Matching the Universal Barrier Without Paying the Costs:
Solving Linear Programs with $\tilde{O}(\sqrt{\text{rank}})$ Linear System Solves

Yin Tat Lee
MIT
yintat@mit.edu

Aaron Sidford
MIT
sidford@mit.edu

Abstract

In this paper we present a new algorithm for solving linear programs that requires only $\tilde{O}(\sqrt{\text{rank}(A)L})$ iterations where $A$ is the constraint matrix of a linear program with $m$ constraints and $n$ variables and $L$ is the bit complexity of a linear program. Each iteration of our method consists of solving $\tilde{O}(1)$ linear systems and additional nearly linear time computation.

Our method improves upon the previous best iteration bound by factor of $\Omega((m/\text{rank}(A))^{1/4})$ for methods with polynomial time computable iterations and by $\Omega((m/\text{rank}(A))^{1/2})$ for methods which solve at most $\tilde{O}(1)$ linear systems in each iteration. Our method is parallelizable and amenable to linear algebraic techniques for accelerating the linear system solver. As such, up to polylogarithmic factors we either match or improve upon the best previous running times for solving linear programs in both depth and work for different ratios of $m$ and $\text{rank}(A)$.

Moreover, our method matches up to polylogarithmic factors a theoretical limit established by Nesterov and Nemirovski in 1994 regarding the use of a “universal barrier” for interior point methods, thereby resolving a long-standing open question regarding the running time of polynomial time interior point methods for linear programming.

1 Introduction

Given a matrix, $A \in \mathbb{R}^{m \times n}$, and vectors, $\vec{b} \in \mathbb{R}^m$ and $\vec{c} \in \mathbb{R}^n$, solving the linear program

$$\min_{\vec{x} \in \mathbb{R}^n : A\vec{x} \geq \vec{b}} \quad \vec{c}^T\vec{x}$$ (1.1)

is a core algorithmic task for both the theory and practice of computer science.

Since Karmarkar’s breakthrough result in 1984, proving that interior point methods can solve linear programs in polynomial time for a relatively small polynomial, interior point methods have been an incredibly active area of research with over 1200 papers written just as of 1994 [26]. Currently, the fastest asymptotic running times for solving (1.1) in many regimes are interior point methods. State of the art interior point methods for solving (1.1) require either $O(\sqrt{mL})$ iterations of solving linear systems [28] or $O((m/\text{rank}(A))^{1/4}L)$ iterations of a more complicated but still

---

[1] This expression is the dual of a linear program written in standard form. It is well known that all linear programs can be written as (1.1). Note that this notation of $m$ and $n$ differs from that in many papers. Here $m$ denotes the number of constraints and $n$ denotes the number of variables. This is the main reason we state our results by $\sqrt{\text{rank}(A)}$ instead of $\sqrt{n}$ to avoid this confusing situation.
polynomial time operation \[34, 36, 38, 1] .

However, in a breakthrough result of Nesterov and Nemirovski in 1994, they showed that there exists a universal barrier function that if computable would allow (1.1) to be solved in \(O(\sqrt{\text{rank}(A)} L)\) iterations \[25\]. Unfortunately, this barrier is more difficult to compute than the solutions to (1.1) and despite this existential result, the \(O((m\sqrt{\text{rank}(A)})^{1/4} L)\) iteration bound for polynomial time linear programming methods has not been improved in over 20 years.

In this paper we present a new interior point method that solves general linear programs in \(\tilde{O}(\sqrt{\text{rank}(A)} L)\) iterations thereby matching the theoretical limit proved by Nesterov and Nemirovski up to polylogarithmic factors.\(^3\) Furthermore, we show how to achieve this convergence rate while only solving \(O(1)\) linear systems and performing additional \(\tilde{O}(\text{nnz}(A))\) work in each iteration.\(^4\) Furthermore, our algorithm is parallelizable and we achieve the first \(\tilde{O}(\sqrt{\text{rank}(A)} L)\) depth polynomial work method for solving linear programs.

Using the regression algorithm in \[21\], our linear programming algorithm has a running time of \(\tilde{O}(\text{nnz}(A) + (\text{rank}(A))^\omega)\sqrt{\text{rank}(A)} L\) where \(\omega\) is the matrix multiplication constant.\(^5\) This is the first polynomial time algorithm for linear programming to achieve a nearly linear dependence on \(\text{nnz}(A)\) for fixed \(n\). Furthermore, we show how to use acceleration techniques as in \[35\] to decrease the amortized per-iteration costs of solving the requisite linear system and thereby achieve a linear programming algorithm with running time \(\tilde{O}(n^{5/2 - 3(\omega - 2)} m^{3(\omega - 2)} L)\) that is faster than the previous fastest running time of \(O(m^{1.5} n L)\) whenever \(m = \tilde{\Omega}(n)\). This is the first provable improvement on both running time and the number of iterations for general interior point methods in over 20 years.

We achieve our results through an extension of standard path following techniques for linear programming \[28, 6\] that we call the weighted central path. We provide a general analysis of properties of the weighted central path, discuss tools for manipulating points along the path, and ultimately produce an efficiently computable path that converges in \(\tilde{O}(\sqrt{\text{rank}(A)} L)\) steps. We hope that these results may be of independent interest and serve as tools for further improving the running time of interior point methods in general. While the analysis in this paper is quite technical, our linear programming method is straightforward and we hope that these techniques may prove useful in practice.

1.1 Previous Work

Linear programming is an extremely well studied problem with a long history. There are numerous algorithmic frameworks for solving linear programming problems, such as simplex methods \[4\], ellipsoid methods \[10\], and interior point methods \[7\]. Each method has a rich history and an impressive body of work analyzing the practical and theoretical guarantees of the methods. We couldn’t possibly cover the long line of beautiful work on this important problem in full, and we make no attempt. Instead, here we present the major improvements on the number of iterations required to solve (1.1) and discuss the asymptotic running times of these methods. For a more

\(^2\)Here and in the rest of the paper \(L\) denotes the standard “bit complexity” of the linear program. The parameter \(L\) is at most the number of bits needed to represent (1.1). For integral \(A, b, c\) the quantity \(L\) is often defined to be the potentially smaller quantity \(L = \log(1 + d_{\text{max}}) + \log(1 + \max\{\|\vec{c}\|_\infty, \|\vec{b}\|_\infty\})\) where \(d_{\text{max}}\) is the largest absolute value of the determinant of a square sub-matrix of \(A\) \[7\].

\(^3\)Here and in the remainder of the paper we use \(\tilde{O}(\cdot)\) to hide \(\text{polylog}(n, m)\) factors.

\(^4\)We assume that \(A\) has no rows or columns that are all zero as these can be remedied by trivially removing constraints or variables respectively or immediately solving the linear program. Therefore \(\text{nnz}(A) \geq \min\{m, n\}\).

\(^5\)Currently \(\omega < 2.3729\) by \[39\].
In 1984 Karmarkar [7] provided the first proof of an interior point method running in polynomial time. This method required $O(mL)$ iterations where the running time of each iteration was dominated by the time needed to solve a linear system of the form $(A^TDA)\bar{x} = \bar{y}$ for some positive diagonal matrix $D \in \mathbb{R}^{m \times m}$ and some $\bar{y} \in \mathbb{R}^n$. Using low rank matrix updates and preconditioning Karmarkar achieved a running time of $O(m^{3.5}L)$ for solving (1.1) inspiring a long line of research into interior point methods.\(^6\)

Karmarkar’s result sparked interest in a particular type of interior point methods, known as path following methods. These methods solve (1.1) by minimizing a penalized objective function $f_t(\bar{x}),$

$$\min_{\bar{x} \in \mathbb{R}^n} f_t(\bar{x}) \text{ where } f_t(\bar{x}) \overset{\text{def}}{=} t \cdot \bar{c}^T \bar{x} + \phi(\bar{x})$$

where $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$ is a barrier function such that $\phi(\bar{x}) \rightarrow \infty$ as $\bar{x}$ tends to boundary of the polytope and $t$ is a parameter. Usually, the standard log barrier $\phi(\bar{x}) \overset{\text{def}}{=} -\sum_{i \in [m]} \log([A\bar{x} - \bar{b}]_i)$ is used. Path following methods first approximately minimize $f_t$ for small $t$, then use this minimizer as an initial point to minimize $f_{(1+c)t}$ for some constant $c$, and then repeat until the minimizer is close to the optimal solution of (1.1).

Using this approach Renegar provided the first polynomial time interior point method which solves in $O(\sqrt{m}L)$ iterations [28]. As with Karmarkar’s result the running time of each iteration of this method was dominated by the time needed to solve a linear system of the form $(A^TDA)\bar{x} = \bar{y}$. Using a combination of techniques involving low rank updates, preconditioning and fast matrix multiplication, the amortized complexity of each iteration was improved [33, 6, 26]. The current fastest running time achieved by such techniques is $O(m^{1.5}nL)$ [35].

In a seminal work of Nesterov and Nemirovski [26], they showed that path-following methods can in principle be applied to any solve convex optimization problem by using a suitable barrier function. Using this technique they showed how various problems such as semi-definite programming, finding extremal ellipsoids, and more can all be solved in polynomial time via path following. In this general setting, the number of iterations required depends on the square root of a quantity associated with the barrier called its, self-concordance. They showed that for any convex set in $\mathbb{R}^n$, there exists a barrier function, called the universal barrier function, with self-concordance $O(n)$. Therefore, in theory any convex optimization problem with $n$ variables can be solved in $O(\sqrt{n}L)$ iterations. However, this result is generally considered to be only of theoretical interest as the universal barrier function is defined as the volume of certain polytopes, a problem which in full generally is NP-hard and its derivatives can only approximated in time $O(n^c)$ for some large constant $c$ [17].

Providing a barrier that supports a fast convergence rate and is easy to be approximately minimized is important theoretical question with numerous implications. Renegar’s path-following method effectively reduces solving a linear program to solving $O(\sqrt{m}L)$ linear systems. Exploiting the structure of these systems yields the fastest known algorithms for combinatorial problems such as minimum cost flow [3] and multicommodity flow [35]. Given recent breakthroughs in solving two broad class of linear systems, symmetric diagonally dominant linear systems [30, 13, 9, 15] and overdetermined system of linear equations [2, 21, 16], there are reasons to believe that improving

\(^6\)Here and in the remainder of the paper when we provide asymptotic running times for linear programming algorithms, for simplicity we hide additional dependencies on $L$ that may arise from the need to carry out arithmetic operations to precision $L$. 

3
the convergence rate of barrier methods while maintaining easy to compute iterations could have far reaching applications.\footnote{Indeed, in follow-up work [14] which is currently in submission to the 55th Annual Symposium on Foundations of Computer Science (FOCS 2014) we show that using some ideas in this paper one can solve maximum flow in $\tilde{O}(m^{3/2})$, providing the first general improvement in the running time since 1998 in [5].}

In 1989, Vaidya [38] made an important breakthrough in this direction. He proposed two barrier functions related to the volume of certain ellipsoids which were shown to yield $O((m \text{ rank}(A))^{1/4} L)$ and $O(\text{rank}(A)L)$ iteration linear programming algorithms [36, 38, 34]. Unfortunately each iteration of these methods required explicit computation of the projection matrix $D^{1/2} A (A^T D A)^{-1} A^T D^{1/2}$ for a positive diagonal matrix $D \in \mathbb{R}^{m \times m}$. This was slightly improved by Anstreicher [1] who showed it sufficed to compute the diagonal of this projection matrix. Unfortunately both these methods do not yield faster running times than [35] unless $m \gg n$ and neither are immediately amenable to take full advantage of fast linear system solvers.

| Year | Author            | Iteration number | Nature of iterations       |
|------|-------------------|------------------|---------------------------|
| 1984 | Karmarkar [7]     | $O(mL)$          | Linear system solve       |
| 1986 | Renegar [28]      | $O(\sqrt{mL})$  | Linear system solve       |
| 1989 | Vaidya [37]       | $O(\text{rank}(A)L)$ | Expensive linear algebra |
| 1989 | Vaidya [37]       | $O((m \text{ rank}(A))^{1/4} L)$ | Expensive linear algebra |
| 1994 | Nesterov and Nemirovskii [26] | $O(\sqrt{\text{rank}(A)L})$ | Volume computation       |
| 2013 | This paper        | $O(\sqrt{\text{rank}(A)L})$ | $O(1)$ Linear system solves |

These results seem to suggest that you can solve linear programs closer to the $\tilde{O}(\sqrt{\text{rank}(A)L})$ bound achieved by the universal barrier only if you pay more in each iteration. In this paper we show that this is not the case. Up to polylogarithmic factors we achieve the convergence rate of the universal barrier function while only having iterations of cost comparable to that of Karmarkar’s and Renegar’s algorithms.

1.2 Our Approach

In this paper our central goal is to produce an algorithm to solve (1.1)\footnote{We work with the dual of the standard form of a linear program, i.e. (1.1) rather than standard form, because it was not clear for us how to deal with the primal form directly. The same issue happens in all interior point methods mentioned above with the number of iterations better than $\sqrt{m} L$, hence they all rely on the dual form. In a follow-up paper [14], we show that it is possible to design interior point method by using a non $\ell_2$ type analysis.} in $\tilde{O}(\sqrt{\text{rank}(A)L})$ iterations where each iteration solves $\tilde{O}(1)$ linear systems of the form $(A^T D A) \vec{x} = \vec{y}$. To achieve our goal ideally we would produce a barrier function $\phi$ such that standard path following yields a $\tilde{O}(\sqrt{\text{rank}(A)L})$ iteration algorithm with low iterations costs. Unfortunately, we are unaware of a barrier function that both yields a fast convergence rate and has a gradient that can be computed with high accuracy using $\tilde{O}(1)$ linear system solves. Instead, we consider manipulating a barrier that we can easily compute the gradient of, the standard logarithmic barrier, $\phi(\vec{x}) = -\sum_{i \in [m]} \log |A \vec{x} - \vec{b}|_i$.

Note that the behavior of the logarithmic barrier is highly dependent on the representation of (1.1). Just duplicating a constraint, i.e. a row of $A$ and the corresponding entry in $\vec{b}$, corresponds to doubling the contribution of some log barrier term $-\log |A \vec{x} - \vec{b}|_i$ to $\phi$. It is not hard to see that repeating a constraint many times can actually slow down the convergence of standard path following...
methods. In other words, there is no intrinsic reason to weight all the $-\log|A\bar{x} - \bar{b}|_i$, the same and the running time of path following methods do depend on the weighting of the $-\log|A\bar{x} - \bar{b}|_i$.

This motivates us to consider adding weights to the log barrier that we change during the course of the algorithm. Ultimately we show that by carefully manipulating these weights we can achieve a convergence rate that depends on the dimension of the polytope, rank$(A)$, rather than the number of constrains $m$. In Section 4, we study this weighted log barrier function given by

$$\phi(\bar{x}) = -\sum_{i \in [m]} g_i ((A\bar{x} - \bar{b})_i) \cdot \log(|A\bar{x} - \bar{b}|_i).$$

where $\bar{g} : \mathbb{R}_+^m \rightarrow \mathbb{R}_+^m$ is a weight function of the current point and we investigate what properties of $\bar{g}(\bar{x})$ yield a faster convergence rate.

To illustrate the properties of the weighted logarithmic barrier, suppose for simplicity that we normalize $A$ and $\bar{b}$ so that $A\bar{x} - \bar{b} = \mathbf{1}$ and let $\bar{g} \overset{\text{def}}{=} \bar{g}(\mathbf{1})$. Under these assumptions, the rate of convergence of path following depends on $\|\bar{g}\|_1$ and

$$\max_{i \in [m]} \mathbf{1}^T A (A^T \text{diag}(\bar{g}) A)^{-1} A^T \mathbf{1}.$$  (1.2)

To improve the convergence rate we would like to keep both these quantities small. For a general matrix $A$, the quantity (1.2) is related to the leverage scores of the rows of $A$, a commonly used measure for the importance of rows in a linear system [19].

For illustration purposes, if we assume that $A$ is the incidence matrix of a certain graph and put a resistor of resistance 1/g on the edge $i$. Then, $\mathbf{1}^T A (A^T \text{diag}(\bar{g}) A)^{-1} A^T \mathbf{1}_i$ is the effective resistance of the edge $i$ [29]. Hence, we wish to to find $g$ to minimize the maximum effective resistance of the graph while keeping $\|\bar{g}\|_1$ small. Thus, the best $\bar{g}$ just makes all effective resistances the same.  

This electric network inverse problem is well studied [31] and motivates us to considering the following weight function

$$\bar{g}(\mathbf{s}) \overset{\text{def}}{=} \arg\max_{\mathbf{w} \in \mathbb{R}^m} -\mathbf{1}^T \mathbf{w} + \frac{1}{\alpha} \log \det (A^T S^{-1} W^\alpha S^{-1} A) + \beta \sum_{i \in [m]} \log w_i.$$  (1.3)

for carefully chosen constants $\alpha, \beta$ where $S \overset{\text{def}}{=} \text{diag}(\mathbf{s}(\bar{x}))$ and $W = \text{diag}(\bar{w})$. The optimality conditions of this optimization problem imply that the effective resistances are small, the total weight is small, no weight is too small, and every term in the logarithmic barrier is sufficiently penalized. This barrier is related to the volumetric barrier function used by Vaidya [37] and can be viewed as searching for the best function in a family of volumetric barrier function. This formulation with some careful analysis can be made to yield an $\tilde{O}(\sqrt{n}L)$ iteration path-following algorithm by solving the following minimax problem

$$\min_{\bar{x} \in \mathbb{R}^n} \max_{\mathbf{w} \in \mathbb{R}^m} \mathbf{t}^T \bar{x} - \mathbf{1}^T \mathbf{w} + \frac{1}{\alpha} \log \det (A^T S^{-1} W^\alpha S^{-1} A) + \beta \sum_{i \in [m]} \log w_i.$$  (1.4)

where $\mathbf{s}(\bar{x}) \overset{\text{def}}{=} A\bar{x} - \bar{b}$, $S \overset{\text{def}}{=} \text{diag}(\mathbf{s}(\bar{x}))$ and $W = \text{diag}(\bar{w})$.

Unfortunately, computing the derivative of the minimax formula still requires computing the diagonal of the projection matrix as in Vaidya and Anstreicher’s work [35, 1] and is therefore too

---

9Weights with exactly this property may not exist and we handle this in Section 6.
inefficient for our purposes. In Section 5 we show how to compute \( \vec{w} \) approximately up to certain multiplicative coordinate-wise error using dimension reduction techniques. However, this error is still too much for path following to handle the directly as multiplicatively changing weights can hurt our measures of centrality too much.

Therefore, rather than using the weighted log barrier

\[
\phi(\vec{x}) = -\sum_{i \in [m]} g_i(\vec{x}) \log(s(\vec{x})_i)
\]

where the weights \( \vec{g}(\vec{x}) \) depends on the \( \vec{x} \) directly, we maintain separate weights \( \vec{w} \) and current point \( \vec{x} \) and use the barrier

\[
\phi(\vec{x}, \vec{w}) = -\sum_{i \in [m]} w_i \log(s(\vec{x})_i).
\]

We then maintain two invariants, (1) \( \vec{x} \) is centered, i.e. \( \vec{x} \) close to the minimum point of \( t \cdot \vec{c}^T \vec{x} + \phi(\vec{x}, \vec{w}) \) and (2) \( \vec{w} \) close to \( \vec{g}(\vec{x}) \) multiplicatively.

We separate the problem of maintaining these invariants into two steps. First, we design a step for changing \( \vec{x} \) and \( \vec{w} \) simultaneously that improves centrality without moving \( \vec{w} \) too far away from \( \vec{g}(\vec{x}) \). We do this by decomposing a standard Newton step into a change in \( \vec{x} \) and a change in \( \vec{w} \) with a ratio chosen using properties of the particular weight function. Second, we show that given a multiplicative approximation to \( \vec{g}(\vec{x}) \) and bounds for how much \( \vec{g}(\vec{x}) \) may have changed, we can maintain the invariant that \( \vec{g}(\vec{x}) \) is close to \( \vec{w} \) multiplicatively without moving \( \vec{w} \) too much. We formulate this as a general two player game and prove that there is an efficient strategy to maintain our desired invariants. Combining these and standard techniques in path-following methods, we obtain an \( \tilde{O}(\sqrt{\text{rank}(A)L}) \) iterations path-following algorithm where each iterations consists of \( \tilde{O}(1) \) linear system solves.

We remark that a key component of our result is a better understanding of the effects of weighting the logarithmic barrier and note that recently Madry [18] has shown another way of using weighted barrier functions to achieve a \( \tilde{O}(m^{10/7}) \) time path-following method for the maximum flow problem on unweighted graphs. We hope this provides further evidence of the utility of the weighted central path discussed in later sections.

1.3 Geometric Interpretation of the Barrier

While to the best of our knowledge the specific weighted barrier, (1.3), presented in the previous section is new, the minimax problem, (1.4), that this weight function induces is closely related to fundamental problems in convex geometry. In particular, if we set \( \alpha = 1, t = 0, \) and consider the limit as \( \beta \to 0 \) in (1.4) then we obtain the following minimax problem

\[
\min_{\vec{x} \in \mathbb{R}^n} \max_{\vec{w} \geq 0} -\vec{x}^T \vec{w} + \log \det (\vec{A}^T \vec{S}^{-1} \vec{W} \vec{S}^{-1} \vec{A}) .
\]  

(1.5)

The maximization problem inside (1.5) is often referred to as \( D \)-optimal design and is directly related to computing the John Ellipsoid of the polytope \( \{ \vec{y} \in \mathbb{R}^n : [A (\vec{y} - \vec{x})]_i \leq s(\vec{x})_i \} \) [11]. Consequently, (1.5) is directly computing the John Ellipsoid of the polytope \( \{ \vec{x} \in \mathbb{R}^n : A \vec{x} \geq \vec{b} \} \) and hence, one can view our linear programming algorithm as using approximate John Ellipsoids to improve the convergence rate of interior point methods.
Our algorithm is not the first instance of using John Ellipsoids in convex optimization or linear programming. In a seminal work of Tarasov, Khachiyan and Erlikh in 1988 [32], they showed that a general convex problem can be solved in $O(n)$ steps of computing John Ellipsoid and querying a separating hyperplane hyperplane oracle. Furthermore, in 2008 Nesterov [23] also demonstrated how to use a John ellipsoid to compute approximation solutions for certain classes of linear programs in $O(\sqrt{n}/\epsilon)$ iteration and $\tilde{O}(n^2m + n^{1.5}m/\epsilon)$ time.

From this geometric perspective, there are two major contributions of this paper. First we show that the logarithmic volume of an approximate John Ellipsoid is an almost optimal barrier function for linear programming and second that computing an approximate John Ellipsoids can be streamlined such that the cost of these operations is comparable to computing the standard logarithmic barrier function. Beyond obtaining a faster linear program solver, this implies that we obtain the fastest algorithm for computing approximate John Ellipsoids in certain regimes. We hope to further elaborate on this connection and show more applications in a followup paper.

1.4 Overview

The rest of the paper is structured as follows: in Section 2 we provide details regarding the mathematical notation we use throughout the paper, in Section 3 we provide some preliminary information on linear programming and interior point methods, in Section 4 we formally introduce the weighted path and analyze this path assuming access to weight function, in Section 5 we consider the weighted path where the weight function is computed approximately, in Section 6 we present our weight function, and in Section 7 we put everything together to present a $\tilde{O}(\sqrt{n}/\epsilon)$ iteration algorithm for linear programming where in each iteration we solve $\tilde{O}(1)$ linear systems and show how to accelerated this algorithm. Finally, in the Appendix we provide some additional mathematical tools we use throughout the paper. Note that throughout this paper we make no attempt to reduce polylogarithmic factors in our running time.

2 Notation

Here we introduce various notation that we will use throughout the paper. This section should be used primarily for reference as we reintroduce notation as needed later in the paper.

Variables: We use the vector symbol, e.g. $\vec{x}$, to denote a vector and we omit the symbol when we denote the vectors entries, e.g. $\vec{x} = (x_1, x_2, \ldots)$. We use bold, e.g. $\mathbf{A}$, to denote a matrix. For integers $z \in \mathbb{Z}$ we use $[z] \subseteq \mathbb{Z}$ to denote the integers 1 through $z$. We let $\vec{1}_i$ denote the vector that has value 1 in coordinate $i$ and is 0 elsewhere.

Vector Operations: We frequently apply scalar operations to vectors with the interpretation that these operations should be applied coordinate wise. For example, for vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$ we let $\vec{x}/\vec{y} \in \mathbb{R}^n$ with $[\vec{x}/\vec{y}]_i \triangleq (x_i/y_i)$ and $\log(\vec{x}) \in \mathbb{R}^n$ with $[\log(\vec{x})]_i = \log(x_i)$ for all $i \in [n]$.

Matrix Operations: We call a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ positive semidefinite (PSD) if $\vec{x}^T \mathbf{A} \vec{x} \geq 0$ for all $\vec{x} \in \mathbb{R}^n$ and we call $\mathbf{A}$ positive definite (PD) if $\vec{x}^T \mathbf{A} \vec{x} > 0$ for all $\vec{x} \in \mathbb{R}^n$. For a positive definite matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ we denote let $\| \cdot \|_\mathbf{A} : \mathbb{R}^n \to \mathbb{R}$ denote the norm such that for all $\vec{x} \in \mathbb{R}^n$ we have $\| \vec{x}\|_\mathbf{A} \triangleq \sqrt{\vec{x}^T \mathbf{A} \vec{x}}$. For symmetric matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ we write $\mathbf{A} \preceq \mathbf{B}$ to indicate that $\mathbf{B} - \mathbf{A}$ is PSD (i.e. $\vec{x}^T \mathbf{A} \vec{x} \leq \vec{x}^T \mathbf{B} \vec{x}$ for all $\vec{x} \in \mathbb{R}^n$) and we write $\mathbf{A} \prec \mathbf{B}$ to indicate that $\mathbf{B} - \mathbf{A}$ is PD (i.e. $\vec{x}^T \mathbf{A} \vec{x} < \vec{x}^T \mathbf{B} \vec{x}$ for all $\vec{x} \in \mathbb{R}^n$). We define $>$ and $\succeq$ analogously. For
A, B ∈ ℝⁿˣᵐ, we let A ⊙ B denote the Schur product, i.e. [A ⊙ B]_{ij} \overset{\text{def}}{=} A_{ij} ⋅ B_{ij} for all i ∈ [n] and j ∈ [m], and we let A(2) \overset{\text{def}}{=} A ⊙ A. We use mzn(A) to denote the number of nonzero entries in A.

Diagonal Matrices: For A ∈ ℝⁿˣⁿ we let diag(A) ∈ ℝⁿ denote the vector such that diag(A)_i = A_{ii} for all i ∈ [n]. For \( \bar{x} \in ℝ^n \) we let diag(\( \bar{x} \)) ∈ ℝⁿˣⁿ be the diagonal matrix such that diag(diag(\( \bar{x} \))) = \( \bar{x} \). For A ∈ ℝⁿˣⁿ we let diag(A) be the diagonal matrix such that diag(diag(A)) = diag(A). For a vector \( \bar{x} \in ℝ^n \) when the meaning is clear from context we use X ∈ ℝⁿˣⁿ to denote X \overset{\text{def}}{=} diag(\( \bar{x} \)).

Multiplicative Approximations: Frequently in this paper we need to convey that two vectors \( \bar{x} \) and \( \bar{y} \) are close multiplicatively. We often write \( \|X^{-1}(\bar{y} − \bar{x})\|_\infty \leq \epsilon \) to convey the equivalent facts that \( y_i ∈ [(1 − \epsilon)x_i, (1 + \epsilon)x_i] \) for all i or \( (1 − \epsilon)X \preceq Y \preceq (1 + \epsilon)X \). At times we find it more convenient to write \( \|\log \bar{x} − \log \bar{y}\| \leq \epsilon \) which is approximately equivalent for small \( \epsilon \). In Lemma 33, we bound the quality of this approximation.

Matrices: We use ℝₘ₊₀ to denote the vectors in ℝᵐ where each coordinate is positive and for a matrix A ∈ ℝᵐˣⁿ and vector \( \bar{x} \in ℝⁿ \) we define the following matrices and vectors

- Projection matrix \( P_{A}(\bar{x}) ∈ ℝᵐˣᵐ : P_{A}(\bar{x}) \overset{\text{def}}{=} X^{1/2}A(A^TXA)^{-1}A^TX^{1/2} \).
- Leverage scores \( σ_{A}(\bar{x}) ∈ ℝⁿ : σ_{A}(\bar{x}) \overset{\text{def}}{=} \text{diag}(P_{A}(\bar{x})) \).
- Leverage matrix \( Σ_{A}(\bar{x}) ∈ ℝᵐˣᵐ : Σ_{A}(\bar{x}) \overset{\text{def}}{=} \text{diag}(P_{A}(\bar{x})) \).
- Projection Laplacian \( Λ_{A}(\bar{x}) ∈ ℝᵐˣᵐ : Λ_{A}(\bar{x}) \overset{\text{def}}{=} Σ_{A}(\bar{x}) − P_{A}(\bar{x})^{(2)} \).

The definitions of projection matrix and leverage scores are standard when the rows of A are reweighed by the values in vector \( \bar{x} \).

Convex Sets: We call a set \( U ⊆ ℝ^k \) convex if for all \( \bar{x}, \bar{y} ∈ ℝ^k \) and all \( t ∈ [0, 1] \) it holds that \( t \cdot \bar{x} + (1 − t) \cdot \bar{y} ∈ U \). We call U symmetric if \( \bar{x} ∈ ℝ^k \Leftrightarrow −\bar{x} ∈ ℝ^k \). For any \( α > 0 \) and convex set \( U ⊆ ℝ^k \) we let \( αU \overset{\text{def}}{=} \{ \bar{x} ∈ ℝ^k | α^{-1} \bar{x} ∈ U \} \). For any \( p ∈ [1, ∞] \) and \( r ∈ ℝ_{≥0} \) we refer to the symmetric convex set \( \{ \bar{x} ∈ ℝ^k | \|\bar{x}\|_p \leq r \} \) as the \( ℓ_p \) ball of radius \( r \).

Calculus: For a function \( f : ℝ^n → ℝ \) differentiable at \( x ∈ ℝ^n \), we denote the gradient of \( f \) at \( \bar{X} \) by \( \nabla f(\bar{X}) ∈ ℝ^n \) where we have \( [\nabla f(\bar{X})]_i = \frac{∂}{∂x_i}f(\bar{X}) \) for all \( i ∈ [n] \). If \( f : ℝ^n → ℝ \) is twice differentiable at \( x ∈ ℝ^n \), we denote the Hessian of \( f \) at \( x \) by \( \nabla^2 f(\bar{X}) \) where we have \( [\nabla^2 f(\bar{X})]_{ij} = \frac{∂^2}{∂x_i∂x_j}f(\bar{X}) \) for all \( i, j ∈ [n] \). Often we will consider functions of two vectors \( g : ℝ^{n₁ × n₂} → ℝ \), and wish to compute the gradient and hessian of \( g \) restricted to one of the two vectors. For \( \bar{x} ∈ ℝ^n \) and \( \bar{y} ∈ ℝ^m \) then we let \( \nabla_{\bar{x}}g(\bar{a}, \bar{b}) ∈ ℝ^{n₁} \) denote the gradient of \( g \) for fixed \( \bar{y} \) at point \( \{\bar{a}, \bar{b}\} ∈ ℝ^{n₁ × n₂} \). We define \( \nabla_{\bar{y}}g, \nabla^2_{\bar{x}}g \) and \( \nabla^2_{\bar{y}}g \) similarly. Furthermore for \( h : ℝ^n → ℝ^m \) differentiable at \( x ∈ ℝ^n \) we let \( J(h(\bar{x})) ∈ ℝ^{mₓ^n} \) denote the Jacobian of \( h \) at \( \bar{x} \) where for all \( i ∈ [m] \) and \( j ∈ [n] \) we let \( [J(h(\bar{x})))]_{ij} \overset{\text{def}}{=} \frac{∂}{∂x_j}h(\bar{x})_i \). For functions of multiple vectors we use subscripts, e.g. \( J_{\bar{x}} \), to denote the Jacobian of the function restricted to the \( \bar{x} \) variable.

Problem Specific Notation: Below is a list of a variety of linear programming specific notation that we will introduce formally later. For many of the quantities we included the typical order of magnitude as they appear during our algorithms.

- Slacks: \( \bar{s}(\bar{x}) = A\bar{x} − \bar{b} \).
• Penalized objective function (4.1): \( f_t(\vec{x}, \vec{w}) = t \cdot \vec{c}^T \vec{x} - \sum_{i \in [m]} w_i \log s(\vec{x})_i. \)

• Newton step (4.2): \( \vec{H}_t(\vec{x}, \vec{w}) = (\nabla^2_{\vec{x}, \vec{x}} f_t(\vec{x}, \vec{w}))^{-1} \nabla_{\vec{x}} f_t(\vec{x}, \vec{w}). \)

• Centrality (4.3): \( \delta_t(\vec{x}, \vec{w}) = \|\vec{H}_t(\vec{x}, \vec{w})\|_{\nabla^2_{\vec{x}, \vec{x}} f_t(\vec{x}, \vec{w})} \approx \frac{1}{\text{polylog}(m)}. \)

• Weight condition number (4.3): \( \gamma(\vec{s}, \vec{w}) = \max_{i \in [m]} \|W_i^{-1/2}I_i\|_{\text{PS}^{-1}\vec{A}(\vec{w})} \approx 1. \)

• Properties of weight function (Def 7): size \( c_1(\vec{g}) = \|\vec{g}(\vec{s})\|_1 \approx \sqrt{\text{rank}(\vec{A})}, \) condition number \( c_\gamma(\vec{g}) = \sup_{\vec{s}} \gamma(\vec{s}, \vec{g}(\vec{s})) \approx 1, \) consistency \( c_r(\vec{g}) \approx \log \left( \frac{m}{\text{rank}(\vec{A})} \right). \)

• Difference between \( \vec{g} \) and \( \vec{w} \) (4.16): \( \Psi(\vec{s}, \vec{w}) = \log(\vec{g}(\vec{s})) - \log(\vec{w}). \)

• Potential function for tracing 0 (Def 12): \( \Phi_{\mu}(\vec{x}) = e^{\mu x} + e^{-\mu x} \approx \text{poly}(m). \)

The weight function proposed (6.1):

\[
\vec{g}(\vec{s}) = \arg\min_{\vec{w} \in \mathbb{R}^m_{>0}} \hat{f}(\vec{s}, \vec{w}) \quad \text{where} \quad \hat{f}(\vec{s}, \vec{w}) = \vec{1}^T \vec{w} - \frac{1}{\alpha} \log \det(\vec{A}_x^T \vec{W}^n \vec{A}_x) - \beta \sum_i \log w_i
\]

where \( \vec{A}_x = \vec{S}^{-1} \vec{A}, \alpha \approx 1 - 1/\log_2 \left( \frac{m}{\text{rank}(\vec{A})} \right), \beta \approx \text{rank}(\vec{A})/m. \)

### 3 Preliminaries

Here we provide a brief introduction to path following methods for linear programming. The purpose of this section is to formally introduce interior point terminology and methodology that we build upon to obtain \( O(\sqrt{\text{rank}(\vec{A})}L) \) iteration solver. The algorithm and the analysis discussed in this section can be viewed as a special case of the framework presented in Section 4. The reader well versed in path following methods can likely skip this section and to the more curious reader we encourage them to consider some of the many wonderful expositions on this subject [25, 40, 6] for further reading.

#### 3.1 The Setup

Given a matrix, \( \vec{A} \in \mathbb{R}^{m \times n} \), and vectors, \( \vec{b} \in \mathbb{R}^m \) and \( \vec{c} \in \mathbb{R}^n \), the central goal of this paper is to efficiently compute a solution to the following linear program

\[
\min_{\vec{x} \in \mathbb{R}^n : \vec{A}\vec{x} \geq \vec{b}} \quad \vec{c}^T \vec{x} \tag{3.1}
\]

It is well known that this is the dual of the standard form of a linear program and hence all linear programs can be expressed by (3.1). We call a vector \( \vec{x} \in \mathbb{R}^m \) feasible if \( \vec{A}\vec{x} \geq \vec{b} \), we call \( \vec{c}^T \vec{x} \) the cost of such a vector, and therefore our goal is to either compute a minimum cost feasible vector or determine that none exists.

We assume that \( \vec{A} \) is full rank, i.e. \( \text{rank}(\vec{A}) = n \), and that \( m \geq n \). Nevertheless, we still write many of our results using \( \text{rank}(\vec{A}) \) rather than \( n \) for two reasons. First, this notation makes clear that \( \text{rank}(\vec{A}) \) is referring to the smaller of the two quantities \( m \) and \( n \). Second, all our results except the Section 7.3 do apply without this assumption so long as 1) the matrix inverse is replaced with
the matrix pseudoinverse, 2) determinant is replaced with pseudo determinant, 3) use linear solver that supports pseudoinverse and take advantages of low rank matrices, 4) In the beginning of the linear program solver, check if \( \vec{c} \not\in \text{im} \ A^T \) and outputs unbounded if it happens.

### 3.2 Path Following Interior Point

Interior point methods solve (3.1) by maintaining a point \( \vec{x} \) that is in the interior of the feasible region, i.e. \( \vec{x} \in S^0 \) where

\[
S^0 \overset{\text{def}}{=} \{ \vec{x} \in \mathbb{R}^n : A \vec{x} > \vec{b} \}.
\]

These methods attempt to iteratively decrease the cost of \( \vec{x} \) while maintaining strict feasibility. This is often done by considering some measurement of the distance to feasibility such as \( s(\vec{x}) \overset{\text{def}}{=} A \vec{x} - \vec{b} \), called the slacks, and creating some penalty for these distances approaching 0. Since \( s(\vec{x}) > 0 \) if and only if \( \vec{x} \in S^0 \) by carefully balancing penalties for small \( s(\vec{x}) \) and penalties for large \( \vec{c}^T \vec{x} \) these methods eventually compute a point close enough to the optimum solution that it can be computed exactly.

Path following methods fix ratios between the the penalty for large \( \vec{c}^T \vec{x} \) and the penalty for small \( s(\vec{x}) \) and alternate between steps of optimizing with respect to this ratio and changing the ratio. These methods typically encode the penalties through a barrier function \( \phi : \mathbb{R}^m_{>0} \to \mathbb{R} \) such that \( \phi(s(\vec{x})) \to \infty \) as \( s(\vec{x}) \to 0 \) for any \( i \in [m] \) and they encode the ratio through some parameter \( t > 0 \). Formally, they attempt to solve optimization problems of the following form for increasing values of \( t \)

\[
\min_{\vec{x} \in \mathbb{R}^n} f_t(\vec{x}) \quad \text{where} \quad f_t(\vec{x}) \overset{\text{def}}{=} t \cdot \vec{c}^T \vec{x} + \phi(s(\vec{x}))
\]  

(3.2)

Since \( \phi(s(\vec{x})) \to \infty \) as \( s(\vec{x}) \to 0 \) the minimizer of \( f_t(\vec{x}) \), denoted \( \vec{x}^* (t) \), is in \( S^0 \) for all \( t \). As \( t \) increases the effect of the cost vector on \( \vec{x}^* (t) \) increases and the distance from the boundary of the feasible region as measured by \( s(\vec{x}) \) decreases. One can think of the points \( \{ \vec{x}^* (t) \mid t > 0 \} \) as a path in \( \mathbb{R}^n \), called the central path, where \( \vec{x}^* (t) \) approaches a solution to (3.1) as \( t \to \infty \). A standard choice of barrier is the standard log barrier, \( \phi(s(\vec{x})) \overset{\text{def}}{=} -\sum_{i} \log(s(\vec{x})_i) \) and for this choice of barrier we refer to \( \{ \vec{x}^* (t) \mid t > 0 \} \) as the standard central path.

Path following methods typically follow the following framework:

1. **[Initial point]** Compute an approximation \( \vec{x}^*(t) \) for some \( t \).

2. **[Follow the central path]** Repeatedly increase \( t \) to some \( t' \) and compute an approximation to \( \vec{x}^*(t') \).

3. **[Round to optimal solution]** Use the current approximation to \( \vec{x}^*(t) \) to compute the solution to (3.1).

Steps (1) and (3) are typically carried out by standard interior point techniques. These techniques are fairly general and covered briefly in Section 7 and Appendix D. However, the manner in which (2) is performed varies greatly from method to method. In the following subsection we provide a simple technique for performing (2) that yields reasonable running times and serves as the foundation for the algorithms considered in the remainder of the paper.
3.3 Following the Path

There are numerous approaches to follow the central path, i.e. approximately compute $\bar{x}^*(t)$ for increasing values of $t$. Even with the barrier fixed there are numerous schemes to balance maintaining a point close to a central path point, advancing to a further central path point, and performing the numerical linear algebra needed for these operations [35, 6, 20, 26].

In this section we present a simple and common method whereby we simply alternate between improving our distance to $\bar{x}^*(t)$ for some fixed $t$, and increasing $t$ by some fixed multiplicative factor. This method reduces the analysis of path following to bounding the computational complexity of centering, i.e. improve the distance to $\bar{x}^*(t)$, and bounding how much increasing $t$ hurts centrality, i.e. increases the distance to $\bar{x}^*(t)$. In the remainder of this section we show how to perform this analysis for the standard central path, $\phi(\bar{x}) \doteq -\sum_{i\in[m]} \log(s(\bar{x})))$.

Typically path following methods center, i.e. minimize $f_t(\bar{x})$, using Newton’s method or some variant thereof. While for arbitrary current point $\bar{x} \in S^0$ and $t > 0$ the function $f_t(\bar{x})$ can be ill-behaved, in a region near $\bar{x}^*(t)$ the hessian of $f_t(\bar{x})$ given by $\nabla^2 f_t(\bar{x}) = A^T S^{-2} A$ for $S \doteq \text{diag}(\bar{s}(\bar{x}))$ changes fairly slowly. More precisely, if one considers the second order approximation of $f_t(\bar{z})$ around some point $\bar{x} \in S^0$ “close enough” to $\bar{x}^*(t)$, then

$$f_t(\bar{z}) \approx f_t(\bar{x}) + (\nabla f_t(\bar{x}), \bar{z} - \bar{x}) + \frac{1}{2} (\bar{z} - \bar{x})^T (\nabla^2 f_t(\bar{x})) (\bar{z} - \bar{x})$$

and applies one step of Newton’s method, i.e. minimizes this quadratic approximation to compute

$$\bar{x}^{(new)} := \bar{x} - (\nabla^2 f_t(\bar{x}))^{-1} \nabla f_t(\bar{x}) = \bar{x} - (A^T S^{-2} A)^{-1}(t\bar{c} - A^T \bar{s})$$

for $\bar{s} \doteq \bar{s}(\bar{x})$ then this procedure rapidly converges to $\bar{x}^*(t)$.

To quantify this, we measure centrality, i.e. how close the current point $\bar{x} \in S^0$ is to $\bar{x}^*(t)$, by the size of this Newton step in the hessian induced norm. For $\bar{x} \in S^0$ and Newton step $\bar{h}_t(\bar{x}) \doteq (\nabla^2 f_t(\bar{x}))^{-1} \nabla f_t(\bar{x})$ we denote centrality by $\delta_t(\bar{x}) \doteq \| \bar{h}_t(\bar{x}) \|_{\nabla^2 f_t(\bar{x})}$. Standard analysis of Newton’s method shows that if $\delta_t(\bar{x})$ is less than some constant then for $\bar{x}^{(new)} := \bar{x} - \bar{h}(\bar{x})$ we have $\delta_t(\bar{x}^{(new)}) = O(\delta_t(\bar{x})^2)$ (See Lemma 5). Furthermore, under these conditions it is not hard to show that for $t' = t(1 + (m)^{-1/2})$ we have $\delta_t'(\bar{x}^{(new)}) \leq O(\delta_t(\bar{x}))$ (See Lemma 1).

Combining these facts yields that in $O(\sqrt{m})$ iterations we can double $t$ while maintaining a nearly centered $\bar{x}$, i.e. $\delta_t(\bar{x})$ at most a constant. With some additional work discussed briefly in Section 7 it can be shown that by maintaining a nearly centered $\bar{x}$ and doubling $t$ at most $O(L)$ times one can compute a solution to (3.1). Therefore, this method solves (3.1) in $O(\sqrt{m}L)$ iterations where the cost of each iteration is $O(\text{nnz}(A))$ plus the time need to solve a linear system in the matrix $A^T S^{-2} A$.

4 Weighted Path Following

In this section we introduce the optimization framework we use to solve the linear programs, the weighted central path. After formally defining the path (Section 4.1), we prove properties of the path (Section 4.2) and show how to center along the path (Section 4.3). We show that the performance of path following methods using a weighted central path depends crucially on how the weights are
computed and in Section 4.4 we characterize the properties we require of such a weight function in order to ensure that our weighted path following scheme converges efficiently. In Section 4.2 we analyze the convergence rate of our weighted path following scheme assuming the ability to compute these weights exactly. In the following section we then show how it suffices to compute the weights approximately (Section 5), we show how to compute these weights efficiently (Section 6), and we show how this yields an efficient linear program solver (Section 7).

4.1 The Weighted Path

Our weighted path following method is a generalization of the path following scheme presented in Section 3.2. Rather than keeping the barrier function $\phi(\bar{x}) = -\sum_{i \in [m]} \log s(\bar{x})$, fixed we allow for greater flexibility in how we penalize slack variables and adaptively modify the barrier function in order to take larger steps. In addition to maintaining a feasible point $\bar{x}$ and a path parameter $t$ we maintain a set of positive weights $\bar{w} \in \mathbb{R}_0^m$ and attempt to minimize the penalized objective function $f_t : S^0 \times \mathbb{R}_0^m \to \mathbb{R}$ given for all $\bar{x} \in S^0$ and $\bar{w} \in \mathbb{R}_0^m$ by

$$f_t(\bar{x}, \bar{w}) = t \cdot c^T \bar{x} - \sum_{i \in [m]} w_i \log s(\bar{x})_i.$$  \hspace{1cm} (4.1)

We maintain a feasible point $\{\bar{x}, \bar{w}\} \in \{S^0 \times \mathbb{R}_0^m\}$ and our goal is to compute a sequence of feasible points for increasing $t$ and changing $\bar{w}$ such that $f_t(\bar{x}, \bar{w})$ is nearly minimized with respect to $\bar{x}$.

Note that trivially any $\bar{x} \in S^0$ can be expressed as $\arg\min_{\bar{w} \in \mathbb{R}_0^m} f_t(\bar{y}, \bar{w})$ for some $\bar{w} \in \mathbb{R}_0^m$ and therefore, every $\bar{x} \in S^0$ is a “weighted central path point” for some choice of weights. However, in order to to convert a weighted central path point $\{\bar{x}, \bar{w}\} \in \{S^0 \times \mathbb{R}_0^m\}$ to a solution for (1.1) we will need to have $t$ large and $\|\bar{w}\|_1$ small which precludes this trivial choice of $t$ and $\bar{w}$.

In the remainder of the paper, we show that by careful updating $\bar{x}, \bar{w}$, and $t$ we can stay close to the weighted central path while making large increases in $t$ and maintaining $\|\bar{w}\|_1$ small. Ultimately, this will allow us to solve linear programs in $\tilde{O}(\sqrt{\text{rank}(A) L})$ iterations while only solving $\tilde{O}(1)$ linear systems in each iteration.

4.2 Properties of the Weighted Path

As in Section 3.3 for a feasible $\{\bar{x}, \bar{w}\} \in \{S^0 \times \mathbb{R}_0^m\}$ we measure the centrality of this point by the size of the newton step on $\bar{x}$ in the hessian norm, denoted by $\delta_t(\bar{x}, \bar{w})$ and we call $\{\bar{x}, \bar{w}\}$ a central path point if $\delta_t(\bar{x}, \bar{w}) = 0$. For the penalized objective function $f_t$, we see that the Newton step, $\tilde{h}_t(\bar{x}, \bar{w})$, is given by

$$\tilde{h}_t(\bar{x}, \bar{w}) = (\nabla^2_{\bar{x}\bar{x}} f_t(\bar{x}, \bar{w}))^{-1} \nabla_{\bar{x}} f_t(\bar{x}, \bar{w})$$

$$= (A^T S^{-1} W S^{-1} A)^{-1} (t\bar{c} - A^T S^{-1} \bar{w})$$  \hspace{1cm} (4.2)

and the centrality, $\delta_t(\bar{x}, \bar{w})$, is given by

$$\forall\{\bar{x}, \bar{w}\} \in \{S^0 \times \mathbb{R}_0^m\} : \delta_t(\bar{x}, \bar{w}) \overset{\text{def}}{=} \left\| \tilde{h}_t(\bar{x}, \bar{w}) \right\|_{\nabla^2_{\bar{x}\bar{x}} f_t(\bar{x}, \bar{w})}$$

$$= \left\| t\bar{c} - A^T S^{-1} \bar{w} \right\|_{(A^T S^{-1} W S^{-1} A)^{-1}}.$$  \hspace{1cm} (4.3)

Whereas in the standard central path we saw that the centrality increased at a rate of $\sqrt{m}$ as $t$ increased, here we show that in this more general case, the $m$ is replaced by the total weight $\|\bar{w}\|_1 = \sum_{i \in [m]} w_i$. 

12
Lemma 1 (Weighted Path Step). For all \( \{\vec{x}, \vec{w}\} \in \{S^0 \times \mathbb{R}^m_{\geq 0}\} \) and \( t, \alpha \geq 0 \), we have
\[
\delta_{(1+\alpha)t}(\vec{x}, \vec{w}) \leq (1 + \alpha)\delta_t(\vec{x}, \vec{w}) + \alpha \sqrt{\vec{w}_1}
\]

Proof. Let \( \vec{s}^\text{def} = \vec{s}(\vec{x}) \). By (4.3) we have
\[
\delta_{(1+\alpha)t}(\vec{x}, \vec{w}) = \| (1 + \alpha)t\vec{c} - A^TS^{-1}\vec{w} \|_{(A^T S^{-1} WS^{-1} A)^{-1}}.
\]

Now, \( \cdot \) \( (A^T S^{-1} WS^{-1} A)^{-1} \) is a norm and therefore by the triangle inequality and the definition of \( \delta_t(\vec{x}, \vec{w}) \)
\[
\delta_{(1+\alpha)t}(\vec{x}, \vec{w}) \leq (1 + \alpha)\delta_t(\vec{x}, \vec{w}) + \alpha \| A^T S^{-1}\vec{w} \|_{(A^T S^{-1} WS^{-1} A)^{-1}}. \tag{4.4}
\]

Recall that \( P_{S^{-1}A}(\vec{w}) = W^{1/2} S^{-1} A (A^T S^{-1} WS^{-1} A)^{-1} A^T S^{-1} W^{1/2} \) is a projection matrix. Consequently \( P_{S^{-1}A}(\vec{w}) \leq I \) and we have
\[
\| A^T S^{-1}\vec{w} \|_{(A^T S^{-1} WS^{-1} A)^{-1}} = \| W^{-1/2} \vec{w} \|_{P_{S^{-1}A}(\vec{w})} \leq \| W^{-1/2} \vec{w} \|_2 = \sqrt{\sum_{i \in [m]} w_i}. \tag{4.5}
\]

Combining (4.4) and (4.5) yields the result. \( \square \)

Now to see how well a Newton step on \( \vec{x} \) can center, i.e. decrease \( \delta_t(\vec{x}, \vec{w}) \), we need to bound how fast the second order approximation of \( f_t(\vec{x}, \vec{w}) \) can change, i.e. how much the Hessian, \( \nabla^2 f_t(\vec{x}, \vec{w}) \), changes as we change \( \vec{x} \). We do this by bounding how much that slacks can change as we change \( \vec{x} \). As \( \nabla^2 f_t(\vec{x}, \vec{w}) = A^T S^{-1} WS^{-1} A \) this immediately bounds how much the Hessian can change as we change \( \vec{x} \). The following lemma is motivated by similar results in [37, 1].

Lemma 2 (Relative Change of Slacks). Let \( \vec{x}^{(\text{new})} = \vec{x} + \Delta \) for some \( \vec{x} \in S^0 \) and \( \Delta \in \mathbb{R}^n \). Let \( \vec{s}^{(\text{new})} \) and \( \vec{s} \) denote the slacks associated with \( \vec{x}^{(\text{new})} \) and \( \vec{x} \) respectively. If \( \| S^{-1} A \Delta \|_\infty < 1 \) then \( \vec{x}^{(\text{new})} \in S^0 \) and
\[
\| S^{-1} A \Delta \|_\infty \leq \| \Delta \|_{A^T S^{-1} WS^{-1} A} \cdot \max_{i \in [m]} \| W^{-1/2} \vec{t}_i \|_{P_{S^{-1}A}(\vec{w})} \tag{4.6}
\]

In particular, choosing \( \Delta = -\vec{h}_t(\vec{x}, \vec{w}) \) yields
\[
\| S^{-1} A \Delta \|_\infty \leq \| \Delta \|_{A^T S^{-1} WS^{-1} A} \cdot \max_{i \in [m]} \| W^{-1/2} \vec{t}_i \|_{P_{S^{-1}A}(\vec{w})}.
\]

Proof. Clearly \( \vec{s}^{(\text{new})} = \vec{s} + A \Delta \) and therefore the multiplicative change in slacks is given by \( \| S^{-1}(\vec{s}^{(\text{new})}) - \vec{s} \|_\infty = \| S^{-1} A \Delta \|_\infty \). Consequently \( \vec{x}^{(\text{new})} \in S^0 \) if and only if \( \| S^{-1} A \Delta \|_\infty < 1 \).

To prove (4.6) we note that by definition of \( \cdot \) \( \| . \|_\infty \)
\[
\| S^{-1} A \Delta \|_\infty = \max_{i \in [m]} \left| \langle S^{-1} A \Delta, \vec{t}_i \rangle \right|.
\]

Using that \( A \) is full rank and therefore \( A^T S^{-1} W S^{-1} A > 0 \) then yields
\[
\| S^{-1} A \Delta \|_\infty = \max_{i \in [m]} \left| \langle (A^T S^{-1} W S^{-1} A)^{1/2} \Delta, (A^T S^{-1} W S^{-1} A)^{-1/2} A^T S^{-1} \vec{t}_i \rangle \right|.
\]

13
Applying Cauchy Schwarz we have
\[
\|S^{-1}A\Delta\|_{\infty} \leq \|\Delta\|_{A^TS^{-1}WS^{-1}A^T} \cdot \max_{i \in [m]} \|A^TS^{-1}x_i\|_{(A^TS^{-1}WS^{-1}A)^{-1}}.
\]
Recalling the definition \(P_{S^{-1}A}(\tilde{w}) = W^{1/2}S^{-1}A(AS^{-1}WS^{-1}A)^{-1}A^TS^{-1}W^{1/2}\) yields the result. \(\square\)

Lemma 2 implies that as \(\|W^{-1/2}x_i\|_{P_{S^{-1}A}(\tilde{w})}\) decreases, the region over which Newton steps do not change the Hessian too much increases. We call this quantity, \(\|W^{-1/2}x_i\|_{P_{S^{-1}A}(\tilde{w})}\), the weight condition number as it measures a particular operator norm of a matrix.

**Definition 3 (Weight Condition Number).** For \(\tilde{s}, \tilde{w} \in \mathbb{R}^m_0\) the weight condition number, \(\gamma(\tilde{s}, \tilde{w})\) is given by
\[
\gamma(\tilde{s}, \tilde{w}) \triangleq \max_{i \in [m]} \|W^{-1/2}x_i\|_{P_{S^{-1}A}(\tilde{w})}.
\]

From Lemmas 1 and 2 our goal in using the weighted central path is clear. We wish to keep \(\|\tilde{w}\|_1\) small so that we can make large increases to \(t\) without increasing centrality and we wish to keep \(\gamma(\tilde{s}(\tilde{x}), \tilde{w})\) small so that over a large region we can improve centrality quickly. Unfortunately, while it is not too difficult to produce weights that meet these criterion, changing the weights can also increase \(\delta_t\). Therefore, we also need to choose weights in such a way that they do not change too drastically as we take Newton steps. In the next subsection we introduce the step that we use to improve centrality and account for possible changes in the weights.

### 4.3 Centering Steps

Here we define the centering step we use to decrease \(\delta_t(\tilde{x}, \tilde{w})\). Rather than simply performing a Newton step on \(\tilde{x}\), we decompose this step into a change in \(\tilde{x}\) and a change in \(\tilde{w}\). This allows us to obtain tighter bounds on how much centrality we lose when we change weights. Our update step takes an input an additional parameter \(r\) that controls the ratio of how much we change \(\tilde{w}\) and how much we change \(\tilde{x}\). Setting \(r = 0\) corresponds to a standard Newton step where the weights are not updated.

**Definition 4 (r-step).** Given a feasible point \(x^{(old)}, \tilde{w}^{(old)} \in \{S^0 \times \mathbb{R}^m_0\}\), a path parameter \(t\), and \(0 \leq r \leq 1\) a r-step
\[
\{\tilde{x}^{(new)}, \tilde{w}^{(new)}\} = \text{step}_r(\tilde{x}^{(old)}, \tilde{w}^{(old)}, r)
\]
is defined as follows
\[
\begin{align*}
\tilde{x}^{(new)} & \triangleq \tilde{x}^{(old)} - \frac{1}{1 + r} \tilde{h}_t(\tilde{x}^{(old)}, \tilde{w}^{(old)}), \\
\tilde{w}^{(new)} & \triangleq \tilde{w}^{(old)} + \frac{r}{1 + r} W_{(old)} S_{(old)}^{-1} A \tilde{h}_t(\tilde{x}^{(old)}, \tilde{w}^{(old)})
\end{align*}
\]
where we recall that
\[
\tilde{h}_t(\tilde{x}^{(old)}, \tilde{w}^{(old)}) \triangleq (A^T S_{(old)}^{-1} W_{(old)} S_{(old)}^{-1} A)^{-1} (t\tilde{c} - A^T S_{(old)}^{-1} \tilde{w}^{(old)})
\]
and we let \(\tilde{s}^{(old)}\) and \(\tilde{s}^{(new)}\) denote the slacks with \(\tilde{x}^{(old)}\) and \(\tilde{x}^{(new)}\) respectively.
Note that for a $r$-step we have
\[
\tilde{s}^{(\text{new})} = \tilde{s}^{(\text{old})} - \frac{1}{1+r} A \tilde{h}(\tilde{x}^{(\text{old})}, \tilde{w}^{(\text{old})})
\]
and therefore
\[
W^{-1}_{(\text{old})}(\tilde{w}^{(\text{new})} - \tilde{w}^{(\text{old})}) = -r S^{-1}_{(\text{old})}(\tilde{s}^{(\text{new})} - \tilde{s}^{(\text{old})}).
\]
In other words, a $r$-step performs a multiplicative update on the weights that is exactly $r$ times larger than the update on the slacks.

Using Lemma 2 we now show that so long as $\delta_t(\tilde{x}^{(\text{old})}, \tilde{w}^{(\text{old})})$ is reasonably small with respect to the weight condition number, any $r$-step produces a feasible $\{\tilde{x}^{(\text{new})}, \tilde{w}^{(\text{new})}\}$ and does not change the Hessian too much.

**Lemma 5** (Stability of $r$-step). Let $\{\tilde{x}^{(\text{new})}, \tilde{w}^{(\text{new})}\} = \text{step}_t(\tilde{s}^{(\text{old})}, \tilde{w}^{(\text{old})}, r)$ where
\[
\gamma \overset{\text{def}}{=} \gamma(\tilde{x}^{(\text{old})}, \tilde{w}^{(\text{old})}) \quad \text{and} \quad \delta_t \overset{\text{def}}{=} \delta_t(\tilde{x}^{(\text{old})}, \tilde{w}^{(\text{old})}) \leq \frac{1}{8\gamma}.
\]
Under these conditions we have
\[
\|S_{(\text{old})}^{-1}(\tilde{s}^{(\text{new})} - \tilde{s}^{(\text{old})})\|_{W_{(\text{old})}} \leq \frac{1}{1+r} \cdot \delta_t, \quad (4.9)
\]
\[
\|S_{(\text{old})}^{-1}(\tilde{s}^{(\text{new})} - \tilde{s}^{(\text{old})})\|_{\infty} \leq \frac{1}{1+r} \cdot \delta_t \cdot \gamma, \quad (4.10)
\]
\[
\|W_{(\text{old})}^{-1}(\tilde{w}^{(\text{new})} - \tilde{w}^{(\text{old})})\|_{\infty} \leq \frac{r}{1+r} \cdot \delta_t \cdot \gamma. \quad (4.11)
\]
Consequently $\{\tilde{x}^{(\text{new})}, \tilde{w}^{(\text{new})}\}$ is feasible and
\[
(1 - 3\delta_t \gamma) \nabla_{\tilde{x}\tilde{x}}^2 f_t(\tilde{x}^{(\text{old})}, \tilde{w}^{(\text{old})}) \preceq \nabla_{\tilde{x}\tilde{x}}^2 f_t(\tilde{x}^{(\text{new})}, \tilde{w}^{(\text{new})}) \preceq (1 + 3\delta_t \gamma) \nabla_{\tilde{x}\tilde{x}}^2 f_t(\tilde{x}^{(\text{old})}, \tilde{w}^{(\text{old})}) \quad (4.12)
\]

**Proof.** Equation (4.9) follows from the definition of $\delta_t$ and (4.7). Equations (4.10) and (4.11) follow from Lemma 2, the definition of $\gamma$, (4.7), and (4.8). Since $\delta_t \leq \frac{1}{8\gamma}$ this implies that slack or weight changes by more than a multiplicative factor of $\frac{1}{8}$ and therefore clearly $\{\tilde{s}^{(\text{new})}, \tilde{w}^{(\text{new})}\} \in \{S^0 \times \mathbb{R}^n_{>0}\}$.

To prove (4.12) note that (4.10) and (4.11) imply that
\[
\left(1 - \frac{r}{1+r} \delta_t \gamma\right) W_{(\text{old})} \preceq W_{(\text{new})} \preceq \left(1 + \frac{r}{1+r} \delta_t \gamma\right) W_{(\text{old})},
\]
\[
\left(1 - \frac{1}{1+r} \delta_t \gamma\right) S_{(\text{old})} \preceq S_{(\text{new})} \preceq \left(1 + \frac{1}{1+r} \delta_t \gamma\right) S_{(\text{old})}.
\]

Since $\nabla_{\tilde{x}\tilde{x}}^2 f_t(\tilde{x}, \tilde{w}) = A^T S^{-1} W S^{-1} A$ for $\tilde{x}, \tilde{w} \in \{S^0 \times \mathbb{R}^n_{>0}\}$ we have that
\[
\left(1 - \frac{r}{1+r} \delta_t \gamma\right) \frac{1}{2} \nabla_{\tilde{x}\tilde{x}}^2 f_t(\tilde{x}^{(\text{old})}, \tilde{w}^{(\text{old})}) \preceq \nabla_{\tilde{x}\tilde{x}}^2 f_t(\tilde{x}^{(\text{new})}, \tilde{w}^{(\text{new})}) \preceq \left(1 + \frac{r}{1+r} \delta_t \gamma\right) \frac{1}{2} \nabla_{\tilde{x}\tilde{x}}^2 f_t(\tilde{x}^{(\text{old})}, \tilde{w}^{(\text{old})}).
\]

Using that $0 \leq \delta_t \gamma \leq \frac{1}{8}$ and computing the Taylor series expansions yields that
\[
\frac{1}{2} \left(1 + \frac{1}{1+r} \delta_t \gamma\right) \leq 1 + 3\delta_t \gamma \quad \text{and} \quad \frac{1}{2} \left(1 - \frac{1}{1+r} \delta_t \gamma\right) \geq 1 - 3\delta_t \gamma.
\]
Using Lemma 5 we now bound how much a \(r\)-step improves centrality.

**Lemma 6** (Centrality Improvement of \(r\)-step). Let \(\{\vec{x}^{(\text{old})}, \vec{w}^{(\text{old})}\} = \text{step}_t(\vec{x}^{(\text{old})}, \vec{w}^{(\text{old})}, r)\) where

\[
\gamma = \gamma(\vec{x}^{(\text{old})}, \vec{w}^{(\text{old})}) \quad \text{and} \quad \delta_t = \delta_t(\vec{x}^{(\text{old})}, \vec{w}^{(\text{old})}) \leq \frac{1}{\delta \gamma}.
\]

We have the following bound on the change in energy

\[
\delta_t(\vec{x}^{(\text{new})}, \vec{w}^{(\text{new})}) \leq \frac{2}{1 + r} \cdot \gamma \cdot \delta_t^2.
\]

**Proof.** Let \(\vec{h}_t \equiv \vec{h}_t(\vec{x}^{(\text{old})}, \vec{w}^{(\text{old})})\) and let \(\Delta \equiv \mathbf{S}_{\text{old}}^{-1}(\vec{s}^{(\text{new})} - \vec{s}^{(\text{old})}) = \frac{1}{1 + r} \mathbf{S}_{\text{old}}^{-1} \mathbf{A} \vec{h}_t\). Recalling the definition of \(\text{step}_t\), we see that

\[
\frac{\vec{w}^{(\text{new})}}{\mathbf{s}^{(\text{new})}} = \frac{\vec{w}^{(\text{old})}}{\mathbf{s}^{(\text{old})}} - r \vec{w}_i \mathbf{\Delta}_i = \left( \frac{\vec{w}^{(\text{old})}}{\mathbf{s}^{(\text{old})}} \right) \cdot \left( \frac{1 - r \mathbf{\Delta}_i}{1 + \mathbf{\Delta}_i} \right)
\]

Using the definition of \(\vec{h}_t\) we have

\[
\nabla_x f_t(\vec{x}^{(\text{old})}, \vec{w}^{(\text{old})}) = \left( t \vec{c} - \mathbf{A}^T \mathbf{S}_{\text{old}}^{-1} \vec{w}^{(\text{old})} \right) \mathbf{A}^T \left( \mathbf{S}_{\text{old}}^{-1} \mathbf{W}_{\text{old}} \mathbf{S}_{\text{old}}^{-1} \mathbf{A} \right) \vec{h}_t
\]

\[
= - (1 + r) \mathbf{A}^T \mathbf{S}_{\text{old}}^{-1} \mathbf{W}_{\text{old}} \mathbf{\Delta}
\]

and therefore

\[
\nabla_x f_t(\vec{x}^{(\text{new})}, \vec{w}^{(\text{new})}) = t \vec{c} - \mathbf{A}^T \mathbf{S}_{\text{new}}^{-1} \vec{w}^{(\text{new})}
\]

\[
= \mathbf{A}^T \mathbf{S}_{\text{old}}^{-1} \mathbf{W}_{\text{old}} \left( \mathbf{I} - (1 + r) \mathbf{\Delta} - \mathbf{I} + \frac{(1 + r) \mathbf{\Delta}}{\mathbf{I} + \mathbf{\Delta}} \right)
\]

\[
= - (1 + r) \mathbf{A}^T \mathbf{S}_{\text{old}}^{-1} \mathbf{W}_{\text{old}} \frac{\mathbf{\Delta}^2}{\mathbf{I} + \mathbf{\Delta}}
\]

\[
= \mathbf{A}^T \mathbf{S}_{\text{old}}^{-1} \mathbf{W}_{\text{old}} \mathbf{S}_{\text{old}}^{-1} \mathbf{\Delta} \mathbf{\Delta} (\mathbf{I} + \mathbf{\Delta})^{-1} \mathbf{A} \vec{h}_t
\]

Now by Lemma 5 we know that

\[
\mathbf{A}^T \mathbf{S}_{\text{old}}^{-1} \mathbf{W}_{\text{old}} \mathbf{S}_{\text{old}}^{-1} \mathbf{A} \succeq (1 - 3 \delta \gamma) \mathbf{A}^T \mathbf{S}_{\text{old}}^{-1} \mathbf{W}_{\text{old}} \mathbf{S}_{\text{old}}^{-1} \mathbf{A}.
\]

Therefore by (4.15) and the fact that

\[
\mathbf{P}_{\mathbf{S}_{\text{old}}^{-1}} \mathbf{A} \left( \vec{w}^{(\text{old})} \right) = \mathbf{W}_{\text{old}}^{1/2} \mathbf{S}_{\text{old}}^{-1} \mathbf{A}^T \left( \mathbf{A} \mathbf{S}_{\text{old}}^{-1} \mathbf{W}_{\text{old}}^{-1} \right)^{-1} \mathbf{A}^T \mathbf{S}_{\text{old}}^{-1} \mathbf{W}_{\text{old}}^{1/2} \succeq \mathbf{I},
\]
we have

\[
\delta_t(\hat{x}^{\text{(new)}}, \hat{w}^{\text{(new)}}) = \| \nabla_x f_t(\hat{x}^{\text{(new)}}, \hat{w}^{\text{(new)}}) \| \left( A^T S_{\text{(new)}}^{-1} W^{\text{(new)}} S_{\text{(new)}}^{-1} A \right)^{-1} \\
= (1 - 3\delta_\gamma)^{-1/2} \left\| \text{diag}(\hat{\Delta})(I + \text{diag}(\hat{\Delta}))^{-1} W_{\text{(old)}}^{1/2} S_{\text{(old)}}^{-1} A \hat{b} \right\|_{P S_{\text{(old)}}^{-1} (\hat{w})^{\text{(old)}}} \\
\leq (1 - 3\delta_\gamma)^{-1/2} \left\| \text{diag}(\hat{\Delta})(I + \text{diag}(\hat{\Delta}))^{-1} W_{\text{(old)}}^{1/2} S_{\text{(old)}}^{-1} A \hat{b} \right\|_2 \\
\leq (1 - 3\delta_\gamma)^{-1/2} \frac{\| \hat{\Delta} \|_\infty}{1 - \| \hat{\Delta} \|_\infty} \left\| W_{\text{(old)}}^{1/2} S_{\text{(old)}}^{-1} A \hat{b} \right\|_2 \\
= (1 - 3\delta_\gamma)^{-1/2} \left( \frac{\| \hat{\Delta} \|_\infty}{1 - \| \hat{\Delta} \|_\infty} \delta_t \right) \leq \frac{2}{1 + r} \gamma \delta_t^2
\]

where in the last step we use that by Lemma 5, \( \| \hat{\Delta} \|_\infty \leq \frac{1}{1 + r} \delta \gamma \) and that \( \delta \leq \frac{1}{8} \gamma \) by assumption. \( \square \)

### 4.4 Weight Functions

In Sections 4.1, 4.2, and 4.3 we saw that to make our weighted path following schemes to converge quickly we need to maintain weights such that \( \| \hat{\Delta} \|_1 \), \( \gamma(\hat{s}, \hat{w}) \), and \( \delta_t(\hat{x}, \hat{w}) \) are small. Rather than showing how to do this directly, here we assume we have access to some fixed differentiable function for computing the weights and we characterize when such a weight function yields an efficient weighted path following scheme. This allows us decouple the problems of using weights effectively and computing weights that yield fast convergence rates.

For the remainder of this paper we assume that we have a fixed differentiable weight function \( \hat{g} : \mathbb{R}_r^{m \times m} \to \mathbb{R}_r^{m} \) from slacks to positive weights (see Section 6 for a description of the function we use). For slacks \( \hat{s} \in \mathbb{R}_r^{m} \) we let \( G(\hat{s}) \overset{\text{def}}{=} \text{diag}(\hat{g}(\hat{s})) \) denote the diagonal matrix associated with the slacks and we let \( G'(\hat{s}) \overset{\text{def}}{=} \text{J}_g(\hat{g}(\hat{s})) \) denote the Jacobian of the weight function with respect to the slacks.

For the weight function to be useful, in addition to yielding weights of small size, i.e. \( \| \hat{g}(\hat{s}) \|_1 \) bounded, and good condition number, i.e. \( \gamma(\hat{x}, \hat{g}(\hat{s}(\hat{x}))) \) small, we need to ensure that the weights do not change too much as we change \( \hat{x} \). For this, we use the operator norm of \( I + r^{-1} G(\hat{s})^{-1} G'(\hat{s}) S \) to measure for how much the weight function can diverge from the change in weights induced by a \( r \)-step, i.e. how consistent \( \hat{g} \) is to the central path. Lastly, to simplify the analysis we make a uniformity assumption that none of the weights are too big, i.e. \( \| \hat{g}(\hat{s}) \|_\infty \) is bounded. Formally we define a weight function as follows.

**Definition 7** (Weight Function). A weight function is a differentiable function from \( \hat{g} : \mathbb{R}_r^{m \times m} \to \mathbb{R}_r^{m} \) such that for constants \( c_1(\hat{g}), c_\gamma(\hat{g}), \) and \( c_r(\hat{g}), \) we have the following for all \( \hat{s} \in \mathbb{R}_r^{m} \):

- **Size**: The size \( c_1(\hat{g}) = \| \hat{g}(\hat{s}) \|_1 \).
- **Condition Number**: The condition number \( c_\gamma(\hat{g}) \) satisfies \( c_\gamma(\hat{g}) \geq 1 \) and \( \gamma(\hat{s}, \hat{g}(\hat{s})) \leq c_\gamma(\hat{g}) \).
- **Consistency**: The consistency \( c_r(\hat{g}) \) satisfies \( c_r(\hat{g}) \geq 1 \) and \( \forall r \geq c_r(\hat{g}) \) and \( \forall \hat{y} \in \mathbb{R}_r^{m} \)

\[
\| I + r^{-1} G(\hat{s})^{-1} G'(\hat{s}) S \| G(\hat{s}) \leq 1 \quad \text{and} \quad \| (I + r^{-1} G(\hat{s})^{-1} G'(\hat{s}) S) \hat{g} \|_\infty \leq \| \hat{g} \|_\infty + c_r \| \hat{y} \| G(\hat{s}) .
\]
• Uniformity: The weight function satisfies $\|g(\bar{s})\|_\infty \leq 2$

When the weight function $g$ is clear from context we often write $c_1$, $c_\gamma$, and $c_r$.

To get a sense of the magnitude of these parameters, in Theorem 16 we prove that there is a weight function with size $O(\sqrt{\text{rank} A})$, condition number $O(1)$ and consistency $O(\log (\frac{m}{\text{rank} A}))$; hence lemmas with polynomial dependence of condition number and consistency suffice for our purposes. However, for the remainder of this section and Section 5 we let the weight function be fixed but arbitrary.

Ideally, in our weighted path following schemes we would just set $\bar{w} = g(\bar{s})$ for any slacks $\bar{s}$ we compute. However, actually computing $g(\bar{s})$ may be expensive to compute exactly and therefore we analyze schemes that maintain separate weights, $\bar{w} \in \mathbb{R}^m_{>0}$ with the invariant that $\bar{w}$ is close to $g(\bar{s})$ multiplicatively. Formally, we define $\bar{\Psi}(\bar{s}, \bar{w})$ for all $\bar{s}, \bar{w} \in \mathbb{R}^m_{>0}$ by

$$
\bar{\Psi}(\bar{s}, \bar{w}) \overset{\text{def}}{=} \log(g(\bar{s})) - \log(\bar{w})
$$

and attempt to keep $\|\bar{\Psi}(\bar{s}, \bar{w})\|_\infty$ small despite changes that occur due to $r$-steps.

Now we wish to show that a $r$-step does not increase $\bar{\Psi}(\bar{s}, \bar{w})$ by too much. To do this, we first prove the following helper lemma.

**Lemma 8.** For a weight function $g$ and $\bar{s}_0, \bar{s}_1 \in S^0$ such that

$$
\epsilon_\infty \overset{\text{def}}{=} \|S_0^{-1}(\bar{s}_1 - \bar{s}_0)\|_\infty \leq \frac{1}{32c_r} \quad \text{and} \quad \epsilon_g \overset{\text{def}}{=} \|S_0^{-1}(\bar{s}_1 - \bar{s}_0)\|_{G(\bar{s}_0)} \leq \frac{\epsilon_\infty}{c_r}.
$$

we have

$$
\left\| \log \left( \frac{\bar{s}_1}{\bar{s}_0} \right) + \frac{1}{c_r} \log \left( \frac{g(\bar{s}_1)}{g(\bar{s}_0)} \right) \right\|_\infty \leq 3\epsilon_\infty \quad \text{and} \quad \left\| \log \left( \frac{\bar{s}_1}{\bar{s}_0} \right) + \frac{1}{c_r} \log \left( \frac{g(\bar{s}_1)}{g(\bar{s}_0)} \right) \right\|_{G(\bar{s}_0)} \leq (1 + 6c_r\epsilon_\infty) \epsilon_g.
$$

**Proof.** Let $\bar{p} : \mathbb{R}^m \rightarrow \mathbb{R}^m$ be defined for all $i \in [m]$ and $s \in \mathbb{R}^m_{>0}$ by

$$
\bar{p}(\bar{s})_i \overset{\text{def}}{=} \log(\bar{s}_i) + \frac{1}{c_r} \log(g(\bar{s}_i)).
$$

Clearly $J_{\bar{s}}[\bar{p}(\bar{s})] = S^{-1} + c_r^{-1}G^{-1}(\bar{s})G'(\bar{s})$. Therefore, letting $\bar{s}_t \overset{\text{def}}{=} \bar{s}_0 + t(\bar{s}_1 - \bar{s}_0)$ for all $t \in [0,1]$ we see that for all $i \in [0,1]$,

$$
\bar{p}(\bar{s}_t) = \bar{p}(\bar{s}_0) + \int_0^t \left[ S_t^{-1} + \frac{1}{c_r} G^{-1}(\bar{s}_t)G'(\bar{s}_t) \right] (\bar{s}_1 - \bar{s}_0) \, dt.
$$

Applying Jensen’s inequality and the definition of $\bar{p}$ then yields that for all $i \in [0,1]$ and any norm $\| \cdot \|$

$$
\left\| \log \left( \frac{\bar{s}_1}{\bar{s}_0} \right) + \frac{1}{c_r} \log \left( \frac{g(\bar{s}_1)}{g(\bar{s}_0)} \right) \right\| \leq \int_0^1 \left\| \left[ I + \frac{1}{c_r} G^{-1}(\bar{s}_t)G'(\bar{s}_t)S_t \right] S_t^{-1}(\bar{s}_1 - \bar{s}_0) \right\| \, dt. \tag{4.17}
$$

Now for all $t \in [0,1]$ define $\bar{a}_t \in \mathbb{R}^m_{>0}$ by

$$
\bar{a}_t \overset{\text{def}}{=} \log \left( \frac{\bar{s}_t}{\bar{s}_0} \right) - \frac{1}{c_r} \log \left( \frac{g(\bar{s}_t)}{g(\bar{s}_0)} \right)
$$

18
and let $M$ be the supremum over all $i \in [0, 1]$ such that $\| \log \tilde{g}(\tilde{s}_i) - \log \tilde{g}(\tilde{s}_0) \|_\infty \leq 3.5 c_r \epsilon_\infty$ for all $t \in [0, i]$. By Lemma 33 and the fact that $\epsilon_\infty \leq \frac{1}{32 c_r}$ this implies that $\| \mathbf{G}(\tilde{s}_i)^{-1} (\tilde{g}(\tilde{s}_i) - \tilde{g}(\tilde{s}_0)) \|_\infty \leq 4 c_r \epsilon_\infty$ and $\| \mathbf{G}(\tilde{s}_i)^{-1} (\tilde{g}(\tilde{s}_0) - \tilde{g}(\tilde{s}_i)) \|_\infty \leq 4 c_r \epsilon_\infty$ for all $i \in [0, M]$. Therefore, choosing $\| \cdot \|_{\mathbf{G}(\tilde{s}_0)}$ in (4.17) and applying Definition 7 yields that

$$
\| \tilde{a} \|_{\mathbf{G}(\tilde{s}_0)} < (1 + 4 c_r \epsilon_\infty)^{1/2} \int_0^i \left\| \mathbf{S}^{-1}_t (\tilde{s}_1 - \tilde{s}_0) \right\|_{\mathbf{G}(\tilde{s}_i)} dt \leq \frac{(1 + 4 c_r \epsilon_\infty)}{1 - \epsilon_\infty} \epsilon_g \leq (1 + 6 c_r \epsilon_\infty) \epsilon_g.
$$

Similarly, by choosing $\| \cdot \|_\infty$ in (4.17), we have $\forall i \in [0, M]$ that

$$
\| \tilde{a} \|_\infty < \int_0^i \left( \| \mathbf{S}^{-1}_t (\tilde{s}_1 - \tilde{s}_0) \|_\infty + c_r \| \mathbf{S}^{-1}_t (\tilde{s}_1 - \tilde{s}_0) \|_{\mathbf{G}(\tilde{s}_i)} \right) dt < \frac{\epsilon_\infty}{1 - \epsilon_\infty} + \sqrt{\frac{1 + 4 c_r \epsilon_\infty}{1 - \epsilon_\infty}} c_r \epsilon_g \leq 2.2 \epsilon_\infty
$$

By the definition of $\tilde{a}_i$, the triangle inequality, and Lemma 33 we then have that

$$
\| \log (\tilde{g}(\tilde{s}_i)) - \log (\tilde{g}(\tilde{s}_0)) \|_\infty < c_r \epsilon_\infty + \| \log (\tilde{s}_i) - \log (\tilde{s}_0) \|_\infty < 3.5 c_r \epsilon_\infty.
$$

Since $\tilde{g}$ is continuous we have that $M = 1$ and the result follows. □

Using this lemma we bound on how much a $r$-step increases $\tilde{\Psi}$ as follows

**Lemma 9.** Let $\{ \tilde{x}^{(new)}, \tilde{w}^{(new)} \} = \text{step}_i (\tilde{x}^{(old)}, \tilde{w}^{(old)}, c_r)$ where

$$
\delta_t \overset{\text{def}}{=} \delta_t (\tilde{x}^{(old)}, \tilde{w}^{(old)}) \leq \frac{1}{64 c_r c_r} \quad \text{and} \quad \epsilon \overset{\text{def}}{=} \| \log (\tilde{g}(\tilde{s}^{(old)})) - \log (\tilde{w}^{(old)}) \|_\infty \leq \frac{1}{8}.
$$

Letting

$$
\Delta \overset{\text{def}}{=} \log \left( \frac{\tilde{g}(\tilde{s}^{(new)})}{\tilde{g}(\tilde{s}^{(old)})} \right) - \log \left( \frac{\tilde{w}^{(new)}}{\tilde{w}^{(old)}} \right) = \tilde{\Psi}(\tilde{s}^{(new)}, \tilde{w}^{(new)}) - \tilde{\Psi}(\tilde{s}^{(old)}, \tilde{w}^{(old)}),
$$

we have

$$
\| \Delta \|_\infty \leq 4 c_r \epsilon_\infty \delta_t \quad \text{and} \quad \| \Delta \|_{\mathbf{W}^{(new)}} \leq \frac{(1 + \epsilon) c_r}{1 + c_r} \delta_t + 17 c_r c_r \delta_r^2.
$$

**Proof.** Recall the following definition of weight condition number

$$
\gamma(\tilde{s}, \tilde{w}) = \max_{i \in [m]} \| \mathbf{W}^{-1/2} \tilde{a}_i \mathbf{S}^{-1} \mathbf{A}(\tilde{w}) \|_{\mathbf{P}^{-1} \mathbf{S}^{-1} \mathbf{A}^T \mathbf{S}^{-1} \mathbf{A}^T} \leq \frac{3}{8}.
$$

Since $\| \log (\tilde{g}(\tilde{s}^{(old)})) - \log (\tilde{w}^{(old)}) \|_\infty \leq \frac{3}{8}$ by Lemma 33 we have

$$
\gamma(\tilde{s}^{(old)}, \tilde{w}^{(old)}) \leq 2 \gamma(\tilde{s}^{(old)}, \tilde{g}(\tilde{s}^{(old)})) \leq 2 c_r.
$$

Therefore, since $\delta_t \leq \frac{1}{64 c_r c_r}$ by Lemma 5 and (4.18) we have

$$
\| \mathbf{W}^{-1} (\tilde{w}^{(new)} - \tilde{w}^{(old)}) \|_\infty \leq \frac{2 c_r \delta_t}{1 + c_r} \leq \frac{1}{2} \quad \text{and} \quad \| \mathbf{S}^{-1} (\tilde{s}^{(new)} - \tilde{s}^{(old)}) \|_\infty \leq \frac{2 c_r \delta_t}{1 + c_r} \leq \frac{1}{2}.
$$
Recalling that \( W^{-1}_{(old)}(\vec{w}^{(new)} - \vec{w}^{(old)}) = -c_r S^{-1}_{(old)}(\vec{s}^{(new)} - \vec{s}^{(old)}) \) and using that \( c_r \geq 1 \) and \( \epsilon - \epsilon^2 \leq \log(1 + \epsilon) \leq \epsilon \) for \( |\epsilon| < \frac{1}{2} \) we have that for all \( i \in [m] \)
\[
\left| \log \left( \frac{w_i^{(new)}}{w_i^{(old)}} \right) + c_r \log \left( \frac{s_i^{(new)}}{s_i^{(old)}} \right) \right| = \left| \log \left( 1 - c_r \frac{s_i^{(new)} - s_i^{(old)}}{s_i^{(old)}} \right) + c_r \log \left( 1 + \frac{s_i^{(new)} - s_i^{(old)}}{s_i^{(old)}} \right) \right|
\leq 2c_r \left| \frac{s_i^{(new)} - s_i^{(old)}}{s_i^{(old)}} \right|^2 \quad (4.20)
\]

Letting \( \| \cdot \| \) denote either \( \| \cdot \|_\infty \) or \( \| \cdot \|_{W_{(old)}} \), recalling that \( \| S^{-1}_{(old)}(\vec{s}^{(new)} - \vec{s}^{(old)}) \|_\infty \leq \frac{2c_r \delta_t}{1 + c_r} \leq \frac{2c_r \delta_t}{c_r} \), and applying (4.20) yields
\[
\| \vec{\Delta} \| \leq \left\| c_r \log \left( \frac{s_i^{(new)}}{s_i^{(old)}} \right) + \log \left( \frac{\vec{g}(\vec{s}^{(new)})}{\vec{g}(\vec{s}^{(old)})} \right) \right\| + \left\| \log \left( \frac{\vec{w}^{(new)}}{\vec{w}^{(old)}} \right) + c_r \log \left( \frac{\vec{s}^{(new)}}{\vec{s}^{(old)}} \right) \right\|
\leq c_r \left\| \log \left( \frac{s_i^{(new)}}{s_i^{(old)}} \right) + \frac{1}{c_r} \log \left( \frac{\vec{g}(\vec{s}^{(new)})}{\vec{g}(\vec{s}^{(old)})} \right) \right\| + 4c_r c_r \delta_t \| S^{-1}_{(old)}(\vec{s}^{(new)} - \vec{s}^{(old)}) \| . \quad (4.21)
\]

By Lemma 5 and (4.19) \( \vec{s}^{(old)} \) and \( \vec{s}^{(new)} \) meet the conditions of Lemma 8 with \( \epsilon_g = \frac{\delta_t}{1 + c_r} \) and \( \epsilon_\infty = \frac{2c_r \epsilon_\infty}{1 + c_r} \). Therefore letting \( \| \cdot \| \) be \( \| \cdot \|_\infty \) in (4.21) we have
\[
\| \vec{\Delta} \|_\infty \leq 3c_r \epsilon_\infty + 4c_r c_r \delta_t \frac{2c_r \delta_t}{1 + c_r} \leq 7c_r c_r \delta_t .
\]

Similarly, letting \( \| \cdot \| \) be \( \| \cdot \|_{W_{(old)}} \) in (4.19) and noting that by definition of \( \epsilon \) and Lemma 33 we have \( W_{(old)} \leq (1 + \epsilon + \epsilon^2)G(\vec{s}^{(old)}) \) yields
\[
\| \vec{\Delta} \|_{W_{(old)}} \leq \left( \sqrt{1 + \epsilon + \epsilon^2} \right) c_r \epsilon_g \left( 1 + 6c_r \epsilon_\infty \right) + 4c_r c_r \delta_t^2 
\leq \left( 1 + \epsilon \right) \frac{c_r}{1 + c_r} \delta_t + 15c_r c_r \delta_t^2 .
\]

Finally, noting that \( \| W^{-1}_{(old)}(\vec{w}^{(new)} - \vec{w}^{(old)}) \|_\infty \leq 2c_r \delta_t \) yields the result. \( \square \)

4.5 Centering Using Exact Weights

Here we bound the rate of convergence rate of path following assuming that we can compute the weight function \( \vec{g} \) exactly. We start by providing a basic lemma regarding how the Newton step size changes as we change \( \vec{w} \).

**Lemma 10 (Effect of Weight Change).** Let \( \vec{x} \in S^0 \) and let \( \vec{w}^{(old)}, \vec{w}^{(new)} \in \mathbb{R}^m_{>0} \) with
\[
\epsilon_\infty \overset{\text{def}}{=} \| \log(\vec{w}^{(new)}) - \log(\vec{w}^{(old)}) \|_\infty \leq \frac{1}{2} \quad (4.22)
\]

it follows that
\[
\delta_t(\vec{x}, \vec{w}^{(new)}) \leq (1 + \epsilon_\infty) \left[ \delta_t(\vec{x}, \vec{w}^{(old)}) + \| \log(\vec{w}^{(new)}) - \log(\vec{w}^{(old)}) \|_{W_{(old)}} \right]
\]
Proof. Let $H_{(old)} \equiv A^T S^{-1} W_{(old)} S^{-1} A$ and let $H_{(new)} \equiv A^T S^{-1} W_{(new)} S^{-1} A$. By the definition of $\delta_t$ and the triangle inequality we have

$$
\delta_t(\tilde{x}, \tilde{w}^{(new)}) = \|t\tilde{c} - A^T S^{-1} \tilde{w}^{(new)}\|_{H^{-1}_{(new)}} \\
\leq \|t\tilde{c} - A^T S^{-1} \tilde{w}^{(old)}\|_{H^{-1}_{(new)}} + \|A^T S^{-1} \tilde{w}^{(new)} - A^T S^{-1} \tilde{w}^{(old)}\|_{H^{-1}_{(new)}} \quad (4.23)
$$

By definition of $\epsilon_{\infty}$ and Lemma 33 $H_{(new)} \leq (1 + \epsilon_{\infty})^2 H_{(old)}$ and therefore

$$
\|t\tilde{c} - AS^{-1} \tilde{w}^{(old)}\|_{H^{-1}_{(new)}} \leq (1 + \epsilon_{\infty})\delta_t(\tilde{x}, \tilde{w}^{(old)}). \quad (4.24)
$$

Furthermore, since $P_{AS^{-1}} (\tilde{w}^{(new)}) \leq 1$ we have

$$
\|A^T S^{-1} \tilde{w}^{(new)} - A^T S^{-1} \tilde{w}^{(old)}\|_{H^{-1}_{(new)}} = \|W^{-1/2}_{(new)} (\tilde{w}^{(new)} - \tilde{w}^{(old)})\|_{P_{AS^{-1}} (\tilde{w}^{(new)})} \\
\leq \frac{1}{\sqrt{W^{(new)}}} \frac{1}{\sqrt{W^{(old)}}} \|W^{(new)} - W^{(old)}\|_{W_{(old)}} \quad (4.25)
$$

Using that $\frac{(e^x - 1)^2}{e^x} \leq (1 + |x|)^2 x^2$ for $|x| \leq \frac{1}{2}$ and letting $x = [\log(\tilde{w}^{(new)}) - \log(\tilde{w}^{(old)})]$, we have

$$
\frac{1}{\sqrt{W^{(new)}}} \frac{1}{\sqrt{W^{(old)}}} \|W^{(new)} - W^{(old)}\|_{W_{(old)}} \leq (1 + \epsilon_{\infty}) \|\log(\tilde{w}^{(new)}) - \log(\tilde{w}^{(old)})\|_{W_{(old)}} \quad (4.26)
$$

Combining (4.23), (4.24), (4.25), and (4.26) completes the proof.

With this lemma we can now show how much centering progress we make by just updating $\tilde{x}$ and using the weight function. Note that in this proof we are just using the $r$-step in the proof, not the algorithm itself. We will need to use the $r$-step itself only later when we drop the assumption that we can compute $\tilde{g}$ exactly.

**Theorem 11 (Centering with Exact Weights).** Fix a weight function $\tilde{g}$, let $\tilde{x}^{(old)} \in S^0$, and let

$$
\tilde{x}^{(new)} = centeringExact(\tilde{x}^{(old)})
$$

If

$$
\delta_t \equiv \delta_t(\tilde{x}^{(old)}, \tilde{g}(s^{(old)})) \leq \frac{1}{100c_x c_r^2}
$$

then

$$
\delta_t(\tilde{x}^{(new)}, \tilde{g}(s^{(new)})) \leq \left(1 - \frac{1}{4c_r}\right) \delta_t(\tilde{x}^{(old)}, \tilde{g}(s^{(old)})).
$$

21
Proof. Let $\{ \tilde{x}^{\text{new}}, \tilde{w}^{\text{new}} \} \in \{ S^0 \times R^m_{\geq 0} \}$ be the result of a $c_r$ step from $\{ \tilde{x}^{\text{old}}, \tilde{w}^{\text{old}} \} \in \{ S^0 \times R^m_{\geq 0} \}$. Note that this $\tilde{s}^{\text{new}}$ is the same as the $\tilde{s}^{\text{new}}$ in the theorem statement.

Now by Lemma 6 we have that

$$
\delta_t(\tilde{s}^{\text{new}}, \tilde{w}^{\text{new}}) \leq c_\gamma \delta_t^2.
$$

(4.27)

Furthermore, defining $\Delta$ as in Lemma 9 and noting that $\tilde{w}^{\text{old}} = \tilde{g}(\tilde{s}^{\text{old}})$ we have

$$
\Delta \overset{\text{def}}{=} \log \left( \frac{\tilde{g}(\tilde{s}^{\text{new}})}{\tilde{g}(\tilde{s}^{\text{old}})} \right) - \log \left( \frac{\tilde{w}^{\text{new}}}{\tilde{w}^{\text{old}}} \right) = \log \left( \frac{\tilde{g}(\tilde{s}^{\text{new}})}{\tilde{w}^{\text{new}}} \right).
$$

we see by Lemma 9 that

$$
\| \log(\tilde{g}(\tilde{s}^{\text{new}})/\tilde{w}^{\text{new}}) \|_\infty \leq 4c_\gamma c_r \delta_t \leq 1/2
$$

(4.28)

and

$$
\| \log(\tilde{g}(\tilde{s}^{\text{new}})/\tilde{w}^{\text{new}}) \|_{\tilde{w}^{\text{new}}} \leq \frac{c_r}{1 + c_r} \delta_t + 17c_\gamma c_r \delta_t^2.
$$

(4.29)

Applying Lemma 10 to (4.27), (4.28), and (4.29) we have

$$
\delta_t(\tilde{x}^{\text{new}}, \tilde{g}(\tilde{s}^{\text{new}})) \leq (1 + 4c_\gamma c_r \delta_t) \left[ c_\gamma \delta_t^2 + \frac{c_r}{1 + c_r} \delta_t + 17c_\gamma c_r \delta_t^2 \right]
$$

$$
\leq \frac{c_r}{1 + c_r} \delta_t + 25c_\gamma c_r \delta_t^2
$$

$$
\leq \left( 1 - \frac{1}{2c_r} + \frac{1}{4c_r} \right) \delta_t \leq \left( 1 - \frac{1}{4c_r} \right) \delta_t
$$

From this lemma we have that if $\delta_t(\tilde{x}, \tilde{g}(\tilde{s}))$ is $O(c_\gamma^{-1}c_r^{-2})$ then in $\Theta(c_r^{-1})$ iterations of $\text{CenteringExact}$ we can decrease $\delta_t(\tilde{x}, \tilde{g}(\tilde{s}))$ by a multiplicative constant. Furthermore by Lemma 1 we see that we can increase $t$ by a multiplicative $(1 + O(c_\gamma^{-1}c_r^{-1}c_1^{-1/2}))$ and maintain $\delta_t(\tilde{x}, \tilde{g}(\tilde{s})) = O(c_\gamma^{-1}c_r^{-2})$. Thus we can double $t$ and maintain $\delta_t(\tilde{x}, \tilde{g}(\tilde{s})) = O(c_\gamma^{-1}c_r^{-2})$ using $O(c_\gamma^{-1}c_r^{3}c_1^{-1/2})$ iterations of $\text{CenteringExact}$. In Section 7 we make this argument rigorously in the more general setting. First in the following section we show how to relax this requirement that $\tilde{g}$ is computed exactly.

## 5 Approximating the Weighted Central Path

In the previous section, we analyzed a weighted path following strategy assuming oracle access to a weight function we could compute exactly. However, to achieve the fastest running times for weighted path following presented in this paper, we will not be able to efficiently compute our desired weight function exactly. In this section we show how it suffices to compute multiplicative approximations to the weight function. Ultimately, having access to this “noisy oracle” will only cause us to lose polylogarithmic factors in the running time as compared to the “exact oracle” case.

This is a non-trivial statement as the weight function serves several roles in our weighted path following scheme. First, it ensures a good ratio between total weight $c_1$ and condition number $c_\gamma$. This allows us to take make large increases to the path parameter $t$ after which we can improve
centrality. Second, the weight function is consistent and does not differ too much from the $c_r$-update step direction. This allows us to change the weights between $c_r$-update steps without moving too far away from the central path. Given a multiplicative approximation to the weight function, this first property is preserved up to an approximation constant however this second property is not.

To effectively use multiplicative approximations to the weight function we cannot simply use the weight function directly. Rather we need to smooth out changes to the weights by using some slowly changing approximation to the weight function. In this section we show how this can be achieved in general. First, in Section 5.1, we present the smoothing problem in a general form that we call the chasing 0 game and we provide an effective strategy for playing this game. Then in Section 5.2 we show how to use this strategy to produce a weighted path following scheme that uses multiplicative approximations to the weight function.

5.1 The Chasing 0 Game

The chasing 0 game is as follows. There is player, an adversary, and a point $\vec{x} \in \mathbb{R}^m$. The goal of the player is to keep the point close to $\vec{0}$ in $\ell_\infty$ norm and the goal of the adversary tries to move $\vec{x}$ away from $\vec{0} \in \mathbb{R}^m$. The game proceeds for an infinite number of iterations where in each iteration $k$ the adversary moves the current point $\vec{x}^{(k)} \in \mathbb{R}^m$ to some new point $\vec{y}^{(k)} \in \mathbb{R}^m$ and the player needs to respond. The player does not know $\vec{x}^{(k)}$, $\vec{y}^{(k)}$, or the move of the adversary. All the player knows is that the adversary moved the point within some convex set $U^{(k)}$ and the player knows some $\vec{z}^{(k)} \in \mathbb{R}^m$ that is close to $\vec{y}^{(k)}$ in $\ell_\infty$ norm. With this information the player is allowed to move the point a little more than the adversary. Formally, the player is allowed to set the next point to $\vec{x}^{(k+1)} \in \mathbb{R}^m$ such that $\vec{z}^{(k)} - \vec{y}^{(k)} \in (1 + \epsilon)U$ for some fixed $\epsilon > 0$.

The question we would like to address is, how close the player can keep $\vec{x}^{(k+1)}$ to $\vec{0}$ in $\ell_\infty$ norm? In particular, we would like an efficient strategy for computing $\vec{z}^{(k)}$ such that $\|\vec{x}^{(k)}\|_\infty$ is bounded for all $k \geq 0$.

| Chasing 0 Game: |
|----------------|
| 1. Given $R > 0, \epsilon > 0, \vec{x}^{(0)} \in \mathbb{R}^m$. |
| 2. For $k = 1, 2, \cdots$ |
| 2a. The adversary announces symmetric convex set $U^{(k)} \subseteq \mathbb{R}^m$ and $\vec{w}^{(k)} \in U^{(k)}$. |
| 2b. The adversary sets $\vec{y}^{(k)} := \vec{x}^{(k)} + \vec{w}^{(k)}$. |
| 2c. The adversary announces $\vec{z}^{(k)}$ such that $\|\vec{z}^{(k)} - \vec{y}^{(k)}\|_\infty \leq R$. |
| 2d. The player chooses $\Delta^{(k)} \in (1 + \epsilon)U^{(k)}$. |
| 2e. The player sets $\vec{x}^{(k+1)} = \vec{y}^{(k)} + \Delta^{(k)}$. |

We show that assuming that the $U^{(k)}$ are sufficiently bounded then there is strategy that the player can follow to ensure that that $\|\vec{x}^{(k)}\|_\infty$ is never too large. Our strategy simply consists of taking “gradient steps” using the following potential function.

**Definition 12.** For any $\mu \geq 0$ let $p_\mu : \mathbb{R} \rightarrow \mathbb{R}$ and $\Phi_\mu : \mathbb{R}^m \rightarrow \mathbb{R}$ be given by

$$\forall x \in \mathbb{R} : \quad p_\mu(x) \overset{\text{def}}{=} e^{\mu x} + e^{-\mu x} \quad \text{and} \quad \Phi_\mu(\vec{x}) \overset{\text{def}}{=} \sum_{i \in [m]} p_\mu(x_i).$$

\(^{10}\)To apply this result to weighted central path following we let the current points $\vec{x}^{(k)}$ denote the difference between $\log(\vec{w})$ and $\log(\vec{y}(\vec{x}))$. The sets $U^{(k)}$ are then related to the $c_r$-update steps and the steps of the player are related to the weights the path following strategy picks.
In other words, for all \( k \) we simply set \( \Delta^{(k)} \) to be the vector in \((1 + \epsilon)U^{(k)}\) that best minimizes the potential function of the observed position, i.e. \( \Phi_\mu(\vec{x}^{(k)}) \) for an appropriate choice of \( \mu \). In the following theorem we show that this suffices to keep \( \Phi_\mu(\vec{x}^{(k)}) \) small and that small \( \Phi_\mu(\vec{x}^{(k)}) \) implies small \( \|\vec{x}^{(k)}\|_\infty \) and hence has the desired properties.

**Theorem 13.** Suppose that each \( U^{(k)} \) is a symmetric convex set that contains an \( \ell_\infty \) ball of radius \( r_k \) and is contained in a \( \ell_\infty \) ball of radius \( R_k \leq R \).

Let \( 0 < \epsilon < \frac{1}{9} \) and consider the strategy

\[
\Delta^{(k)} = (1 + \epsilon) \arg \min_{\vec{\Delta} \in U^{(k)}} \langle \nabla \Phi_\mu(\vec{\Delta}), \vec{\Delta} \rangle \quad \text{where} \quad \mu = \frac{\epsilon}{12R}.
\]

Let \( \tau \) be defined as \( \max_k R_k \) and suppose \( \Phi_\mu(\vec{x}^{(0)}) \leq \frac{8 m \tau}{\epsilon} \) (or more specifically \( \|\vec{x}^{(0)}\|_\infty \leq \frac{12R}{\epsilon} \log \left( \frac{4\tau}{\epsilon} \right) \) ) then

\[
\forall k \geq 0 : \quad \Phi_\mu(\vec{x}^{(k+1)}) \leq \left( 1 - \frac{\epsilon^2 \tau k}{24R} \right) \Phi_\mu(\vec{x}^{(k)}) + \epsilon m \frac{R_k}{3R} \leq \frac{8 m \tau}{\epsilon}.
\]

In particular, we have \( \|\vec{x}^{(k)}\|_\infty \leq \frac{12R}{\epsilon} \log \left( \frac{8 m \tau}{\epsilon} \right) \).

To prove Theorem 13 we first provide the following lemma regarding properties of the potential function \( \Phi_\mu \).

**Lemma 14 (Properties of the Potential Function).** For all \( x \in \mathbb{R}^m \), we have

\[
e^{-\mu \|\vec{x}\|_\infty} \leq \Phi_\mu(\vec{x}) \leq 2m e^{\mu \|\vec{x}\|_\infty} \quad \text{and} \quad \mu \Phi_\mu(\vec{x}) - \mu m \leq \|\nabla \Phi_\mu(\vec{x})\|_1 \tag{5.1}\]

Furthermore, for any symmetric convex set \( U \subseteq \mathbb{R}^m \) and any \( \vec{x} \in \mathbb{R}^m \), let \( \vec{z}^{\delta} = \arg\max_{\vec{z} \in U} \langle \vec{x}, \vec{z} \rangle \)

and \( \|\vec{x}\|_U \) be defined as \( \max_{\vec{y} \in U} \langle \vec{x}, \vec{y} \rangle \). Then for all \( \vec{x}, \vec{y} \in \mathbb{R}^m \) with \( \|\vec{x} - \vec{y}\|_\infty \leq \delta \leq \frac{1}{5\mu} \) we have

\[
e^{-\mu \delta} \|\nabla \Phi_\mu(\vec{y})\|_U - \mu m R \leq \|\nabla \Phi_\mu(\vec{x})\|_U \leq e^{\mu \delta} \|\nabla \Phi_\mu(\vec{y})\|_U + \mu e^{\mu \delta} m R. \tag{5.3}\]

**Proof.** First we note that for all \( x \in \mathbb{R} \) we have

\[
e^{\mu |x|} \leq p_\mu(x) \leq 2e^{\mu |x|} \quad \text{and} \quad p_\mu'(x) = \mu \text{sign}(x) \left( e^{\mu |x|} - e^{-\mu |x|} \right)
\]

and therefore we have (5.1).

Next let \( x, y \in \mathbb{R} \) such that \( |x - y| \leq \delta \). Note that \( |p_\mu'(x)| = p_\mu'(|x|) = \mu (e^{\mu |x|} - e^{-\mu |x|}) \) and since \( |x - y| \leq \delta \) we have that \( |x| = |y| + z \) for some \( z \in (-\delta, \delta) \). Using that \( p'(|x|) \) is monotonic in \( |x| \) we then have

\[
|p_\mu'(x)| = p_\mu'(|x|) = p_\mu'(|y| + z) \leq p_\mu'(|y| + \delta)
= \mu \left( e^{\mu |y|+\delta} - e^{-\mu |y|-\delta} \right) = e^{\mu \delta} p'(|y|) + \mu \left( e^{\mu \delta} - e^{-\mu |y|-\delta} \right)
\leq e^{\mu \delta} |p'(y)| + \mu e^{\mu \delta}. \tag{5.4}\]

\[\text{Footnote:}\]

\[\text{Footnote:}\]
By symmetry (i.e. replacing $x$ and $y$) this implies that
\[ |p'_\mu(x)| \geq e^{-\mu \delta} |p'(y)| - \mu \] (5.5)

Since $U$ is symmetric this implies that for all $i \in [m]$ we have sign($\nabla \Phi_\mu(y^i)$)$_i = \text{sign}(y_i)$. Therefore, if for all $i \in [n]$ we have sign($x_i$) = sign($y_i$), by (5.4) and (5.5), we see that (5.2) holds. On the other hand if sign($x_i$) $\neq$ sign($y_i$) then we know that $|x_i| \leq \delta$ and consequently $|p'_\mu(x)| \leq e^{\mu \delta} - e^{-\mu \delta} \leq \frac{\mu}{2}$ since $\delta \leq \frac{1}{5\mu}$. Thus, we have
\[
eq -\frac{\mu}{2} \leq \text{sign}(y) p'_\mu(x) \leq 0 \leq e^{\mu \delta} |p'_\mu(y)| + \mu e^{\mu \delta}.
\]

Taking inner product on both sides with $p'_\mu(y)$ and using definition of $\| \cdot \|_U$ and $\hat{\cdot}$, we get (5.2). Thus, (5.2) holds in general.

Finally we note that since $U$ is contained in a $\ell_\infty$ ball of radius $R$, we have $\|\bar{y}\|_1 \leq mR$ for all $\bar{y}$. Using this fact, (5.2), and the definition of $\| \cdot \|_U$, we obtain
\[
eq e^{-\mu \delta} \| \nabla \Phi_\mu(\bar{y}) \|_U - \mu mR \leq \left\langle \nabla \Phi_\mu(\bar{x}), \nabla \Phi_\mu(\bar{y}) \right\rangle \leq \| \nabla \Phi_\mu(\bar{x}) \|_U.
\]

By symmetry (5.3) follows.

Using Lemma 14 we prove Theorem 13.

Proof. [Theorem 13] For the remainder of the proof, let $\|\bar{x}\|_{U(k)} = \max_{\bar{y} \in U(k)} \langle \bar{x}, \bar{y} \rangle$ and $\bar{x}^{(k)} = \arg \max_{\bar{y} \in U(k)} \langle \bar{x}, \bar{y} \rangle$. Since $U(k)$ is symmetric, we know that $\bar{\Delta}^{(k)} = -(1 + \epsilon) \left( \nabla \Phi_\mu(\bar{x}^{(k)}) \right)^{\hat{\cdot}}$ and therefore by applying the mean value theorem twice we have that
\[
\Phi_\mu(\bar{x}^{(k+1)}) = \Phi_\mu(\bar{y}^{(k)}) + \left\langle \nabla \Phi_\mu(\bar{z}), \bar{x}^{(k+1)} - \bar{y}^{(k)} \right\rangle
\]
\[
= \Phi_\mu(\bar{x}^{(k)}) + \left\langle \nabla \Phi_\mu(\bar{y}), \bar{x}^{(k)} - \bar{y}^{(k)} \right\rangle + \left\langle \nabla \Phi_\mu(\bar{z}), \bar{x}^{(k+1)} - \bar{y}^{(k)} \right\rangle
\]
for some $\bar{y}$ between $\bar{y}^{(k)}$ and $\bar{x}^{(k)}$ and some $\bar{z}$ between $\bar{x}^{(k+1)}$ and $\bar{y}^{(k)}$. Now, using that $\bar{y}^{(k)} - \bar{x}^{(k)} \in U(k)$ and that $\bar{x}^{(k+1)} - \bar{y}^{(k)} = \bar{\Delta}^{(k)}$ we have
\[
\Phi_\mu(\bar{x}^{(k+1)}) \leq \Phi_\mu(\bar{x}^{(k)}) + \| \nabla \Phi_\mu(\bar{z}) \|_{U(k)} - (1 + \epsilon) \left\langle \nabla \Phi_\mu(\bar{z}), \left( \nabla \Phi_\mu(\bar{x}^{(k)}) \right)^{\hat{\cdot}} \right\rangle.
\]
(5.6)

Since $U^k$ is contained within the $\ell_\infty$ ball of radius $R_k$ Lemma 14 shows that
\[
\| \nabla \Phi_\mu(\bar{y}) \|_{U(k)} \leq e^{\mu R_k} \| \nabla \Phi_\mu(\bar{x}^{(k)}) \|_{U(k)} + m \mu R_k e^{\mu R_k}.
\]
(5.7)

Furthermore, since $\epsilon < \frac{1}{5}$ and $R_k \leq R$, by triangle inequality we have $\|\bar{z} - \bar{x}^{(k)}\|_\infty \leq (1 + \epsilon) R_k + R \leq 3R$ and $\|\bar{z}^{(k)} - \bar{x}^{(k)}\|_\infty \leq 2R$. Therefore, applying Lemma 14 twice yields that
\[
\left\langle \nabla \Phi_\mu(\bar{z}), \left( \nabla \Phi_\mu(\bar{x}^{(k)}) \right)^{\hat{\cdot}} \right\rangle \geq e^{-3\mu R_k} \| \nabla \Phi_\mu(\bar{x}^{(k)}) \|_{U(k)} - \mu m R_k
\]
\[
\geq e^{-5\mu R_k} \| \nabla \Phi_\mu(\bar{x}^{(k)}) \|_{U(k)} - 2\mu m R_k.
\]
(5.8)
Combining (5.6), (5.7), and (5.8) then yields that
\[ \Phi_{\mu}(\bar{x}^{(k+1)}) \leq \Phi_{\mu}(\bar{x}^{(k)}) - \left( (1 + \epsilon)e^{-5\mu R} - e^{\mu R} \right) \| \nabla \Phi_{\mu}(\bar{x}^{(k)}) \|_{U(k)} + m\mu R_ke^{\mu R} + 2(1 + \epsilon)m\mu R_k. \]
Since we chose \( \mu = \frac{\epsilon}{124R} \), we have
\[ 1 + \epsilon \leq \frac{\epsilon}{2} (1 + 5\mu R) + (1 + 6\mu R) \leq \frac{\epsilon}{2} e^{5\mu R} + e^{6\mu R}. \]
Hence, we have \( (1 + \epsilon)e^{-5\mu R} - e^{\mu R} \leq \frac{\epsilon}{2} \). Also, since \( 0 < \epsilon < \frac{1}{5} \) we have
\[ m\mu R_ke^{\mu R} + 2(1 + \epsilon)m\mu R_k \leq (e^{\mu R} + 2(1 + \epsilon)) m\mu R_k \leq \frac{7R_k}{24R}. \]
Thus, we have
\[ \Phi_{\mu}(\bar{x}^{(k+1)}) \leq \Phi_{\mu}(\bar{x}^{(k)}) - \frac{\epsilon}{2} \| \nabla \Phi_{\mu}(\bar{x}^{(k)}) \|_{U(k)} + \frac{7R_k}{24R}. \]
Using Lemma 14 and the fact that \( U_k \) contains a \( \ell_\infty \) ball of radius \( r_k \), we have
\[ \| \nabla \Phi_{\mu}(\bar{x}^{(k)}) \|_{U(k)} \geq r_k \| \nabla \Phi_{\mu}(\bar{x}^{(k)}) \|_1 \geq \frac{\epsilon r_k}{12R} \left( \Phi_{\mu}(\bar{x}^{(k)}) - m \right). \]
Therefore, we have that
\[ \Phi_{\mu}(\bar{x}^{(k+1)}) \leq \left( 1 - \frac{\epsilon^2 r_k}{24R} \right) \Phi_{\mu}(\bar{x}^{(k)}) + \frac{\epsilon r_k}{12R} m + \frac{7R_k}{24R}. \]
Hence, if \( \Phi_{\mu}(\bar{x}^{(k)}) \leq \frac{8m\tau}{\epsilon} \), we have \( \Phi_{\mu}(\bar{x}^{(k+1)}) \leq \frac{8m\tau}{\epsilon} \). Since \( \Phi_{\mu}(\bar{x}^{(0)}) \leq \frac{8m\tau}{\epsilon} \) by assumption we have by induction that \( \Phi_{\mu}(\bar{x}^{(k)}) \leq \frac{8m\tau}{\epsilon} \) for all \( k \). The necessary bound on \( \| \bar{x}^{(k)} \|_\infty \) then follows immediately from Lemma 14.

### 5.2 Centering Step With Noisy Weight

Here we show how to use the results of the previous section to perform weighted path following given access only to a multiplicative approximation of the weight function. In particular, we show how to use Theorem 13 to improve the centrality of \( \bar{x} \) while maintaining the invariant that \( \bar{w} \) is close to \( \bar{g}(\bar{x}) \) multiplicatively.

As in Section 4 given a feasible point, \( \{ \bar{x}, \bar{w} \} \in \{ S^0 \times \mathbb{R}^m_{>0} \} \), we measure the distance between the current weights, \( \bar{w} \in \mathbb{R}^m_{>0} \), and the weight function, \( \bar{g}(\bar{s}) \in \mathbb{R}^m_{>0} \), in log scale \( \bar{\Psi}(\bar{s}, \bar{w}) \coloneqq \log(\bar{g}(\bar{s})) - \log(\bar{w}) \). Our goal is to keep \( \| \bar{\Psi}(\bar{s}, \bar{w}) \|_\infty \leq K \) for some error threshold \( K \). We choose \( K \) to be just small enough that we can still decrease \( \delta_t(\bar{x}, \bar{w}) \) linearly and still approximate \( \bar{g}(\bar{s}) \), as in general it may be difficult to compute \( \bar{g}(\bar{s}) \) when \( \bar{w} \) is far from \( \bar{g}(\bar{s}) \). Furthermore, we ensure that \( \bar{\Psi} \) doesn’t change too much in either \( \| - \|_\infty \) or \( \| - \|_{\bar{W}_{\text{new}}} \) and thereby ensure that the centrality does not increase too much as we move \( \bar{w} \) towards \( \bar{g}(\bar{s}) \).

We meet these goals by playing the chasing 0 game where the vector we wish to keep near \( \bar{0} \) is \( \bar{\Psi}(\bar{s}, \bar{w}) \), the adversaries moves are \( c_t \)-steps, and our moves change \( \log(\bar{w}) \). The \( c_t \)-step decreases \( \delta_t \) and since we are playing the chasing 0 game we keep \( \bar{\Psi}(\bar{s}, \bar{w}) \) small. Finally, since by the rules of the chasing 0 game we do not move \( \bar{w} \) much more than \( \bar{g}(\bar{s}) \) has moved, we have by similar reasoning to the exact weight computation case, Theorem 11 that changing \( \bar{w} \) does not increase \( \delta_t \) too much. This inexact centering operation and the analysis are formally defined and analyzed below.
\[
\begin{align*}
(\tilde{x}^{(new)}, \tilde{w}^{(apx)}) &= \text{centeringInexact}(\tilde{x}^{(old)}, \tilde{w}^{(old)}, K, \text{approxWeight}) \\
1. R &= \frac{K}{120c_r c_c^2} \delta_t = \delta_t(\tilde{x}^{(old)}, \tilde{w}^{(old)}), \epsilon = \frac{1}{5c_r} \text{ and } \mu = \frac{e}{12R} \\
2. \{\tilde{x}^{(new)}, \tilde{w}^{(new)}\} &= \text{step}(\tilde{x}^{(old)}, \tilde{w}^{(old)}, c_r) \text{ as in Definition 4.} \\
3. \text{Let } U &= \{\tilde{y} \in \mathbb{R}^m \mid \|\tilde{y}\|_{\tilde{w}^{(new)}} \leq \frac{c_r + 1/16}{c_r + 1} \delta_t \text{ and } \|\tilde{y}\|_{\infty} \leq 4c_r c_c \delta_t\} \\
4. \text{Compute } \tilde{z} &= \text{approxWeight}(s, \tilde{w}, R). \\
5. \tilde{w}^{(apx)} := \exp(\log(\tilde{w}^{(new)})) + (1 + \epsilon) \arg \min_{\tilde{u} \in U} \langle \nabla \Phi_{\mu}(\log(\tilde{z}) - \log(\tilde{w}^{(new)})), \tilde{u} \rangle
\end{align*}
\]

Note that in step 5 in centeringInexact, we need to project a certain vector onto the intersection of ball, \(\|\cdot\|_{\tilde{w}^{(new)}}\), and box, \(\|\cdot\|_{\infty}\). In Section B we show that this can be computed in parallel in depth \(O(1)\) and work \(O(m)\) and therefore this step is not a bottleneck in the computational cost of our weighted path following schemes.

**Theorem 15 (Centering with Inexact Weights).** Given current point \(\{\tilde{x}^{(old)}, \tilde{w}^{(old)}\} \in \{S^0 \times \mathbb{R}_0\}^m\), error parameter \(K \leq \frac{1}{2c_r}\), and approximate weight computation oracle, approxWeight, such that

\[\|\log(\text{approxWeight}(\tilde{s}, \tilde{w}, R)) - \log(\tilde{g}(\tilde{s}))\|_{\infty} \leq R \text{ for } \tilde{s}, \tilde{w} \in \mathbb{R}_0^m \text{ and } \|\log(\tilde{w}) - \log(\tilde{g}(\tilde{s}))\|_{\infty} \leq 2K,\]

assume that

\[\delta_t = \delta_t(\tilde{x}^{(old)}, \tilde{w}^{(old)}) \leq \frac{K}{240c_r c_c^2 \log(26c_r c_c m)} \quad \text{ and } \quad \Phi_{\mu} = \Phi_{\mu}(\tilde{x}^{(old)}, \tilde{w}^{(old)}) \leq 640c_r^2 c_c m^{3/2},\]

where \(\mu = 2R = 2 \log(26c_r c_c m)/K\). Let \((\tilde{x}^{(new)}, \tilde{w}^{(apx)}) = \text{centeringInexact}(\tilde{x}^{(old)}, \tilde{w}^{(old)}, K),\) then

\[\delta_t(\tilde{x}^{(new)}, \tilde{w}^{(apx)}) \leq \left(1 - \frac{0.5}{1 + c_r}\right) \delta_t\]

and

\[\Phi_{\mu}(\tilde{x}^{(new)}, \tilde{w}^{(apx)}) \leq \left(1 - \frac{\delta_t}{2400c_r^2 R \sqrt{m}}\right) \Phi_{\mu}(\tilde{x}^{(old)}) + \frac{4mc_r \delta_t}{15R} \leq 640c_r^2 c_c m^{3/2}.\]

Also, we have \(\|\log(\tilde{g}(\tilde{s}^{(new)})) - \log(\tilde{w}^{(apx)})\|_{\infty} \leq K.\)

**Proof.** By Lemma 9, we know that for a \(c_r\)-update step, we have \(\tilde{g}(\tilde{x}^{(new)}, \tilde{w}^{(new)}) - \tilde{g}(\tilde{x}^{(old)}, \tilde{w}^{(old)}) \in \tilde{U}\) where \(\tilde{U}\) is the symmetric convex set given by

\[\tilde{U} = \{\tilde{y} \in \mathbb{R}^m \mid \|\tilde{y}\|_{\tilde{w}^{(new)}} \leq C_w \quad \text{ and } \quad \|\tilde{y}\|_{\infty} \leq C_{\infty}\}\]

where

\[C_{\infty} = 4c_r c_c \delta_t \quad \text{ and } \quad C_w = \frac{c_r + 1/8}{c_r + 1} \delta_t + 17c_r c_c \delta_t^2.\]

Note that since \(\delta_t \leq K \left(240c_r c_c^2 \log(26c_r c_c m)\right)^{-1}\) we have

\[C_{\infty} \leq 4c_r c_c \left(\frac{K}{240c_r c_c^2 \log(26c_r c_c m)}\right) \leq \frac{K}{60c_r \log(26c_r c_c m)} = R.\]

Therefore \(\tilde{U}\) is contained in a \(\ell_{\infty}\) ball of radius \(R\). Again using the bound on \(\delta_t\) we have

\[C_w = \frac{c_r + 1/8}{c_r + 1} \delta_t + 17c_r c_c \delta_t^2 \leq \frac{c_r + 1/8}{c_r + 1} \delta_t + \frac{0.003}{c_r} \delta_t \leq \frac{c_r + 0.14}{c_r + 1} \delta_t.\]
Consequently, $\overline{U} \subseteq U$ where we recall that $U$ is the symmetric convex set defined by

$$U = \{ \tilde{y} \in \mathbb{R}^n \mid \|\tilde{y}\|_{W_{\text{new}}} \leq \frac{c_r + 0.14}{c_r + 1} \delta_t \quad \text{and} \quad \|\tilde{y}\|_\infty \leq 4c_r c_r \delta_t \}. $$

Therefore, we can play the chasing 0 game on $\overline{\Psi(s^{\text{old}}, \tilde{w}^{\text{old}})}$ attempting to maintain the invariant that $\|\overline{\Psi(s^{\text{old}}, \tilde{w}^{\text{old}})}\|_\infty \leq K \leq \frac{1}{8c_r}$ without taking steps that are more than $1 + \epsilon$ times the size of $U$. We pick $\epsilon = \frac{1}{8c_r}$ so to not interfere with our ability to decrease $\delta_t$ linearly.

To use the chasing 0 game to maintain $\|\overline{\Psi(s^{\text{old}}, \tilde{w}^{\text{old}})}\|_\infty \leq K$ we need to ensure that $R$ satisfies the following

$$\frac{12R}{\epsilon} \log \left( \frac{8m \tau}{\epsilon} \right) \leq K$$

where here $\tau$ is as defined in Theorem 13. To bound $\tau$ we need to lower bound the radius of the $\ell_\infty$ ball that $U$ contains. Since $\|\bar{y}(s^{\text{old}})\|_\infty \leq 2$ by Definition 7 and since $\|\bar{\Psi}(x^{\text{old}}, \tilde{w}^{\text{old}})\|_\infty \leq \frac{1}{8}$ by assumption we have that $\|\bar{w}^{\text{old}}\|_\infty \leq 3$. By Lemma 5 we know that $\|\bar{w}^{\text{new}}\|_\infty \leq 4$ if $\delta_t \gamma \leq \frac{1}{8}$ and consequently

$$\forall u \in \mathbb{R}^n : \|u\|^2 \geq \frac{1}{4m} \|u\|_{W_{\text{new}}}^2.$$ 

Consequently, if $\|\bar{u}\|_\infty \leq \frac{\delta_t}{4\sqrt{m}}$, then $\bar{u} \in U$. Thus $U$ contains a a box of radius $\frac{\delta_t}{4\sqrt{m}}$ and since $U$ is contained in a box of radius $4c_r c_r \delta_t$, we have that $\tau \leq 16c_r c_r \sqrt{m}$ and consequently

$$\frac{12R}{\epsilon} \log \left( \frac{8m \tau}{\epsilon} \right) \leq 60c_r R \log \left( 640c_r^2 c_r m^{3/2} \right) \leq 120c_r R \log (26c_r c_r m) \leq K.$$ 

This proves that we meet the conditions of Theorem 13. Therefore, we have

$$\Phi_\mu(\overline{\Psi}(x^{\text{new}}, \tilde{w}^{\text{apx}})) \leq \left( 1 - \frac{\epsilon^2}{24R} \frac{\delta_t}{4\sqrt{m}} \right) \Phi_\mu(x^{\text{apx}}) + \epsilon m \frac{1}{3R} (4c_r c_r \delta_t)
\leq \left( 1 - \frac{\delta_t}{240c_r^2 R \sqrt{m}} \right) \Phi_\mu(x^{\text{apx}}) + \frac{4mc_r \delta_t}{15R}
\leq 640c_r^2 c_r m^{3/2}. $$

where we do not need to re-derive the last line because it follows from Theorem 13.

Consequently, $\|\overline{\Psi}(x^{\text{old}}, \tilde{w}^{\text{old}})\|_\infty \leq K$ and $\Phi_\mu(\overline{\Psi}(x^{\text{new}}, \tilde{w}^{\text{apx}})) \leq 640c_r^2 c_r m^{3/2}$. Since $K \leq \frac{1}{8}$, we have $\|G(s^{\text{old}})^{-1}(\tilde{w}^{\text{old}} - \tilde{y}(s^{\text{old}}))\|_\infty \leq 1.2$ and $\gamma(s^{\text{old}}, \tilde{w}^{\text{old}}) \leq 2c_r$. Consequently, by Lemma 6 we have

$$\delta_t(x^{\text{new}}, \tilde{w}^{\text{apx}}) \leq \gamma(x^{\text{old}}, \tilde{w}^{\text{old}}) \cdot \delta_t^2 \leq 2 \cdot c_r \cdot \delta_t^2$$

Let

$$\epsilon_\infty \overset{\text{def}}{=} \|\log(\tilde{w}^{\text{apx}}) - \log(\tilde{w}^{\text{new}})\|_\infty \quad \text{and} \quad \epsilon_w \overset{\text{def}}{=} \|\log(\tilde{w}^{\text{apx}}) - \log(\tilde{w}^{\text{new}})\|_{W_{\text{new}}}.$$ 

By our bounds on $U$, we have

$$\epsilon_\infty \leq (1 + \epsilon) R \leq \frac{1}{100c_r} \quad \text{and} \quad \epsilon_w = (1 + \epsilon) \left[ \frac{c_r + 0.14}{c_r + 1} \delta_t \right] \leq \frac{c_r + 0.37}{c_r + 1} \delta_t.$$ 

28
Using Lemma 10, we have that
\[
\delta_t(\hat{x}^{(\text{new})}, \hat{y}^{(\text{apx})}) \leq (1 + \epsilon_\infty) \left[ \delta_t(\hat{x}^{(\text{new})}, \hat{y}^{(\text{new})}) + \epsilon_w \right] \leq 3c_\gamma \delta_t ^2 + (1 + \epsilon_\infty) \epsilon_w \leq \left(1 + \frac{1}{100c_r} \right) \left( \frac{c_r + 0.34}{c_r + 1} \right) \delta_t + 3c_\gamma \delta_t ^2 \leq \left( \frac{c_r + 0.5}{c_r + 1} \right) \delta_t
\]

\[\square\]

6 A Weight Function for \( \tilde{O}(\sqrt{\text{rank}(A)L}) \) Convergence

Here, we present the weight function \( \tilde{g} : \mathbb{R}^m_{>0} \to \mathbb{R}^m_{>0} \) that when used in the framework proposed in Section 4 and Section 5 yields an \( \tilde{O}(\sqrt{\text{rank}(A)L}) \) iteration interior point method. In Section 6.1 we motivate and describe the weight function \( \tilde{g} \); in Section 6.2 we prove that \( \tilde{g} \) satisfies Definition 7 with nearly optimal \( c_1(\tilde{g}), c_\gamma(\tilde{g}) \), and \( c_r(\tilde{g}) \), and in Section 6.3 we show how to compute approximations to \( \tilde{g} \) efficiently.

6.1 The Weight Function

Our weight function was inspired by the volumetric barrier methods of [37, 1]. These papers considered using the volumetric barrier, \( \phi(\bar{s}) = -\log \det(A^T S^{-2} A) \), in addition to the standard log barrier, \( \phi(\bar{s}) = -\sum_{i \in [m]} \log s_i \). If viewed as a weight function the standard log barrier has a good condition number, 1, but a large size, \( m \), and the volumetric barrier has a worse condition number, \( \sqrt{m} \), but better total weight, \( n \). By carefully applying a weighted combination of these two barriers [37] and [1] achieved an \( \tilde{O}((m \text{rank}(A))^{1/4}) \) iteration interior point method at the cost of more expensive linear algebra in each iteration.

Instead of using a fixed barrier, our weight function \( \tilde{g} : \mathbb{R}^m_{>0} \to \mathbb{R}^m_{>0} \) is computed by solving a convex optimization problem whose optimality conditions imply both good size and good condition number. We define \( \tilde{g} \) for all \( \bar{s} \in \mathbb{R}^m_{>0} \) by

\[
\tilde{g}(\bar{s}) = \arg\min_{\bar{w} \in \mathbb{R}^m_{>0}} f(\bar{s}, \bar{w}) \quad \text{where} \quad f(\bar{s}, \bar{w}) = \bar{w}^T \bar{w} - \frac{1}{\alpha} \log \det(A_s^T W^\alpha A_s) - \beta \sum_{i \in [m]} \log w_i \quad (6.1)
\]

where here and in the remainder of this section we let \( A_s \overset{\text{def}}{=} S^{-1}A \) and the parameters \( \alpha, \beta \in \mathbb{R} \) are chosen later such that the following hold

\[
\alpha \in (0, 1) \quad , \quad \beta \in (0, 1) \quad , \quad \beta^{1-\alpha} \geq \frac{1}{2} \quad . \quad (6.2)
\]

To get a sense for why \( \tilde{g} \) has the desired properties, suppose for illustration purposes that \( \alpha = 1 \) and \( \beta = 0 \) and fix \( \bar{s} \in \mathbb{R}^m_{>0} \). Using Lemma 34 and setting the gradient of (6.1) to 0 we see that if \( \tilde{g} \) exists then

\[
\tilde{g}(\bar{s}) = \tilde{\sigma}_{A_s}(\tilde{g}(\bar{s})) = \text{diag} \left( (G(\bar{s}))^{1/2} A_s (A_s^T G(\bar{s}) A_s)^{-1} A_s^T (G(\bar{s}))^{1/2} \right)
\]

and consequently

\[
\max_i \left\| G^{-1/2} \bar{z}_i \right\|_{p_{A_s}(\bar{g})} = 1 \quad \text{and} \quad \gamma(\bar{s}, \tilde{g}(\bar{s})) = 1 \quad .
\]

\[13\] See Section 1.2 for further intuition.
Furthermore, since \( \| \vec{s}_A, (\vec{g}(\vec{s})) \|_1 = \text{rank}(A) \) we see that this would yield a weight function with good \( c_\gamma \) and \( c_1 \).

Unfortunately picking \( \alpha = 1 \) and \( \beta = 0 \) makes the optimization problem for computing \( \vec{g} \) degenerate. In particular for this choice of \( \alpha \) and \( \beta \), it is unclear if \( \vec{g}(\vec{s}) \) always exists. In the follow sections we will see that by picking better values for \( \alpha \) and \( \beta \) we can trade off how well \( \vec{g} \) performs as a weight function and how difficult it is to compute approximations to \( \vec{g} \).

### 6.2 Weight Function Properties

Here, we show that \( \vec{g} : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0} \) as given by (6.1) is a weight function with respect to Definition 7 and we bound the values of \( c_1(\vec{g}), c_\gamma(\vec{g}) \), and \( c_r(\vec{g}) \). The goal of this section is to prove the following.

**Theorem 16** (Properties of Weight Function). *Let us define \( \alpha \) and \( \beta \) by* \[
\alpha = 1 - \frac{1}{\log_2 \left( \frac{2m}{\text{rank}(A)} \right)} \quad \text{and} \quad \beta = \frac{\text{rank}(A)}{2m}.
\]

For this choice of parameters \( \vec{g} \) is a weight function meeting the criterion of Definition 7 with

- **Size**: \( c_1(\vec{g}) = 2 \text{rank}(A) \).
- **Condition Number**: \( c_\gamma(\vec{g}) = 2 \).
- **Consistency**: \( c_r(\vec{g}) = 2 \log_2 \left( \frac{2m}{\text{rank}(A)} \right) \).

We break the proof into several parts. In Lemma 17, we prove basic properties of \( \hat{f} \). In Lemma 18 we prove that \( \vec{g} \) is a weight function and bound its size. In Lemma 19 we bound the condition number of \( \vec{g} \) and in Lemma 20 we show that \( \vec{g} \) is consistent.

We start by computing the gradient and Hessian of \( \hat{f}(\vec{s}, \vec{w}) \) with respect to \( \vec{w} \).

**Lemma 17.** For all \( \vec{s} \in \mathbb{R}^m_{>0} \) and \( \vec{w} \in \mathbb{R}^m_{>0} \), we have
\[
\nabla_{\vec{w}} \hat{f}(\vec{s}, \vec{w}) = \left( \vec{1} - \frac{1}{\alpha} \Sigma \vec{w}^{-\alpha} \right) - \beta \vec{w}^{-1} \vec{1} = \left( \vec{1} - \Sigma \vec{w}^{-1} - \beta \vec{w}^{-1} \right) \vec{1}
\]

and
\[
\nabla^2_{\vec{w}\vec{w}} \hat{f}(\vec{s}, \vec{w}) = \Sigma - \alpha \frac{\Lambda_{\alpha}}{\alpha w} + \beta \vec{1}
\]

where \( \Sigma \overset{\text{def}}{=} \Sigma_{\vec{A}_s}(\vec{w}^\alpha \vec{1}) \) and \( \Lambda \overset{\text{def}}{=} \Lambda_{\vec{A}_s}(\vec{w}^\alpha \vec{1}) \).

**Proof.** Using Lemma 34 and the chain rule we compute the gradient of \( \nabla_{\vec{w}} \hat{f}(\vec{s}, \vec{w}) \) as follows
\[
\nabla_{\vec{w}} \hat{f}(\vec{s}, \vec{w}) = \left( \vec{1} - \frac{1}{\alpha} \Sigma \vec{w}^{-\alpha} \right) - \beta \vec{w}^{-1} \vec{1} = \left( \vec{1} - \Sigma \vec{w}^{-1} - \beta \vec{w}^{-1} \right) \vec{1}
\]

Next, using Lemma 36 and chain rule, we compute the following for all \( i, j \in [m] \),
\[
\frac{\partial \nabla_{\vec{w}} \hat{f}(\vec{s}, \vec{w})}{\partial w_j} = - \frac{\vec{w}_i \Lambda_{ij} \vec{w}_j^{-\alpha}}{w_i w_j} - \frac{\vec{w}_i \vec{w}_j^{-\alpha}}{w_i w_j} - \frac{\Lambda_{ij} \vec{w}_j}{w_i w_j} + \beta \vec{1}
\]

Consequently \( \nabla^2_{\vec{w}\vec{w}} \hat{f}(\vec{s}, \vec{w}) = \Sigma - \alpha \frac{\Lambda_{\alpha}}{\alpha w} + \beta \vec{1} \) as desired. \( \square \)
Using this lemma, we prove that $g$ is a weight function with good size.

**Lemma 18.** The function $g$ is a weight function meeting the criterion of Definition 7. For all $s \in \mathbb{R}_{>0}^m$ and $i \in [m]$ we have

$$\beta \leq g_i(s) \leq 1 + \beta \quad \text{and} \quad \|g(s)\|_1 = \text{rank}(A) + \beta \cdot m.$$  

Furthermore, for all $s \in \mathbb{R}_{>0}^m$, the weight function obeys the following equations

$$G(s) = (\Sigma_g + \beta I) I \quad \text{and} \quad G'(s) = -2G(s) (G(s) - \alpha \Lambda_g)^{-1} \Lambda_g S^{-1}$$

where $\Sigma_g \equiv \Sigma_A(G(s)\alpha I)$, $\Lambda_g \equiv \Lambda_A(G(s)\alpha I)$, and $G'(s)$ is the Jacobian matrix of $g$ at $s$.

**Proof.** By Lemma 32 and (6.2) we have that for all $\bar{w}, \bar{s} \in \mathbb{R}_{>0}^m$, 

$$\Sigma_A(\bar{w}) \succeq \Lambda_A(\bar{w}) \succeq \alpha \Lambda_A(\bar{w}).$$

Therefore, by Lemma 17, $\nabla^2_{\bar{w}\bar{w}} f(\bar{s}, \bar{w}) \succeq \beta W^{-2}$ and $f(\bar{s}, \bar{w})$ is convex for $\bar{w}, \bar{s} \in \mathbb{R}_{>0}^m$. Using Lemma 17, we see that for all $i \in [m]$ it is the case that

$$\left[ \nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w}) \right]_i = \frac{1}{w_i^2} (w_i - \sigma_i - \beta)$$

Since $0 \leq \sigma_i \leq 1$ for all $i$ by Lemma 32 and $\beta \in (0, 1)$ by (6.2), we see that if $\bar{w}_i \in (0, \beta)$ then $\left[ \nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w}) \right]_i$ is strictly negative and if $\bar{w}_i \in (1 + \beta, \infty)$ then $\left[ \nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w}) \right]_i$ is strictly positive. Therefore, for any $s \in \mathbb{R}_{>0}^m$, the $\bar{w}$ that minimizes this convex function $f(\bar{s}, \bar{w})$ lies in the box between $\beta$ and $1 + \beta$. Since $\hat{f}$ is strongly convex in this region, the minimizer is unique.

The formula for $G(s)$ follows by setting $\nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w}) = 0$ and the size of $g$ follows from the fact that $\|\bar{w}\|_1 = \text{tr}(P_A(G(s)\alpha I))$. Since $P_A(G(s)\alpha I)$ is a projection onto the image of $G(s)\alpha/2A_s$ and since $g(s) > 0$ and $\bar{s} > 0$, the dimension of the image of $G(s)\alpha/2A_s$ is the rank of $A$. Hence, we have that

$$\|g(s)\|_1 \leq \|\bar{w}\|_1 + \|\beta\|_1 = \text{rank}(A) + \beta \cdot m.$$  

To compute $G'(s)$, we note that for $\bar{w} \in \mathbb{R}_{>0}^m$ and $\Lambda_w \equiv \Lambda_WA(A^{-1}I)$, by Lemma 36 and chain rule, we get the following for all $i, j \in [m]$,

$$\frac{\partial \left[ \nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w}) \right]_i}{\partial s_j} = -w_i^{-1} \Lambda_{ij} s_j^2 \left( -2s_j^{-3} \right) = 2w_i^{-1} \Lambda_{ij} s_j^{-1}.$$  

Consequently, $J_s(\nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w})) = 2W^{-1} \Lambda \omega S^{-1}$ where $J_s$ denotes the Jacobian matrix of the function $\nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w})$ with respect to $s$. Since we have already shown that $J_{\bar{w}}(\nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w})) = \nabla^2_{\bar{w}\bar{w}} f(\bar{s}, \bar{w}) = W^{-1} (\Sigma_w + \beta I - \alpha \Lambda_w) W^{-1}$ is positive definite (and hence invertible), by applying the implicit function theorem to the specification of $g(s)$ as the solution to $\nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w}) = 0$, we have

$$G'(s) = -\left( J_{\bar{w}}(\nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w})) \right)^{-1} \left( J_s(\nabla_{\bar{w}} \hat{f}(\bar{s}, \bar{w})) \right) = -2G(s) (G(s) - \alpha \Lambda_g)^{-1} \Lambda g S^{-1}$$

Using Lemma 18 we now show that $g$ has a good condition number.
Lemma 19 (Weight Function Condition Number). For all $\bar{s} \in \mathbb{R}^m_0$, we have $\gamma(\bar{s}, \bar{g}(\bar{s})) \leq 2$.

Proof. Fix an arbitrary $\bar{s} \in \mathbb{R}^m_0$ and let $\bar{g} \overset{\text{def}}{=} \bar{g}(\bar{s})$, and $\Sigma \overset{\text{def}}{=} \Sigma_{A_s}(\bar{g}^\alpha)$. Recall that by Lemma 18 we know that $\bar{g} = (\Sigma + \beta I)\bar{\bar{I}}$, and $\beta \leq g_i \leq 1 + \beta \leq 2$ for all $i \in [m]$. Furthermore, since $\beta^{1-\alpha} \geq \frac{1}{2}$ and $\alpha \in (0, 1)$ by (6.1) and clearly $G = G^{1-\alpha}G^\alpha$ we have

$$\frac{1}{2}G^\alpha \leq \beta^{1-\alpha}G^\alpha \leq G \leq (2)^{1-\alpha}G^\alpha \leq 2G^\alpha \quad (6.3)$$

Applying this and using the definition of $P_{A_s}(\bar{g})$ yields

$$A_s(A_s^TGA_s)^{-1}A_s^T \leq 2A_s(A_s^TG^\alpha A_s)^{-1}A_s^T = 2G^{-\alpha/2}P_{A_s}(\bar{g}^\alpha)G^{-\alpha/2} \quad (6.4)$$

Hence, by definition of the weight condition number we have

$$\gamma(\bar{s}, \bar{g}) = \max_i \|G^{-1/2\Sigma}\|_{P_{A_s}(\bar{g})}$$

$$= \max_i \sqrt{\frac{1}{2}A_s(A_s^TGA_s)^{-1}A_s^T}$$

$$\leq \max_i \sqrt{2\frac{1}{\Sigma}G^{-\alpha/2}P_{A_s}(\bar{g}^\alpha)G^{-\alpha/2}}$$

$$= \max_i \sqrt{2\frac{\sigma_i}{\bar{g}_i}} \leq 2 \max_i \sqrt{\frac{\sigma_i}{\bar{g}_i}} \leq 2.$$  

Finally, we bound the consistency of $\bar{g}$.

Lemma 20 (Weight Function Consistency). For all $\bar{s} \in \mathbb{R}^m_0$, $\bar{g} \in \mathbb{R}^m$, $r \geq \frac{2}{1-\alpha}$, and

$$B \overset{\text{def}}{=} I + \frac{1}{r} G(\bar{s})^{-1} G'(\bar{s}) \bar{S},$$

we have

$$\|B\bar{g}\|_{G(\bar{s})} \leq \|\bar{g}\|_{G(\bar{s})} \quad \text{and} \quad \|B\bar{g}\|_{\infty} \leq \|\bar{g}\|_{\infty} + \frac{2}{1-\alpha} \|\bar{g}\|_{G(\bar{s})}.$$  

Proof. Fix an arbitrary $\bar{s} \in \mathbb{R}^m_0$ and let $\bar{g} \overset{\text{def}}{=} \bar{g}(\bar{s})$, $\bar{\sigma} \overset{\text{def}}{=} \bar{\sigma}_{A_s}(\bar{g}^\alpha)$, $\Sigma \overset{\text{def}}{=} \Sigma_{A_s}(\bar{g}^\alpha)$, $P \overset{\text{def}}{=} P_{A_s}(\bar{g}^\alpha)$, $\Lambda \overset{\text{def}}{=} \Lambda_{A_s}(\bar{g}^\alpha)$. Also, fix an arbitrary $\bar{g} \in \mathbb{R}^m_0$ and let $\bar{z} \overset{\text{def}}{=} B\bar{g}$.

By Lemma 18, $G' = -2G(G - \alpha\Lambda)^{-1}\Lambda S^{-1}$ and therefore

$$B = I + r^{-1}G^{-1} \left( -2G(G - \alpha\Lambda)^{-1}\Lambda S^{-1} \right) \bar{S}$$

$$= (G - \alpha\Lambda)^{-1}(G - \alpha\Lambda)(G - \alpha\Lambda)^{-1}L$$

$$= (G - \alpha\Lambda)^{-1}(G - \alpha\Lambda) \quad (6.5)$$

Furthermore, since $0 < 2r^{-1} \leq 1 - \alpha$ and $0 \leq \Lambda \leq \Sigma \leq G$, we have that

$$0 \leq G - (\alpha + 2r^{-1})\Lambda \leq G - \alpha\Lambda.$$  

Thus, $G - \alpha\Lambda$ is positive definite and therefore $\bar{z}$ is the unique vector such that

$$G - \alpha\Lambda \bar{z} = (G - (\alpha + 2r^{-1})\Lambda) \bar{g}.$$  

32
To bound $\|\tilde{z}\|_G$, we note that since $G \succ 0$ we have

$$
\left( I - \alpha G^{-1/2} \Lambda G^{-1/2} \right) G^{1/2} \tilde{z} = \left( I - (\alpha + 2r^{-1}) G^{-1/2} \Lambda G^{-1/2} \right) G^{1/2} \tilde{y}
$$

Furthermore, since $0 \preceq G^{-1/2} \Lambda G^{-1/2} \preceq I$, we have that

$$
0 \preceq I - (\alpha + 2r^{-1}) G^{-1/2} \Lambda G^{-1/2} \preceq I - \alpha G^{-1/2} \Lambda G^{-1/2}
$$

and consequently

$$
\|\tilde{z}\|_G = \left\| \left( I - \alpha G^{-1/2} \Lambda G^{-1/2} \right) G^{1/2} \tilde{z} \right\| (I - \alpha G^{-1/2} \Lambda G^{-1/2})^{-2}
\leq \left\| \left( I - \alpha G^{-1/2} \Lambda G^{-1/2} \right) G^{1/2} \tilde{z} \right\| (I - (\alpha + 2r^{-1}) G^{-1/2} \Lambda G^{-1/2})^{-2}
= \|G^{1/2} \tilde{y}\| = \|\tilde{y}\|_G.
$$

Therefore, $\|B\tilde{y}\|_G \leq \|\tilde{y}\|_G$ as desired.

Next, to bound $\|\tilde{z}\|_\infty$, we use that $\Lambda = \Sigma - P^{(2)}$ and $\tilde{y} = \tilde{\sigma} + \beta \tilde{t}$ and (6.5) to derive

$$(1 - \alpha) \Sigma \tilde{z} + \beta \tilde{z} + \alpha P^{(2)} \tilde{z} = (1 - \alpha - 2r^{-1}) \Sigma \tilde{y} + \beta \tilde{y} + (\alpha + 2r^{-1}) P^{(2)} \tilde{y}.
$$

Left multiplying this equation by $\tilde{t}_i^T$ for arbitrary $i \in [m]$ and using that $\tilde{\sigma}_i \geq 0$ then yields that

$$
((1 - \alpha) \tilde{\sigma}_i + \beta) |\tilde{z}_i| \leq \alpha |\tilde{t}_i^T P^{(2)} \tilde{z}| + \left| \left((1 - \alpha - 2r^{-1}) \tilde{\sigma}_i + \beta\right) \tilde{y}_i + (\alpha + 2r^{-1}) \tilde{t}_i^T P^{(2)} \tilde{y}\right|
\leq \alpha |P^{(2)} \tilde{z}|_i + \left| \left((1 - \alpha) \tilde{\sigma}_i + \beta\right) \|\tilde{y}\|_\infty + \left| \left(P^{(2)} \tilde{y}\right)_i\right| \right|
\leq \alpha \tilde{\sigma}_i \|\tilde{z}\|_\Sigma + \left| \left((1 - \alpha) \tilde{\sigma}_i + \beta\right) \|\tilde{y}\|_\infty + \tilde{\sigma}_i \|\tilde{y}\|_\Sigma\right|
\leq \left((1 - \alpha) \tilde{\sigma}_i + \beta\right) \|\tilde{y}\|_\infty + (1 + \alpha) \tilde{\sigma}_i \|\tilde{y}\|_G
$$

(\Sigma \preceq G and $\|\tilde{z}\|_G \leq \|\tilde{y}\|_G$)

Consequently,

$$
|\tilde{z}_i| \leq \|\tilde{y}\|_\infty + \frac{(1 + \alpha) \tilde{\sigma}_i}{(1 - \alpha) \tilde{\sigma}_i + \beta} \|\tilde{y}\|_G
\leq \|\tilde{y}\|_\infty + \frac{2}{1 - \alpha} \|\tilde{y}\|_G
$$

and therefore $\|B\tilde{y}\|_\infty = \|\tilde{z}\|_\infty \leq \|\tilde{y}\|_\infty + \frac{2}{1 - \alpha} \|\tilde{y}\|_G$.

From Lemmas 18, 19 and 20, the proof of Theorem 16 is immediate. Since $m \geq \text{rank}(A)$ we have $\log_2 (2m/\text{rank}(A)) \geq 1$ and $\alpha \in (0, 1)$. Furthermore $\beta \in (0, 1)$ and

$$
\beta^{1-\alpha} = \left(\frac{\text{rank}(A)}{2m}\right)^{1\left(\log_2(2m/\text{rank}(A))\right)} = \frac{1}{2}
$$

and therefore (6.2) is satisfied. Furthermore, for all $\tilde{s} \in \mathbb{R}^m_{\geq 0}$ we have $\|\tilde{g}(\tilde{s})\|_1 \leq 2 \cdot \text{rank}(A)$ by Lemma 18. The bounds on $c_\gamma(\tilde{g})$ and $c_r(\tilde{g})$ then follow from Lemma 19 and Lemma 20 respectively.
### 6.3 Computing The Weights

Here, we describe how to efficiently compute approximations to the weight function \( \vec{g} : \mathbb{R}^m_{\geq 0} \to \mathbb{R}^m_{\geq 0} \) as given by (6.1). The two main technical tools we use towards this end are the gradient descent method, Theorem 21, a standard result in convex optimization, and fast numerical methods for estimating leverage scores using the Johnson-Lindenstrauss Lemma, Theorem 24, a powerful tool in randomized numerical linear algebra.

Since the weight function, \( \vec{g} \), is defined as the minimizer of a convex optimization problem (6.1), we could use the gradient descent method directly to minimize \( \hat{f} \) and hence compute \( \vec{g} \). Indeed, in Lemma 23 we show how applying the gradient descent method in a carefully scaled space allows us to compute \( \vec{g}(\vec{s}) \) to high accuracy in \( \tilde{O}(1) \) iterations. Unfortunately, this result makes two assumptions to compute \( \vec{g}(\vec{s}) \): (1) we are given a weight \( \vec{w} \in \mathbb{R}^m_{\geq 0} \) that is not too far from \( \vec{g}(\vec{s}) \) and (2) we compute the gradient of \( \hat{f} \) exactly.

Assumption (1) is not an issue as we always ensure that \( \vec{g} \) does not change too much between calls to compute \( \vec{g} \) and therefore can always use our previous weights as the approximation to \( \vec{g}(\vec{s}) \).

However, naively computing the gradient of \( \hat{f} \) is computationally expensive and hence assumption (2) is problematic. To deal with this issue we use the fact that by careful application of Johnson-Lindenstrauss one can compute a multiplicative approximation to the gradient efficiently and in Theorem 25 we show that this suffices to compute an approximation to \( \vec{g} \) that suffices to use in our weighted path following scheme.

First we prove the theorem regarding gradient descent method we use in our analysis. This theorem shows that if we take repeated projected gradient steps then we can achieve linear convergence up to bounds on how much the hessian of the function changes over the domain of interest. \(^{14}\)

**Theorem 21** (Simple Constrained Minimization for Twice Differentiable Function [22]). Let \( H \) be a positive definite matrix and \( Q \subseteq \mathbb{R}^m \) be a convex set. Let \( f(\vec{x}) : Q \to \mathbb{R} \) be a twice differentiable function such that there are constants \( L \geq \mu \geq 0 \) such that for all \( \vec{x} \in Q \) we have \( \mu H \preceq \nabla^2 f(\vec{x}) \preceq L H \). If for some \( \vec{x}(0) \in Q \) and all \( k \geq 0 \) we apply the update rule

\[
\vec{x}(k+1) = \arg \min_{\vec{x} \in Q} \langle \nabla f(\vec{x}(k)), \vec{x} - \vec{x}(k) \rangle + \frac{L}{2} \| \vec{x} - \vec{x}(k) \|_H^2
\]

then for all \( k \geq 0 \) we have

\[
\| \vec{x}(k) - \vec{x}^* \|_H^2 \leq \left( 1 - \frac{\mu}{L} \right)^k \| \vec{x}(0) - \vec{x}^* \|_H^2.
\]

To apply this Theorem 21 to compute \( \vec{g}(\vec{s}) \) we first need to show that there is a region around the optimal point \( \vec{g}(\vec{s}) \) such that the Hessian of \( \hat{f} \) does not change too much.

**Lemma 22** (Hessian Approximation). For \( \| W^{-1}(\vec{g}(\vec{s}) - \vec{w}) \|_\infty \leq \frac{1}{12} \) we have

\[
\frac{2(1 - \alpha)}{3} W^{-1} \leq \nabla^2_{\vec{w} \vec{w}} \hat{f}(\vec{s}, \vec{w}) \leq \frac{3}{2} W^{-1}.
\]

**Proof.** From Lemma 17, we know that

\[
\nabla^2_{\vec{w} \vec{w}} \hat{f}(\vec{s}, \vec{w}) = W^{-1} (\Sigma + \beta I - \alpha \Lambda) W^{-1}
\]

\(^{14}\)Note that this theorem is usually stated with \( H = I \), i.e. the standard Euclidean norm rather than the one induced by \( H \). However, Theorem 21 can be proved by these standard results just by a change of variables.
where $\Sigma = \Sigma_{A_4}(\hat{w}^\alpha)$ and $\Lambda = \Lambda_{A_4}(\hat{w}^\alpha)$. Using $0 \preceq \Lambda \preceq \Sigma$, we have

$$(1 - \alpha)\Lambda^{-1}(\Sigma + \beta \Lambda)\Lambda^{-1} \preceq \nabla^2_{\hat{w}\hat{w}}\hat{f}(\hat{s}, \hat{w}) \preceq \Lambda^{-1}(\Sigma + \beta \Lambda)\Lambda^{-1}$$

Using that $\|\Lambda^{-1}(\hat{g}(\hat{s}) - \hat{w})\|_\infty \leq \frac{1}{12}$ and applying Lemma 18 we have

$$\Sigma + \beta \Lambda \preceq \left(1 - \frac{1}{12}\right)^{-2} \Sigma_{A_4}(\alpha) + \beta \Lambda \preceq \left(1 - \frac{1}{12}\right)^{-2} \Gamma \preceq \frac{3}{2} \Lambda$$

and

$$\Sigma + \beta \Lambda \succeq \left(1 - \frac{1}{12}\right)^{2} \Sigma_{A_4}(\alpha) + \beta \Lambda \succeq \left(1 - \frac{1}{12}\right)^{2} \Gamma \succeq \frac{2}{3} \Lambda.$$  

\[ \square \]

Combining Theorem 21 and Lemma 22, we get the following algorithm to compute the weight function using the exact computation of the gradient of $\hat{f}$. Note that this algorithm applies Theorem 21 multiple times as in each iteration we are taking a gradient step with respect to a different norm.

**Lemma 23** (Exact Weight Computation). Given $\hat{w}^{(0)} \in \mathbb{R}_0^m$ such that $\|\Lambda^{-1}(\hat{g}(\hat{s}) - \hat{w}^{(0)})\|_\infty \leq \frac{1 - \alpha}{24}$. Let

$$Q = \left\{ \hat{w} \in \mathbb{R}^m \mid \|\Lambda^{-1}(\hat{w} - \hat{w}^{(0)})\|_\infty \leq \frac{1 - \alpha}{24} \right\}.$$

For all $k \geq 0$ let

$$\hat{w}^{(k+1)} = \arg\min_{\hat{w} \in Q} \left\| \hat{w} - \frac{1}{2} \left( \hat{w}^{(k)} + \sigma_{A_4} \left( \left( \hat{w}^{(k)} \right)^\alpha \right) \right) \right\|^2_{\Lambda^{-1}}.$$

This implies that for all $k$,

$$\|\Gamma^{-1}(\hat{g}(\hat{s}) - \hat{w}^{(k)})\|_\infty^2 \leq 4m^2 \left(1 - \frac{1 - \alpha}{12}\right)^k.$$

**Proof.** Note that iterations of Theorem 21 can be rewritten as

$$\hat{w}^{(k+1)} = \arg\min_{\hat{w} \in Q} \left\| \hat{w} - \frac{1}{2} \left( \hat{w}^{(k)} + \sigma_{A_4} \left( \left( \hat{w}^{(k)} \right)^\alpha \right) \right) \right\|^2_{\Lambda^{-1}}$$

which is the same as in the statement of this lemma. To apply Theorem 21 we note that for any $\hat{w} \in Q$ the definition of $Q$ and the fact that $\alpha \in (0, 1)$ implies that $\left(1 - \frac{1}{12}\right)\Lambda(0) \preceq \Lambda \preceq \left(1 + \frac{1}{12}\right)\Lambda(0)$. Therefore Lemma 22 shows that for all $\hat{w}^{(k)} \in Q$,

$$\frac{1 - \alpha}{2} \Lambda^{-1}(k) \preceq \frac{2(1 - \alpha)}{3} \Lambda^{-1}(0) \preceq \nabla^2_{\hat{w}\hat{w}}\hat{f}(\hat{s}, \hat{w}) \preceq \frac{3}{2} \Lambda^{-1}(0) \preceq 2 \Lambda^{-1}(k).$$

Hence, Theorem 21 and inequality (6.6) shows that

$$\|\hat{w}^{(k+1)} - \hat{g}\|^2_{\Lambda^{-1}(k)} \leq \left(1 - \frac{1 - \alpha}{4}\right) \|\hat{w}^{(k)} - \hat{g}\|^2_{\Lambda^{-1}(k)}.$$
Since \( \| W_{(0)}^{-1}(\hat{g}(\bar{s}) - \bar{w}(0)) \|_\infty \leq \frac{1-\alpha}{24} \) and \( \bar{w}(k) \in Q \), we know that \( G \succeq (1 - \frac{1-\alpha}{24})^2 W_{(k)} \). Hence, we have

\[
\| \bar{w}(k) - \bar{g} \|_{G^{-1}}^2 \leq \left( 1 - \frac{1-\alpha}{24} \right)^{-2} \left( 1 - \frac{1-\alpha}{4} \right) \| \bar{w}(k-1) - \bar{g} \|_{G^{-1}}^2 \leq \left( 1 - \frac{1-\alpha}{12} \right) \| \bar{w}(0) - \bar{g} \|_{G^{-1}}^2
\]

The result follows from the facts that

\[
\| \bar{w}(0) - \bar{g} \|_{G^{-1}}^2 \leq m \| G \|_\infty \| G^{-1}(\bar{g} - \bar{w}(0)) \|_\infty^2 \leq \frac{m(1+\beta)}{(1-\frac{1-\alpha}{24})^2} \left( W_{(0)}^{-1}(\bar{g} - \bar{w}(0)) \right)_\infty^2
\]

and \( \| G^{-1}(\bar{w}(k) - \bar{g}) \|_\infty^2 \leq \beta^{-1} \| \bar{w}(k) - \bar{g} \|_{G^{-1}}^2 \) where \( \beta = \frac{\text{rank}(A)}{2m} \).

Unfortunately, we cannot use the previous lemma directly as computing \( \bar{\sigma}_A \) exactly is too expensive for our purposes. However, in [29] they showed that we can compute leverage scores, \( \bar{\sigma}_A \), approximately by solving only polylogarithmically many regression problems (See [19, 16] for more details). These results use the fact that the leverage scores of the the \( \ell_2 \) length of vector \( P_A(x)X_i \) and that by the Johnson-Lindenstrauss Lemma these lengths are persevered up to multiplicative error if we project these vectors onto certain random low dimensional subspace. Consequently, to approximate the \( \bar{\sigma}_A \) we first compute the projected vectors and then use it to approximate \( \bar{\sigma}_A \), and hence only need to solve \( \tilde{O}(1) \) regression problems. For completeness, we provide the algorithm and theorem here:

\begin{center}
\begin{tabular}{|l|}
\hline
\( \bar{\sigma}^{(apx)} = \text{computeLeverageScores}(A, \bar{x}, \epsilon) \) \\
1. Let \( k = \lceil 24 \log(m)/\epsilon^2 \rceil \).
2. Let \( \bar{q}^{(j)} \) be \( k \) random \( \pm 1/\sqrt{k} \) vectors of length \( m \).
3. Compute \( \bar{l}^{(j)} = (A^T X X A)^{-1} A^T \bar{X}^{1/2} \bar{q}^{(j)} \).
4. Compute \( \bar{p}^{(j)} = X^{1/2} A \bar{l}^{(j)}. \)
5. Return \( \bar{\sigma}_i^{(apx)} = \sum_{j=1}^{k} \left( \bar{p}_i^{(j)} \right)^2 \).
\hline
\end{tabular}
\end{center}

**Theorem 24 ([29]).** For \( 0 < \epsilon < 1 \) with probability at least \( 1 - \frac{1}{m} \) the algorithm computeLeverageScores returns \( \bar{\sigma}^{(apx)} \) such that for all \( i \in [m] \),

\[
(1 - \epsilon) \bar{\sigma}_A(\bar{x})_i \leq \bar{\sigma}_i^{(apx)} \leq (1 + \epsilon) \bar{\sigma}_A(\bar{x})_i.
\]

by solving only \( O(\epsilon^{-2} \cdot \log m) \) linear systems.

Now, we show that we can modify Lemma 23 to use computeLeverageScores and we prove that this still provides adequate error guarantees. Our weight computation and the analysis is as follows.
\[ \bar{w} = \text{computeWeight}(\bar{s}, \bar{w}^{(0)}, K) \]

1. Let \( c_r = 2 \log_2 \left( \frac{2m}{\text{rank}(A)} \right) \), \( \alpha = 1 - \frac{1}{\log_2 \left( \frac{2m}{\text{rank}(A)} \right)} \), \( \beta = \frac{\text{rank}(A)}{2m} \), and \( \epsilon = \frac{K}{48c_r \log \left( \frac{2m}{K} \right)} \).

2. \( Q = \left\{ \bar{w} \in \mathbb{R}^m \mid \left\| W^{-1}(0) (\bar{w} - \bar{w}^{(0)}) \right\|_\infty \leq \frac{1}{12c_r} \right\} \)

3. For \( j = 1 \) to \( k \) where \( k = \left\lceil 12c_r \log \left( \frac{4m}{K} \right) \right\rceil \)

3a. \( \bar{\sigma}(j) = \text{computeLeverageScores}(S^{-1}A, (\bar{w}^{(j)})^\alpha, \epsilon) \)

3b. \( \bar{w}^{(j)} = \arg \min_{\bar{w} \in Q} \left\| \bar{w} - \frac{1}{2} \left( \bar{w}^{(j-1)} + \bar{\sigma}(j) + \beta \bar{w}^{(j)} \right) \right\|^2 \left( W^{-1}_{(i-1)} \right) \)

4. Output \( \bar{w}^{(j)} \).

Note that the convex set \( Q \) is aligned with standard basis and hence the step 3b can be computed by explicit formula (6.8).

**Theorem 25** (Approximate Weight Computation). Let \( \bar{s} \in \mathbb{R}^m \), \( \left\| W^{-1}(0) (\bar{g}(\bar{s}) - \bar{w}^{(0)}) \right\|_\infty \leq \frac{1}{12c_r} \), and \( K \in (0, 1) \). The algorithm computeWeight(\( \bar{s}, \bar{w}^{(0)}, K \)) returns \( \bar{w} \) such that

\[ \left\| G(\bar{s})^{-1}(\bar{g}(\bar{s}) - \bar{w}) \right\|_\infty \leq K \]

with probability \( 1 - \frac{\tilde{O}(1)}{m} \).

The running time is dominated by the time needed to solve \( O(c_r^2 \log^3 (m/K) \log(m)/K^2) \) linear systems.

**Proof.** Consider an execution of computeWeight(\( \bar{s}, \bar{w}^{(0)}, K \)) where each computeLeverageScores computes \( \bar{\sigma}_{A_i} \left( ([\bar{w}^{(j)})^\alpha] \right) \) exactly, i.e. \( \bar{\sigma}(j) = \bar{\sigma}_{A_i} \left( ([\bar{w}^{(j)})^\alpha] \right), \) and let \( \bar{v}^{(j)} \) denote the \( \bar{w}^{(j)} \) computed during this idealized execution of computeWeight.

Now suppose that for all \( i \in [m] \) we have

\[ (1 - \epsilon)^M \bar{v}_i^{(j)} \leq \bar{w}_i^{(j)} \leq (1 + \epsilon)^M \bar{v}_i^{(j)} \] \hspace{1cm} (6.7)

for some \( M \geq 0 \) and \( j \in [k - 1] \). Since the objective function and the constraints for step 3b. are axis-aligned we can compute \( \bar{w}^{(j)} \) coordinate-wise and we see that

\[ \bar{w}^{(j+1)} = \text{median} \left( \left( 1 - \frac{1}{12c_r} \right) \bar{w}^{(0)}(i), \bar{w}^{(j)} + \frac{1}{2} \left( \bar{\sigma}_{A_i} \left( ([\bar{w}^{(j)})^\alpha] \right) + \beta \right), \left( 1 + \frac{1}{12c_r} \right) \bar{w}^{(0)}(i) \right) \] \hspace{1cm} (6.8)

where \( \text{median}(x, y, z) \) is equal to the median of \( x, y, \) and \( z \) for all \( i \in [m] \). By (6.7), (6.8), and the fact that \( (1 - \epsilon) \bar{\sigma}_{A_i} \left( ([\bar{w}^{(j+1)})^\alpha] \right) \leq \bar{\sigma}_i^{(j+1)} \leq (1 + \epsilon) \bar{\sigma}_{A_i} \left( ([\bar{w}^{(j+1)})^\alpha] \right) \) for all \( i \in [m] \), we have that

\[ (1 - \epsilon)^{M+1} \bar{v}_i^{(j+1)} \leq \bar{w}_i^{(j+1)} \leq (1 + \epsilon)^{M+1} \bar{v}_i^{(j+1)}. \]

Since \( \bar{w}^{(0)} = \bar{w}^{(0)} \) and since \( j \in [k - 1] \) was arbitrary we can apply induction and we have that for all \( j \in [k] \)

\[ (1 - \epsilon)^j \bar{v}_i^{(j)} \leq \bar{w}_i^{(j)} \leq (1 + \epsilon)^j \bar{v}_i^{(j)}. \]

\[ \text{Recall that } c_r = \frac{2}{1 - \alpha} = 2 \log \left( \frac{2m}{\text{rank}(A)} \right) \geq 2. \]
The algorithm \( O(\text{Computating Initial Weights}) \)

Theorem 26

Furthermore since \( \tilde{v}(k) \in Q \) we know that \( G(s) \succeq \left( 1 - \frac{1}{12c_r} \right)^2 V(k) \). Putting these together, applying Lemma 23, and recalling that \( k = \lceil 12c_r \log \left( \frac{4m}{K} \right) \rceil \) we have

\[
\| G(s)^{-1}(\tilde{g}(s) - \tilde{w}(k)) \|_\infty \leq \| G(s)^{-1}(\tilde{g}(s) - \tilde{v}(k)) \|_\infty + \| G(s)^{-1}(\tilde{v}(k) - \tilde{w}(k)) \|_\infty
\]

\[
\leq 2m \left( 1 - \frac{1}{6c_r} \right)^{k/2} + \left( 1 - \frac{1}{12c_r} \right)^{-2} \| V^{-1}(\tilde{v}(k) - \tilde{w}(k)) \|_\infty
\]

\[
\leq 2m \cdot \exp \left( - \frac{k}{12c_r} \right) + 1.5\epsilon
\]

\[
\leq \frac{K}{2} + 1.5\epsilon \lceil 12c_r \log \left( \frac{4m}{K} \right) \rceil \leq K
\]

\[\Box\]

Finally, we show how to compute an initial weight without having an approximate weight to help the computation. The algorithm \( \text{computeInitialWeight}(s, K) \) computes an initial weight in \( \tilde{O} \left( \sqrt{\text{rank}(A)} \right) \) iterations of \( \text{computeWeight} \) by computing \( \tilde{g} \) for a large enough value of \( \beta \) that the calculation is trivial and then decreasing \( \beta \) gradually.

\[
\tilde{w} = \text{computeInitialWeight}(s, K)
\]

1. Let \( c_r = 2 \log_2 \left( \frac{2m}{\text{rank}(A)} \right) \), \( \alpha = 1 - \frac{1}{\log_2 \left( \frac{2m}{\text{rank}(A)} \right)} \), \( \beta = 12c_r \) and \( \tilde{w} = \beta \tilde{v} \).

2. Loop until \( \beta = \frac{\text{rank}(A)}{2m} \cdot \frac{1}{1000c_r} \).

2a. \( \tilde{w} = \text{computeWeight}(s, \tilde{w}, \frac{1}{1000c_r}) \).

2b. \( \beta = \max \left\{ 1 - \frac{(1-\alpha)^{3/2}}{1000c_r \sqrt{\text{rank}(A)}} \beta, \frac{\text{rank}(A)}{2m} \right\} \).

3. Output \( \text{computeWeight}(s, \tilde{w}, K) \).

**Theorem 26** (Computating Initial Weights). For \( s \in \mathbb{R}_{\geq 0}^m \) and \( K > 0 \), with constant probability the algorithm \( \text{computeInitialWeight}(s, K) \) returns \( \tilde{w} \in \mathbb{R}_{\geq 0}^m \) such that

\[
\| G(s)^{-1}(\tilde{g}(s) - \tilde{w}) \|_\infty \leq K.
\]

The total running time of \( \text{computeInitialWeight}(s, K) \) is dominated by the time needed to solve \( \tilde{O}(\sqrt{\text{rank}(A)} \log(1/K)/K^2) \) linear systems.

**Proof.** Fix \( s \in \mathbb{R}_{\geq 0}^m \) and let \( A_s \overset{\text{def}}{=} S^{-1} A \). For all \( \beta > 0 \) let \( \tilde{g} : \mathbb{R}_{>0} \rightarrow \mathbb{R}^m \) be defined by\(^{16}\)

\[
\tilde{g}(\beta) \overset{\text{def}}{=} \arg \min_{\tilde{w} \in \mathbb{R}_{\geq 0}^m} \tilde{w}^T \tilde{w} - \frac{1}{\alpha} \log \det(A_s^T W^\alpha A_s) - \beta \sum_{i \in [m]} \log w_i
\]

The algorithm \( \text{computeInitialWeight}(s, K) \) maintains the invariant that before step 2a

\[
\| W^{-1}(\tilde{g}(\beta) - \tilde{w}) \|_\infty \leq \frac{1}{12c_r}.
\]

\(^{16}\)Note that early we assumed that \( \beta < 1 \) and here we use much larger values of \( \beta \). However, this bound on \( \beta \) was primarily to assist in bounding \( c_1 \) and does not affect this proof.
Since \( g(\beta) = \bar{g}(\beta) + \beta \) where \( \bar{g}(\beta) \overset{\text{def}}{=} \bar{g}_A(\bar{g}^A(\beta)) \), we have that for all \( i \in [m] \)
\[
\beta \leq g(\beta)_i \leq 1 + \beta.
\]
Therefore, in the step 1, the initial weight, \( \bar{w} = \beta \bar{1} \in \mathbb{R}^{m_A} \) satisfies the invariant \((6.9)\). After step 2a, by Theorem 25 we have
\[
\|G(\beta)^{-1}(\bar{g}(\beta) - \bar{w})\|_{\infty} \leq \frac{1}{50c_r}.
\]
(6.10)
Therefore, it suffices to prove that \( g(\beta) \) is close to \( g(\beta - \theta) \) for small \( \theta \).

To bound how much \( g(\beta) \) changes for small changes in \( \beta \) we proceed similarly to Lemma 18. First by the implicit function theorem and direct calculation we know that
\[
\frac{dg}{d\beta} = - \left( J_{\bar{g}}(\nabla \hat{f}(\bar{s}, \bar{w})) \right)^{-1} \left( J_{\bar{g}}(\nabla \hat{f}(\bar{s}, \bar{w})) \right) = G(\beta) (G(\beta) - \alpha A_g)^{-1} \bar{1}
\]
where \( A_g \overset{\text{def}}{=} A_{A_g}(G(\beta)^{\alpha} \bar{1}) \). Next to estimate how fast \( \bar{g} \) can change as a function of \( \beta \) we estimate (6.11) in a similar manner to Lemma 20. Note that
\[
G(\beta) - \alpha A_g \geq (1 - \alpha) G(\beta) \geq (1 - \alpha) \Sigma(\beta)
\]
where \( \Sigma(\beta) \overset{\text{def}}{=} \Sigma_{A_g}(\bar{g}^A(\beta)) \). Consequently,
\[
\left\| \left( G(\beta) - \alpha A_g \right)^{-1} \bar{1} \right\|_{\Sigma(\beta)}^2 \leq \frac{1}{1 - \alpha} \left\| \bar{1} \right\|_{\Sigma(\beta)}^2 = \frac{\text{rank}(A)}{1 - \alpha}.
\]
(6.12)

Using this estimate of how much \( \bar{g} \) changes in the \( \Sigma(\beta) \) norm, we now estimate how much \( \bar{g} \) changes in the \( \ell_\infty \) norm. Let \( \bar{z} \overset{\text{def}}{=} (G(\beta) - \alpha A_g)^{-1} \bar{1} \). Then, we have
\[
(1 - \alpha) \bar{s}_i(\beta) + \beta |\bar{z}_i| \leq |\alpha \bar{u}_i^T P^{(2)} \bar{z}| + 1 \\
\leq \alpha \bar{s}_i(\beta) \left( \bar{z} \right)_{\Sigma(\beta)} + 1.
\]

Using (6.12) and \( \alpha < 1 \), we have
\[
\left\| \frac{d \ln \bar{g}}{d\beta} \right\|_{\infty} = \left\| \bar{z} \right\|_{\infty} \leq \max \left( \frac{\alpha \left\| \bar{z} \right\|_{\Sigma(\beta)} + 1}{1 - \alpha} \cdot \frac{1}{\beta} \right) \leq \max \left( \frac{\sqrt{\text{rank} (A)}}{(1 - \alpha)^{3/2}} \cdot \frac{1}{\beta} \right).
\]

Using (6.10), integrating, and applying Lemma 33 we have that
\[
\left\| G(\beta - \theta)^{-1} (\bar{g}(\beta - \theta) - \bar{w}) \right\|_{\infty} \leq \frac{1}{12c_r}
\]
for \( \theta \leq \frac{(1 - \alpha)^{3/2} \beta}{1000 \sqrt{\text{rank} (A)}} \). Hence, this proves that step 2a preserves the invariant \((6.9)\) at step 2a. Hence, the algorithm satisfies the assumptions needed for Theorem 25 throughout and \texttt{computeWeight} ins step 2a works as desired. Since each iteration \( \beta \) decreased by \( \tilde{O} \left( \frac{1}{\sqrt{\text{rank} (A)}} \right) \) portion and the initial \( \beta \) is \( \tilde{O}(1) \) we see that the algorithm requires only \( \tilde{O} \left( \sqrt{\text{rank} (A)} \right) \) iterations. Using Theorem 25 to bound the total number of linear systems solved then yields the result.
7 The Algorithm

In this section we show how to put together the results of the previous sections to solve a linear program. First, in Section 7.1 we provide a path following routine that allows us to move quickly from one approximate central path point to another. Using this subroutine, in Section 7.2 we show how to obtain an algorithm for solving a linear program in \( O(\sqrt{\text{rank}(A)}L) \) iterations that consist of solving linear systems in the original constraint matrix. Furthermore, in Section 7.3 we show that our algorithm is amenable to acceleration techniques as in [35] and we obtain an even faster linear system solver by using fast matrix multiplication, low rank updates, and preconditioning. In the Appendix we provide additional proof details such as how these algorithm only require approximate linear system solvers (Appendix C), how to initialize our interior point technique and round approximate solutions to optimal ones (Appendix D), and how to solve a sequence of similar linear systems efficiently (Appendix E).

7.1 Path Following

We start by analyzing the running time of \texttt{pathFollowing} a subroutine for following the weighted central path.

\[
(x_{\text{new}}, w_{\text{new}}) = \text{pathFollowing}(x_{\text{old}}, w_{\text{old}}, t_{\text{start}}, t_{\text{end}})
\]

1. \( c_r = 2 \log_2 \left( \frac{2m}{\text{rank}(A)} \right) \), \( t = t_{\text{start}}, K = \frac{1}{24c_r} \).
2. While \( t < t_{\text{end}} \)
   2a. \( (x_{\text{new}}, w_{\text{apx}}) = \text{centeringInexact}(x_{\text{old}}, w_{\text{old}}, K, \text{computeWeight}) \)
   2b. \( t_{\text{new}} := t \left( 1 + \frac{10c_r^2 \log(52c_r m) \sqrt{\text{rank}(A)}}{\text{rank}(A)} \right) \).
   2c. \( x_{\text{old}} := x_{\text{new}}, w_{\text{old}} := w_{\text{apx}}, t := t_{\text{new}} \)
   2d. For every \( \frac{m}{c_r \log \left( \frac{n}{m} \right)} \) steps, check if the current \( x, w \) satisfies the \( \delta \) and \( \Phi \) invariants. If it does not satisfies, roll back to the last time the invariants were met.
3. Output \( (x_{\text{old}}, w_{\text{old}}) \).

**Theorem 27** (Main Result). \( \{x_{\text{old}}, w_{\text{old}}\} \in \{S^0 \times \psi_{>0}^m\} \) and \( t_{\text{start}} \leq t_{\text{end}} \). Suppose that

\[
\delta t_{\text{start}}(x_{\text{old}}, w_{\text{old}}) \leq \frac{1}{11520c_r^2 \log(52c_r m)} \quad \text{and} \quad \Phi(\psi(x_{\text{old}}, w_{\text{old}})) \leq 1280c_r^2 m^{3/2}
\]

where \( \mu = 2 \log(52c_r m) / K \). Let \( (x_{\text{new}}, w_{\text{apx}}) = \text{pathFollowing}(x_{\text{old}}, w_{\text{old}}, t_{\text{start}}, t_{\text{end}}) \), then

\[
\delta t_{\text{end}}(x_{\text{new}}, w_{\text{new}}) \leq \frac{1}{11520c_r^2 \log(52c_r m)} \quad \text{and} \quad \Phi(\psi(x_{\text{new}}, w_{\text{new}})) \leq 1280c_r^2 m^{3/2}.
\]

Furthermore, computing \((x_{\text{new}}, w_{\text{new}})\) takes \( \tilde{O} \left( \sqrt{\text{rank}(A)} \log \left( \frac{t_{\text{end}}}{t_{\text{start}}} \right) \right) \) iterations in expectation where the cost of each iteration is dominated by the time need to solve \( \tilde{O}(1) \) linear system solves.

**Proof.** This algorithm maintains the invariant that

\[
\delta t(x_{\text{old}}, w_{\text{old}}) \leq \frac{1}{11520c_r^2 \log(52c_r m)} \quad \text{and} \quad \Phi(\psi(x_{\text{old}}, w_{\text{old}})) \leq 1280c_r^2 m^{3/2}
\]
in each iteration in the beginning of the step (2a). Note that our oracle computeWeight satisfies the assumption of Theorem 15 since $2K \leq \frac{1}{12c_r}$. Hence, centeringInexact can use computeWeight to find the approximations of $\hat{g}(\hat{s}^{\text{new}})$. Hence, Theorem 15 shows that we have

$$
d_t(\hat{x}^{\text{new}}, \hat{w}^{\text{apx}}) \leq \left(1 - \frac{0.5}{1 + c_r}\right) \delta_t \quad \text{and} \quad \Phi_\mu(\hat{\Psi}(\hat{x}^{\text{new}}, \hat{w}^{\text{apx}})) \leq 1280c_r^2 m^{3/2}.
$$

Using the fact $c_1(\hat{g}) \leq 2 \text{rank}(A)$ and that $\hat{w}^{\text{new}}$ is within a multiplicative factor of two of $\hat{g}(\hat{s}^{\text{new}})$ by Lemma 1 we have

$$
d_t^{\text{new}}(\hat{x}^{\text{new}}, \hat{w}^{\text{apx}}) \\
\leq \left(1 + \frac{1}{10^5 c_r^4 \log (52c_rm) \sqrt{\text{rank}(A)}}\right) \left(1 - \frac{0.5}{1 + c_r}\right) \delta_t + \frac{\sqrt{\|\hat{w}^{\text{new}}\|_1}}{10^5 c_r^4 \log (52c_rm) \sqrt{\text{rank}(A)}} \\
\leq \frac{1}{11520c_r^3 \log (52c_rm)}
$$

With probability $(1 - \frac{1}{m})c_r \log \left(\frac{4m}{\epsilon}\right)$, computeWeight outputs a correct answer. Therefore, for each $\frac{m}{c_r \log \left(\frac{4m}{\epsilon}\right)}$ iterations there is constant probability that the whole procedure runs correctly. Hence, we only need to know how long it takes to check the current state satisfies $\delta_t$ and $\Phi_\mu$ invariants. We can check the $\delta_t$ easily using only 1 linear system solve. To check $\Phi_\mu$, we need to compute the weight function exactly. To do this, we use lemma 23 and note that computing the leverage scores exactly takes $m$ linear system solve. Therefore, the averaged cost of step 2d is just $\tilde{O}(1)$ linear system solves and this justified the total running time.

7.2 Solving a Linear Program

Here we show how to use the properties of pathFollowing proved in Theorem 27 to obtain a linear program solver. Given the previous theorem all that remains is to show how to get the initial central point and round the optimal point to a vertex. We defer much of the proof of how to obtain an initial point, deal with unbounded solutions, and round to an optimal vertex to Lemma 40 proved in Appendix D.

**Theorem 28.** Consider a linear programming problem of the form

$$
\min_{\hat{x} \in \mathbb{R}^n} \quad \hat{c}^T \hat{x} \quad \text{(7.1)}
$$

where $A \in \mathbb{R}^{m \times n}$, $\bar{b} \in \mathbb{R}^m$, and $\hat{c} \in \mathbb{R}^n$ have integer coefficients. Let $L$ denote the bit complexity of (7.1) and suppose that for any positive definite diagonal matrix $D \in \mathbb{R}^{m \times m}$ with condition number $2\tilde{O}(L)$ there is an algorithm solve$(A, \bar{b}, D, \epsilon)$ such that

$$
\|\text{solve}(A, \bar{b}, D, \epsilon) - (DA)^+ \bar{b}\|_{A^T D^2 A} \leq \epsilon \|\text{solve}(DA)^+ \bar{b}\|_{A^T D^2 A} \quad \text{(7.2)}
$$

in time $O(T \log(1/\epsilon))$ for any $\epsilon > 0$ with success probability greater than $1 - \frac{1}{m}$. Then, there is an algorithm to solve (7.1) in expected time $\tilde{O}\left(\sqrt{\text{rank}(A)}(T + \text{nnz}(A)) L\right)$, i.e., find the active constraints of an optimal solution or prove that the program is unfeasible or unbounded.
Using [21] as the Solve algorithm, we obtain an algorithm that solves (7.1) in time
\[
\tilde{O} \left( \sqrt{\text{rank}(A)} \left( \text{nnz}(A) + (\text{rank}(A))^\omega \right) L \right).
\]
where \( \omega < 2.3729/39 \) is the matrix multiplication constant.

Proof. Applying the Lemma 40 we obtain a modified linear program
\[
\min \left( \vec{c}(\text{new}), \vec{x} \right) \text{ given } A(\text{new}) \vec{x} \geq \vec{b}(\text{new})
\]
which is bounded and feasible with \( O(n) \) variables, \( O(m) \) constraints, \( O(\text{rank}(A)) \) rank and \( \tilde{O}(L) \) bit complexity. Also, we are given an explicit interior point \( \vec{x}_0 \).

To obtain an initial weighted central path point, we can use Theorem 26. However, \( \vec{x} \) may not be close to central path, i.e. \( \delta_t \) could be large. To fix this, we can temporarily change the cost function such that \( \delta_t = 0 \). In particular, we can set \( \vec{c}_\text{modified} = AS^{-1} \vec{w} \) and get \( \delta_t = 0 \) for this modified cost function. One can think of Theorem 27 as showing that we can get the central path point from a certain cost function at start \( \vec{c} \) to another cost function at end \( \vec{c} \) in time that depends only logarithmically on the multiplicative difference between these two vectors. Clearly, instead of increasing \( t \) we can decrease \( t \) similarly. Hence, we can decrease \( t \) such that we get the central path point \( \vec{x}_\text{center} \) for the cost function \( 2^{-\tilde{O}(L)} \vec{c}_\text{modified} \). Since \( 2^{-\tilde{O}(L)} \) is close enough to zero, it can be shown that \( \delta_t \) is small also for the cost function \( 2^{-\tilde{O}(L)} \vec{c} \). Then, we could use Theorem 27 to increase \( t \) and obtain the central path point for \( t = 2^{-\tilde{O}(L)} \).

Then, we can use centeringInexact to make \( \delta_t \) becomes and hence \( \vec{c}^T \vec{x}_t \) close to \( \vec{c}^T \vec{x} \). By a standard duality gap theorem,\(^{17}\) we know that the duality gap of \( \vec{x}_t \) is less than \( \|\vec{w}\|_1/t \) and in this case it is less than \( 2^{-\tilde{O}(L)} \) because \( \|\vec{w}\|_1 \leq 2 \text{ rank } (A) \). Now, we can use the conclusion of the Lemma 40 to find the active constraints of an optimal solution of the original linear program or prove that it is infeasible or unbounded.

During the algorithm, we only called the function centeringInexact \( \tilde{O}(L) \) times and hence the algorithm only executes \( \tilde{O}(L) \) linear system solves. In Section C, we show that these linear systems do not need to be solved exactly and that inexact linear algebra suffices. Using this observation and letting using [21] as the solve routine yields the total running time of
\[
\tilde{O} \left( \sqrt{\text{rank}(A)} \left( \text{nnz}(A) + (\text{rank}(A))^\omega \right) L \right).
\]

\( ^{17} \)See [14] or [22] for a more detailed treatment of this fact in a more general regime.

In Section B, we show that the projection problem in centeringInexact can be computed in \( \tilde{O}(1) \) depth and \( \tilde{O}(m) \) work and other operations are standard parallelizable linear algebra operations. Therefore, we achieve the first \( \tilde{O}(\sqrt{\text{rank}(A)} L) \) depth polynomial work method for solving linear programs.

**Theorem 29.** There is an \( \tilde{O}(\sqrt{\text{rank}(A)} L) \) depth polynomial work algorithm to solve linear program of the form
\[
\min_{\vec{x} \in \mathbb{R}^n} \vec{c}^T \vec{x} \quad : \quad A \vec{x} \geq \vec{b}
\]
where \( L \) denote the bit complexity of the linear program.
7.3 Accelerating the Solver

In this section, we show that how we can apply acceleration methods for decreasing the iterations of interior point techniques can be applied to our algorithm to yield a faster method. In particular we show how to adapt techniques of Vaidya [35] for using fast matrix multiplication to obtain a faster running time. Our goal here is to provide a simple exposition of how the iteration costs of our method can be decrease. We make no attempt to explore the running time of our algorithm in all regimes and we note that since our algorithm only needs to solve linear systems in scalings of the original constraint matrix there may be techniques to improve our algorithm further in specific regimes by exploiting structure in $A$.

To accelerate our path following method, we note that we solve systems of two forms: we solve systems in $A^T S^{-1} W S^{-1} A$ to update $\bar{x}$ and we solve systems in $A^T S^{-1} W^\alpha S^{-1} A$ to update $\bar{w}$. Since we have proven in Lemma 19 that two system are spectrally similar, we only need to know how to solve system of the form $A^T S^{-1} W S^{-1} A$ and then we can use preconditioning to solve either system. Furthermore, we note similarly to Vaidya [35] that the $S$ and $W$ matrices do not change too much from iteration to iteration and therefore a sequence of the necessary linear system can be solved faster then considering them individually. Below we state and Appendix E we prove a slight improvement of a result in [35] formally analyzing one way of solving these systems faster.

**Theorem 30.** Let $\bar{d}^{(i)} \in \mathbb{R}^m_{>0}$ be a sequence of $r$ positive vectors. Suppose that the number of times that $d^{(i)}_j \neq d^{(i+1)}_j$ for any $i \in [r]$ and $j \in [m]$ is bounded by $Cr^2$ for some $C \geq 1$. Then if we are given the $\bar{d}^{(i)}$ in a sequence, in each iteration $i$ we can compute $(A^T D_i A)^{-1} \bar{x}_i$ for $D_i = \text{diag}(\bar{d}_i)$ and arbitrary $\bar{x}_i \in \mathbb{R}^n$ such that the average cost per iteration is

$$
\tilde{O}
\left(\frac{mn^\omega - 1}{r} + n^2 + C\omega r^{\omega + 2} + Cn^\omega \right)
$$

where $\omega < 2.3729[39]$ is the matrix multiplication constant.

Using Theorem 30 we simply need to estimate how much the diagonal entries $S^{-1} W S^{-1}$ to obtain a faster linear program solver. We prove the following.

**Theorem 31.** For any $\frac{n}{m} \leq \beta \leq 1$ and $r > 1$, there is an

$$
\tilde{O}
\left(\sqrt{m} \beta \left( n^2 + \frac{mn^\omega - 1}{r} + \beta^{-\omega r^{2\omega}} + \beta^{-(\omega - 1)n^\omega} \right) \right)
$$

(7.4)

time algorithm for solving linear programming problems of the form

$$
\min c^T \bar{x} \text{ given } A \bar{x} \geq \bar{b}
$$

where $A \in \mathbb{R}^{m \times n}$. In particular, we have a LP algorithm with running time $\tilde{O} \left( n^{5/2-3(\omega - 2)m^3(\omega - 2)} L \right)$.

**Proof.** Instead of using $\beta = \frac{n}{m}$ in the weight function we let $\beta \in [\frac{n}{m}, 1]$ be arbitrary as in the theorem statement. Looking at the analysis in Section 6 we see that this yields a weight function with $c_1 = O(\beta m)$, $c_\gamma = O(1)$ and $c_r = \tilde{O}(1)$. Consequently, it takes $\tilde{O}(\sqrt{3mL})$ iterations to solve the linear program.

We separate the sequence of the linear systems involved into groups of size $r$. We want to use the previous theorem to see the averaged cost to approximate $(A^T D_j A)^{-1} \bar{x}$ for each group of
operations. Therefore, we need to estimate the change of the diagonal entries $S^{-1}WS^{-1}$. We can simply consider the change of $S$ because the effect of $W$ is similar. Lemma 5 shows that

$$\left\| \log (\vec{s}_j) - \log (\vec{s}_{j+1}) \right\|_{w_j} = O(1).$$

Since we have added $\beta$ in the weight function, we have $\bar{w}_i \geq \beta$. Therefore, we have

$$\left\| \log (\vec{s}_j) - \log (\vec{s}_{j+1}) \right\|_2 = O(\beta^{-1/2}).$$

Therefore, in a period of $r$ operations, at most $O(\beta^{-1}r^2)$ coordinates can change multiplicatively by a constant factor. Similarly, we can use inequality (5.9) to analyze the change of $W$.

Therefore, we can maintain a vector $\vec{d}$ such that $D$ is spectrally similar to $S^{-1}WS^{-1}$ while only changing $\vec{d}$ a total of $O(\beta^{-1}r^2)$ over a sequence of $r$ operations. Using Theorem 30 and using ADA as pre-conditioner for the necessary linear system solves we see that we can solve the linear system we need to solve with average cost

$$\tilde{O}\left(n^2 + \frac{mn^{\omega-1}}{r} + \beta^{-\omega} r^{2\omega} + \beta^{-(\omega-1)}nr^\omega\right).$$

Using that the total number of iterations is $\tilde{O}(\sqrt{\beta m}L)$ then yields (7.4).

For the last result, we put $\beta = \frac{n}{m}$ and $r = n^{\frac{2\omega-1}{\omega+1}} m^{\frac{1}{\omega+1}}$ into the final running time, we get

$$\tilde{O}\left(\sqrt{n}\left(n^2 + n^{\omega-\frac{2\omega}{\omega+1}} m^{\frac{2\omega}{\omega+1}}\right) L\right)$$

when $m < n^{\frac{2\omega-1}{\omega+1}}$. Balancing it with the running time $\tilde{O}(\sqrt{n}(mnz(A) + n^\omega) L)$ in Theorem 28, we get

$$\tilde{O}\left(\sqrt{n}\left(n^2 + n^{\omega} \min\left((m^3/n^4)^{\omega/(2\omega+1)}, 1\right)\right) L\right)$$

The last result follows from this.

For $\omega = 2.3729$, we have $\tilde{O}\left(m^{1.1187}n^{1.3813}L\right)$ which is strictly better than the previously fastest linear programming algorithm for dense matrix $\tilde{O}\left(m^{1.5}nL\right)$ in [35]. To appreciate this running time, we note that it takes time $O(mn^{1.3729})$ to exactly solve a dense linear system of size $m \times n$, which is a necessary step to compute the solution exactly. However, we want to emphasis again our algorithm is amenable to acceleration techniques and linear solvers with low amortized cost can be designed for specific class of linear programs, such as linear programs involving sparse matrix. Therefore, running times better than $\tilde{O}\left(m^{1.1187}n^{1.3813}L\right)$ could be easily obtained for specific situations.

8 Acknowledgments

We thank Yan Kit Chim, Andreea Gane, Jonathan A. Kelner, Lap Chi Lau, Aleksander Mądry, Cameron Musco, Christopher Musco, Lorenzo Orecchia, Ka Yu Tam and Nisheeth Vishnoi for many helpful conversations. This work was partially supported by NSF awards 0843915 and 1111109, NSF Graduate Research Fellowship (grant no. 1122374) and Hong Kong RGC grant 2150701. Finally, we thank the referees for extraordinary efforts and many helpful suggestions.
References

[1] Kurt M. Anstreicher. Volumetric path following algorithms for linear programming. Math. Program., 76:245–263, 1996.

[2] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In Proceedings of the 45th annual ACM symposium on Symposium on theory of computing, pages 81–90. ACM, 2013.

[3] Samuel I. Daitch and Daniel 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. ACM, 2008.

[4] George B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. New York, 1951.

[5] Andrew V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. J. ACM, 45(5):783–797, 1998.

[6] Clovis C. Gonzaga. Path-following methods for linear programming. SIAM review, 34(2):167–224, 1992.

[7] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In Proceedings of the sixteenth annual ACM symposium on Theory of computing, pages 302–311. ACM, 1984.

[8] Jonathan A. Kelner, Lorenzo Orecchia, Yin Tat Lee, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. arXiv preprint arXiv:1304.2338, 2013.

[9] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A Simple, Combinatorial Algorithm for Solving SDD Systems in Nearly-Linear Time. January 2013.

[10] Leonid G. Khachiyan. Polynomial algorithms in linear programming. USSR Computational Mathematics and Mathematical Physics, 20(1):53–72, 1980.

[11] Leonid G. Khachiyan. Rounding of polytopes in the real number model of computation. Mathematics of Operations Research, 21(2):307–320, 1996.

[12] Adam R. Klivans and Daniel Spielman. Randomness efficient identity testing of multivariate polynomials. In Proceedings of the thirty-third annual ACM symposium on Theory of computing, pages 216–223. ACM, 2001.

[13] Ioannis Koutis, Gary L. Miller, and Richard Peng. A nearly-\(m \log n\) time solver for sdd linear systems. In Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on, pages 590–598, oct. 2011.

[14] Yin Tat Lee and Aaron Sidford. An \(\tilde{O}(m \sqrt{n})\) algorithm for the minimum cost flow problem. arXiv preprint arXiv:1312.6713 (in submission to FOCS 2014), 2013.

[15] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In The 54th Annual Symposium on Foundations of Computer Science (FOCS), 2013.
[16] Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. 2012.

[17] László Lovász and Santosh Vempala. Simulated annealing in convex bodies and an $o^*(n^4)$ volume algorithm. *J. Comput. Syst. Sci.*, 72(2):392–417, 2006.

[18] Aleksander Madry. Navigating central path with electrical flows: from flows to matchings, and back. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*, 2013.

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

[20] Shinji Mizuno, Michael J Todd, and Yinyu Ye. On adaptive-step primal-dual interior-point algorithms for linear programming. *Mathematics of Operations research*, 18(4):964–981, 1993.

[21] Jelani Nelson and Huy L Nguyễn. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. *arXiv preprint arXiv:1211.1002*, 2012.

[22] Yu Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume I. 2003.

[23] Yu. Nesterov. Rounding of convex sets and efficient gradient methods for linear programming. *Optimization Methods Software*, 23(1):109–128, February 2008.

[24] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[25] Yu E Nesterov and Michael J Todd. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations research*, 22(1):1–42, 1997.

[26] Yurii Nesterov and Arkadii Semenovich Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. Society for Industrial and Applied Mathematics, 1994.

[27] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications, 1998.

[28] James Renegar. A polynomial-time algorithm, based on newton’s method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, 1988.

[29] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *STOC*, pages 563–568, 2008.

[30] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2004.

[31] Gilbert Strang. Inverse problems and derivatives of determinants. *Archive for Rational Mechanics and Analysis*, 114(3):255–265, 1991.

[32] SP Tarasov, LG Khachiyan, and I Erelikh. The method of inscribed ellipsoids. In *Soviet Mathematics Doklady*, volume 37, pages 226–230, 1988.
A Technical Tools

In this section, we provide and prove various mathematical facts that we use throughout the paper.

A.1 Matrix Properties

First, we prove various properties regarding projection matrices that we use throughout the paper.

Lemma 32 (Projection Matrices). Let $P \in \mathbb{R}^{n \times n}$ be an arbitrary projection matrix and let $\Sigma = \text{diag}(P)$. For all $i, j \in [n]$ and $\vec{x} \in \mathbb{R}^n$ we have the following

1. $\Sigma_{ii} = \sum_{j \in [n]} P_{ij}^{(2)}$,
2. $0 \preceq P^{(2)} \preceq \Sigma \preceq I$,
3. $P_{ij}^{(2)} \leq \Sigma_{ii} \Sigma_{jj}$,
4. $|\vec{x}_i^T P_{ij}^{(2)} \vec{x}| \leq \Sigma_{ii} \|\vec{x}\|_\Sigma$.

Proof. To prove (1), we simply note that by definition of a projection matrix $P = PP$ and therefore

$$\Sigma_{ii} = P_{ii} = \vec{x}_i^T P \vec{x}_i = \vec{x}_i^T PP \vec{x}_i = \sum_{j \in [n]} P_{ij}^2 = \sum_{j \in [n]} P_{ij}^{(2)}$$

To prove (2), we observe that since $P$ is a projection matrix, all its eigenvectors are either 0 or 1. Therefore, $\Sigma \preceq I$ and by (1) $\Sigma - P^{(2)}$ is diagonally dominant. Consequently, $\Sigma - P^{(2)} \succeq 0$. 


47
Rearranging terms and using the well known fact that the Shur product of two positive semi-definite matrices is positive semi-definite yields (2).

To prove (3), we use \( P = PP \), Cauchy-Schwarz, and (1) to derive

\[
P_{ij} = \sum_{k \in [n]} P_{ik} P_{kj} \leq \left( \sum_{k \in [n]} P_{ik}^2 \right)^{1/2} \left( \sum_{k \in [n]} P_{kj}^2 \right)^{1/2} = \sqrt{\Sigma_{ii} \Sigma_{jj}}.
\]

Squaring then yields (3).

To prove (4), we note that by the definition of \( P^{(2)} \) and Cauchy-Schwarz, we have

\[
\left| \bar{\mathbf{1}}_i^T P^{(2)} \bar{\mathbf{x}} \right| = \left| \sum_{j \in [n]} \bar{\mathbf{1}}_i^T P^{(2)}_{ij} \bar{x}_j \right| \leq \left( \sum_{j \in [n]} \Sigma_{jj} \bar{x}_j^2 \right)^{1/2} \cdot \sum_{j \in [n]} \frac{P^{(4)}_{ij}}{\Sigma_{jj}} \quad (A.1)
\]

Now, by (1) and (3), we know that

\[
\sum_{j \in [n]} \frac{P^{4}_{ij}}{\Sigma_{jj}} \leq \sum_{j \in [n]} \frac{P^{2}_{ij} \Sigma_{ij} \Sigma_{jj}}{\Sigma_{jj}} = \Sigma_{ii} \sum_{j \in [n]} P^{2}_{ij} = \Sigma_{ii}^2 \quad (A.2)
\]

Since \( \| \bar{\mathbf{x}} \|_\Sigma \stackrel{\text{def}}{=} \sqrt{\sum_{j \in [n]} \Sigma_{jj} \bar{x}_j^2} \), combining (A.1) and (A.2) yields \( \left| \bar{\mathbf{1}}_i^T P^{(2)} \bar{\mathbf{x}} \right| \leq \Sigma_{ii} \| \bar{\mathbf{x}} \|_\Sigma \) as desired.

\[\square\]

### A.2 Taylor Expansions and Multiplicative Approximations

Throughout this paper we use \( \log(\bar{\mathbf{a}}) - \log(\bar{\mathbf{b}}) \) as a convenient way of working with \( \mathbf{B}^{-1}(\bar{\mathbf{a}} - \bar{\mathbf{b}}) \) or \( \mathbf{A}^{-1}(\bar{\mathbf{b}} - \bar{\mathbf{a}}) \). In this section we make this connection rigorous by providing several helper lemmas used throughout the paper.

**Lemma 33** (Log Notation). Suppose \( \| \log(\bar{\mathbf{a}}) - \log(\bar{\mathbf{b}}) \|_\infty = \epsilon \leq 1/2 \) then

\[
\| \mathbf{B}^{-1}(\bar{\mathbf{a}} - \bar{\mathbf{b}}) \|_\infty \leq \epsilon + \epsilon^2.
\]

If \( \| \mathbf{B}^{-1}(\bar{\mathbf{a}} - \bar{\mathbf{b}}) \|_\infty = \epsilon \leq 1/2, \) then

\[
\| \log(\bar{\mathbf{a}}) - \log(\bar{\mathbf{b}}) \|_\infty \leq \epsilon + \epsilon^2.
\]

**Proof.** Using the Taylor expansion of \( e^x \) and \( \log(1 + x) \), we get the following two inequalities which prove the claim

\[
1 + x \leq e^x \leq 1 + x + x^2 \text{ for } |x| \leq \frac{1}{2},
\]

\[
x - x^2 \leq \log(1 + x) \leq x \text{ for } |x| \leq \frac{1}{2}.
\]

\[\square\]
A.3 Matrix Calculus

Here, we derive various matrix calculus formulas used in Section 6. These are now somewhat standard and also discussed in [37, 1] but we derive them here for completeness. In this section, we define

\[ R_A(\bar{w})_{ij} \overset{\text{def}}{=} \bar{a}_i^T (A^T W A)^{-1} \bar{a}_j. \]

We start by computing the derivative of the volumetric barrier function, \( f(\bar{w}) \overset{\text{def}}{=} \log \det(A^T W A) \).

**Lemma 34** (Derivative of Volumetric Barrier). For \( A \in \mathbb{R}^{n \times m} \), let \( f : \mathbb{R}_{>0}^m \rightarrow \mathbb{R} \) be given by \( f(\bar{w}) \overset{\text{def}}{=} \log \det(A^T W A) \). Then the following holds

\[ \forall \bar{w} \in \mathbb{R}_{>0}^m \quad : \quad \nabla f(\bar{w}) = \text{diag}(R_A(\bar{w})) \overset{\text{def}}{=} \Sigma_A(\bar{w}) W^{-1} \mathbb{1}. \]

**Proof.** For all \( i \in \{m\} \) and \( \bar{w} \in \mathbb{R}^m \), we know that

\[ \frac{\partial}{\partial \bar{w}_i} f(\bar{w}) = \lim_{\alpha \to 0} \frac{1}{\alpha} \left[ f(\bar{w} + \alpha \mathbb{1}_i) - f(\bar{w}) \right] = \lim_{\alpha \to 0} \frac{1}{\alpha} \left[ \log \det(A^T W A + \alpha \bar{a}_i \bar{a}_i^T) - \log \det(A^T W A) \right]. \]

Applying the matrix determinant lemma then yields that

\[ \frac{\partial}{\partial \bar{w}_i} f(\bar{w}) = \lim_{\alpha \to 0} \frac{1}{\alpha} \left[ \log \left( \det(A^T W A) \cdot (1 + \alpha \bar{a}_i (A^T W A)^{-1} \bar{a}_i) \right) - \log \left( \det(A^T W A) \right) \right]. \]

Therefore,

\[ \frac{\partial}{\partial \bar{w}_i} f(\bar{w}) = \lim_{\alpha \to 0} \frac{\log(1 + \alpha R(\bar{w})_{ii})}{\alpha} = R(\bar{w})_{ii}. \]

\( \square \)

Next we bound the rate of change of entries of the resistance matrix.

**Lemma 35** (Derivative of Effective Resistance). For all \( A \in \mathbb{R}^{m \times n}, \bar{w} \in \mathbb{R}_{>0}^m \), and \( i, j, k \in \{m\} \) we have

\[ \frac{\partial}{\partial \bar{w}_k} [R_A(\bar{w})]_{ij} = -R_A(\bar{w})_{ik} R_A(\bar{w})_{kj} \]

where \( \text{diag}(R_A(\bar{w})) \overset{\text{def}}{=} \Sigma_A(\bar{w}) W^{-1} \mathbb{1} \).

**Proof.** By definition, we have that

\[ \frac{\partial}{\partial \bar{w}_k} R_A(\bar{w})_{ij} = \lim_{\alpha \to 0} \frac{1}{\alpha} \left[ R(\bar{w} + \alpha \mathbb{1}_k)_{ij} - R(\bar{w})_{ij} \right] \quad (A.3) \]

and

\[ R(\bar{w} + \alpha \mathbb{1}_k)_{ij} = \mathbb{1}_i^T A (A^T W A + \alpha A^T \mathbb{1}_k \mathbb{1}_k^T A)^{-1} A^T \mathbb{1}_j. \quad (A.4) \]

Furthermore, by applying the Sherman-Morrison formula, we know that

\[ (A^T W A + \alpha A^T \mathbb{1}_k \mathbb{1}_k^T A)^{-1} = (A^T W A)^+ - \frac{\alpha (A^T W A)^{-1} A^T \mathbb{1}_k \mathbb{1}_k^T A (A^T W A)^{-1}}{1 + \alpha \mathbb{1}_k^T A (A^T W A)^{-1} A^T \mathbb{1}_k}. \quad (A.5) \]

Combining (A.3), (A.4), and (A.5) yields the result. \( \square \)
Finally, we use this to derive the Jacobian of leverage scores.

**Lemma 36 (Derivative of Leverage Scores).** For all $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\bar{\mathbf{w}} \in \mathbb{R}^n_{\geq 0}$ we have the following

$$
\mathbf{J}_{\bar{\mathbf{w}}} (\bar{\sigma}_{\mathbf{A}}(\bar{\mathbf{w}})) = \mathbf{\Lambda}_{\mathbf{A}}(\bar{\mathbf{w}}) \mathbf{W}^{-1}.
$$

**Proof.** Since by definition $\bar{\sigma}_{\mathbf{A}}(\bar{\mathbf{w}})_i = \bar{w}_i \mathbf{R}_{\mathbf{A}}(\bar{\mathbf{w}})_{ii}$ by the previous lemma, we have that

$$
\frac{\partial}{\partial \bar{w}_j} \bar{\sigma}_{\mathbf{A}}(\bar{\mathbf{w}})_i = \bar{w}_j \mathbf{R}(\bar{\mathbf{w}})_{ij} - \bar{w}_i \mathbf{R}(\bar{\mathbf{w}})_{ij}^{(2)}.
$$

Writing this in matrix form and recalling the definition of the Jacobian then yields

$$
\mathbf{J}_{\bar{\mathbf{w}}} (\bar{\sigma}_{\mathbf{A}}(\bar{\mathbf{w}})) = \text{diag}(\mathbf{R}_{\mathbf{A}}(\bar{\mathbf{w}})) - \mathbf{W} \mathbf{R}_{\mathbf{A}}(\bar{\mathbf{w}})^{(2)}.
$$

Right multiplying by $\mathbf{I} = \mathbf{W} \mathbf{W}^{-1}$ and recalling the definition of $\mathbf{\Lambda}_{\mathbf{A}}$ then yields the result.

**B Projecting Onto Ball Intersect Box**

In the algorithm **centeringInexact**, we need to compute

$$
\arg \min_{\bar{\mathbf{u}} \in \mathcal{U}} \langle \bar{\mathbf{a}}, \bar{\mathbf{u}} \rangle
$$

where $\mathcal{U} = \{ \bar{x} \in \mathbb{R}^m \mid \| \bar{x} \|_W \leq b \text{ and } \| \bar{x} \|_\infty \leq c \}$ for some $\bar{\mathbf{w}} \geq 0$, i.e. we need to project $\bar{\mathbf{a}}$ onto the intersection of the ball, $\{ \bar{x} \in \mathbb{R}^m \mid \| \bar{x} \|_W \leq b \}$, and the box $\{ \bar{x} \in \mathbb{R}^m \mid \| \bar{x} \|_\infty \leq c \}$. In this section we show how this can be computed in nearly linear time and in particular it can be computed in parallel in depth $\tilde{O}(1)$ and work $\tilde{O}(m)$.

Note that by rescaling we can rewrite (B.1) as

$$
\arg \max_{\| \bar{x} \|_2^2 \leq 1, -l_i \leq x_i \leq l_i} \langle \bar{\mathbf{a}}, \bar{x} \rangle
$$

for some $l_i$. Let us consider a simple algorithm which first ignore the box constraint and find the best vector $\bar{\mathbf{a}}$. If $\bar{\mathbf{a}}$ does not violate any box constraint, then it is the solution. Otherwise, we pick a most violated constraint $i$, i.e. the coordinate with highest $|a_i| / l_i$. Then, we threshold this coordinates and repeat the procedure on the remaining coordinate.

**Lemma 37.** The algorithm **projectOntoBallBox** outputs a solution of the problem (B.2).
Proof. We claim that for all \( k \leq i \) where \( i \) is the last step in the algorithm, we have
\[
\max_{\vec{x} \in \Omega} \langle \vec{a}, \vec{x} \rangle = \max_{\vec{x} \in \Omega_k} \langle \vec{a}, \vec{x} \rangle
\]
where \( \Omega = \{ x : \| \vec{x} \|_2 \leq 1, -l_i \leq x_i \leq l_i \} \) and \( \Omega_k = \Omega \cap \{ x : |x_i| = l_i \text{ for } i \in \{1, 2, \ldots, k\} \} \). Since \( \vec{x} \) is feasible at the last step, we have
\[
\vec{x}_{\text{last}} = \arg \max_{\vec{x} \in \Omega_k} \langle \vec{a}, \vec{x} \rangle = \arg \max_{\vec{x} \in \Omega} \langle \vec{a}, \vec{x} \rangle.
\]
Therefore, the correctness of the algorithm follows from the claim.

Now, we prove the claim by induction. The base case is trivial because \( \Omega = \Omega_0 \). Now proceed by contradiction and suppose that
\[
\max_{\vec{x} \in \Omega_k} \langle \vec{a}, \vec{x} \rangle > \max_{\vec{x} \in \Omega_{k+1}} \langle \vec{a}, \vec{x} \rangle. \tag{B.3}
\]
Let \( \vec{y} = \arg \max_{\vec{x} \in \Omega_k} \langle \vec{a}, \vec{x} \rangle \). If for all \( j \) \( \leq k \), we have \( |y_j| \leq l_j \). Then, the \( \vec{x} \) found in the \( (k+1) \)th iteration is exactly \( \vec{y} \) and it is feasible and hence the algorithm outputs \( \vec{y} \). Otherwise, there is \( j \) such that \( |y_j| = l_j \). Since \( \vec{y} \notin \Omega_{k+1} \), we have \( |y_{k+1}| < l_{k+1} \) and hence \( j \geq k + 1 \).

Consider
\[
\vec{z}(t) = \vec{y} + \frac{\text{sign} (y_{k+1}) t}{|y_{k+1}| + \epsilon} \vec{z}_{k+1} - \frac{\text{sign} (y_j) t}{l_j} \vec{z}_j
\]
where \( \epsilon \) is a very small positive number. Note that \( \frac{d}{dt} \| \vec{z}(t) \|_2 \bigg|_{t=0} = 2 \frac{|y_{k+1}|}{|y_{k+1}| + \epsilon} - 2 < 0 \) and hence \( \| \vec{z}(t) \|_2 \leq 1 \) for \( t > 0 \) but close to 0. Also, we have
\[
\frac{d}{dt} \langle \vec{a}, \vec{z} \rangle = \frac{|a_{k+1}|}{|y_{k+1}| + \epsilon} - \frac{|a_j|}{l_j}.
\]
Take \( \epsilon = l_{k+1} - |y_{k+1}| \), then we have
\[
\frac{d}{dt} \langle \vec{a}, \vec{z} \rangle = \frac{|a_{k+1}|}{l_{k+1}} - \frac{|a_j|}{l_j} > 0
\]
because \( j \geq k + 1 \) and \( |a_i| / l_i \) is in descending order. Therefore, \( \vec{z}(t) \) is a feasible and better solution for small positive \( t \). Hence, it proves \( \vec{y} \) is not the optimal solution of \( \max_{\vec{x} \in \Omega_k} \langle \vec{a}, \vec{x} \rangle \) that contradicts to the definition of \( \vec{y} \).

Hence, \( \max_{\vec{x} \in \Omega} \langle \vec{a}, \vec{x} \rangle = \max_{\vec{x} \in \Omega_k} \langle \vec{a}, \vec{x} \rangle \) and the algorithm outputs an optimal solution. \( \square \)

| \( \vec{x} = \text{projectOntoBallBoxParallel}(\vec{a}) \) |
|---|
| 1. Set \( \vec{a} = \vec{a} / \| \vec{a} \|_2 \). |
| 2. Sort the coordinate such that \( |a_i| / l_i \) is in descending order. |
| 3. Precompute \( \sum_{k=0}^{i-1} l_k^2 \) and \( \sum_{k=0}^{i-1} a_k^2 \) for all \( i \). |
| 4. Find the first \( i \) such that \( \frac{1 - \sum_{k=0}^{i-1} a_k^2}{1 - \sum_{k=0}^{i} a_k^2} \leq \frac{t_{k+1}}{a_{k+1}} \). |
| 5. Output \( \vec{x} = \begin{cases} 
   \text{sign}(a_j) l_j & \text{if } j \in \{1, 2, \ldots, i\} \\
   \sqrt{\frac{1 - \sum_{k=0}^{i-1} a_k^2}{1 - \sum_{k=0}^{i} a_k^2} a_j} & \text{otherwise}
\end{cases} \). |

51
The algorithm \texttt{projectOntoBallBoxParallel} is a parallel and more efficient version of \texttt{projectOntoBallBox}. All other operations in our algorithm are standard linear algebra and hence the following theorem shows that our linear programming solver is indeed parallelizable.

\textbf{Lemma 38.} The algorithm \texttt{projectOntoBallBoxParallel} outputs an solution of the optimization problem (B.2) in depth $O(1)$ and work $O(m)$.

\textit{Proof.} Note that in the algorithm \texttt{projectOntoBallBox}, the value
\[\frac{1 - \sum_{k=0}^{i} l_k^2}{1 - \sum_{k=0}^{i} a_k^2}\]
is increasing through the algorithm. To see this, note that in step 3b, if $\bar{x}$ is not feasible, that means there is $j$ such that
\[\frac{1 - \sum_{k=0}^{i} l_k^2}{1 - \sum_{k=0}^{i} a_k^2} > \frac{l_j^2}{a_j^2}.
\]
Since $a_i/l_i$ is in descending order, $j = i + 1$. Therefore, we have
\[\frac{1 - \sum_{k=0}^{i} l_k^2}{1 - \sum_{k=0}^{i} a_k^2} > \frac{l_{i+1}^2}{a_{i+1}^2}.
\]
Hence, we have
\[\frac{1 - \sum_{k=0}^{i+1} l_k^2}{1 - \sum_{k=0}^{i+1} a_k^2} > \frac{1 - \sum_{k=0}^{i} l_k^2}{1 - \sum_{k=0}^{i} a_k^2}.
\]
Using this fact, it is easy to see the algorithm \texttt{projectOntoBallBoxParallel} and the algorithm \texttt{projectOntoBallBox} outputs the same vector. Obviously, all steps can be computed in depth $O(1)$ and work $O(m)$.

\hfill \blacksquare

\section{Inexact Linear Algebra}

Throughout much of our analysis of weighted path following we assumed that linear systems in $A$ could be solved exactly. In this section we relax this assumption and discuss the effect of using inexact linear algebra in our linear programming algorithms. We show that rather than computing $(A^TDA)^{-1}x$ precisely for positive diagonal matrix $D$ it suffices to solve these systems approximately.

Throughout this section we assume that for any matrix $A \in \mathbb{R}^{n \times m}$ and vector $\vec{b} \in \mathbb{R}^m$ there is an algorithm $\text{solve}(A, \vec{b})$ which outputs an vector $\vec{x}$ such that
\[\|\vec{x} - A^+\vec{b}\|_{A^TA} \leq \epsilon \|A^+\vec{b}\|_{A^TA}.
\]
(C.1)

Since $A$ is full rank, we can write $\vec{c} = A^T\vec{d}$ for some $\vec{d}$. From equation (4.2), the Newton step is
\[\vec{h}_t(\vec{x}, \vec{w}) = (A^T S^{-1} WS^{-1} A)^{-1} A^T S^{-1} \sqrt{W} \left(t \frac{\vec{s}d}{\sqrt{\vec{w}}} - \sqrt{\vec{w}}\right)\]
\[= \left(\sqrt{W} S^{-1} A\right)^+ \left(t \frac{\vec{s}d}{\sqrt{\vec{w}}} - \sqrt{\vec{w}}\right).
\]
Suppose that we compute \( \tilde{h}_t \) by the algorithm \texttt{solve} above, then we have

\[
\left\| \texttt{solve} \left( \sqrt{\mathbf{W}\mathbf{S}^{-1}\mathbf{A}}, t \frac{\tilde{\mathbf{s}}}{\sqrt{\tilde{\mathbf{w}}}} - \sqrt{\mathbf{w}} \right) - \tilde{h}_t \right\|_{\mathbf{A}^T\mathbf{S}^{-1}\mathbf{W}\mathbf{S}^{-1}\mathbf{A}} \leq \epsilon \left\| \tilde{h}_t \right\|_{\mathbf{A}^T\mathbf{S}^{-1}\mathbf{W}\mathbf{S}^{-1}\mathbf{A}} = \epsilon \delta_t (\tilde{x}, \tilde{w}).
\]

Hence, the outcome of \texttt{solve} differs from the Newton step \( \tilde{h}_t \) by a relative small amount in \( \| \cdot \|_{\mathbf{A}^T\mathbf{S}^{-1}\mathbf{W}\mathbf{S}^{-1}\mathbf{A}} \). Hence, it suffices to prove that \( \delta_t \) is stable under this small amount in \( \| \cdot \|_{\mathbf{A}^T\mathbf{S}^{-1}\mathbf{W}\mathbf{S}^{-1}\mathbf{A}} \) and hence is the algorithm \texttt{solve} will only increase \( \delta \) by a little compared with using exact linear algebra.

**Lemma 39.** Let \( \gamma \overset{\text{def}}{=} \gamma(\tilde{x}, \tilde{w}) \) and \( \tilde{x}^{(\text{new})} = \tilde{x} + \Delta \). Let \( \eta = \left\| \Delta \right\|_{\mathbf{A}^T\mathbf{S}^{-1}\mathbf{W}\mathbf{S}^{-1}\mathbf{A}} \leq \frac{1}{8\gamma} \). Then, we have

\[
\delta_t (\tilde{x}^{(\text{new})}, \tilde{w}) \leq (1 - \gamma\eta)^{-1} (\delta_t (\tilde{x}, \tilde{w}) + \eta).
\]

**Proof.** By the same proof in Lemma 5, we have that

\[
\left\| \mathbf{S}^{-1}(\tilde{s}^{(\text{new})} - \tilde{s}) \right\|_\infty \leq \gamma\eta.
\]

Therefore, we have

\[
\delta_t (\tilde{x}^{(\text{new})}, \tilde{w}) = \left\| t\tilde{c} - \mathbf{A}^T\mathbf{S}^{-1}_{(\text{new})} \tilde{w} \right\|_{(\mathbf{A}^T\mathbf{S}^{-1}_{(\text{new})} \mathbf{W}\mathbf{S}^{-1}_{(\text{new})}\mathbf{A})^{-1}} \\
\leq (1 + \gamma\eta) \left\| t\tilde{c} - \mathbf{A}^T\mathbf{S}^{-1}_{(\text{new})} \tilde{w} \right\|_{(\mathbf{A}^T\mathbf{S}^{-1}_{(\text{new})} \mathbf{W}\mathbf{S}^{-1}_{(\text{new})}\mathbf{A})^{-1}} \\
\leq (1 + \gamma\eta) \left\| t\tilde{c} - \mathbf{A}^T\mathbf{S}^{-1} \tilde{w} \right\|_{(\mathbf{A}^T\mathbf{S}^{-1} \mathbf{W}\mathbf{S}^{-1}\mathbf{A})^{-1}} + \left\| \mathbf{A}^T \left( \frac{\tilde{w}}{\tilde{s}} - \frac{\tilde{w}}{\tilde{s}^{(\text{new})}} \right) \right\|_{(\mathbf{A}^T\mathbf{S}^{-1}\mathbf{W}\mathbf{S}^{-1}\mathbf{A})^{-1}} \\
= (1 + \gamma\eta) \delta_t (\tilde{x}, \tilde{w}) + \left\| \frac{\tilde{s}^{(\text{new})} - \tilde{s}}{\tilde{s}^{(\text{new})}} \right\|_{\mathbf{W}} \\
\leq (1 + \gamma\eta) \delta_t (\tilde{x}, \tilde{w}) + (1 - \gamma\eta)^{-1} \left\| \frac{\tilde{s}^{(\text{new})} - \tilde{s}}{\tilde{s}} \right\|_{\mathbf{W}}.
\]

By the same proof in Lemma 5, we have that

\[
\left\| \mathbf{S}^{-1}(\tilde{s}^{(\text{new})} - \tilde{s}) \right\|_{\mathbf{W}} \leq \eta.
\]

Thus, we have the result.

Therefore, as long as we choose \( \epsilon \) small enough, the algorithm \texttt{solve} gives an accurate enough \( \tilde{x}^{(\text{new})} \) for the centering step. Similarly, it is easy to see that it also gives accurate enough \( \tilde{w}^{(\text{new})} \) because the error of \( \tilde{w}^{(\text{new})} \) due to \texttt{solve} is small in \( \| \cdot \|_{\mathbf{W}} \) norm and the tracing 0 game can afford for this error.

At last, we need to check \texttt{solve} gives us a way to compute weight function. Since the weight function computation relies on the function \texttt{computeLeverageScores}, we only need to know if we can compute \( \tilde{t} \) in the \texttt{computeLeverageScores} with high enough accuracy. Now, we use the notation
is the `computeLeverageScores`. Without loss of generality, we can assume $X = I$. Let $\bar{t}^{(apx)}$ and $\tilde{p}^{(apx)}$ be the approximate $t$ and $p$ computed by the algorithm `Solve`. Then, we have

$$
\left\| \left( \bar{t}^{(j)} \right)^{(apx)} - (A^T A) + A^T \tilde{q}^{(j)} \right\|_{A^T A} = \left\| \left( \bar{t}^{(j)} \right)^{(apx)} - A^T \tilde{q}^{(j)} \right\|_{A^T A}
\leq \epsilon \left\| A^T \tilde{q}^{(j)} \right\|_{(A^T A)^{-1}}
= \epsilon \left\| \tilde{q}^{(j)} \right\|_2 \leq \epsilon \sqrt{\frac{n}{k}}.
$$

Hence, for any $i,j$, we have

$$
\left\| \bar{p}_i^{(j)} - (\bar{p}_i^{(apx)})^{(j)} \right\|_\infty \leq \left\| \tilde{p}_i^{(j)} - (\tilde{p}_i^{(apx)})^{(j)} \right\|_2
= \left\| A \left( \left( \bar{t}^{(j)} \right)^{(apx)} - \bar{t}^{(j)} \right) \right\|_2
\leq \epsilon \sqrt{\frac{n}{k}}.
$$

Therefore, we have

$$
\sqrt{\sum_{j=1}^{k} \left( \bar{p}_i^{(j)} \right)^2} - \sqrt{\sum_{j=1}^{k} \left( (\bar{p}_i^{(apx)})^{(j)} \right)^2} \leq \sqrt{\sum_{j=1}^{k} \left( \bar{p}_i^{(j)} - (\bar{p}_i^{(apx)})^{(j)} \right)^2}
\leq \epsilon \sqrt{nk}.
$$

Therefore, if $\epsilon \leq \sqrt{\frac{1}{m \text{polylog}(m)}}$, the error is small enough for `computeLeverageScores`.

### D Bit Complexity and Linear Program Reductions

In this section, we show how to reduce solving an arbitrary linear program to finding a low cost solution in a bounded linear program for which we have an explicit interior point. Throughout this section let $A \in \mathbb{R}^{m \times n}$, $\bar{b} \in \mathbb{R}^m$, $\bar{c} \in \mathbb{R}^n$, and consider the following general linear program

$$
\min_{\bar{x} \in \mathbb{R}^n : A\bar{x} \geq \bar{b}} \bar{c}^T \bar{x}. 
$$

We assume that the entries of $A$, $\bar{b}$, and $\bar{c}$ are integers and we let $OPT$ denote the optimal value of (D.1) and we let $L$ denote the bit complexity of (D.1) where

$$
L \overset{\text{def}}{=} \log(1 + d_{max}(A)) + \log(1 + \max\{\|\bar{c}\|_\infty, \|\bar{b}\|_\infty\})
$$

and $d_{max}(A)$ denotes the largest absolute value of the determinant of a square sub-matrix of $A$. Our goal is to efficiently transform (D.1) to a linear program of the same form

$$
\min_{\bar{x} \in \mathbb{R}^n : A\bar{x} \geq \bar{b}} \bar{c}^T \bar{x}. 
$$

Therefore, if $\epsilon \leq \sqrt{\frac{1}{m \text{polylog}(m)}}$, the error is small enough for `computeLeverageScores`. 

54
where \( \mathbf{A}' \in \mathbb{R}^{m' \times n'} \), \( \mathbf{b}' \in \mathbb{R}^{m'} \), and \( \mathbf{c}' \in \mathbb{R}^{n'} \) are integer, and \( \text{nnz}(\mathbf{A}') \), \( n' \), \( m' \), and the bit complexity of (D.2) denote, \( L' \), are comparable to \( \text{nnz}(\mathbf{A}) \), \( n \), \( m \), and \( L \). Furthermore, we require that (D.2) is bounded, has an explicit efficiently computable interior point, and that we can convert any low cost feasible solution to a solution of (D.1) in linear time.

While there are standard tools to perform reductions to ensure that (D.1) is bounded and has an explicit initial feasible point to or to ensure that the optimal integral solution can be easily computed explicitly, we need to particularly careful when using these reductions to ensure that \( \text{nnz}(\mathbf{A}) \), \( n \), and \( m \) are not increased significantly. As the running times of our path following techniques in Section (7) depend crucially on these parameters in this section we prove the following Lemma claiming that such an efficient reduction is possible.

**Lemma 40.** In \( O(\text{nnz}(\mathbf{A}) + n + m) \) time we can compute integer \( \mathbf{A}' \in \mathbb{R}^{m' \times n'} \), \( \mathbf{b}' \in \mathbb{R}^{m'} \), \( \mathbf{c}' \in \mathbb{R}^{n'} \), \( \bar{\mathbf{x}}' \in \mathbb{R}^{m'} \). Such that \( \text{nnz}(\mathbf{A}') = O(\text{nnz}(\mathbf{A}) + n + m) \), \( n' = O(n) \), \( m' = O(m) \), \( \mathbf{A}' \bar{\mathbf{x}}' \geq \mathbf{b}' \), and (D.2) is bounded and has bit complexity at most \( 12L_1 + 7 \log(20n) \). Furthermore, if we can find a feasible point in (D.2) such that the cost of that point is at most the \( \text{OPT} + 2^{-12(L+\log(20n))} \) where \( \text{OPT} \) is the value of (D.2) then we can either

1. Find the active constraints of a basic feasible optimal solution (D.1) using only one matrix vector multiplication by \( \mathbf{A} \); or

2. Prove that (D.1) is infeasible or unbounded.

We break this proof into two parts. First in Lemma 41 we show how to transform (D.1) so that the linear program is bounded and has an explicit feasible point. Then in Lemma 43 we follow the approach of [3] and show that we can perturb the cost of a linear program to make the optimal solution unique and thereby make it easy to compute an exact integral solution.

**Lemma 41.** Consider the following modified linear program

\[
\min \mathbf{c}'^T \bar{\mathbf{x}} + n2^{3L+4}z \quad \text{such that} \quad \mathbf{A}\bar{\mathbf{x}} + z\mathbf{1} \geq \mathbf{b}, 2^{L+1} \geq z \geq 0, 2^{L+1}\mathbf{1} \geq \bar{\mathbf{x}} \geq -2^{L+1}\mathbf{1} \quad \text{(D.3)}
\]

where \( \mathbf{A} \), \( \mathbf{b} \), and \( \mathbf{c} \) are as in (D.1) and \( L \) is the bit complexity of (D.1). (D.3) is bounded with an explicit interior point \( \bar{\mathbf{x}} = 0, z = 2^L + 1 \). Furthermore, (D.1) is bounded and feasible with an optimal solution \( \bar{\mathbf{x}} \) if and only if \( (\bar{\mathbf{x}}, 0) \) is an optimal solution of (D.3) with \( 2^L \geq x_i \geq -2^L \). (D.1) is unbounded if and only if there is a basic feasible solution, \( (\bar{\mathbf{x}}, z) \), of (D.3) with \( |x_i| > 2^L \) for some \( i \), and (D.1) is infeasible if and only if there is a basic feasible solution, \( (\bar{\mathbf{x}}, z) \), of (D.3) with \( z \neq 0 \). Furthermore, (D.3) can be written in the form (D.2) such that all these properties hold with \( \text{nnz}(\mathbf{A}') = O(\text{nnz}(\mathbf{A}) + n + m) \), \( n' = O(n) \), \( m' = O(m) \), and \( L' \leq 4L + 2\log(16n) \).

**Proof.** Case 1: Suppose (D.1) is bounded and feasible. It is known that any basic feasible solution of (D.1) is a vector of rational numbers with both absolute value of numerator and denominator are bounded by \( 2^L \) [27]. Therefore, \( -n2^{2L} \leq \text{OPT} \leq n2^{2L} \). Given any feasible solution \( \bar{\mathbf{x}} \) of (D.1), the point \( (\bar{\mathbf{x}}, z = 0) \) is a feasible solution of (D.3) with same cost value. Hence, the linear program (D.3) is feasible and the optimal value of (D.3) is at most \( n2^{2L} \).

On the other hand, clearly (D.3) is feasible because \( \bar{\mathbf{x}} = 0, z = 2^L + 1 \) is an interior point. Furthermore, (D.3) is bounded and therefore has some optimal value. Consider any optimal basic feasible solution \( (\bar{\mathbf{x}}, z) \) of (D.3), we have \( \mathbf{c}'^T \bar{\mathbf{x}} \) is between \( -n2^{2L+1} \) and \( n2^{2L+1} \). Also, \( z \) is a rational number with the absolute value of denominator are bounded by \( 2^L \) using Cramer’s rule. Therefore,
we have \( z \geq 2^{-L_1} \) or \( z = 0 \). If \( z \geq 2^{-L} \), then the total cost is at least \( n2^{3L+4} 2^{-L} - n2^{2L+1} > n2^{2L} \). However, as we argued above, the optimal value of (D.3) is at most \( n2^{2L} \). Therefore, optimal solution has \( z = 0 \) and \( 2^L \geq x_i \geq -2^L \) for all \( i \).

Case 2: Suppose (D.1) is not feasible. In this case, any feasible point \((\bar{x}, z)\) in (D.3) has \( z \neq 0 \) and by the reasoning in the previous section any basic feasible solution has cost greater than \( n2^{2L} \).

Case 3: Suppose (D.1) is not bounded. Let \( \text{OPT}_k = \min \bar{c}^T \bar{x} \) such that \( A \bar{x} \geq \bar{b}, k + 2^L \geq x_i \geq -2^L - k \). Thus, we have \( \text{OPT}_1 < \text{OPT}_0 \) and any optimal point of the case \( k = 1 \) has some coordinate larger than \( 2^L \) or smaller \(-2^L \). By similar argument as above, we have that the optimal point of (D.3) is of the form \((\bar{x}, 0)\) and some coordinate of \( \bar{x} \) is larger than \( 2^L \) or smaller \(-2^L \).

To compute the bit complexity of (D.3) note that we can write (D.3) in the form of (D.2) by choosing

\[
A' = \begin{bmatrix} A & \bar{1} \\ \bar{I} & \bar{0} \\ -\bar{I} & \bar{0} \\ \bar{0}^T & 1 \\ \bar{0}^T & -1 \end{bmatrix}, \quad \bar{b}' = \begin{pmatrix} \bar{b} \\ -2^{L_1+1} \\ 2^{L_1+1} \\ 0 \\ 2^{L_1+1} \end{pmatrix}, \quad \bar{c}' = \begin{pmatrix} \bar{c} \\ n2^{3L+4} \end{pmatrix}
\]

where \( \bar{I} \in \mathbb{R}^{m \times m} \) and \( \bar{0} \in \mathbb{R}^m \) (D.4)

Thus \( n' = n + 1 \), \( m' = 3m + 2 \), and it is easy to see that

\[
d_{\text{max}}(A') = d_{\text{max}}(\begin{bmatrix} A & \bar{1} \end{bmatrix}) \leq n \cdot d_{\text{max}}(A).
\]

Therefore, the bit complexity of (D.3) is at most \( \log(1 + nd_{\text{max}}(A)) + \log(1 + n2^{3L+4}) \leq 4L + 2\log(16n) \) as desired.

Following the approach in [3] to use the following isolation lemma, we show that it is possible to transform the linear program into one with unique optimal solution by randomly perturbing the cost function.

**Lemma 42** ([12]). Given any collection of linear functions on \( n \) variables \( c_1, c_2, \ldots, c_n \) with coefficients in the range \( \{-K, -K-1, \ldots, K-1, K\} \). If \( c_1, \ldots, c_n \) are independently chosen uniformly at random in \( \{-2Kn, \ldots, 2Kn\} \). Then, with probability greater than \( \frac{1}{2} \), there is a unique linear function of minimum value at \( c_1, c_2, \ldots, c_n \).

**Lemma 43.** Suppose that (D.1) is feasible and bounded and consider the following modified linear program

\[
\min \left( 2^{2L+3} n \bar{c} + \bar{r} \right)^T \bar{x} \text{ given } A \bar{x} \geq \bar{b}, \quad (D.5)
\]

where each coordinate in \( \bar{r} \in \mathbb{R}^m \) is chosen uniformly at random from the integers \( \{-2^{L+1} n, \ldots, 2^{L+1} n\} \).

Let \( \text{OPT}' \) denote the optimal value of the linear program (D.5). Given any feasible solution for the linear program (D.5) with cost less than \( \text{OPT} + n^{-1} 2^{-3L-2} \), we can find the active constraints of a basic feasible optimal solution of (D.1) by using only one matrix vector multiplication with \( A \). Furthermore, the bit complexity of (D.5) is at most \( 3L + \log(8n) \).

**Proof.** Since the set of basic solutions to (D.5) and (D.1) are the same, we know that any basic feasible solution of (D.5) is a vector of rational numbers with absolute value of numerator and denominator both bounded by \( 2^L \). Consequently our perturbation of the cost function maintains
that an optimum solution to (D.5) is an optimal solution to (D.1). Hence, the Isolation Lemma shows that with probability greater than \( \frac{1}{2} \), the linear program (D.5) has a unique solution \( \bar{x}^* \).

Now consider the polytope \( P_t = \{ \bar{x} \text{ such that } A\bar{x} \geq \bar{b} \text{ and } (2^{2L+3}n\bar{c} + \bar{n})^T \bar{x} \leq OPT + t2^{-2L-1} \} \) for \( t > 0 \). Since (D.5) has a unique solution, by a similar argument as before, \( P_t \) contains only one basic feasible solution of (D.5) and hence \( P_t - \bar{x}^* = t (P_t - \bar{x}^*) \) for any \( t \leq 1 \). Also, for any \( \bar{x} \in P_t \), \( \bar{x} \) is in the polytope of \( \{ A\bar{x} \geq \bar{b} \} \) and hence \( \| \bar{x} \|_\infty \leq 2^L \). Therefore, for any \( \bar{x} \in P_t \), we have \( \| \bar{x} - \bar{x}^* \|_\infty \leq t \cdot 2^{L+1} \) for any \( t \leq 1 \). Therefore, for any \( \bar{x} \in P_t \), \( \| A\bar{x} - A\bar{x}^* \|_\infty \leq nt2^{L+1} \). Since \( A\bar{x}^* \) is a vector of rational numbers with the absolute value of denominator are bounded by \( 2^L \), we can distinguish if a constraint is satisfied or not when \( nt2^{L+1} < 2^{-L-1} \).

Combining Lemma 41 and Lemma 42 proves Lemma 40.

### E Numerical Linear Algebra for Acceleration

Here we prove Theorem 30 needed for the accelerated linear program solver. Below we restate the theorem for convenience.

**Theorem 44.** Let \( \bar{d}^{(i)} \in \mathbb{R}^m_\geq 0 \) be a sequence of \( r \) positive vectors. Suppose that the number of times that \( d_j^{(i)} \neq d_j^{(i+1)} \) for any \( i \in [r] \) and \( j \in [m] \) is bounded by \( Cr^2 \) for some \( C \geq 1 \). Then if we are given the \( \bar{d}^{(i)} \) in a sequence, in each iteration \( i \) we can compute \( (A^TD_iA)^{-1} \bar{x}_i \) for \( D_i = \text{diag}(\bar{d}_i) \) and arbitrary \( \bar{x}_i \in \mathbb{R}^n \) such that the average cost per iteration is

\[
\tilde{O}\left( \frac{mn^{\omega-1}}{r} + n^2 + C^\omega r^{2\omega} + C^{\omega-1}nr^\omega \right)
\]

where \( \omega < 2.3729/39 \) is the matrix multiplication constant.

**Proof.** For all \( i \in [r] \) let \( B_i = A^TD_iA \). Since \( D_i \in \mathbb{R}^{m \times m} \) is diagonal and \( A \in \mathbb{R}^{n \times m} \) we can compute \( D_1A \) trivially in \( O(mn) \) time. Furthermore from this we can compute \( B_1 = A^TD_1A \) in \( O(mn^{\omega-1}) \) time using fast matrix multiplication by splitting \( A \) into \( \frac{m}{n^2} \) blocks of size \( n \) and using that \( m > n \). Furthermore, using fast matrix multiplication we can then compute \( B_1^{-1} \) in \( O(n^{\omega}) \) time and similarly we can compute \( B_1^{-1}A^T \) in \( O(mn^{\omega-1}) \) time. We use this computation of \( B_1^{-1} \) and \( B_1^{-1}A^T \) in \( O(mn^{\omega-1}) \) time to decrease the running time of future iterations and to trivially compute \( (A^TD_1A)^{-1} \bar{x}_1 \) in \( O(n^2) \) time.

Now for all \( k > 1 \) let \( D_k = D_1 + \Delta_k \) for some diagonal \( \Delta_k \in \mathbb{R}^{m \times m} \) and let \( r_k \overset{\text{def}}{=} \text{nnz}(\Delta_k) \). Thus we can let \( P_k \in \mathbb{R}^{rk \times n} \) be the 1 – 0 matrix that selects the rows of \( A \) for which the diagonal entry in \( \Delta_k \) is nonzero and we can let \( S_k \in \mathbb{R}^{rk \times rk} \) be the diagonal matrix whose diagonal entries are the non-zero diagonal entries of \( \Delta_k \) so that for \( A_k \overset{\text{def}}{=} P_kA \) by the Woodbury matrix identity

\[
B_i^{-1} = (A^TD_1A + A^TP_kS_kP_kA)^{-1} = (B_1 + A_k^TS_kA_k)^{-1} = B_1^{-1} - B_1^{-1}A_k^T(S_k^{-1} + A_kB_1^{-1}A_k^T)^{-1}A_kB_1^{-1}
\]

(E.1)

Now assuming we have \( A_kB_k^{-1}A_k \in \mathbb{R}^{rk \times rk} \) we can use fast matrix multiplication to compute \( (S_k^{-1} + A_kB_k^{-1}A_k^T)^{-1} \) in time \( O(n_k^{\omega}) \). Then using (E.1) the time to compute \( B_i^{-1}\bar{x}_i \) is just

\[
O\left( \text{nnz}(B_1^{-1}) + \text{nnz}(A_k) + \text{nnz}\left((S_k^{-1} + A_kB_k^{-1}A_k^T)^{-1}\right)\right) = O(nr_k + n^2).
\]

57
Consequently, not counting the time to compute $A_kB_k^{-1}A_k^T \in \mathbb{R}^{r_k \times r_k}$ we have that the average cost of computing $B_k^{-1}\bar{x}_i$ is

$$\tilde{O}\left(\frac{mn^{\omega-1}}{r} + n^2 + nr_k + r_k^\omega\right) = \tilde{O}\left(\frac{mn^{\omega-1}}{r} + n^2 + C^{\omega - r^{2\omega}}\right) \quad (E.2)$$

because $r_k \leq Cr^2$ and $Cnr^2 \leq \sqrt{n^2 \sqrt{C} \omega 2^2}$.

All that remains is to estimate the cost of computing $A_k^T B_k^{-1} A_k$. For simplicity, let us suppose that the rows of $A$ are ordered by the iteration in which $P_k$ first selects them with all the unselected rows occurring at the end. This assumption is simply a permutation of $A$ and does not affect the asymptotic running time of our algorithm. With this assumption we have that for all $k \geq 1$ we can write $A_k^T = [A_k^T - \mathbf{U}_k]$ where $\mathbf{U}_k \in \mathbb{R}^{n \times u_k}$ where $u_k = r_k - r_{k-1}$ and the columns of $\mathbf{U}_k$ are simply the rows of $A$. From this, to compute $A_k^T B_k^{-1} A_k$ we see that it suffices to compute

$$\begin{pmatrix} A_k B_k^{-1} A_k^T & A_k B_k^{-1} \mathbf{U}_k \\ \mathbf{U}_k^T B_k^{-1} A_k & \mathbf{U}_k^T B_k^{-1} \mathbf{U}_k \end{pmatrix}$$

Now, since we precomputed $B_k^{-1} A^T$ an $\mathbf{U}$ is just a subset of the rows of $A$ we see that we can compute $B_k^{-1} \mathbf{U}_k$ in $O(nu_k)$ time. Thus, we see that the time to compute $A_k^T B_k^{-1} A_k$ is dominated by the time to multiply a matrix of size at most $r_k \times n$ and $n \times u_k$. We can do this by multiplying $O\left(\frac{r_k \cdot n}{u_k} \cdot \frac{n}{u_k}\right)$ matrices of size $u_k \times u_k$ which can be done in $O(r_k nu_k^{\omega-2})$ time. Thus the average cost of computing $A_k^T B_k^{-1} A_k$ is

$$O\left(\sum_{1 \leq k < r} \left(\frac{1}{r}\right) \cdot (r_k nu_k^{\omega-2})\right) \leq O(Crn \cdot r \cdot (Cr)^{\omega - 2}) = O(C^{\omega - 1} nr^{\omega})$$

where we used the fact that since $\sum_k u_k = r_k$, $c_k \leq Cr^2$ and the minimum value of $\sum_k u_k^{\omega-2}$ is achieve when each $u_k = Cr$. Combining this with (E.2) yields the result. 

\[\square\]