} \{\mathbf{a}_i^T \mathbf{x}_j - b_i, 0\} = \underset{i \in \tau_{j+1}}{\operatorname{argmax}} (A\mathbf{x}_j - \mathbf{b})^+_i$,

we have

$$d(\mathbf{x}_{j+1}, P_{A,\mathbf{b}})^2 = \|\mathbf{x}_{j+1} - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_{j+1})\|^2 \leq \|\mathbf{x}_{j+1} - \mathcal{P}_{P_{A,\mathbf{b}}}(x_j)\|^2 = \|\mathbf{x}_j - \lambda(A\mathbf{x}_j - \mathbf{b})^+_{i^*}\mathbf{a}_{i^*} - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)\|^2$$
$$= \|\mathbf{x}_j - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)\|^2 + \lambda^2((A\mathbf{x}_j - \mathbf{b})^+_{i^*})^2\|\mathbf{a}_{i^*}\|^2 - 2\lambda(A\mathbf{x}_j - \mathbf{b})^+_{i^*}\mathbf{a}_{i^*}^T(\mathbf{x}_j - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)).$$

Since $\mathbf{a}_{i^*}^T(\mathbf{x}_j - \mathcal{P}_{P_{A,\mathbf{b}}}(\mathbf{x}_j)) \geq \mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*}$, we have that

$$\begin{aligned} d(\mathbf{x}_{j+1}, P_{A,\mathbf{b}})^2 &\leq d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 + \lambda^2((A\mathbf{x}_j - \mathbf{b})^+_{i^*})^2\|\mathbf{a}_{i^*}\|^2 - 2\lambda(A\mathbf{x}_j - \mathbf{b})^+_{i^*}(\mathbf{a}_{i^*}^T\mathbf{x}_j - b_{i^*}) \\ &= d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 - (2\lambda - \lambda^2)((A\mathbf{x}_j - \mathbf{b})^+_{i^*})^2 \\ &= d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 - (2\lambda - \lambda^2)\|(A_{\tau_{j+1}}\mathbf{x}_j - \mathbf{b}_{\tau_{j+1}})^+\|^2_\infty. \end{aligned} \tag{2.10}$$

Now, we take advantage of the fact that, if we consider the size of the entries of $(A\mathbf{x}_j - \mathbf{b})^+$, we can determine the precise probability that a particular entry of the residual vector is selected. Let $(A\mathbf{x}_j - \mathbf{b})^+_{i_k}$ denote the $(k+\beta)$th smallest entry of the residual vector (i.e., if we order the entries of $(A\mathbf{x}_j - \mathbf{b})^+$ from smallest to largest, we denote by $(A\mathbf{x}_j - \mathbf{b})^+_{i_k}$ the entry in the $(k+\beta)$th position). Each sample has equal probability of being selected, $\binom{m}{\beta}^{-1}$. However, the frequency that each entry of the residual vector will be expected to be selected (as $\underset{i\in\tau_{j+1}}{\operatorname{argmax}}\ \mathbf{a}_i^T\mathbf{x}_j - b_i$) depends on its size. The βth smallest entry will be selected from only one sample, while the m-th smallest entry (i.e., the largest entry) will be selected from all samples in which it appears. Each entry is selected according to the number of samples in which it appears and is largest. Thus, if we take expectation of both sides (with respect to the probabilistic choice of sample, τ_{j+1}, of size β), then

$$\mathbb{E}[\|(A_{\tau_{j+1}}\mathbf{x}_j - \mathbf{b}_{\tau_{j+1}})^+\|^2_\infty] = \frac{1}{\binom{m}{\beta}}\sum_{k=0}^{m-\beta}\binom{\beta-1+k}{\beta-1}((A\mathbf{x}_j - \mathbf{b})^+_{i_k})^2 \tag{2.11}$$

$$\geq \frac{1}{\binom{m}{\beta}}\sum_{k=0}^{m-\beta}\frac{\sum_{\ell=0}^{m-\beta}\binom{\beta-1+\ell}{\beta-1}}{m-\beta+1}((A\mathbf{x}_j - \mathbf{b})^+_{i_k})^2 \tag{2.12}$$

$$= \sum_{k=0}^{m-\beta}\frac{1}{m-\beta+1}((A\mathbf{x}_j - \mathbf{b})^+_{i_k})^2 \tag{2.13}$$

$$\geq \frac{1}{m-\beta+1}\min\left\{\frac{m-\beta+1}{m-s_j}, 1\right\}\|(A\mathbf{x}_j - \mathbf{b})^+\|_2^2, \tag{2.14}$$

where (2.12) follows from Lemma 2.3.2, because $\{\binom{\beta-1+k}{\beta-1}\}_{k=0}^{m-\beta}$ is strictly increasing and $\{(A\mathbf{x}_j - \mathbf{b})_{i_k}^+\}_{k=0}^{m-\beta}$ is non-decreasing. Equality (2.13) follows from (2.12) due to the fact that $\sum_{\ell=0}^{m-\beta}\binom{\beta-1+\ell}{\beta-1} = \binom{m}{\beta}$ which is known as the column-sum property of Pascal's triangle, among other names. Inequality (2.14) follows from the fact that the ordered summation in (2.13) is at least $\frac{m-\beta+1}{m-s_j}$ of the norm of the residual vector (since s_j of the entries are zero) or is the entire residual vector provided $s_j \geq \beta-1$.

Thus, we have

$$\begin{aligned}\mathbb{E}[d(\mathbf{x}_{j+1}, P_{A,\mathbf{b}})^2] &\leq d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 - (2\lambda - \lambda^2)\mathbb{E}[\|(A_{\tau_{j+1}}\mathbf{x}_j - \mathbf{b}_{\tau_{j+1}})^+\|_\infty^2] \\ &\leq d(\mathbf{x}_j, P_{A,\mathbf{b}})^2 - \frac{2\lambda-\lambda^2}{V_j}\|(A\mathbf{x}_j - \mathbf{b})^+\|_2^2 \leq \left(1 - \frac{2\lambda-\lambda^2}{V_j L_2^2}\right) d(\mathbf{x}_j, P_{A,\mathbf{b}})^2.\end{aligned}$$

Since $V_j \leq m$ in each iteration,

$$\mathbb{E}[d(\mathbf{x}_{j+1}, P_{A,\mathbf{b}})^2] \leq \left(1 - \frac{2\lambda-\lambda^2}{mL_2^2}\right) d(\mathbf{x}_j, P_{A,\mathbf{b}})^2.$$

Thus, inductively, we get that

$$\mathbb{E}[d(\mathbf{x}_k, P_{A,\mathbf{b}})^2] \leq \left(1 - \frac{2\lambda-\lambda^2}{mL_2^2}\right)^k d(\mathbf{x}_0, P_{A,\mathbf{b}})^2.$$

□

Now, we have that the SKM methods will perform at least as well as the Randomized Kaczmarz method in expectation; however, if we know that after a certain point the iterates satisfy some of the constraints, we can improve our expected convergence rate guarantee. Clearly, after the first iteration, if $\lambda \geq 1$, in every iteration at least one of the constraints will be satisfied so we can guarantee a very slightly increased expected convergence rate. However, we can say more based on the geometry of the problem.

Lemma 2.3.3. *The sequence of iterates, $\{\mathbf{x}_k\}$ generated by an SKM method (Method 2.8) are pointwise closer to the feasible polyhedron $P_{A,\mathbf{b}}$. That is, for all $\mathbf{a} \in P_{A,\mathbf{b}}$, $\|\mathbf{x}_k - \mathbf{a}\| \leq \|\mathbf{x}_{k-1} - \mathbf{a}\|$ for all iterations k.*

PROOF. For $\mathbf{a} \in P_{A,\mathbf{b}}$, $\|\mathbf{x}_k - \mathbf{a}\| \leq \|\mathbf{x}_{k-1} - \mathbf{a}\|$ for all k since $\mathbf{a} \in P_{A,\mathbf{b}} \subset H^{\leq}_{\mathbf{a}_{i_k}, b_{i_k}}$ and $\mathbf{x}_k$ is the projection of $\mathbf{x}_{k-1}$ towards or into the half-space $H^{\leq}_{\mathbf{a}_{i_k}, b_{i_k}}$ (provided $\mathbf{x}_{k-1} \notin H^{\leq}_{\mathbf{a}_{i_k}, b_{i_k}}$, otherwise the inequality is true with equality). □

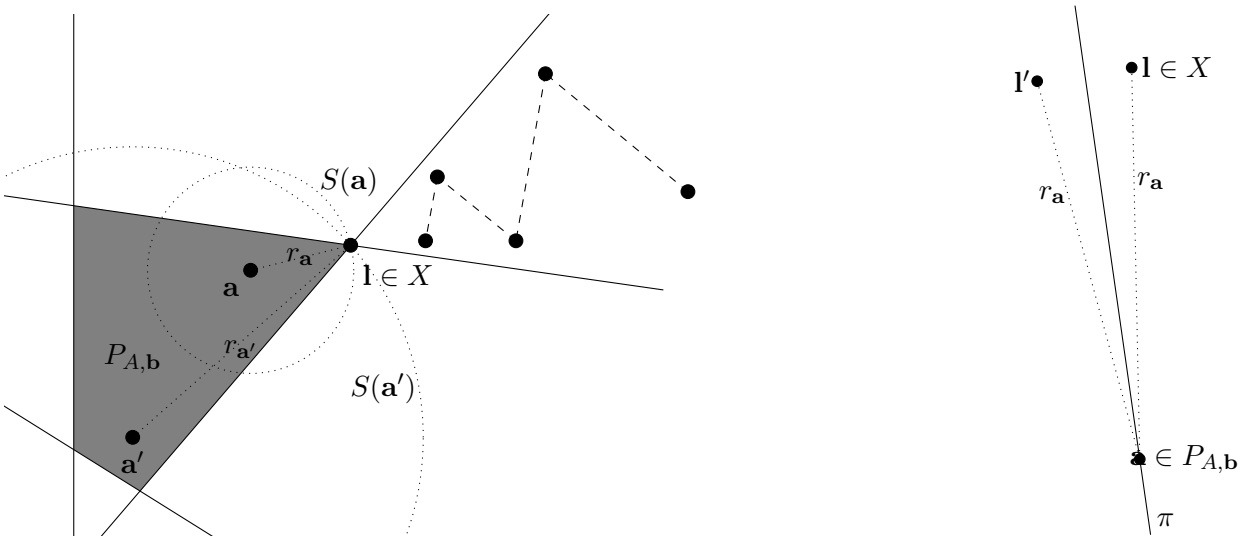

FIGURE 2.26. Left: image of $\mathbf{a} \in P_{A,\mathbf{b}}$, $r_{\mathbf{a}}$ and $S(\mathbf{a})$ and $\mathbf{l} \in \cap_{\mathbf{a} \in P_{A,\mathbf{b}}} S(\mathbf{a})$ as defined in Lemma 2.3.4. Right: image of $\mathbf{l}, \mathbf{l}' \in X$ contradicting the full-dimensionality of $P_{A,\mathbf{b}}$.

Lemma 2.3.4. *If $P_{A,\mathbf{b}}$ is n-dimensional (full-dimensional) then the sequence of iterates $\{\mathbf{x}_k\}$ generated by an SKM method (Method 2.8) converge to a point $\mathbf{l} \in P_{A,\mathbf{b}}$.*

PROOF. Let $\mathbf{a} \in P_{A,\mathbf{b}}$. Note that the limit, $\lim_{k\to\infty} \|\mathbf{x}_k - \mathbf{a}\| =: r_{\mathbf{a}}$ exists since $\{\|\mathbf{x}_k - \mathbf{a}\|\}$ is bounded and decreasing (with probability 1). Define

$$S(\mathbf{a}) := \{\mathbf{x} : \|\mathbf{x} - \mathbf{a}\| = r_{\mathbf{a}}\} \text{ and } X := \bigcap_{\mathbf{a} \in P_{A,\mathbf{b}}} S(\mathbf{a}).$$

Note that X is not empty since the bounded sequence $\{\mathbf{x}_k\}$ must have a limit point, $\mathbf{l}$, achieving $\|\mathbf{l} - \mathbf{a}\| = r_{\mathbf{a}}$. Moreover, suppose there were two such points, $\mathbf{l}, \mathbf{l}' \in X$. Define $\pi := \{\mathbf{x} : \|\mathbf{l} - \mathbf{x}\| = \|\mathbf{l}' - \mathbf{x}\|\}$ to be the hyperplane of points equidistance between $\mathbf{l}, \mathbf{l}'$. Then for $\mathbf{a} \in P_{A,\mathbf{b}}$, we have $\mathbf{l}, \mathbf{l}' \in S(\mathbf{a})$. Hence, $\mathbf{a} \in \pi$ and we have that $P_{A,\mathbf{b}} \subset \pi$, which contradicts the full dimensionality of $P_{A,\mathbf{b}}$. Thus X contains only one point, $\mathbf{l}$, and it must be a limit point of $\{\mathbf{x}_k\}$. Now, since $\{\mathbf{x}_k\}$ is converging to $P_{A,\mathbf{b}}$ (with probability one), we must have that $\mathbf{l} \in P_{A,\mathbf{b}}$.

Now, suppose that $\mathbf{x}_k \not\to \mathbf{l}$ (i.e., only a subsequence of $\{\mathbf{x}_k\}$ converges to $\mathbf{l}$). Thus, there exists an $\epsilon > 0$ so that for all K there exists $k \geq K$ with $\|\mathbf{x}_k - \mathbf{l}\| > \epsilon$. However, there exists a subsequence of $\{\mathbf{x}_k\}$ which is converging to $\mathbf{l}$, so there must exist some K_1 with $\|\mathbf{x}_{K_1} - \mathbf{l}\| < \epsilon$. Thus, at some point the sequence $\|\mathbf{x}_k - \mathbf{l}\|$ must increase, which contradicts Lemma 2.3.3. Hence, $\mathbf{x}_k \to \mathbf{l}$. □

Lemma 2.3.5. *Let $\mathbf{l}$ be the limit point of the $\{\mathbf{x}_k\}$. There exists an index K so that if $\mathbf{a}_j^T \mathbf{l} < b_j$ then $\mathbf{a}_j^T \mathbf{x}_k \leq b_j$ for all $k \geq K$.*

PROOF. This is obvious from $\mathbf{x}_k \to \mathbf{l}$. □

We would like to conclude with a small "qualitative" proposition that indicates there are two stages of behavior of the SKM algorithms. After the K-th iteration the point is converging to a particular face of the polyhedron. At that moment one has essentially reduced the calculation to an equality system problem, because the inequalities that define the face of convergence need to be met with equality in order to reach the polyhedron.

Proposition 2.3.6. *If the feasible region $P_{A,\mathbf{b}}$ is generic and nonempty (i.e., full-dimensional and every vertex satisfies exactly n constraints with equality), then an SKM method (Method 2.8) with samples of size $\beta \leq m-n$ will converge to a single face F of $P_{A,\mathbf{b}}$ and all but the constraints defining F will eventually be satisfied. Thus, the method is guaranteed an increased convergence rate after some index K; for $k \geq K$*

$$\mathbb{E}[d(\mathbf{x}_k, P_{A,\mathbf{b}})^2] \leq \left(1 - \frac{2\lambda - \lambda^2}{mL_2^2}\right)^K \left(1 - \frac{2\lambda - \lambda^2}{(m-\beta+1)L_2^2}\right)^{k-K} d(\mathbf{x}_0, P_{A,\mathbf{b}})^2.$$

PROOF. Since a generic polyhedron is full-dimensional, by Lemma 2.3.4, we have that the SKM method iterates converge to a point on the boundary of $P_{A,\mathbf{b}}$, $\mathbf{l}$. Now, since this $\mathbf{l}$ lies on a face of P and P is generic, this face is defined by at most n constraints. By Lemma 2.3.5, there exists K so that for $k \geq K$ at least $m - n$ of the constraints have been satisfied. Thus, our proposition follows from Theorem 2.3.1. □

2.3.2. Finiteness. Our second main theoretical result notes that, for rational data, one can provide a *certificate of feasibility* after finitely many iterations of SKM. Recall the definition of the maximum violation of a point $\mathbf{x} \in \mathbb{R}^n$ given in Section 1.2.2.3, $\theta(\mathbf{x}) = \max\{0, \max_{i\in[m]}\{\mathbf{a}_i^T\mathbf{x} - b_i\}\}$. Additionally, recall the following lemma which demonstrates that to detect feasibility of an `LF`, one need only find a certificate of feasibility, a point $\mathbf{x}$ so that $\theta(\mathbf{x}) < 2^{-\sigma_{A,\mathbf{b}}+1}$.

Lemma 2.3.7. *If the polyhedron, $P_{A,\mathbf{b}}$, defined by the rational system, $A\mathbf{x} \leq \mathbf{b}$, is empty, then for all $\mathbf{x} \in \mathbb{R}^n$, the maximum violation satisfies $\theta(\mathbf{x}) \geq 2^{-(\sigma_{A,\mathbf{b}})+1}$.*

Our second main theorem demonstrates that we expect to detect feasibility of a feasible `LF` in finitely many iterations of SKM and can bound the probability that an iterate will not be a certificate of feasibility. Thus, if SKM does not provide a certificate of feasibility, this result provides a level of confidence that the `LF` is infeasible.

Theorem 2.3.8. *Suppose $A, \mathbf{b}$ are rational matrices and that we run an SKM method (Method 2.8) on the normalized system $\tilde{A}\mathbf{x} \leq \tilde{\mathbf{b}}$ $\left(\text{where } \tilde{\mathbf{a}}_i = \frac{1}{\|\mathbf{a}_i\|}\mathbf{a}_i \text{ and } \tilde{b}_i = \frac{1}{\|\mathbf{a}_i\|}b_i\right)$ with $\mathbf{x}_0 = \mathbf{0}$. Suppose the number of iterations k satisfies*

$$k > \frac{4\sigma_{A,\mathbf{b}} - 4 - \log n + 2\log\left(\max_{j\in[m]}\|\mathbf{a}_j\|\right)}{\log\left(\frac{mL_2^2}{mL_2^2 - 2\lambda + \lambda^2}\right)}.$$

If the system $A\mathbf{x} \leq \mathbf{b}$ is feasible, the probability that the iterate, $\mathbf{x}_k$, is not a certificate of feasibility is at most

$$\frac{\max\|\mathbf{a}_j\|\, 2^{2\sigma_{A,\mathbf{b}}-2}}{n^{1/2}}\left(1 - \frac{2\lambda - \lambda^2}{mL_2^2}\right)^{k/2},$$

which decreases with k.

Now, we show that the general SKM method (when $\lambda \neq 2$) on rational data is finite in expectation.

We will additionally make use of the following lemma (which is key in demonstrating that Khachiyan's ellipsoidal algorithm is finite and polynomial-time [**Kha79**]) in our proof:

Lemma 2.3.9. *If the rational system $A\mathbf{x} \leq \mathbf{b}$ is feasible, then there is a feasible solution $\hat{\mathbf{x}}$ whose coordinates satisfy $|\hat{x}_j| \leq \frac{2^{\sigma_{A,\mathbf{b}}}}{2n}$ for $j = 1, ..., n$.*

Using the bound on the expected distance to the solution polyhedron, $P_{A,\mathbf{b}}$, we can show a bound on the expected number of iterations needed to detect feasibility (which does not depend on the size of block selected).

PROOF. (of Theorem 2.3.8) First, note that if $\tilde{P} := \{\mathbf{x}|\tilde{A}\mathbf{x} \leq \tilde{\mathbf{b}}\}$, then $P_{A,\mathbf{b}} = \tilde{P}$. Then, by Lemma 2.3.9, if $\tilde{A}\mathbf{x} \leq \tilde{\mathbf{b}}$ is feasible (so $A\mathbf{x} \leq \mathbf{b}$ is feasible) then there is a feasible solution $\hat{\mathbf{x}}$ with $|\hat{x}_j| < \frac{2^{\sigma}_{A,\mathbf{b}}}{2n}$ for all $j = 1, 2, ..., n$ (here $\sigma_{A,\mathbf{b}}$ is the binary encoding length for the unnormalized $A, \mathbf{b}$). Thus, since $\mathbf{x}_0 = \mathbf{0}$,

$$d(\mathbf{x}_0, P_{A,\mathbf{b}}) = d(\mathbf{x}_0, \tilde{P}) \leq \|\hat{\mathbf{x}}\| \leq \frac{2^{\sigma_{A,\mathbf{b}}-1}}{n^{1/2}}.$$

Now, define $\tilde{\theta}(\mathbf{x})$ to be the maximum violation for the new, normalized system $\tilde{A}\mathbf{x} \leq \tilde{\mathbf{b}}$,

$$\tilde{\theta}(\mathbf{x}) := \max\{0, \max_{i\in[m]} \tilde{\mathbf{a}}_i^T\mathbf{x} - \tilde{b}_i\} = \max\left\{0, \max_{i\in[m]} \frac{\mathbf{a}_i^T\mathbf{x} - b_i}{\|\mathbf{a}_i\|}\right\}.$$

By Lemma 2.3.7, if the system $\tilde{A}\mathbf{x} \leq \tilde{\mathbf{b}}$ is infeasible (so $A\mathbf{x} \leq \mathbf{b}$ is infeasible), then

$$\tilde{\theta}(\mathbf{x}) = \max\{0, \max_{i\in[m]} \frac{\mathbf{a}_i^T\mathbf{x} - b_i}{\|\mathbf{a}_i\|}\} \geq \frac{\max\{0, \max_{i\in[m]} \mathbf{a}_i^T\mathbf{x} - b_i\}}{\max_{j\in[m]}\|\mathbf{a}_j\|} = \frac{\theta(\mathbf{x})}{\max_{j\in[m]}\|\mathbf{a}_j\|} \geq \frac{2^{1-\sigma_{A,\mathbf{b}}}}{\max_{j\in[m]}\|\mathbf{a}_j\|}.$$

When running SKM on $\tilde{A}\mathbf{x} \leq \tilde{\mathbf{b}}$, we can conclude that the system is feasible when $\tilde{\theta}(\mathbf{x}) < \frac{2^{1-\sigma_{A,\mathbf{b}}}}{\max_{j\in[m]}\|\mathbf{a}_j\|}$. Now, since every point of $P_{A,\mathbf{b}}$ is inside the half-space defined by $\{\mathbf{x}|\tilde{\mathbf{a}}_i^T\mathbf{x} \leq \tilde{b}_i\}$ for all $i = 1, \cdots, m$, we have $\tilde{\theta}(\mathbf{x}) = \max\{0, \max_{i\in[m]} \tilde{\mathbf{a}}_i^T\mathbf{x} - \tilde{b}_i\} \leq d(\mathbf{x}, P_{A,\mathbf{b}})$. Therefore, if $A\mathbf{x} \leq \mathbf{b}$ is feasible, then

$$\mathbb{E}(\tilde{\theta}(\mathbf{x}_k)) \leq \mathbb{E}(d(\mathbf{x}_k, P_{A,\mathbf{b}})) \leq \left(1 - \frac{2\lambda - \lambda^2}{mL_2^2}\right)^{k/2} d(\mathbf{x}_0, P_{A,\mathbf{b}}) \leq \left(1 - \frac{2\lambda - \lambda^2}{mL_2^2}\right)^{k/2} \frac{2^{\sigma_{A,\mathbf{b}}-1}}{n^{1/2}},$$

where the second inequality follows from Theorem 2.3.1 and the third inequality follows from Lemma 2.3.9 and the discussion above.

Now, we anticipate to have detected feasibility when $\mathbb{E}(\tilde{\theta}(\mathbf{x}_k)) < \frac{2^{1-\sigma_{A,\mathbf{b}}}}{\max\limits_{j\in[m]}\|\mathbf{a}_j\|}$, which is true for

$$k > \frac{4\sigma_{A,\mathbf{b}} - 4 - \log n + 2\log\left(\max\limits_{j\in[m]}\|\mathbf{a}_j\|\right)}{\log\left(\frac{mL_2^2}{mL_2^2-2\lambda+\lambda^2}\right)}.$$

Furthermore, by Markov's inequality (see e.g., [**She02**, Section 8.2]), if the system $A\mathbf{x} \leq \mathbf{b}$ is feasible, then the probability of not having a certificate of feasibility is bounded:

$$\mathbb{P}\left(\tilde{\theta}(\mathbf{x}_k) \geq \frac{2^{1-\sigma_{A,\mathbf{b}}}}{\max\limits_{j\in[m]}\|\mathbf{a}_j\|}\right) \leq \frac{\mathbb{E}(\tilde{\theta}(\mathbf{x}_k))}{\frac{2^{1-\sigma_{A,\mathbf{b}}}}{\max\limits_{j\in[m]}\|\mathbf{a}_j\|}} < \frac{\left(1-\frac{2\lambda-\lambda^2}{mL_2^2}\right)^{k/2}\frac{2^{\sigma_{A,\mathbf{b}}-1}}{n^{1/2}}}{\frac{2^{1-\sigma_{A,\mathbf{b}}}}{\max\limits_{j\in[m]}\|\mathbf{a}_j\|}} = \frac{2^{2\sigma_{A,\mathbf{b}}-2}\max\limits_{}\|\mathbf{a}_j\|}{n^{1/2}}\left(1-\frac{2\lambda-\lambda^2}{mL_2^2}\right)^{k/2}.$$

This completes the proof. □

2.3.3. Termination of SKM Reflection Method. Now, as Motzkin and Schoenberg [**MS54**] showed that their method (with $\lambda = 2$) is finite, we show that SKM with $\lambda = 2$ is finite for full-dimensional systems.

Theorem 2.3.10. *When the polyhedron, $P_{A,\mathbf{b}}$ is full-dimensional, SKM (Method 2.8) with $\lambda = 2$ terminates with a solution.*

We prove this in a manner similar to [**Agm54**].

Proof. By way of contradiction, assume the sequence $\{\mathbf{x}_k\}$ is infinite. By Lemma 2.3.4, we have that the sequence $\{\mathbf{x}_k\}$ converges to $\mathbf{l} \in P_{A,\mathbf{b}}$. Since we assumed $\{\mathbf{x}_k\}$ is infinite, we know that $\mathbf{x}_k \notin P_{A,\mathbf{b}}$ for all k. Thus, since $\mathbf{x}_k \to \mathbf{l} \in P_{A,\mathbf{b}}$, we have that $\mathbf{l}$ is on the boundary of $P_{A,\mathbf{b}}$. Now, since $\lim_{k\to\infty}\mathbf{x}_k = \mathbf{l}$ and $\mathbf{x}_k$ is obtained from $\mathbf{x}_{k-1}$ by reflection across the hyperplane $H_{\mathbf{a}_{i_k},b_{i_k}}$, we find that $\lim_{k\to\infty}\operatorname{dist}(\mathbf{l}, H_{\mathbf{a}_{i_k},b_{i_k}}) = 0$. However, since the family of hyperplanes is finite, there is some smallest N so that for $k > N$, $\operatorname{dist}(\mathbf{l}, H_{\mathbf{a}_{i_k},b_{i_k}}) = 0$ and $\mathbf{l} \in H_{\mathbf{a}_{i_k},b_{i_k}}$. However, all $\mathbf{x}_k$ are obtained from $\mathbf{x}_{k-1}$ by reflection across the hyperplane $H_{\mathbf{a}_{i_k},b_{i_k}}$. We have that for $k > N$, $\|\mathbf{x}_k - \mathbf{l}\| = \|\mathbf{x}_N - \mathbf{l}\|$ since $\mathbf{l} \in H_{\mathbf{a}_{i_k},b_{i_k}}$ which contradicts $\mathbf{x}_k \to \mathbf{l}$. Thus, $\{\mathbf{x}_k\}$ is finite. □

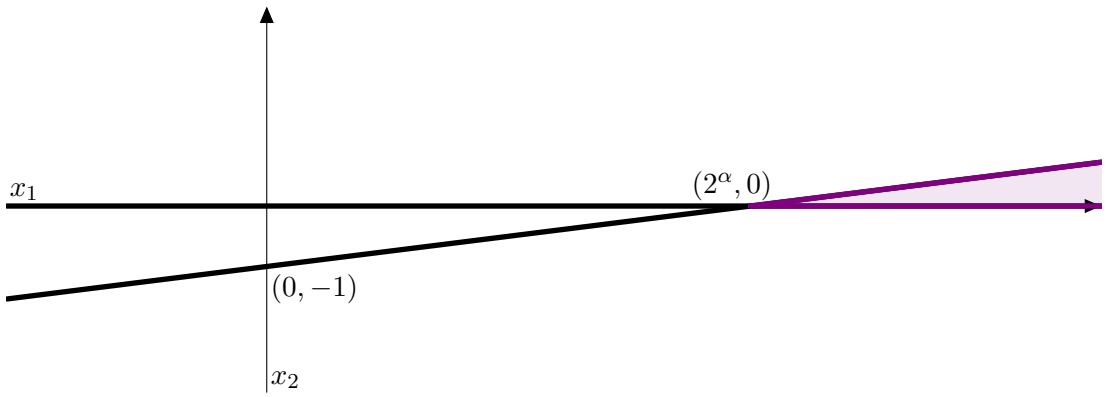

FIGURE 2.27. Telgen's example that iterative projection methods are not polynomial.

Now, note that we have shown that the SKM reflection method ($\lambda = 2$) terminates with a solution, but not that it is polynomially bounded. Telgen showed that Motzkin's method with $\lambda = 1$ requires exponentially many iterations on certain classes of problems [**Tel82**]. Of course, in the worst case, these problems will then require exponentially many iterations by the Sampling Kaczmarz-Motzkin method. We provide a figure of Telgen's example in Figure 2.27.

2.4. Experimental Results

The final contribution of this subsection is a small computational study. The main purpose of our experiments is not to compare the running times versus established methods. Rather, we wanted to determine how our new algorithms compare with the classical algorithms of Agmon, Motzkin and Schoenberg, and Kaczmarz. We examine how the sampling and projection parameters affect the performance of SKM. We try different types of data, but we assume in most of the data that the number of rows m is large, much larger than n. The reason is that this is the regime in which the SKM methods are most relevant and often the only alternative. Iterative projection methods are truly interesting in cases where the number of constraints is very large (possibly so large it is unreadable in memory) or when the constraints can only be sampled due to uncertainty or partial information. Such regimes arise naturally in applications of machine learning [**CE14**] and in online linear programming (see [**AWY14**] and its references). Finally, it has already been shown in prior experiments that, for typical small values of m, n where the system can be read entirely, iterative

projection methods are not able to compete with the simplex method (see [**BDJ14**, **HMSW53**]). Here we compare our SKM code with MATLAB's interior-point methods and active set methods code. We also compare SKM with another iterative projection method, the block Kaczmarz method [**NT13**].

We implemented the SKM methods in MATLAB [**MAT16**] on a 32GB RAM 8-node cluster (although we did not exploit any parallelization), each with 12 cores of Intel Xeon E5-2640 v2 CPUs running at 2 GHz, and ran them on systems while varying the projection parameter, λ, and the sample size, β. We divided our tests into three broad categories: random data, non-random data, and comparisons to other methods. Our experiments focus on the regime $m \gg n$, since as mentioned earlier, this is the setting in which iterative methods are usually applied; however, we see similar behavior in the underdetermined setting as well.

2.4.1. Experiments on random data. First we considered systems $A\mathbf{x} \leq \mathbf{b}$ where A has entries consisting of standard normal random variables and $\mathbf{b}$ is chosen to force the system to have a solution set with non-empty interior (we generated a consistent system of equations and then perturbed the right hand side with the absolute value of a standard normal error vector). We additionally considered systems where the rows of A are highly correlated (each row consists only of entries chosen uniformly at random from $[.9, 1]$ or only of entries chosen uniformly at random from $[-1, -.9]$) and $\mathbf{b}$ is chosen as above. We vary the size of $A \in \mathbb{R}^{m\times n}$, which we note in each example presented below.

In Figure 2.28, we provide experimental evidence that for each problem there is an optimal choice for the sample size, β, in terms of computation. We measure the average computational time necessary for SKM with several choices of sample size β to reach halting (positive) residual error 2^{-14} (i.e., $\|(A\mathbf{x}_k - \mathbf{b})^+\| \leq 2^{-14}$). Regardless of choice of projection parameter, λ, we see a minimum for performance occurs for β between 1 and m.

For the experiments in Figures 2.29, 2.30, and 2.31, we fixed the projection parameter at $\lambda = 1.6$ (for reasons discussed below). On the left of Figure 2.30, we see the residual error decreases more quickly per iteration as the sample size, β increases. However, on the right, when measuring the computational time, SKM with $\beta \approx 5000$ performs best.

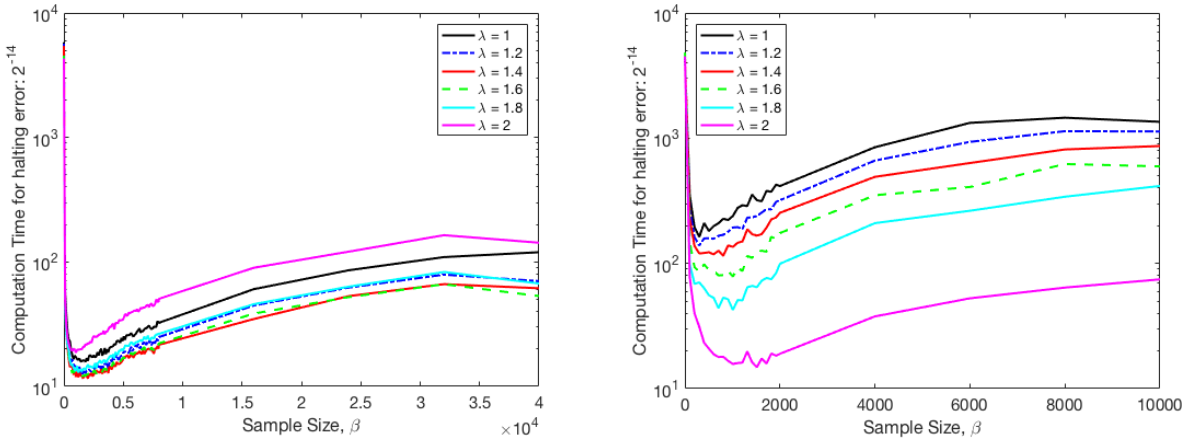

FIGURE 2.28. Left: Average comp. time for SKM on 40000×100 Gaussian system to reach residual error 2^{-14}. Right: Average comp. time for SKM on 10000×100 correlated random system to reach residual error.

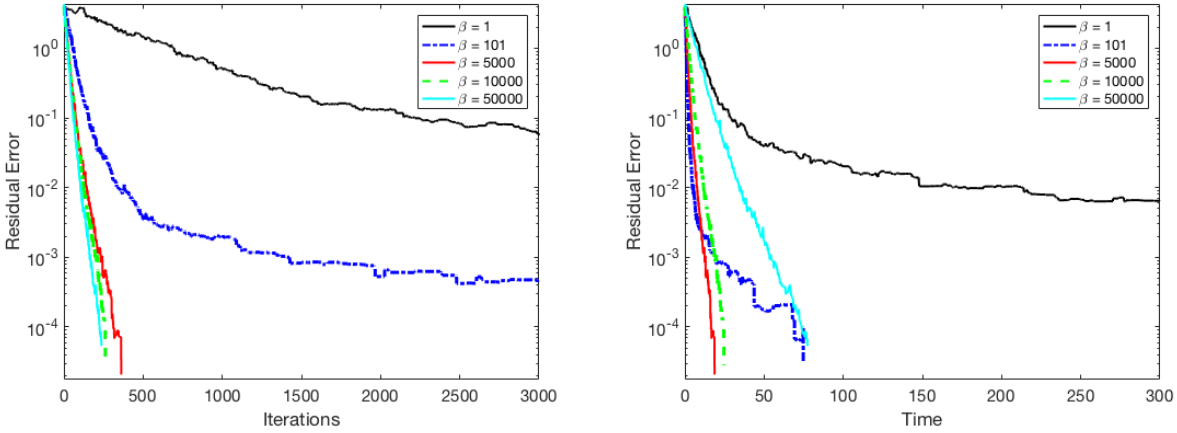

FIGURE 2.29. Left: Iterations vs. residual error for SKM with various sample sizes on 50000×100 Gaussian system. Right: Time vs. residual error.

In Figure 2.31, we ran experiments varying the halting error and see that the sample size selection, β, depends additionally on the desired final distance to the feasible region, $P_{A,\mathbf{b}}$. On the right, we attempted to pinpoint the optimal choice of β by reducing the sample sizes we were considering.

Like [**SV09**], we observe that 'overshooting' ($\lambda > 1$) outperforms other projection parameters, $\lambda \leq 1$. In Figure 2.28, we see that the optimal projection parameter, λ is system dependent. For the experiments in Figure 2.28, we ran SKM on the same system until the iterates had residual error less than 2^{-14} and averaged the computational time taken over ten runs. The best choice of

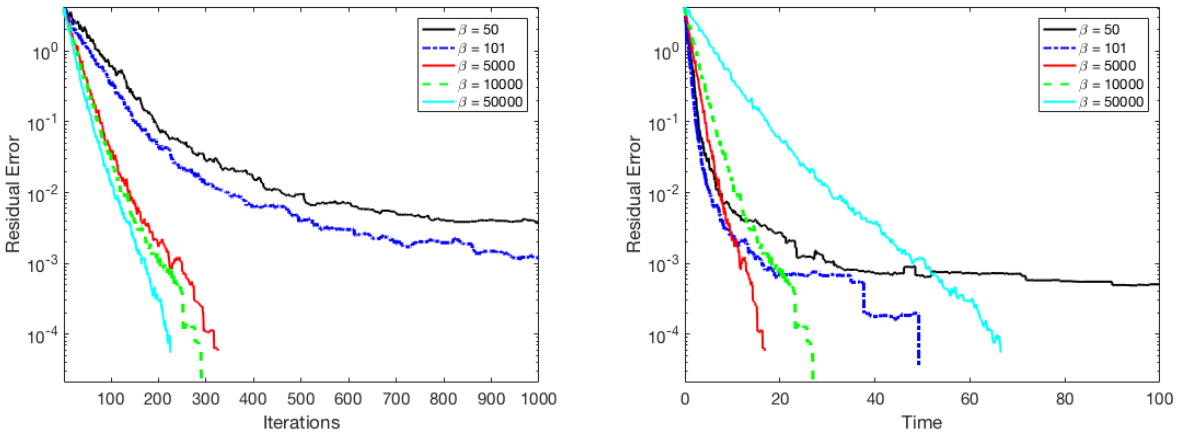

FIGURE 2.30. Left: Iterations vs. residual error for SKM with sample sizes from 50 to m on 50000×100 Gaussian system. Right: Time vs. residual error.

λ differed greatly between the Gaussian random systems and the correlated random systems; for Gaussian systems it was $1.4 < \lambda < 1.6$ while for correlated systems it was $\lambda = 2$.

Our bound on the distance remaining to the feasible region decreases as the number of satisfied constraints increases. In Figure 2.32, we see that the fraction of satisfied constraints initially increased most quickly for SKM with sample size, $1 < \beta < m$ and projection parameter, $\lambda > 1$. On the left, we show that SKM with $\beta = m$ is faster in terms of number of iterations. However, on the right, we show that SKM with $1 < \beta < m$ outperforms $\beta = m$ in terms of time because of its computational cost in each iteration.

2.4.2. Experiments on real data. We consider next some non-random, non-fabricated test problems: support vector machine (SVM) linear classification instances and feasibility problems equivalent to linear programs arising in well-known benchmark libraries.

We first consider instances that fit the classical SVM problem (see [**CE14**]). We used the SKM methods to solve the SVM problem (find a linear classifier) for several data sets from the UCI Machine Learning Repository [**Lic13**]. The first data set is the well-known Wisconsin (Diagnostic) Breast Cancer data set, which includes data points (vectors) whose features (components) are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. Each data point is classified as malignant

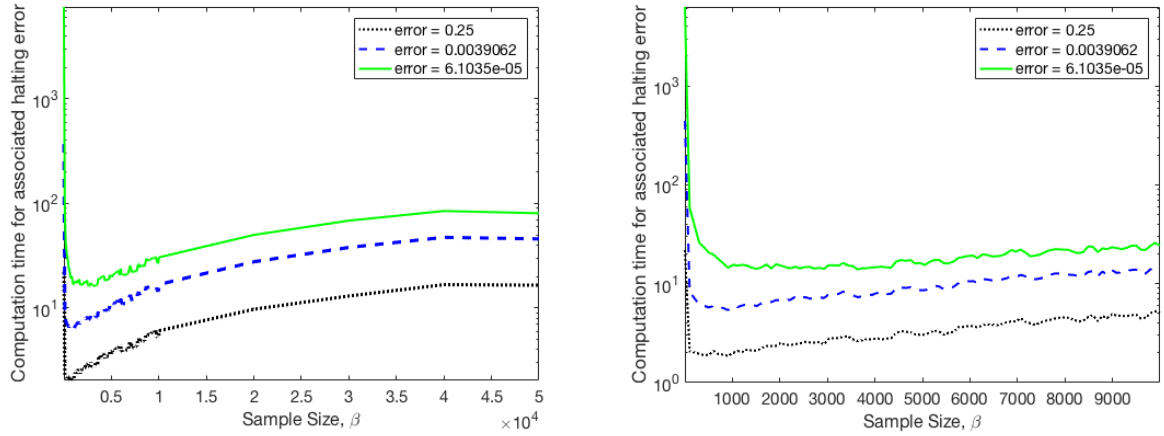

FIGURE 2.31. Left: Average comp. time for SKM on 50000×100 Gaussian system to reach various residual errors for β between 1 and m. Right: Average comp. time for β between 1 and $m/5$.

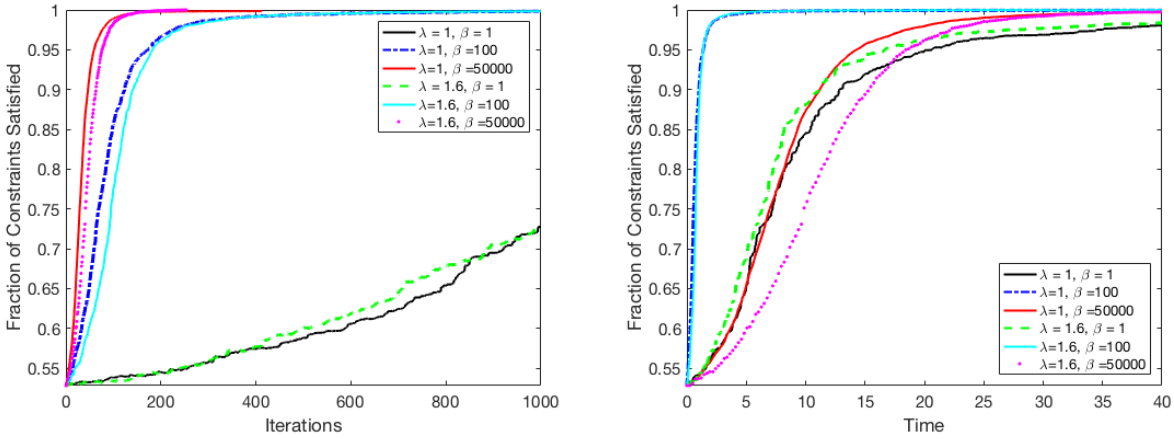

FIGURE 2.32. Left: Iterations vs. fraction of contraints satisfied for SKM methods on 50000×100 Gaussian system. Right: Time vs. fraction of contraints satisfied.

or benign. The resulting solution to the homogenous system of inequalities, $A\mathbf{x} \leq \mathbf{0}$ would ideally define a hyperplane which separates given malignant and benign data points. However, this data set is not separable. The system of inequalities has $m = 569$ constraints (569 data points) and $n = 30$ variables (29 data features). Here, SKM is minimizing the residual norm, $\|A\mathbf{x}_k\|$ and is run until $\|A\mathbf{x}_k\| \leq 0.5$. See Figure 2.33 for results of SKM runtime on this data set.

The second data set is a credit card data set, whose data points include features describing the payment profile of a credit card user and the binary classification is for on-time payment or default

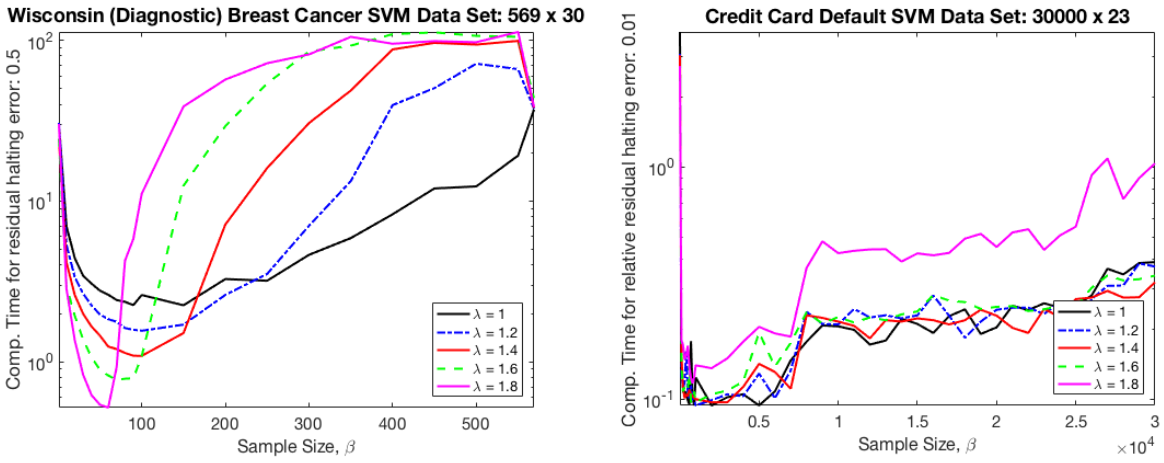

FIGURE 2.33. Left: Breast Cancer Data SVM. Right: Credit Card Data SVM.

payment in a billing cycle [**YL09**]. The resulting solution to the homogenous system of inequalities would ideally define a hyperplane which separates given on-time and default data points. However, this data set is not separable. The system of inequalities has $m = 30000$ (30000 credit card user profiles) and $n = 23$ (22 profile features). Here, SKM is run until $\|A\mathbf{x}_k\|/\|A\mathbf{x}_0\| \leq 0.01$. See Figure 2.33 for results of SKM runtime on this data set.

In the experiments, we again see that for each problem there is an optimal choice for the sample size, β, in terms of smallest computation time. We measure the average computation time necessary for SKM with several choices of sample size β to reach the halting (positive) residual error. Regardless of choice of projection parameter, λ, we see again that best performance occurs for β between 1 and m. Note that the curves are not as smooth as before, which we attribute to the wider irregularity of coefficients, which in turn forces the residual error more to be more dependent on the actual constraints.

We next implemented SKM on several *Netlib* linear programming (LP) problems [**Net**]. Each of these problems was originally formulated as the LP $\min \mathbf{c}^T\mathbf{x}$ subject to $A\mathbf{x} = \mathbf{b}$, $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$ with optimum value p^*. We reformulated these problems as the equivalent linear feasibility problem $\tilde{A}\mathbf{x} \leq \tilde{\mathbf{b}}$ where

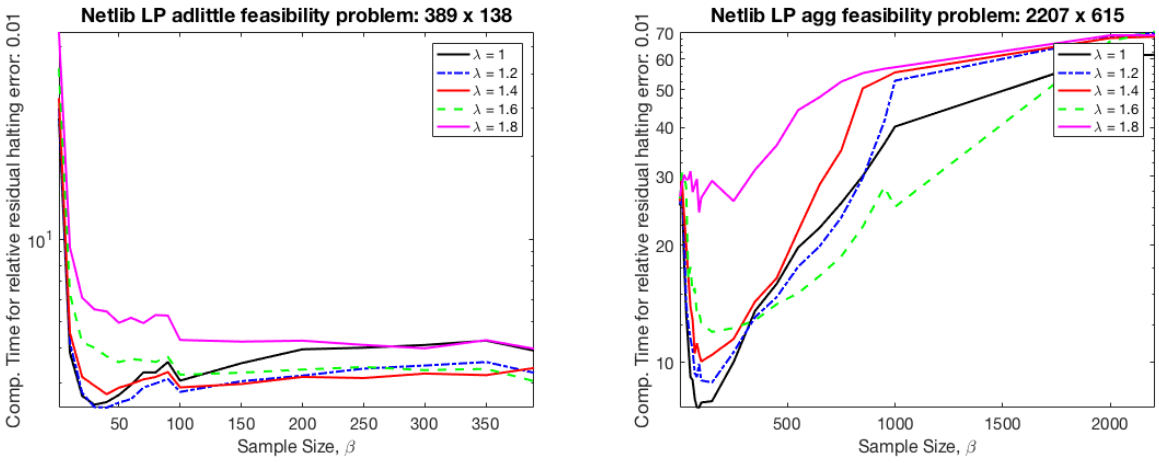

FIGURE 2.34. Left: SKM behavior for *Netlib* LP adlittle. Right: SKM behavior for *Netlib* LP agg

$$\tilde{A} = \begin{bmatrix} A \\ -A \\ I \\ -I \\ \mathbf{c}^T \end{bmatrix} \text{ and } \tilde{\mathbf{b}} = \begin{bmatrix} \mathbf{b} \\ -\mathbf{b} \\ \mathbf{u} \\ -\mathbf{l} \\ p^* \end{bmatrix}.$$

See Figures 2.34, 2.35, 2.36, 2.37, and 2.38 for results of SKM runtime on these problems as we vary β and λ. Once more, regardless of choice of projection parameter, λ, we see optimal performance occurs for β between 1 and m.

It would be possible to handle these equalities without employing our splitting technique to generate inequalities. This splitting technique only increases m ($||A||_F^2$) and does not affect the Hoffman constant, which is $||\tilde{A}^{-1}||_2$ in this case. It may be useful to explore such an extension.

2.4.3. Comparison to existing methods. In Table 2.1, we investigate the performance behavior of SKM versus interior-point and active-set methods on several *Netlib* LPs. For fairness of comparison, we gauge our code written in MATLAB versus the MATLAB Optimization Toolbox function *fmincon*. The function *fmincon* allows a user to select either an 'interior-point' algorithm or an 'active-set' algorithm.

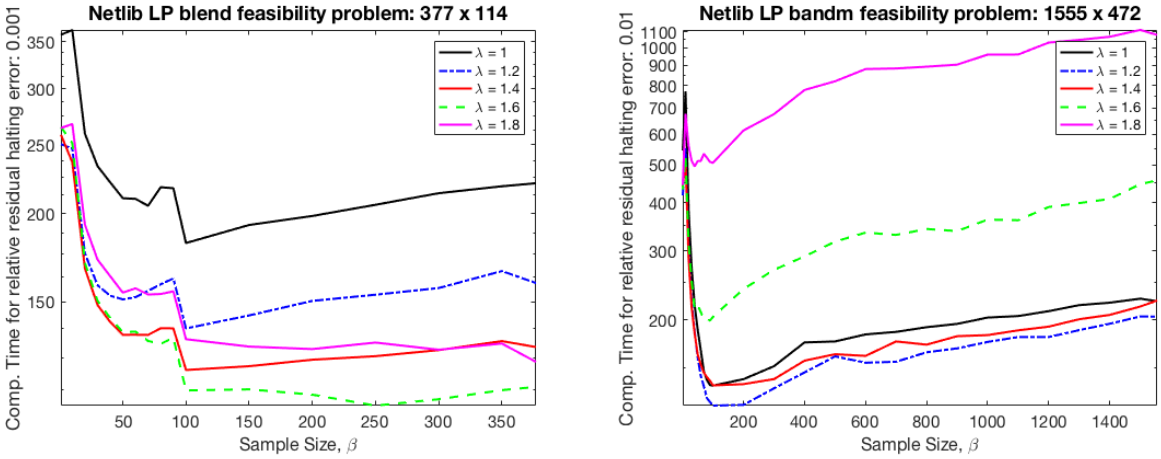

FIGURE 2.35. Left: SKM behavior for *Netlib* LP blend. Right: SKM behavior for *Netlib* LP bandm.

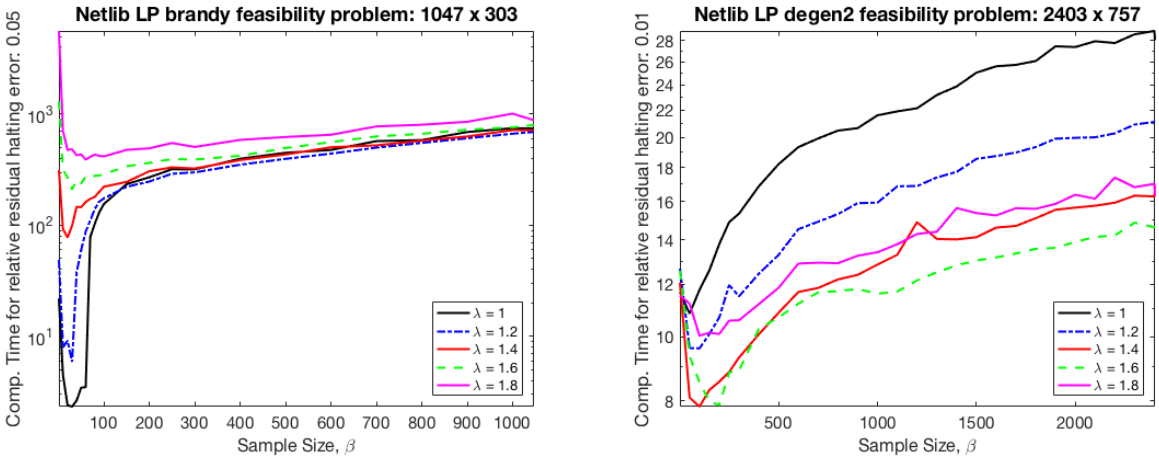

FIGURE 2.36. Left: SKM behavior for *Netlib* LP brandy. Right: SKM behavior for *Netlib* LP degen2.

We first used fmincon to solve the feasibility problem as described in Section 2.4.2 by applying this function to $\min 0$ such that $\tilde{A}\mathbf{x} \leq \tilde{\mathbf{b}}$. However, the interior-point method and active-set method were mostly unable to solve these feasibility form problems. The interior-point algorithm was never able to solve feasibility, due to the fact that the system of equations defined by the KKT conditions in each iteration was numerically singular. Similarly, in most cases, the active-set method was halted in the initial step of finding a feasible point. For fairness of comparison, we do not list these results.

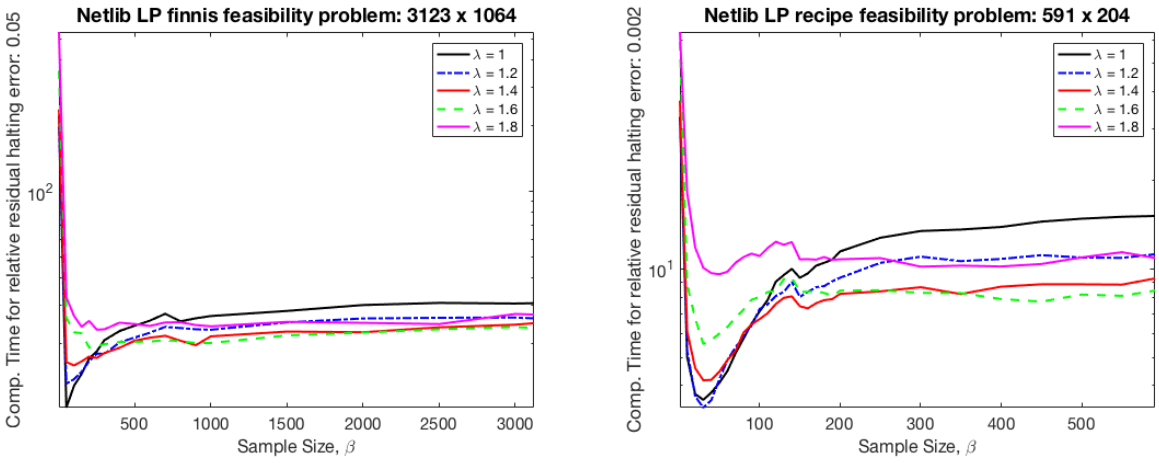

FIGURE 2.37. Left: SKM behavior for *Netlib* LP finnis. Right: SKM behavior for *Netlib* LP recipe.

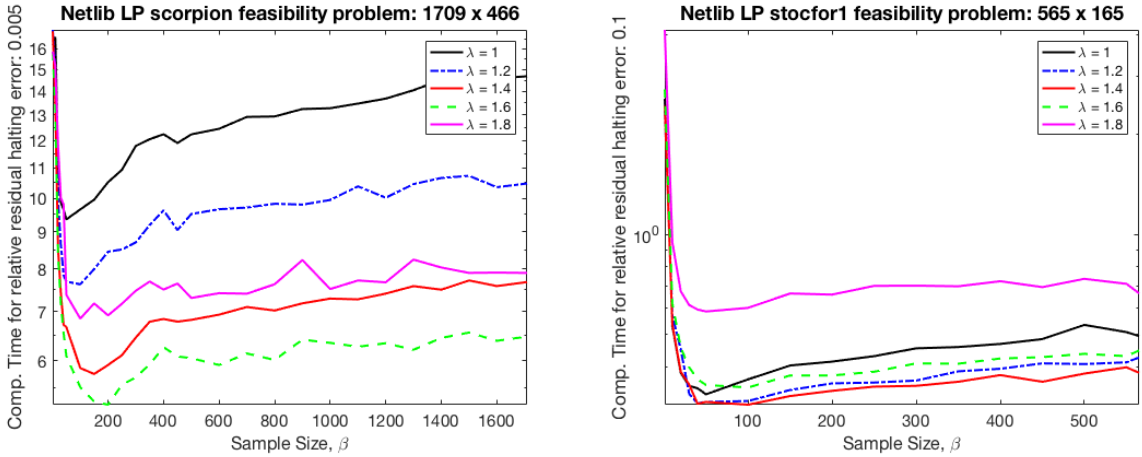

FIGURE 2.38. Left: SKM behavior for *Netlib* LP scorpion. Right: SKM behavior for *Netlib* LP stocfor1.

In Table 2.1, we list CPU timings for the MATLAB interior-point and active-set fmincon algorithms to solve the original optimization LPs ($\min \mathbf{c}^T\mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$), and SKM to solve the equivalent feasibility problem, $\tilde{A}\mathbf{x} \leq \tilde{\mathbf{b}}$, as described in Section 2.4.2. Note that this is not an obvious comparion as SKM is designed for feasibility problems, and in principle, the stopping criterion may force SKM to stop near a feasible point, but not necessarily near an optimum. On the other hand, interior point methods and active set methods decrease the value of the objective and simultaneously solve feasibility. The halting criterion for SKM remains that $\frac{\max(\tilde{A}\mathbf{x}_k - \tilde{\mathbf{b}})}{\max(\tilde{A}\mathbf{x}_0 - \tilde{\mathbf{b}})} \leq \epsilon_{\text{err}}$ where ϵ_{err} is the halting error bound listed for each problem in the table. The halting criterion

Problem Title	Dimensions	Interior-Point	SKM	Active-Set	ϵ_{err}	SKM λ	SKM β
LP adlittle	389×138	2.08	0.29	1.85	10^{-2}	1.2	30
LP agg	2207×615	109.54*	20.55	554.52*	10^{-2}	1	100
LP bandm	1555×472	27.21	756.71	518.44*	10^{-2}	1.2	100
LP blend	337×114	1.87	367.33	2.20	10^{-3}	1.6	250
LP brandy	1047×303	21.26	240.83	90.46	0.05	1	20
LP degen2	2403×757	6.70	22.41	25725.23	10^{-2}	1.4	100
LP finnis	3123×1064	115.47*	13.76	431380.82*	0.05	1	50
LP recipe	591×204	2.81	2.62	5.56	0.002	1.2	30
LP scorpion	1709×466	11.80	22.22	10.38	0.005	1.6	200
LP stocfor1	565×165	0.53	0.34	3.29	0.1	1.4	50

TABLE 2.1. CPU time comparisons for MATLAB methods solving LP and SKM solving feasibility.

$*$ indicates that the solver did not solve the problem to the desired accuracy due to reaching an upper limit on function evaluations of 100000

for the fmincon algorithms is that $\frac{\max(A\mathbf{x}_k-\mathbf{b},\mathbf{l}-\mathbf{x}_k,\mathbf{x}_k-\mathbf{u})}{\max(A\mathbf{x}_0-\mathbf{b},\mathbf{l}-\mathbf{x}_0,\mathbf{x}_0-\mathbf{u})} \leq \epsilon_{\text{err}}$ and $\frac{\mathbf{c}^T\mathbf{x}_k}{\mathbf{c}^T\mathbf{x}_0} \leq \epsilon_{\text{err}}$ where ϵ_{err} is the halting error bound listed for each problem in the table. Each of the methods were started with the same initial point far from the feasible region. The experiments show our SKM method compares favorably with the other codes.

For the experiments in Table 2.1, the interior-point method was not able to solve for LP agg and LP finnis before hitting the upper bound on function evaluations due to slow progression towards feasibility. The active-set method was not able to solve for LP agg, LP bandm and LP finnis before hitting the upper bound on function evaluations due to a very slow (or incomplete) initial step in finding a feasible point. As mentioned before, the methods were initialized with a point far from the feasible region which may have contributed to the interior-point and active-set methods poor performances.

In Figures 2.39 and 2.40, we compare the SKM method to the block Kaczmarz (BK) method (with randomly selected blocks). Here we solve only systems of linear equations, not inequalities, and we consider only random data as our implemented block Kaczmarz method selects blocks at random. We see that the performance of the block Kaczmarz method is closely linked to the conditioning of the selected blocks, as the BK method must solve a system of equations in each iteration, rather than one equation as for SKM.

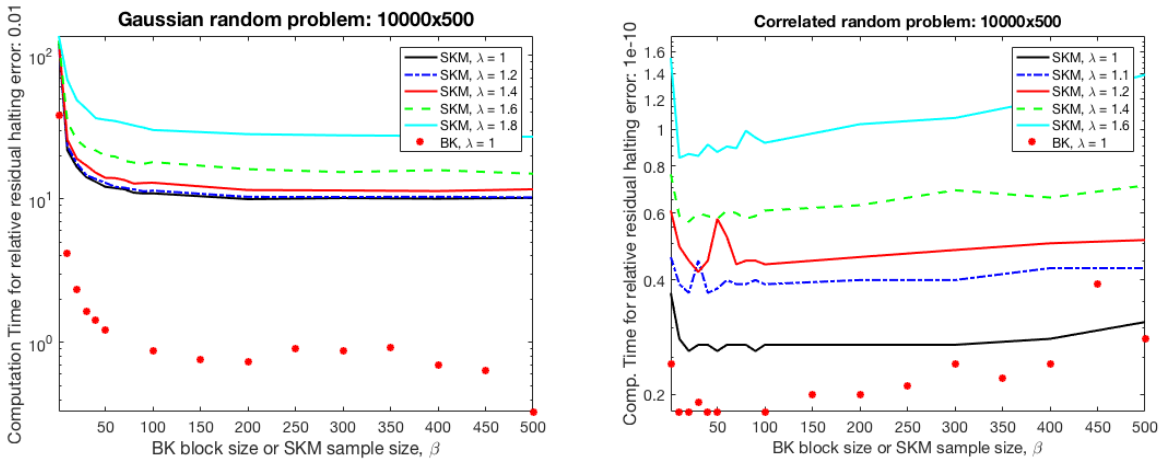

FIGURE 2.39. Comparison of SKM method runtimes with various choices of sample size, β and block Kaczmarz method runtimes with various choices of block size on different types of random systems. Left: Gaussian random system. Right: Correlated random system with entries chosen uniformly from $[0.9, 0.9 + 10^{-5}]$.

For the Gaussian random data, the selected blocks are well-conditioned and with high probability, the block division has formed a row-paving of the matrix. Here we see that BK outperforms SKM. However, when we consider correlated data instead, the behavior of BK reflects the poor conditioning of the blocks. In the three included figures, we test with correlated matrices with increasingly poorly conditioned blocks. If the blocks are numerically ill-conditioned, SKM is able to outperform BK. For systems of equations in which blocks are well conditioned and easy to identify, BK has advantages over SKM. However, if you are unable or unwilling to find a good paving, SKM can be used and is able to outperform BK. When BK is used with inequalities, a paving with more strict geometric properties must be found, and this can be computationally challenging, see [**BN**] for details. SKM avoids this issue.

2.4.4. Remarks on Parameter Selection.

2.4.4.1. *Choice of* β. As observed by Theorem 2.3.1, the sample size β used in each iteration of SKM plays a role in the convergence rate of the method. By the definition of V_{k-1} in Theorem 2.3.1 and by the bound in Proposition 2.3.6 the choice $\beta = m$ yields the fastest convergence rate. Indeed, this coincides with the classical method of Motzkin; one selects the most violated constraint out of *all* the constraints in each iteration. However, it is also clear that this choice of β is extremely

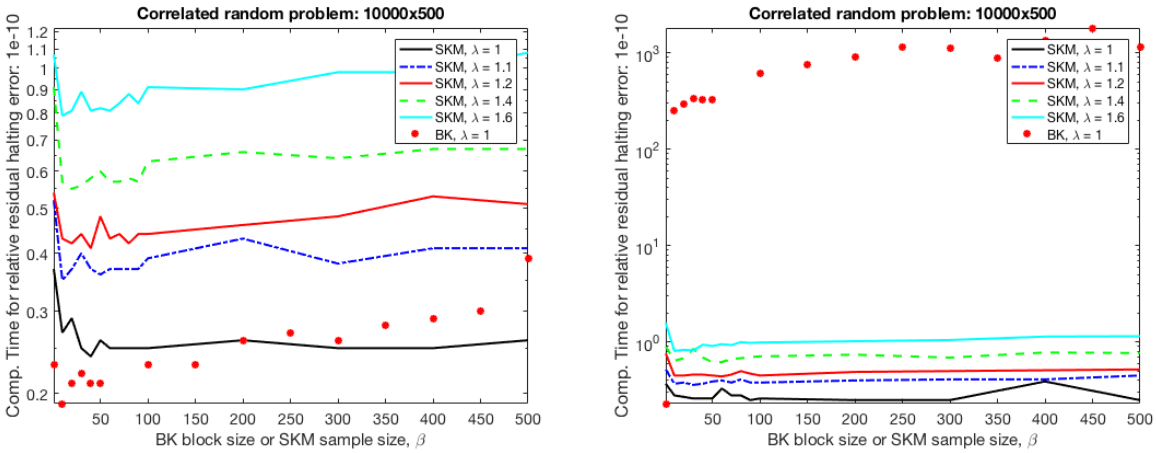

FIGURE 2.40. Left: Correlated random system with entries chosen uniformly from $[0.9, 0.9 + 10^{-16}]$. Right: Correlated random system with entries chosen uniformly from $[0.9, 0.9 + 10^{-20}]$.

costly in terms of computation, and so the more relevant question is about the choice of β that optimizes the convergence rate in terms of total computation.

To gain an understanding of the tradeoff between convergence rate and computation time in terms of the parameter β, we consider a fixed iteration j and for simplicity choose $\lambda = 1$. Denote the residual by $\mathbf{r} := (A\mathbf{x}_j - \mathbf{b})^+$, and suppose s inequalities are satisfied in this iteration; that is, $\mathbf{r}$ has s zero entries. Write $\mathbf{r}_{\tau_j}$ for the portion of the residual selected in Step 3 of SKM (so $|\tau_j| = \beta$). Then as seen from Equation (2.10) in the proof of Theorem 2.3.1, the expected improvement (i.e., $d(\mathbf{x}_j, P_{A,\mathbf{b}}) - d(\mathbf{x}_{j+1}, P_{A,\mathbf{b}})$) made in this iteration is given by $\mathbb{E}\|\mathbf{r}_{\tau_j}\|_\infty^2$. Expressing this quantity as in (2.11) along with Lemma 2.3.2, one sees that the worst case improvement will be made when the $m - s$ non-zero components of the residual vector are all the same magnitude (i.e., $\mathbb{E}\|\mathbf{r}_{\tau_j}\|_\infty \geq \frac{1}{m-s}\|\mathbf{r}\|_1$). We thus focus on this scenario in tuning β to obtain a minimax heuristic for the optimal selection. We model the computation count in a fixed iteration as some constant computation time for overhead C plus a factor that scales like $n\beta$, since checking the feasibility of β constraints takes time $O(n\beta)$. We therefore seek a value for β that maximizes the ratio of improvement made and computation cost:

$$\text{gain}(\beta) := \frac{\mathbb{E}\|\mathbf{r}_{\tau_j}\|_\infty^2}{C + cn\beta}, \tag{2.15}$$

when the residual $\mathbf{r}$ consists of $m - s$ non-zeros of the same magnitude. Call the support of the residual $T := \text{supp}(\mathbf{r}) = \{i : r_i \neq 0\}$. Without loss of generality, we may assume that the magnitude of these entries is just 1. In that case, one easily computes that

$$\mathbb{E}\|\mathbf{r}_{\tau_j}\|_\infty^2 = \mathbb{P}(T \cap \tau_j \neq \emptyset) = \begin{cases} 1 - \dfrac{\binom{s}{\beta}}{\binom{m}{\beta}} \approx 1 - \left(\frac{s}{m}\right)^\beta & \text{if } \beta \leq s, \\ 1 & \text{if } \beta > s, \end{cases}$$

where we have used Stirling's approximation in the first case.

We may now plot the quantity

$$\text{gain}(\beta) \approx \frac{1 - \left(\frac{s}{m}\right)^\beta}{C + cn\beta} \tag{2.16}$$

as a function of β, for various choices of s. Figure 2.41 shows an example of this function for some specific parameter settings. We see that, as in the experiments of Sections 2.4.1 and 2.4.2, optimal β selection need not necessarily be at either of the endpoints $\beta = 1$ or $\beta = m$ (corresponding to classical randomized Kaczmarz and Motzkin's method, respectively). In particular, one observes that as the number of satisfied constraints s increases, the optimal size of β also increases. This of course is not surprising, since with many satisfied constraints if we use a small value of β we are likely to see mostly satisfied constraints in our selection and thus make little to no progress in that iteration. Again, Figure 2.41 is for the worst case scenario when the residual has constant non-zero entries, but serves as a heuristic for how one might tune the choice of β. In particular, it might be worthwhile to increase β throughout the iterations.

2.4.4.2. *Choice of* λ. Additionally, the optimal choice of projection parameter λ is system dependent (e.g., for certain systems, one should choose $\lambda = 1$ while for certain full-dimensional systems, one should choose $\lambda > 1$). Theoretically, the convergence rate we provided in Theorem 2.3.1 depends upon λ in a weak way; one would always choose $\lambda = 1$. However, we see experimentally that overshooting outperforms other choices of λ. Additionally, one can easily imagine that for systems whose polyhedral feasible region is full-dimensional, choosing $\lambda > 1$ will outperform $\lambda \leq 1$, as eventually, the iterates could 'hop' into the feasible region. The proof of Proposition 2.3.6 suggests

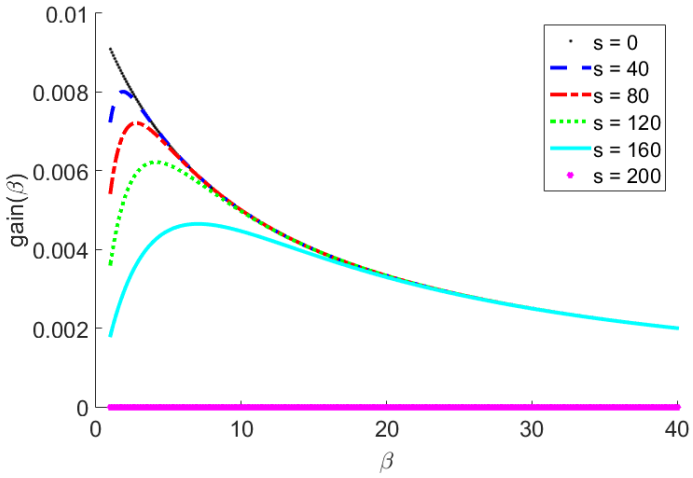

FIGURE 2.41. The quantity gain(β) as in (2.16) as a function of β for various numbers of satisfied constraints s. Here we set $m = 200$, $n = 10$, $c = 1$ and $C = 100$. Optimal values of β maximize the gain function.

a possible reason why we see this in our experiments. This proposition is a consequence of the fact that if the method does not terminate then it will converge to a unique face of P. If $\lambda > 1$, then this face cannot be a facet of P, as if the method converged to such a face, it would eventually terminate, 'hopping' over the facet into P. Thus, for $\lambda > 1$, the number of possible faces of P that the sequence of iterates can converge to is decreased. Further work is needed before defining the optimal choice of λ or β for any class of systems.

CHAPTER 3

Wolfe's Methods for Minimum Norm Point

Now we present our study of one of the most famous algorithms for the minimum norm point problem, Wolfe's algorithm. Wolfe's method is a combinatorial method for solving the minimum norm point problem over a polytope, $P = \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m) \subset \mathbb{R}^n$. It was introduced by P. Wolfe in [**Wol76**]. For convenience of the reader, we first state some definitions and elementary results, and follow with a brief description of Wolfe's method. We then discuss similar methods in convex optimization and present some related results. We will then describe our exponential example in detail, proving the exponential behavior of Wolfe's method as stated in Theorem 1.3.5 [**DLHR17**, **DLHR18**].

3.1. Background

Wolfe's method iteratively solves `MNP` over a sequence of subsets of no more than $n+1$ affinely independent points from $\mathbf{p}_1, ..., \mathbf{p}_m$ and it checks to see if the solution to the subproblem is a solution to the problem over P using the following lemma due to Wolfe. We call this *Wolfe's criterion.*

Lemma 3.1.1 (Wolfe's criterion [**Wol76**])**.** *Let $P = \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m) \subset \mathbb{R}^n$, then $\mathbf{x} \in P$ is the minimum norm point in P if and only if*

$$\mathbf{x}^T \mathbf{p}_j \geq \|\mathbf{x}\|_2^2 \quad \textit{for all} \quad j \in [m].$$

Note that this tells us that if there exists a point p_j so that $x^T p_j < \|x\|_2^2$ (i.e., the hyperplane $\{\mathbf{y} : \mathbf{x}^T \mathbf{y} = \|\mathbf{x}\|_2^2\}$ does not weakly separate P from $\mathbf{0}$) then x is not the minimum norm point in P. We may check optimality of a given point $\mathbf{x}$ by simply computing inner products and comparing to $\|\mathbf{x}\|^2$, so this may be done in strongly-polynomial time. We say that p_j violates Wolfe's criterion and using this point should decrease the minimum norm point of the current subproblem. See Figure 3.1 for an illustration.

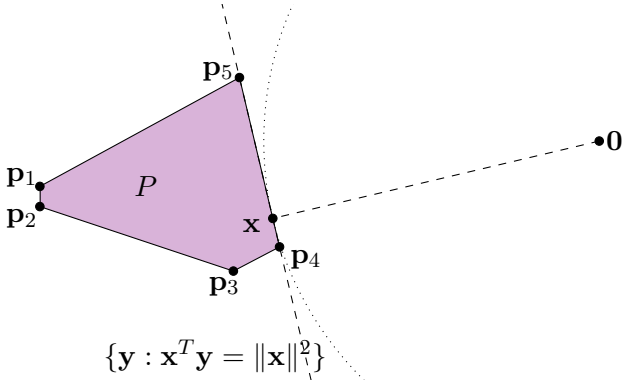

FIGURE 3.1. A visualization of Wolfe's criterion. Note that $\{\mathbf{y} : \mathbf{x}^T\mathbf{y} = \|\mathbf{x}\|^2\}$ weakly separates P from $\mathbf{0}$, so $\mathbf{x}$ is the minimum norm point in P.

It should be observed that just as Wolfe's criterion is a rule to decide optimality over $\text{conv}(P)$, one has a very similar rule for deciding optimality over the affine hull, $\text{aff}(P)$. We state and prove this result below since we do not know of a reference.

Lemma 3.1.2 (Wolfe's criterion for the affine hull)**.** *Let $P = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\} \subseteq \mathbb{R}^d$ be a non-empty finite set of points. Then $\mathbf{x} \in \text{aff}(P)$ is the minimum norm point in $\text{aff}(P)$ iff for all $\mathbf{p}_i \in P$ we have $\mathbf{p}_i^T\mathbf{x} = \|\mathbf{x}\|_2^2$.*

PROOF. Let $\mathbf{p} = \sum_{i=1}^n \rho_i \mathbf{p}_i$ with $\sum_{i=1}^n \rho_i = 1$ be an arbitrary point in $\text{aff}(P)$ and suppose $\mathbf{p}_i^T\mathbf{x} = \|\mathbf{x}\|_2^2$ for $i = 1, 2, ..., n$. We have

$$\mathbf{p}^T\mathbf{x} = \sum_{i=1}^n \rho_i \mathbf{p}_i^T\mathbf{x} = \sum_{i=1}^n \rho_i \|\mathbf{x}\|_2^2 = \|\mathbf{x}\|_2^2.$$

Then $0 \leq \|\mathbf{p} - \mathbf{x}\|_2^2 = \|\mathbf{p}\|_2^2 - 2\mathbf{p}^T\mathbf{x} + \|\mathbf{x}\|_2^2 = \|\mathbf{p}\|_2^2 - \|\mathbf{x}\|_2^2$ and so $\|\mathbf{x}\|_2^2 \leq \|\mathbf{p}\|_2^2$.

Suppose $\mathbf{x} \in \text{aff}(P)$ is the minimum norm point in $\text{aff}(P)$. Suppose that $\mathbf{x}^T(\mathbf{p}_i - \mathbf{x}) \neq 0$ for some $i \in [n]$. First, consider the case when $\mathbf{x}^T(\mathbf{p}_i - \mathbf{x}) > 0$ and define $0 < \epsilon < \frac{2\mathbf{x}^T(\mathbf{p}_i - \mathbf{x})}{\|\mathbf{p}_i - \mathbf{x}\|_2^2}$. Then we have

$$\|(1+\epsilon)\mathbf{x} - \epsilon\mathbf{p}_i\|_2^2 = \|x + \epsilon(x - p_i)\|_2^2 = \|\mathbf{x}\|_2^2 - 2\epsilon\mathbf{x}^T(\mathbf{p}_i - \mathbf{x}) + \epsilon^2\|\mathbf{p}_i - \mathbf{x}\|_2^2 < \|\mathbf{x}\|_2^2$$

since $0 < \epsilon^2\|\mathbf{p}_i - \mathbf{x}\|_2^2 < 2\epsilon x^T(\mathbf{p}_i - \mathbf{x})$. This contradicts our assumption that $\mathbf{x}$ is the minimum norm point in $\text{aff}(P)$. The case when $\mathbf{x}^T(\mathbf{p}_i - \mathbf{x}) < 0$ is likewise proved by considering $\|(1-\epsilon)\mathbf{x} + \epsilon\mathbf{p}_i\|_2^2$ with $0 < \epsilon < -\frac{2\mathbf{x}^T(\mathbf{p}_i - \mathbf{x})}{\|\mathbf{p}_i - \mathbf{x}\|_2^2}$. Thus, we have that $\mathbf{x}^T(\mathbf{p}_i - \mathbf{x}) = 0$. □

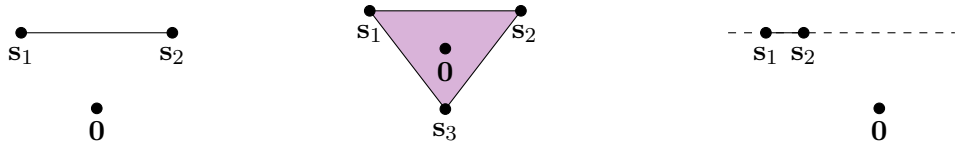

FIGURE 3.2. Left: example of corral. Middle: example of corral. Right: not a corral.

We say a set of affinely independent points S is a *corral* if the affine minimizer of S lies in the relative interior of $\text{conv}(S)$. See Figure 3.2 for two examples of corrals and one non-example in $\mathbb{R}^2$. Note that singletons are always corrals. Carathéodory's theorem (Theorem 1.1.3) implies that the minimum norm point of P will lie in the convex hull of some corral of points among $\mathbf{p}_1, ..., \mathbf{p}_n$. The goal of Wolfe's method is to search for a corral containing the (unique) minimizing point. Wolfe's method operates over subsets of points, checking to see first if they are a corral and then checking to see if they are the optimal corral. Both of these operations are simple and require time polynomial in the dimension and number of points.

Before moving on, we wish to clarify several questions regarding the definition of *corral*. In Wolfe's original paper [**Wol76**], he defined a corral to be a set of affinely independent points, S, whose *convex* minimizer lies in the relative interior of $\text{conv}(S)$. However, in the pseudo-code for his method, he implemented checking whether the current set is a corral by solving for the *affine* minimizer and checking to see if it lies in the relative interior of $\text{conv}(S)$. We comment that these two definitions are in fact the same and provide a proof below in Lemma 3.1.3. Note that computing the affine minimizer in order to check if the current set is a corral requires only solving a system of linear equations (Chapter 4, Lemma 4.1.1); this is exploited by Wolfe's method.

Lemma 3.1.3. *The convex minimizer of a set* $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\}$ *lies in* $\textbf{relint}(\text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m))$ *if and only if the affine minimizer of the points lies in* $\textbf{relint}(\text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m))$.

PROOF. Let $\mathbf{x}$ be the convex minimizer of $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\}$ and $\mathbf{y}$ be the affine minimizer. First, note that $\textbf{relint}(\text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m)) \subseteq \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m)$ and $\|\mathbf{y}\| \leq \|\mathbf{x}\|$ since $\text{conv}(\mathbf{p}_1, ..., \mathbf{p}_m) \subseteq \text{aff}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m)$. If $\mathbf{y} \in \textbf{relint}(\text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m))$ then $\mathbf{x} = \mathbf{y}$.

Assume $\mathbf{y} \notin \textbf{relint}(\text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m))$ and suppose $\mathbf{x} \in \textbf{relint}(\text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m))$. Note that there is some point which is a strict convex combination of $\mathbf{x}$ and $\mathbf{y}$ which lies in $\text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m)$;

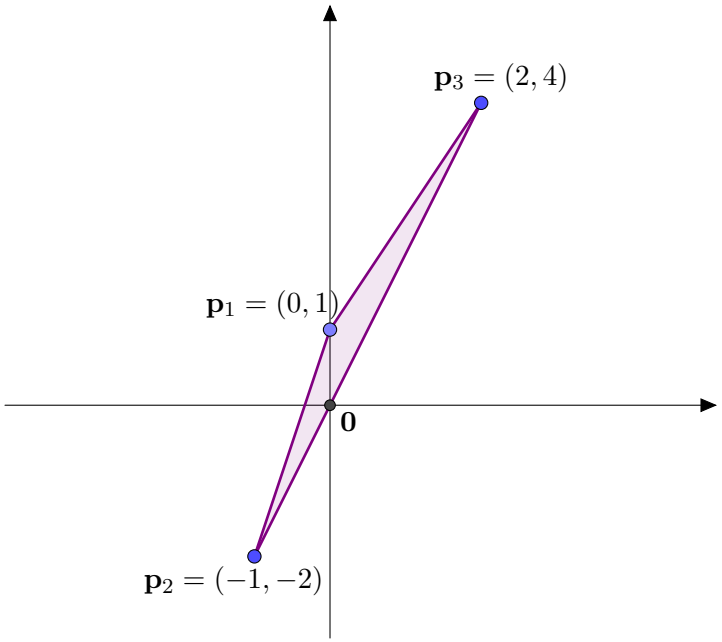

FIGURE 3.3. An example showing that a requiring the affine minimizer of $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ to lie in the convex hull will not ensure that the set is of minimum dimension.

that is, there exists $0 < \alpha < 1$ so that $\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m)$. Finally, note that $\|\mathbf{y}\| < \|\mathbf{y}\|$. Thus, $\|\alpha\mathbf{x} + (1-\alpha)\mathbf{y}\| \leq \alpha\|\mathbf{x}\| + (1-\alpha)\|\mathbf{y}\| < \|\mathbf{x}\|$ which contradicts the assumption that $\mathbf{x}$ was the convex minimizer. □

Finally, note that the definition of a corral requires that the affine minimizer lies in the *relative interior* of the convex hull, and not just within the convex hull, so that corrals will be of minimal size and without unnecessary points. This ensures that the corral of points considered last by Wolfe's method is a subset of points on the face of minimal dimension that contains the minimum norm point. Consider the set of points in Figure 3.3. Note that the affine minimizer and the convex minimizer are the origin. The affine minimizer lies within the convex hull, but the set $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ does not lie in the face of minimum dimension that contains the minimizer. Requiring the affine minimizer to lie within the relative interior of the convex hull ensures that the corral contains no points unnecessary to express the affine minimizer as a convex combination.

3.2. Wolfe's Method

The pseudo-code in Method 3.9 below presents the iterations of Wolfe's method. It is worth noticing that some steps of the method can be implemented in more than one way (for example, the choice of the initial point in line 2) and Wolfe proved that all of them lead to a correct, terminating algorithm. We therefore use the word *method* to encompass all these variations and we discuss specific choices when they are relevant to our analysis of the method. MATLAB code for this algorithm may be found in Appendix A.

Method 3.9 Wolfe's Method [**Wol76**]

1: **procedure** WOLFE($\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m$)
2: Initialize $\mathbf{x} = \mathbf{p}_i$ for some $i \in [m]$, initial corral $C = \{\mathbf{p}_i\}$, $I = \{i\}$, $\lambda = \mathbf{e}_i$, $\alpha = \mathbf{0}$.
3: **while** $\mathbf{x} \neq \mathbf{0}$ and there exists $\mathbf{p}_j$ with $\mathbf{x}^T\mathbf{p}_j < \|\mathbf{x}\|_2^2$ **do**
4: Add $\mathbf{p}_j$ to the potential corral: $C = C \cup \{\mathbf{p}_j\}$, $I = I \cup \{j\}$.
5: Find the affine minimizer of C, $\mathbf{y} = \mathbf{argmin}_{\mathbf{y} \in \text{aff}(C)} \|\mathbf{y}\|_2$, and the affine coefficients, α.
6: **while** $\mathbf{y}$ is not a strict convex combination of points in C; $\alpha_i \leq 0$ for some $i \in I$ **do**
7: Find $\mathbf{z}$, closest point to $\mathbf{y}$ on $[\mathbf{x}, \mathbf{y}] \cap \text{conv}(C)$; $\mathbf{z} = \theta\mathbf{y} + (1-\theta)\mathbf{x}$, $\theta = \min_{i \in I: \alpha_i \leq 0} \frac{\lambda_i}{\lambda_i - \alpha_i}$.
8: Select $\mathbf{p}_i \in \{\mathbf{p}_j \in C : \theta\alpha_j + (1-\theta)\lambda_j = 0\}$.
9: Remove this point from C; $C = C - \{\mathbf{p}_i\}$, $I = I - \{i\}$, $\alpha_i = 0$, $\lambda_i = 0$.
10: Update $\mathbf{x} = \mathbf{z}$ and the convex coefficients, λ, of $\mathbf{x}$ for C; solve $\mathbf{x} = \sum_{\mathbf{p}_i \in C} \lambda_i \mathbf{p}_i$ for λ.
11: Find the affine minimizer of C, $\mathbf{y} = \mathbf{argmin}_{\mathbf{y} \in \text{aff}(C)} \|\mathbf{y}\|_2$ and affine coefficients, α.
12: **end while**
13: Update $\mathbf{x} = \mathbf{y}$ and $\lambda = \alpha$.
14: **end while**
15: **Return** $\mathbf{x}$.
16: **end procedure**

The subset of points being considered as the *potential corral* is maintained in the set C. Iterations of the outer-loop, where points are added to C, are called *major cycles* and iterations of the inner-loop, where points are removed from C, are called *minor cycles*. The potential corral, C, is named so because at the beginning of a major cycle it is guaranteed to be a corral, while within the minor cycles it may or may not be a corral. Intuitively, a major cycle of Wolfe's method inserts an

improving point which violates Wolfe's criterion ($\mathbf{p}_j$ so that $\mathbf{x}^T\mathbf{p}_j < \|\mathbf{x}\|_2^2$) into C, then the minor cycles remove points until C is a corral, and this process is repeated until no points are improving and C is guaranteed to be a corral containing the minimizer. As the minor cycles remove points from C, the method checks whether C is a corral by computing the affine minimizer and checking to see if it lies in the relative interior of the convex hull of C. Computing the affine minimizer (lines 5 and 11 of Method 3.9) only requires solving a system of linear equations (see Chapter 4, Lemma 4.1.1).

It can be shown that this method terminates because the norm of the convex minimizer of the corrals visited monotonically decreases and thus, no corral is visited twice [**Wol76**]. Like [**CJK14**], we sketch the argument in [**Wol76**]. One may see that the norm monotonically decreases by noting that the convex minimizer over the polytope may result from one of two updates to $\mathbf{x}$, either at the end of a major cycle or at the end of a minor cycle. Let C be the corral at the beginning of a major cycle (line 3 of Method 3.9) and let $\mathbf{x}$ be the current minimizer, then the affine minimizer $\mathbf{y}$ has norm strictly less than that of $\mathbf{x}$ by Lemma 3.1.2, uniqueness of the affine minimizer and the fact that $\mathbf{p}_i^T\mathbf{x} < \|\mathbf{x}\|_2^2$ where $\mathbf{p}_i$ is the added point. Now, either $\mathbf{x}$ is updated to $\mathbf{y}$ or a minor cycle begins. Let S be the potential corral at the beginning of a minor cycle (line 6 of 3.9), let $\mathbf{x}$ be the current convex combination of points of S and let $\mathbf{y}$ be the affine minimizer of S. Note that $\mathbf{z}$ is a proper convex combination of $\mathbf{x}$ and $\mathbf{y}$ and since $\|\mathbf{y}\|_2 < \|\mathbf{x}\|_2$, we have $\|\mathbf{z}\|_2 < \|\mathbf{x}\|_2$. Thus, we see that every update of $\mathbf{x}$ decreases its norm. Note that the number of minor cycles within any major cycle is bounded by $n+1$, where n is the dimension of the space. Thus, the total number of iterations is bounded by the number of corrals visited multiplied by $n+1$. It is nevertheless not clear how the number of corrals grows, beyond the bound of $\sum_{i=1}^{n+1}\binom{m}{i}$.

Within the method, there are two moments at which one may choose which points to add to the potential corral. Observe that at line 2 of the pseudocode, one may choose which initial point to add to the potential corral. In this thesis we will only consider one *initial rule*, which is to initialize with the point of minimum norm. Observe that at line 4 of the pseudocode, there are several potential choices of which point to add to the potential corral. Two important examples of *insertion rules* are, first, the *minnorm rule* which dictates that one chooses, out of the improving points for the

potential corral, to add the point $\mathbf{p}_j$ of minimum norm. Second, the *linopt rule* dictates that one chooses, out of the improving points for the potential corral, to add the point $\mathbf{p}_j$ minimizing $\mathbf{x}^T\mathbf{p}_j$. Notice that insertion rules are to Wolfe's method what *pivot rules* are to the Simplex Method (see [**TZ93**] for a summary).

As with pivot rules, there are advantages and disadvantages of insertion rules. For example, the *minnorm* rule has the advantage that its implementation only requires an initial ordering of the points, then in each iteration it need only search for an improving point in order of increasing norm and add the first found. However, the *linopt* insertion rule has the advantage that, if the polytope is given in H-representation (intersection of halfspaces) rather than V-representation (convex hull of points), one may still perform Wolfe's method by using linear programming to find $\mathbf{p}_j$ minimizing $\mathbf{x}^T\mathbf{p}_j$ over the polytope. In other words, Wolfe's method does not need to have the list of vertices explicitly given, but suffices to have a linear programming oracle that provides the new vertex to be inserted. This feature of Wolfe's method means that each iteration can be implemented efficiently even for certain polyhedra having too many vertices and facets: specifically, over zonotopes (presented as a Minkowski sum of segments) [**FHI06**] and over the base polyhedron of a submodular function [**Fuj80**].

As we mentioned previously, the choice of *deletion* rule has far less impact on the behavior of Wolfe's algorithm. If there are multiple choices for points to remove from the potential corral, the *deletion* rule determines the order they leave, but they will all leave during the course of the same major cycle. Thus, *deletion* rules do not have an impact on the computational complexity of Wolfe's algorithm.

3.2.1. Examples. We begin by illustrating the behavior of Wolfe's method with a simple example. This example with the *linopt* insertion rule also appears in Wolfe's original paper [**Wol76**]. The example is the triangle formed by the three points in $\mathbb{R}^2$, $\mathbf{p}_1 = (0, 2)$, $\mathbf{p}_2 = (3, 0)$, and $\mathbf{p}_3 = (-2, 1)$. The table of major and minor cycles, the list of sets C, and points $\mathbf{x}$ and $\mathbf{y}$ are listed in Table 3.1 and visualizations of the steps are in Figures 3.4 and 3.5 for the *linopt* insertion rule. The table of major and minor cycles, the list of sets C, and points $\mathbf{x}$ and $\mathbf{y}$ are listed in Table 3.2 and visualizations of the steps are in Figures 3.6 and 3.7 for the *minnorm* insertion rule. In

Major Cycle	Minor Cycle	C	$\mathbf{x}$	$\mathbf{y}$
0	0	$\{\mathbf{p}_1\}$	$\mathbf{p}_1$	
1	0	$\{\mathbf{p}_1, \mathbf{p}_2\}$	$\mathbf{p}_1$	$(0.92, 1.38)$
2	0	$\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$	$(0.92, 1.38)$	$\mathbf{0}$
2	1	$\{\mathbf{p}_2, \mathbf{p}_3\}$	$(0.35, 0.53)$	$(0.12, 0.58)$

TABLE 3.1. iterations for *linopt* insertion rule on a triangle in $\mathbb{R}^2$

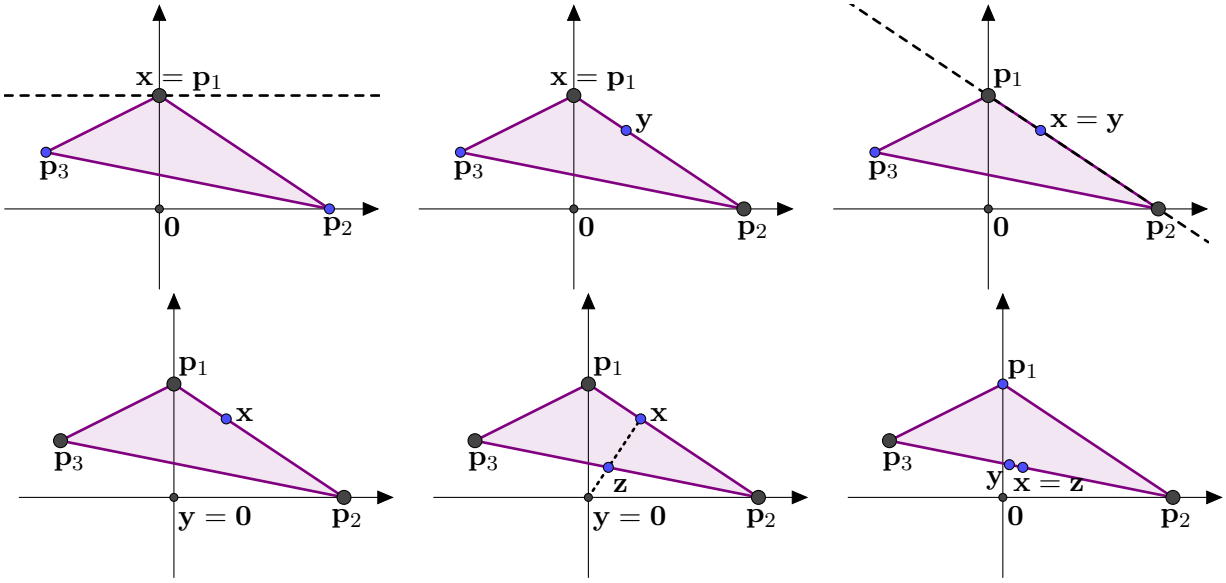

FIGURE 3.4. The first major and minor cycles of Wolfe's algorithm with the *linopt* insertion rule on Wolfe's example, a triangle in $\mathbb{R}^2$.

Major Cycle	Minor Cycle	C	$\mathbf{x}$	$\mathbf{y}$
0	0	$\{\mathbf{p}_1\}$	$\mathbf{p}_1$	
1	0	$\{\mathbf{p}_1, \mathbf{p}_3\}$	$\mathbf{p}_1$	$(-0.8, 1.6)$
2	0	$\{\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_2\}$	$(-0.8, 1.6)$	$\mathbf{0}$
2	1	$\{\mathbf{p}_3, \mathbf{p}_2\}$	$(-0.33, 0.67)$	$(0.12, 0.58)$

TABLE 3.2. iterations for *minnorm* insertion rule on a triangle in $\mathbb{R}^2$

Figures 3.4, 3.5, 3.6, and 3.7, the potential corrals of the iterations are represented by the bold, black vertices of the triangle.

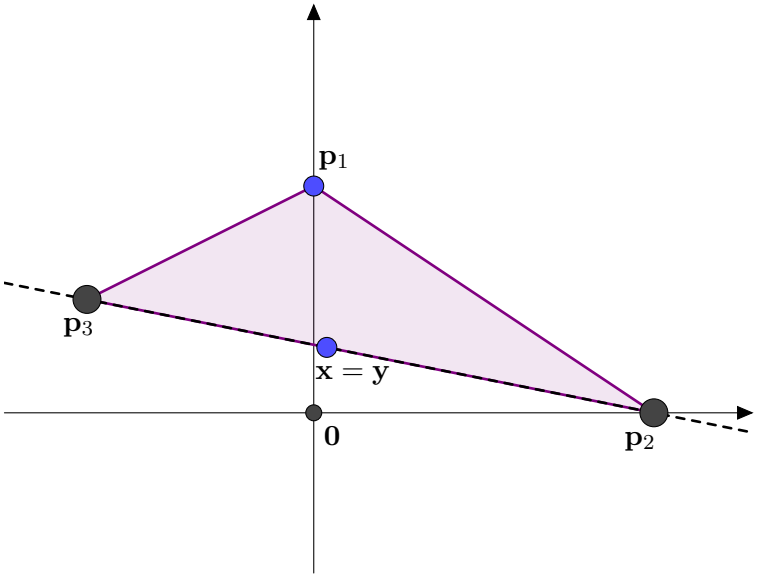

FIGURE 3.5. The last major cycle of Wolfe's algorithm with the *linopt* insertion rule on Wolfe's example, a triangle in $\mathbb{R}^2$. The point $\mathbf{x}$ is the minimum norm point and all points satisfy Wolfe's criterion.

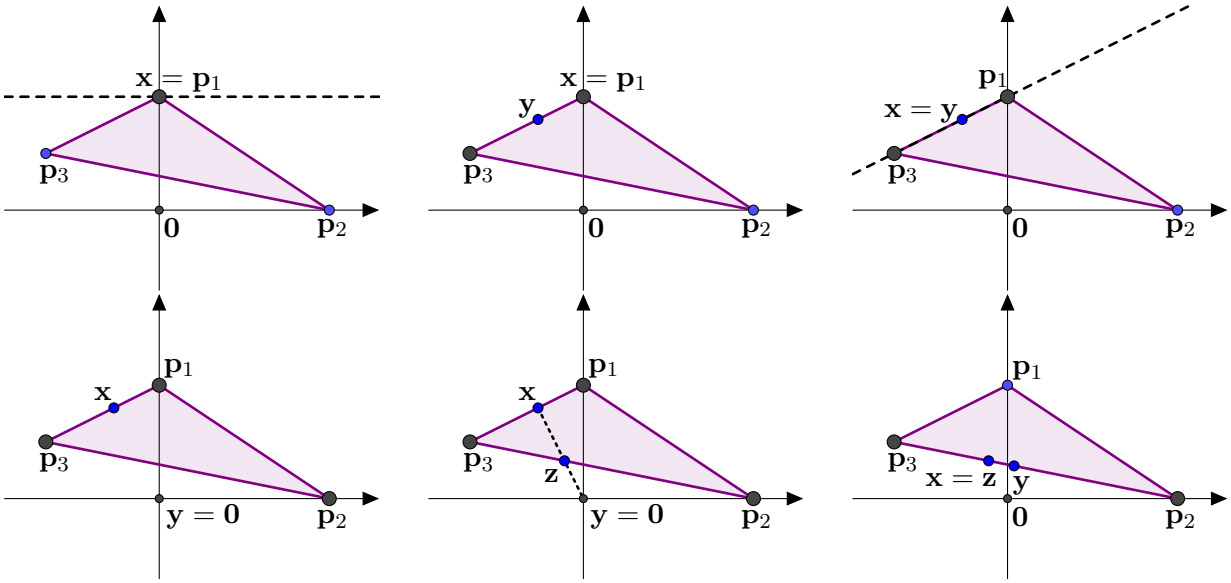

FIGURE 3.6. The first major and minor cycles of Wolfe's algorithm with the *minnorm* insertion rule on Wolfe's example, a triangle in $\mathbb{R}^2$.

We note additionally that there are examples on which Wolfe's algorithm with the *minnorm* insertion rule and the *linopt* insertion rule have the same behavior. In particular, the sequence of sets C, and points $\mathbf{x}$, and $\mathbf{y}$ are the same for these algorithms on the example in Figure 3.3.

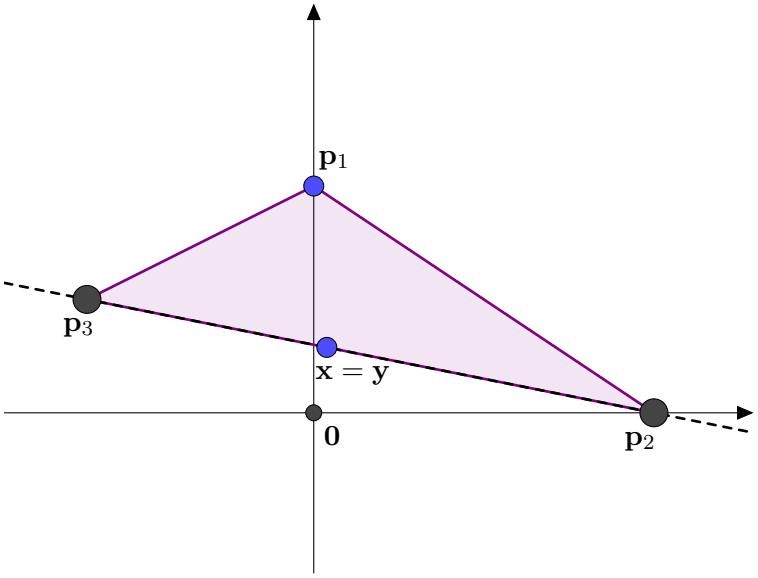

FIGURE 3.7. The last major cycle of Wolfe's algorithm with the *minnorm* insertion rule on Wolfe's example, a triangle in $\mathbb{R}^2$. The point $\mathbf{x}$ is the minimum norm point and all points satisfy Wolfe's criterion.

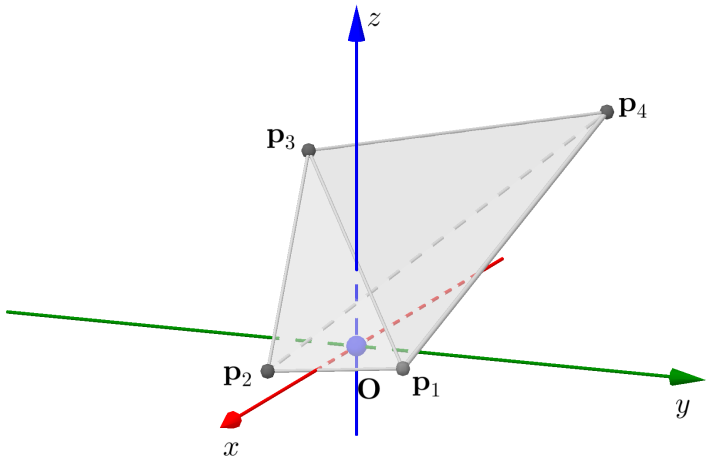

FIGURE 3.8. The simplex $P = \text{conv}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) \subset \mathbb{R}^3$ where $\mathbf{p}_1 = (0.8, 0.9, 0), \mathbf{p}_2 = (1.5, -0.5, 0), \mathbf{p}_3 = (-1, -1, 2)$ and $\mathbf{p}_4 = (-4, 1.5, 2)$.

We now present a simple example where the *minnorm* rule outperforms the *linopt* rule. That is, the *minnorm* insertion rule is not in obvious disadvantage to the *linopt* rule. This is in contrast to the family of examples we present in Section 3.4 where the *minnorm* rule takes exponential time, while we expect the *linopt* rule to take polynomial time.

Consider the simplex P shown in Figure 3.8 (we present the coordinates of vertices in the figure's caption). We list the steps of Wolfe's method on P for the *minnorm* and *linopt* insertion rules in Tables 3.3 and 3.4 and demonstrate a single step from each set of iterations in Figure 3.9. Each row lists major cycle and minor cycle iteration number, the vertices in the potential corral, and the value of $\mathbf{x}$ and $\mathbf{y}$ at the end of the iteration (before $\mathbf{x} = \mathbf{y}$ for major cycles). Note that the vertex $\mathbf{p}_4$ is added to the potential corral twice with the *linopt* insertion rule, as evidenced in Table 3.4.

Major Cycle	Minor Cycle	C	$\mathbf{x}$	$\mathbf{y}$
0	0	$\{\mathbf{p}_1\}$	$\mathbf{p}_1$	
1	0	$\{\mathbf{p}_1, \mathbf{p}_2\}$	$\mathbf{p}_1$	$(1, 0.5, 0)$
2	0	$\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$	$(1, 0.5, 0)$	$(0.3980, 0.199, 0.5473)$
3	0	$\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$	$(0.3980, 0.199, 0.5473)$	$(0, 0, 0)$
3	1	$\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4\}$	$(0.2878, 0.1439, 0.3957)$	$(0.1980, 0.0990, 0.4455)$

TABLE 3.3. iterations for *minnorm* insertion rule on simplex P

Major Cycle	Minor Cycle	C	$\mathbf{x}$	$\mathbf{y}$
0	0	$\{\mathbf{p}_1\}$	$\mathbf{p}_1$	
1	0	$\{\mathbf{p}_1, \mathbf{p}_4\}$	$\mathbf{p}_1$	$(0.2219, 0.9723, 0.2409)$
2	0	$\{\mathbf{p}_1, \mathbf{p}_4, \mathbf{p}_3\}$	$(0.2219, 0.9723, 0.2409)$	$(0.2848, 0.3417, 0.5810)$
2	1	$\{\mathbf{p}_1, \mathbf{p}_3\}$	$(0.2835, 0.3548, 0.5739)$	$(0.2774, 0.3484, 0.5807)$
3	0	$\{\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_2\}$	$(0.2774, 0.3484, 0.5807)$	$(0.3980, 0.199, 0.5473)$
4	0	$\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$	$(0.3980, 0.199, 0.5473)$	$(0, 0, 0)$
4	1	$\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4\}$	$(0.2878, 0.1439, 0.3957)$	$(0.1980, 0.0990, 0.4455)$

TABLE 3.4. iterations for *linopt* insertion rule on simplex P

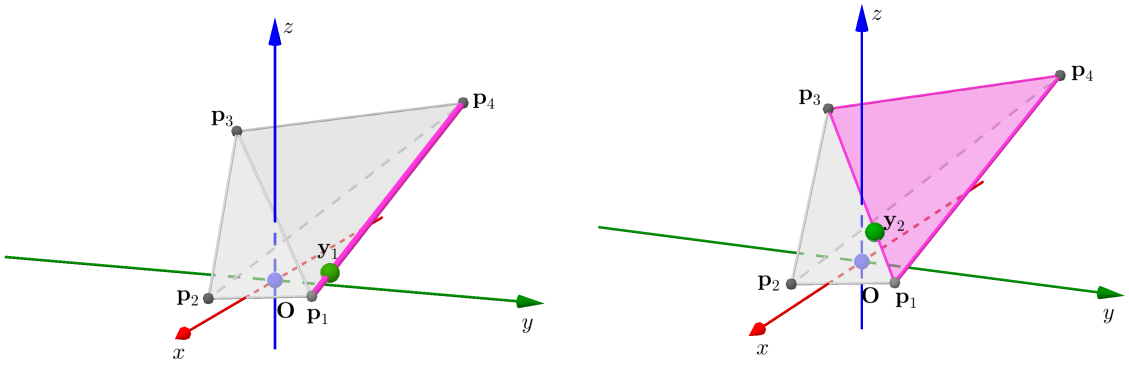

FIGURE 3.9. Left: Major cycle 1, minor cycle 0 for the *linopt* rule on P illustrates the end of a major cycle; the affine minimizer $\mathbf{y}_1 \in \text{relint}(\text{conv}(C)) = \text{relint}(\text{conv}(\mathbf{p}_1, \mathbf{p}_4))$. Right: Major cycle 2, minor cycle 0 for the *linopt* rule on P illustrates the beginning of a minor cycle; the affine minimizer $\mathbf{y}_2 \notin \text{relint}(\text{conv}(C)) = \text{relint}(\text{conv}(\mathbf{p}_1, \mathbf{p}_4, \mathbf{p}_3))$ and the vertex $\mathbf{p}_4$ will be removed in the next minor cycle.

3.3. Discussion of Related Results

Before presenting our example of Wolfe's exponential behavior, we discuss several previously known results and related methods. Wolfe proved that his method terminates in a finite number of steps [**Wol76**], giving an upper bound on the number of major cycles of $\sum_{i=1}^{n+1} \binom{m}{i}$ for $P = \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m) \subset \mathbb{R}^n$. Since then, several results have shown that Wolfe's method has pseudo-polynomial complexity. Before discussing these results, we describe several methods which are highly related to Wolfe's method.

First, we discuss von Neumann's algorithm [**Dan92**], which solves the feasibility problem: is $\mathbf{0} \in \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m)$? This is the problem `ZVPMD` defined in Chapter 4. The first steps of the method are very similar to Wolfe's method; it requires an initial point, $\mathbf{x}_0$, described as a convex combination of vertices of the polytope. It checks to see if the current point, $\mathbf{x}_k$, defines a separating hyperplane between the origin and the vertices of the polytope via Wolfe's criterion (Lemma 3.1.1). If not, it uses the *linopt* insertion rule to select a new vertex, $\mathbf{p}_{i_k}$. However, at this point in the algorithm, it parts from Wolfe's method. Using the newly selected vertex $\mathbf{p}_{i_k}$, von Neumann's algorithm computes the point of minimum norm along the line segment between $\mathbf{x}_k$ and $\mathbf{p}_{i_k}$, which is an easy one-dimensional problem. This convex minimizer (of $\overline{\mathbf{x}_k \mathbf{p}_{i_k}}$) becomes the new point

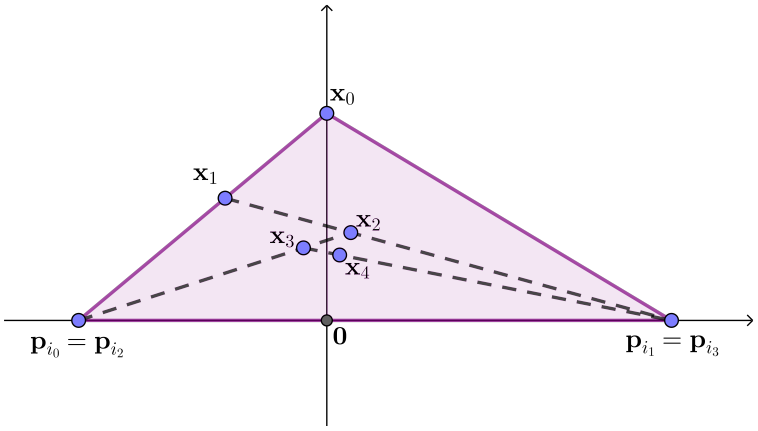

FIGURE 3.10. An illustration of the non-terminating behavior of von Neumann's algorithm for solving `ZVPMD`.

$\mathbf{x}_{k+1}$ and the algorithm repeats, checking to see if the current point forms a separating hyperplane, selecting a new vertex, and computing the convex minimizer.

One weakness of von Neumann's algorithm and a difference from Wolfe's method is that it is not a terminating algorithm. We illustrate the method and its non-terminating behavior in Figure 3.10. This behavior is known as zig-zagging. In [**EF00**], it was shown that if $\mathbf{0} \in \mathbf{relint}(\text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m))$, then von Neumann's algorithm converges linearly. In [**PnRS16**], the authors analyzed an extension of von Neumann's algorithm which allows for *away steps* when the progress of von Neumann's algorithm slows. They showed that this extension converges linearly when $\mathbf{0} \in \text{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m)$ and provides a certificate of infeasibility otherwise. They extended this line of thinking to demonstrate that the Frank-Wolfe method, similarly extended to allow for *away steps*, converges linearly [**PnRS16**].

We next discuss the Frank-Wolfe method for minimizing a convex function over a convex region (often a polytope). The classical Frank-Wolfe algorithm [**FW56**], solves a quadratic program over a polytope. The algorithm operates nearly identically to von Neumann's algorithm. It was quickly generalized to convex programs where linearization of the problem is easily computed, such as when the objective function, f is differentiable. At each iteration, the current iterate, $\mathbf{x}_k$, is the best seen (minimal objective value) point of the feasible polytope. The method proceeds by computing the vertex of the feasible polytope which minimizes the inner product with the gradient of the function, $\nabla f(\mathbf{x}_k)^T \mathbf{p}_{i_k}$. It then computes the convex minimizer of the objective function along the

line segment between $\mathbf{x}_k$ and $\mathbf{p}_{i_k}$, which is an easy one-dimensional problem. This convex minimizer becomes $\mathbf{x}_{k+1}$ and the algorithm proceeds.

Note that when the objective function is $\|\cdot\|^2$, this method specializes to the von Neumann algorithm. For this reason, it is known to be non-terminating and to converge with objective function decreasing at a rate of $\mathcal{O}(1/t)$. It was previously shown that when the minimizer lies in the relative interior of the feasible polytope, then the algorithm converges linearly [**BT04**] and an affine invariant convergence analysis was provided in [**LJJ13**]. Meanwhile, [**PnRS16**] showed that the Frank-Wolfe method with *away steps* converges with objective function decreasing linearly if the minimizer lies in the feasible polytope (even on the boundary), while [**LJJ13**] showed that this variant converges linearly without any dependency upon the location of the minimizer. In [**LJJ15**], the authors showed the linear convergence of several generalizations of the Frank-Wolfe method, including Wolfe's algorithm with the *linopt* insertion rule. They showed linear convergence of the objective function value for fully-corrective Frank-Wolfe, Frank-Wolfe with *away steps*, pairwise Frank-Wolfe, and Wolfe's algorithm for the minimum norm point problem with the *linopt* insertion rule.

Wolfe's method is additionally highly similar to Gilbert's procedure for computing the minimum of a quadratic form on a convex set [**Gil66**]. Gilbert's method iterates highly-similarly to the Frank-Wolfe method; however, the iterate $\mathbf{x}_{k+1}$, while guaranteed to lie along the line segment between $\mathbf{x}_k$ and the point selected via the *linopt* selection rule for $\mathbf{x}_k$, is not necessarily the convex minimizer of the quadratic form along this line segment. Wolfe's algorithm with the *linopt* selection rule has additionally been studied as the non-negative least-squares procedure of Hanson and Lawson [**LH95**, Section 23.3]. The difference between Wolfe's method (or the non-negative least-squares procedure) and the other methods discussed in this section is in the use of the vertex selected via the *linopt* selection rule. The algorithms of von Neumann, Gilbert, and classical Frank-Wolfe produce iterates which lie along the line segment between the previous iterate and the selected vertex. There are variants of the Frank-Wolfe method which produce iterates which are the convex minimizer of the current active-set (larger than the line segment). In comparison, Wolfe's method (or the non-negative least-squares procedure) computes the affine minimizer of the current active set

and uses a line search to return the iterate to the convex hull of the active set. In particular, this ensures that the active set is no larger than $n+1$ where n is the dimension of the input points.

In [**CJK14**], the authors analyzed Wolfe's method in order to give a complexity analysis of the Fujishige-Wolfe algorithm for submodular function minimization. They showed that for any polytope, P, if $M = \max_{p \in P} \|p\|$, then in $\mathcal{O}(nM^2/\epsilon)$ iterations Wolfe's method returns a point $\mathbf{x}$ with $\|\mathbf{x}\|^2 \leq \|\mathbf{x}_*\|^2 + \epsilon$ where $\mathbf{x}_*$ is the point of minimum norm in P. For problems given by rational data, this is a pseudo-polynomial bound for Wolfe's method, since the required number of iterations depends upon the magnitude of the largest integer in the input. A polynomial bound may only depend upon the encoding length of the integers in the input data, not the magnitude of these numbers. Note that M is at least as large as the largest integer appearing in any input point $\mathbf{p}_i$. Furthermore, the presence of $1/\epsilon$ additionally gives a dependence upon the magnitude of the integers in the input. Concretely, in order to leverage rationality of the solution (Chapter 4, Lemma 4.1.1) and round an approximation to the exact rational solution, one would need a $2^{-\sigma_P}$-approximate solution. Using the previous bound, this would require $\mathcal{O}(nM^2 2^{\sigma_P})$ iterations, which is a polynomial in the magnitude of the largest integers in any $\mathbf{p}_i$, not their binary size.

In [**LJJ15**], the authors proved that Wolfe's algorithm with the *linopt* insertion rule converges with objective function decreasing linearly. Specifically, they define $0 < \rho < 1$ to be an *eccentricity* parameter of P and show that $\|\mathbf{x}_k\|^2 - \|\mathbf{x}_*\|^2 \leq (1-\rho)^{k/2}(\|\mathbf{x}_0\|^2 - \|\mathbf{x}_*\|^2)$. First, note that if we have knowledge of ρ, this gives us a good stopping criterion for using Wolfe's method to approximate the minimum norm point solution. For $\mathbf{x}_k \in P$ we have $\mathbf{x}^T\mathbf{x}_* \geq \|\mathbf{x}_*\|^2$ by Wolfe's criterion (Lemma 3.1.1), so $\|\mathbf{x}_k - \mathbf{x}_*\|^2 = \|\mathbf{x}_k\|^2 - 2\mathbf{x}_k^T\mathbf{x}_* + \|\mathbf{x}_*\|^2 \leq \|\mathbf{x}_k\|^2 - \|\mathbf{x}_*\|^2 \leq (1-\rho)^{k/2}(\|\mathbf{x}_0\|^2 - \|\mathbf{x}_*\|^2) \leq (1-\rho)^{k/2}\|\mathbf{x}_0\|^2$. Thus, for an ϵ-approximate solution, one need only run $k \geq 2^{\frac{-\log(1/\epsilon) - \log(\|\mathbf{x}_0\|^2)}{\log(1-\rho)}}$ iterations of Wolfe's method (possibly stopping before the natural termination of the algorithm). However, we point out that this does not give a polynomial bound on Wolfe's method. While this dependence upon ϵ, $\log(1/\epsilon)$, contributes to the lower bound polynomially in the encoding length of the problem, the dependence upon ρ does not. Note that $-1/\log(1-\rho) > 1/2\rho$ for $0 < \rho < \frac{1}{2}$, and ρ may be exponentially small in the encoding length of the problem. In fact, for Wolfe's method, $\rho = c(\delta/M)^2$ where δ is the pyramidal width of P and M is the diameter of P. We point out

that the simplex S formed as the convex combination of $(0,0,0),(0,0,1),(1/2^k,0,1),(0,1/2^k,1)$ has diameter $1 < M < 2$, while the pyramidal width is bounded above by the width of the face formed by $(0,0,1),(1/2^k,0,1),(0,1/2^k,1)$; thus, $\delta \leq 1/2^k$. As the encoding length of this polytope is polynomial in k, we have that δ/M is not polynomial in the encoding length.

3.4. Example of Exponential Behavior

To understand our hard instance, it is helpful to consider first a simple instance that shows an inefficiency of Wolfe's method. The example is a set of points where a point leaves and reenters the current corral: four points in $\mathbb{R}^3$, $(1,0,0),(1/2,1/4,1),(1/2,1/4,-1),(-2,1/4,0)$. If one labels the points $1,2,3,4$, the sequence of corrals with the *minnorm* rule is $1, 12, 23, 234, 14$, where point 1 enters, leaves and reenters (For succintness, sets of points like $\{a,b,c\}$ may be denoted abc.). The idea now is to recursively replace point 1 (that reenters) in this construction by a recursively constructed set of points whose corrals are then considered twice by Wolfe's method. To simplify the proof, our construction uses a variation of this set of 4 points with an additional point and modified coordinates. This modified construction is depicted in Fig. 3.11, where point 1 corresponds to a set of points $P(d-2)$, points 2,3 correspond to points $\mathbf{p}_d, \mathbf{q}_d$ and point 4 corresponds to points $\mathbf{r}_d, \mathbf{s}_d$.

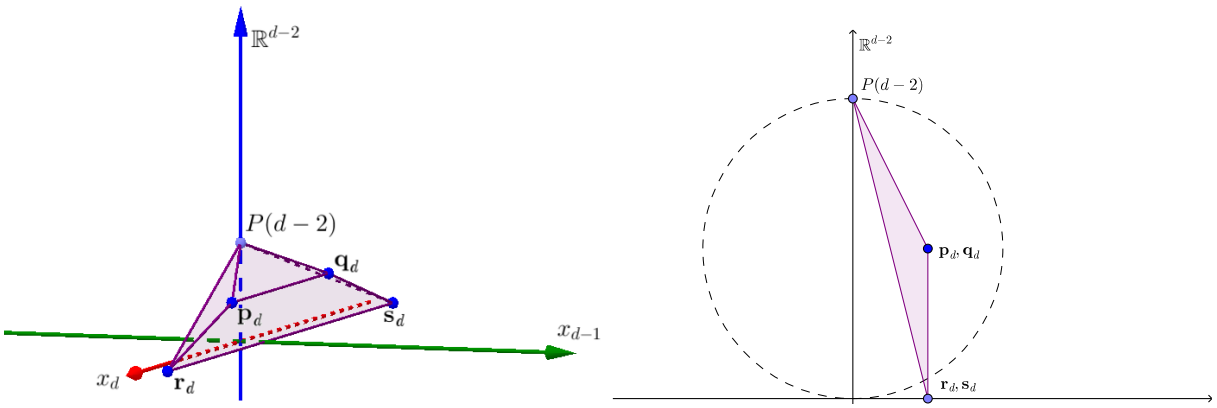

FIGURE 3.11. Left: In this view of $P(d)$, the point labeled $P(d-2)$ represents all points from $P(d-2)$ embedded into $\mathbb{R}^d$. The axis labeled $\mathbb{R}^{d-2}$ represents the $(d-2)$-dimensional subspace, span $(P(d-2))$ projected into span $\left(\mathbf{o}^*_{d-2}\right)$. Right: A two-dimensional view of $P(d)$ projected along the x_d coordinate axis.

The high-level idea of our exponential lower bound example is the following. We will inductively define a sequence of instances of increasing dimension of the minimum norm point problem. Given an instance in dimension $d-2$, we will add a few dimensions and points so that, when given to Wolfe's method, the number of corrals of the new augmented instance in dimension d has about twice the number of corrals of the input instance in dimension $d-2$. More precisely, our augmentation procedure takes an instance $P(d-2)$ in $\mathbb{R}^{d-2}$, adds two new coordinates and adds four points, $\mathbf{p}_d, \mathbf{q}_d, \mathbf{r}_d, \mathbf{s}_d$, to get an instance $P(d)$ in $\mathbb{R}^d$.

Points $\mathbf{p}_d, \mathbf{q}_d$ are defined so that the method on instance $P(d)$ goes first through every corral given by the points in the prior configuration $P(d-2)$ and then goes to corral $\mathbf{p}_d\mathbf{q}_d$. To achieve this under the minimum norm rule, the four new points have greater norm than any point in $P(d-2)$ and they are in the geometric configuration sketched in Fig. 3.11.

At this time, no point in $P(d-2)$ is in the current corral or available to add; see the left of Figure 3.12 for a visualization. If a point in $P(d-2)$ is part of the optimal corral, it will have to reenter, which is expensive. Points $\mathbf{r}_d, \mathbf{s}_d$ are defined so that $\mathbf{r}_d\mathbf{s}_d$ is a corral after $\mathbf{p}_d\mathbf{q}_d$, but now every point in $P(d-2)$ is improving according to Wolfe's criterion and may enter again; see the right of Figure 3.12. Specifically, every corral in $P(d-2)$, with $\mathbf{r}_d\mathbf{s}_d$ appended, is visited again.

Before proving this behavior for the example in dimension d, we list the iterations, major and minor cycle counts, and the points $\mathbf{x}$ and $\mathbf{y}$ for the example in dimension three. Note that as $P(1) = \{1\}$, we have $C(1) = 1$. Thus, we expect the sequence of corrals for $P(3)$ with the *minnorm* insertion rule to be $(1,0,0), (1,0,0)\mathbf{p}_3, \mathbf{p}_3\mathbf{q}_3, \mathbf{q}_3\mathbf{r}_3, \mathbf{r}_3\mathbf{s}_3, \mathbf{r}_3\mathbf{s}_3(1,0,0)$. Table 3.5 below demonstrates that this is the case. Note that the corrals correspond to (major cycle, minor cycle): $(0,0), (1,0), (2,1), (3,1), (4,1), (5,0)$. For comparison, we include the iterations on $P(3)$ with the *linopt* insertion rule in Table 3.6.

3.4.1. Preliminary Lemmas. Before we start describing the exponential example in detail, we wish to review some preliminary lemmas of independent interest which will be used in the arguments. The first lemma demonstrates that orthogonality between finite point sets allows us to easily describe the affine minimizer of their union. Figure 3.13 shows two such situations, one in which the affine hull of the union of the point sets span all of $\mathbb{R}^3$ and one in which it does not.

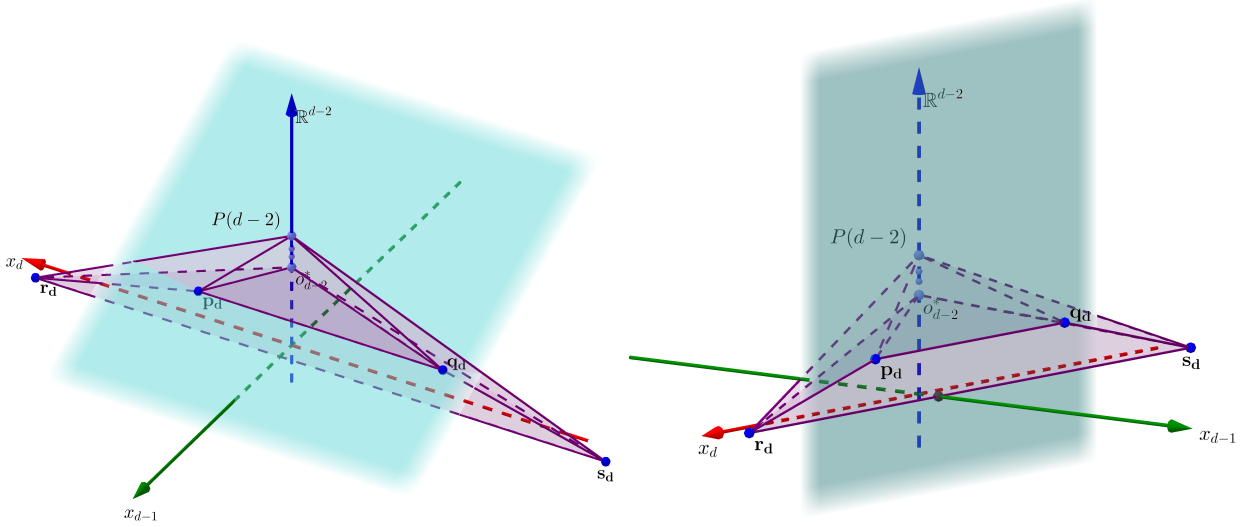

FIGURE 3.12. Left: $\mathbf{p}_d\mathbf{q}_d$ is a corral and none of $P(d-2)$ is available to add. Right: $\mathbf{r}_d\mathbf{s}_d$ is a corral and all of $P(d-2)$ is available to add. The planes represent the hyperplanes defined by Wolfe's criterion. The points on the 'away' side of the plane are available to add to the potential corral.

Major Cycle	Minor Cycle	C	$\mathbf{x}$	$\mathbf{y}$
0	0	$\{(1,0,0)\}$	$(1,0,0)$	
1	0	$\{(1,0,0), \mathbf{p}_3\}$	$(1,0,0)$	$(0.810, 0.095, 0.381)$
2	0	$\{(1,0,0), \mathbf{p}_3, \mathbf{q}_3\}$	$(0.810, 0.095, 0.381)$	$(0.2, 0.4, 0)$
2	1	$\{\mathbf{p}_3, \mathbf{q}_3\}$	$(0.5, 0.25, 0.1875)$	$(0.5, 0.25, 0)$
3	0	$\{\mathbf{p}_3, \mathbf{q}_3, \mathbf{r}_3\}$	$(0.5, 0.25, 0)$	$(0, 0.25, 0)$
3	1	$\{\mathbf{q}_3, \mathbf{r}_3\}$	$(0.3, 0.25, 0)$	$(0.297, 0.25, 0.0297)$
4	0	$\{\mathbf{q}_3, \mathbf{r}_3, \mathbf{s}_3\}$	$(0.297, 0.25, 0.0297)$	$(0, 0.25, 0)$
4	1	$\{\mathbf{r}_3, \mathbf{s}_3\}$	$(0, 0.25, 0)$	$(0, 0.25, 0)$
5	0	$\{\mathbf{r}_3, \mathbf{s}_3, (1,0,0)\}$	$(0, 0.25, 0)$	$(0.059, 0.235, 0)$

TABLE 3.5. iterations for *minnorm* insertion rule on $P(3)$

Lemma 3.4.1. *Let $A \subseteq \mathbb{R}^d$ be a proper linear subspace. Let $P \subseteq A$ be a non-empty finite set. Let $Q \subseteq A^\perp$ be another non-empty finite set. Let $\mathbf{x}$ be the minimum norm point in* $\text{aff}(P)$. *Let $\mathbf{y}$ be the minimum norm point in* $\text{aff}(Q)$. *Let $\mathbf{z}$ be the minimum norm point in* $\text{aff}(P \cup Q)$. *We have:*

Major Cycle	Minor Cycle	C	$\mathbf{x}$	$\mathbf{y}$
0	0	$\{(1,0,0)\}$	$(1,0,0)$	
1	0	$\{(1,0,0), \mathbf{r}_3\}$	$(1,0,0)$	$(0.901, 0.025, 0.298)$
2	0	$\{(1,0,0), \mathbf{r}_3, \mathbf{s}_3\}$	$(0.901, 0.025, 0.298)$	$(0.059, 0.235, 0)$

TABLE 3.6. iterations for *linopt* insertion rule on $P(3)$

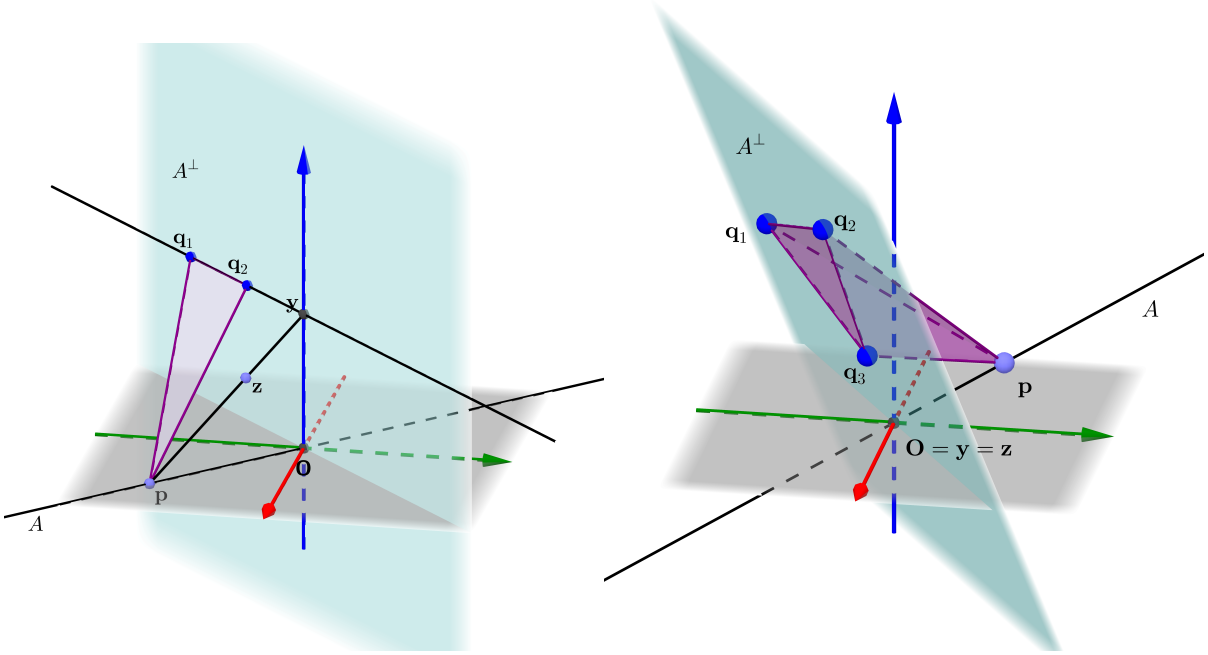

FIGURE 3.13. Examples of Lemma 3.4.1. Left: the affine hull of $P \cup Q$ is not full dimensional, and thus the affine minimizer lies at $\mathbf{z}$ along the line segment connecting $\mathbf{x} = \mathbf{p}$ and $\mathbf{y}$. Right: the convex hull of $P \cup Q$ is full-dimensional and thus the affine hull of $P \cup Q$ includes O which is the affine minimizer.

(1) $\mathbf{z}$ *is the minimum norm point in* $[\mathbf{x}, \mathbf{y}]$ *and therefore, if* $\mathbf{x} \neq \mathbf{0}$ *or* $\mathbf{y} \neq \mathbf{0}$*, then* $\mathbf{z} = \lambda\mathbf{x} + (1-\lambda)\mathbf{y}$ *with* $\lambda = \frac{\|\mathbf{y}\|_2^2}{\|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2}$.

(2) If $\mathbf{x} \neq \mathbf{0}$ *and* $\mathbf{y} \neq \mathbf{0}$*, then* $\mathbf{z}$ *is a strict convex combination of* $\mathbf{x}$ *and* $\mathbf{y}$.

(3) If $\mathbf{x} \neq \mathbf{0}$*,* $\mathbf{y} \neq \mathbf{0}$ *and* P *and* Q *are corrals, then* $P \cup Q$ *is also a corral.*

PROOF. If $\mathbf{x} = \mathbf{y} = \mathbf{0}$ then part 1 follows immediately. If at least one of $\mathbf{x}, \mathbf{y}$ is non-zero, then they are also distinct by the orthogonality assumption. Given two distinct points $\mathbf{a}, \mathbf{b}$, one can show

that the minimum norm point in the line through them is $\lambda\mathbf{a}+(1-\lambda)\mathbf{b}$ where $\lambda = \mathbf{b}^T(\mathbf{b}-\mathbf{a})/\|\mathbf{b}-\mathbf{a}\|_2^2$. For points $\mathbf{x}, \mathbf{y}$ as in the statement, the minimum norm point in $\text{aff}(\mathbf{x} \cup \mathbf{y})$ is $\mathbf{z}' = \lambda\mathbf{x} + (1-\lambda)\mathbf{y}$ with $\lambda = \frac{\|\mathbf{y}\|_2^2}{\|\mathbf{x}\|_2^2+\|\mathbf{y}\|_2^2} \in [0,1]$. Thus, $\mathbf{z}'$ is also the minimum norm point in $[\mathbf{x}, \mathbf{y}]$. We will now use the optimality condition in Lemma 3.1.2 to conclude that $\mathbf{z}' = \mathbf{z}$. Let $\mathbf{p} \in P$. Then $\mathbf{p}^T\mathbf{z}'$ can be computed in two steps: First project $\mathbf{p}$ onto $\text{span}\,(\mathbf{x}, \mathbf{y})$ (a subspace that contains $\mathbf{z}'$). This projection is $\mathbf{x}$ by optimality of $\mathbf{x}$. Then project onto $\mathbf{z}'$. This shows that $\mathbf{p}^T\mathbf{z}' = \mathbf{x}^T\mathbf{z}' = \|\mathbf{z}'\|_2^2$. A similar calculation shows $\mathbf{q}^T\mathbf{z}' = \|\mathbf{z}'\|_2^2$ for any $\mathbf{q} \in Q$. We conclude that $\mathbf{z}'$ is the minimum norm point in $\text{aff}(P \cup Q)$. This proves part 1.

Part 2 follows from our expression for λ above, which is in $(0,1)$ when $\mathbf{x} \neq \mathbf{0}$ and $\mathbf{y} \neq \mathbf{0}$.

Under the assumptions of part 3, we have that $\mathbf{x}$ is a strict convex combination of P and $\mathbf{y}$ is a strict convex combination of Q. This combined with the conclusion of part 2 gives that $\mathbf{z}$ is a strict convex combination of $P \cup Q$. The claim in part 3 follows. □

The following lemma shows conditions under which, if we have a corral and a new point that only has components along the minimum norm point of the corral and along new coordinates, then the corral with the new point added is also a corral. Moreover, the new minimum norm point is a convex combination of the old minimum norm point and the added point. Figure 3.14 gives an example of such a situation in $\mathbb{R}^3$. Denote by $\text{span}\,(M)$ the linear span of the set M.

Lemma 3.4.2. *Let $P \subseteq \mathbb{R}^d$ be a finite set of points that is a corral. Let $\mathbf{x}$ be the minimum norm point in $\text{aff}(P)$. Let $\mathbf{q} \in \text{span}\big(\mathbf{x}, \text{span}\,(P)^\perp\big)$, and assume $\mathbf{q}^T\mathbf{x} < \min\big\{\|\mathbf{q}\|_2^2, \|\mathbf{x}\|_2^2\big\}$. Then $P \cup \{\mathbf{q}\}$ is a corral. Moreover, the minimum norm point $\mathbf{y}$ in $\text{conv}(P \cup \{\mathbf{q}\})$ is a (strict) convex combination of $\mathbf{q}$ and the minimum norm point of P: $\mathbf{y} = \lambda\mathbf{x} + (1-\lambda)\mathbf{q}$ with $\lambda = \mathbf{q}^T(\mathbf{q}-\mathbf{x})/\|\mathbf{q}-\mathbf{x}\|_2^2$.*

Proof. Let $\mathbf{y}$ be the minimum norm point in $\text{aff}(P \cup \{\mathbf{q}\})$. Intuitively, $\mathbf{y}$ should be the minimum norm point in the line through $\mathbf{x}$ and $\mathbf{q}$. We will characterize $\mathbf{y}$ and show that it is a strict convex combination of $P \cup \{\mathbf{q}\}$ (which implies that it is a corral). Given two points $\mathbf{a}, \mathbf{b}$, one can show that the minimum norm point in the line through them is $\lambda\mathbf{a} + (1-\lambda)\mathbf{b}$ where $\lambda = \mathbf{b}^T(\mathbf{b}-\mathbf{a})/\|\mathbf{b}-\mathbf{a}\|_2^2$. Thus, we define $\mathbf{y} = \lambda\mathbf{x} + (1-\lambda)\mathbf{q}$ with $\lambda = \mathbf{q}^T(\mathbf{q}-\mathbf{x})/\|\mathbf{q}-\mathbf{x}\|_2^2$. By definition we have $\mathbf{y} \in \text{aff}(P \cup \{\mathbf{q}\})$.

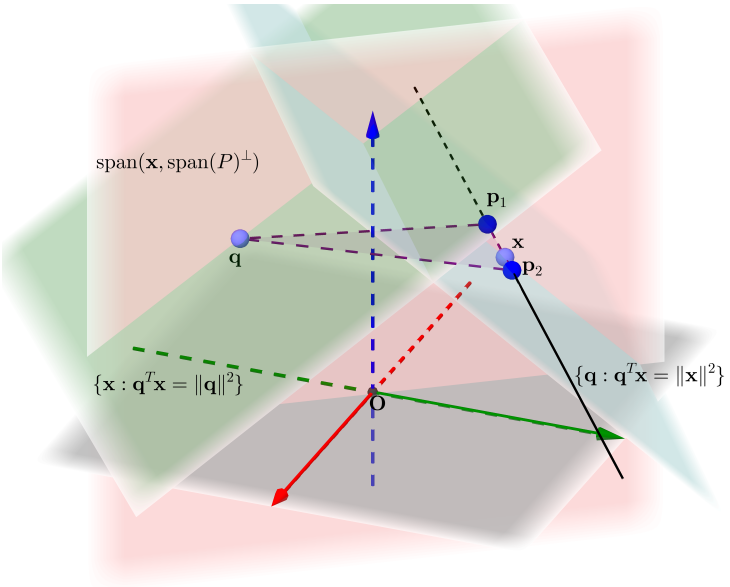

FIGURE 3.14. An example of Lemma 3.4.2 in which point $\mathbf{q}$ satisfies all assumptions and $P \cup \{\mathbf{q}\}$ is a corral. The hyperplanes are labeled with their defining properties and demonstrate that $\mathbf{q}^T\mathbf{x} < \min\{\|\mathbf{x}\|^2, \|\mathbf{q}\|^2\}$. The minimizer of $P \cup \{\mathbf{q}\}$ lies at the intersection of the blue, vertical axis and $\text{conv}(P \cup \{\mathbf{q}\})$.

The minimality of the norm of $\mathbf{y}$ follows from the optimality condition in Lemma 3.1.2. It holds by construction for $\mathbf{q}$. It also holds for $\mathbf{p} \in P$: The projection of $\mathbf{p}$ onto $\mathbf{y}$ can be computed in two steps. First, project onto span $(\mathbf{x}, \mathbf{q})$ (a subspace that contains $\mathbf{y}$), which is $\mathbf{x}$ by optimality of $\mathbf{x}$. Then project onto $\mathbf{y}$. This shows that $\mathbf{p}^T\mathbf{y} = \mathbf{x}^T\mathbf{y} = \|\mathbf{y}\|^2$ (the second equality by optimality of $\mathbf{y}$). We conclude that $\mathbf{y}$ is of minimum norm in $\text{aff}(P \cup \{\mathbf{q}\})$.

To conclude that $P \cup \{\mathbf{q}\}$ is a corral, we show that $\mathbf{y}$ is a strict convex combination of points $P \cup \{\mathbf{q}\}$. It is enough to show that $\mathbf{y}$ is a strict convex combination of $\mathbf{x}$ and $\mathbf{q}$. We have $\lambda = \mathbf{q}^T(\mathbf{q}-\mathbf{x})/\|\mathbf{q}-\mathbf{x}\|_2^2 = \frac{\|\mathbf{q}\|_2^2 - \mathbf{q}^T\mathbf{x}}{\|\mathbf{q}-\mathbf{x}\|_2^2} > 0$ by assumption. We also have $1 - \lambda = -\mathbf{x}^T(\mathbf{q}-\mathbf{x})/\|\mathbf{q}-\mathbf{x}\|_2^2 = \frac{\|\mathbf{x}\|_2^2 - \mathbf{q}^T\mathbf{x}}{\|\mathbf{q}-\mathbf{x}\|_2^2} > 0$ by assumption. □

Our last lemma shows that if we have points in two orthogonal subspaces, A and $A^\perp$, then adding a point from $A^\perp$ to a set from A does not cause any points from A that previously did not violate Wolfe's criterion to violate it. Figure 3.15 demonstrates this situation.

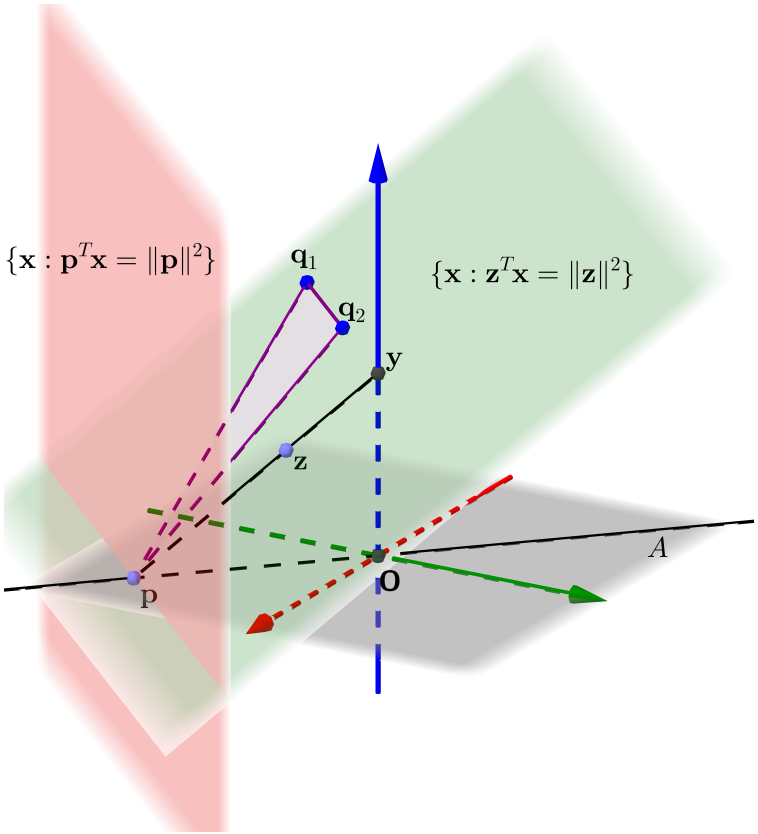

FIGURE 3.15. An example of Lemma 3.4.3 in which adding points Q from $A^\perp$ to points P from A create a new affine minimizer, $\mathbf{z}$, but the points satisfying Wolfe's criterion in A remain the same. Note that both hyperplanes intersect at the affine minimizer of P, so the halfspace intersections with A are the same.

Lemma 3.4.3. *For a point* $\mathbf{z}$ *define* $H_{\mathbf{z}} = \{\mathbf{w} \in \mathbb{R}^n : \mathbf{w}^T\mathbf{z} < \|\mathbf{z}\|_2^2\}$. *Suppose that we have an instance of the minimum norm point problem in* $\mathbb{R}^d$ *as follows: Some points,* P*, live in a proper linear subspace* A *and some,* Q*, in* $A^\perp$*. Let* $\mathbf{x}$ *be the minimum norm point in* $\text{aff}(P)$ *and* $\mathbf{y}$ *be the minimum norm point in* $\text{aff}(P \cup Q)$*. Then* $H_{\mathbf{y}} \cap A = H_{\mathbf{x}} \cap A$.

PROOF. Let B be the span of $\mathbf{x}$ and Q. We first show $\mathbf{y} \in B$. To see this, suppose not. Decompose $\mathbf{y}$ as $\mathbf{y} = \lambda\mathbf{v} + \sum_{\mathbf{q}\in Q} \mu_q\mathbf{q}$, where $\mathbf{v} \in \text{aff}(P)$ and $\lambda + \sum \mu_q = 1$. Decompose $\mathbf{v}$ as $\mathbf{v} = \mathbf{u} + \mathbf{x}$ where $\mathbf{u} \perp \mathbf{x}$ and $\mathbf{u} \in A$ (this is possible because $\mathbf{v} - \mathbf{x}$ is orthogonal to $\mathbf{x}$, by optimality of $\mathbf{x}$, Lemma 3.1.2). Thus, $\mathbf{y} = \lambda\mathbf{u} + \lambda\mathbf{x} + \sum_{\mathbf{q}\in Q} \mu_q\mathbf{q}$ with $\lambda\mathbf{u}$ orthogonal to $\lambda\mathbf{x} + \sum_{\mathbf{q}\in Q} \mu_q\mathbf{q}$. This implies that $\mathbf{y}' = \lambda\mathbf{x} + \sum_{\mathbf{q}\in Q} \mu_q\mathbf{q}$ has a smaller norm than $\mathbf{y}$ and $\mathbf{y}' \in \text{aff}(P \cup Q)$. This is a contradiction.

To conclude, we have $H_{\mathbf{y}} \cap A$ is a halfspace in A whose normal is parallel to the projection of $\mathbf{y}$ onto A (It is helpful to understand how to compute the intersection of a hyperplane with a subspace. If $T_{\mathbf{g}} = \{\mathbf{w} : \mathbf{w} \cdot \mathbf{g} = 1\}$ and S is a linear subspace, then $T_{\mathbf{g}} \cap S = \{\mathbf{w} \in S : \mathbf{w} \cdot \text{proj}_S \mathbf{g} = 1\}$. In other words, in order to intersect a hyperplane with a subspace we project the normal.). That is, it is parallel to $\mathbf{x}$. But that halfspace must also contain $\mathbf{x}$ on its boundary. Thus, that halfspace is equal to $H_{\mathbf{x}} \cap A$. □

3.4.2. Proof of Exponential Behavior. We will now describe our example in detail. The simplest version of our construction uses square roots and real numbers. We present instead a version with a few additional tweaks so that it only involves rational numbers.

Let $P(1) = \{1\} \subseteq \mathbb{Q}$. For odd $d > 1$, let $P(d)$ be a list of points in $\mathbb{Q}^d$ defined inductively as follows: Let $\mathbf{o}_d^*$ denote the minimum norm point in $\text{conv}(P(d))$. Let $M_d := \max_{\mathbf{p} \in P(d)} \|\mathbf{p}\|_1$, which is a rational upper bound to the maximum 2-norm among the points in $P(d)$. (For a first reading one can let M_d be the maximum 2-norm among points in $P(d)$, which leads to an essentially equivalent instance except that it is not rational.) Similarly, let $m_d = \|\mathbf{o}_d^*\|_\infty$, which is a rational lower bound to the minimum norm among points in $\text{conv}(P(d))$. (Again, for a first reading one can define $m_d = \|\mathbf{o}_d^*\|_2$ which leads to an essentially equivalent instance, except that it is not rational.)

We finally present the example. If we identify $P(d)$ with a matrix where the points are rows, then the points in $P(d)$ are given by the following block matrix:

$$P(d) = \begin{pmatrix} P(d-2) & 0 & 0 \\ \frac{1}{2}\mathbf{o}_{d-2}^* & \frac{m_{d-2}}{4} & M_{d-2} \\ \frac{1}{2}\mathbf{o}_{d-2}^* & \frac{m_{d-2}}{4} & -(M_{d-2}+1) \\ 0 & \frac{m_{d-2}}{4} & M_{d-2}+2 \\ 0 & \frac{m_{d-2}}{4} & -(M_{d-2}+3). \end{pmatrix}.$$

The last four rows of the matrix $P(d)$ are the points $\mathbf{p}_d, \mathbf{q}_d, \mathbf{r}_d, \mathbf{s}_d$ of the configuration. For a picture of the case of $P(3)$ see Figure 3.16.

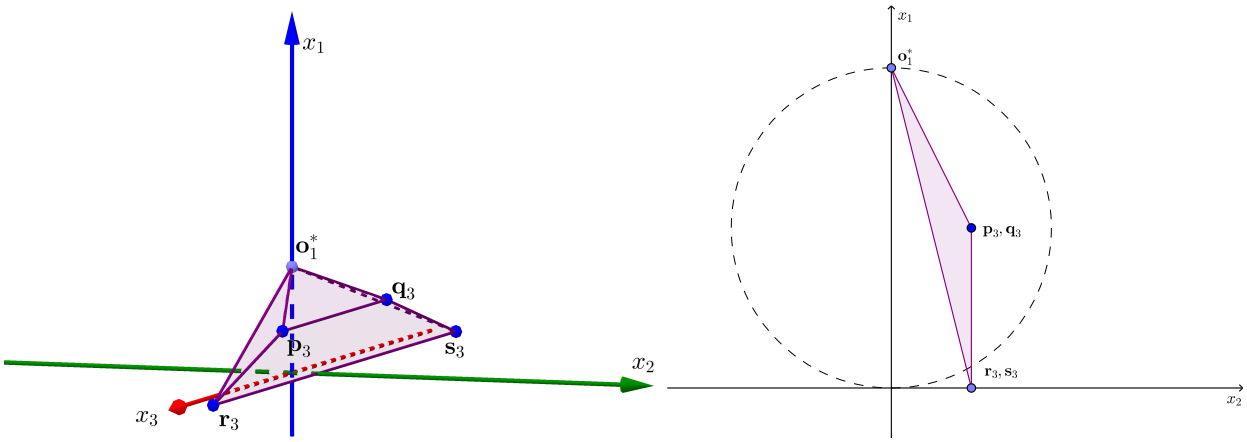

FIGURE 3.16. Left: Three-dimensional view of $P(3)$. Right: A two-dimensional view of $P(3)$ projected along the x_3 coordinate axis.

REMARK 3.4.1. *First note that strictly speaking $P(d-2) \subset \mathbb{Q}^{d-2}$, and that we are defining an embedding of it into $\mathbb{Q}^d$, for which we have to use a recursive process. To avoid unnecessary notation, we will abuse the notation. The point $\mathbf{v}_{d-2}$ denotes both a point of $P(d-2)$ and of the subsequent $P(d)$, i.e., $\mathbf{v}_{d-2} = (\mathbf{v}, 0, 0)$ will be the identical copy of $\mathbf{v}_{d-2}$ within $P(d)$, but we add two extra zero coordinates. Depending on the context $\mathbf{v}_{d-2}$ will be understood as both a $(d-2)$-dimensional vector or as a d-dimensional vector (e.g., when doing dot products). The points of $P(d-2)$ become a subset of the point configuration $P(d)$ by padding extra zeros. See Figures 3.11 and 3.17 which illustrate this embedding and address our visualizations of these sets in three dimensions.*

Theorem 3.4.4. *Consider the execution of Wolfe's algorithm (Method 3.9) with the* minnorm *point rule on input $P(d)$ where $d = 2k-1$. Then the sequence of corrals has length $5 \cdot 2^{k-1} - 4$.*

PROOF. Points in $P(d)$ are shown in order of increasing norm. Let $\mathbf{p}_d, \mathbf{q}_d, \mathbf{r}_d, \mathbf{s}_d$ denote the last four points of $P(d)$, respectively. Let $C(d)$ denote the ordered sequence of corrals in the execution of Wolfe's method on $P(d)$. Let $O(d)$ denote the last (optimal) corral in $C(d)$.

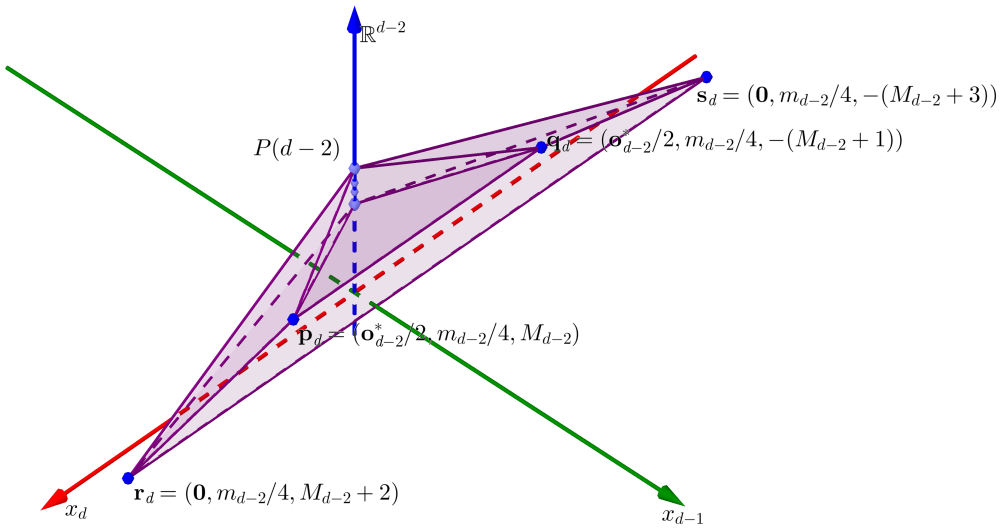

FIGURE 3.17. As described in Figure 3.11, the axis labeled $\mathbb{R}^{d-2}$ represents the $(d-2)$-dimensional subspace span $(P(d-2))$ projected onto the one dimensional subspace span $\left(\mathbf{o}^*_{d-2}\right)$. The projection of the set $P(d-2)$ forms a 'cloud' of points and the convex hull of this projection has many fewer faces than the unprojected convex hull. We visualize $P(d-2)$ and subsets as a single point in span $\left(\mathbf{o}^*_{d-2}\right)$.

The rest of the proof will establish that the sequence of corrals $C(d)$ is

$$C(d-2)$$

$$O(d-2)\mathbf{p}_d$$

$$\mathbf{p}_d\mathbf{q}_d$$

$$\mathbf{q}_d\mathbf{r}_d$$

$$\mathbf{r}_d\mathbf{s}_d$$

$$C(d-2)\mathbf{r}_d\mathbf{s}_d$$

(where a concatenation such as $C(d-2)\mathbf{r}_d\mathbf{s}_d$ denotes every corral in $C(d-2)$ with $\mathbf{r}_d$ and $\mathbf{s}_d$ added). After this sequence of corrals is established, we solve the resulting recurrence relation: Let $T(d)$ denote the length of $C(d)$. We have $T(1) = 1$, $T(d) = 2T(d-2)+4$. This implies $T(d) = 5 \cdot 2^{k-1} - 4$ (with $d = 2k-1$).

All we must show now to complete the proof of Theorem 3.4.4 is that $C(d)$ has indeed the stated recursive form. We do this by induction on d. The steps of the proof are written as claims with individual proofs which are concluded with $\diamondsuit$.

By construction, $C(d)$ starts with $C(d-2)$. This happens because points in $C(d)$ are ordered by increasing norm and the proof proceeds inductively as follows: The first corral in $C(d)$ is the minimum norm point in $P(d)$, which is also the first corral in $C(d-2)$. Suppose now that the first t corrals of $C(d)$ coincide with the first t corrals of $C(d-2)$. We will show that corral $t+1$ in $C(d)$ is the same as corral $t+1$ in $C(d-2)$. To see this, it is enough to see that the set of points in $P(d)$ that can enter (improving points) contains the point that enters in $C(d-2)$ (with two zeros appended) and contains no point of smaller norm. This two-part claim is true because the two new zero coordinates play no role in this and all points in $P(d)$ but not in $P(d-2)$ have a larger norm than any point in $P(d)$.

Once $O(d-2)$ is reached (with minimum norm point $\mathbf{o}^*_{d-2}$), the set of improving points, as established by Wolfe's criterion, consist of $\{\mathbf{p}_d, \mathbf{q}_d, \mathbf{r}_d, \mathbf{s}_d\}$. Now, because we are using the minimum-norm insertion rule, the next point to enter is $\mathbf{p}_d$.

Claim 1. *$O(d-2)\mathbf{p}_d$ is a corral.*

Proof of Claim. This is a special case of Lemma 3.4.2. We have $\mathbf{p}_d = (\mathbf{o}^*_{d-2}/2, m_{d-2}/4, M_{d-2})$. We just need to verify the two inequalities in Lemma 3.4.2:

$$(\mathbf{o}^*_{d-2})^T\mathbf{p}_d = \|\mathbf{o}^*_{d-2}\|_2^2/2 < \|\mathbf{o}^*_{d-2}\|_2^2 < \|\mathbf{p}_d\|_2^2.$$

$\diamondsuit$

Claim 2. *The next improving point to enter is $\mathbf{q}_d$.*

Proof of Claim. We first check that no point in $P(d-2)$ can enter. From Lemma 3.4.2 we know the optimal point $\mathbf{y}$ in corral $O(d-2)\mathbf{p}_d$ explicitly in terms of the optimal point $\mathbf{o}^*_{d-2}$ of $O(d-2)$ and $\mathbf{p}_d$, namely $\mathbf{y}$ is a convex combination $\lambda\mathbf{o}^*_{d-2} + (1-\lambda)\mathbf{p}_d$, with $\lambda = \frac{\|\mathbf{p}_d\|_2^2 - \mathbf{p}_d{}^T\mathbf{o}^*_{d-2}}{\|\mathbf{p}_d - \mathbf{o}^*_{d-2}\|_2^2}$. Let $\mathbf{p} \in P(d-2)$. We check that it cannot enter via Wolfe's criterion. We compute $\mathbf{p}^T\mathbf{y}$ in two

steps: First project $\mathbf{p}$ onto span $\left(\mathbf{o}^*_{d-2}, \mathbf{p}_d\right)$ (a subspace that contains $\mathbf{y}$). This projection is longer than $\mathbf{o}^*_{d-2}$ by optimality of $\mathbf{o}^*_{d-2}$. Then project onto $\mathbf{y}$. This shows that $\mathbf{p}^T\mathbf{y} \geq {\mathbf{o}^*_{d-2}}^T\mathbf{y} = \|\mathbf{y}\|_2^2$ and $\mathbf{p}$ cannot enter as it is not an improving point according to Wolfe's criterion.

By construction, $\mathbf{q}_d$ is closer to the origin than $\mathbf{r}_d, \mathbf{s}_d$, so to conclude it is enough to check that $\mathbf{q}_d$ is an improving point per Wolfe's criterion. Compute

$$\begin{aligned}
\mathbf{y}^T\mathbf{q}_d &= \lambda(\mathbf{o}^*_{d-2})^T\mathbf{q}_d + (1-\lambda)\mathbf{p}_d^T\mathbf{q}_d \\
&\leq \frac{\lambda}{2}\|\mathbf{o}^*_{d-2}\|_2^2 + (1-\lambda)\left[\frac{1}{4}\|\mathbf{o}^*_{d-2}\|_2^2 + \frac{1}{16}\|\mathbf{o}^*_{d-2}\|_2^2 - M_{d-2}^2 - M_{d-2}\right] \\
&\leq \frac{\lambda}{2}\|\mathbf{o}^*_{d-2}\|_2^2
\end{aligned}$$

since by construction $M_{d-2} \geq 1$ and $\left\|\mathbf{o}^*_{d-2}\right\|_2 \leq 1$. On the other hand,

$$\begin{aligned}
\|\mathbf{y}\|_2^2 &= \lambda^2\|\mathbf{o}^*_{d-2}\|_2^2 + (1-\lambda)^2\|\mathbf{p}_d\|_2^2 + 2\lambda(1-\lambda)\frac{1}{2}\|\mathbf{o}^*_{d-2}\|_2^2 \\
&= \lambda\|\mathbf{o}^*_{d-2}\|_2^2 + (1-\lambda)^2\|\mathbf{p}_d\|_2^2 \\
&\geq \lambda\|\mathbf{o}^*_{d-2}\|_2^2.
\end{aligned}$$

Thus, $\mathbf{y}^T\mathbf{q}_d < \|\mathbf{y}\|^2$, that is, $\mathbf{q}_d$ is an improving point. ♢

Claim 3. *The current set of points, $O(d-2) \cup \{\mathbf{p}_d, \mathbf{q}_d\}$, is not a corral. Points in $O(d-2)$ leave one by one. The next corral is $\mathbf{p}_d\mathbf{q}_d$.*

Proof of Claim. Instead of analyzing the iterations of Wolfe's inner loop, we use the key fact, from Section 3.2, that the inner loop must end with a corral whose distance to the origin is strictly less than the previous corral. We look at the alternatives: This new corral cannot be $O(d-2) \cup \{\mathbf{p}_d\}$ (the previous corral) or any subset of it because it would not decrease the distance. An analysis similar to that of Claim 1 or basic trigonometry (in three-dimensions) shows that $O(d-2) \cup \{\mathbf{q}_d\}$ is a corral whose distance to the origin is larger than the distance for $O(d-2) \cup \{\mathbf{p}_d\}$. See Fig. 3.18, where we show a projection, the perpendicular line segments to conv$(O(d-2), \mathbf{p}_d)$

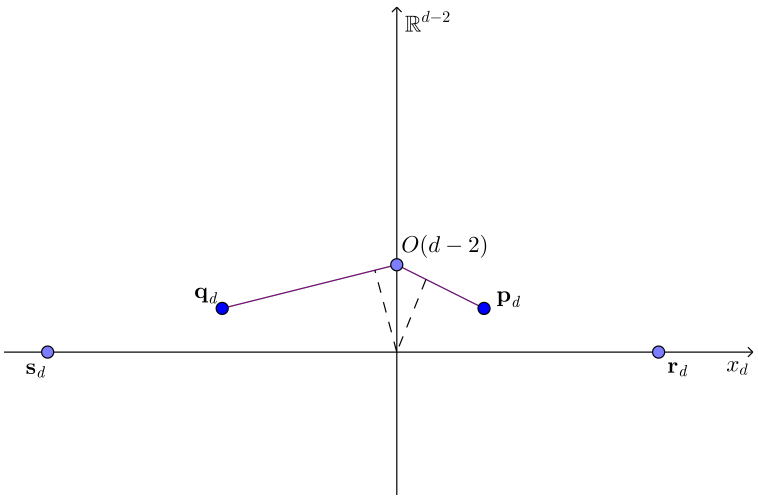

FIGURE 3.18. A projection of the point set in the direction of x_{d-1}. Any corral of the form $S\mathbf{q}_d$ where $S \subset O(d-2)$ would have distance larger than the previous corral, $O(d-2)\mathbf{p}_d$.

and $\text{conv}(O(d-2), \mathbf{q}_d)$ are shown in dotted line after projection. Thus, the new corral cannot be $O(d-2) \cup \{\mathbf{q}_d\}$ or any subset of it.

No set of the form $S \cup \{\mathbf{p}_d, \mathbf{q}_d\}$ with $S \subseteq O(d-2)$ and S non-empty can be a corral: To see this, first note that the minimum norm point in $\text{conv}(S \cup \{\mathbf{p}_d, \mathbf{q}_d\})$ is in the segment $[\mathbf{p}_d, \mathbf{q}_d]$, specifically, point $(\mathbf{o}^*_{d-2}/2, m_{d-2}/4, 0)$ (minimality follows from Wolfe's criterion, Lemma 3.1.1). This implies that the minimum norm point in $\text{aff}(S \cup \{\mathbf{p}_d, \mathbf{q}_d\})$ cannot be in the relative interior of $\text{conv}(S \cup \{\mathbf{p}_d, \mathbf{q}_d\})$ when S is non-empty (see Figure 3.19).

The only remaining non-empty subset is $\{\mathbf{p}_d, \mathbf{q}_d\}$, which is the new corral. ♢

CLAIM 4. *The set of improving points is now* $\{\mathbf{r}_d, \mathbf{s}_d\}$.

PROOF OF CLAIM. Recall that the optimal point in corral $\{\mathbf{p}_d, \mathbf{q}_d\}$ has coordinates $(\mathbf{o}^*_{d-2}/2, m_{d-2}/4, 0)$. Thus, when computing distances and checking Wolfe's criterion it is enough to do so in the two-dimensional situation depicted in Figure 3.20. A hyperplane orthogonal to the segment from the origin to $(\mathbf{o}^*_{d-2}/2, m_{d-2}/4, 0)$ is shown in Figure 3.20. It leaves the points in $P(d-2)$ above and both $\mathbf{r}_d$ and $\mathbf{s}_d$ below making them the only improving points. ♢

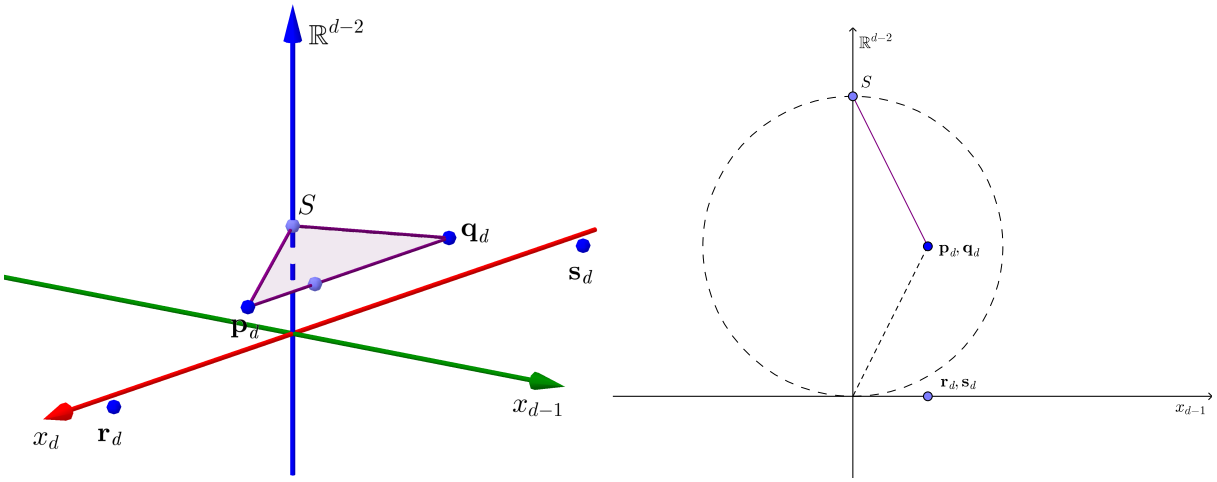

FIGURE 3.19. The minimum norm point in $\text{conv}(S \cup \{\mathbf{p}_d, \mathbf{q}_d\})$ is in the line segment between $\mathbf{p}_d$ and $\mathbf{q}_d$.

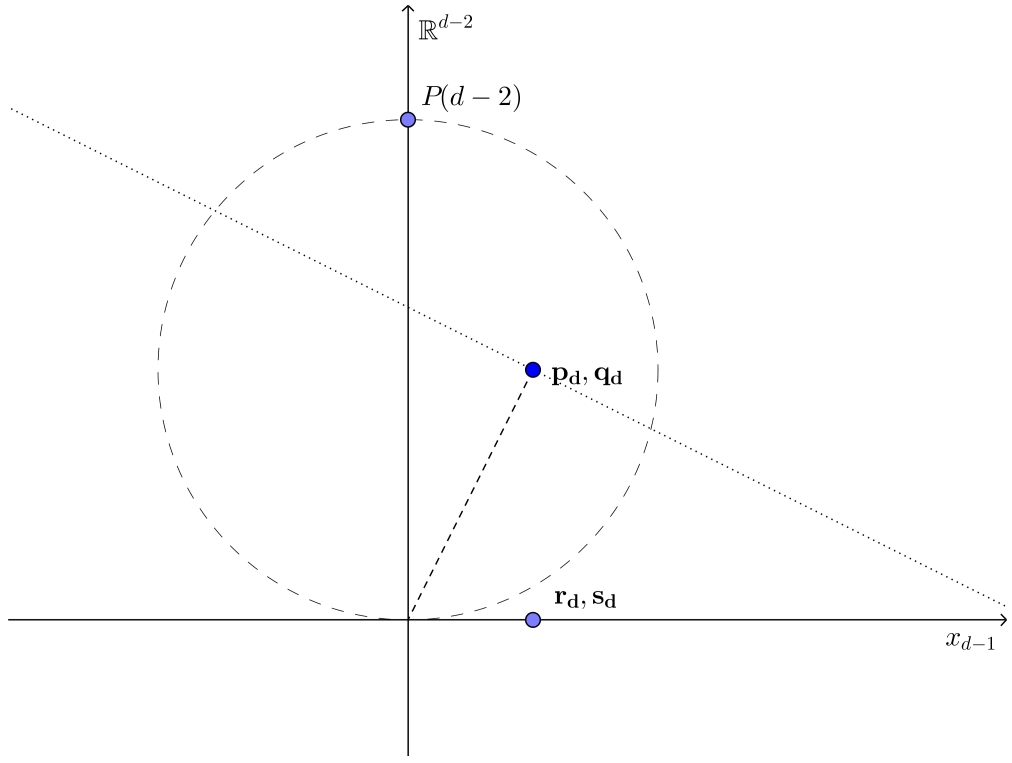

FIGURE 3.20. The set of improving points is now $\{\mathbf{r}_d, \mathbf{s}_d\}$.

Point $\mathbf{r}_d$ enters since it has smallest norm.

CLAIM 5. *Point $\mathbf{p}_d$ leaves and the next corral is $\mathbf{q}_d\mathbf{r}_d$.*

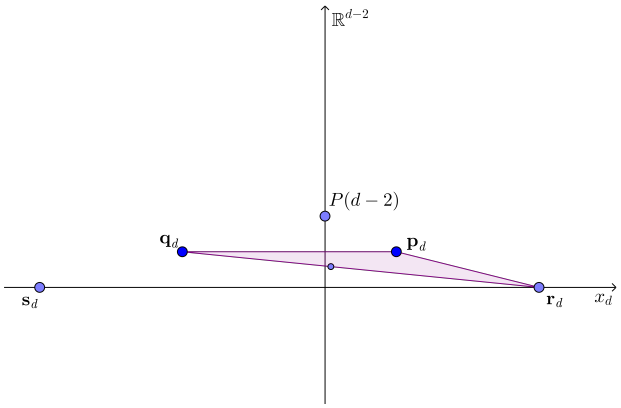

FIGURE 3.21. The set $\{\mathbf{p}_d, \mathbf{q}_d, \mathbf{r}_d\}$ is not a corral.

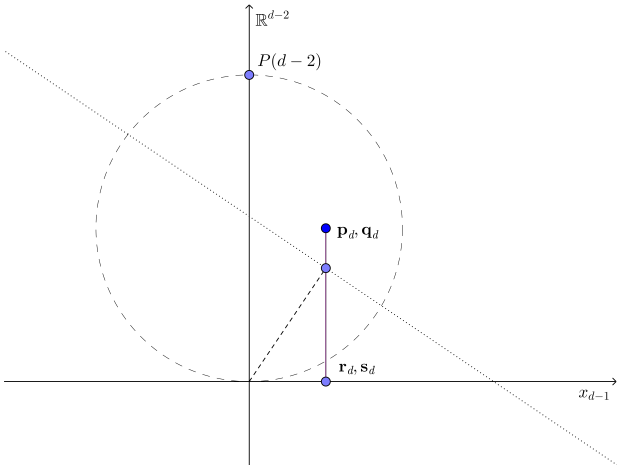

FIGURE 3.22. The only improving point is $\mathbf{s}_d$.

PROOF OF CLAIM. To start, notice that by construction the four points $\mathbf{p}_d, \mathbf{q}_d, \mathbf{r}_d, \mathbf{s}_d$ lie on a common hyperplane, L, parallel to the subspace spanned by $\mathbf{o}^*_{d-2}$. Thus, one does not need to do distance calculations but rather Fig. 3.21 is a faithful representation of the positions of points. The origin is facing the hyperplane L, parallel to span $\left(\mathbf{o}^*_{d-2}\right)$. The closest point to the origin within L is in the line segment joining $\mathbf{r}_d, \mathbf{s}_d$ thus, as we move vertically, the closest point to the origin in triangle $\mathbf{p}_d, \mathbf{q}_d, \mathbf{r}_d$ must be in the line segment joining $\mathbf{r}_d$ and $\mathbf{q}_d$. ♢

CLAIM 6. *The only improving point now is* $\mathbf{s}_d$.

PROOF OF CLAIM. Once more we rely in two different orthogonal two-dimensional projections of $P(d)$ to estimates distances and to check Wolfe's criterion. The line segment from the origin to the optimum of the corral $\mathbf{q}_d, \mathbf{r}_d$ (we could calculate this exactly, but it is not necessary), and its orthogonal hyperplane are shown in Figure 3.22. This shows only $\mathbf{r}_d$ or $\mathbf{s}_d$ are candidates but $\mathbf{r}_d$ is in the current corral, so only $\mathbf{s}_d$ may be added. ♢

Point $\mathbf{s}_d$ enters as the closest improving point to the origin.

CLAIM 7. *Point* $\mathbf{q}_d$ *leaves. The next corral is* $\mathbf{r}_d\mathbf{s}_d$.

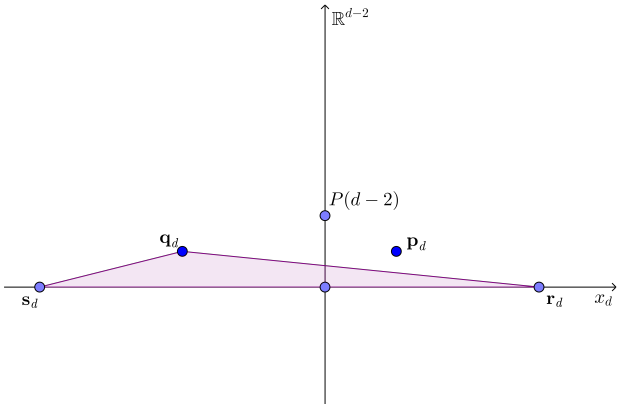

FIGURE 3.23. The point $\mathbf{q}_d$ leaves.

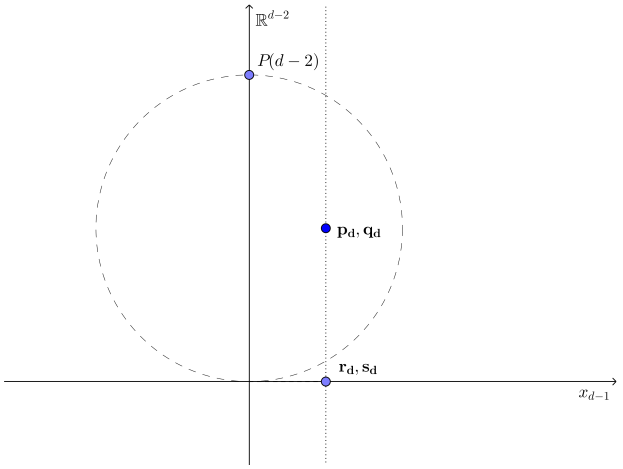

FIGURE 3.24. The improving points are $P(d-2)$.

PROOF OF CLAIM. We wish to find the closest point to the origin in triangle $\mathbf{q}_d, \mathbf{r}_d, \mathbf{s}_d$. From Figure 3.23 the optimum is between $\mathbf{r}_d, \mathbf{s}_d$; this point is $(m_{d-2}/4)\mathbf{e}_{d-1}$. Clearly this point is below $\mathbf{q}_d$, so it must leave the corral. ♢

CLAIM 8. *The set of improving points is now $P(d-2)$ (with two zero coordinates appended).*

PROOF OF CLAIM. Now Wolfe's criterion hyperplane contains the four points $\mathbf{p}_d, \mathbf{q}_d, \mathbf{r}_d, \mathbf{s}_d$ by construction leaving $P(d-2)$ on the same side as the origin (see Figure 3.24). ♢

The first (and minimum norm) point in $P(d)$ enters and the next corral is this point together with $\mathbf{r}_d$ and $\mathbf{s}_d$. That is, the next corral is precisely the first corral in $C(d-2)\mathbf{r}_d\mathbf{s}_d$. We will prove inductively that the sequence of corrals from now on is exactly all of $C(d-2)\mathbf{r}_d\mathbf{s}_d$. To see this, we repeatedly invoke Lemma 3.4.3 after every corral with A equal to the subspace spanned by the first $d-2$ coordinate vectors of $\mathbb{R}^d$. Suppose that the current corral is $C\mathbf{r}_d\mathbf{s}_d$, where C is one of the corrals in $C(d-2)$ and denote the next corral in $C(d-2)$ by C'. From Lemma 3.4.3, we get that the set of improving points for corral $C\mathbf{r}_d\mathbf{s}_d$ is obtained by taking the set of improving points for corral C and removing $\{\mathbf{p}_d, \mathbf{q}_d, \mathbf{r}_d, \mathbf{s}_d\}$. Thus, the point that enters is the same that would enter after corral C. Let $\mathbf{a}$ denote that point.

CLAIM 9. *The next corral is $C'\mathbf{r}_d\mathbf{s}_d$.*

PROOF OF CLAIM. The current set of points is $C\mathbf{r}_d\mathbf{s}_d\mathbf{a}$. If $C\mathbf{a}$ is a corral then so is $C\mathbf{r}_d\mathbf{s}_d\mathbf{a} = C'\mathbf{r}_d\mathbf{s}_d$ (by Lemma 3.4.1, part 3) and the claim holds. If $C\mathbf{a}$ is not a corral, it is enough to prove that the sequence of points removed by the inner loop of Wolfe's method on this set is the same as the sequence on set $C\mathbf{r}_d\mathbf{s}_d\mathbf{a}$. We will show this now by simultaneously analyzing the execution of the inner loop on $C\mathbf{a}$ and $C\mathbf{r}_d\mathbf{s}_d\mathbf{a}$. We distinguish the two cases with the following notation: variables are written without a bar ($^-$) and with a bar, respectively.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_k$ be the sequence of current points constructed by the inner loop on $C\mathbf{a}$. Let $\mathbf{p}_1, \ldots, \mathbf{p}_k$ be the sequence of removed points. Let $C_1, \ldots, C_k$ be the sequence of current sets of points at every iteration. Let $\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_{\overline{k}}$ be the corresponding sequence on $C\mathbf{r}_d\mathbf{s}_d\mathbf{a}$. Let $\overline{\mathbf{p}}_1, \ldots, \overline{\mathbf{p}}_{\overline{k}}$ be the corresponding sequence of removed points. Let $\bar{C}_1, \ldots, \bar{C}_{\bar{k}}$ be the corresponding sequence of current sets of points. We will show inductively: $k = \bar{k}$, there is a one-to-one correspondence between sequences $(\mathbf{x}_i)$ and $(\overline{\mathbf{x}}_i)$, and $(\mathbf{p}_i) = (\overline{\mathbf{p}}_i)$. More specifically, the correspondence is realized by maintaining the following invariant in the inner loop: $\overline{\mathbf{x}}_i$ is a strict convex combination of $\mathbf{x}_i$ and the minimum norm point in $[\mathbf{r}_d, \mathbf{s}_d]$.

For the base case, from Lemma 3.4.1, part 2, we have that $\overline{\mathbf{x}}_1$ is a strict convex combination of $\mathbf{x}_1$ (which is the minimum norm point in $\text{conv}(C)$) and the minimum norm point in segment $[\mathbf{r}_d, \mathbf{s}_d]$, specifically $\mathbf{w} := \frac{m_{d-2}}{4}\mathbf{e}_{d-1}$.

For the inductive step, if $\mathbf{x}_i$ is a strict convex combination of the current set of points C_i, then so is $\overline{\mathbf{x}}_i$ of $\bar{C}_i$ and the inner loop ends in both cases with corrals $C_i = C'$ and $\bar{C}_i = C'\mathbf{r}_d\mathbf{s}_d$, respectively. The claim holds. If $\mathbf{x}_i$ is not a strict convex combination of the current set of points C_i, then neither is $\overline{\mathbf{x}}_i$ of $\bar{C}_i$. The inner loop then continues by computing the minimum norm point $\mathbf{y}$ in $\text{aff}(C_i)$ and $\overline{\mathbf{y}}$ in $\text{aff}(\bar{C}_i)$, respectively. It then finds point $\mathbf{z}$ in $\text{conv}(C_i)$ that is closest to $\mathbf{y}$ in segment $[\mathbf{x}_i, \mathbf{y}]$. It finds $\overline{\mathbf{z}}$, respectively. It then selects a point $\mathbf{p}_i$ to be removed, and a point $\overline{\mathbf{p}}_i$, respectively. From Lemma 3.4.1, part 2, we have that $\overline{\mathbf{y}}$ is a strict convex combination of $\mathbf{y}$ and $\mathbf{w}$.

We will argue that $\overline{\mathbf{z}}$ is a strict convex combination of $\mathbf{z}$ and $\mathbf{w}$. To see this, we note that segment $[\overline{\mathbf{x}}_i, \overline{\mathbf{y}}]$ lies in the hyperplane where the last coordinate is 0. Therefore we only need to intersect it with the part of $\text{conv}(\bar{C}_i)$ that lies in that hyperplane. This part is exactly $\text{conv}(C_i \cup \{\mathbf{w}\})$, which can be written in a more explicit way as the union of all segments of the form $[\mathbf{b}, \mathbf{w}]$ with

$\mathbf{b} \in C_i$. Even more, we only need to look at triangle $\mathbf{w}, \mathbf{x}_i, \mathbf{y}$, as all relevant segments lie on it. The intersection of this triangle with $\text{conv}(C_i)$ is segment $[\mathbf{x}_i, \mathbf{z}]$ and therefore the intersection of the triangle with $\text{conv}(\bar{C}_i)$ is simply triangle $\mathbf{x}_i, \mathbf{z}, \mathbf{w}$. This implies that the intersection between segment $[\overline{\mathbf{x}}_i, \overline{\mathbf{y}}]$ and $\text{conv}(\bar{C}_i)$ is the same as the intersection between segment $[\overline{\mathbf{x}}_i, \overline{\mathbf{y}}]$ and triangle $\mathbf{x}_i, \mathbf{z}, \mathbf{w}$. This intersection is an interval $[\overline{\mathbf{x}}_i, \overline{\mathbf{z}}]$ where $\overline{\mathbf{z}}$ is a strict convex combination of $\mathbf{w}$ and $\mathbf{z}$ and $\overline{\mathbf{z}}$ is the closest point to $\overline{\mathbf{y}}$ in that intersection.

It follows that the set of potential points to be removed is the same for the two executions. Specifically, if $\mathbf{z}$ is a strict convex combination of a certain subset C^* of C_i, then $\overline{\mathbf{z}}$ is a strict convex combination of $C^* \cup \{\mathbf{r}_d, \mathbf{s}_d\}$. The sets of points that can potentially be removed are $C_i \setminus C^*$ and $\bar{C}_i \setminus (C^* \cup \{\mathbf{r}_d, \mathbf{s}_d\}) = C_i \setminus C^*$ (the same), respectively. In particular (under a mild consistency assumption on the way a point is chosen when there is more than one choice; for example, "choose the point with smallest index among potential points"), $\mathbf{p}_i = \overline{\mathbf{p}}_i$. This implies $C_{i+1} = \bar{C}_{i+1}$. Also, $\mathbf{x}_{i+1} = \mathbf{z}$ and $\overline{\mathbf{x}}_{i+1} = \overline{\mathbf{z}}$ is in $[\mathbf{x}_{i+1}, \mathbf{w}]$. This completes the inductive argument about the inner loop and proves the claim. ♢

This completes the proof of Theorem 3.4.4. □

REMARK 3.4.2. *Lastly, we wish to comment that, while we do not expect that the example $P(d)$ provides an exponential lower bound for the behavior of Wolfe's method with the* linopt *insertion rule, the pseudo-polynomial complexity bound in [**LJJ15**] does not give a polynomial bound for Wolfe's method with the* linopt *insertion rule on our example. The eccentricity parameter $\rho = \mathcal{O}(\delta^2/M^2)$ is exponentially small given that $M \geq 1$ and $\delta \leq \frac{1}{4} m_{d-2} \leq 2^{-d+2}$ for $P(d)$.*

CHAPTER 4

Connections and Conclusions

In this chapter, we present results connecting LF and MNP, discuss the computational complexity of these problems, and present some concluding ideas and future directions. Recall that we stated that LF and LP are strongly-polynomial time equivalent in Section 1.2.1. Here we reduce linear programming to the minimum norm point problem over a simplex via a series of strongly-polynomial time reductions [**DLHR18**], thus demonstrating the deep connection between the LF and MNP problems, the two main problems studied in this thesis.

4.1. Issues of Computational Complexity for LP and MNP Problems

In this section, we discuss some preliminary results and problems related to MNP. Before discussing the complexity of methods for LF, LP and MNP, we must guarantee that solutions to these problems, when given by rational data, will be rational (otherwise we can only hope to approximate a solution). We additionally discuss the related problem of computing the *vertex* of minimum norm in a polyhedron, which is an NP-hard problem for an H-polyhedron but an easy strongly-polynomial time problem for a V-polytope.

4.1.1. Rationality of LP and MNP Solutions. First, we prove that computing the minimum norm point in an affine subspace given by either a system of equations or as the affine hull of finitely many points requires only solving a system of linear equations. Additionally, this proves that if the data given is rational then the point of minimum norm will also be rational. Together these show that computation of the point of minimum norm in an affine subspace may be done in strongly-polynomial time. This also proves that the vertices of a polyhedron, $P_{A,\mathbf{b}}$, defined by rational $A, \mathbf{b}$ will be rational. Thus, if a rational LP or LF is feasible, a rational solution will exist.

Lemma 4.1.1. *Computing the point of minimum norm in an affine subspace (given as either $P^{=}_{A,\mathbf{b}}$ for rational $A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$ or as the affine hull of rational $\mathbf{p}_1, ..., \mathbf{p}_m \in \mathbb{R}^n$) requires only solving a system of linear equations (and thus the unique solution is rational).*

PROOF. Proving this claim for either representation is simple after noting that each problem is a convex program with affine constraints. For each, we need only form the Lagrangian and find a zero of the gradient of the Lagrangian with respect to the primal variable.

First, consider the problem $\min \frac{1}{2}\|\mathbf{x}\|^2$ subject to $A\mathbf{x} = \mathbf{b}$. The Lagrangian of this problem moves the constraint into the objective function as $L(\mathbf{x}, \mu) = \frac{1}{2}\|\mathbf{x}\|^2 + \mathbf{y}^T(A\mathbf{x} - \mathbf{b})$. Then the gradient is $\nabla_{\mathbf{x}} L(\mathbf{x}, \mathbf{y}) = \mathbf{x} + A^T\mathbf{y}$. Thus, we have that at the optimum, $\mathbf{x} = -A^T\mathbf{y}$. Substituting this into the constraint $A\mathbf{x} = \mathbf{b}$ yields $-AA^T\mathbf{y} = \mathbf{b}$. Assuming that A has linearly independent rows (meaning that the normals of the hyperplanes defining the affine subspace are independent), AA^T is invertible, so we have $\mathbf{y} = -(AA^T)^{-1}\mathbf{b}$ ($\mathbf{y}$ is the solution to the previous system of linear equations). Substituting this back into our previous constraint yields $\mathbf{x} = A^T(AA^T)^{-1}\mathbf{b}$.

Next, consider the problem $\min \frac{1}{2}\|\mathbf{x}\|^2$ subject to $\mathbf{x} \in \text{aff}(\mathbf{p}_1, ..., \mathbf{p}_m)$. It simplifies the analysis significantly to consider the problem instead in terms of the affine coefficients of $\mathbf{x}$ in terms of the points $\mathbf{p}_i$, $\mathbf{y}$. We will consider P to be the matrix whose columns are the points $\mathbf{p}_1, ..., \mathbf{p}_m$, so $P \in \mathbb{R}^{n\times m}$. Then $\mathbf{x} = P\mathbf{y}$ and the problem becomes $\min \frac{1}{2}\|P\mathbf{y}\|^2$ subject to $\mathbb{1}^T\mathbf{y} = 1$. The Lagrangian of this problem is $L(\mathbf{y}, \mu) = \frac{1}{2}\mathbf{y}^T P^T P\mathbf{y} + \mu(\mathbb{1}^T\mathbf{y} - 1)$ and the gradient is $\nabla_{\mathbf{y}} L(\mathbf{y}, \mu) = P^T P\mathbf{y} + \mu\mathbb{1}$. Thus, we have the constraints $P^T P\mathbf{y} + \mu\mathbb{1} = 0$ and $\mathbb{1}^T\mathbf{y} = 1$ which we can write as the single linear system of equations

$$\begin{bmatrix} P^T P & \mathbb{1} \\ \mathbb{1}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mu \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}.$$

After we solve this system of linear equations, $\mathbf{x}$ is given as $P\mathbf{y}$.

Thus, we have that computing the point of minimum norm in an affine subspace requires only solving a system of linear equations. It also follows that if all of the given input data is rational, then the minimum norm point is rational. □

We use this result to prove the following corollary by considering the face of minimum dimension that contains the point of minimum norm. Computing the point of minimum norm in the polyhedron is equivalent to computing the point of minimum norm in the affine hull of the minimum dimension face containing the minimum norm point.

Corollary 4.1.2. *The point of minimum norm in a polyhedron given by rational data (given either as $P_{A,\mathbf{b}}$ for rational $A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$ or as the convex hull of rational $\mathbf{p}_1, ..., \mathbf{p}_m \in \mathbb{R}^n$) is rational.*

PROOF. Let F be the face of the polyhedron of minimum dimension which contains the point of minimum norm, $\mathbf{x}$; this is equivalent to $\mathbf{x} \in \text{relint}(F)$. Note that if the point of minimum norm in F lies in $\text{relint}(F)$ then it is simultaneously the affine minimizer and convex minimizer; $\mathbf{x}$ is the minimum norm point in $\text{aff}(F)$ and $\text{conv}(F)$. Now, by Lemma 4.1.1, we have that $\mathbf{x}$ is rational. □

We additionally use Lemma 4.1.1 to prove that the vertices of a polyhedron given by rational data are rational. This result is obvious if the polyhedron is a polytope given as the convex hull of finitely many rational points; the vertices are a subset of these points.

Corollary 4.1.3. *The vertices of a polyhedron given as $P_{A,\mathbf{b}}$ for rational $A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$ are rational.*

PROOF. Note that the vertices of a polyhedron $P_{A,\mathbf{b}}$ are given by a subsystem of equations indexed by $I \subset [m]$ where $|I| = n$, $A_I\mathbf{x} = \mathbf{b}_I$. The point that solves this system of equations is the point of minimum norm in the affine subspace given by $P^{=}_{A_I,\mathbf{b}_I}$ (the solution to this system of equations must be unique). The result follows from Lemma 4.1.1. □

Finally, we mention that the ℓ^1-projection of the origin onto a polyhedron in either representation will additionally be rational. These problems may be reformulated as LPs and rationality of an optimal point (it may not be unique) is given by the rationality of the vertices of the feasible polyhedron for this LP reformulation; see [**Sch86**] and references therein. We further note that the ℓ^p-projection of the origin onto a rational polyhedron for $p \neq 1, 2$ need not be rational.

4.1.2. Minimum Norm Vertex of Polyhedron (MIN-NORM VERTEX). Next we wish to mention a related problem. In this thesis, we have considered computing the *point* of minimum norm in a polyhedron. This problem may be computed in polynomial time for polyhedra in either representation via interior point methods or the ellipsoid method. However, one may instead wish to compute the *vertex* of minimum norm in a polyhedron. First, note that if the polytope is given as the convex hull of a finite set of m points, then this problem requires only computing the norm of each point, as some subset of the points will be the vertices of the polytope. This computation takes only $\mathcal{O}(mn)$ algebraic operations where m is the number of points describing the polytope and n is the dimension. Meanwhile, if the polyhedron is given as $P_{A,\mathbf{b}}$, this problem is NP-hard.

Lemma 4.1.4. *MIN-NORM VERTEX of an H-polytope is NP-hard.*

By MIN-NORM VERTEX we mean the problem of determining whether there is a vertex of squared norm at most a given value K.

PROOF. The hardness reduction is from the directed Hamiltonian path problem which is known to be NP-hard (see [**GJ79**, p. 199] and references therein). Let P be the directed st-path polytope of a directed graph $G = (V, A)$ and vertices s, t mentioned in [**FLM97**]. The polytope is defined by the equations and inequalities

$$\begin{aligned}
\sum_{j|(s,j)\in A} x_{sj} - \sum_{j|(j,s)\in A} x_{js} &= 1,\\
\sum_{j|(t,j)\in A} x_{tj} - \sum_{j|(j,t)\in A} x_{jt} &= -1,\\
\sum_{j|(i,j)\in A} x_{ij} - \sum_{j|(j,i)\in A} x_{ji} &= 0 \text{ for all } i \in V - \{s,t\},\\
0 \le x_{ij} &\le M \text{ for all } (i,j) \in A,
\end{aligned}$$

where there is a variable x_{ij} for every edge (i, j) in A. For $M \geq |V|$, there are two types of vertices of P. The vertices of P are either characteristic vectors of the directed st-paths in G [**FLM97**] or at least one of the entries of the vertex must be $x_{ij} = M$. Meanwhile, the *points* in the polytope P correspond to unions of st-paths and cycles. The characteristic vectors of the cycles define the

directions of the edges that extend from the vertices defining an *st*-path to the vertices for which some $x_{ij} = M$. Hamiltonian paths in G correspond to 0-1 vectors in P with exactly $|V| - 1$ ones (which would be a vertex).

Now we introduce the *reflected st-path polytope*, $\bar{P}$, by applying the affine transformation $y_{ij} = 1 - x_{ij}$. Note that the vertices of this polytope correspond to the vertices of polytope P. The vertices of $\bar{P}$ corresponding to directed *st*-paths are 0-1 vectors with at least $|A| - |V| + 1$ ones (since by the absence of cycles in the paths defining the original vertices these vectors have at most $|V| - 1$ ones). Moreover, they have exactly $|A| - |V| + 1$ ones if and only if they correspond to a directed *st*-Hamiltonian path in the graph. The vertices of $\bar{P}$ corresponding to vertices of P with at least one entry equal to M must have at least one entry equal to $-M + 1$, so these have squared norm at least $(M - 1)^2$. We choose $M = \max\{|A| + 2, |V|\}$ so that $(M - 1)^2 \geq (|A| + 1)^2 > |A| - |V| + 1$. Thus, the minimum norm vertex of $\bar{P}$ has squared norm less than or equal to $|A| - |V| + 1$ iff it corresponds to a directed *st*-Hamiltonian path in G. Taking $K := |A| - |V| + 1$ completes the correctness of the hardness reduction. □

4.2. Strongly-Polynomial Reduction of LP to MNP

The main result in this chapter reduces linear programming to finding the minimum norm point in a V-simplex. It was previously known that linear programming reduces to the minimum norm point problem over a polytope in polynomial-time [**FHI06**]. This result is stronger than previous results and connects the MNP problem to the question of the existence of a strongly-polynomial time algorithm for LP [**Sma00**].

4.2.1. Problem Definitions. We give definitions for the problems of linear programming (LP), feasibility (LFE), bounded feasibility (BFP), V-polytope membership (VPM), zero V-polytope membership (ZVPM), zero V-polytope membership decision (ZVPMD), distance to a V-polytope (DVP), and distance to a V-simplex (DVS). (Prefix "V-" means that the respective object is specified as the convex hull of a set of points.) See [**Sch86**, **GLS88**, **Sch03**] for a detailed discussions of strongly polynomial time algorithms.

Definition 4.2.1. *Consider the following computational problems:*

- `LP`: *Given a rational matrix A, a rational column vector $\mathbf{b}$, and a rational row vector $\mathbf{c}^T$, output rational $\mathbf{x} \in \operatorname{argmax}\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$ if $\max\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$ is finite, otherwise output INFEASIBLE if $\{\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$ is empty and else output INFINITE.*
- `LFE`: *Given a rational matrix A and a rational vector $\mathbf{b}$, if $P := \{\mathbf{x} : A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ is nonempty, output a rational $\mathbf{x} \in P$, otherwise output NO.*
- `BFP`: *Given a rational $d \times n$ matrix A, a rational vector $\mathbf{b}$ and a rational value $M > 0$, if $P := \{\mathbf{x} : A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \sum_{i=1}^{n} x_i \leq M\}$ is nonempty, output a rational $\mathbf{x} \in P$, otherwise output NO.*
- `VPM`: *Given a rational $d \times n$ matrix A and a rational vector $\mathbf{b}$, if $P := \{\mathbf{x} : A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \sum_{i=1}^{n} x_i = 1\}$ is nonempty, output a rational $\mathbf{x} \in P$, otherwise output NO.*
- `ZVPM`: *Given a rational $d \times n$ matrix A, if $P := \{\mathbf{x} : A\mathbf{x} = \mathbf{0}, \mathbf{x} \geq \mathbf{0}, \sum_{i=1}^{n} x_i = 1\}$ is nonempty, output a rational $\mathbf{x} \in P$, otherwise output NO.*
- `ZVPMD`: *Given rational points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n \in \mathbb{R}^d$, output YES if $\mathbf{0} \in \operatorname{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n)$ and NO otherwise.*
- `DVP`: *Given rational points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n \in \mathbb{R}^d$ defining $P = \operatorname{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n)$, output $d(\mathbf{0}, P)^2$.*
- `DVS`: *Given $n \leq d + 1$ affinely independent rational points $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n \in \mathbb{R}^d$ defining $(n-1)$-dimensional simplex $P = \operatorname{conv}(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n)$, output $d(\mathbf{0}, P)^2$.*

4.2.2. Reductions. Fujishige et al. first observed that linear programs may be solved by computing a minimum norm point problem [**FHI06**], so this simple geometric problem is also relevant to the theory of algorithmic complexity of linear optimization. The following result demonstrates that we may reduce `LP` to `MNP` over a simplex in strongly-polynomial time. Note that `MNP` solves `DVS` as we may compute the distance, $d(\mathbf{0}, P)^2$, given the point of minimum norm.

Theorem 4.2.2. *`LP` reduces to `DVS` in strongly-polynomial time.*

To prove each of the lemmas below, we illustrate the problem transformation and its strong polynomiality. The first two reductions are highly classical, while those following are intuitive, but we do not believe have been written elsewhere.

Here we prove that LP reduces to LFE in strongly-polynomial time. Note that we include a proof only for completeness; this result may be found in [**Sch86**]. In Section 1.2.1.1, we stated that LP and LF are strongly-polynomial time equivalent. Reducing LFE to LP is trivial as we may simply add an arbitrary linear objective to the LF defined by the given LFE. Thus, LFE, LF and LP are all strongly-polynomial time equivalent.

Lemma 4.2.3. *LP reduces in strongly-polynomial time to LFE.*

PROOF. Let $\mathcal{O}$ denote the FP oracle.

Require: $A \in \mathbb{Q}^{d\times n}, \mathbf{b} \in \mathbb{Q}^d, \mathbf{c} \in \mathbb{Q}^n$.

Invoke $\mathcal{O}$ on

$$\begin{bmatrix} A & -A & I_d \end{bmatrix} \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \mathbf{s} \end{bmatrix} = \mathbf{b}, \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \mathbf{s} \end{bmatrix} \geq \mathbf{0}. \tag{4.1}$$

If the output is NO, output INFEASIBLE.

Invoke $\mathcal{O}$ on

$$\begin{bmatrix} -\mathbf{c}^T & \mathbf{c}^T & \mathbf{b}^T & \\ A & -A & 0 & I_{d+2n+1} \\ 0 & 0 & A^T & \\ 0 & 0 & -A^T & \end{bmatrix} \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \mathbf{y} \\ \mathbf{s} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \\ \mathbf{c} \\ -\mathbf{c} \end{bmatrix}, \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \mathbf{y} \\ \mathbf{s} \end{bmatrix} \geq \mathbf{0}. \tag{4.2}$$

If the output is NO, output INFINITE, else output rational $\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^-$.

Note that a solution

$$\tilde{\mathbf{x}} := \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \mathbf{s} \end{bmatrix}$$

to (4.1) gives a solution to $A\mathbf{x} \leq \mathbf{b}$ and vice versa. Suppose $\tilde{\mathbf{x}}$ satisfies (4.1). Then $A\mathbf{x}^+ - A\mathbf{x}^- + \mathbf{s} = \mathbf{b}$. Define $\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^-$ and note $\mathbf{s} \geq \mathbf{0}$. Then $A\mathbf{x} \leq \mathbf{b}$. Now, suppose $\mathbf{x}$ satisfies $A\mathbf{x} \leq \mathbf{b}$. Let $\mathbf{x}^+$ be the positive coordinates of the vector $\mathbf{x}$ and $\mathbf{x}^-$ be the negative components in absolute value, so $x_i^+ = \max(x_i, 0)$ and $x_i^- = \max(-x_i, 0)$. Define $\mathbf{s} = \mathbf{b} - A\mathbf{x}$. Since $A\mathbf{x} \leq \mathbf{b}$, we have that $\mathbf{s} \geq \mathbf{0}$ and by construction, $\mathbf{x}^+, \mathbf{x}^- \geq \mathbf{0}$. Note that $\begin{bmatrix} A & -A & I \end{bmatrix} \tilde{\mathbf{x}} = A\mathbf{x}^+ - A\mathbf{x}^- + \mathbf{s} = A(\mathbf{x}^+ - \mathbf{x}^-) + \mathbf{b} - A\mathbf{x} = A\mathbf{x} + \mathbf{b} - A\mathbf{x} = \mathbf{b}$.

Note that a solution

$$\tilde{\mathbf{z}} := \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \\ \mathbf{y} \\ \mathbf{s} \end{bmatrix}$$

to (4.2) gives a solution to $\text{argmax}\{\mathbf{c}^T\mathbf{x} | A\mathbf{x} \leq \mathbf{b}\}$ and vice versa. Suppose $\tilde{\mathbf{z}}$ is a solution to (4.2). These are the KKT conditions for the LP $\text{argmax}\{\mathbf{c}^T\mathbf{x} | A\mathbf{x} \leq \mathbf{b}\}$, so $\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^-$ is the optimum. Suppose $\mathbf{x} \in \text{argmax}\{\mathbf{c}^T\mathbf{x} | A\mathbf{x} \leq \mathbf{b}\}$. By strong duality, there exists $\mathbf{y}$ so that $\mathbf{b}^T\mathbf{y} \leq \mathbf{c}^T\mathbf{x}$ and $A^T\mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0}$. Thus, letting $\mathbf{x}^+$ and $\mathbf{x}^-$ be as above, we have

$$-\mathbf{c}^T(\mathbf{x}^+ - \mathbf{x}^-) + \mathbf{b}^T\mathbf{y} \leq 0,\ A(\mathbf{x}^+ - \mathbf{x}^-) \leq \mathbf{b},\ A^T\mathbf{y} \leq \mathbf{c},\ -A^T\mathbf{y} \leq -\mathbf{c}.$$

Now choose $\mathbf{s} \geq \mathbf{0}$ so that

$$\mathbf{c}^T\mathbf{x}^+ - \mathbf{c}^T\mathbf{x}^- + \mathbf{b}^T\mathbf{y} + s_1 = 0,\ A\mathbf{x}^+ - A\mathbf{x}^- + \mathbf{s}_2^{m+1} = \mathbf{b},\ A^T\mathbf{y} + \mathbf{s}_{m+2}^{n+m+1} = \mathbf{c},\ -A^T\mathbf{y} + \mathbf{s}_{n+m+2}^{2n+m+1} = -\mathbf{c}$$

where $\mathbf{s}_i^j$ denotes the subvector of $\mathbf{s}$ of coordinates $s_i, s_{i+1}, ..., s_{j-1}, s_j$. Thus, $\tilde{\mathbf{z}}$ satisfies (4.2).

Clearly, constructing the required FP problems takes strongly polynomial time and we have only two calls to $\mathcal{O}$, so the reduction is strongly-polynomial time. □

Next we prove that LFE reduces to BFP in strongly-polynomial time. We include a proof only for completeness; results equivalent to this lemma are well-known and may be found in [**Sch86**, **Kha79**] among others.

Lemma 4.2.4. *LFE reduces in strongly-polynomial time to BFP.*

PROOF. Let $\mathcal{O}$ denote the oracle for BFP. Suppose $A = (a_{ij}/\alpha_{ij})_{i,j=1}^{d,n}$, $\mathbf{b} = (b_j/\beta_j)_{j=1}^{d}$ and define $D := \max(\max_{i\in[d],j\in[n]} |\alpha_{ij}|, \max_{k\in[d]} |\beta_k|)$ and $N := \max(\max_{i\in[d],j\in[n]} |a_{ij}|, \max_{k\in[d]} |b_k|) + 1$. If the entry of A, $a_{ij}/\alpha_{ij} = 0$ or the entry of $\mathbf{b}$, $b_j/\beta_j = 0$ define $a_{ij} = 0$ and $\alpha_{ij} = 1$ or $b_j = 0$ and $\beta_j = 1$.

Require: $A \in \mathbb{Q}^{d\times n}, \mathbf{b} \in \mathbb{Q}^d$.

Invoke $\mathcal{O}$ on $A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \sum_{i=1}^n x_i \leq nD^{d(n+1)\min(d^3,n^3)}N^{d(n+1)}$. If the output is NO, output NO, else output rational $\mathbf{x}$.

Note that the FP $A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ is feasible if and only if the BFP $A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \sum_{i=1}^n x_i \leq nD^{d(n+1)\min(d^3,n^3)}N^{d(n+1)}$ is feasible. If the BFP is feasible then clearly the FP is feasible. Suppose the FP is feasible. By the theory of minimal faces of polyhedra, we can reduce this to a FP defined by a square matrix, A, in the following way: By [**Che65**, Theorem 1.1], there is a solution, $\mathbf{x}$, with no more than $\min(d, n)$ positive entries so that $A\mathbf{x} = \mathbf{b}$ and the positive entries of $\mathbf{x}$ combine linearly independent columns of A to form $\mathbf{b}$. Let A' denote the matrix containing only these linearly independent columns and $\mathbf{x}'$ denote only the positive entries of $\mathbf{x}$. Then $A'\mathbf{x}' = \mathbf{b}$. Now, note that $A' \in \mathbb{Q}^{d\times m}$ where $m \leq d$. Since the column rank of A' equals the row rank of A', we may remove $d - m$ linearly dependent rows of A' and the corresponding entries of $\mathbf{b}$, forming A'' and $\mathbf{b}'$ so that $A''\mathbf{x}' = \mathbf{b}'$ where $A'' \in \mathbb{Q}^{m\times m}$, $\mathbf{b}' \in \mathbb{Q}^m$ and A'' is a full-rank matrix.

Define $M := \prod_{i,j=1}^m |\alpha''_{i,j}| \prod_{k=1}^m |\beta'_k|$ and note that $M \leq D^{d(n+1)}$. Define $L := \prod_{i,j=1}^m (|a''_{i,j}| + 1) \prod_{k=1}^m (|b'_k| + 1)$ and note that $L \leq N^{d(n+1)}$. Define $\bar{A} = MA''$ and $\bar{\mathbf{b}} = Mb'$ and note that $\bar{A}$ and $\bar{\mathbf{b}}$ are integral. By Cramer's rule, we known that $x'_i = \frac{|\det \bar{A}_i|}{|\det \bar{A}|}$ where $\bar{A}_i$ denotes $\bar{A}$ with the ith column replaced by $\bar{\mathbf{b}}$. By integrality, $|\det \bar{A}| \geq 1$, so $x'_i \leq |\det \bar{A}_i| \leq \prod_{i,j=1}^m M(|a_{ij}| + 1) \prod_{k=1}^m M(|b_k| + 1) = M^{m^3} L \leq D^{d(n+1)\min(d^3,n^3)}N^{d(n+1)}$. Now, note that $\mathbf{x}'$ defines a solution, $\mathbf{x}$, to the original system of

equations. Let $x_i = x'_j$ if the jth column of A' was the selected ith column of A and $x_i = 0$ if the ith column of A was not selected. Note then that $A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \sum_{i=1}^{n} x_i \leq nD^{d(n+1)\min(d^3,n^3)}N^{d(n+1)}$.

Thus, we have that the FP and BFP are equivalent. To see that this is a strongly-polynomial time reduction, note that adding this additional constraint takes time for constructing the number $nD^{d(n+1)\min(d^3,n^3)}N^{d(n+1)}$ plus small constant time. This number takes $d(n+1)$ comparisons and $d(n+1)\min(d^3, n^3)$ multiplications to form. Additionally, this number takes space which is polynomial in the size of the input (polynomial in d,n and size of D, N). □

Next we prove that BFP reduces to VPM in strongly-polynomial time. We include a proof for completeness as we do not know of a reference.

Lemma 4.2.5. *BFP reduces in strongly-polynomial time to VPM.*

PROOF. Let $\mathcal{O}$ denote the oracle for VPM.

Require: $A \in \mathbb{Q}^{d\times n}, b \in \mathbb{Q}^d, 0 < M \in \mathbb{Q}$.

Invoke $\mathcal{O}$ on

$$\begin{bmatrix} MA & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ z \end{bmatrix} = \mathbf{b}, \begin{bmatrix} \mathbf{y} \\ z \end{bmatrix} \geq \mathbf{0}, z + \sum_{i=1}^{n} y_i = 1. \tag{4.3}$$

If the output from $\mathcal{O}$ is NO, then output NO, else output rational $\mathbf{x} = M\mathbf{y}$.

Note that a solution

$$\tilde{\mathbf{w}} := \begin{bmatrix} \mathbf{y} \\ z \end{bmatrix}$$

to (4.3) gives a solution the BFP instance, $A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \sum_{i=1}^{n} x_i \leq M$ and vice versa. Suppose $\tilde{\mathbf{w}}$ satisfies (4.3). Then $\mathbf{x} = M\mathbf{y}$ is a solution to the BFP instance since $A\mathbf{x} = MA\mathbf{y} = \mathbf{b}$ and since $\mathbf{y} \geq \mathbf{0}$, $\mathbf{x} = M\mathbf{y} \geq \mathbf{0}$ and since $\sum_{i=1}^{n} y_i + z = 1$, we have $\sum_{i=1}^{n} y_i \leq 1$ so $\sum_{i=1}^{n} x_i = M\sum_{i=1}^{n} y_i \leq M$. Suppose $\mathbf{x}$ is a solution to the BFP instance. Then $\mathbf{y} = \frac{1}{M}\mathbf{x}$ and $z = 1 - \sum_{i=1}^{n} y_i$ satisfies (4.3), since $\begin{bmatrix} MA & 0 \end{bmatrix} \tilde{\mathbf{w}} = MA\mathbf{y} = A\mathbf{x} = \mathbf{b}$, $\mathbf{y} \geq \mathbf{0}$ since $\mathbf{x} \geq \mathbf{0}$ and since $\sum_{i=1}^{n} x_i \leq M$, we have $\sum_{i=1}^{n} y_i = \frac{1}{M}\sum_{i=1}^{n} x_i \leq 1$ so $z \geq 0$.

Clearly, this reduction is simply a rewriting, so the reduction is strongly-polynomial time. □

Here we prove that VPM reduces to ZVPM in strongly-polynomial time. We include a proof only for completeness as we do not know of a reference. Note that this result is obvious due to simple translation arguments.

Lemma 4.2.6. *VPM reduces in strongly-polynomial time to ZVPM.*

PROOF. Let $\mathcal{O}$ be the oracle for ZVPM.

Require: $A \in \mathbb{Q}^{d\times n}, b \in \mathbb{Q}^d$.

Invoke $\mathcal{O}$ on

$$\left[\mathbf{a}_1 - \mathbf{b} \quad \mathbf{a}_2 - \mathbf{b} \quad \cdots \quad \mathbf{a}_n - \mathbf{b}\right] \mathbf{x} = \mathbf{0}, \mathbf{x} \geq \mathbf{0}, \sum_{i=1}^{n} x_i = 1 \tag{4.4}$$

where $\mathbf{a}_i \in \mathbb{Q}^m$ is the ith column of A. If the output is from $\mathcal{O}$ is NO, then output NO, else output rational $\mathbf{x}$.

Note that a solution to (4.4) gives a solution to the VPM instance and vice versa. Note that $\mathbf{x}$ satisfies (4.4) if and only if $0 = \sum_{i=1}^n x_i(\mathbf{a}_i - \mathbf{b}) = \sum_{i=1}^n x_i\mathbf{a}_i - \mathbf{b}\sum_{i=1}^n x_i = A\mathbf{x} - \mathbf{b}$ so $A\mathbf{x} = \mathbf{b}$. Thus, $\mathbf{x}$ is a solution to the VPM instance if and only if $\mathbf{x}$ is a solution to (4.4).

Clearly, this reduction is simply a rewriting, so the reduction is strongly-polynomial time. □

Here we prove that ZVPM reduces to ZVPMD in strongly-polynomial time. We include a proof for completeness; this result follows from the strong-polynomiality of solving systems of linear equations which is a classical result.

Lemma 4.2.7. *ZVPM reduces in strongly-polynomial time to ZVPMD.*

Proof idea. The reduction sequentially asks for every vertex whether it is redundant and if so, it removes it and continues. This process ends with at most $d+1$ vertices so that $\mathbf{x}$ is a strict convex combination of them and the coefficients x_i can be found in this resulting case by solving a linear system. □

PROOF. Let $\mathcal{O}$ denote the ZVPMD oracle.

Require: $P := \{\mathbf{A}_1, \ldots, \mathbf{A}_n\} \subseteq \mathbb{Q}^d$ where A_i is the ith column of A.

Invoke $\mathcal{O}$ on P. If the output is NO, output NO.

for $i = 1, \ldots, n$ **do**

Invoke $\mathcal{O}$ on instance P without $\mathbf{A}_i$. If the output from $\mathcal{O}$ is YES, remove $\mathbf{A}_i$ from P.

end for

Let m be the cardinality of P.

Output the solution $x_1, \ldots, x_m$ to the linear system $\sum x_i = 1$, $\sum_{\mathbf{p}_i \in P} x_i \mathbf{p}_i = \mathbf{0}$

Let P^* be the resulting set of points P after the loop in the reduction. Claim: P^* contains at most $d+1$ points so that $\mathbf{0}$ is a strict convex combination of (all of) them. Proof of claim: By Caratheodory's theorem there is a subset $Q \subseteq P^*$ of at most $d+1$ points so that $\mathbf{0}$ is a strict convex combination of points in Q. We will see that P^* is actually equal to Q. Suppose not, for a contradiction. Let $\mathbf{p} \in P^* \setminus Q$. At the time the loop in the reduction examines $\mathbf{p}$, no point in Q has been removed and therefore $\mathbf{p}$ is redundant and is removed. This is a contradiction. □

Here we prove that ZVPMD reduces to DVS in strongly-polynomial time. This reduction, as far as we know, is new and the proof of our theorem depends upon it.

In our next lemma, we make use of the following elementary fact.

CLAIM 10. *Given A an $m \times n$ matrix let B be A with a row of 1's appended. The columns of A are affinely independent if and only if the columns of B are linearly independent. The convex hull of the columns of A is full dimensional if and only if rank of B is $m+1$.*

Lemma 4.2.8. *ZVPMD reduces in strongly-polynomial time to DVS.*

PROOF. Clearly ZVPMD reduces in strongly-polynomial time to DVP: Output YES if the distance is 0, output NO otherwise.

Given an instance of distance to a V-polytope, $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$, we reduce it to an instance of DVS as follows: We lift the points to an affinely independent set in higher dimension, a simplex,

by adding small-valued new coordinates. Claim 10 allows us to handle affine independence in matrix form. Let A be the $d \times n$ matrix having columns $(\mathbf{p}_i)_{i=1}^n$. Let $\mathbf{v}_1, \ldots, \mathbf{v}_d$ be the rows of A. Let $\mathbf{v}_0 \in \mathbb{R}^n$ be the all-ones vector. We want to add vectors $\mathbf{v}_{d+1}, \ldots, \mathbf{v}_{d+t}$, for some t, so that $\mathbf{v}_0, \ldots, \mathbf{v}_{d+t}$ is of rank n. To this end, we construct an orthogonal basis (but not normalized, to preserve rationality) of the orthogonal complement of span $(\mathbf{v}_0, \ldots, \mathbf{v}_d)$. The basis is obtained by applying the Gram-Schmidt orthogonalization procedure (without the normalization step) to the sequence $\mathbf{v}_0, \ldots, \mathbf{v}_d, \mathbf{e}_1, \ldots, \mathbf{e}_n$. Denote $\mathbf{v}_{d+1}, \ldots, \mathbf{v}_{d+t}$ the resulting orthogonal basis of the orthogonal complement of span $(\mathbf{v}_0, \ldots, \mathbf{v}_d)$. The matrix with rows $\mathbf{v}_0, \ldots, \mathbf{v}_d, \mathbf{v}_{d+1}, \ldots, \mathbf{v}_{d+t}$ is of rank n and so is the matrix with rows

$$\mathbf{v}_0, \ldots, \mathbf{v}_d, \epsilon\mathbf{v}_{d+1}, \ldots, \epsilon\mathbf{v}_{d+t}$$

for any $\epsilon > 0$ (to be fixed later). Therefore, the n columns of this matrix are linearly independent. Let B be the matrix with rows

$$\mathbf{v}_1, \ldots, \mathbf{v}_d, \epsilon\mathbf{v}_{d+1}, \ldots, \epsilon\mathbf{v}_{d+t}.$$

Let $\mathbf{w}_1, \ldots, \mathbf{w}_n$ be the columns of B. By construction and Claim 10 they are affinely independent. Let S denote the convex hull of these $(n-1)$-dimensional rational points. Polytope S is a simplex. Moreover, if $Q := \text{conv}(\mathbf{p}_1, \ldots, \mathbf{p}_n)$, then

$$d(\mathbf{0}, S)^2 \leq d(\mathbf{0}, Q)^2 + \epsilon^2 \sum_{i=d+1}^{d+t} \|\mathbf{v}_i\|_2^2 \leq d(\mathbf{0}, Q)^2 + \epsilon^2 n$$

(where we use that $\|\mathbf{v}_i\|_2 \leq 1$, from the Gram-Schmidt construction). Note that the application of Gram-Schmidt without normalization causes the bit-size of the numbers involved in the problem to grow, but only polynomially, thus preserving the strong-polynomiality of the reduction; see [**Sch86**, Section 3.3].

The reduction proceeds as follows: Let T be the maximum of the absolute values of all numerators and denominators of entries in $(\mathbf{p}_i)_{i=1}^n$. This can be computed in strongly polynomial time; without loss of generality we can assume that the input is integral, and then take T to be the maximum of the absolute values of all entries in $(\mathbf{p}_i)_{i=1}^n$, as done in Schrijver's [**Sch86**, Section 15.2]. From

Lemma 4.2.9, we have $d(\mathbf{0}, Q)^2 \geq \frac{1}{d(dT)^{2d}}$ if $\mathbf{0} \notin Q$. Compute rational $\epsilon > 0$ so that $\epsilon^2 n < \frac{1}{d(dT)^{2d}}$. For example, let $\epsilon := \frac{1}{nd(dT)^d}$. The reduction queries $d(\mathbf{0}, S)^2$ for S constructed as above and given by the choice of ϵ we just made. It then outputs YES if $d(\mathbf{0}, S)^2 < \frac{1}{d(dT)^{2d}}$ and NO otherwise. □

Lemma 4.2.9. *Let $P = \text{conv}(\mathbf{p}_1, \ldots, \mathbf{p}_n)$ be a V-polytope with $\mathbf{p}_i \in \mathbb{Q}^d$. Let T be the maximum of the absolute values of all numerators and denominators of entries in $(\mathbf{p}_i)_{i=1}^n$. If $\mathbf{0} \notin P$ then $d(\mathbf{0}, P) \geq \frac{1}{(dT)^d\sqrt{d}}$.*

Proof. The claim is clearly true if P is empty. If P is non-empty, let $\mathbf{y} = \text{proj}_P(\mathbf{x})$. We have that every facet of P can be written as $\mathbf{a}^T\mathbf{x} \leq k$, where $\mathbf{a}(\neq 0)$ is an integral vector, k is an integer and the absolute values of the entries of $\mathbf{a}$ as well as k are less than $(dT)^d$ ([**GLS81**, Theorem 3.6]). By assumption at least one these facet inequalities is violated by $\mathbf{0}$. Denote by $\mathbf{a}^T\mathbf{x} \leq k$ one such inequality. Let $H = \{\mathbf{x} : \mathbf{a}^T\mathbf{x} = k\}$. We have $\|\mathbf{y}\|_2 = d(\mathbf{0}, P) \geq d(\mathbf{0}, H)$, and $d(\mathbf{0}, H)^2 = k^2 / \|\mathbf{a}\|_2^2 \geq \frac{1}{d(dT)^{2d}}$. The claim follows. □

Note that we have that LP reduces to ZVPMD in strongly polynomial time. Thus, to solve LP, we need only be able to decide if $\mathbf{0}$ is a member of a given polytope. By positive-definiteness of norms ($\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = \mathbf{0}$), we have that if projection of $\mathbf{0}$ in *any norm* onto a V-polytope may be computed in strongly-polynomial time (computing this projection computes distance, which computes membership), then this provides a strongly-polynomial time algorithm for LP.

4.3. Conclusions and Future Work

In this thesis, we have considered several projection methods used in optimization and data science. We presented analyses of Motzkin's methods, the randomized Kaczmarz methods, and their generalization, the *Sampling Kaczmarz-Motzkin* (SKM) methods, which are all members of the family of iterative projection methods for LF. We also considered Wolfe's methods for MNP, which compute projections onto a convex polytope. We additionally demonstrated the strong connection between LF and MNP. Below we outline the main contributions of this thesis and some future directions in more detail.

We provided a theoretical analysis for Motzkin's method for inconsistent systems of linear equations [**HN18a**]. We showed that by using this greedy selection strategy, Motzkin's method exhibits an accelerated convergence rate until a particular threshold is reached. This threshold depends on the dynamic range of the residual, and could be estimated to employ a strategy that yields acceleration without sacrificing convergence accuracy. We provided experiments and concrete analysis for Gaussian systems that support our claims. This work has raised some intuitive questions concerning this method.

Future directions regarding Motzkin's method:

- Provide an analysis of the expected dynamic range for relevant systems (other than Gaussian systems).
- Use the analysis of the accelerated convergence rate of Motzkin's method to provide an analysis of the acceleration of SKM.
- Explore the advantage of Motzkin's method in parallel architectures.
- Extend this work beyond systems of linear equations to general `LF` problems.

We presented a framework of methods for using the randomized Kaczmarz method to detect and remove corruptions in a system of linear equations [**HN17**, **HN18b**]. We provided theoretical bounds on the probability that the windowed Kaczmarz methods will successfully detect and remove all corrupted equations. Moreover, we provided ample experimental evidence that these methods successfully detect corrupted equations and these results far surpass the theoretical guarantees. We present next some natural future work.

Future directions regarding the randomized Kaczmarz method:

- Provide a tighter convergence rate analysis of the randomized Kaczmarz method for specific types of systems. This will additionally provide better detection guarantees for the windowed Kaczmarz methods.
- Analyze how the windowed Kaczmarz methods depend upon the size of the consistent system in order to apply methods to `MAX-FS`.

- Generalize the windowed Kaczmarz methods and guarantees beyond systems of linear equations to general LF problems.

We naturally generalized the Motzkin's methods and the randomized Kaczmarz methods to the class of Sampling Kaczmarz-Motzkin methods [**DLHN17**]. We provided a theoretical analysis that generalizes earlier results and presents a potential acceleration of this convergence rate with the right choices of parameters. We wish to note that, by easy polarization-homogenization of the information (where the hyperplane normals a_i are thought of as points and the solution vector x is a separating plane), one can reinterpret SKM as a type of *stochastic gradient descent* (SGD). We additionally showed that the SKM methods detect and certify feasibility. We gave a lower bound on the probability that an SKM iterate is not a certificate of feasibility if the LF has a solution. We additionally showed that the SKM methods on a full-dimensional system terminate with projection parameter $\lambda = 2$. Finally, we showed numerous experimental results comparing the SKM methods to Motzkin's method and the randomized Kaczmarz methods. We plan to consider the following questions next.

Future directions regarding the Sampling Kaczmarz-Motzkin methods:

- Explore the SKM sampling scheme for more general stochastic gradient descent methods.
- Identify the optimal choices for β and λ for specific classes of systems.
- Explore applying Chubanov's style of generating additional linear inequalities to the SKM methods.

In the second half of this thesis, we presented a thorough exposition on Wolfe's method for MNP. We presented examples that demonstrate that the behavior of the algorithm depends upon the choice of insertion rule, and demonstrated that the algorithm may have inefficiencies with all insertion rules. Our main result demonstrated that Wolfe's method using a natural insertion rule exhibits exponential behavior [**DLHR17**, **DLHR18**]. This analysis has raised the following natural questions.

Future directions regarding Wolfe's methods:

- Present exponential examples for other insertion rules for Wolfe's method, especially the *linopt* rule which is used in submodular function minimization.
- Analyze the expected behavior of Wolfe's method with a randomized insertion rule.
- Study the behavior of Wolfe's method on base polytopes, the polytopes involved in submodular function minimization.
- Perform a smoothed analysis of Wolfe's method, or at least study the behavior of Wolfe's method on data following a prescribed random distribution.
- Present examples of exponential behavior of Wolfe's method on simplices, which are deeply connected to the theoretical complexity of linear programming.
- Explore extensions of Wolfe's method for other convex ℓ^p norms for $p \geq 1$ and other projection-like operators. Note that the Frank-Wolfe method (which is similar to Wolfe's method) solves general convex programs; however, we intend to extend Wolfe's method (which uses affine minimization rather than a line search) to other convex functions. This has not yet been explored, as far as we are aware.

Finally, we also showed that the minimum norm point problem for simplices is intimately related to the complexity of linear programming. Our strongly-polynomial reduction of `LP` to `MNP` over a simplex demonstrated that a strongly-polynomial time algorithm for `MNP` would provide a strongly-polynomial time algorithm for `LP`.

APPENDIX A

MATLAB Code

In this appendix, we provide MATLAB code for all methods discussed in this thesis. This code is also freely available from `https://www.math.ucdavis.edu/~jhaddock`. This code has been written in MATLAB [**MAT16**].

The following is MATLAB code for Method 2.1, Motzkin's method (MM) for linear feasibility.

```
%NAME: Motzkin's method (MM) for linear feasibility
%TASK: approximately solves feasible Ax <= b
%AUTHOR: Jamie Haddock
%OUTSIDE FUNCTIONS: none
%INPUT: matrix A, vector b, vector x0 (starting point),
%       scalar lambda in (0,2] (projection parameter),
%       pos. integer k (number of iterations)
%OUTPUT: approximate solution vector, x

function x=Motzkin(A,b,x0,lambda,k)
    xj = x0;

    for j = 1:k
        %compute residual
        res = A*xj - b;

        %find most violated constraint
        [maxres,maxind] = max(res);

        % project into corresponding halfspace
        if maxres > 0
            a = A(maxind,:);
            xj = xj - lambda*(maxres/(norm(a)^2)) * a';
        end
    end

    %return last iterate
    x = xj;
end
```

The following is MATLAB code for Method 2.2, Motzkin's method for systems of linear equations.

```
%NAME: Motzkin's method for systems of linear equations
%TASK: approximately solves feasible Ax = b
%AUTHOR: Jamie Haddock
%OUTSIDE FUNCTIONS: none
%INPUT: matrix A, vector b, vector x0 (starting point),
%       pos. integer k (number of iterations)
%OUTPUT: approximate solution vector, x

function x=MotzkinLS(A,b,x0,k)

    xj = x0;

    for j = 1:k
        %compute residual
        res = A*xj - b;

        %find most violated equation
        [~,maxind] = max(abs(res));

        % project into corresponding hyperplane
        a = A(maxind,:);
        xj = xj - (res(maxind)/(norm(a)^2)) * a';
    end

    %return last iterate
    x = xj;
end
```

The following is MATLAB code for Method 2.3, the randomized Kaczmarz method (RK) for linear feasibility.

```
%NAME: Randomized Kaczmarz (RK) for linear feasibility
%TASK: approximately solves feasible Ax <= b
%AUTHOR: Jamie Haddock
%OUTSIDE FUNCTIONS: none
%INPUT: matrix A, vector b, vector x0 (starting point),
%       scalar lambda in (0,2] (projection parameter),
%       pos. integer k (number of iterations)
%OUTPUT: approximate solution vector, x

function x=RK(A,b,x0,lambda,k)
    xj = x0;
    fronorm = norm(A,'fro');

    for j = 1:k
        %generate random index with probability proportional to squared row norm
        randval = rand * fronorm^2;
        subfronorm = 0;
        ind = 1;
        a = A(ind,:);
        rownorm = norm(a)^2;
        while subfronorm + rownorm < randval
            subfronorm = subfronorm + rownorm;
            ind = ind + 1;
            a = A(ind,:);
            rownorm = norm(a)^2;
        end

        % project into corresponding halfspace
        resval = a*xj - b(ind);
        if resval > 0
            xj = xj - lambda*(resval/(norm(a)^2)) * a';
        end
    end

    %return last iterate
    x = xj;
end
```

The following is MATLAB code for Method 2.4, the randomized Kaczmarz method for systems of linear equations.

```
%NAME: Randomized Kaczmarz for systems of linear equations
%TASK: approximately solves feasible Ax = b
%AUTHOR: Jamie Haddock
%OUTSIDE FUNCTIONS: none
%INPUT: matrix A, vector b, vector x0 (starting point),
%       pos. integer k (number of iterations)
%OUTPUT: approximate solution vector, x

function x=RKLS(A,b,x0,k)
    xj = x0;
    fronorm = norm(A,'fro');

    for j = 1:k
        %generate random index with probability proportional to squared row norm
        randval = rand * fronorm^2;
        subfronorm = 0;
        ind = 1;
        a = A(ind,:);
        rownorm = norm(a)^2;
        while subfronorm + rownorm < randval
            subfronorm = subfronorm + rownorm;
            ind = ind + 1;
            a = A(ind,:);
            rownorm = norm(a)^2;
        end

        % project into corresponding hyperplane
        xj = xj - ((a*xj-b(ind))/(norm(a)^2)) * a';
    end

    %return last iterate
    x = xj;
end
```

The following is MATLAB code for Method 2.5, the windowed Kaczmarz method with removal for corrupted systems of linear equations.

```
%NAME: Windowed Kaczmarz with Removal for corrupted linear systems
%TASK: removes corrupted equations from Ax = b and solves remaining system
%AUTHOR: Jamie Haddock
%OUTSIDE FUNCTIONS: none
%INPUT: matrix A, vector b, pos. integer k (number of RK iterations),
%       pos. integer W (number of windows),
%       pos. integer d (number of equations removed each window)
%OUTPUT: solution to remaining system, x

function x = WKwR(A,b,k,W,d)
    B = A; c = b;
    n = size(B,2);

    for i = 1:W
        %compute the window iterate by running k iterations of RK on 0
        xi = RKLS(B,c,zeros(n,1),k);

        %remove the rows corresponding to the d largest entries of the residual
        [~,maxind] = sort(abs(B*xi-c),'descend');
        B(maxind(1:d),:) = [];
        c(maxind(1:d)) = [];
    end

    %return a solution of the remaining equations
    x = B\c;
end
```

The following is MATLAB code for Method 2.6, the windowed Kaczmarz method without removal for corrupted systems of linear equations.

```
%NAME: Windowed Kaczmarz without Removal for corrupted linear systems
%TASK: detects corrupted equations from Ax = b and solves remaining system
%AUTHOR: Jamie Haddock
%OUTSIDE FUNCTIONS: none
%INPUT: matrix A, vector b, pos. integer k (number of RK iterations),
%       pos. integer W (number of windows),
%       pos. integer d (number of equations removed each window)
%OUTPUT: solution to remaining system, x

function x = WKwoR(A,b,k,W,d)
    n = size(A,2);
    S = [];

    for i = 1:W
        %compute the window iterate by running k iterations of RK on 0
        xi = RKLS(A,b,zeros(n,1),k);

        %record the rows corresponding to the d largest entries of the residual
        [~,maxind] = sort(abs(A*xi-b),'descend');
        D = maxind(1:d);
        S = [S; D];
    end

    %return a solution of the remaining equations
    A(S,:) = [];
    b(S,:) = [];
    x = A\b;
end
```

The following is MATLAB code for Method 2.7, the windowed Kaczmarz method without removal with unique selection for corrupted systems of linear equations.

```
%NAME: Windowed Kaczmarz without Removal with Unique Selection for corrupted linear
%      systems
%TASK: detects corrupted equations from Ax = b and solves remaining system
%AUTHOR: Jamie Haddock
%OUTSIDE FUNCTIONS: none
%INPUT: matrix A, vector b, pos. integer k (number of RK iterations),
%       pos. integer W (number of windows),
%       pos. integer d (number of equations removed each window)
%OUTPUT: solution to remaining system, x

function x = WKwoRUS(A,b,k,W,d)
    n = size(A,2);
    S = [];

    for i = 1:W
        %compute the window iterate by running k iterations of RK on 0
        xi = RKLS(A,b,zeros(n,1),k);

        %record the rows corresponding to the d largest unrecorded entries of the
        %residual
        [~,maxind] = sort(abs(A*xi-b),'descend');
        maxind = setdiff(maxind,S);
        D = maxind(1:d);
        S = [S; D];
    end

    %return a solution of the remaining equations
    A(S,:) = [];
    b(S,:) = [];
    x = A\b;
end
```

The following is MATLAB code for Method 2.8, the sampling Kaczmarz-Motzkin method (SKM) for linear feasibility.

```
%NAME: Sampling Kaczmarz-Motzkin (SKM) for linear feasibility
%TASK: approximately solves feasible Ax <= b
%AUTHOR: Jamie Haddock
%OUTSIDE FUNCTIONS: none
%INPUT: m x n matrix A, vector b, vector x0 (starting point),
%       scalar lambda in (0,2] (projection parameter),
%       pos. integer beta in [1,m] (sample size),
%       pos. integer k (number of iterations)
%OUTPUT: approximate solution vector, x

function x=SKM(A,b,x0,lambda,beta,k)
    xj = x0;
    m = size(A,1);

    for j = 1:k
        %generate uniform random sample of size beta
        tauj = sort(randsample(m,beta),'ascend');

        %find most violated constraint of sample
        [maxres,sampind] = max(A(tauj,:)*xj-b(tauj));
        maxind = tauj(sampind);
        a = A(maxind,:);

        % project into corresponding halfspace
        if maxres > 0
            xj = xj - lambda*(maxres/(norm(a)^2)) * a';
        end
    end

    %return last iterate
    x = xj;
end
```

The following is MATLAB code for Method 3.9, Wolfe's method for minimum norm point.

```
%NAME: Wolfe's method for minimum norm point
%TASK: computes the point of minimum norm in a convex polytope given as the
%      convex hull of finitely many points
%AUTHOR: Jamie Haddock
%OUTSIDE FUNCTIONS: none
%INPUT: m x n matrix P containing points as rows (m points in R^n),
%       initrule in (‘minnorm’,‘first’),
%       addrule in (‘minnorm’,‘linopt’,‘first’),
%       displayOn in (‘on’,‘off’)
%OUTPUT: approximate solution vector, x

function [solution, data] = wolfe(P,initRule,addRule,displayOn)
    data = [];
    corrals = {};
    corralnum = 1;
    n = size(P,2);
    m = size(P,1);

    %implementation of Wolfe's method on a polytope given as an mxn matrix
    %of the vertices (m points in R^n)

    %initRule options are 'minnorm',‘first’: give rule for initial vertex
    %selection; if 'minnorm', adds the vertex of minimum norm; if ‘first’,
    %adds the first vertex in the given matrix order

    %addRule options are ‘minnorm’, ‘linopt’, ‘first’: give rule for vertex to be
    %added selection; if ‘minnorm’, adds the vertex out of the candidates
    %that has minimum norm; if ‘linopt’, adds the vertex, p, out of the
    %candidates that minimizes p'x; if ‘first’, adds the vertex out of the
    %candidates that appears first in the given matrix order

    %display options are ‘on’, ‘off’: if ‘on’, we print information in each
    %cycle; if ‘off’, we don't print any cycle information

    majorCycle = 0;
    minorCycle = 0;

    %check displayOn value
    if ~strcmp(displayOn,‘on’) && ~strcmp(displayOn,‘off’)
        display(‘Check your displayOn value.’);
        solution = inf;
        return
    end

    %initial vertex selection according to selected rule
```

```
if strcmp(initRule,‘first’)
    x = P(1,:); %define x to be the first listed vertex
    C = [x]; %set potential corral to be this vertex
    lambda = [1];
    Cind = [1]; %keep track of index of this vertex in P
elseif strcmp(initRule,‘minnorm’)
    N = sqrt(sum(P.^2,2));
    [val, i] = min(N);
    x = P(i,:);
    C = [x];
    lambda = [1];
    Cind = [i];
else
    display(‘Check your initRule value.’);
    solution = inf;
    return;
end

%Step 0: initial vertex selection
if strcmp(displayOn,‘on’)
    display([‘Step 0: ’,num2str(majorCycle),‘   ’, num2str(minorCycle),
     ‘       ’, num2str(x)]);
    display([Cind’ C]);
end

data = [data; majorCycle + 0.1*minorCycle size(C,1)];
corrals{corralnum} = Cind;
corralnum = corralnum + 1;
while (norm(x,2) >= eps) && (min(P*x’) + eps < x*x’)
    %check that x not= 0 and
    %that we don’t already have the min. norm point (i.e., no vertex
    %has inner product with x less than ||x||^2
    Cprevious = C;

    majorCycle = majorCycle + 1;
    minorCycle = 0;

    %added vertex selected according to selected rule
    if strcmp(addRule,‘first’)
        %find vertices with inner product with x less than ||x||^2
        indOptions = find(P*x’ < x*x’);
        %select first option, but only if this is not already in C
        while (size(indOptions,1)>0) && (ismember(indOptions(1),Cind))
            indOptions = indOptions(2:end);
        end
        if size(indOptions,1) > 0
            j = indOptions(1);
```

```
        C = [C; P(j,:)]; %add this vertex to potential corral
        lambda = [lambda; 0];
        Cind = [Cind,j]; %keep track of indices of potential corral
    end
elseif strcmp(addRule,'minnorm')
    %find vertices with inner product with x less than ||x||^2
    indOptions = find(P*x' < x*x');
    %select minimum norm option, if this is not already in C
    N = sqrt(sum(P(indOptions,:).^2,2));
    [val,i] = min(N);
    while (size(indOptions,1)>0) && (ismember(indOptions(i),Cind))
        indOptions(i) = [];
        if size(indOptions,1) > 0
            N = sqrt(sum(P(indOptions,:).^2,2));
            [val,i] = min(N);
        end
    end
    if size(indOptions,1) > 0
        j = indOptions(i);
        C = [C; P(j,:)]; %add this vertex to potential corral
        lambda = [lambda; 0];
        Cind = [Cind,j]; %keep track of indices of potential corral
    end
elseif strcmp(addRule,'linopt')
    %find vertices with inner product with x less than ||x||^2
    indOptions = find(P*x' < x*x');
    %select minimum p'x option, if this is not already in C
    [minproduct,i] = min(P(indOptions,:)*x');
    while (size(indOptions,1)>0) && (ismember(indOptions(i),Cind))
        indOptions(i) = [];
        if size(indOptions,1) > 0
            [minproduct,i] = min(P(indOptions,:)*x');
        end
    end
    if size(indOptions,1) > 0
        j = indOptions(i);
        C = [C; P(j,:)]; %add this vertex to potential corral
        lambda = [lambda; 0];
        Cind = [Cind,j]; %keep track of indices of potential corral
    end
else
    display('Check your addRule value.');
    solution = inf;
    return
end

%check to see that a vertex was added, otherwise end
```

```
if size(Cprevious) == size(C)
    if Cprevious == C
        solution = x;
        display('Repeated a corral.');
        return
    end
end

%Step 1: major cycle vertex addition
if strcmp(displayOn,'on')
    display(['Step 1: ',num2str(majorCycle),'   ', num2str(minorCycle),
     '       ', num2str(x)]);
    display([Cind' C]);
end

%find min morm point in aff(C)
k = size(C,1);
alpha = lsqlin([0 ones(1,k); ones(k,1) C*C'],[1; zeros(k,1)],[],[]);
alpha = alpha(2:end,:);
y = (C'*alpha)';

%Step 2: found MNP(aff(C)) for current potential corral
if strcmp(displayOn,'on')
    display(['Step 2: ',num2str(majorCycle),'   ', num2str(minorCycle),
     '       ', num2str(x), '       ', num2str(y)]);
    display([Cind' C]);
end

data = [data; majorCycle + 0.1*minorCycle size(C,1)];


while min(alpha) < -eps  %check if y is in conv(C)
    minorCycle = minorCycle + 1;
    negind = find(alpha < -eps);
    %x is in conv(C)
    lambda = lsqlin(C',x',[],[]);
    %find convex constant that describes point along x-y that is in
    %conv(C) and closest to y
    theta = min(lambda(negind)./(lambda(negind)-alpha(negind)));

    x = (C'*(theta*alpha + (1-theta)*lambda))';

    %this new point is not a convex combination of one of the
    %points in C
    i = find(theta*alpha + (1-theta)*lambda<=eps);
    lambda = theta*alpha + (1-theta)*lambda;
    %take this point out of C and Cind
```

```
            if size(i,1) > 0
                C(i(1),:) = [];
                lambda(i(1)) = [];
                Cind(i(1)) = [];
            else
                i = negind(1);
                C(i,:) = [];
                lambda(i) = [];
                Cind(i) = [];
            end

            %Step 3: minor cycle vertex removal
            if strcmp(displayOn,'on')
                display(['Step 3: ',num2str(majorCycle),'   ',
                 num2str(minorCycle), '       ', num2str(x)]);
                display([Cind' C]);
            end

            %find min morm point in aff(C)
            k = size(C,1);
            alpha = lsqlin([0 ones(1,k); ones(k,1) C*C'],[1; zeros(k,1)],[],[]);
            alpha = alpha(2:end,:);
            y = (C'*alpha)';

            %Step 2: found MNP(aff(C)) for current potential corral
            if strcmp(displayOn,'on')
                display(['Step 2: ',num2str(majorCycle),'   ',
                 num2str(minorCycle), '       ', num2str(x), '       ',
                 num2str(y)]);
                display([Cind' C]);
            end
            data = [data; majorCycle + 0.1*minorCycle size(C,1)];
        end

        x = y;
        lambda = alpha;
        corrals{corralnum} = Cind;
        corralnum = corralnum + 1;
    end
    solution = x;
    if strcmp(displayOn,'on')
        display(['Corrals: ',num2str(size(corrals,2))]);
        for i = 1:size(corrals,2)
            display(corrals{i});
        end
    end
end
```

Bibliography

[ABGJ18] X. Allamigeon, P. Benchimol, S. Gaubert, and M. Joswig, *Log-barrier interior point methods are not strongly polynomial*, SIAM J. Appl. Algebra Geom. (2018), to appear.

[ADG14] H. Avron, A. Druinsky, and A. Gupta, *Revisiting asynchronous linear solvers: Provable convergence rate through randomization*, IEEE 28th Int. Parallel and Distributed Processing Symposium, IEEE, 2014, pp. 198–207.

[Agm54] S. Agmon, *The relaxation method for linear inequalities*, Canadian J. Math. **6** (1954), 382–392.

[AH05] E. Amaldi and R. Hauser, *Boundedness theorems for the relaxation method*, Math. Oper. Res. **30** (2005), no. 4, 939–955.

[AK95] E. Amaldi and V. Kann, *The complexity and approximability of finding maximum feasible subsystems of linear relations*, Theoret. Computer Science **147** (1995), no. 1-2, 181–210.

[Aub93] J.-P. Aubin, *Optima and equilibria: An introduction to nonlinear analysis, vol. 140 of*, Graduate Texts in Mathematics (1993).

[AWL14] A. Agaskar, C. Wang, and Y. M. Lu, *Randomized Kaczmarz algorithms: Exact MSE analysis and optimal sampling probabilities*, IEEE Global Conf. on Signal and Information Processing (GlobalSIP), IEEE, 2014, pp. 389–393.

[AWY14] S. Agrawal, Z. Wang, and Y. Ye, *A dynamic near-optimal algorithm for online linear programming*, Oper. Res. **62** (2014), no. 4, 876–890.

[AZ99] N. Amenta and G. M. Ziegler, *Deformed products and maximal shadows of polytopes*, Contemporary Math. **223** (1999), 57–90.

[B+13] F. Bach et al., *Learning with submodular functions: A convex optimization perspective*, Foundations and Trends® in Machine Learning **6** (2013), no. 2-3, 145–373.

[Bár82] I. Bárány, *A generalization of Carathéodory's theorem*, Discrete Math. **40** (1982), no. 2-3, 141–152.

[Bar02] A. Barvinok, *A course in convexity*, vol. 54, American Mathematical Society, 2002.

[BBK17] C. Battaglino, G. Ballard, and T. G. Kolda, *A practical randomized CP tensor decomposition*, January 2017, `https://arxiv.org/abs/1701.06600`.

[BDE09] A. M. Bruckstein, D. L. Donoho, and M. Elad, *From sparse solutions of systems of equations to sparse modeling of signals and images*, SIAM Rev. **51** (2009), no. 1, 34–81.

[BDJ14] A. Basu, J. A. De Loera, and M. Junod, *On Chubanov's method for linear programming*, INFORMS J. Computing **26** (2014), no. 2, 336–350.

[Bet04] U. Betke, *Relaxation, new combinatorial and polynomial algorithms for the linear feasibility problem*, Discrete Comput. Geom. **32** (2004), no. 3, 317–338.

[BK80] A. Bachem and B. Korte, *Minimum norm problems over transportation polytopes*, Linear Algebra Appl. **31** (1980), 103–118.

[BN] J. Briskman and D. Needell, *Block Kaczmarz method with inequalities*, J. Math. Imaging Vis. **52**, no. 3, 385–396.

[BO97] I. Bárány and S. Onn, *Colourful linear programming and its relatives*, Math. Oper. Res. **22** (1997), no. 3, 550–567.

[Bor82] K.-H. Borgwardt, *The average number of pivot steps required by the simplex-method is polynomial*, Math. Methods Oper. Res. **26** (1982), no. 1, 157–177.

[Bre67] L. Bregman, *The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming*, USSR Comp. Math. and Mathematical Physics **7** (1967), no. 3, 200 – 217, translated by G. Kiss.

[BT04] A. Beck and M. Teboulle, *A conditional gradient method with linear rate of convergence for solving convex linear systems*, Math. Methods Oper. Res. **59** (2004), no. 2, 235–247.

[BV04] S. Boyd and L. Vanderberghe, *Convex optimization*, Cambridge Univ. Press, 2004.

[CE14] G. Calafiore and L. El Ghaoui, *Optimization models*, Control systems and optimization series, Cambridge University Press, October 2014.

[CEG83] Y. Censor, P. P. Eggermont, and D. Gordon, *Strong underrelaxation in Kaczmarz's method for inconsistent systems*, Numer. Math. **41** (1983), no. 1, 83–92.

[Cen81] Y. Censor, *Row-action methods for huge and sparse systems and their applications*, SIAM review **23** (1981), no. 4, 444–466.

[Che65] S. N. Chernikov, *The convolution of finite systems of linear inequalities*, USSR Comp. Math. and Mathematical Physics **5** (1965), no. 1, 1–24, translated by H.F. Cleaves.

[Chu12] S. Chubanov, *A strongly polynomial algorithm for linear systems having a binary solution*, Math. Program. **134** (2012), no. 2, 533–570.

[CJK14] D. Chakrabarty, P. Jain, and P. Kothari, *Provable submodular minimization using Wolfe's algorithm*, Proc. Advances Neural Info. Proc. Systems (NIPS), 2014, pp. 802–809.

[CLO07] D. Cox, J. Little, and D. O'shea, *Ideals, varieties, and algorithms*, vol. 3, Springer, 2007.

[CP12] X. Chen and A. Powell, *Almost sure convergence of the Kaczmarz algorithm with random measurements*, J. Fourier Anal. Appl. (2012), 1–20.

[CR16] G. Calvillo and D. Romero, *On the closest point to the origin in transportation polytopes*, Discrete Applied Math. **210** (2016), 88–102.

[CT08] E. J. Candes and T. Tao, *Reflections on compressed sensing*, IEEE Information Theory Society Newsletter **58** (2008), no. 4, 20–23.

[Dan48] G. B. Dantzig, *Linear programming in problems for the numerical analysis of the future*, Proc. Symposium on Modern Calculating Machinery and Numer. Methods, UCLA, July, 1948, pp. 29–31.

[Dan92] ———, *An ϵ-precise feasible solution to a linear program with a convexity constraint in $1/\epsilon^2$ iterations independent of problem size*, Tech. report, Stanford University, 1992.

[DHO+16] J. Draisma, E. Horobeţ, G. Ottaviani, B. Sturmfels, and R. R. Thomas, *The Euclidean distance degree of an algebraic variety*, Foundations Comput. Math. **16** (2016), no. 1, 99–149.

[DHST08] A. Deza, S. Huang, T. Stephen, and T. Terlaky, *The colourful feasibility problem*, Discrete Applied Math. **156** (2008), no. 11, 2166–2177.

[DLHN17] J. A. De Loera, J. Haddock, and D. Needell, *A sampling Kaczmarz-Motzkin algorithm for linear feasibility*, SIAM J. Sci. Comp. **39** (2017), no. 5, S66–S87.

[DLHR17] J. A. De Loera, J. Haddock, and L. Rademacher, *The minimum Euclidean-norm point on a convex polytope: Wolfe's combinatorial algorithm is exponential*, 2017, arXiv:1710.02608.

[DLHR18] ———, *The minimum Euclidean-norm point on a convex polytope: Wolfe's combinatorial algorithm is exponential*, Proc. 50th ACM Symposium on Theory of Computing (STOC), 2018.

[DLRS10] J. A. De Loera, J. Rambau, and F. Santos, *Triangulations: Structures for algorithms and applications*, Springer, 2010.

[DSX11] A. Deza, T. Stephen, and F. Xie, *More colourful simplices*, Discrete Comput. Geom. **45** (2011), no. 2, 272–278.

[Dum14] B. Dumitrescu, *On the relation between the randomized extended Kaczmarz algorithm and coordinate descent*, BIT Numer. Math. (2014), 1–11.

[EF00] 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** (2000), no. 3, Ser. A, 451–485.

[EHL81] P. P. B. Eggermont, G. T. Herman, and A. Lent, *Iterative algorithms for large partitioned linear systems, with applications to image reconstruction*, Linear Algebra Appl. **40** (1981), 37–67.

[EK12] Y. C. Eldar and G. Kutyniok, *Compressed sensing: theory and appl.*, Cambridge University Press, 2012.

[Elf80] T. Elfving, *Block-iterative methods for consistent and inconsistent linear equations*, Numer. Math. **35** (1980), no. 1, 1–12.

[EN11] Y. C. Eldar and D. Needell, *Acceleration of randomized Kaczmarz method via the Johnson-Lindenstrauss lemma*, Numer. Algorithms **58** (2011), no. 2, 163–177.

[FHI06] S. Fujishige, T. Hayashi, and S. Isotani, *The minimum-norm-point algorithm applied to submodular function minimization and linear programming*, Kyoto University. Research Institute for Mathematical Sciences [RIMS], 2006.

[FI11] S. Fujishige and S. Isotani, *A submodular function minimization algorithm based on the minimum-norm base*, Pacific J. Optimization **7** (2011), 3–17.

[FLM97] K. Fukuda, T. M. Liebling, and F. Margot, *Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron*, Comput. Geom. **8** (1997), no. 1, 1 – 12.

[FO85] R. M. Freund and J. B. Orlin, *On the complexity of four polyhedral set containment problems*, Math. Program. **33** (1985), no. 2, 139–145.

[FR13] S. Foucart and H. Rauhut, *A mathematical introduction to compressive sensing*, vol. 1, Birkhäuser Basel, 2013.

[Fre87] R. M. Freund, *Dual gauge programs, with applications to quadratic programming and the minimum-norm problem*, Math. Program. **38** (1987), no. 1, 47–67.

[Fuj80] S. Fujishige, *Lexicographically optimal base of a polymatroid with respect to a weight vector*, Math. Oper. Res. **5** (1980), no. 2, 186–196.

[FW56] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Research Logistics (NRL) **3** (1956), no. 1-2, 95–110.

[GBH70] R. Gordon, R. Bender, and G. T. Herman, *Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and x-ray photography*, J. Theoret. Biol. **29** (1970), 471–481.

[GHJ75] R. Gordon, G. T. Herman, and S. A. Johnson, *Image reconstruction from projections*, Sci. Am. **233** (1975), no. 4, 56–71.

[Gil66] E. G. Gilbert, *An iterative procedure for computing the minimum of a quadratic form on a convex set*, SIAM J. Control **4** (1966), 61–80.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and intractability*, W. H. Freeman and Co., San Francisco, Calif., 1979, A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[GJY11] D. Ge, X. Jiang, and Y. Ye, *A note on the complexity of L_p minimization*, Math. Program. **129** (2011), no. 2, 285–299.

[GLS81] M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica **1** (1981), no. 2, 169–197.

[GLS88] ———, *Geometric algorithms and combinatorial optimization*, Springer-Verlag, Berlin, 1988.

[GO12] M. Griebel and P. Oswald, *Greedy and randomized versions of the multiplicative Schwarz method*, Linear Algebra Appl. **437** (2012), no. 7, 1596–1610.

[Gof80] J.-L. Goffin, *The relaxation method for solving systems of linear inequalities*, Math. Oper. Res. **5** (1980), no. 3, 388–414.

[Gof82] ———, *On the nonpolynomiality of the relaxation method for systems of linear inequalities*, Math. Program. **22** (1982), no. 1, 93–103.

[GPS16] E. Gallopoulos, B. Philippe, and A. H. Sameh, *Preconditioners*, Parallelism in Matrix Computations, Springer, 2016, pp. 311–341.

[GR15] R. M. Gower and P. Richtárik, *Randomized iterative methods for linear systems*, SIAM J. Matrix Anal. A. **36** (2015), no. 4, 1660–1690.

[Han07] P. C. Hansen, *Regularization tools version 4.0 for MATLAB 7.3*, Numer. Algorithms **46** (2007), no. 2, 189–194.

[HLL78] G. T. Herman, A. Lent, and P. H. Lutz, *Relaxation methods for image reconstruction*, Commun. ACM **21** (1978), no. 2, 152–158.

[HM93] G. T. Herman and L. B. Meyer, *Algebraic reconstruction techniques can be made computationally efficient*, IEEE Trans. Medical Imaging **12** (1993), no. 3, 600–609.

[HMSW53] A. J. Hoffman, M. Mannos, D. Sokolowsky, and N. Wiegmann, *Computational experience in solving linear programs*, J. Soc. Industrial Appl. Math. **1** (1953), no. 1, pp. 17–33.

[HN90] M. Hanke and W. Niethammer, *On the acceleration of Kaczmarz's method for inconsistent linear systems*, Linear Algebra Appl. **130** (1990), 83–98.

[HN01] J. K. Hunter and B. Nachtergaele, *Applied analysis*, World Scientific Publishing Company, 2001.

[HN17] J. Haddock and D. Needell, *Randomized projections for corrupted linear systems*, Proc. 15th Int. Conf. of Numerical Analysis and Applied Mathematics, 2017.

[HN18a] ———, *On Motzkin's method for inconsistent linear systems*, (2018), Submitted.

[HN18b] ———, *Randomized projection methods for corrupted linear systems*, (2018), Submitted.

[HNR15] A. Hefny, D. Needell, and A. Ramdas, *Rows vs. columns: Randomized Kaczmarz or Gauss-Seidel for ridge regression*, SIAM J. Sci. Comp. (2015), To appear.

[Hof52] A. J. Hoffman, *On approximate solutions of systems of linear inequalities*, J. Research Nat. Bur. Standards **49** (1952), 263–265.

[HS78] C. Hamaker and D. C. Solmon, *The angles between the null spaces of x-rays*, J. Math. Anal. Appl. **62** (1978), no. 1, 1–23.

[HYZ08] E. T. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation for ℓ_1-minimization: Methodology and convergence*, SIAM J. Optimization **19** (2008), no. 3, 1107–1130.

[Kac37] S. Kaczmarz, *Angenäherte auflösung von systemen linearer gleichungen*, Bull. Internat. Acad. Polon. Sci. Lettres A (1937), 335–357.

[Kar84] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica **4** (1984), no. 4, 373–395.

[Kha79] L. G. Khachiyan, *A polynomial algorithm in linear programming*, Dokl. Akad. Nauk SSSR **244** (1979), no. 5, 1093–1096, translated by D. E. Brown.

[KKM09] B. Kirchheim, E. Kopecká, and S. Müller, *Do projections stay close together?*, J. Math. Anal. Appl. **350** (2009), no. 2, 859–871.

[KM72] V. Klee and G. J. Minty, *How good is the simplex algorithm?*, Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; dedicated to the memory of Theodore S. Motzkin), Academic Press, New York, 1972, pp. 159–175.

[KR12] E. Kopecká and S. Reich, *A note on alternating projections in Hilbert space*, J. Fixed Point Theory Appl. **12** (2012), no. 1-2, 41–47.

[KTK80] M. K. Kozlov, S. P. Tarasov, and L. G. Khachiyan, *The polynomial solvability of convex quadratic programming*, USSR Comp. Math. and Mathematical Physics **20** (1980), no. 5, 223–228, translated by J. Berry.

[Lay82] S. R. Lay, *Convex sets and their applications*, John Wiley & Sons, Inc., New York, 1982, Pure and Applied Mathematics, A Wiley-Interscience Publication.

[LH95] C. L. Lawson and R. J. Hanson, *Solving least squares problems*, Classics in Applied Mathematics, vol. 15, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995, Revised reprint of the 1974 original.

[Lic13] M. Lichman, *UCI machine learning repository*, 2013, `http://archive.ics.uci.edu/ml`.

[LJJ13] S. Lacoste-Julien and M. Jaggi, *An affine invariant linear convergence analysis for Frank-Wolfe algorithms*, arXiv preprint arXiv:1312.7864 (2013).

[LJJ15] ———, *On the global linear convergence of Frank-Wolfe optimization variants*, Proc. Advances Neural Info. Proc. Systems (NIPS), 2015, pp. 496–504.

[LL10] D. Leventhal and A. S. Lewis, *Randomized methods for linear constraints: convergence rates and conditioning*, Math. Oper. Res. **35** (2010), no. 3, 641–654.

[LMY16] Y. Li, K. Mo, and H. Ye, *Accelerating random Kaczmarz algorithm based on clustering information.*, Proc. 30th Conference on Artificial Intelligence (AAAI), 2016, pp. 1823–1829.

[LW15] J. Liu and S. J. Wright, *An accelerated randomized Kaczmarz algorithm*, Math. Computation (2015).

[LWS14] J. Liu, S. J. Wright, and S. Sridhar, *An asynchronous parallel randomized Kaczmarz algorithm*, arXiv preprint arXiv:1401.4780 (2014).

[MAT16] MATLAB, *version 9.0.0 (r2016a)*, The MathWorks Inc., Natick, Massachusetts, 2016.

[MD13] F. Meunier and A. Deza, *A further generalization of the colourful Carathéodory theorem*, Discrete Geometry and Optimization, Springer, 2013, pp. 179–190.

[MGC11] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and Bregman iterative methods for matrix rank minimization*, Math. Program. **128** (2011), no. 1-2, 321–353.

[MKC00] K. G. Murty, S. N. Kabadi, and R. Chandrasekaran, *Infeasibility analysis for linear systems, a survey*, Arabian J. Sci. Engineering **25** (2000), no. 1; PART C, 3–18.

[MNR15] A. Ma, D. Needell, and A. Ramdas, *Convergence properties of the randomized extended Gauss-Seidel and Kaczmarz methods*, SIAM J. Matrix Anal. A. **36** (2015), no. 4, 1590–1604.

[MS54] T. S. Motzkin and I. J. Schoenberg, *The relaxation method for linear inequalities*, Canadian J. Math. **6** (1954), 393–404.

[MS16] F. Meunier and P. Sarrabezolles, *Colorful linear programming, nash equilibrium, and pivots*, Discrete Applied Math. (2016).

[MTA81] J.-F. Maurras, K. Truemper, and M. Akgül, *Polynomial algorithms for a class of linear programs*, Math. Program. **21** (1981), no. 2, 121–136.

[Nat95] B. K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM J. Comput. **24** (1995), no. 2, 227–234.

[Nat01] F. Natterer, *The mathematics of computerized tomography*, vol. 32, Society for Industrial and Applied Mathematics, Philadelphia, PA; SIAM, 2001.

[Nee10] D. Needell, *Randomized Kaczmarz solver for noisy linear systems*, BIT Numer. Math. **50** (2010), no. 2, 395–403.

[Net] Netlib, *The Netlib Linear Programming Library*, `www.netlib.org/lp`.

[NSV+16] J. Nutini, B. Sepehry, A. Virani, I. Laradji, M. Schmidt, and H. Koepke, *Convergence Rates for Greedy Kaczmarz Algorithms*, Proc. 32nd Conference on Uncertainty in Artificial Intelligence (UAI), 2016.

[NSW14] D. Needell, N. Srebro, and R. Ward, *Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm*, Proc. Advances Neural Info. Proc. Systems (NIPS), 2014.

[NT09] D. Needell and J. A. Tropp, *CoSaMP: Iterative signal recovery from incomplete and inaccurate samples*, Applied and Computational Harmonic Analysis **26** (2009), no. 3, 301–321.

[NT13] ———, *Paved with good intentions: Analysis of a randomized block Kaczmarz method*, Linear Algebra Appl. (2013).

[NV09] D. Needell and R. Vershynin, *Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit*, Foundations of computational mathematics **9** (2009), no. 3, 317–334.

[NW13] D. Needell and R. Ward, *Two-subspace projection method for coherent overdetermined linear systems*, J. Fourier Anal. Appl. **19** (2013), no. 2, 256–269.

[NZZ15] D. Needell, R. Zhao, and A. Zouzias, *Randomized block Kaczmarz method with projection for solving least squares*, Linear Algebra Appl. **484** (2015), 322–343.

[OZ15] P. Oswald and W. Zhou, *Convergence analysis for Kaczmarz-type methods in a Hilbert space framework*, Linear Algebra Appl. **478** (2015), 131–161.

[OZ17] ———, *Random reordering in SOR-type methods*, Numerische Mathematik **135** (2017), no. 4, 1207–1220.

[PnRS16] J. Peña, D. Rodríguez, and N. Soheili, *On the von Neumann and Frank-Wolfe algorithms with away steps*, SIAM J. Optim. **26** (2016), no. 1, 499–512.

[PP15] S. Petra and C. Popa, *Single projection Kaczmarz extended algorithms*, Numer. Algorithms (2015), 1–16, 1504.00231.

[PPKR12] C. Popa, T. Preclik, H. Köstler, and U. Rüde, *On Kaczmarz's projection iteration as a direct solver for linear least squares problems*, Linear Algebra Appl. **436** (2012), no. 2, 389–404.

[Ros58] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, Cornell Aeronautical Laboratory, Psychological Review **65** (1958), no. 6, 386–408.

[RT12] P. Richtárik and M. Takáč, *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Math. Program. (2012), 1–38.

[RT17] ———, *Stochastic reformulations of linear systems: algorithms and convergence theory*, arXiv preprint arXiv:1706.01108 (2017).

[Sch86] A. Schrijver, *Theory of linear and integer programming*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons Ltd., 1986, A Wiley-Interscience Publication.

[Sch03] ———, *Combinatorial optimization. Polyhedra and efficiency. Vol. A*, Algorithms and Combinatorics, vol. 24, Springer-Verlag, Berlin, 2003, Paths, flows, matchings, Chapters 1–38.

[She02] R. Sheldon, *A first course in probability*, Pearson Education India, 2002.

[Sma00] S. Smale, *Mathematical problems for the next century*, Mathematics: frontiers and perspectives, Amer. Math. Soc., Providence, RI, 2000, pp. 271–294.

[SNW12] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for machine learning*, MIT Press, 2012.

[ST04] D. A. Spielman and S.-H. Teng, *Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time*, J. Assoc. Computing Machinery **51** (2004), no. 3, 385–463.

[ST08] T. Stephen and H. Thomas, *A quadratic lower bound for colourful simplicial depth*, J. Combinatorial Optimization **16** (2008), no. 4, 324–327.

[SV09] T. Strohmer and R. Vershynin, *A randomized Kaczmarz algorithm with exponential convergence*, J. Fourier Anal. Appl. **15** (2009), 262–278.

[Tan71] K. Tanabe, *Projection method for solving a singular system of linear equations and its applications*, Numer. Math. **17** (1971), no. 3, 203–214.

[Tar86] E. Tardos, *A strongly polynomial algorithm to solve combinatorial linear programs*, Oper. Res. **34** (1986), no. 2, 250–256.

[Tel82] J. Telgen, *On relaxation methods for systems of linear inequalities*, European J. Oper. Res. **9** (1982), no. 2, 184–189.

[TG07] J. A. Tropp and A. C. Gilbert, *Signal recovery from random measurements via orthogonal matching pursuit*, IEEE Trans. Inform. Theory **53** (2007), no. 12, 4655–4666.

[Tib96] R. Tibshirani, *Regression shrinkage and selection via the LASSO*, J. Royal Statistical Society. Series B (Methodological) (1996), 267–288.

[Tod13] M. J. Todd, *The computation of fixed points and applications*, vol. 124, Springer Science & Business Media, 2013.

[TZ93] T. Terlaky and S. Zhang, *Pivot rules for linear programming: A survey on recent theoretical developments*, Annals of Operations Research **46** (1993), no. 1, 203–233.

[Vég16] L. A. Végh, *A strongly polynomial algorithm for a class of minimum-cost flow problems with separable convex objectives*, SIAM J. Comput. **45** (2016), no. 5, 1729–1761.

[Ver09] R. Vershynin, *Beyond hirsch conjecture: Walks on random polytopes and smoothed complexity of the simplex method*, SIAM J. Comput. **39** (2009), no. 2, 646–678.

[Ver12] ———, *Introduction to the non-asymptotic analysis of random matrices*, Compressed Sensing, Cambridge Univ. Press, 2012, pp. 210–268.

[VZ14] L. A. Végh and G. Zambelli, *A polynomial projection-type algorithm for linear programming*, Oper. Res. Lett. **42** (2014), no. 1, 91–96.

[WAL15] C. Wang, A. Agaskar, and Y. M. Lu, *Randomized Kaczmarz algorithm for inconsistent linear systems: An exact MSE analysis*, Sampling Theory and Applications (SampTA), 2015 International Conference on, IEEE, 2015, pp. 498–502.

[WM67] T. M. Whitney and R. K. Meany, *Two algorithms related to the method of steepest descent*, SIAM J. Numer. Anal. **4** (1967), no. 1, 109–118.

[Wol76] P. Wolfe, *Finding the nearest point in a polytope*, Math. Program. **11** (1976), no. 1, 128–149.

[XZ02] J. Xu and L. Zikatanov, *The method of alternating projections and the method of subspace corrections in Hilbert space*, J. Amer. Math. Soc. **15** (2002), no. 3, 573–597.

[YL09] I. C. Yeh and C. H. Lien, *The comparison of data mining techniques for the predictive accuracy of probability of default of credit card clients*, Expert Systems Appl. **36** (2009), no. 2, 2473–2480.

[ZF13] A. Zouzias and N. M. Freris, *Randomized extended Kaczmarz for solving least-squares*, SIAM J. Matrix Anal. A. **34** (2013), no. 2, 773–793.

[Zie12] G. M. Ziegler, *Lectures on polytopes*, vol. 152, Springer Science & Business Media, 2012.

Index