Journal of Machine Learning Research 23 (2022) 1-48 Submitted 6/21; Revised 9/22; Published 9/22

Faster Randomized Interior Point Methods for Tall/Wide Linear Programs*

Agniva Chowdhury CHOWDHURYA@ORNL.GOV
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37830, USA

Gregory Dexter GDEXTER@PURDUE.EDU
Department of Computer Science
Purdue University
West Lafayette, IN 47907, USA

Palma London PLONDON@CORNELL.EDU
Operations Research and Information Engineering
Cornell University
Ithaca, NY 14853, USA

Haim Avron HAIMAV@TAUEX.TAU.AC.IL
School of Mathematical Sciences
Tel Aviv University
Tel Aviv, Israel

Petros Drineas PDRINEAS@PURDUE.EDU
Department of Computer Science
Purdue University
West Lafayette, IN 47907, USA

Editor: Manfred Warmuth

Abstract

Linear programming (LP) is an extremely useful tool which has been successfully applied to solve various problems in a wide range of areas, including operations research, engineering, economics, or even more abstract mathematical areas such as combinatorics. It is also used in many machine learning applications, such as ℓ_1-regularized SVMs, basis pursuit, nonnegative matrix factorization, etc. Interior Point Methods (IPMs) are one of the most popular methods to solve LPs both in theory and in practice. Their underlying complexity is dominated by the cost of solving a system of linear equations at each iteration. In this paper, we consider both *feasible* and *infeasible* IPMs for the special case where the number of variables is much larger than the number of constraints. Using tools from Randomized Linear Algebra, we present a preconditioning technique that, when combined with the iterative solvers such as Conjugate Gradient or Chebyshev Iteration, provably guarantees that IPM algorithms (suitably modified to account for the error incurred by the approximate solver), converge to a feasible, approximately optimal solution, without increasing their iteration complexity. Our empirical evaluations verify our theoretical results on both real-world and synthetic data.

Keywords: Randomized Linear Algebra; Linear Programming; Interior Point Methods.

∗. This paper includes some material from the conference paper (Chowdhury et al., 2020).

©2022 Agniva Chowdhury, Gregory Dexter, Palma London, Haim Avron, and Petros Drineas.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v23/21-0709.html.

1. Introduction

Linear programming (LP) is one of the most useful tools available to theoreticians and practitioners throughout science and engineering. It has been extensively used to solve various problems in a wide range of areas, including operations research, engineering, economics, or even in more abstract mathematical areas such as combinatorics. In machine learning and numerical optimization, LP appears in numerous settings, including ℓ_1-regularized SVMs (Zhu et al., 2004), basis pursuit (BP) (Yang and Zhang, 2011), sparse inverse covariance matrix estimation (SICE) (Yuan, 2010), the nonnegative matrix factorization (NMF) (Recht et al., 2012), MAP inference (Meshi and Globerson, 2011), adversarial deep learning (Weng et al., 2018; Wong and Kolter, 2018) etc. Not surprisingly, designing and analyzing LP algorithms is a topic of paramount importance in computer science and applied mathematics.

The first algorithm for general-purpose LPs was the famous *simplex algorithm*, proposed by (Dantzig, 1951). It worked well in practice, but was shown to have exponential worst-case running times (Klee and Minty, 1972). The first polynomial time algorithm for general LPs was the *ellipsoid method* (Khachiyan, 1979), which is rather slow in practice compared to the simplex algorithm. This motivated further research on LP algorithms which are efficient in both theory and practice. One of the most successful paradigms for solving LPs is the family of Interior Point Methods (IPMs), pioneered by Karmarkar in the mid 1980s (Karmarkar, 1984). Path-following IPMs (also called central-path algorithms) and, in particular, long-step path following IPMs, are among the most practical approaches for solving linear programs. See Section 1.2 for a detailed overview of recent work on path-following IPMs.

Consider the standard form of the primal LP problem:

$$\min \mathbf{c}^\mathsf{T}\mathbf{x}\,, \text{ subject to } \mathbf{A}\mathbf{x} = \mathbf{b}\,, \mathbf{x} \geq \mathbf{0}\,, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{c} \in \mathbb{R}^n$ are the inputs, and $\mathbf{x} \in \mathbb{R}^n$ is the vector of the primal variables. The associated dual problem is

$$\max \mathbf{b}^\mathsf{T}\mathbf{y}\,, \text{ subject to } \mathbf{A}^\mathsf{T}\mathbf{y} + \mathbf{s} = \mathbf{c}\,, \mathbf{s} \geq \mathbf{0}\,, \tag{2}$$

where $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{s} \in \mathbb{R}^n$ are the vectors of the dual and slack variables respectively. Triplets $(\mathbf{x}, \mathbf{y}, \mathbf{s})$ that uphold both eqns. (1) and (2) are called *primal-dual solutions*. Path-following IPMs typically converge towards a primal-dual solution by operating as follows: given the current iterate $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k)$, they compute the Newton search direction $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$ and update the current iterate by making a step towards the search direction.

To compute the search direction, one standard approach (Nocedal and Wright, 2006) involves solving the *normal equations*[1]:

$$\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\Delta\mathbf{y} = \mathbf{p}. \tag{3}$$

Here, $\mathbf{D} = \mathbf{X}^{1/2}\mathbf{S}^{-1/2}$ is a diagonal matrix, $\mathbf{X}, \mathbf{S} \in \mathbb{R}^{n\times n}$ are diagonal matrices whose i-th diagonal entries are equal to $\mathbf{x}_i$ and $\mathbf{s}_i$, respectively, and $\mathbf{p} \in \mathbb{R}^m$ is a vector whose

1. Another widely used approach is to solve the augmented system (Nocedal and Wright, 2006). This approach is less relevant for this paper.

exact definition is given in eqn. (23)[2]. Given $\Delta\mathbf{y}$, computing $\Delta\mathbf{s}$ and $\Delta\mathbf{x}$ only involves matrix-vector products.

The core computational bottleneck in IPMs is the need to solve the linear system of eqn. (3) at each iteration. This leads to two key challenges: first, for high-dimensional matrices $\mathbf{A}$, solving the linear system is computationally prohibitive. Most implementations of IPMs use a *direct solver*; see Chapter 6 of (Nocedal and Wright, 2006). However, if $\mathbf{AD}^2\mathbf{A}^\mathsf{T}$ is large and dense, direct solvers are computationally impractical. If $\mathbf{AD}^2\mathbf{A}^\mathsf{T}$ is sparse, specialized direct solvers have been developed, but these do not apply to many LP problems, especially those arising in machine learning applications, due to irregular sparsity patterns. Second, an alternative to direct solvers is the use of iterative solvers, but the situation is further complicated since $\mathbf{AD}^2\mathbf{A}^\mathsf{T}$ is typically ill-conditioned. Indeed, as IPM algorithms approach the optimal primal-dual solution, the diagonal matrix $\mathbf{D}$ becomes ill-conditioned, which also results in the matrix $\mathbf{AD}^2\mathbf{A}^\mathsf{T}$ becoming ill-conditioned. Additionally, using approximate solutions for the linear system of eqn. (3) causes certain invariants, which are crucial for guaranteeing the convergence of IPMs, to be violated; see Section 1.1 for details.

In this paper, we address the aforementioned challenges, for the special case where $m \ll n$, i.e., the number of constraints is much smaller than the number of variables; see Section 6 for a generalization. This is a common setting in many applications of LP solvers. For example, in machine learning, ℓ_1-SVMs and basis pursuit problems often exhibit such structure when the number of available features (n) is larger than the number of objects (m). Indeed, this setting has been of interest in recent work on LPs (Donoho and Tanner, 2005; Bienstock and Iyengar, 2006; London et al., 2018).

For simplicity of exposition, we also assume that the constraint matrix $\mathbf{A}$ has full rank, equal to m. First, we propose and analyze two Krylov subspace-based solvers, namely, preconditioned Conjugate Gradient (CG) and preconditioned Chebyshev iteration for the normal equations of eqn. (3), using matrix sketching constructions from the Randomized Linear Algebra (RLA) literature. We develop a preconditioner for $\mathbf{AD}^2\mathbf{A}^\mathsf{T}$ using matrix sketching which allows us to prove strong convergence guarantees for the *residual* of both the CG and the Chebyshev iteration. Second, building upon the work of Monteiro and O'Neal (2003), we propose and analyze a provably accurate long-step IPM algorithm. Our framework works for both *feasible* and *infeasible* starting points. The proposed IPM solves the normal equations using iterative solvers. We note that a non-trivial concern is that the use of iterative solvers and matrix sketching tools implies that the normal equations at each iteration will be solved only approximately. In our proposed IPM framework, we develop a novel way to *correct* for the error induced by the approximate solution in order to guarantee convergence. Importantly, this correction step is relatively computationally light, unlike a similar step previously proposed by Monteiro and O'Neal (2003), which works similarly, but is computationally inefficient. Third, we empirically show that our algorithm performs well in practice. We consider solving LPs that arise from ℓ_1-regularized SVMs and test them on a variety of synthetic and real-world data sets. Several extensions of our work are discussed in Section 6.

2. The superscript k in eqn. (23) simply indicates iteration count and is omitted here for notational simplicity.

1.1 Our contributions

Our point of departure in this work is the introduction of preconditioned, iterative solvers for solving eqn. (3). Preconditioning is used to address the ill-conditioning of the matrix $\mathbf{AD}^2\mathbf{A}^\mathsf{T}$. Iterative solvers allow the computation of approximate solutions using only matrix-vector products while avoiding matrix inversion, Cholesky or LU factorizations, etc. A preconditioned formulation of eqn. (3) is:

$$\mathbf{Q}^{-1}\mathbf{AD}^2\mathbf{A}^\mathsf{T}\Delta\mathbf{y} = \mathbf{Q}^{-1}\mathbf{p}, \tag{4}$$

where $\mathbf{Q} \in \mathbb{R}^{m\times m}$ is the preconditioning matrix; $\mathbf{Q}$ should be easily invertible (see (Axelsson and Barker, 1984; Golub and Van Loan, 2013) for background). An alternative yet equivalent formulation of eqn. (4), which is more amenable to theoretical analysis, is

$$\mathbf{Q}^{-1/2}\mathbf{AD}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}\mathbf{z} = \mathbf{Q}^{-1/2}\mathbf{p}, \tag{5}$$

where $\mathbf{z} \in \mathbb{R}^m$ is a vector such that $\Delta\mathbf{y} = \mathbf{Q}^{-1/2}\mathbf{z}$. Note that the matrix in the left-hand side of the above equation is always symmetric, which is not necessarily the case for eqn. (4). We do emphasize that one can use eqn. (4) in the actual implementation of the preconditioned solver; eqn. (5) is much more useful in theoretical analyses.

Recall that we focus on the special case where $\mathbf{A} \in \mathbb{R}^{m\times n}$ has $m \ll n$, i.e., it is a short-and-fat matrix. Our first contribution starts with the design and analysis of a preconditioner for the Conjugate Gradient solver. The preconditioner satisfies, with high probability, the following bound:

$$\frac{2}{2+\zeta} \leq \sigma^2_{\min}(\mathbf{Q}^{-\frac{1}{2}}\mathbf{AD}) \leq \sigma^2_{\max}(\mathbf{Q}^{-\frac{1}{2}}\mathbf{AD}) \leq \frac{2}{2-\zeta}, \tag{6}$$

for some error parameter $\zeta \in [0, 1]$. In the above, $\sigma_{\min}(\cdot)$ and $\sigma_{\max}(\cdot)$ correspond to the smallest and largest singular value of the matrix in parentheses. The above condition says that the preconditioner effectively reduces the condition number of $\mathbf{AD}$ to a constant. We note that the particular form of the lower and upper bounds in eqn. (6) was chosen to simplify our derivations.

RLA matrix-sketching techniques allow us to construct preconditioners for all short-and-fat matrices that satisfy the above inequality *and* can be inverted efficiently. Such constructions go back to the work of Avron et al. (2010); see Section 3 for details on the construction of $\mathbf{Q}$ and its inverse. Importantly, given such a preconditioner, we then prove that the resulting CG iterative solver satisfies

$$\|\mathbf{Q}^{-1/2}\mathbf{AD}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}\tilde{\mathbf{z}}^t - \mathbf{Q}^{-1/2}\mathbf{p}\|_2 \leq \zeta^t\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2. \tag{7}$$

Here $\tilde{\mathbf{z}}^t$ is the approximate solution returned by the CG iterative solver after t iterations. In words, the above inequality states that the *residual* achieved after t iterations of the CG iterative solver drops exponentially fast. Given eqn. (6), we derive eqn. (7) using the monotonic decrease of the preconditioned residual norms of CG. To the best of our knowledge, such monotonicity of the residual error is not known in the CG literature: indeed, it is actually well-known that the residual error of CG may oscillate (Fong and Saunders, 2012),

even in cases where the energy norm of the solution error decreases monotonically. However, we prove that if the preconditioner is sufficiently good, i.e., it satisfies the constraint of eqn. (6), then the residual error decreases monotonically as well, resulting in eqn. (7). It is slightly better than the residual-norm bound derived directly from the energy norm of the solution-error, which is the standard bound for CG. In the later case, the bound in eqn. (7) would have another constant factor involving $\kappa(\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D})$, the condition number of the preconditioned matrix. Using the aforementioned monotonicity of the residual norms, we are able to get rid of that constant factor. In addition, such monotonic decrease of the residual norms can also be of independent interest in CG literature. See Section 3.1 for details.

In addition, we also analyze another popular Krylov subspace-based solver, namely, *Chebyshev iteration* (Barrett et al., 1994; Gutknecht and Röllin, 2002). This method avoids the computation of the inner products which is typically needed for CG or other non-stationary methods. Inner products are communication intensive in parallel or distributed settings, and, as such, are detrimental to the performance in such setups. However, there is a trade-off; in order to avoid the computation of the inner products, it requires adequate knowledge about spectrum of the coefficient matrix $\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}$, which, in our context, is nothing but the condition in eqn. (6). Therefore, given eqn. (6), we prove that Chebyshev iteration also satisfies eqn. (7).

Our second contribution is the analysis of a novel variant of a long-step IPM algorithm proposed by Monteiro and O'Neal (2003). First, we analyze the feasible version of it, which is the one that starts from a strictly feasible point and stays feasible across all the iterations – namely, any point $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k)$ with $(\mathbf{x}^k, \mathbf{s}^k) > \mathbf{0}$ such that such that they are both primal and dual feasible *i.e.*, $\mathbf{A}\mathbf{x}^k = \mathbf{b}$ and $\mathbf{A}^\mathsf{T}\mathbf{y}^k + \mathbf{s}^k = \mathbf{c}$. However, the use of an approximate solver prevents the iterates $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k)$ from satisfying the above primal and dual feasibility constraints *exactly* right after the first iteration of the IPM. In order to account for the error caused by the CG solver and to push the iterates back to the feasibility, Monteiro and O'Neal (2003) introduced a perturbation vector $\mathbf{v}$ which needs to satisfy a certain linear invariant exactly. Again, we use RLA matrix sketching principles to propose an efficient construction for $\mathbf{v}$ that provably satisfies the invariant.

Finally, we combine the above two primitives to prove that Algorithm 2 in Section 4 satisfies the following theorem.

Theorem 1. *Let $0 \leq \epsilon \leq 1$ be an accuracy parameter. Consider the long-step feasible IPM Algorithm 2 (Section 4) that solves eqn. (5) using the iterative solver of Algorithm 1 (Section 3). Assume that the iterative solver runs with accuracy parameter $\zeta = 1/2$ and iteration count $t = \mathcal{O}(\log n)$. Then, with probability at least 0.9, the long-step feasible IPM converges after $\mathcal{O}(n \log 1/\epsilon)$ iterations.*

We note that the constant success probability above is for simplicity of exposition and can be easily amplified using standard techniques. Also, at each iteration of our long-step feasible IPM algorithm, the running time is $\mathcal{O}((\mathsf{nnz}(\mathbf{A}) + m^3) \log n)$. See Section 4 for a detailed discussion of the overall running time. However, finding a strictly feasible initial point is a non-trivial task. Therefore, we also briefly discuss the infeasible version of the long-step IPM algorithm in Section 5.

Our empirical evaluation demonstrates that our algorithm requires an order of magnitude much fewer inner CG iterations than a standard IPM using CG, while producing a comparably accurate solution (see Section 7). In practice, our empirical evaluation also indicates that using a CG solver with our sketching-based preconditioner does not increase the number of (outer) iterations of the infeasible IPM, compared to unpreconditioned CG or a direct linear solver. Furthermore, there are instances where our solver performs much better than non-preconditioned CG in terms of (outer) iteration count.

1.2 Comparison with Related Work

There is a large body of literature on solving LPs using IPMs, thus we only review literature that is immediately relevant to our work. Recall that we solve the normal equations inexactly at each iteration, and develop a method to *correct* for the error incurred. We focus on IPMs that start with a strictly feasible initial point and discuss papers that present related ideas.

The use of an approximate iterative solver for eqn. (3), followed by a correction step to "fix" the approximate solution was proposed by Monteiro and O'Neal (2003) (see our discussion in Section 1.1). We propose efficient, RLA-based approaches to precondition and solve eqn. (3), as well as a novel approach to correct for the approximation error in order to guarantee the convergence of the IPM algorithm. Specifically, Monteiro and O'Neal (2003) propose to solve eqn. (3) using the so-called *maximum weight basis* preconditioner (Resende and Veiga, 1993). However, computing such a preconditioner needs access to a maximal linearly independent set of columns of $\mathbf{AD}$ in each iteration, which is costly, taking $\mathcal{O}(m^2n)$ time in the worst-case. More importantly, while (Monteiro et al., 2004) provides a bound on the condition number of the preconditioned matrix that depends only on properties of $\mathbf{A}$, and is independent of $\mathbf{D}$, this bound might, in general, be very large. In contrast, our bound is a constant and it does not depend on properties of $\mathbf{A}$ or its dimension. In addition, Monteiro and O'Neal (2003) assume a bound on the two-norm of the residual of the preconditioned system, but it is unclear how the proposed preconditioner guarantees such a bound. Similar concerns exist for the construction of the correction vector $\mathbf{v}$ proposed by Monteiro and O'Neal (2003), which our work alleviates. In this context, there is a long list of works on designing efficient preconditioners for solving the linear system at each iteration of IPM. For a more detailed discussion, we refer the interested readers to the survey of (Gondzio, 2012).

In the Theoretical Computer Science community, following the lines of (Karmarkar, 1984), there has been a series of efforts to design faster LP solvers with improved worst-case time complexity. The running time of (Karmarkar, 1984) was improved by (Renegar, 1988) and (Vaidya, 1989) which proposed an algorithm that takes $\widetilde{\mathcal{O}}(n^{2.5+o(1)})$ time. Current state-of-the-art running times involve fast matrix multiplication (a theoretically appealing yet impractical approach). For example, (Cohen et al., 2019) proposed an algorithm that runs in $\widetilde{\mathcal{O}}((n^\omega + n^{2.5-\alpha/2+o(1)} + n^{2+1/6}))$ time, where ω is the exponent of matrix multiplication and α is the dual exponent of matrix multiplication. For $\omega \approx 2.38$ and $\alpha \approx 0.31$ this time complexity boils down to $\widetilde{\mathcal{O}}(n^{\omega+o(1)})$. More recently, (Jiang et al., 2021) reduced the running time of (Cohen et al., 2019) to $\widetilde{\mathcal{O}}((n^\omega + n^{2.5-\alpha/2+o(1)} + n^{2+1/18}))$, which further reduces the gap between matrix multiplication and solving LPs.

Work by (Lee et al., 2019; Brand, 2020; Song and Yu, 2021) achieved the same running time as (Cohen et al., 2019), using the same so-called *lazy update* framework of (Cohen et al., 2019). However, there are subtle differences between these works in terms of the underlying sketching and sampling techniques, as well as on approaches that achieve fast queries for the so-called *projection maintenance* data structure that handles infeasibilities over iterations. For example, while the work of (Cohen et al., 2019) involves a non-oblivious sampling scheme whose sampling set and size changes over iterations, (Song and Yu, 2021) utilizes oblivious sketching through an iterative framework to approximate the central path. On the other hand, while (Cohen et al., 2019; Brand, 2020; Song and Yu, 2021) solve the linear system exactly, other works only maintain infeasible updates in each iteration. Similarly, while (Song and Yu, 2021) can leverage sparse embeddings, most of the aforementioned works (including (Lee et al., 2019)) require the usage of dense sketching matrices, which could hinder the sparsity structure of the original linear program.

All the aforementioned solvers are designed for the $m \approx n$ setting. If $\mathbf{A}$ is dense and rectangular with $n \gg m$, (Brand et al., 2020) provides the theoretically fastest solver, with an iteration complexity of $\widetilde{\mathcal{O}}(\sqrt{m})$ and a total time complexity equal to $\widetilde{\mathcal{O}}\left(mn + m^3\right)$. For sparse rectangular matrices, the best per iteration complexities are given by $\widetilde{\mathcal{O}}\left(\text{nnz}(\mathbf{A}) + m^\omega\right)$ (Lee and Sidford, 2014) and $\widetilde{\mathcal{O}}(\mathsf{nnz}(\mathbf{A}) + m^2)$ (see (Lee and Sidford, 2015)); both algorithms have iteration complexity $\widetilde{\mathcal{O}}(\sqrt{m})$. Overall, we note that (Lee and Sidford, 2014, 2015; Lee et al., 2019; Brand et al., 2020; Cohen et al., 2021; Song and Yu, 2021; Jiang et al., 2021) proposed and analyzed theoretically ground-breaking algorithms for LPs based on novel tools such as the so-called *projection maintenance*, *inverse maintenance*, *fast matrix multiplication*, etc. for accelerating the linear system solvers in IPMs. In contrast, our paper differs from all these aforementioned TCS works in at least the following three directions:

- First, all these aforementioned approaches are primarily focused on theoretically fast but practically inefficient short-step path following methods, where the iterates are constrained within a narrow, restrictive neighborhood of the central path in the interior of the feasible region. Therefore, the algorithms based on short-step IPMs do not have much room to maneuver and the progress that they make in each iteration is limited, at least in practice[3]. Some of the recent TCS works (for example, Cohen et al. (2019); Song and Yu (2021) etc.) rely on a stochastic version of the short step central path method, in which the neighborhood of the central path is slightly wider than that of the traditional short-step IPMs. This is still considered to be quite restrictive, as it needs all the pairwise products $x_i s_i$ to be close to to the duality measure μ (more precisely, it needs $0.9\mu \leq x_i s_i \leq 1.1\mu$ for all i), which is not a very practical idea in general. On the other hand, our algorithm is based on long-step path following IPMs which explore a much larger neighborhood around the central path and offer significant flexibility in each iteration. As a result, the iterates can take much longer steps towards optimality. Long-step IPMs are known to be more efficient in practice compared to short-step IPMs, despite the fact that they exhibit worst-case iteration complexity $\mathcal{O}(n)$.

3. Theoretically, the worst-case iteration complexity of short-step central path methods is given by $\widetilde{\mathcal{O}}(\sqrt{n})$, which is the best complexity bound known for IPMs.

- Second, the theoretical benefits of the aforementioned TCS works rely on techniques such as *projection maintenance* or *inverse maintenance*, where one needs to preserve the orthogonal projection matrix $\mathbf{D}\mathbf{A}^\top(\mathbf{A}\mathbf{D}^2\mathbf{A}^\top)^{-1}\mathbf{A}\mathbf{D} \in \mathbb{R}^{n\times n}$ or $(\mathbf{A}\mathbf{D}^2\mathbf{A}^\top)^{-1} \in \mathbb{R}^{n\times n}$ in order to solve the linear system at each iteration of the IPM. The idea of maintaining such projection matrices depends on the assumption that the matrix $\mathbf{D}$ does not change much from one iteration to the next. Thus, one only needs to compute the matrix $(\mathbf{A}\mathbf{D}^2\mathbf{A}^\top)^{-1}$ a few times, which is also known as lazy updating. In addition, if $\mathbf{D}$ only changes in a few of its diagonal entries, one can further use the idea of low-rank updating to maintain the above projection matrix via the Sherman–Morrison–Woodbury (SMW) formula. However, due to large constant factors involved in the lazy updates framework and the numerical instability of the SMW formula, the notion of *projection maintenance* is generally inefficient in practice. In contrast, our methods do not depend on any of the aforementioned techniques. Instead, we use randomized preconditioners combined with iterative solvers to approximately solve the linear system. We also propose a computationally efficient way to correct for the error caused by the solver. As a result, our method is much more relevant in practice and, when $n \gg m$, the per iteration cost of our algorithm is $\widetilde{\mathcal{O}}(\mathsf{nnz}(\mathbf{A}) + m^3)$

- Third, all the aforementioned papers involve fast matrix multiplication, which is a theoretically relevant yet practically inefficient tool. To the best of our knowledge, there are no in- or out-of-core implementations of such algorithms that work better than traditional matrix multiplication approaches and it seems unlikely that such techniques will become practically relevant in the near future. Our methods leverage standard matrix multiplication routines and in Section 7 we show how an implementation of our approach offers advantages over existing, practically relevant approaches.

Another line of research in the Theoretical Computer Science literature that is very close to our work is (Daitch and Spielman, 2008), who presented an IPM that uses an approximate solver in each iteration. However, their accuracy guarantee is in terms of the final objective value, which is different from ours. More importantly, (Daitch and Spielman, 2008) focuses on *short-step* IPMs, whereas our approach is a *long-step* algorithm that works for both feasible and infeasible starting points. Finally, the approximate solver proposed by (Daitch and Spielman, 2008) works only for the special case of input matrices that correspond to graph Laplacians, following the lines of (Spielman and Teng, 2004, 2014). In this context, we note that there are also other relevant works including (Madry, 2013, 2016; Cohen et al., 2017) that used IPM-based algorithms with approximate solvers for various combinatorial optimization problems on graphs. However, similar to (Spielman and Teng, 2004), the aforementioned papers also focus on short-step IPMs and the linear systems associated with them are either Laplacian or symmetric diagonally dominant (SDD).

Another relevant line of research is the work by Cui et al. (2019), which proposed solving eqn. (3) using preconditioned Krylov subspace methods, including variants of *generalized minimum residual* (GMRES) or CG methods. Indeed, Cui et al. (2019) conducted extensive numerical experiments on LP problems taken from standard benchmark libraries, but did not provide any theoretical guarantees. Beyond experiments, the methods of (Cui et al., 2019) primarily differ from ours in the way they used preconditioning. Instead of applying the preconditioner explicitly, their empirical evaluations rely on an implicit preconditioning

technique called *(stationary) inner-iterations preconditioning* (Morikuni and Hayami, 2013, 2015). As the name suggests, the key idea is to use another iterative method as a preconditioner within the linear system that needs to be solved at each iteration of the IPM. From a theoretical perspective, it is not clear how big or small the resulting condition number of their preconditioned system could be; moreover, there are two hyperparameters associated with their preconditioner which need to be tuned optimally and there is no theoretical guideline on how that can be done.

From a matrix-sketching perspective, our work was also partially motivated by (Chowdhury et al., 2018), which presented an iterative, sketching-based algorithm to solve under-constrained ridge regression problems, but did not address how to make use of such approaches in an IPM-based framework, as we do here. We refer the reader to the surveys (Woodruff, 2014; Drineas and Mahoney, 2018; Mahoney, 2011; Drineas and Mahoney, 2016; Martinsson and Tropp, 2020) for more background on Randomized Linear Algebra and regression solvers. In the context of deep neural networks (DNNs), (van den Brand et al., 2021) recently presented an iterative method to speed up the training of overparametrized DNNs by a similar type of randomized preconditioner as ours; but the algorithm of (van den Brand et al., 2021) neither exploited the sparsity in the data, nor they had to correct for the error caused by the inexact solver, as we do here. In another work, (Avron et al., 2017) also proposed a similar sketching-based preconditioning technique. However, their efforts broadly revolved around speeding up and scaling *kernel ridge regression*. (Pilanci and Wainwright, 2017) proposed the so-called *Newton sketch* to construct an approximate Hessian matrix for more general convex problems, of which LP is a special case. Nevertheless, based on their local convergence guarantee for the sketched Newton updates, their paper only derived the underlying iteration complexity of the IPM (see, for example, Theorem 4.3 of (Pilanci and Wainwright, 2017)). It is not clear how to use their approach to bound the number of inner iterations and, as a result, deriving the per iteration cost of their algorithm is not straightforward. Moreover, their convergence guarantees for the IPM is with respect to the objective value and not in terms of the duality measure. Finally, the Newton-sketch of (Pilanci and Wainwright, 2017) does not exploit the sparsity of the data, whereas the running time of our algorithm depends on the sparsity of $\mathbf{A}$. We also note that (Vu et al., 2018) proposed a probabilistic algorithm to solve LPs approximately using a random projection reduced feature space. A possible drawback of this work is that the approximate solution is infeasible with respect to the original region.

On the empirical side, there are prior implementations of $\widetilde{\mathcal{O}}(\mathsf{nnz}(\mathbf{A}) + \text{poly}(m))$ solvers for speeding various ML applications including ordinary least square regression (Clarkson and Woodruff, 2017; Cormode and Dickens, 2019), general ℓ_p-regression (Yang et al., 2017), ridge regression (Chen et al., 2015; Chowdhury et al., 2018), Fisher linear discriminant analysis (Ye et al., 2017; Chowdhury et al., 2019), more general Newton updates (Dahiya et al., 2018) and many more. However, to the best of our knowledge, there are no such implementations in the context of general linear programming problems, perhaps with the exception of ℓ_1-regression. As discussed before, prior work on short-step IPM for LP that came up with per iteration cost $\widetilde{\mathcal{O}}(\mathsf{nnz}(\mathbf{A}) + \text{poly}(m))$ was due to (Lee and Sidford, 2014, 2015), but there is no empirical evaluation because of its heavy reliance on various theoretical tools such as inverse maintenance, fast matrix multiplication etc.

Finally, in addition to IPMs, LPs can be solved using the Simplex method. In commercial LP packages like Gurobi, both methods are often used in conjunction. For example, multiple solvers are run on multiple threads simultaneously and the one that is finishes first is chosen. Alternatively, an IPM is used initially to get close to the optimal solution and then the simplex algorithm is used to improve the solution (Bixby et al., 1992; Glavelis et al., 2018). Thus, developing efficient IPMs is vital for solving LPs and provides a crucial building block in commerical packages like Gurobi.

2. Notation and Background

$\mathbf{A}, \mathbf{B}, \ldots$ denote matrices and $\mathbf{a}, \mathbf{b}, \ldots$ denote vectors. For vector $\mathbf{a}$, $\|\mathbf{a}\|_2$ denotes its Euclidean norm; for a matrix $\mathbf{A}$, $\|\mathbf{A}\|_2$ denotes its spectral norm and $\|\mathbf{A}\|_F$ denotes its Frobenius norm. We use $\mathbf{0}$ to denote a null vector or null matrix, dependent upon context, and $\mathbf{1}$ to denote the all-ones vector. For any matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$ with $m \leq n$ of rank m a thin Singular Value Decomposition (SVD) is a product $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$, with $\mathbf{U} \in \mathbb{R}^{m\times m}$ (the matrix of the left singular vectors), $\mathbf{V} \in \mathbb{R}^{n\times m}$(the matrix of the top-$m$ right singular vectors), and $\mathbf{\Sigma} \in \mathbb{R}^{m\times m}$ a diagonal matrix whose entries are equal to the singular values of $\mathbf{X}$. We use $\sigma_i(\cdot)$ to denote the i-th singular value of the matrix in parentheses. For any two vectors $\mathbf{a} = (a_1, \ldots, a_\ell)^\mathsf{T}$ and $\mathbf{b} = (b_1, \ldots, b_\ell)^\mathsf{T}$, we denote $\mathbf{a} \circ \mathbf{b} = (a_1 b_1, \ldots, a_\ell b_\ell)^\mathsf{T}$. For any two symmetric positive semidefinite (resp. positive definite) matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ of appropriate dimensions, $\mathbf{A}_1 \preccurlyeq \mathbf{A}_2$ ($\mathbf{A}_1 \prec \mathbf{A}_2$) denotes that $\mathbf{A}_2 - \mathbf{A}_1$ is positive semidefinite (resp. positive definite). For any vector $\mathbf{a} \in \mathbb{R}^n$ its ℓ_∞ norm is defined as $\|\mathbf{a}\|_\infty = \max_i |a_i|$. We extensively use the following standard inequality to prove several results in the paper:

$$\left|\frac{\mathbf{a}^\mathsf{T}\mathbf{1}_n}{n}\right| \leq \|\mathbf{a}\|_\infty \leq \|\mathbf{a}\|_2. \tag{8}$$

We now briefly discuss a result on matrix sketching (Cohen et al., 2016; Cohen, 2016) that is particularly useful in our theoretical analyses. In our parlance, Cohen et al. (2016) proved that, for any matrix $\mathbf{Z} \in \mathbb{R}^{m\times n}$, there exists a sketching matrix $\mathbf{W} \in \mathbb{R}^{n\times w}$ such that

$$\left\|\mathbf{Z}\mathbf{W}\mathbf{W}^\mathsf{T}\mathbf{Z}^\mathsf{T} - \mathbf{Z}\mathbf{Z}^\mathsf{T}\right\|_2 \leq \frac{\zeta}{4}\Big(\|\mathbf{Z}\|_2^2 + \frac{\|\mathbf{Z}\|_F^2}{r}\Big) \tag{9}$$

holds with probability at least $1 - \delta$ for any $r \geq 1$. Here $\zeta \in [0, 1]$ is a (constant) accuracy parameter. Ignoring constant terms, $w = \mathcal{O}(r \log(r/\delta))$; $\mathbf{W}$ has $\mathcal{O}(\log(r/\delta))$ non-zero entries per row; and the product $\mathbf{Z}\mathbf{W}$ can be computed in time $\mathcal{O}(\log(r/\delta) \cdot \mathsf{nnz}(\mathbf{Z}))$.

3. Preconditioned Iterative Solver

In this section, we discuss the computation of the preconditioner $\mathbf{Q}$ (and its inverse), followed by a discussion on how such a preconditioner can be used to satisfy eqns. (6) and (7).

Algorithm 1 Solving eqn. (5) via CG or Chebyshev iteration

Input: $\mathbf{AD} \in \mathbb{R}^{m\times n}$, $\mathbf{p} \in \mathbb{R}^m$, sketching matrix $\mathbf{W} \in \mathbb{R}^{n\times w}$, iteration count t;

1: Compute $\mathbf{ADW}$ and its SVD: let $\mathbf{U_Q} \in \mathbb{R}^{m\times m}$ be the matrix of its left singular vectors and let $\mathbf{\Sigma_Q^{1/2}} \in \mathbb{R}^{m\times m}$ be the matrix of its singular values;

2: Compute $\mathbf{Q}^{-1/2} = \mathbf{U_Q}\mathbf{\Sigma_Q}^{-1/2}\mathbf{U_Q^\mathsf{T}}$;

3: Initialize $\tilde{\mathbf{z}}^0 \leftarrow \mathbf{0}_m$ and run standard CG or Chebyshev iteration on the preconditioned system of eqn. (5) for t iterations;

Output: return $\tilde{\mathbf{z}}^t$;

Algorithm 1 takes as input the sketching matrix $\mathbf{W} \in \mathbb{R}^{n\times w}$, which we construct as discussed in Section 2. Our preconditioner $\mathbf{Q}$ is equal to

$$\mathbf{Q} = \mathbf{ADWW}^\mathsf{T}\mathbf{DA}^\mathsf{T}. \tag{10}$$

Notice that we only need to compute $\mathbf{Q}^{-1/2}$ in order to use it to solve eqn. (5). Towards that end, we first compute the sketched matrix $\mathbf{ADW} \in \mathbb{R}^{m\times w}$. Then, we compute the SVD of the matrix $\mathbf{ADW}$: let $\mathbf{U_Q}$ be the matrix of its left singular vectors and let $\mathbf{\Sigma_Q^{1/2}}$ be the matrix of its singular values. Notice that the left (and right) singular vectors of $\mathbf{Q}^{-1/2}$ are equal to $\mathbf{U_Q}$ and its singular values are equal to $\mathbf{\Sigma_Q}^{-1/2}$. Therefore, $\mathbf{Q}^{-1/2} = \mathbf{U_Q}\mathbf{\Sigma_Q}^{-1/2}\mathbf{U_Q^\mathsf{T}}$.

Let $\mathbf{AD} = \mathbf{U\Sigma V}^\mathsf{T}$ be the thin SVD representation of $\mathbf{AD}$. We apply the results of (Cohen et al., 2016) (see Section 2) to the matrix $\mathbf{Z} = \mathbf{V}^\mathsf{T} \in \mathbb{R}^{m\times n}$ with $r = m$ to get that, with probability at least $1-\delta$,

$$\left\|\mathbf{V}^\mathsf{T}\mathbf{WW}^\mathsf{T}\mathbf{V} - \mathbf{I}_m\right\|_2 \le \frac{\zeta}{4}\Big(\|\mathbf{V}\|_2^2 + \frac{\|\mathbf{V}\|_F^2}{m}\Big) \le \frac{\zeta}{2}. \tag{11}$$

In the above we used $\|\mathbf{V}\|_2 = 1$ and $\|\mathbf{V}\|_F^2 = m$. The running time needed to compute the sketch $\mathbf{ADW}$ is equal to (ignoring constant factors) $\mathcal{O}(\mathsf{nnz}(\mathbf{A})\cdot\log(m/\delta))$. Note that $\mathsf{nnz}(\mathbf{AD}) = \mathsf{nnz}(\mathbf{A})$. The cost of computing the SVD of $\mathbf{ADW}$ (and therefore $\mathbf{Q}^{-1/2}$) is $\mathcal{O}(m^3\log(m/\delta))$. Overall, computing $\mathbf{Q}^{-1/2}$ can be done in time

$$\mathcal{O}(\mathsf{nnz}(\mathbf{A})\cdot\log(m/\delta) + m^3\log(m/\delta)). \tag{12}$$

Given these results, we now discuss how to satisfy eqns. (6) and (7) using the sketching matrix $\mathbf{W}$. We start with the following bound, which is relatively straightforward given prior RLA work.

Lemma 2. *If the sketching matrix* $\mathbf{W}$ *satisfies eqn.* (11)*, then, for all* $i = 1\dots m$*,*

$$(1+\zeta/2)^{-1} \le \sigma_i^2(\mathbf{Q}^{-1/2}\mathbf{AD}) \le (1-\zeta/2)^{-1}.$$

Proof Consider the condition of eqn. (11):

$$\|\mathbf{V}^\mathsf{T}\mathbf{WW}^\mathsf{T}\mathbf{V} - \mathbf{I}_m\|_2 \le \frac{\zeta}{2} \;\Leftrightarrow\; -\frac{\zeta}{2}\mathbf{I}_m \preccurlyeq \mathbf{V}^\mathsf{T}\mathbf{WW}^\mathsf{T}\mathbf{V} - \mathbf{I}_m \preccurlyeq \frac{\zeta}{2}\mathbf{I}_m \tag{13}$$

$$\Leftrightarrow\; -\frac{\zeta}{2}\mathbf{AD}^2\mathbf{A}^\mathsf{T} \preccurlyeq \mathbf{ADWW}^\mathsf{T}\mathbf{DA}^\mathsf{T} - \mathbf{AD}^2\mathbf{A}^\mathsf{T} \preccurlyeq \frac{\zeta}{2}\mathbf{AD}^2\mathbf{A}^\mathsf{T} \tag{14}$$

$$\Leftrightarrow \quad \left(1-\frac{\zeta}{2}\right)\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T} \preccurlyeq \underbrace{\mathbf{A}\mathbf{D}\mathbf{W}\mathbf{W}^\mathsf{T}\mathbf{D}\mathbf{A}^\mathsf{T}}_{\mathbf{Q}} \preccurlyeq \left(1+\frac{\zeta}{2}\right)\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}. \tag{15}$$

We obtain eqn. (14) by pre- and post-multiplying the previous inequality by $\mathbf{U}\mathbf{\Sigma}$ and $\mathbf{\Sigma}\mathbf{U}^\mathsf{T}$ respectively and using the facts that $\mathbf{A}\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$ and $\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T} = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\mathsf{T}$. Also, from eqn. (13), note that all the eigenvalues of $\mathbf{V}^\mathsf{T}\mathbf{W}\mathbf{W}^\mathsf{T}\mathbf{V}$ lie between $(1-\frac{\zeta}{2})$ and $(1+\frac{\zeta}{2})$ and thus $\text{rank}(\mathbf{V}^\mathsf{T}\mathbf{W}) = m$. Therefore, $\text{rank}(\mathbf{A}\mathbf{D}\mathbf{W}) = \text{rank}(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}\mathbf{W}) = m$, as $\mathbf{U}\mathbf{\Sigma}$ is non-singular and we know that the rank of a matrix remains unaltered by pre- or post-multiplying it by a non-singular matrix. So, we have $\text{rank}(\mathbf{Q}) = m$; in words $\mathbf{Q}$ has full rank. Therefore, all the diagonal entries of $\mathbf{\Sigma}_\mathbf{Q}$ are positive and $\mathbf{Q}^{-1/2}\mathbf{Q}\mathbf{Q}^{-1/2} = \mathbf{I}_m$.

Using the above arguments, pre- and post- multiplying eqn. (15) by $\mathbf{Q}^{-1/2}$, we get

$$\begin{aligned}&\left(1-\frac{\zeta}{2}\right)\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2} \preccurlyeq \mathbf{I}_m \preccurlyeq \left(1+\frac{\zeta}{2}\right)\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}\\ \Rightarrow\ &\left(1+\frac{\zeta}{2}\right)^{-1}\mathbf{I}_m \preccurlyeq \mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2} \preccurlyeq \left(1-\frac{\zeta}{2}\right)^{-1}\mathbf{I}_m. \end{aligned}\tag{16}$$

Eqn. (16) implies that all the eigenvalues of $\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}$ are bounded between $\left(1+\frac{\zeta}{2}\right)^{-1}$ and $\left(1-\frac{\zeta}{2}\right)^{-1}$, which concludes the proof of the lemma. ■

The above lemma directly implies eqn. (6). We now proceed to show that the above construction for $\mathbf{Q}^{-1/2}$, when combined with the conjugate gradient solver or Chebyshev iteration to solve eqn. (5), indeed satisfies eqn. (7).

3.1 Conjugate Gradient Solver

As already mentioned in Section 1.1, we derive eqn. (7) using the monotonicity property of CG resudual norms. It is known that even if the energy norm of the error of the approximate solution decreases monotonically, the norms of the CG residuals may oscillate. Interestingly, we can combine a result on the residuals of CG from (Bouyouli et al., 2009) with Lemma 2 to prove that in our setting the norms of the CG residuals also decrease monotonically[4]. We do note that in prior work most of the convergence guarantees for CG focus on the error of the approximate solution, from which, directly deriving eqn. (7) induces a constant factor in terms of the condition number of the preconditioned matrix. Here, we are able to avoid that constant factor by deriving and using the aforementioned monotonicity of the CG residuals.

Let $\tilde{\mathbf{f}}^{(j)}$ be the residual at the j-th iteration of the CG algorithm:

$$\tilde{\mathbf{f}}^{(j)} = \mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}\tilde{\mathbf{z}}^j - \mathbf{Q}^{-1/2}\mathbf{p}.$$

Recall from Algorithm 1 that $\tilde{\mathbf{z}}^0 = \mathbf{0}$ and thus $\tilde{\mathbf{f}}^{(0)} = -\mathbf{Q}^{-1/2}\mathbf{p}$. In our parlance, Theorem 8 of (Bouyouli et al., 2009) proved the following bound.

Lemma 3 (Theorem 8 of (Bouyouli et al., 2009)). *Let $\tilde{\mathbf{f}}^{(j-1)}$ and $\tilde{\mathbf{f}}^{(j)}$ be the residuals obtained by the CG solver at steps $j-1$ and j. Then,*

$$\|\tilde{\mathbf{f}}^{(j)}\|_2 \leq \frac{\kappa^2(\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D})-1}{2}\|\tilde{\mathbf{f}}^{(j-1)}\|_2,$$

4. See Chapter 9 of (Luenberger and Ye, 2008) for a detailed overview of CG.

where $\kappa(\mathbf{Q}^{-1/2}\mathbf{AD})$ *is the condition number of* $\mathbf{Q}^{-1/2}\mathbf{AD}$.

Satisfying eqn. (7)**.** From Lemma 2, we get

$$\kappa^2(\mathbf{Q}^{-1/2}\mathbf{AD}) = \frac{\sigma^2_{\max}(\mathbf{Q}^{-1/2}\mathbf{AD})}{\sigma^2_{\min}(\mathbf{Q}^{-1/2}\mathbf{AD})} \leq \frac{1+\zeta/2}{1-\zeta/2}. \tag{17}$$

Combining eqn. (17) with Lemma 3,

$$\|\tilde{\mathbf{f}}^{(j)}\|_2 \leq \frac{\frac{1+\zeta/2}{1-\zeta/2}-1}{2}\|\tilde{\mathbf{f}}^{(j-1)}\|_2 = \frac{\zeta}{2-\zeta}\|\tilde{\mathbf{f}}^{(j-1)}\|_2 \leq \zeta\|\tilde{\mathbf{f}}^{(j-1)}\|_2\,, \tag{18}$$

where the last inequality follows from $\zeta \leq 1$. Applying eqn. (18) recursively, we get

$$\|\tilde{\mathbf{f}}^{(t)}\|_2 \leq \zeta\|\tilde{\mathbf{f}}^{(t-1)}\|_2 \leq \cdots \leq \zeta^t\|\tilde{\mathbf{f}}^{(0)}\|_2 = \zeta^t\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2\,,$$

which proves the condition of eqn. (7).

We remark that one can consider using MINRES (Paige and Saunders, 1975) instead of CG. Our results hinges on bounding the two-norm of the residual. MINRES finds, at each iteration, the optimal vector with respect the two-norm of the residual inside the same Krylov subspace of CG for the corresponding iteration. Thus, the bound we prove for CG applies to MINRES as well.

3.2 Chebyshev Iteration

Now, we show that we could potentially replace CG with Chebyshev iteration (see Algorithm 1 of (Gutknecht and Röllin, 2002)) in the step (3) of Algorithm 1. As already discussed in Section 1.1, the only requirement Chebyshev iteration needs is to have an upper bound and a lower bound for the singular values of $\mathbf{Q}^{-1/2}\mathbf{AD}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}$, which we already have in the form of Lemma 2. Therefore, all we need to show is that the sketching matrix $\mathbf{W}$ satisfies eqn. (7) using Chebyshev iteration. For this, we state the following result from (Gutknecht, 2008), which is instrumental in proving eqn. (7).

Lemma 4 (Theorem 1.6.2 of (Gutknecht, 2008))**.** *The residual norm reduction of the Chebyshev iteration, when applied to an symmetric positive definite (SPD) system whose condition number is upper bounded by* $\mathcal{U}$*, is bounded according to*

$$\frac{\|\tilde{\mathbf{f}}^{(t)}\|_2}{\|\tilde{\mathbf{f}}^{(0)}\|_2} \leq 2\left[\left(\frac{\sqrt{\mathcal{U}}+1}{\sqrt{\mathcal{U}}-1}\right)^t + \left(\frac{\sqrt{\mathcal{U}}-1}{\sqrt{\mathcal{U}}+1}\right)^t\right]^{-1} \tag{19}$$

Satisfying eqn. (7)**.** From Lemma 2, we directly have $\mathcal{U} = \frac{2+\zeta}{2-\zeta}$. Note that for $t = 0$ and starting from $\tilde{\mathbf{z}}^0 = \mathbf{0}$ (*i.e.,* $\tilde{\mathbf{f}}^{(0)} = \mathbf{Q}^{-1/2}\mathbf{p}$), we directly get eqn. (7) from eqn. (19). Therefore, we only show that eqn. (7) is satisfied for $t \geq 1$. We already have $\tilde{\mathbf{f}}^{(0)} = \mathbf{Q}^{-1/2}\mathbf{p}$ and letting $a = \left(\frac{\sqrt{\mathcal{U}}-1}{\sqrt{\mathcal{U}}+1}\right)^t$, we rewrite eqn. (19) as follows

$$\|\tilde{\mathbf{f}}^{(t)}\|_2 \leq \frac{2}{a+\frac{1}{a}}\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 = \frac{2\,a}{(a^2+1)}\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 \leq 2\,a\,\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2\,, \tag{20}$$

where the last inequality in eqn. (20) holds as $a^2 > 0$. Now, we'll work on the bound in eqn. (20). Putting back $a = \left(\frac{\sqrt{\mathcal{U}}-1}{\sqrt{\mathcal{U}}+1}\right)^t$ and $\mathcal{U} = \frac{2+\zeta}{2-\zeta}$, we rewrite eqn. (20) as

$$\begin{aligned}
\|\tilde{\mathbf{f}}^{(t)}\|_2 &\leq 2\left(\frac{\sqrt{\mathcal{U}}-1}{\sqrt{\mathcal{U}}+1}\right)^t \|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 = 2\left(\frac{\sqrt{\frac{2+\zeta}{2-\zeta}}-1}{\sqrt{\frac{2+\zeta}{2-\zeta}}+1}\right)^t \|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 \\
&= 2\left(\frac{\sqrt{2+\zeta}-\sqrt{2-\zeta}}{\sqrt{2+\zeta}+\sqrt{2-\zeta}}\right)^t \|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 = 2\left(\frac{2\zeta}{\left(\sqrt{2+\zeta}+\sqrt{2-\zeta}\right)^2}\right)^t \|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 \\
&= 2\left(\frac{2\zeta}{4\left(1+\sqrt{1-(\zeta/2)^2}\right)}\right)^t \|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 = \frac{\zeta^t}{2^{t-1}\left(1+\sqrt{1-(\zeta/2)^2}\right)^t}\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 \\
&\leq \zeta^t \|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 ,
\end{aligned}$$

where the last inequality holds as $t \geq 1$ and the denominator is greater than unity. This establishes eqn. (7).

4. The Feasible IPM algorithm

In order to avoid spurious solutions, primal-dual path-following IPMs bias the search direction towards the *central path* and restrict the iterates to a neighborhood of the central path. This search is controlled by the *centering parameter* $\sigma \in [0, 1]$. At each iteration, given the current feasible solution $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k)$, a standard feasible IPM obtains the search direction $(\Delta\mathbf{x}^k, \Delta\mathbf{y}^k, \Delta\mathbf{s}^k)$ by solving the following system of linear equations:

$$\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\Delta\mathbf{y}^k = \mathbf{p}^k , \tag{21a}$$

$$\Delta\mathbf{s}^k = -\mathbf{A}^\mathsf{T}\Delta\mathbf{y}^k , \tag{21b}$$

$$\Delta\mathbf{x}^k = -\mathbf{x}^k + \sigma\mu_k\mathbf{S}^{-1}\mathbf{1}_n - \mathbf{D}^2\Delta\mathbf{s}^k. \tag{21c}$$

Here $\mathbf{D}$ and $\mathbf{S}$ are computed given the current iterate ($\mathbf{x}^k$ and $\mathbf{s}^k$); we skip the indices on $\mathbf{D}$ and $\mathbf{S}$ for notational simplicity. After solving the above system, the feasible IPM Algorithm 2 proceeds by computing a step-size $\bar{\alpha}$ to return:

$$(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}) = (\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k) + \bar{\alpha}(\Delta\mathbf{x}^k, \Delta\mathbf{y}^k, \Delta\mathbf{s}^k). \tag{22}$$

In the above linear system in eqn. (21), we also use *duality measure* $\mu_k = {\mathbf{x}^k}^\mathsf{T}\mathbf{s}^k/n$ and the vector

$$\mathbf{p}^k = -\sigma\mu_k\mathbf{A}\mathbf{S}^{-1}\mathbf{1}_n + \mathbf{A}\mathbf{x}^k. \tag{23}$$

Given $\Delta\mathbf{y}^k$ from eqn. (21a), $\Delta\mathbf{s}^k$ and $\Delta\mathbf{x}^k$ are easy to compute from eqns. (21b) and (21c), as they only involve matrix-vector products. However, since we use Algorithm 1 to solve eqn. (21a) approximately using the sketching-based preconditioned solver, the iterates $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k)$ do not satisfy the primal and dual constraints exactly.

For notational simplicity, we now drop the dependency of vectors and scalars on the iteration counter k. Let $\hat{\Delta\mathbf{y}} = \mathbf{Q}^{-1/2}\tilde{\mathbf{z}}^t$ be the approximate solution to eqn. (21a). In order to account for the loss of accuracy due to the approximate solver, we compute $\hat{\Delta\mathbf{x}}$ as follows:

$$\hat{\Delta\mathbf{x}} = -\mathbf{x} + \sigma\mu\mathbf{S}^{-1}\mathbf{1}_n - \mathbf{D}^2\hat{\Delta\mathbf{s}} - \mathbf{S}^{-1}\mathbf{v}. \tag{24}$$

Here $\mathbf{v} \in \mathbb{R}^n$ is a perturbation vector that needs to exactly satisfy the following invariant at each iteration of the feasible IPM:

$$\mathbf{A}\mathbf{S}^{-1}\mathbf{v} = \mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y} - \mathbf{p}\,. \tag{25}$$

We note that the computation of $\hat{\Delta}\mathbf{s}$ is still done using, essentially, eqn. (21b), namely

$$\hat{\Delta}\mathbf{s}^k = \; -\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y}^k. \tag{26}$$

At each iteration of the IPM, if $\mathbf{v}$ satisfies eqn. (25), then it can be shown that the primal and dual feasibility constraints are satisfied exactly.

Construction of v. There are many choices for $\mathbf{v}$ satisfying eqn. (25). Intuitively, we would expect the approximation error due to the solver to be reasonably small. Therefore, to prove convergence, it is desirable for $\mathbf{v}$ to have a small norm and hence a natural choice is

$$\mathbf{v} = (\mathbf{A}\mathbf{S}^{-1})^\dagger(\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y} - \mathbf{p})\,.$$

The aforementioned choice of $\mathbf{v}$ has a clear geometric interpretation: it not only ensures that $\mathbf{A}\mathbf{S}^{-1}\mathbf{v}$ is a Euclidean projection of the infeasible solution onto the column space of $\mathbf{A}\mathbf{D}$, but it is also the minimum norm least squares solution and satisfies the invariant in eqn. (25) exactly. However, computing $\mathbf{v}$ such a way is expensive, as it involves the evaluation of the pseudoinverse of $\mathbf{A}\mathbf{S}^{-1}$, which is expensive, taking time $\mathcal{O}(m^2n)$. Instead, we propose to construct $\mathbf{v}$ using the sketching matrix $\mathbf{W}$ of Section 2. More precisely, we construct the perturbation vector

$$\mathbf{v} = (\mathbf{X}\mathbf{S})^{1/2}\mathbf{W}(\mathbf{A}\mathbf{D}\mathbf{W})^\dagger\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y} - \mathbf{p}). \tag{27}$$

Similar to the minimum-norm solution mentioned above, our sketching based solution in eqn. (27) also guarantees that $\mathbf{A}\mathbf{S}^{-1}\mathbf{v}$ is a projection of the "infeasibility" vector $\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y}$ $-\mathbf{p}$ onto the column space of $\mathbf{A}\mathbf{D}\mathbf{W}$ (which is identical to the column space of $\mathbf{A}\mathbf{D}$) and satisfies eqn. (25) exactly (see Lemma 5 below). The computation of our proposed $\mathbf{v}$ is dominated by the cost of computing $(\mathbf{A}\mathbf{D}\mathbf{W})^\dagger$, which can be done much more efficiently as discussed in Section 3. Actually, we do not even need to compute it while evaluating $\mathbf{v}$, since we have already computed it during the construction of $\mathbf{Q}^{-1/2}$ in Algorithm 1. Finally, in Lemma 8, we showed that using our choice of $\mathbf{v}$, $\|\mathbf{v}\|_2$ remains small enough (with a few iterations of the iterative solver), which essentially leads to the convergence of the IPM.

Lemma 5. *Let $\mathbf{W} \in \mathbb{R}^{n\times w}$ be the sketching matrix of Section 2 and $\mathbf{v}$ be the perturbation vector of eqn. (27). Then, with probability at least $1-\delta$,* $\mathrm{rank}(\mathbf{A}\mathbf{D}\mathbf{W}) = m$ *and $\mathbf{v}$ satisfies eqn.* (25)*.*

Proof Let $\mathbf{A}\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$ be the thin SVD representation of $\mathbf{A}\mathbf{D}$. We use the exact same $\mathbf{W}$ as discussed in Section 3. Therefore, eqn. (11) holds with probability $1-\delta$ and it directly follows from the proof of Lemma 2 that $\mathrm{rank}(\mathbf{A}\mathbf{D}\mathbf{W}) = m$. Recall that $\mathbf{A}\mathbf{D}\mathbf{W}$ has full *row-rank* and thus $\mathbf{A}\mathbf{D}\mathbf{W}\,(\mathbf{A}\mathbf{D}\mathbf{W})^\dagger = \mathbf{I}_m$. Therefore, taking $\mathbf{v} = (\mathbf{X}\mathbf{S})^{1/2}\mathbf{W}(\mathbf{A}\mathbf{D}\mathbf{W})^\dagger(\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y} - \mathbf{p})$, we get

$$\mathbf{A}\mathbf{S}^{-1}\,\mathbf{v} = \mathbf{A}\mathbf{S}^{-1}(\mathbf{X}\mathbf{S})^{1/2}\mathbf{W}(\mathbf{A}\mathbf{D}\mathbf{W})^\dagger(\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y} - \mathbf{p})$$

$$= \mathbf{ADW}(\mathbf{ADW})^{\dagger}(\mathbf{AD}^2\mathbf{A}^{\mathsf{T}}\hat{\Delta}\mathbf{y} - \mathbf{p})$$
$$= \mathbf{AD}^2\mathbf{A}^{\mathsf{T}}\hat{\Delta}\mathbf{y} - \mathbf{p}\,,$$

where the second equality follows from $\mathbf{D} = \mathbf{X}^{1/2}\mathbf{S}^{-1/2}$. ■

We emphasize here that we use the same exact sketching matrix $\mathbf{W} \in \mathbb{R}^{n\times w}$ to form the preconditioner used in the iterative solver of Section 3 *as well as* the vector $\mathbf{v}$ in eqn. (27). This allows us to sketch $\mathbf{AD}$ only once, thus saving time in practice. Next, we present a bound for the two-norm of the perturbation vector $\mathbf{v}$ of eqn. (27).

Lemma 6. *With probability at least* $1-\delta$*, our perturbation vector* $\mathbf{v}$ *in Lemma 5 satisfies*

$$\|\mathbf{v}\|_2 \leq \sqrt{3n\mu}\,\|\tilde{\mathbf{f}}^{(t)}\|_2, \tag{28}$$

with $\tilde{\mathbf{f}}^{(t)} = \mathbf{Q}^{-1/2}\mathbf{AD}^2\mathbf{A}^{\mathsf{T}}\mathbf{Q}^{-1/2}\tilde{\mathbf{z}}^t - \mathbf{Q}^{-1/2}\mathbf{p}$.

Proof Recall that $\mathbf{Q} = \mathbf{ADW}(\mathbf{ADW})^{\mathsf{T}} = \mathbf{U_Q}\mathbf{\Sigma_Q}\mathbf{U_Q^{\mathsf{T}}}$. Also, $\mathbf{U_Q}$ and $\mathbf{\Sigma_Q^{1/2}}$ are (respectively) the matrices of the left singular vectors and the singular values of $\mathbf{ADW}$. Now, let $\widehat{\mathbf{V}}$ be the right singular vector of $\mathbf{ADW}$. Therefore, $\mathbf{ADW} = \mathbf{U_Q}\mathbf{\Sigma_Q^{1/2}}\widehat{\mathbf{V}}^{\mathsf{T}}$ is the thin SVD representation of $\mathbf{ADW}$. Also, from Lemma 2, we know that $\mathbf{Q}$ has full rank. Therefore, $\mathbf{Q}^{1/2}\mathbf{Q}^{-1/2} = \mathbf{I}_m$. Next, we bound $\|\mathbf{v}\|_2$:

$$\begin{aligned}\|\mathbf{v}\|_2 &= \|(\mathbf{XS})^{1/2}\mathbf{W}(\mathbf{ADW})^{\dagger}(\mathbf{AD}^2\mathbf{A}^{\mathsf{T}}\hat{\Delta}\mathbf{y} - \mathbf{p})\|_2\\ &= \|(\mathbf{XS})^{1/2}\mathbf{W}(\mathbf{ADW})^{\dagger}\mathbf{Q}^{1/2}\mathbf{Q}^{-1/2}(\mathbf{AD}^2\mathbf{A}^{\mathsf{T}}\hat{\Delta}\mathbf{y} - \mathbf{p})\|_2\\ &\leq \|(\mathbf{XS})^{1/2}\mathbf{W}(\mathbf{ADW})^{\dagger}\mathbf{Q}^{1/2}\|_2\,\|\tilde{\mathbf{f}}^{(t)}\|_2.\end{aligned} \tag{29}$$

In the above we used $\mathbf{Q}^{-1/2}(\mathbf{AD}^2\mathbf{A}^{\mathsf{T}}\hat{\Delta}\mathbf{y} - \mathbf{p}) = \tilde{\mathbf{f}}^{(t)}$. Using the SVD of $\mathbf{ADW}$ and $\mathbf{Q}$, we get $(\mathbf{ADW})^{\dagger}\mathbf{Q}^{1/2} = \widehat{\mathbf{V}}\mathbf{\Sigma_Q^{-1/2}}\mathbf{U_Q^{\mathsf{T}}}\,\mathbf{U_Q}\mathbf{\Sigma_Q^{1/2}}\mathbf{U_Q^{\mathsf{T}}} = \widehat{\mathbf{V}}\mathbf{U_Q^{\mathsf{T}}}$. Now, note that $\mathbf{U_Q} \in \mathbb{R}^{m\times m}$ is an orthogonal matrix and $\|\widehat{\mathbf{V}}\|_2 = 1$. Therefore, combining with eqn. (29) yields

$$\begin{aligned}\|\mathbf{v}\|_2 \leq \|(\mathbf{XS})^{1/2}\mathbf{W}\widehat{\mathbf{V}}\mathbf{U_Q^{\mathsf{T}}}\|_2\|\tilde{\mathbf{f}}^{(t)}\|_2 = \|(\mathbf{XS})^{1/2}\mathbf{W}\widehat{\mathbf{V}}\|_2\|\tilde{\mathbf{f}}^{(t)}\|_2\\ \leq\|(\mathbf{XS})^{1/2}\mathbf{W}\|_2\|\tilde{\mathbf{f}}^{(t)}\|_2.\end{aligned} \tag{30}$$

The first equality follows from the unitary invariance property of the spectral norm and the second inequality follows from the sub-multiplicativity of the spectral norm and $\|\widehat{\mathbf{V}}\|_2 = 1$. Our construction for $\mathbf{W}$ implies that eqn. (9) holds for any matrix $\mathbf{Z}$ and, in particular, for $\mathbf{Z} = (\mathbf{XS})^{1/2}$. Eqn. (9) implies that

$$\left\|(\mathbf{XS})^{1/2}\mathbf{W}\mathbf{W}^{\mathsf{T}}(\mathbf{XS})^{1/2} - (\mathbf{XS})\right\|_2 \leq \frac{\zeta}{4}\left(\|(\mathbf{XS})^{1/2}\|_2^2 + \frac{\|(\mathbf{XS})^{1/2}\|_F^2}{m}\right) \tag{31}$$

holds with probability at least $1-\delta$. Applying Weyl's inequality on the left hand side of the eqn. (31), we get

$$\left|\left\|(\mathbf{XS})^{1/2}\mathbf{W}\right\|_2^2 - \left\|(\mathbf{XS})^{1/2}\right\|_2^2\right| \leq \frac{\zeta}{4}\left(\|(\mathbf{XS})^{1/2}\|_2^2 + \frac{\|(\mathbf{XS})^{1/2}\|_F^2}{m}\right). \tag{32}$$

Using $\zeta \leq 1$ and $\|(\mathbf{XS})^{1/2}\|_2^2 \leq \|(\mathbf{XS})^{1/2}\|_F^2 = \mathbf{x}^\mathsf{T}\mathbf{s} = n\mu$, we get[5]

$$\left\|(\mathbf{XS})^{1/2}\mathbf{W}\right\|_2^2 \leq 3\|(\mathbf{XS})^{1/2}\|_F^2 = 3n\mu. \tag{33}$$

Finally, combining eqns. (30) and (33), we conclude

$$\|\mathbf{v}\|_2 \leq \sqrt{3n\mu}\|\tilde{\mathbf{f}}^{(t)}\|_2.$$

■

Intuitively, the bound in Lemma 6 implies that $\|\mathbf{v}\|_2$ depends on how close the approximate solution $\hat{\Delta}\mathbf{y}$ is to the exact solution. Lemma 6 is particularly useful in proving the convergence of Algorithm 2, which needs $\|\mathbf{v}\|_2$ to be a small quantity. Next, using the properties of our preconditioner $\mathbf{Q}^{-1/2}$, we prove that $\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 = \mathcal{O}(\sqrt{n})\sqrt{\mu}$. This bound allows us to further show that that if we run Algorithm 1 for $\mathcal{O}(\log n)$ iterations, then $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma}{4}\mu$. This inequality is critical in the convergence analysis of Algorithm 2 (see Section 4.1 for details). Before presenting our feasible IPM algorithm, we first prove the above two inequalities using a couple of lemmas.

Let $\mathcal{F}^0$ be the set of strictly feasible points respectively *i.e.*,

$$\mathcal{F}^0 = \{(\mathbf{x}, \mathbf{y}, \mathbf{s}) : \ (\mathbf{x}, \mathbf{s}) > \mathbf{0}, \ \mathbf{Ax} = \mathbf{b}, \ \mathbf{A}^\mathsf{T}\mathbf{y} + \mathbf{s} = \mathbf{c}\}.$$

In addition, we will need the following definition for the neighborhood

$$\mathcal{N}(\gamma) = \Big\{(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{F}^0 : x_i s_i \geq (1-\gamma)\mu\Big\}. \tag{34}$$

Here $\gamma \in (0, 1)$ and μ is the duality measure. Note that $\mathcal{N}(\gamma) \subseteq \mathcal{F}^0$ and we assume that $\mathcal{F}^0$ is non-empty.

Lemma 7. *Let $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}(\gamma)$ and let the sketching matrix $\mathbf{W} \in \mathbb{R}^{n\times w}$ satisfy the condition in eqn.* (6)*. Then,*

$$\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 \leq \ \left(1 + \frac{\sigma}{\sqrt{1-\gamma}}\right)\sqrt{2n\mu}\,. \tag{35}$$

Proof To bound $\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2$, first we express $\mathbf{p}$ as in eqn. (23) and rewrite

$$\mathbf{Q}^{-1/2}\mathbf{p} = \mathbf{Q}^{-1/2}\left(-\sigma\mu\mathbf{AS}^{-1}\mathbf{1}_n + \mathbf{Ax}\right). \tag{36}$$

Applying the triangle inequality on $\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2$ in eqn. (36), we get

$$\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 \leq \Delta_1 + \Delta_2\,, \tag{37}$$

where $\Delta_1 = \sigma\mu\|\mathbf{Q}^{-1/2}\mathbf{AD}(\mathbf{XS})^{-1/2}\mathbf{1}_n\|_2$ and $\Delta_2 = \|\mathbf{Q}^{-1/2}\mathbf{ADD}^{-1}\mathbf{x}\|_2$. In order to bound Δ_1 and Δ_2, we use the condition of eqn. (6). In particular, eqn. (6) implies that $\|\mathbf{Q}^{-1/2}\mathbf{AD}\|_2 \leq \sqrt{2}$ as $\zeta \leq 1$.

5. The constant three in eqn. (33) could be slightly improved to $3/2$; we chose to keep the suboptimal constant as the better constant does not result in any significant improvements in the number of iterations of Algorithm 2.

Bounding Δ_1. Applying submultiplicativity, we get

$$\begin{aligned}\Delta_1 &= \sigma\mu\,\|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\,(\mathbf{X}\mathbf{S})^{-1/2}\mathbf{1}_n\|_2 \\ &\leq \sigma\mu\,\|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\|_2\|(\mathbf{X}\mathbf{S})^{-1/2}\mathbf{1}_n\|_2 \leq \ \sqrt{2}\,\sigma\mu\,\|(\mathbf{X}\mathbf{S})^{-1/2}\mathbf{1}_n\|_2 \\ &= \sqrt{2}\,\sigma\mu\sqrt{\sum_{i=1}^{n}\frac{1}{x_i s_i}} \leq \ \sqrt{2}\,\sigma\mu\sqrt{\sum_{i=1}^{n}\frac{1}{(1-\gamma)\mu}} = \ \sqrt{2}\,\sigma\sqrt{\frac{n\,\mu}{(1-\gamma)}}\,, \end{aligned} \tag{38}$$

where we used the fact that $(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathcal{N}(\gamma)$.

Bounding Δ_2. Since $\mathbf{D} = \mathbf{S}^{-1/2}\mathbf{X}^{1/2}$ and $\mathbf{x} = \mathbf{X}\,\mathbf{1}_n$, we get

$$\begin{aligned}\Delta_2 &= \|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\,(\mathbf{S}^{1/2}\mathbf{X}^{-1/2})\,\mathbf{X}\,\mathbf{1}_n\|_2 = \ \|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\,(\mathbf{S}\mathbf{X})^{1/2}\,\mathbf{1}_n\|_2 \\ &\leq \|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\|_2\|(\mathbf{S}\mathbf{X})^{1/2}\,\mathbf{1}_n\|_2 \leq \ \sqrt{2}\sqrt{\sum_{i=1}^{n}x_i s_i} = \ \sqrt{2n\,\mu}. \end{aligned} \tag{39}$$

Final bound. Combining eqns. (37), (38), and (39), we get

$$\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 \leq \ \left(1+\frac{\sigma}{\sqrt{1-\gamma}}\right)\sqrt{2n\mu}\,. \tag{40}$$

This concludes the proof of Lemma 7. ■

Lemma 8. *Let $(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathcal{N}(\gamma)$ and let the sketching matrix $\mathbf{W}$ satisfy the conditions of eqns.* (6) *and* (7)*. Then, after $t \geq \frac{\log(n\,\psi)}{\log(1/\zeta)}$ iterations of the iterative solver in Algorithm 1, we have $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma}{4}\mu$. Here $\psi = \frac{4\sqrt{6}(1+\sigma/\sqrt{1-\gamma})}{\gamma\sigma}$ and $\tilde{\mathbf{f}}^{(t)} = \mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}\tilde{\mathbf{z}}^t - \mathbf{Q}^{-1/2}\mathbf{p}$ is the residual of the solver.*

Proof Combining Lemma 7 and the condition in eqn. (7), we get

$$\|\tilde{\mathbf{f}}^{(t)}\|_2 \leq \zeta^t\left(1+\frac{\sigma}{\sqrt{1-\gamma}}\right)\sqrt{2n\mu}. \tag{41}$$

Next, combining Lemma 6 and eqn. (41) we get

$$\|\mathbf{v}\|_2 \leq \sqrt{3n\mu}\,\|\tilde{\mathbf{f}}^{(t)}\|_2 \leq \sqrt{6}n\,\zeta^t\left(1+\frac{\sigma}{\sqrt{1-\gamma}}\right)\mu$$

Therefore, $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$ holds if $\sqrt{6}n\,\zeta^t\,(1+\sigma/\sqrt{1-\gamma})\,\mu \leq \frac{\gamma\sigma\mu}{4}$, which holds for our choice of t. Now, fixing γ, σ, and ζ, after $t = \mathcal{O}(\log n)$ iterations of Algorithm 1 the conclusions of the lemma hold. ■

Now, we are ready to present the feasible IPM algorithm. Recall the definition the neighborhood $\mathcal{N}(\gamma)$ in eqn. (34).

Algorithm 2 Feasible IPM

Input: $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$, $\gamma \in (0,1)$, tolerance $\epsilon > 0$, $\sigma \in (0, 4/5)$;
Initialize: $k \leftarrow 0$; initial point $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0) \in \mathcal{F}^0$;

1: **while** $\mu_k > \epsilon$ **do**
2: Compute sketching matrix $\mathbf{W} \in \mathbb{R}^{n\times w}$ (Section 2) with $\zeta = 1/2$ and $\delta = O(n^{-1})$;
3: Solve eqn. (5) for $\mathbf{z}$ using Algorithm 1 with $\mathbf{W}$ from step (2) and $t = \mathcal{O}(\log n)$, and then Compute $\hat{\Delta}\mathbf{y} = \mathbf{Q}^{-1/2}\mathbf{z}$;
4: Compute $\mathbf{v}$ using eqn. (27) with $\mathbf{W}$ from step (2); $\hat{\Delta}\mathbf{s}$ using eqn. (21b); $\hat{\Delta}\mathbf{x}$ using eqn. (24);
5: Compute $\tilde{\alpha} = \text{argmax}\{\alpha \in [0,1] : (\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k) + \alpha(\hat{\Delta}\mathbf{x}^k, \hat{\Delta}\mathbf{y}^k, \hat{\Delta}\mathbf{s}^k) \in \mathcal{N}(\gamma)\}$.
6: Compute $\bar{\alpha} = \text{argmin}\{\alpha \in [0,\tilde{\alpha}] : (\mathbf{x}^k + \alpha\hat{\Delta}\mathbf{x}^k)^\mathsf{T}(\mathbf{s}^k + \alpha\hat{\Delta}\mathbf{s}^k)\}$.
7: Compute $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}) = (\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k) + \bar{\alpha}(\hat{\Delta}\mathbf{x}^k, \hat{\Delta}\mathbf{y}^k, \hat{\Delta}\mathbf{s}^k)$; set $k \leftarrow k+1$;
8: **end while**

Running time. We start by discussing the running time to compute $\mathbf{v}$. As discussed in Section 3, $(\mathbf{ADW})^\dagger$ can be computed in $\mathcal{O}(\mathsf{nnz}(\mathbf{A}) \cdot \log(m/\delta) + m^3 \log(m/\delta))$ time. Now, as $\mathbf{W}$ has $\mathcal{O}(\log(m/\delta))$ non-zero entries per row, pre-multiplying by $\mathbf{W}$ takes $\mathcal{O}(\mathsf{nnz}(\mathbf{A}) \log(m/\delta))$ time (assuming $\mathsf{nnz}(A) \geq n$). Since $\mathbf{X}$ and $\mathbf{S}$ are diagonal matrices, computing $\mathbf{v}$ takes $\mathcal{O}(\mathsf{nnz}(\mathbf{A}) \cdot \log(m/\delta) + m^3 \log(m/\delta))$ time, which is asymptotically the same as computing $\mathbf{Q}^{-1/2}$ (see eqn. (12)).

We now discuss the overall running time of Algorithm 2. At each iteration, with failure probability δ, the preconditioner $\mathbf{Q}^{-1/2}$ and the vector $\mathbf{v}$ can be computed in $\mathcal{O}(\mathsf{nnz}(\mathbf{A}) \cdot \log(m/\delta) + m^3 \log(m/\delta))$ time. In addition, for $t = \mathcal{O}(\log n)$ iterations of Algorithm 1, all the matrix-vector products in the CG or Chebyshev iteration can be computed in $\mathcal{O}(\mathsf{nnz}(\mathbf{A}) \cdot \log n)$ time. Therefore, the computational time for steps (2)-(4) is given by $\mathcal{O}(\mathsf{nnz}(\mathbf{A}) \cdot (\log n + \log(m/\delta)) + m^3 \log(m/\delta))$. Finally, considering ϵ to be a constant, if we assume that the IPM needs $k = c\,n$ iterations to converge and accordingly, if we fix the failure probability $\delta = \frac{0.1}{c\,n}$ for some suitable constant c, then taking a union bound over all the IPM iterations, our algorithm converges with probability at least $1 - c\,n \cdot \frac{0.1}{c\,n} = 0.9$ and the running time at each iteration is given by $\mathcal{O}((\mathsf{nnz}(\mathbf{A}) + m^3) \log n)$.

4.1 Convergence Analysis of Algorithm 2

In this section, we prove a set of results that ultimately establish Theorem 1 and guarantee the convergence of Algorithm 2. Due to the use of an approximate solver, these proofs typically differ from the standard analysis of long-step feasible IPM (Wright, 1997) in many aspects. For example, all the major results in this section rely on the condition that the error due to the linear solver is small *i.e.*, $\|\mathbf{v}\|_2$ is small, whereas the standard convergence analysis does not have this requirement as the linear system there is solved exactly *i.e.*$\|\mathbf{v}\|_2$ is always zero. This difference makes our case more intricate as we deal with an extra term involving $\|\mathbf{v}\|_2$ which needs more care.

On the other hand, while the origin of the statements of our feasible IPM results is essentially (Monteiro and O'Neal, 2003), the proofs are different from that of (Monteiro and O'Neal, 2003). The exact same analysis of (Monteiro and O'Neal, 2003) (just by

making the primal and dual residuals equal to zero) neither directly applies to the feasible case, nor matches the best iteration complexity of it, whereas our current analysis has the iteration complexity $\mathcal{O}(n \log 1/\epsilon)$, which is the best known for feasible long-step path following IPM algorithms. The proofs that look similar to (Monteiro and O'Neal, 2003) also have differences. We discuss them individually before the respective lemmas. The only overlap we have with (Monteiro and O'Neal, 2003) is our Lemma 11 that works for both feasible and infeasible setting. Now, we proceed to prove our Theorem 1.

First, we can rewrite the linear system of eqns. (24), (25), (26) as follows:

$$\mathbf{A}\hat{\Delta}\mathbf{x} = \mathbf{0}, \tag{42a}$$

$$\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y} + \hat{\Delta}\mathbf{s} = \mathbf{0}, \tag{42b}$$

$$\mathbf{X}\hat{\Delta}\mathbf{s} + \mathbf{S}\hat{\Delta}\mathbf{x} = -\mathbf{XS}\,\mathbf{1}_n + \sigma\mu\,\mathbf{1}_n - \mathbf{v}. \tag{42c}$$

Indeed, we now show how to derive eqns. (24), (25), (26) from eqn. (42). Pre-multiplying both sides of eqn. (42c) by $\mathbf{AS}^{-1}$ and noting that $\mathbf{D}^2 = \mathbf{XS}^{-1}$, we get

$$\begin{aligned} &\mathbf{AD}^2\hat{\Delta}\mathbf{s} + \mathbf{A}\hat{\Delta}\mathbf{x} = -\mathbf{AX1}_n + \sigma\mu\mathbf{AS}^{-1}\mathbf{1}_n - \mathbf{AS}^{-1}\mathbf{v} \\ \Rightarrow\ &\mathbf{AD}^2\hat{\Delta}\mathbf{s} = -\mathbf{Ax} + \sigma\mu\mathbf{AS}^{-1}\mathbf{1}_n - \mathbf{AS}^{-1}\mathbf{v}. \end{aligned} \tag{43}$$

Eqn. (43) holds as $\mathbf{AX1}_n = \mathbf{Ax}$ and, from eqn. (42a), $\mathbf{A}\hat{\Delta}\mathbf{x} = \mathbf{0}$. Next, pre-multiplying eqn. (42b) by $\mathbf{AD}^2$, we get

$$\begin{aligned} &\mathbf{AD}^2\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y} + \mathbf{AD}^2\hat{\Delta}\mathbf{s} = \mathbf{0} \\ \Rightarrow\ &\mathbf{AD}^2\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y} = \mathbf{Ax} - \sigma\mu\mathbf{AS}^{-1}\mathbf{1}_n + \mathbf{AS}^{-1}\mathbf{v} = \mathbf{p} + \mathbf{AS}^{-1}\mathbf{v}. \end{aligned} \tag{44}$$

The first equality in eqn. (44) follows from eqn. (43) and the definition of $\mathbf{p}$. This establishes eqn. (25). Eqn. (26) directly follows from eqn. (42b). Finally, we get eqn. (24) by pre-multiplying eqn. (42c) by $\mathbf{S}^{-1}$. We will now use a slightly different notations. We define the next point traversed by the algorithm as $(\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{s}(\alpha))$, where

$$(\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{s}(\alpha)) = (\mathbf{x}, \mathbf{y}, \mathbf{s}) + \alpha(\hat{\Delta}\mathbf{x}, \hat{\Delta}\mathbf{y}, \hat{\Delta}\mathbf{s}), \text{ and} \tag{45}$$

$$\mu(\alpha) = (1/n)\,\mathbf{x}(\alpha)^\mathsf{T}\mathbf{s}(\alpha). \tag{46}$$

Our goal is to bound the number of outer iterations required by the feasible IPM algorithm. To do so, we bound the magnitude of the step size α. First, we provide an upper bound on α, which allows us to show that each new point $(\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{s}(\alpha))$ traversed by the algorithm stays within the neighborhood $\mathcal{N}(\gamma)$. Second, we provide a lower bound on α, which allows us to bound the number of iterations required. The following Lemma will be used throughout the section.

Lemma 9. *Assume* $(\hat{\Delta}\mathbf{x}, \hat{\Delta}\mathbf{s}, \hat{\Delta}\mathbf{y})$ *satisfies eqns.* (87) *for some* $\sigma \in \mathbb{R}$ *and* $\mathbf{v} \in \mathbb{R}^n$. *Let* $(\mathbf{x}, \mathbf{y}, \mathbf{s})$ *be any point such that* $(\mathbf{x}, \mathbf{s}) > 0$. *Then, for every* $\alpha \in \mathbb{R}$,

$$(a)\quad \mathbf{x}(\alpha) \circ \mathbf{s}(\alpha) = (1-\alpha)\mathbf{x} \circ \mathbf{s} + \alpha\sigma\mu\mathbf{1}_n - \alpha\mathbf{v} + \alpha^2\hat{\Delta}\mathbf{x} \circ \hat{\Delta}\mathbf{s}\,,$$

$$(b)\quad \mu(\alpha) = [1 - \alpha(1-\sigma)]\mu - \frac{\alpha\,\mathbf{v}^\mathsf{T}\mathbf{1}_n}{n}.$$

Proof Proving (a):

$$\begin{aligned}\mathbf{x}(\alpha)\circ\mathbf{s}(\alpha) &= (\mathbf{x}+\alpha\hat{\Delta}\mathbf{x})\circ(\mathbf{s}+\alpha\hat{\Delta}\mathbf{s})\\ &= \mathbf{x}\circ\mathbf{s}+\alpha(\mathbf{x}\circ\hat{\Delta}\mathbf{s}+\mathbf{s}\circ\hat{\Delta}\mathbf{x})+\alpha^2\hat{\Delta}\mathbf{x}\circ\hat{\Delta}\mathbf{s}\\ &= \mathbf{x}\circ\mathbf{s}+\alpha(-\mathbf{x}\circ\mathbf{s}+\sigma\mu\mathbf{1}_n-\mathbf{v})+\alpha^2\hat{\Delta}\mathbf{x}\circ\hat{\Delta}\mathbf{s}\\ &= (1-\alpha)\mathbf{x}\circ\mathbf{s}+\alpha\sigma\mu\mathbf{1}_n-\alpha\mathbf{v}+\alpha^2\hat{\Delta}\mathbf{x}\circ\hat{\Delta}\mathbf{s}\,,\end{aligned}$$

where the third equality follows from eqn. (42c). Now, left-multiply the above equality by $\mathbf{1}_n^\mathsf{T}$ and divide by n to obtain (b). (Notice that $\hat{\Delta}\mathbf{x}^\mathsf{T}\hat{\Delta}\mathbf{s}=0$ from eqns. (42a) and (42b).) ■

Next, we provide an upper bound on α, ensuring that each new point $(\mathbf{x}(\alpha),\mathbf{y}(\alpha),\mathbf{s}(\alpha))$ traversed by the algorithm stays within the neighborhood $\mathcal{N}(\gamma)$. Note that the following result resembles Lemma 3.5 of (Monteiro and O'Neal, 2003), but what makes Lemma 10 different from it is the fact that here we need to additionally prove the strict feasibility of the new iterate *i.e.* $(\mathbf{x}(\alpha),\mathbf{y}(\alpha),\mathbf{s}(\alpha))\in\mathcal{F}^0$ (in order to show $(\mathbf{x}(\alpha),\mathbf{y}(\alpha),\mathbf{s}(\alpha))\in\mathcal{N}(\gamma)$), which was not proven in Lemma 3.5 of (Monteiro and O'Neal, 2003).

Lemma 10. *Assume* $(\hat{\Delta}\mathbf{x},\hat{\Delta}\mathbf{y},\hat{\Delta}\mathbf{s})$ *satisfies eqns.* (42) *for some* $\sigma>0$, $(\mathbf{x},\mathbf{y},\mathbf{s})\in\mathcal{N}(\gamma)$ *for* $\gamma\in(0,1)$, *and* $\|\mathbf{v}\|_2\le\frac{\gamma\sigma\mu}{4}$. *Then,* $(\mathbf{x}(\alpha),\mathbf{y}(\alpha),\mathbf{s}(\alpha))\in\mathcal{N}(\gamma)$ *for every scalar* α *such that*

$$0\le\alpha\le\min\left\{1,\frac{\gamma\sigma\mu}{4\|\hat{\Delta}\mathbf{x}\circ\hat{\Delta}\mathbf{s}\|_\infty}\right\}. \tag{47}$$

Proof First, we show that $\mathbf{x}(\alpha)\circ\mathbf{s}(\alpha)\ge(1-\gamma)\mu(\alpha)\mathbf{1}_n$. From Lemma 9, we get

$$\begin{aligned}&\mathbf{x}(\alpha)\circ\mathbf{s}(\alpha)-(1-\gamma)\mu(\alpha)\mathbf{1}_n\\ &=(1-\alpha)\left(\mathbf{x}\circ\mathbf{s}-(1-\gamma)\mu\,\mathbf{1}_n\right)+\alpha\gamma\sigma\mu\,\mathbf{1}_n-\alpha\left(\mathbf{v}-(1-\gamma)\frac{\mathbf{v}^\mathsf{T}\mathbf{1}_n}{n}\mathbf{1}_n\right)\\ &\qquad\qquad+\alpha^2\left(\hat{\Delta}\mathbf{x}\circ\hat{\Delta}\mathbf{s}\right)\\ &\ge\alpha\left(\gamma\sigma\mu-\left\|\mathbf{v}-(1-\gamma)\frac{\mathbf{v}^\mathsf{T}\mathbf{1}_n}{n}\,\mathbf{1}_n\right\|_\infty-\alpha\left\|\hat{\Delta}\mathbf{x}\circ\hat{\Delta}\mathbf{s}\right\|_\infty\right)\mathbf{1}_n\\ &\ge\alpha\left(\gamma\sigma\mu-2\|\mathbf{v}\|_\infty-\alpha\|\hat{\Delta}\mathbf{x}\circ\hat{\Delta}\mathbf{s}\|_\infty\right)\mathbf{1}_n\\ &\ge\alpha\left(\gamma\sigma\mu-\frac{\gamma\sigma\mu}{2}-\frac{\gamma\sigma\mu}{4}\right)\mathbf{1}_n=\alpha\frac{\gamma\sigma\mu}{4}\mathbf{1}_n\ge\mathbf{0}.\end{aligned}$$

The first inequality follows from $\mathbf{x}\circ\mathbf{s}\ge(1-\gamma)\mu\,\mathbf{1}_n$, because $(\mathbf{x},\mathbf{y},\mathbf{s})\in\mathcal{N}(\gamma)$ and $\mathbf{a}\le\|\mathbf{a}\|_\infty\,\mathbf{1}_n$ for any vector $\mathbf{a}\in\mathbb{R}^n$. The second-to-last inequality follows from the fact that for any $\mathbf{u}\in\mathbb{R}^n$ and $\delta\in[0,n]$, $\left\|\mathbf{u}-\delta\frac{\mathbf{u}^\mathsf{T}\mathbf{1}_n}{n}\,\mathbf{1}_n\right\|_\infty\le(1+\delta)\|\mathbf{u}\|_\infty$. Thus, we prove that the point $(\mathbf{x}(\alpha),\mathbf{y}(\alpha),\mathbf{s}(\alpha))$ satisfies the proximity condition for $\mathcal{N}(\gamma)$.

Finally, we show that $(\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{s}(\alpha)) \in \mathcal{F}^0$ *i.e.* it satisfies the primal and dual constraints and $(\mathbf{x}(\alpha), \mathbf{s}(\alpha)) > \mathbf{0}$. From eqn. (42) and the fact that $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{F}^0$, we get $\mathbf{A}\,\mathbf{x}(\alpha) = \mathbf{A}\mathbf{x} + \alpha\mathbf{A}\hat{\Delta}\mathbf{x} = \mathbf{b}$. Similarly,

$$\mathbf{A}^\mathsf{T}\,\mathbf{y}(\alpha) + \mathbf{s}(\alpha) = (\mathbf{A}^\mathsf{T}\mathbf{y} + \mathbf{s}) + \alpha(\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y} + \hat{\Delta}\mathbf{s}) = \mathbf{A}^\mathsf{T}\mathbf{y} + \mathbf{s} = \mathbf{c}.$$

We now show that $(\mathbf{x}(\alpha), \mathbf{s}(\alpha)) > \mathbf{0}$. For $\alpha = 0$, we trivially have$(\mathbf{x}(\alpha), \mathbf{s}(\alpha)) = (\mathbf{x}, \mathbf{s}) > \mathbf{0}$. To prove $(\mathbf{x}(\alpha), \mathbf{s}(\alpha)) > \mathbf{0}$ for $0 < \alpha \leq 1$, we first show $\mu(\alpha) > 0$. Using $\gamma \in (0, 1)$, the inequality $\left|\frac{\mathbf{v}^\mathsf{T}\mathbf{1}_n}{n}\right| \leq \|\mathbf{v}\|_\infty \leq \|\mathbf{v}\|_2$, and the assumption $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$, we get $\frac{\mathbf{v}^\mathsf{T}\mathbf{1}_n}{n} < \frac{\sigma\mu}{4}$. Thus, from Lemma 9(b),

$$\begin{aligned}
\mu(\alpha) &= [1 - \alpha(1-\sigma)]\mu - \alpha\,\frac{\mathbf{v}^\mathsf{T}\mathbf{1}_n}{n} \\
&> [1 - \alpha(1-\sigma)]\mu - \alpha\,\frac{\sigma\mu}{4} \\
&= (1-\alpha)\mu + \alpha\,\frac{3\,\sigma\mu}{4} > 0.
\end{aligned} \tag{48}$$

The last inequality holds because $\alpha \in (0, 1]$, $\sigma \in (0, 1)$, and $\mu > 0$. We already have $\mathbf{x}(\alpha) \circ \mathbf{s}(\alpha) \geq (1-\gamma)\mu(\alpha)\mathbf{1}_n$. Combining with $\mu(\alpha) > 0$ and $\gamma \in (0, 1)$ this implies that $\mathbf{x}(\alpha) \circ \mathbf{s}(\alpha) > \mathbf{0}$. Therefore, $x_i(\alpha)\, s_i(\alpha) > 0$ for all $i = 1 \ldots n$ which implies that, for each i, either both $x_i(\alpha)$ and $s_i(\alpha)$ are positive or both $x_i(\alpha)$ and $s_i(\alpha)$ are negative. We will use contradiction to prove that the second case is not possible.

Indeed, assume that $x_i(\alpha) < 0$ and $s_i(\alpha) < 0$ for some $i = 1 \ldots n$. First, we rewrite $x_i(\alpha)$ and $s_i(\alpha)$ as follows[6]

$$x_i(\alpha) = x_i + \alpha\hat{\Delta}x_i < 0, \tag{49a}$$

$$s_i(\alpha) = s_i + \alpha\hat{\Delta}s_i < 0. \tag{49b}$$

Recall that both x_i and s_i are positive. Therefore, pre-multiplying eqn. (49a) by s_i and eqn. (49b) by x_i we get

$$x_i s_i + \alpha s_i \hat{\Delta}x_i < 0, \tag{50a}$$

$$x_i s_i + \alpha x_i \hat{\Delta}s_i < 0. \tag{50b}$$

Adding eqns. (50a) and (50b) and applying eqn. (42c) (element-wise), we get

$$\begin{aligned}
& 2x_i s_i + \alpha(s_i\hat{\Delta}x_i + x_i\hat{\Delta}s_i) < 0 \\
\Rightarrow\ & 2x_i s_i + \alpha(-x_i s_i + \sigma\mu - v_i) < 0 \\
\Rightarrow\ & (2-\alpha)x_i s_i + \alpha\sigma\mu - \alpha v_i < 0 \\
\Rightarrow\ & v_i > \frac{2-\alpha}{\alpha} x_i s_i + \sigma\mu > \sigma\mu.
\end{aligned} \tag{51}$$

In the above v_i is the i-th element of $\mathbf{v}$; the first inequality in eqn. (51) holds because $\alpha > 0$; the second inequality in eqn. (51) because $\frac{2-\alpha}{\alpha} x_i s_i > 0$ ($x_i, s_i > 0$ and $0 < \alpha \leq 1$). Using $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$ we get

$$v_i \leq \|\mathbf{v}\|_\infty \leq \|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4} < \sigma\mu\,,$$

6. Here, $x_i(\alpha)$, $s_i(\alpha)$, x_i, s_i, $\hat{\Delta}x_i$, and $\hat{\Delta}s_i$ are the i-th elements of $\mathbf{x}(\alpha)$, $\mathbf{s}(\alpha)$, $\mathbf{x}$, $\mathbf{s}$, $\hat{\Delta}\mathbf{x}$, and $\hat{\Delta}\mathbf{s}$, respectively.

for all $i = 1 \dots n$. This contradicts the inequality of eqn. (51); thus, both $x_i(\alpha) > 0$ and $s_i(\alpha) > 0$ for all $i = 1 \dots n$ and all $\alpha \in [0, 1]$. ■

We now cite a result from (Monteiro and O'Neal, 2003) that provides a lower bound on $\bar{\alpha}$ and the corresponding $\mu(\bar{\alpha})$. Note that (Monteiro and O'Neal, 2003) presented it in context of infeasible IPM; however, it holds for the feasible case as well, as long as the perturbation vector $\mathbf{v}$ satisfies $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$ at each iteration.

Lemma 11 (Lemma 3.6 of (Monteiro and O'Neal, 2003))**.** *At each iteration of the Algorithm 2, if $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$, then the step size $\bar{\alpha}$ satisfies*

$$\bar{\alpha} \geq \min\left\{1, \frac{\min\{\gamma\sigma, (1 - \frac{5}{4}\sigma)\}\mu}{4\|\hat{\Delta}x \circ \hat{\Delta}s\|_\infty}\right\} \tag{52}$$

and

$$\mu(\bar{\alpha}) = \Big[1 - \frac{\bar{\alpha}}{2}(1 - \frac{5}{4}\sigma)\Big]\mu. \tag{53}$$

At this point, we have provided a lower bound (eqn. (52)) for the allowed values of the step size $\bar{\alpha}$. Next, we will show that this lower bound is bounded away from zero. From eqn. (52), is suffices to show that $\|\hat{\Delta}\mathbf{x} \circ \hat{\Delta}\mathbf{s}\|_\infty$ is bounded. First, we state the following inequality that will be instrumental in proving Lemma 13.

Lemma 12. *Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ be any two vectors such that $\mathbf{a}^\mathsf{T}\mathbf{b} \geq 0$. Then*

$$\|\mathbf{a} \circ \mathbf{b}\|_2 \leq \|\mathbf{a} + \mathbf{b}\|_2^2 .$$

See (Wright, 1997) for a proof of Lemma 12; as a matter of fact, (Wright, 1997) proved $\|\mathbf{a} \circ \mathbf{b}\|_2 \leq 2^{-3/2}\|\mathbf{a} + \mathbf{b}\|_2^2$, which is tighter. This tighter bound is not needed in our proof.

Lemma 13. *Let $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}(\gamma)$ and $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$. Then $(\hat{\Delta}\mathbf{x}, \hat{\Delta}\mathbf{y}, \hat{\Delta}\mathbf{s})$ satisfies*

$$\|\hat{\Delta}\mathbf{x} \circ \hat{\Delta}\mathbf{s}\|_\infty \leq \left(1 + \frac{\sigma^2}{1 - \gamma}\right) n\mu + \frac{\gamma^2\sigma^2}{16(1 - \gamma)}\mu + \frac{\gamma\sigma^2}{2}\mu . \tag{54}$$

Proof First, we multiply eqn. (42c) on the left by $(\mathbf{XS})^{-1/2}$ to get

$$\mathbf{D}^{-1}\hat{\Delta}\mathbf{x} + \mathbf{D}\hat{\Delta}\mathbf{s} = -(\mathbf{XS})^{1/2}\mathbf{1}_n + \sigma\mu(\mathbf{XS})^{-1/2}\mathbf{1}_n - (\mathbf{XS})^{-1/2}\mathbf{v} . \tag{55}$$

Next, pre-multiplying eqn. (42b) by $\hat{\Delta}\mathbf{x}^\mathsf{T}$ and applying eqn. (42a), we have $\hat{\Delta}\mathbf{x}^\mathsf{T}\hat{\Delta}\mathbf{s} = 0$. This also implies that $\hat{\Delta}\mathbf{x}^\mathsf{T}\hat{\Delta}\mathbf{s} = (\mathbf{D}^{-1}\hat{\Delta}\mathbf{x})^\mathsf{T}(\mathbf{D}\hat{\Delta}\mathbf{s})$. Applying Lemma 12 with $\mathbf{a} = \mathbf{D}^{-1}\hat{\Delta}\mathbf{x}$ and $\mathbf{b} = \mathbf{D}\hat{\Delta}\mathbf{s}$, and using eqn. (55) we get

$$\begin{aligned}
\|\hat{\Delta}\mathbf{x} \circ \hat{\Delta}\mathbf{s}\|_2 &= \|(\mathbf{D}^{-1}\hat{\Delta}\mathbf{x}) \circ (\mathbf{D}\hat{\Delta}\mathbf{s})\|_2 \leq \|\mathbf{D}^{-1}\hat{\Delta}\mathbf{x} + \mathbf{D}\hat{\Delta}\mathbf{s}\|_2^2 \\
&= \| - (\mathbf{XS})^{1/2}\mathbf{1}_n + \sigma\mu(\mathbf{XS})^{-1/2}\mathbf{1}_n - (\mathbf{XS})^{-1/2}\mathbf{v}\|_2^2 \\
&= \|(\mathbf{XS})^{1/2}\mathbf{1}_n + (\mathbf{XS})^{-1/2}(\mathbf{v} - \sigma\mu\mathbf{1}_n)\|_2^2
\end{aligned}$$

$$= \|(\mathbf{XS})^{1/2}\mathbf{1}_n\|_2^2 + 2\cdot\mathbf{1}_n^\mathsf{T}(\mathbf{v}-\sigma\mu\mathbf{1}_n) + \|(\mathbf{XS})^{-1/2}(\mathbf{v}-\sigma\mu\mathbf{1}_n)\|_2^2$$
$$\leq n\mu + 2n(\|\mathbf{v}\|_2 - \sigma\mu) + \|(\mathbf{XS})^{-1/2}(\mathbf{v}-\sigma\mu\mathbf{1}_n)\|_2^2. \tag{56}$$

The inequality in eqn. (56) follows from $\|(\mathbf{XS})^{1/2}\mathbf{1}_n\|_2^2 = n\mu$ and $\left|\mathbf{1}_n^\mathsf{T}\mathbf{v}\right| \leq n\|\mathbf{v}\|_\infty \leq n\|\mathbf{v}\|_2$. Next, consider the last term on the right hand side of eqn. (56):

$$\|(\mathbf{XS})^{-1/2}(\mathbf{v}-\sigma\mu\mathbf{1}_n)\|_2^2 = \|(\mathbf{XS})^{-1/2}\mathbf{v}\|_2^2 + \sigma^2\mu^2\|(\mathbf{XS})^{-1/2}\mathbf{1}_n\|_2^2 - 2\sigma\mu\,\mathbf{1}_n^\mathsf{T}(\mathbf{XS})^{-1}\mathbf{v}$$
$$\leq \frac{\|\mathbf{v}\|_2^2}{\min_i x_i s_i} + \sigma^2\mu^2\sum_{i=1}^n \frac{1}{x_i s_i} - 2\sigma\mu\sum_{i=1}^n \frac{v_i}{x_i s_i}. \tag{57}$$

Eqn. (57) follows from $\|(\mathbf{XS})^{-1/2}\mathbf{v}\|_2^2 \leq \|(\mathbf{XS})^{-1/2}\|_2^2\|\mathbf{v}\|_2^2$ and $\|(\mathbf{XS})^{-1/2}\|_2^2 = 1/\min_i x_i s_i$. Moreover, it is easy to verify that $\|(\mathbf{XS})^{-1/2}\mathbf{1}_n\|_2^2 = \sum_{i=1}^n 1/x_i s_i$ and $\mathbf{1}_n^\mathsf{T}(\mathbf{XS})^{-1}\mathbf{v} = \sum_{i=1}^n v_i/x_i s_i$. Now, we have $x_i s_i \geq (1-\gamma)\mu$ for $i = 1\ldots n$ as $(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathcal{N}(\gamma)$. Also, $x_i s_i \leq \sum_{i=1}^n x_i s_i = n\mu$. Using the above we rewrite eqn. (57) as

$$\|(\mathbf{XS})^{-1/2}(\mathbf{v}-\sigma\mu\mathbf{1}_n)\|_2^2 \leq \frac{\|\mathbf{v}\|_2^2}{(1-\gamma)\mu} + \frac{n\sigma^2\mu^2}{(1-\gamma)\mu} - 2\sigma\mu\frac{\sum_{i=1}^n v_i}{n\mu}. \tag{58}$$

Combining eqns. (56) and (58) we get

$$\|\hat{\Delta\mathbf{x}}\circ\hat{\Delta\mathbf{s}}\|_2 \leq n\mu + 2n(\|\mathbf{v}\|_2 - \sigma\mu) + \frac{\|\mathbf{v}\|_2^2}{(1-\gamma)\mu} + \frac{n\sigma^2\mu}{(1-\gamma)} - 2\sigma\frac{\sum_{i=1}^n v_i}{n}$$
$$= \left(1+\frac{\sigma^2}{1-\gamma}\right)n\mu + 2n(\|\mathbf{v}\|_2 - \sigma\mu) + \frac{\|\mathbf{v}\|_2^2}{(1-\gamma)\mu} - 2\sigma\frac{\sum_{i=1}^n v_i}{n}. \tag{59}$$

Using $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$ and the fact that $\left|\mathbf{v}^\mathsf{T}\mathbf{1}_n/n\right| \leq \|\mathbf{v}\|_\infty \leq \|\mathbf{v}\|_2$, we get

$$\|\hat{\Delta\mathbf{x}}\circ\hat{\Delta\mathbf{s}}\|_2 \leq \left(1+\frac{\sigma^2}{1-\gamma}\right)n\mu + \frac{\gamma^2\sigma^2}{16(1-\gamma)}\mu + \frac{\gamma\sigma^2}{2}\mu. \tag{60}$$

Finally, we conclude the proof using $\|\hat{\Delta\mathbf{x}}\circ\hat{\Delta\mathbf{s}}\|_\infty \leq \|\hat{\Delta\mathbf{x}}\circ\hat{\Delta\mathbf{s}}\|_2$. ■

The next result guarantees the convergence of Algorithm 2 .

Lemma 14. *Assume that the constants γ and σ are such that $\max\{\gamma^{-1},(1-\gamma)^{-1},\sigma^{-1},(1-\frac{5}{4}\sigma)^{-1}\} = \mathcal{O}(1)$. At each iteration of Algorithm 2, if $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$, then after $k = \mathcal{O}(n\log 1/\epsilon)$ iterations, $(\mathbf{x}^k,\mathbf{s}^k,\mathbf{y}^k)$ satisfies*

$$\mu_k \leq \epsilon\mu_0.$$

Proof From Lemma 13,

$$\|\hat{\Delta\mathbf{x}}\circ\hat{\Delta\mathbf{s}}\|_\infty \leq \left(1+\frac{\sigma^2}{1-\gamma}+\frac{\gamma^2\sigma^2}{16(1-\gamma)}+\frac{\gamma\sigma^2}{2}\right)n\mu$$
$$\Rightarrow \mu\|\hat{\Delta\mathbf{x}}\circ\hat{\Delta\mathbf{s}}\|_\infty^{-1} \geq n^{-1}\left(1+\frac{\sigma^2}{1-\gamma}+\frac{\gamma^2\sigma^2}{16(1-\gamma)}+\frac{\gamma\sigma^2}{2}\right)^{-1}. \tag{61}$$

Combining eqns. (52) and (61) we get

$$\bar{\alpha} \geq \min\left\{1, \frac{\min\{\gamma\sigma, (1-\frac{5}{4}\sigma)\}}{4\,n\left(1+\frac{\sigma^2}{1-\gamma}+\frac{\gamma^2\sigma^2}{16(1-\gamma)}+\frac{\gamma\sigma^2}{2}\right)}\right\}. \tag{62}$$

Let $\max\{\gamma^{-1}, (1-\gamma)^{-1}, \sigma^{-1}, (1-\frac{5}{4}\sigma)^{-1}\} \leq \lambda$ for some constant $\lambda > 1$. Therefore, $\gamma\sigma \geq \frac{1}{\lambda^2}$ and $(1-\frac{5}{4}\sigma) \geq \frac{1}{\lambda}$, which further implies that $\min\{\gamma\sigma, (1-\frac{5}{4}\sigma)\} \geq \min\{\frac{1}{\lambda}, \frac{1}{\lambda^2}\} = \frac{1}{\lambda^2}$. Also, $\frac{1}{1-\gamma} \leq \lambda$. Combining these with eqn. (62), we get

$$\bar{\alpha} \geq \min\left\{1, \frac{\frac{1}{\lambda^2}}{4n(1+\lambda\sigma^2+\frac{\lambda\gamma^2\sigma^2}{16}+\frac{\gamma\sigma^2}{2})}\right\} = \min\left\{1, \frac{1}{4n(\lambda^2+\lambda^3\sigma^2+\frac{\lambda^3\gamma^2\sigma^2}{16}+\frac{\lambda^2\gamma\sigma^2}{2})}\right\}. \tag{63}$$

Note that in eqn.(63), $4n(\lambda^2+\lambda^3\sigma^2+\frac{\lambda^3\gamma^2\sigma^2}{16}+\frac{\lambda^2\gamma\sigma^2}{2}) > 1$. Thus,

$$\begin{aligned} &\bar{\alpha} \geq \frac{1}{4n(\lambda^2+\lambda^3\sigma^2+\frac{\lambda^3\gamma^2\sigma^2}{16}+\frac{\lambda^2\gamma\sigma^2}{2})} \\ \Rightarrow\ &\frac{\bar{\alpha}}{2}\left(1-\frac{5}{4}\sigma\right) \geq \frac{1-\frac{5}{4}\sigma}{8n(\lambda^2+\lambda^3\sigma^2+\frac{\lambda^3\gamma^2\sigma^2}{16}+\frac{\lambda^2\gamma\sigma^2}{2})} \geq \frac{1}{8n\lambda^3(1+\lambda\sigma^2+\frac{\lambda\gamma^2\sigma^2}{16}+\frac{\gamma\sigma^2}{2})} = \frac{\beta}{n}, \end{aligned} \tag{64}$$

where

$$\beta = \frac{1}{8\lambda^3(1+\lambda\sigma^2+\frac{\lambda\gamma^2\sigma^2}{16}+\frac{\gamma\sigma^2}{2})}. \tag{65}$$

We also note that the second inequality in eqn. (64) holds because $1-\frac{5}{4}\sigma \geq \frac{1}{\lambda}$. Let $\mu = \mu_k$ and $\mu(\bar{\alpha}) = \mu_{k+1}$ in eqn. (53); applying eqn. (64), we get, $\mu_{k+1} \leq (1-\beta/n)\,\mu_k, \forall k \geq 0$. Applying the above inequality recursively, we get $\mu_k \leq (1-\beta/n)^k\,\mu_0$. Therefore, for any accuracy parameter $\epsilon \in (0,1)$, $\mu_k \leq \epsilon\mu_0$ holds, if $(1-\beta/n)^k \leq \epsilon$ holds. Thus, it suffices for k to be at least

$$k \geq \frac{n}{\beta}\log(1/\epsilon). \tag{66}$$

Therefore, since β, as defined in eqn. (65), is a constant, we need $\mathcal{O}(n\log\frac{1}{\epsilon})$ iterations to satisfy $\mu_k \leq \epsilon\mu_0$. ■

Proof of Theorem 1. Finally, the proof of our Theorem 1 directly follows from combining Lemma 8 and Lemma 14.

5. Infeasible IPM

In this section, we briefly discuss the long-step infeasible IPM using approximate solver with our sketching-based preconditioner. Recall that such algorithms can, in general, start with an initial point that is not necessarily feasible, but the initial point does need to satisfy some, more relaxed, constraints. Following the lines of (Zhang, 1994; Monteiro and O'Neal, 2003), let $\mathcal{S}$ be the set of feasible and optimal solutions of the form $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$ for the

primal and dual problems of eqns. (1) and (2) and assume that $\mathcal{S}$ is not empty. Then, long-step infeasible IPMs can start with any initial point $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0)$ that satisfies $(\mathbf{x}^0, \mathbf{s}^0) > 0$ *and* $(\mathbf{x}^0, \mathbf{s}^0) \geq (\mathbf{x}^*, \mathbf{s}^*)$, for some feasible and optimal solution $(\mathbf{x}^*, \mathbf{s}^*) \in \mathcal{S}$. In words, the starting primal and slack variables must be strictly positive *and* larger (element-wise) when compared to some feasible, optimal primal-dual solution. See Chapter 6 of (Wright, 1997) for a discussion regarding why such choices of starting points are are relevant to computational practice and can be identified more efficiently than feasible points.

The flexibility of infeasible IPMs comes at a cost: long-step *feasible* IPMs converge in $\mathcal{O}(n \log 1/\epsilon)$ iterations, while long-step *infeasible* IPMs need $\mathcal{O}(n^2 \log 1/\epsilon)$ iterations to converge (Zhang, 1994; Monteiro and O'Neal, 2003). Here ϵ is the accuracy of the approximate LP solution returned by the IPM. Let

$$\mathbf{A}\mathbf{x}^k - \mathbf{b} = \mathbf{r}_p^k, \tag{67a}$$

$$\mathbf{A}^\mathsf{T}\mathbf{y}^k + \mathbf{s}^k - \mathbf{c} = \mathbf{r}_d^k, \tag{67b}$$

where $\mathbf{r}_p^k \in \mathbb{R}^n$ and $\mathbf{r}_d^k \in \mathbb{R}^m$ are the *primal* and *dual* residuals, respectively that characterize how far the iterate $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k)$ is from being feasible.

As long-step infeasible IPM algorithms iterate and update the primal and dual solutions, the residuals are updated as well. In case of convergence, these residuals $\mathbf{r}_p^k$ and $\mathbf{r}_d^k$ are reduced at the same rate as the duality measure μ_k and eventually converge to zero (Wright, 1997). Let $\mathbf{r}^k = (\mathbf{r}_p^k, \mathbf{r}_d^k) \in \mathbb{R}^{n+m}$ be the primal and dual residual at the k-th iteration: it is well-known that the convergence analysis of infeasible long-step IPMs critically depends on $\mathbf{r}^k$ lying on the line segment between 0 and $\mathbf{r}^0$ *i.e.*, the initial residual. Unfortunately, using approximate solvers for the normal equations violates this invariant. Similar to the feasible case, a simple solution to fix this problem by adding a perturbation vector $\mathbf{v}$ to the current primal-dual solution that guarantees that the invariant is satisfied is proposed in (Monteiro and O'Neal, 2003). In this case, the following modified system is slightly different from eqns. (24)-(26), as it now involves the residuals[7] $\mathbf{r}_p^k$ and $\mathbf{r}_d^k$:

$$\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y} = \mathbf{A}\mathbf{S}^{-1}\mathbf{v} + \mathbf{p} \tag{68a}$$

$$\hat{\Delta}\mathbf{x} = -\mathbf{x} + \sigma\mu\mathbf{S}^{-1}\mathbf{1}_n - \mathbf{D}^2\hat{\Delta}\mathbf{s} - \mathbf{S}^{-1}\mathbf{v} \tag{68b}$$

$$\hat{\Delta}\mathbf{s} = -\mathbf{r}_d - \mathbf{A}^\mathsf{T}\hat{\Delta}\mathbf{y}\,, \tag{68c}$$

where the expression of $\mathbf{p}$ is now slightly different from eqn. (23) due to infeasibility and is given by, $\mathbf{p} = -\mathbf{r}_p - \sigma\mu\mathbf{A}\mathbf{S}^{-1}\mathbf{1}_n + \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{D}^2\mathbf{r}_d$. Again, we use the exact same sketching-based construction of $\mathbf{v}$ that provably satisfies the invariant. Next, we present our main theorem for long-step infeasible IPM:

Theorem 15. *Let $0 \leq \epsilon \leq 1$ be an accuracy parameter. Consider the long-step infeasible IPM Algorithm 3 that solves eqn. (5) using the CG or Chebyshev iteration of Algorithm 1 (Section 3). Assume that the iterative solver runs with accuracy parameter $\zeta = 1/2$ and iteration count $t = \mathcal{O}(\log n)$. Then, with probability at least 0.9, the long-step infeasible IPM converges after $\mathcal{O}(n^2 \log 1/\epsilon)$ iterations.*

7. For notational simplicity, we drop the index k.

Before presenting the infeasible IPM algorithm, we will need the following definition for the neighborhood. The involvement of the residuals $\mathbf{r}^k$ makes it different from the one in the feasible case:

$$\mathcal{N}(\gamma) = \left\{ (\mathbf{x}, \mathbf{y}, \mathbf{s}) : (\mathbf{x}, \mathbf{s}) > \mathbf{0}, x_i s_i \geq (1-\gamma)\mu \text{ and } \frac{\|\mathbf{r}\|_2}{\|\mathbf{r}^0\|_2} \leq \frac{\mu}{\mu_0} \right\}.$$

Notice that Lemma 6 also holds for our infeasible IPM. The only difference is the expression of the vector $\mathbf{p}$ which now contains the residuals. Combining a result from (Monteiro and O'Neal, 2003) with our preconditioner $\mathbf{Q}^{-1/2}$, we can prove that $\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 = \mathcal{O}(n)\sqrt{\mu}$. Again, it is to be noted that the above bound is worse than Lemma 7 by a factor of $\sqrt{n}$. This bound allows us to prove that if we run Algorithm 1 for $\mathcal{O}(\log n)$ iterations, then $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma}{4}\mu$. However, the extra $\sqrt{n}$ factor essentially contributes to the $\mathcal{O}(n^2)$ iteration complexity of Algorithm 3. See Appendix A for details.

Algorithm 3 Infeasible IPM

Input: $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$, $\gamma \in (0,1)$, tolerance $\epsilon > 0$, centering parameter $\sigma \in (0, 4/5)$;

Initialize: $k \leftarrow 0$; initial point $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0)$;

1: **while** $\mu_k > \epsilon$ **do**
2: Compute sketching matrix $\mathbf{W} \in \mathbb{R}^{n \times w}$ (Section 2) with $\zeta = 1/2$ and $\delta = O(n^{-2})$;
3: Compute $\mathbf{r}_p^k = \mathbf{A}\mathbf{x}^k - \mathbf{b}$; $\mathbf{r}_d^k = \mathbf{A}^\mathsf{T}\mathbf{y}^k + \mathbf{s}^k - \mathbf{c}$; and $\mathbf{p}^k$ from eqn. (67);
4: Solve the linear system of eqn. (5) for $\mathbf{z}$ using Algorithm 1 with $\mathbf{W}$ from step (2) and $t = \mathcal{O}(\log n)$. Compute $\hat{\Delta\mathbf{y}} = \mathbf{Q}^{-1/2}\mathbf{z}$;
5: Compute $\mathbf{v}$ using eqn. (27) with $\mathbf{W}$ from step (2); $\hat{\Delta\mathbf{s}}$ using eqn. (68c); $\hat{\Delta\mathbf{x}}$ using eqn. (68b);
6: Compute $\tilde{\alpha} = \text{argmax}\{\alpha \in [0,1] : (\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k) + \alpha(\hat{\Delta\mathbf{x}}^k, \hat{\Delta\mathbf{y}}^k, \hat{\Delta\mathbf{s}}^k) \in \mathcal{N}(\gamma)\}$.
7: Compute $\bar{\alpha} = \text{argmin}\{\alpha \in [0, \tilde{\alpha}] : (\mathbf{x}^k + \alpha\hat{\Delta\mathbf{x}}^k)^\mathsf{T}(\mathbf{s}^k + \alpha\hat{\Delta\mathbf{s}}^k)\}$.
8: Compute $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}) = (\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k) + \bar{\alpha}(\hat{\Delta\mathbf{x}}^k, \hat{\Delta\mathbf{y}}^k, \hat{\Delta\mathbf{s}}^k)$; set $k \leftarrow k+1$;
9: **end while**

Notice that as compared to the feasible IPM *i.e.* Algorithm 2, Algorithm 3 needs an additional step to compute the primal and dual residuals, namely, $\mathbf{r}_p$ and $\mathbf{r}_d$ respectively (see Step (3)). However, per iteration cost of Algorithm 3 is asymptotically the same as that of Algorithm 2 (see Section 4) since computing $\mathbf{r}_p$ and $\mathbf{r}_d$ only involve a matrix-vector product and therefore, are dominated by the SVD of $\mathbf{ADW}$ and the computation of the perturbation vector $\mathbf{v}$. See Appendix A for the convergence analysis of Algorithm 3.

6. Extensions

We briefly discuss extensions of our work. Note that we focus only on analyzing preconditioned CG and preconditioned Chebyshev iteration due to their practical advantages over other solvers. In addition, Chebyshev iteration also offers several advantages in a parallel environment as it does not need to evaluate communication-intensive inner products for computing the recurrence parameters. However, from a theoretical perspective, in

(Chowdhury et al., 2020), we analyzed two more solvers, namely, preconditioned Richardson Iteration and the preconditioned Steepest Descent that could replace the proposed CG or Chebyshev iteration without any loss in accuracy or any increase in the number of iterations for the long-step feasible IPM Algorithm 2 of Section 4.

Second, recall that our approach focused on full rank input matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $m \ll n$. Our overall approach still works if $\mathbf{A}$ is any $m \times n$ matrix that is low-rank, e.g., $\text{rank}(\mathbf{A}) = k \ll \min\{m, n\}$. In that case, using the thin SVD of $\mathbf{A}$, we can rewrite the linear constraints as follows $\mathbf{U_A}\mathbf{\Sigma_A}\mathbf{V_A^\mathsf{T}}\mathbf{x} = \mathbf{b}$, where $\mathbf{U_A} \in \mathbb{R}^{m\times k}$ and $\mathbf{V_A} \in \mathbb{R}^{n\times k}$ are the matrices of left and right singular vectors of $\mathbf{A}$ respectively; $\mathbf{\Sigma_A} \in \mathbb{R}^{k\times k}$ is the diagonal matrix with the k non-zero singular values of $\mathbf{A}$ as its diagonal elements. The LP of eqn. (1) can be restated as

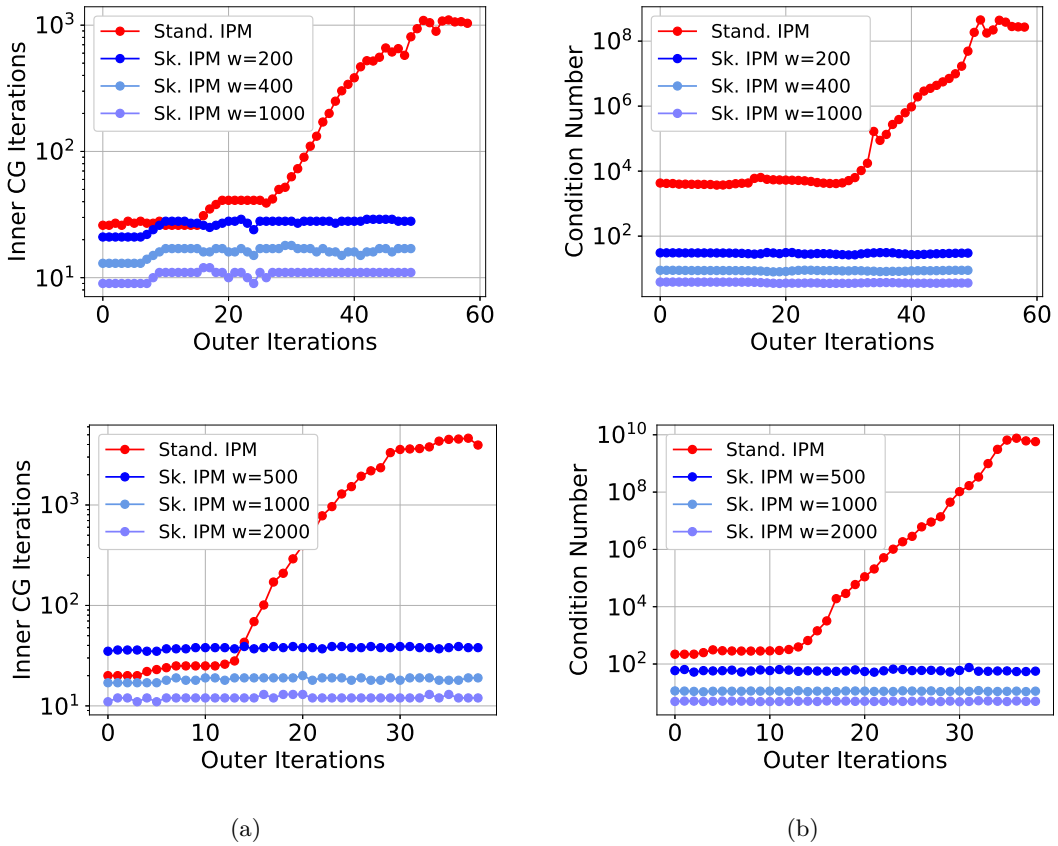
$$\min \mathbf{c}^\mathsf{T}\mathbf{x}\,,\ \text{subject to } \mathbf{V_A^\mathsf{T}}\mathbf{x} = \widetilde{\mathbf{b}}\,, \mathbf{x} \geq \mathbf{0}\,, \tag{69}$$

where $\widetilde{\mathbf{b}} = \mathbf{\Sigma_A^{-1}}\mathbf{U_A^\mathsf{T}}\mathbf{b}$. Note that, $\text{rank}(\mathbf{V_A}) = k \ll n$ and therefore eqn. (69) can be solved using our framework. The matrices $\mathbf{U_A}$, $\mathbf{V_A}$, and $\mathbf{\Sigma_A}$ can be approximately recovered using the fast SVD algorithms of (Halko et al., 2011; Boutsidis et al., 2014; Clarkson and Woodruff, 2017). However, the accuracy of the final solution will depend on the accuracy of the approximate SVD and we defer this analysis to future work.

Third, even though we chose to use the Count-Min sketch and its analysis from (Cohen et al., 2016) (Section 2), there are many other alternative sketching matrix constructions that would lead to similar results. A particularly simple one is the Gaussian sketching matrix $\mathbf{W}_G \in \mathbb{R}^{n\times w}$, where every entry is a $\mathcal{N}(0, 1)$ random variable. Setting $w = \mathcal{O}\left(m+\log(1/\delta)/\zeta^2\right)$ would result in the same accuracy guarantees as the sketching matrix of Section 2. However, the (theoretical) running time needed to compute $\mathbf{ADW}$ increases to $\mathcal{O}(m \cdot \mathsf{nnz}(\mathbf{A}))$. In practice, at least for relatively small matrices, using Gaussian sketching matrices is a reasonable alternative; see the discussion in (Meng et al., 2014) which argued that the Gaussian matrix sketching-based solvers are considerably better than direct solvers. We also opted to use Gaussian matrices in our empirical evaluation, since we primarily interested in measuring the accuracy of the final solution as a function of the number of iterations of the solver and the IPM algorithm. Other known constructions of sketching matrices that are also applicable in our setting include (any) sub-gaussian sketching matrix; the Subsampled Randomized Hadamard transform (SRHT); and any of the Sparse Subspace Embeddings of (Clarkson and Woodruff, 2017; Nelson and Nguyên, 2013; Meng and Mahoney, 2013; Cohen, 2016).

7. Experiments

Here we demonstrate the empirical performance of our algorithm on a variety of real-world data sets from the UCI ML Repository (Dua and Graff, 2017). More specifically, we consider two problems that were part of the NeurIPS 2003 feature selection challenge: ARCENE and DEXTER (Guyon et al., 2005). For the ARCENE data set, the task is to distinguish between cancer and normal patterns from mass-spectrometric data and DEXTER data set is for a text classification problem. Further, we consider DrivFace (Diaz-Chito et al., 2016), a problem concerned with identifying the gaze direction in photos of human subjects taken while driving, and a gene expression cancer RNA-Sequencing data set, accessible on the

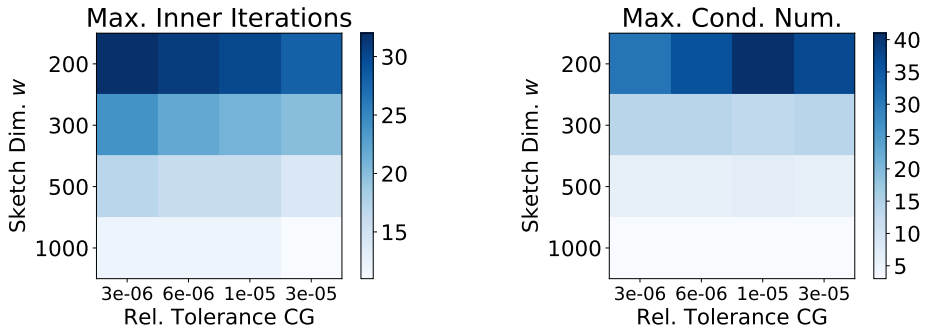


(a) (b)

Figure 1: *ARCENE (top row) and DEXTER (bottom row) data sets*: Our algorithm (Sk. IPM) requires an order of magnitude fewer inner iterations than the Standard IPM with CG at each outer iteration, as demonstrated in (a). This is possibly due to the improved conditioning of $\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}$ compared to $\mathbf{A}\mathbf{D}^2\mathbf{A}^T$, as shown in (b). For all experiments $tolCG = 10^{-5}$ and $\tau = 10^{-9}$.

UCI ML Repository, which is part of the RNA-Seq (HiSeq) PANCAN data set (Weinstein et al., 2013). It is a random extraction of gene expressions from patients who have different types of tumors: BRCA, KIRC, COAD, LUAD and PRAD. We considered the binary classification task of identifying BRCA versus other types. We also perform experiments on synthetic data sets (see Appendix B.2 for details). The experiments were implemented in Python and we observed that the results for both synthetic data (generated as described in Appendix B.2) and real-world data were qualitatively similar. Thus, we highlight results on several representative real-world datasets. The experiments were implemented in Python and run on a server with Intel E5-2623V3@3.0GHz 8 cores and 64GB RAM.

As an application, we consider ℓ_1-regularized SVMs. All of the data sets are concerned with binary classification with $m \ll n$, where n is the number of features. The SVM problem is a core model in machine learning that is crucial for applications in both regression and



(a) Max. Inner CG Iterations.

(b) Max. Condition Number.

Figure 2: *ARCENE data set*: for various (w, *tolCG*) settings, (a) the maximum number of inner iterations used by our algorithm and (b) the maximum condition number of $\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}$, across outer iterations. The standard IPM, across all settings, needed on the order of 1,000 iterations and $\kappa(\mathbf{A}\mathbf{D}^2\mathbf{A}^T)$ was on the order of 10^8. The relative error was fixed to 0.04%.

classification. While there are many variations of SVMs, we use the classical version of SVMs with an ℓ_1 regularizer to illustrate the application of our algorithm. In Appendix B.1, we describe the ℓ_1-SVM problem and how it can be formulated as an LP. Here, m is the number of training points, n is the feature dimension, and the size of the constraint matrix in the LP becomes $m \times (2n+1)$.

Comparisons and Metrics. Our empirical evaluations serve as a proof-of-concept verifying our theoretical findings, by evaluating the effectiveness of our randomized preconditioner combined with an approximate solver. State-of-the-art implementations of LP solvers are highly optimized; therefore, it is unlikely to get a fair time-comparison between our algorithm and industrial-grade solvers, since the true algorithmic efficiency of commercial solvers is confounded by the built-in optimization strategies. We do not report running times to avoid such direct comparisons with heavily optimized benchmark LP solvers.

In most of our evaluations, we use the infeasible case *i.e.*, Algorithm 3 as finding a strictly feasible staring point is a non-trivial task. In addition, we focus on CG iterative solver to compute the approximate search directions. We compare Algorithm 3 with a standard IPM (see (Press et al., 2007, Ch. 10)) using CG, and a standard IPM using a direct solver. We also use CVXPY as a benchmark to compare the accuracy of the solutions; we define the *relative error* $\|\hat{\mathbf{x}}-\mathbf{x}^\star\|_2/\|\mathbf{x}^\star\|_2$, where $\hat{\mathbf{x}}$ is our solution and $\mathbf{x}^\star$ is the solution generated by CVXPY. In addition, in some of the experiments, we also consider the primal-dual error $\mathbf{c}^\mathsf{T}\mathbf{x}-\mathbf{b}^\mathsf{T}\mathbf{y}$ as a key metric to evaluate the quality of the solution returned by our algorithm. We also consider the number of *outer iterations*, namely the number of iterations of the IIPM algorithm, as well as the number of *inner iterations*, namely the number of iterations of the CG solver. The *inner iterations* is highly dependent on the condition number of the matrices in the normal equations (eqn. (3) or (4)), which we also report. We denote the relative stopping tolerance for CG by *tolCG* and we denote the outer iteration residual

error by τ. If not specified: $\tau = 10^{-9}$, $tolCG = 10^{-5}$, and $\sigma = 0.5$. We evaluated a Gaussian sketching matrix, and the initial triplet $(\mathbf{x}, \mathbf{y}, \mathbf{s})$ for all IPM algorithms was set to be all ones.

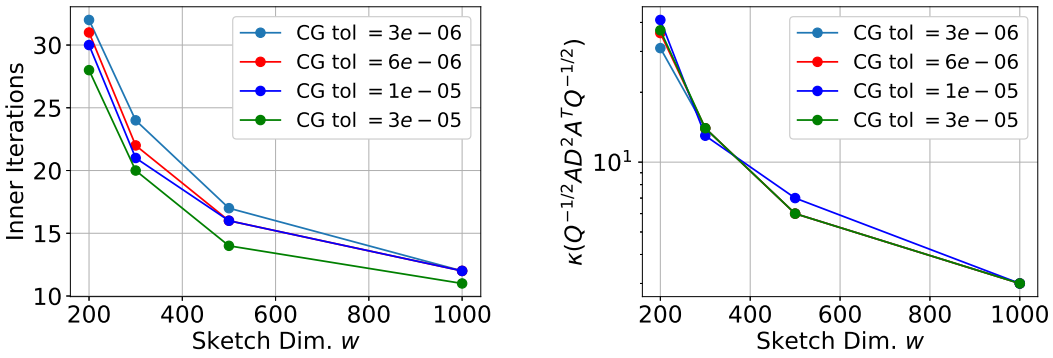
Table 1: Comparison of (our) sketched IPM with CG, standard IPM with CG, and Standard IPM with a direct solver, for the ℓ_1-SVM problem on UCI Machine Learning Repository (Dua and Graff, 2017) data sets. Across all, $\tau = 10^{-9}$ and a relative error of 10^{-3} or less was achieved. We define $\kappa_{\text{Sk}} = \kappa(\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2})$ and $\kappa_{\text{Stan}} = \kappa(\mathbf{A}\mathbf{D}^2\mathbf{A}^T)$.

Problem	**Size**	**Sketch IPM w/ Precond. CG**				**Stand. IPM w/ Unprec. CG**			**IPM w/ Dir.**
	$(m \times N)$	w	In. It.	Out. It.	κ_{Sk}	In. It.	Out. It.	κ_{Stan}	Out. It.
ARCENE	$(100 \times 10K)$	200	**30**	50	38.09	**1.1K**	59	4.4×10^8	50
DEXTER	$(300 \times 20K)$	500	**39**	39	75.42	**4.6K**	39	7.6×10^9	39
DrivFace	(606×6400)	1000	**50**	42	68.87	**139K**	43	17×10^{12}	42
Gene RNA	(801×20531)	2000	**27**	44	20.03	**101K**	208	4.7×10^{12}	44

Experimental Results. Figure 1(a) shows that our Algorithm 2 uses an order of magnitude fewer *inner* iterations than the un-preconditioned standard solver. This is due to the improved conditioning of the respective matrices in the normal equations, as demonstrated in Figure 1(b). Across various real-world and synthetic data sets, the results were qualitatively similar to those shown in Figure 1. Results for several real-world data sets are summarized in Table 1.

In general, our preconditioned CG solver used in Algorithm 2 does not increase the total number of *outer* iterations as compared to the standard IPM with CG, and the standard IPM with a direct linear solver (denoted IPM w/Dir), as seen in Table 1. Actually, for unpreconditioned CG there is clearly more outer iterations, especially for Gene RNA, which has x5 outer iterations. Figure 1 also demonstrates the relative insensitivity to the choice of w (the sketching dimension, i.e., the number of columns of the sketching matrix $\mathbf{W}$ of Section 2). For smaller values of w, our algorithm requires more inner iterations. However, across various choices of w, the number of inner iterations is always an order of magnitude smaller than the number required by the standard solver.

Figure 2 shows the performance of our algorithm for a range of (w, $tolCG$) pairs. Figure 2(a) demonstrates that the number of the inner iterations is robust to the choice of $tolCG$ and w. The number of inner iterations varies between 15 and 35 for the ARCENE data set, while the standard IPM took on the order of $1,000$ iterations across all parameter settings. Across all settings, the relative error was fixed at 0.04%. In general, our sketched IPM is able to produce an extremely high accuracy solution across parameter settings. Thus we do not report additional numerical results for the relative error, which was consistently 10^{-3} or less. Figure 2(b) demonstrates a trade-off of our approach: as both $tolCG$ and w are increased, the condition number $\kappa(\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2})$ decreases, corresponding to better conditioned systems. As a result, fewer inner iterations are required. In this context, Figure 3 shows that how the number of inner CG iterations (Figure 3(a)) or the condi-



(a) (b)

Figure 3: *ARCENE data set*: As w increases, (a) the number of inner iterations decreases, and is relatively robust to *tolCG*and (b) the condition number decreases as well.

tion number of $\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\mathbf{Q}^{-1/2}$ (Figure 3(b)) decreases with the increase in sketching dimension w for various $tolCG$.

Next, we further evaluate the performance of our algorithm in terms of the total number of *outer iterations* when starting from an infeasible point vs. starting from a strictly feasible point. The objective is to study the effect of feasibility in convergence of our algorithms. For the feasible IPM, we assume that we already have a strictly feasible starting point. See Appendix B.4 on how to find a strictly feasible point of an LP. Figure 4 shows that if we already know a feasible starting point beforehand, then, across all the data sets, our algorithm indeed takes much fewer number of outer iterations as compared to that of infeasible-start IPM. Additionally, starting from a feasible or an infeasible point does not seem to affect the rate with which the primal-dual error decreases

Finally, we run our IPM solver without a *v-correction i.e.*, without using the perturbation vector $\mathbf{v}$ in step (5) of Algorithm 3 and notice that our algorithm still converges without significantly changing the inner or outer iteration counts (see Table 2). We leave the corresponding theoretical analysis for future work. We use the same tolerance parameters and sketching dimension as in Table 1.

8. Conclusions and Open Problems

We proposed and analyzed a long-step IPM algorithm (both feasible and infeasible) using a preconditioned conjugate gradient solver for the normal equations and a novel perturbation vector to correct for the error due to the approximate solver. Thus, we speed up each iteration of the IPM algorithm, without increasing the overall number of iterations. We demonstrate empirically that our IPM requires an order of magnitude fewer inner iterations within each linear solve than standard IPMs.

Several important questions remain open. First of all, from a theoretical perspective, using the vector $\mathbf{v}$ to correct for infeasibility was necessary for our theoretical analysis, but,

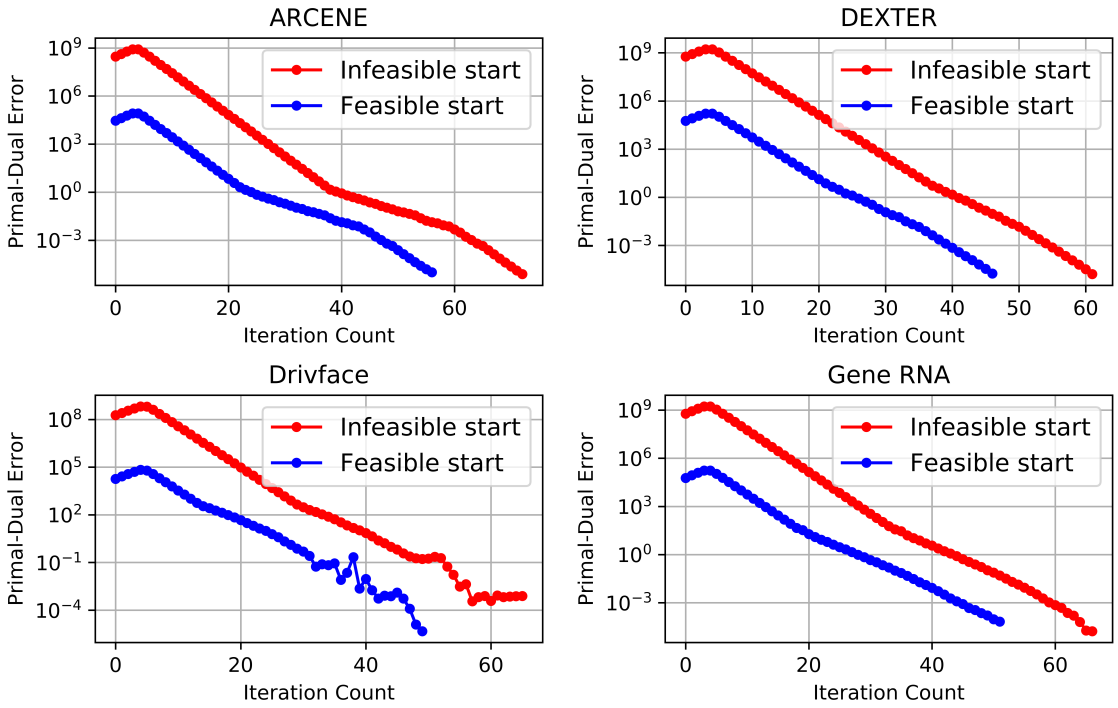


Figure 4: *Feasible vs. infeasible start*: For all four data sets, we see that the algorithm takes much fewer number of outer iteration if one can start from a feasible point. Here, we take $w = 1000$. Primal-dual errors are in log scale.

Table 2: Comparison of (our) sketched IPM with and without correction, for the ℓ_1-SVM problem on UCI Machine Learning Repository (Dua and Graff, 2017) data sets. Across all, $\tau = 10^{-9}$ and a relative error of 10^{-3} or less was achieved.

Problem	**Size**	**Sketch Size**	**Precond. Sketch IPM without Correction**			**Precond. Sketch IPM with Correction**		
	$(m \times N)$	w	Max In. It.	Sum In. It.	Out. It.	Max In. It.	Sum In. It.	Out. It.
ARCENE	$(100 \times 10K)$	200	29	1868	73	29	1873	73
DEXTER	$(300 \times 20K)$	500	40	2307	62	40	2271	62
DrivFace	(606×6400)	1000	52	2820	66	50	2804	66
Gene RNA	(801×20531)	2000	27	1445	67	27	1434	67

from an empirical perspective, we observed that the correction was not needed. A theoretical analysis of long-step IPMs without a correction vector would be of interest. Second, it would be interesting to explore whether there are other ways to use the preconditioner to design a feasible step instead of the **v**-correction. Third, a thorough empirical evaluation of the effect of preconditioning and approximate solvers, with or without the **v**-correction, would be a significant undertaking in future work. Finally, it would be interesting to investigate

what other theoretically impactful ideas could be used to efficiently solve linear program in practice. There exists barriers to using methods such as *inverse maintenance* and *lazy updates* in practice, as discussed in Section 1.2. However, it is unknown whether these issues are fundamental or avoidable.

Acknowledgments

We thank the anonymous reviewers for their comments and suggestions which helped us improve our work. AC, PD, and GD were partially supported by NSF 1760353 and 1814041 and DOE SC0022085. HA was partially supported by BSF 2017698. PL was supported by an Amazon Graduate Fellowship in Artificial Intelligence. This work was done when AC was a graduate student in the Department of Statistics, Purdue University. Oak Ridge National Laboratory is operated by UT-Battelle LLC for the U.S. Department of Energy under contract number DEAC05-00OR22725.

Appendix A. Convergence analysis of Algorithm 3

The proofs of long-step feasible IPM and long-step infeasible IPM are different from each other, since the latter needs additional assumptions on the initial iterate $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0)$. For a detailed comparison between proofs of these two variants, we refer the readers to Chapters 5 and 6 of (Wright, 1997). In the context of an approximate solver, most of the proofs related to the convergence of the long-step infeasible IPMs followed from (Monteiro and O'Neal, 2003), except for the fact that we used our sketching-based preconditioner $\mathbf{Q}^{-1/2}$, as well as our choice of the vector $\mathbf{v}$ that corrects for the error caused by the inexact solver. Here, we only prove results that are different from (Monteiro and O'Neal, 2003). For our feasible IPM proofs, there is no prior work that analyzed the theoretical aspects of long-step feasible IPMs with an approximate solver. Therefore, compared to the prototypical, long-step feasible IPM of (Wright, 1997), our proofs needed extra care in bounding the duality gap decrease in each iteration when the linear system is only approximately solved.

A.1 Number of iterations for the iterative solver

In this section, most of the proofs follow (Monteiro and O'Neal, 2003) except for the fact that we used our sketching based preconditioner $\mathbf{Q}^{-1/2}$. Recall that $\mathcal{S}$ is the set of optimal and feasible solutions for the proposed LP.

Lemma 16. *Let $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0)$ be the initial point with $(\mathbf{x}^0, \mathbf{s}^0) > \mathbf{0}$ and $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*) \in \mathcal{S}$ such that $(\mathbf{x}^*, \mathbf{s}^*) \leq (\mathbf{x}^0, \mathbf{s}^0)$ with $\mathbf{s}^0 \geq |\mathbf{A}^\mathsf{T}\mathbf{y}^0 - \mathbf{c}|$. Then, for any point $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}(\gamma)$ such that $\mathbf{r} = \eta\,\mathbf{r}^0$ and $0 \leq \eta \leq \min\left\{1, \frac{\mathbf{s}^\mathsf{T}\mathbf{x}}{\mathbf{s}^{0\mathsf{T}}\mathbf{x}^0}\right\}$, we get*

$$(i)\quad \eta\,(\mathbf{x}^\mathsf{T}\mathbf{s}^0 + \mathbf{s}^\mathsf{T}\mathbf{x}^0) \leq\ 3n\mu\,, \tag{70a}$$

$$(ii)\quad \eta\,\|\mathbf{S}(\mathbf{x}^* - \mathbf{x}^0)\|_2 \leq \eta\,\|\mathbf{S}\mathbf{x}^0\|_2 \leq \eta\mathbf{s}^\mathsf{T}\mathbf{x}^0 \leq\ 3n\mu\,, \tag{70b}$$

$$(iii)\quad \eta\,\|\mathbf{X}(\mathbf{s}^0 + \mathbf{A}^\mathsf{T}\mathbf{y}^0 - \mathbf{c})\|_2 \leq\ 2\eta\,\|\mathbf{X}\mathbf{s}^0\|_2 \leq\ 2\eta\,\mathbf{x}^\mathsf{T}\mathbf{s}^0 \leq\ 6n\mu\,. \tag{70c}$$

Proof We prove eqns. (70a)–(70c) below.

Proof of eqn. (70a)**.** For completeness, we provide a proof of eqn. (70a) following (Monteiro and O'Neal, 2003). Since $(\mathbf{x}^*, \mathbf{s}^*, \mathbf{y}^*) \in \mathcal{S}$, the following equalities hold:

$$\mathbf{A}\mathbf{x}^* = \mathbf{b} \tag{71a}$$

$$\mathbf{A}^\mathsf{T}\mathbf{y}^* + \mathbf{s}^* = \mathbf{c}. \tag{71b}$$

Furthermore, $\mathbf{r} = \eta\mathbf{r}^0$ implies

$$\mathbf{A}\mathbf{x} - \mathbf{b} = \eta(\mathbf{A}\mathbf{x}^0 - b) \tag{72a}$$

$$\mathbf{A}^\mathsf{T}\mathbf{y} + \mathbf{s} - \mathbf{c} = \eta(\mathbf{A}^\mathsf{T}\mathbf{y}^0 + \mathbf{s}^0 - \mathbf{c}). \tag{72b}$$

Combining eqn. (71a) with eqn. (72a) and eqn. (71b) with eqn. (72b), we get

$$\mathbf{A}\big(\mathbf{x} - \eta\mathbf{x}^0 - (1-\eta)\mathbf{x}^*\big) = \mathbf{0} \tag{73a}$$

$$\mathbf{A}^\mathsf{T}(\mathbf{y} - \eta\mathbf{y}^0 - (1-\eta)\mathbf{y}^*) + (\mathbf{s} - \eta\mathbf{s}^0 - (1-\eta)\mathbf{s}^*) = \mathbf{0}. \tag{73b}$$

Multiplying eqn. (73b) by $\left(\mathbf{x} - \eta\mathbf{x}^0 - (1-\eta)\mathbf{x}^*\right)^\mathsf{T}$ on the left and using eqn. (73a), we get

$$\left(\mathbf{x} - \eta\mathbf{x}^0 - (1-\eta)\mathbf{x}^*\right)^\mathsf{T}\left(\mathbf{s} - \eta\mathbf{s}^0 - (1-\eta)\mathbf{s}^*\right) = 0.$$

Expanding we get

$$\begin{aligned}\eta\left(\mathbf{x}^{0^\mathsf{T}}\mathbf{s} + \mathbf{x}^\mathsf{T}\mathbf{s}^0\right) &= \eta^2\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0 + (1-\eta)^2(\mathbf{x}^*)^\mathsf{T}\mathbf{s}^* + \mathbf{x}^\mathsf{T}\mathbf{s} \\ &\quad + \eta(1-\eta)\left(\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^* + (\mathbf{x}^*)^\mathsf{T}\mathbf{s}^0\right) - (1-\eta)\left((\mathbf{x}^*)^\mathsf{T}\mathbf{s} + \mathbf{x}^\mathsf{T}\mathbf{s}^*\right). \qquad (74)\end{aligned}$$

Next, we use the given conditions and rewrite eqn. (74) as

$$\begin{aligned}\eta\left(\mathbf{x}^{0^\mathsf{T}}\mathbf{s} + \mathbf{s}^{0^\mathsf{T}}\mathbf{x}\right) &\leq \eta^2\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0 + \mathbf{x}^\mathsf{T}\mathbf{s} + \eta(1-\eta)\left(\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^* + \mathbf{s}^{0^\mathsf{T}}\mathbf{x}^*\right) \\ &\leq \eta^2\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0 + \mathbf{x}^\mathsf{T}\mathbf{s} + 2\eta(1-\eta)\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0 \\ &\leq 2\eta\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0 + \mathbf{x}^\mathsf{T}\mathbf{s} \leq 3\mathbf{x}^\mathsf{T}\mathbf{s} = 3n\mu. \qquad (75)\end{aligned}$$

The first inequality in eqn. (75) follows from the following facts. First, $(1-\eta)((\mathbf{x}^*)^\mathsf{T}\mathbf{s} + \mathbf{x}^\mathsf{T}\mathbf{s}^*) \geq 0$ as $(\mathbf{x}^*, \mathbf{s}^*) \geq \mathbf{0}$ and $(\mathbf{x}^0, \mathbf{s}^0) \geq \mathbf{0}$. Second, as $(\mathbf{x}^*, \mathbf{s}^*, \mathbf{y}^*) \in \mathcal{S}$ (which implies $\mathbf{x}^* \circ \mathbf{s}^* = \mathbf{0}$), we have $(\mathbf{x}^*)^\mathsf{T}\mathbf{s}^* = 0$. The second inequality in eqn. (75) holds as $\mathbf{x}^* \leq \mathbf{x}^0$, $\mathbf{s}^* \leq \mathbf{s}^0$, $(\mathbf{x}^*, \mathbf{s}^*) \geq \mathbf{0}$, and $(\mathbf{x}^0, \mathbf{s}^0) \geq \mathbf{0}$; combining them we get $(\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^* + \mathbf{s}^{0^\mathsf{T}}\mathbf{x}^*) \leq 2\,\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0$. Third inequality in eqn. (75) is true as we have $\eta^2\mathbf{x}^{0^\mathsf{T}} + 2\eta(1-\eta)\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0 = 2\eta\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0 - \eta^2\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0 \leq 2\eta\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0$. The final inequality holds as $\eta \leq \frac{\mathbf{x}^\mathsf{T}\mathbf{s}}{\mathbf{x}^{0^\mathsf{T}}\mathbf{s}^0}$. □

Proof of eqn. (70b)**.** The last inequality follows from eqn. (70a). The second to last inequality is also easy to prove as

$$\|\mathbf{S}\mathbf{x}^0\|_2 = \sqrt{\sum_{i=1}^{s}(s_i x_i^0)^2} \leq \sqrt{\left(\sum_{i=1}^{s} s_i x_i^0\right)^2} = \mathbf{s}^\mathsf{T}\mathbf{x}^0\,. \qquad (76)$$

To prove the first inequality in eqn. (70b), we use the fact $\mathbf{x}^0 \geq \mathbf{x}^*$ as follows:

$$\begin{aligned}\|\mathbf{S}\mathbf{x}^0\|_2^2 - \|\mathbf{S}(\mathbf{x}^* - \mathbf{x}^0)\|_2^2 &= \sum_{i=1}^{n}(s_i x_i^0)^2 - \sum_{i=1}^{n} s_i^2\left((x_i^*)^2 + (x_i^0)^2 - 2x_i^* x_i^0\right) \\ &= \sum_{i=1}^{n} s_i^2\left(2x_i^* x_i^0 - (x_i^*)^2\right) \geq 0\,.\end{aligned}$$

□

Proof of eqn. (70c)**.** To prove this we use a similar approach as in eqn. (70b). The last inequality directly follows from eqn. (70a); the second to last inequality is also easy to prove as

$$\|\mathbf{X}\mathbf{s}^0\|_2 = \sqrt{\sum_{i=1}^{n}(x_i s_i^0)^2} \leq \sqrt{\left(\sum_{i=1}^{n} x_i s_i^0\right)^2} = \mathbf{x}^\mathsf{T}\mathbf{s}^0\,. \qquad (77)$$

For the first inequality, we proceed as follows:

$$\|\mathbf{X}(\mathbf{s}^0 + \mathbf{A}^\mathsf{T}\mathbf{y}^0 - \mathbf{c})\|_2^2 = \|\mathbf{X}\mathbf{s}^0\|_2^2 + \|\mathbf{X}(\mathbf{A}^\mathsf{T}\mathbf{y}^0 - \mathbf{c})\|_2^2 + 2\mathbf{s}^{0^\mathsf{T}}\mathbf{X}^\mathsf{T}\mathbf{X}(\mathbf{A}^\mathsf{T}\mathbf{y}^0 - \mathbf{c})$$

$$
\begin{aligned}
&= \|\mathbf{X}\mathbf{s}^0\|_2^2 + \sum_{i=1}^{n} x_i^2(\mathbf{A}^\mathsf{T}\mathbf{y}^0 - \mathbf{c})_i^2 + 2\sum_{i=1}^{n} x_i^2 s_i^0(\mathbf{A}^\mathsf{T}\mathbf{y}^0 - \mathbf{c})_i \\
&\leq \|\mathbf{X}\mathbf{s}^0\|_2^2 + \sum_{i=1}^{n}(x_i s_i^0)^2 + 2\sum_{i=1}^{n}(x_i s_i^0)^2 \\
&= \|\mathbf{X}\mathbf{s}^0\|_2^2 + \|\mathbf{X}\mathbf{s}^0\|_2^2 + 2\|\mathbf{X}\mathbf{s}^0\|_2^2 = 4\|\mathbf{X}\mathbf{s}^0\|_2^2. \qquad (78)
\end{aligned}
$$

The inequality in eqn. (78) follows from $x_i \geq 0$, $s_i^0 \geq 0$ and $\left|(\mathbf{A}^\mathsf{T}\mathbf{y}^0 - \mathbf{c})_i\right| \leq s_i^0$ for all $i = 1 \ldots n$. ■

Our next result bounds $\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2$ which is instrumental in proving the final bound.

Lemma 17. *Let* $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0)$ *be the initial point with* $(\mathbf{x}^0, \mathbf{s}^0) > \mathbf{0}$ *such that* $\mathbf{x}^0 \geq \mathbf{x}^*$ *and* $\mathbf{s}^0 \geq \max\{\mathbf{s}^*, |\mathbf{c} - \mathbf{A}^\mathsf{T}\mathbf{y}^0|\}$ *for some* $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*) \in \mathcal{S}$. *Furthermore, let* $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}(\gamma)$ *with* $\mathbf{r} = \eta\,\mathbf{r}^0$ *for some* $0 \leq \eta \leq 1$. *If the sketching matrix* $\mathbf{W} \in \mathbb{R}^{n\times w}$ *satisfies the condition in eqn.* (6)*, then*

$$
\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 \leq \sqrt{2}\left(\frac{9n}{\sqrt{1-\gamma}} + \sigma\sqrt{\frac{n}{1-\gamma}} + \sqrt{n}\right)\sqrt{\mu}\,.
$$

Recall that $\mathbf{r} = (\mathbf{r}_p, \mathbf{r}_d) = (\mathbf{A}\mathbf{x}-\mathbf{b}, \mathbf{A}^\mathsf{T}\mathbf{y}+\mathbf{s}-\mathbf{c})$ *and* $\mathbf{r}^0 = (\mathbf{r}_p^0, \mathbf{r}_d^0) = (\mathbf{A}\mathbf{x}^0-\mathbf{b}, \mathbf{A}^\mathsf{T}\mathbf{y}^0+\mathbf{s}^0-\mathbf{c})$.

Proof Note that after correcting the approximation error of the iterative solver using $\mathbf{v}$, the primal and dual residuals $\mathbf{r} = (\mathbf{r}_p, \mathbf{r}_d)$ corresponding to an iterate $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}(\gamma)$ always lie on the line segment between zero and $\mathbf{r}^{(0)}$. In other words, $\mathbf{r} = \eta\mathbf{r}^{(0)}$ always holds for some $\eta \in [0, 1]$. This was formally proven in (Monteiro and O'Neal, 2003, Lemma 3.3). In order to bound $\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2$, first we express $\mathbf{p}$ as in eqn. (3) and rewrite

$$
\mathbf{Q}^{-1/2}\mathbf{p} = \mathbf{Q}^{-1/2}\left(-\mathbf{r}_p - \sigma\mu\mathbf{A}\mathbf{S}^{-1}\mathbf{1}_n + \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{D}^2\mathbf{r}_d\right). \qquad (79)
$$

Then, applying the triangle inequality to $\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2$ in eqn. (79), we get

$$
\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 \leq \Delta_1 + \Delta_2 + \Delta_3 + \Delta_4\,, \qquad (80)
$$

where

$$
\begin{aligned}
\Delta_1 &= \|\mathbf{Q}^{-1/2}\mathbf{r}_p\|_2\,, \\
\Delta_2 &= \sigma\mu\|\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}(\mathbf{X}\mathbf{S})^{-1/2}\mathbf{1}_n\|_2\,, \\
\Delta_3 &= \|\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}\mathbf{D}^{-1}\mathbf{x}\|_2\,, \\
\Delta_4 &= \|\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}^2\mathbf{r}_d\|_2\,.
\end{aligned}
$$

To bound Δ_1, Δ_2, Δ_3 and Δ_4 we heavily use the condition of eqn. (6).

Bounding Δ_1. Using $\mathbf{r}_p = \eta\,\mathbf{r}_p^0$, $\mathbf{r}_p^0 = \mathbf{A}\mathbf{x}^0 - \mathbf{b}$ and $\mathbf{b} = \mathbf{A}\mathbf{x}^*$, we rewrite Δ_1 as

$$\begin{aligned}
\Delta_1 &= \eta\,\|\mathbf{Q}^{-1/2}\mathbf{A}(\mathbf{x}^0 - \mathbf{x}^*)\|_2 \\
&= \eta\,\|\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}\mathbf{D}^{-1}(\mathbf{x}^0 - \mathbf{x}^*)\|_2 \\
&\leq \eta\,\|\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}\|_2\|\mathbf{D}^{-1}(\mathbf{x}^0 - \mathbf{x}^*)\|_2 \\
&\leq \sqrt{2}\eta\,\|\mathbf{D}^{-1}(\mathbf{x}^0 - \mathbf{x}^*)\|_2 \\
&= \sqrt{2}\eta\,\|(\mathbf{X}\mathbf{S})^{-1/2}\mathbf{S}(\mathbf{x}^0 - \mathbf{x}^*)\|_2 \\
&\leq \sqrt{2}\eta\,\|(\mathbf{X}\mathbf{S})^{-1/2}\|_2\,\|\mathbf{S}(\mathbf{x}^0 - \mathbf{x}^*)\|_2\,, \qquad (81)
\end{aligned}$$

where the above steps follow from submultiplicativity and eqn. (6). From eqn. (6), note that we have $\|\mathbf{Q}^{-1/2}\mathbf{A}\mathbf{D}\|_2 \leq \sqrt{2}$ as $\zeta \leq 1$. Now, applying eqn. (70b) and $\|(\mathbf{X}\mathbf{S})^{-1/2}\|_2 = \max_{1\leq i\leq n}\frac{1}{\sqrt{x_i s_i}}$, we further have

$$\begin{aligned}
\Delta_1 &\leq \sqrt{2}\,\max_{1\leq i\leq n}\frac{1}{\sqrt{x_i s_i}}\cdot 3n\mu \\
&\leq 3\sqrt{2}\,n\sqrt{\frac{\mu}{1-\gamma}}\,, \qquad (82)
\end{aligned}$$

where the last inequality follows from $(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathcal{N}(\gamma)$.

Bounding Δ_2. Applying submultiplicativity, we get

$$\begin{aligned}
\Delta_2 &= \sigma\mu\,\|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\,(\mathbf{X}\mathbf{S})^{-1/2}\mathbf{1}_n\|_2 \\
&\leq \sigma\mu\,\|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\|_2\|(\mathbf{X}\mathbf{S})^{-1/2}\mathbf{1}_n\|_2 \\
&\leq \sqrt{2}\,\sigma\mu\,\|(\mathbf{X}\mathbf{S})^{-1/2}\mathbf{1}_n\|_2 \\
&= \sqrt{2}\,\sigma\mu\sqrt{\sum_{i=1}^n\frac{1}{x_i s_i}} \leq\ \sqrt{2}\,\sigma\mu\sqrt{\sum_{i=1}^n\frac{1}{(1-\gamma)\mu}} \\
&= \sqrt{2}\,\sigma\sqrt{\frac{n\,\mu}{(1-\gamma)}}\,, \qquad (83)
\end{aligned}$$

where the second to last inequality follows from eqn. (6) and the last inequality holds as $(\mathbf{x},\mathbf{y},\mathbf{s}) \in \mathcal{N}(\gamma)$.

Bounding Δ_3. Using $\mathbf{D} = \mathbf{S}^{-1/2}\mathbf{X}^{1/2}$ and $\mathbf{x} = \mathbf{X}\,\mathbf{1}_n$ we get

$$\begin{aligned}
\Delta_3 &= \|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\,(\mathbf{S}^{1/2}\mathbf{X}^{-1/2})\,\mathbf{X}\,\mathbf{1}_n\|_2 \\
&= \|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\,(\mathbf{S}\mathbf{X})^{1/2}\,\mathbf{1}_n\|_2 \\
&\leq \|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\|_2\|(\mathbf{S}\mathbf{X})^{1/2}\,\mathbf{1}_n\|_2 \\
&\leq \sqrt{2}\,\sqrt{\sum_{i=1}^n x_i s_i} =\ \sqrt{2n\,\mu}\,, \qquad (84)
\end{aligned}$$

where the inequalities follow from submultiplicativity and eqn. (6).

Bounding Δ_4. Using $\mathbf{r}_d = \eta\,\mathbf{r}_d^0$, we have

$$\begin{aligned}\Delta_4 &= \eta\|\mathbf{Q}^{-1/2}\,\mathbf{A}\,\mathbf{D}^2\mathbf{r}_d^0\|_2\\ &\le \eta\|\mathbf{Q}^{-1/2}\,\mathbf{A}\mathbf{D}\|_2\|(\mathbf{X}\mathbf{S})^{-1/2}\mathbf{X}\mathbf{r}_d^0\|_2\\ &\le \sqrt{2}\eta\,\|(\mathbf{X}\mathbf{S})^{-1/2}\mathbf{X}(\mathbf{A}^\mathsf{T}\mathbf{y}^0+\mathbf{s}^0-\mathbf{c})\|_2\\ &\le \sqrt{2}\eta\,\|(\mathbf{X}\mathbf{S})^{-1/2}\|_2\,\|\mathbf{X}(\mathbf{A}^\mathsf{T}\mathbf{y}^0+\mathbf{s}^0-\mathbf{c})\|_2\,,\end{aligned}$$

where the above inequalities follow from submultiplicativity and eqn. (6). Now, applying eqn. (70c) and $\|(\mathbf{X}\mathbf{S})^{-1/2}\|_2 \le \frac{1}{\sqrt{(1-\gamma)\mu}}$, we further have

$$\Delta_4 \le\ 6\sqrt{2}n\sqrt{\frac{\mu}{1-\gamma}}. \tag{85}$$

Final bound. Combining eqns. (80), (82), ,(83), (84) and (85), we get

$$\|\mathbf{Q}^{-1/2}\mathbf{p}\|_2 \le\ \sqrt{2}\left(\frac{9n}{\sqrt{1-\gamma}}+\sigma\sqrt{\frac{n}{1-\gamma}}+\sqrt{n}\right)\sqrt{\mu}\,. \tag{86}$$

This concludes the proof of Lemma 17. ■

A.2 Determining step-size, bounding the number of iterations, and proof of Theorem 15

Assume that the triplet $(\hat{\Delta\mathbf{x}}, \hat{\Delta\mathbf{y}}, \hat{\Delta\mathbf{s}})$ satisfies eqns. (68a), (68b), and (68c). We rewrite this system in the following alternative form:

$$\mathbf{A}\hat{\Delta\mathbf{x}} = -\,\mathbf{r}_p, \tag{87a}$$

$$\mathbf{A}^\mathsf{T}\hat{\Delta\mathbf{y}} + \hat{\Delta\mathbf{s}} = -\,\mathbf{r}_d, \tag{87b}$$

$$\mathbf{X}\hat{\Delta\mathbf{s}} + \mathbf{S}\hat{\Delta\mathbf{x}} = -\,\mathbf{X}\mathbf{S}\,\mathbf{1}_n + \sigma\mu\,\mathbf{1}_n - \mathbf{v}. \tag{87c}$$

Indeed, we now show how to derive eqns. (68b), (68a) and (68c) from eqn. (87). Pre-multiplying both sides of eqn. (87c) by $\mathbf{A}\mathbf{S}^{-1}$ and noting that $\mathbf{D}^2 = \mathbf{X}\mathbf{S}^{-1}$, we get

$$\begin{aligned}&\mathbf{A}\mathbf{D}^2\hat{\Delta\mathbf{s}} + \mathbf{A}\hat{\Delta\mathbf{x}} = -\mathbf{A}\mathbf{X}\mathbf{1}_n + \sigma\mu\mathbf{A}\mathbf{S}^{-1}\mathbf{1}_n - \mathbf{A}\mathbf{S}^{-1}\mathbf{v}\\ \Rightarrow\ &\mathbf{A}\mathbf{D}^2\hat{\Delta\mathbf{s}} = -\mathbf{A}\mathbf{x} + \mathbf{r}_p + \sigma\mu\mathbf{A}\mathbf{S}^{-1}\mathbf{1}_n - \mathbf{A}\mathbf{S}^{-1}\mathbf{v}.\end{aligned} \tag{88}$$

Eqn. (88) holds as $\mathbf{A}\mathbf{X}\mathbf{1}_n = \mathbf{A}\mathbf{x}$ and, from eqn. (87a), $\mathbf{A}\hat{\Delta\mathbf{x}} = -\mathbf{r}_p$. Next, pre-multiplying eqn. (87b) by $\mathbf{A}\mathbf{D}^2$, we get

$$\begin{aligned}&\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\hat{\Delta\mathbf{y}} + \mathbf{A}\mathbf{D}^2\hat{\Delta\mathbf{s}} = -\mathbf{A}\mathbf{D}^2\mathbf{r}_d\\ \Rightarrow\ &\mathbf{A}\mathbf{D}^2\mathbf{A}^\mathsf{T}\hat{\Delta\mathbf{y}} = -\mathbf{r}_p - \sigma\mu\mathbf{A}\mathbf{S}^{-1}\mathbf{1}_n + \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{D}^2\mathbf{r}_d + \mathbf{A}\mathbf{S}^{-1}\mathbf{v} = \mathbf{p} + \mathbf{A}\mathbf{S}^{-1}\mathbf{v}.\end{aligned} \tag{89}$$

The first equality in eqn. (89) follows from eqn. (88) and the definition of $\mathbf{p}$. This establishes eqn. (68a). Eqn. (68c) directly follows from eqn. (87b). Finally, we get eqn. (68b) by pre-multiplying eqn. (87c) by $\mathbf{S}^{-1}$.

Next, we define each new point traversed by the algorithm as $(\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{s}(\alpha))$, where

$$(\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{s}(\alpha)) = (\mathbf{x}, \mathbf{y}, \mathbf{s}) + \alpha(\hat{\Delta}\mathbf{x}, \hat{\Delta}\mathbf{y}, \hat{\Delta}\mathbf{s}), \tag{90}$$

$$\mu(\alpha) = \mathbf{x}(\alpha)^{\mathsf{T}}\mathbf{s}(\alpha)/n, \tag{91}$$

$$\mathbf{r}(\alpha) = \mathbf{r}\left(\mathbf{x}(\alpha), \mathbf{s}(\alpha), \mathbf{y}(\alpha)\right). \tag{92}$$

The goal in this section is to bound the number of iterations required by Algorithm 3. Towards that end, we bound the magnitude of the step size α. First, we provide an upper bound on α, which allows us to show that each new point $(\mathbf{x}(\alpha), \mathbf{s}(\alpha), \mathbf{y}(\alpha))$ traversed by the algorithm stays within the neighborhood $\mathcal{N}(\gamma)$. Second, we provide a lower bound on α, which allows us to bound the number of iterations required. We use multiple lemmas from (Monteiro and O'Neal, 2003), which we reproduce here, without their proofs.

First, we provide an upper bound on α, ensuring that each new point $(\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{s}(\alpha))$ traversed by the algorithm stays within the neighborhood $\mathcal{N}(\gamma)$.

Lemma 18 (Lemma 3.5 of (Monteiro and O'Neal, 2003)). *Assume $(\hat{\Delta}\mathbf{x}, \hat{\Delta}\mathbf{y}, \hat{\Delta}\mathbf{s})$ satisfies eqns. (87) for some $\sigma > 0$, $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}(\gamma)$ (for $\gamma \in (0, 1)$), and $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$. Then, $(\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{s}(\alpha)) \in \mathcal{N}(\gamma)$ for every scalar α such that*

$$0 \leq \alpha \leq \min\left\{1, \frac{\gamma\sigma\mu}{4\|\hat{\Delta}\mathbf{x} \circ \hat{\Delta}\mathbf{s}\|_\infty}\right\}. \tag{93}$$

We now provide a lower bound on the values of $\bar{\alpha}$ and the corresponding $\mu(\bar{\alpha})$; see Algorithm 3.

Lemma 19 (Lemma 3.6 of (Monteiro and O'Neal, 2003)). *In each iteration of Algorithm 3, if $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$, then the step size $\bar{\alpha}$ satisfies*

$$\bar{\alpha} \geq \min\left\{1, \frac{\min\{\gamma\sigma, (1 - \frac{5}{4}\sigma)\}\mu}{4\|\hat{\Delta}x \circ \hat{\Delta}s\|_\infty}\right\} \tag{94}$$

and

$$\mu(\bar{\alpha}) = \left[1 - \frac{\bar{\alpha}}{2}(1 - \frac{5}{4}\sigma)\right]\mu. \tag{95}$$

At this point, we have provided a lower bound (eqn. (94)) for the allowed values of the step size $\bar{\alpha}$. Next, we show that this lower bound is bounded away from zero. From eqn. (94) this is equivalent to showing that $\|\hat{\Delta}\mathbf{x} \circ \hat{\Delta}\mathbf{s}\|_\infty$ is bounded.

Lemma 20 (Lemma 3.7 of (Monteiro and O'Neal, 2003) (slightly modified)). *Let $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0)$ be the initial point with $(\mathbf{x}^0, \mathbf{s}^0) > 0$ and $(\mathbf{x}^0, \mathbf{s}^0) \geq (\mathbf{x}^*, \mathbf{s}^*)$ for some $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*) \in \mathcal{S}$. Let $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}(\gamma)$ be such that $\mathbf{r} = \eta\mathbf{r}^0$ for some $\eta \in [0, 1]$ and $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$. Then, the search direction $(\hat{\Delta}\mathbf{x}, \hat{\Delta}\mathbf{y}, \hat{\Delta}\mathbf{s})$ produced by Algorithm 3 at each iteration satisfies*

$$\max\{\|\mathbf{D}^{-1}\hat{\Delta}\mathbf{x}\|_2, \|\mathbf{D}\hat{\Delta}\mathbf{s}\|_2\} \leq \left(1 + \frac{\sigma^2}{1 - \gamma} - 2\sigma\right)^{1/2}\sqrt{n\mu} + \frac{6n}{\sqrt{(1 - \gamma)}}\sqrt{\mu} + \frac{\gamma\sigma}{4\sqrt{1 - \gamma}}\sqrt{\mu}. \tag{96}$$

We should note here that the above lemma is slightly different from (Monteiro and O'Neal, 2003, Lemma 3.7). Indeed, (Monteiro and O'Neal, 2003, Lemma 3.7) actually proves the following bound:

$$\max\{\|\mathbf{D}^{-1}\hat{\Delta}\mathbf{x}\|_2, \|\mathbf{D}\hat{\Delta}\mathbf{s}\|_2\} \leq \left(1 + \frac{\sigma^2}{1-\gamma} - 2\sigma\right)^{1/2} \sqrt{n\mu} + \frac{6n}{\sqrt{(1-\gamma)}}\sqrt{\mu} + \frac{\gamma\sigma}{4\sqrt{n}}\sqrt{\mu}\,. \tag{97}$$

Notice that there is slight difference in the last term in the right-hand side, which does not asymptotically change the bound. The underlying reason for this difference is the fact that (Monteiro and O'Neal, 2003) constructed the vector $\mathbf{v}$ differently. In our case, we need to bound $\|(\mathbf{XS})^{-1/2}\mathbf{v}\|_2$, which we do as follows:

$$\|(\mathbf{XS})^{-1/2}\mathbf{v}\|_2 \leq \|(\mathbf{XS})^{-1/2}\|_2 \, \|\mathbf{v}\|_2 \leq \frac{1}{\min_i \sqrt{x_i s_i}} \frac{\gamma\sigma\mu}{4}\,, \tag{98}$$

where in the above expression we use the fact that $\|(\mathbf{XS})^{-1/2}\|_2 = \frac{1}{\min_i \sqrt{x_i s_i}}$. Now as $(\mathbf{x}, \mathbf{y}, \mathbf{s}) \in \mathcal{N}(\gamma)$, we further have $x_i s_i \geq (1-\gamma)\mu$ for all $i = 1 \dots n$. Combining this with eqn. (98), we get

$$\|(\mathbf{XS})^{-1/2}\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4\sqrt{(1-\gamma)\mu}} = \frac{\gamma\sigma}{4\sqrt{1-\gamma}}\sqrt{\mu}. \tag{99}$$

On the other hand, (Monteiro and O'Neal, 2003) had a different construction of $\mathbf{v}$ for which $\|(\mathbf{XS})^{-1/2}\mathbf{v}\|_2 = \|\tilde{\mathbf{f}}^{(t)}\|_2$ holds. Therefore they had the following bound:

$$\|(\mathbf{XS})^{-1/2}\mathbf{v}\|_2 = \|\tilde{\mathbf{f}}^{(t)}\|_2 \leq \frac{\gamma\sigma}{4\sqrt{n}}\sqrt{\mu}.$$

The next lemma bounds the number of iterations that Algorithm 3 needs when started with an infeasible point that is sufficiently positive.

Lemma 21 (Theorem 2.6 of (Monteiro and O'Neal, 2003))**.** *Assume that the constants γ and σ are such that $\max\{\gamma^{-1}, (1-\gamma)^{-1}, \sigma^{-1}, (1-\frac{5}{4}\sigma)^{-1}\} = \mathcal{O}(1)$. Let the initial point $(\mathbf{x}^0, \mathbf{s}^0, \mathbf{y}^0)$ satisfy $(\mathbf{x}^0, \mathbf{s}^0) \geq (\mathbf{x}^*, \mathbf{s}^*)$ for some $(\mathbf{x}^*, \mathbf{s}^*, \mathbf{y}^*) \in \mathcal{S}$ and $\|\mathbf{v}\|_2 \leq \frac{\gamma\sigma\mu}{4}$. Algorithm 3 generates an iterate $(\mathbf{x}^k, \mathbf{s}^k, \mathbf{y}^k)$ satisfying $\mu_k \leq \epsilon\mu_0$ and $\|\mathbf{r}^k\|_2 \leq \epsilon\|\mathbf{r}^0\|_2$ after $\mathcal{O}(n^2 \log 1/\epsilon)$ iterations.*

Finally, Theorem 15 follows from Lemmas 8 and 21.

Appendix B. Additional notes on experiments

B.1 Support Vector Machines (SVMs)

The classical ℓ_1-SVM problem is as follows. We consider the task of fitting an SVM to data pairs $S = \{(x_i, y_i)\}_{i=1}^m$, where $x_i \in \mathbb{R}^n$ and $y_i \in \{+1, -1\}$. Here, m is the number of training points, and n is the feature dimension. The SVM problem with an ℓ_1 regularizer has the following form:

$$\underset{w}{\text{minimize}} \quad \|w\|_1 \tag{100}$$

$$\text{subject to} \quad y_i(w^T x_i + b') \geq 1, \quad i = 1 \ldots m.$$

This problem can be written as an LP by introducing the variables w^+ and w^-, where $w = w^+ - w^-$. The objective becomes $\sum_{j=1}^{n} w_j^+ + w_j^-$, and we constrain $w_i^+ \geq 0$ and $w_i^- \geq 0$. Note that the size of the constraint matrix in the LP becomes $m \times (2n + 1)$.

B.2 Random data

We generate random synthetic instances of linear programs as follows. To generate $\mathbf{A} \in \mathbb{R}^{m \times n}$, we set $a_{ij} \sim_{i.i.d.} U(0, 1)$ with probability p and $a_{ij} = 0$ otherwise. We then add $\min\{m, n\}$ i.i.d. draws from $U(0, 1)$ to the main diagonal, to ensure each row of $\mathbf{A}$ has at least one nonzero entry. We set $\mathbf{b} = \mathbf{A}\mathbf{x} + 0.1\mathbf{z}$, where $\mathbf{x}$ and $\mathbf{z}$ are random vectors drawn from $N(0, 1)$. Finally, we set $c \sim N(0, 1)$.

B.3 Real-world data

We used a gene expression cancer RNA-Sequencing dataset, taken from the UCI Machine Learning repository. It is part of the RNA-Seq (HiSeq) PANCAN data set (Weinstein et al., 2013) and is a random extraction of gene expressions from patients who have different types of tumors: BRCA, KIRC, COAD, LUAD, and PRAD. We considered the binary classification task of identifying BRCA versus other types.

We also used the DrivFace dataset taken from the UCI Machine Learning repository. In the DrivFace dataset, each sample corresponds to an image of a human subject, taken while driving in real scenarios. Each image is labeled as corresponding to one of three possible gaze directions: left, straight, or right. We considered the binary classification task of identifying two different gaze directions: straight, or to either side (left or right).

B.4 Feasible starting point

We construct a linear program to find a primal feasible starting point $(\mathbf{x}_0, \mathbf{s}_0, \mathbf{y}_0)$ such that $\mathbf{A}\mathbf{x}_0 = \mathbf{b}$. Without loss of generality, assume that all entries of $\mathbf{b}$ are positive and let $\mathbf{z} \in \mathbb{R}^m$. Then, $(\mathbf{x}, \mathbf{z})$ is an optimal solution to the following linear program when $\mathbf{z} = \mathbf{0}$ and $\mathbf{A}\mathbf{x} = \mathbf{b}$.

$$\min \mathbf{z}\,, \text{ subject to } \mathbf{A}\mathbf{x} + \mathbf{I}\mathbf{z} = \mathbf{b}\,, \mathbf{x}, \mathbf{z} \geq \mathbf{0}\,,$$

References

H. Avron, P. Maymounkov, and S. Toledo. Blendenpik: Supercharging LAPACK's Least-squares Solver. *SIAM Journal on Scientific Computing*, 32(3):1217–1236, 2010.

H. Avron, K. L. Clarkson, and D. P. Woodruff. Faster Kernel Ridge Regression using Sketching and Preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 38 (4):1116–1138, 2017.

O. Axelsson and V. A. Barker. *Finite Element Solution of Boundary Value Problems: Theory and Computation*, volume 35. Society for Industrial and Applied Mathematics, 1984.

R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 1994.

D. Bienstock and G. Iyengar. Approximating Fractional Packings and Coverings in $\mathcal{O}(1/\epsilon)$ iterations. *SIAM Journal on Computing*, 35(4):825–854, 2006.

R. E. Bixby, J. W. Gregory, I. J. Lustig, R. E. Marsten, and D. F. Shanno. Very Large-scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods. *Operations Research*, 40(5):885–897, 1992.

C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal Column-based Matrix Reconstruction. *SIAM Journal on Computing*, 43(2):687–717, 2014.

R. Bouyouli, G. Meurant, L. Smoch, and H. Sadok. New Results on the Convergence of the Conjugate Gradient Method. *Numerical Linear Algebra with Applications*, 16(3):223–236, 2009.

J. v. d. Brand. A Deterministic Linear Program Solver in Current Matrix Multiplication Time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 259–278. SIAM, 2020.

J. v. d. Brand, Y. T. Lee, A. Sidford, and Z. Song. Solving Tall Dense Linear Programs in Nearly Linear Time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, page 775–788, 2020.

S. Chen, Y. Liu, M. R. Lyu, I. King, and S. Zhang. Fast Relative-Error Approximation Algorithm for Ridge Regression. In *UAI*, pages 201–210, 2015.

A. Chowdhury, J. Yang, and P. Drineas. An Iterative, Sketching-based Framework for Ridge Regression. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 988–997, 2018.

A. Chowdhury, J. Yang, and P. Drineas. Randomized Iterative Algorithms for Fisher Discriminant Analysis. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 239–249. PMLR, 2019.

A. Chowdhury, P. London, H. Avron, and P. Drineas. Faster Randomized Infeasible Interior Point Methods for Tall/Wide Linear Programs. In *Advances in Neural Information Processing Systems*, volume 33, pages 8704–8715, 2020.

K. L. Clarkson and D. P. Woodruff. Low-rank Approximation and Regression in Input Sparsity Time. *Journal of the ACM*, 63(6):54, 2017.

M. B. Cohen. Nearly Tight Oblivious Subspace Embeddings by Trace Inequalities. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 278–287, 2016.

M. B. Cohen, J. Nelson, and D. P. Woodruff. Optimal Approximate Matrix Product in Terms of Stable Rank. In *43rd International Colloquium on Automata, Languages, and Programming*, pages 11:1–11:14, 2016.

M. B. Cohen, A. Mądry, P. Sankowski, and A. Vladu. Negative-weight Shortest Paths and Unit Capacity Minimum Cost Flow in $\tilde{\mathcal{O}}(m^{10/7} \log W)$ time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 752–771. SIAM, 2017.

M. B. Cohen, Y. T. Lee, and Z. Song. Solving Linear Programs in the Current Matrix Multiplication Time. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing*, pages 938–942, 2019.

M. B. Cohen, Y. T. Lee, and Z. Song. Solving Linear Programs in the Current Matrix Multiplication Time. *Journal of the ACM (JACM)*, 68(1):1–39, 2021.

G. Cormode and C. Dickens. Iterative Hessian Sketch in Input Sparsity Time. In *NeurIPS Workshop: Beyond First-Order Optimization Methods in Machine Learning*, 2019.

Y. Cui, K. Morikuni, T. Tsuchiya, and K. Hayami. Implementation of Interior-point Methods for LP based on Krylov Subspace Iterative Solvers with Inner-iteration Preconditioning. *Computational Optimization and Applications*, 74(1):143–176, 2019.

Y. Dahiya, D. Konomis, and D. P. Woodruff. An Empirical Evaluation of Sketching for Numerical Linear Algebra. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1292–1300, 2018.

S. I. Daitch and D. A. Spielman. Faster Approximate Lossy Generalized Flow via Interior Point Algorithms. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 451–460, 2008.

G. B. Dantzig. Maximization of a Linear Function of Variables Subject to Linear Inequalities. *Activity analysis of production and allocation*, 13:339–347, 1951.

K. Diaz-Chito, A. Hernández-Sabaté, and A. M. López. A Reduced Feature Set for Driver Head Pose Estimation. *Applied Soft Computing*, 45:98–107, 2016. doi: https://doi.org/10.1016/j.asoc.2016.04.027.

D. L. Donoho and J. Tanner. Sparse Nonnegative Solution of Underdetermined Linear Equations by Linear Programming. In *Proceedings of the National Academy of Sciences of the United States of America*, pages 9446–9451, 2005.

P. Drineas and M. W. Mahoney. RandNLA: Randomized Numerical Linear Algebra. *Communications of the ACM*, 59(6):80–90, 2016.

P. Drineas and M. W. Mahoney. *Lectures on randomized numerical linear algebra*, volume 25 of *The Mathematics of Data, IAS/Park City Mathematics Series*. American Mathematical Society, 2018.

D. Dua and C. Graff. UCI Machine Learning Repository, 2017. URL `http://archive.ics.uci.edu/ml`.

D. C.-L. Fong and M. Saunders. CG vs. MINRES: An Empirical Comparison. *Sultan Qaboos University Journal for Science*, 17(1):44–62, 2012.

T. Glavelis, N. Ploskas, and N. Samaras. Improving a Primal–dual Simplex-type Algorithm using Interior Point Methods. *Optimization*, 67(12):2259–2274, 2018.

G. H. Golub and C. F. Van Loan. *Matrix Computations*, volume 4. JHU press, 2013.

J. Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012.

M. H. Gutknecht. *Software for Numerical Linear Algebra*, volume 2. ETH Zurich, 2008. URL `http://www.sam.math.ethz.ch/~mhg/unt/SWNLA/itmethSWNLA08.pdf`.

M. H. Gutknecht and S. Röllin. The Chebyshev Iteration Revisited. *Parallel Computing*, 28(2):263–283, 2002.

I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result Analysis of the NIPS 2003 Feature Selection Challenge. In *Advances in Neural Information Processing Systems*, pages 545–552, 2005.

N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2):217–288, 2011.

S. Jiang, Z. Song, O. Weinstein, and H. Zhang. A Faster Algorithm for Solving General LPs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 823–832, 2021.

N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

L. G. Khachiyan. A Polynomial Algorithm in Linear Programming. In *Doklady Akademii Nauk*, volume 244, pages 1093–1096. Russian Academy of Sciences, 1979.

V. Klee and G. J. Minty. How Good is the Simplex Algorithm. *Inequalities*, 3(3):159–175, 1972.

Y. T. Lee and A. Sidford. Path Finding Methods for Linear Programming: Solving Linear Programs in $\tilde{O}(\sqrt{rank})$ Iterations and Faster Algorithms for Maximum Flow. In *Proceedings of the 55th IEEE Symposium on Foundations of Computer Science*, pages 424–433, 2014.

Y. T. Lee and A. Sidford. Efficient Inverse Maintenance and Faster Algorithms for Linear Programming. In *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science*, pages 230–249, 2015.

Y. T. Lee, Z. Song, and Q. Zhang. Solving Empirical Risk Minimization in the Current Matrix Multiplication Time. In *Conference on Learning Theory*, pages 2140–2157, 2019.

P. London, S. Vardi, A. Wierman, and H. Yi. A Parallelizable Acceleration Framework for Packing Linear Programs. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 3706 – 3713, 2018.

D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. Springer Publishing Company, Incorporated, 3rd edition, 2008. ISBN 3319188410.

A. Madry. Navigating Central Path with Electrical Flows: From Flows to Matchings, and Back. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 253–262. IEEE, 2013.

A. Madry. Computing Maximum Flow with Augmenting Electrical Flows. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 593–602. IEEE, 2016.

M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011.

P.-G. Martinsson and J. Tropp. Randomized Numerical Linear Algebra: Foundations & algorithms. *arXiv preprint arXiv:2002.01387*, 2020.

X. Meng and M. W. Mahoney. Low-distortion Subspace Embeddings in Input-sparsity Time and Applications to Robust Linear Regression. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 91–100, 2013.

X. Meng, M. A. Saunders, and M. W. Mahoney. LSRN: A parallel iterative solver for strongly over- or underdetermined systems. *SIAM Journal on Scientific Computing*, 36 (2):95–118, 2014.

O. Meshi and A. Globerson. An Alternating Direction Method for Dual MAP LP Relaxation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 470–483. Springer, 2011.

R. D. C. Monteiro and J. W. O'Neal. Convergence Analysis of a Long-step Primal-dual infeasible Interior-point LP Algorithm based on Iterative Linear Solvers. *Georgia Institute of Technology*, 2003.

R. D. C. Monteiro, J. W. O'Neal, and T. Tsuchiya. Uniform boundedness of a preconditioned normal matrix used in interior-point methods. *SIAM Journal on Optimization*, 15(1):96–100, 2004.

K. Morikuni and K. Hayami. Inner-iteration Krylov Subspace Methods for Least Squares Problems. *SIAM Journal on Matrix Analysis and Applications*, 34(1):1–22, 2013.

K. Morikuni and K. Hayami. Convergence of Inner-iteration GMRES Methods for Rank-deficient Least Squares Problems. *SIAM Journal on Matrix Analysis and Applications*, 36(1):225–250, 2015.

J. Nelson and H. L. Nguyên. OSNAP: Faster Numerical Linear Algebra Algorithms via Sparser Subspace Embeddings. In *Proceedings of the 54th IEEE Symposium on Foundations of Computer Science*, pages 117–126, 2013.

J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

C. C. Paige and M. A. Saunders. Solution of Sparse Indefinite Systems of Linear Equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

M. Pilanci and M. J. Wainwright. Newton sketch: A Near Linear-time Optimization Algorithm with Linear-quadratic Convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical recipes 3rd edition: The art of scientific computing. In *The Oxford Handbook of Innovation*, chapter 10. Cambridge University Press, 2007.

B. Recht, C. Re, J. Tropp, and V. Bittorf. Factoring Nonnegative Matrices with Linear Programs. In *Advances in Neural Information Processing Systems*, pages 1214–1222, 2012.

J. Renegar. A Polynomial-time Algorithm, Based on Newton's Method, for Linear Programming. *Mathematical programming*, 40(1):59–93, 1988.

M. G. C. Resende and G. Veiga. An Implementation of the Dual Affine Scaling Algorithm for Minimum-cost Flow on Bipartite Uncapacitated Networks. *SIAM Journal on Optimization*, 3(3):516–537, 1993.

Z. Song and Z. Yu. Oblivious Sketching-based Central Path Method for Linear Programming. In *International Conference on Machine Learning*, pages 9835–9847, 2021.

D. A. Spielman and S.-H. Teng. Nearly-linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving Linear Systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, volume 4, pages 81–90, 2004.

D. A. Spielman and S.-H. Teng. Nearly Linear Time Algorithms for Preconditioning and Solving Symmetric, Diagonally Dominant Linear Systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014.

P. M. Vaidya. Speeding-up Linear Programming using Fast Matrix Multiplication. In *30th annual symposium on foundations of computer science*, pages 332–337. IEEE Computer Society, 1989.

J. van den Brand, B. Peng, Z. Song, and O. Weinstein. Training (Overparametrized) Neural Networks in Near-linear Time. In *12th Innovations in Theoretical Computer Science Conference*, 2021.

K. Vu, P.-L. Poirion, and L. Liberti. Random Projections for Linear Programming. *Mathematics of Operations Research*, 43(4):1051–1071, 2018.

J. N. Weinstein, E. A. Collisson, G. B. Mills, K. R. M. Shaw, B. A. Ozenberger, K. Ellrott, I. Shmulevich, C. Sander, J. M. Stuart, et al. The Cancer Genome Atlas Pan-Cancer Analysis Project. *Nature Genetics*, 45(10):1113–1120, 2013.

T. Weng, H. Zhang, H. Chen, Z. Song, C. Hsieh, L. Daniel, D. S. Boning, and I. S. Dhillon. Towards Fast Computation of Certified Robustness for ReLU Networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 5273–5282, 2018.

E. Wong and J. Z. Kolter. Provable Defenses Against Adversarial Examples via the Convex Outer Adversarial Polytope. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 5283–5292, 2018.

D. P. Woodruff. Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2), 2014.

S. J. Wright. *Primal-dual Interior-point Methods*, volume 54. Society for Industrial and Applied Mathematics, 1997.

J. Yang and Y. Zhang. Alternating Direction Algorithms for ℓ_1-problems in Compressive Sensing. *SIAM Journal on Scientific Computing*, 33(1):250–278, 2011.

J. Yang, Y.-L. Chow, C. Ré, and M. W. Mahoney. Weighted SGD for ℓ_p regression with randomized preconditioning. *The Journal of Machine Learning Research*, 18(1):7811–7853, 2017.

H. Ye, Y. Li, C. Chen, and Z. Zhang. Fast Fisher Discriminant Analysis with Randomized Algorithms. *Pattern Recognition*, 72:82–92, 2017.

M. Yuan. High Dimensional Inverse Covariance Matrix Estimation via Linear Programming. *Journal of Machine Learning Research*, 11(Aug):2261–2286, 2010.

Y. Zhang. On the Convergence of a Class of Infeasible Interior-point Methods for the Horizontal Linear Complementarity Problem. *SIAM Journal on Optimization*, 4(1):208–227, 1994.

J. Zhu, S. Rosset, R. Tibshirani, and T. J. Hastie. 1-norm Support Vector Machines. In *Advances in Neural Information Processing Systems*, pages 49–56, 2004.