Solving Linear Programs in the Current Matrix Multiplication Time

MICHAEL B. COHEN, Massachusetts Institute of Technology
YIN TAT LEE, The University of Washington & MSR Redmond
ZHAO SONG, The University of Texas at Austin

This article shows how to solve linear programs of the form $\min_{Ax=b, x\geq 0} c^\top x$ with n variables in time

$$O^*((n^\omega + n^{2.5-\alpha/2} + n^{2+1/6}) \log(n/\delta)),$$

where ω is the exponent of matrix multiplication, α is the dual exponent of matrix multiplication, and δ is the relative accuracy. For the current value of $\omega \sim 2.37$ and $\alpha \sim 0.31$, our algorithm takes $O^*(n^\omega \log(n/\delta))$ time. When $\omega = 2$, our algorithm takes $O^*(n^{2+1/6} \log(n/\delta))$ time.

Our algorithm utilizes several new concepts that we believe may be of independent interest:

- We define a stochastic central path method.
- We show how to maintain a projection matrix $\sqrt{W}A^\top(AWA^\top)^{-1}A\sqrt{W}$ in sub-quadratic time under ℓ_2 multiplicative changes in the diagonal matrix W.

CCS Concepts: • **Theory of computation → Linear programming**;

Additional Key Words and Phrases: Linear program, interior point method, matrix multiplication

ACM Reference format:
Michael B. Cohen, Yin Tat Lee, and Zhao Song. 2020. Solving Linear Programs in the Current Matrix Multiplication Time. *J. ACM* 68, 1, Article 3 (January 2021), 39 pages.
https://doi.org/10.1145/3424305

1 INTRODUCTION

Linear programming is one of the key problems in computer science. In both theory and practice, many problems can be reformulated as linear programs to take advantage of fast algorithms. For an

This work was supported in part by NSF Awards No. CCF-1740551, No. CCF-1749609, and No. DMS-1839116. This work was partially supported by Ma Huateng Foundation, Schmidt Foundation, Simons Foundation, NSF, DARPA/SRC, Google, and Amazon.

Authors' addresses: M. B. Cohen, Massachusetts Institute of Technology, Cambridge, Massachusetts; email: micohen@mit.edu; Y. T. Lee, The University of Washington, Paul G. Allen School of Computer Science and Engineering, 3800 E Stevens Way NE, Seattle, WA, 98195, USA; email: yintat@uw.edu; Z. Song, 23 Pine Street, Princeton, NJ, 08542, USA; email: magic.linuxkde@gmail.com.

0004-5411/2020/01-ART3 $15.00
https://doi.org/10.1145/3424305

arbitrary linear program $\min_{Ax=b, x\geq 0} c^\top x$ with n variables and d constraints,[1] the fastest algorithm takes $O^*(\sqrt{d}\cdot \text{nnz}(A) + d^{2.5})$,[2] where $\text{nnz}(A)$ is the number of non-zeros in A [33, 34].

For the generic case $d = \Omega(n)$ that we focus on in this article, the current fastest runtime is dominated by $O^*(n^{2.5})$. This runtime has not been improved since a result by Vaidya in 1989 [55, 57]. The $n^{2.5}$ bound originated from two factors: the cost per iteration n^2 and the number of iterations $\sqrt{n}$. The n^2 cost per iteration looks optimal, because this is the cost to compute Ax for a dense A. Therefore, many efforts [22, 33, 43, 47, 56] have been focused on decreasing the number of iterations while maintaining the cost per iteration. As for many important linear programs (and convex programs), the number of iterations has been decreased, including maximum flow [38, 39], minimum cost flow [14], geometric median [13], matrix scaling and balancing [15], and ℓ_p regression [5]. Unfortunately, beating $\sqrt{n}$ iterations (or $\sqrt{d}$ when $d \ll n$) for the general case remains one of the biggest open problems in optimization.

Avoiding this open problem, this article develops a stochastic central path method that has a runtime of $O^*(n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})$, where ω is the exponent of matrix multiplication and α is the dual exponent of matrix multiplication.[3] For the current value of $\omega \sim 2.38$ and $\alpha \sim 0.31$, the runtime is simply $O^*(n^\omega)$. This achieves a natural barrier for solving linear programs, because linear systems are a special case of linear program and this is the best known runtime for solving linear systems. Although References [1–3] showed that the exact and similar[4] approaches used in References [16, 17, 30, 60] cannot give a bound on ω better than 2.168, we believe improving the additive $2 + 1/6$ term remains important for understanding linear programming. A recent work [21] improved the $2 + 1/6$ term to $2 + 1/18$.

Our method is a stochastic version of the short step central path method. This short step method takes $O^*(\sqrt{n})$ steps and each step decreases $x_i s_i$ by a $1 - 1/\sqrt{n}$ factor for all i where x is the primal variable and s is the dual variable [47] (see the definition of s in Equation (1)). This results in $O^*(\sqrt{n}) \times n = O^*(n^{1.5})$ coordinate updates. Our method takes the same number of steps but only updates $\widetilde{O}(\sqrt{n})$ coordinates each step. Therefore, we only update $O^*(n)$ coordinates in total, which is nearly optimal.

Our framework is efficient enough to take a much smaller step while maintaining the same running time. For the current value of $\omega \sim 2.38$, we show how to obtain the same runtime of $O^*(n^\omega)$ by taking $O^*(n)$ steps and $\widetilde{O}(1)$ coordinates update per steps. This is because the complexity of each step decreases proportionally when the step size decreases. Beyond the cost per iteration, we remark that our algorithm is one of the very few central path algorithms [38, 39, 46] that does not maintain $x_i s_i$ close to some ideal vector in ℓ_2 norm. We are hopeful that our stochastic method and our proof will be useful for future research on interior point methods.

1.1 Related Work

Interior point method has a long history, for more detailed surveys, we refer the readers to References [40, 48, 49, 54, 62, 63]. This article is in part inspired by the use of data-structure in Laplacian solvers [9, 11, 12, 23–28, 53], in particular the cycle update in Reference [23].

[1] Throughout this article, we assume there is no redundant constraints and hence $n \geq d$. Note that papers in different communities uses different symbols to denote the number of variables and constraints in a linear program.

[2] We use O^* to hide $n^{o(1)}$ and $\log^{O(1)}(1/\delta)$ factors and $\widetilde{O}$ to hide $\log^{O(1)}(n/\delta)$ factors.

[3] The dual exponent of matrix multiplication α is the supremum among all $a \geq 0$ such that it takes $n^{2+o(1)}$ time to multiply an $n \times n$ matrix by an $n \times n^a$ matrix.

[4] Improving the matrix multiplication constant boils down to constructing/analyzing the tensors in better sense. Those work about limitation of matrix multiplication constant explore the exact same tensor and also variation of tensor in the previous work. For more details, we refer the readers to matrix multiplication literatures.

In a few recent follow-ups of this paper, the techniques developed in this work are generalized to a more broad class of optimization problems, i.e., Empirical Risk Minimization [36], Cutting plane method [20], semi-definite programming [19], deep neural network training [4, 7]. A deterministic variant of our algorithm has been developed [58], a sketching variant of our algorithm has been developed [51], a streaming variant has been developed [37], the additive 1/6 term has been improved to 1/18 [21], and the runtime has been improved to nearly linear time for dense linear programs with $n \gg d$ [59].

Matrix vector multiplication is a subtask of our iterative algorithm for solving linear programs. Online matrix-vector multiplication [6, 18, 29] is closely related to our problem, but usually the computational model is different than our setting.

2 RESULTS AND TECHNIQUES

Theorem 2.1 (Main Result). *Given a linear program* $\min_{Ax=b, x\geq 0} c^\top x$ *with no redundant constraints. Assume that the polytope has diameter* R *in* ℓ_1 *norm, namely, for any* $x \geq 0$ *with* $Ax = b$, *we have* $\|x\|_1 \leq R$.

Then, for any $0 < \delta \leq 1$, Main(A, b, c, δ) *outputs* $x \geq 0$ *such that*

$$c^\top x \leq \min_{Ax=b, x\geq 0} c^\top x + \delta \cdot \|c\|_\infty R \quad \text{and} \quad \|Ax - b\|_1 \leq \delta \cdot \left(R \sum_{i,j} |A_{i,j}| + \|b\|_1 \right)$$

in expected time

$$\left(n^{\omega + o(1)} + n^{2.5 - \alpha/2 + o(1)} + n^{2+1/6+o(1)} \right) \cdot \log\left(\frac{n}{\delta}\right),$$

where ω *is the exponent of matrix multiplication, and* α *is the dual exponent of matrix multiplication.*

For the current value of $\omega \sim 2.38$ *and* $\alpha \sim 0.31$, *the expected time is simply* $n^{\omega+o(1)} \log(\frac{n}{\delta})$.

See References [47] and [32, Sections E and F] on the discussion on converting an approximation solution to an exact solution. For integral A, b, c, it suffices to pick $\delta = 2^{-O(L)}$ to get an exact solution where $L = \log(1 + d_{\max} + \|c\|_\infty + \|b\|_\infty)$ is the bit complexity and $d_{\max}$ is the largest absolute value of the determinant of a square sub-matrix of A. For many combinatorial problems, $L = O(\log(n + \|b\|_\infty + \|c\|_\infty))$.

In this article, we assume all floating point calculations are done exactly for simplicity. In general, the algorithm can be carried out with $O(L)$ bits of accuracy. See Reference [47] for some discussions on the numerical stability of the interior point methods.

If $T(n)$ is the current cost of matrix multiplication and inversion with $T(n) \sim n^{2.38}$, then our runtime is simply $O(T(n) \log n \log(\frac{n}{\delta}))$. The $\log(\frac{n}{\delta})$ comes from iteration count and the $\log n$ factor comes from the doubling trick ($|y_{\pi(1.5r)}| \geq (1 - 1/\log n)|y_{\pi(r)}|$) in the projection maintenance section. We left the problem of obtaining $O(T(n) \log(\frac{n}{\delta}))$ as an open problem.

Finally, we note that our runtime holds for any square and rectangular matrix multiplication algorithm as long as $\omega \leq 3 - \alpha$ (see Lemma A.4).[5] For example, Strassen algorithm together with a simple rectangular multiplication algorithm gives a runtime of roughly $n^{2.807}$.

2.1 Central Path Method

Our algorithm relies on two new ingredients: stochastic central path and projection maintenance. The central path method considers the linear programs

$$\min_{Ax=b, x\geq 0} c^\top x \quad \text{(primal)} \quad \text{and} \quad \max_{A^\top y \leq c} b^\top y \quad \text{(dual)},$$

[5] Reference [8] proved a stronger result $\omega + 0.5\omega\alpha \leq 3$.

with $A \in \mathbb{R}^{d \times n}$. Any solution of the linear program satisfies the following optimality conditions:

$$\begin{aligned} x_i s_i &= 0 \text{ for all } i, \\ Ax &= b, \\ A^\top y + s &= c, \\ x_i, s_i &\geq 0 \text{ for all } i. \end{aligned} \tag{1}$$

We call (x, s, y) feasible if it satisfies the last three equations above. For any feasible (x, s, y), the duality gap of (x, s, y) is $\sum_i x_i s_i$. The central path method finds a solution of the linear program by following the central path, which uniformly decreases the duality gap. The central path $(x_t, s_t, y_t) \in \mathbb{R}^{n+n+d}$ is a path parameterized by t and defined by

$$\begin{aligned} x_{t,i} s_{t,i} &= t \text{ for all } i, \\ Ax_t &= b, \\ A^\top y_t + s_t &= c, \\ x_{t,i}, s_{t,i} &\geq 0 \text{ for all } i, \end{aligned} \tag{2}$$

where $x_{t,i}$ is the ith coordinate of x_t and $s_{t,i}$ is the ith coordinate of s_t. It is known [64] how to transform linear programs by adding $O(n)$ many variables and constraints so that:

- The optimal solution remains the same.
- The central path at $t = 1$ is near $(1_n, 1_n, 0_d)$ where 1_n and 0_d are all 1 and all 0 vectors with lengths n and d.
- It is easy to convert an approximate solution of the transformed program to the original one.

For completeness, a theoretical version of such result is included in Lemma A.6. This result shows that it suffices to move gradually (x_1, s_1, y_1) to (x_t, s_t, y_t) for small enough t.

2.1.1 Short Step Central Path Method. The short step central path method maintains $x_i s_i = \mu_i$ for some vector μ such that

$$\sum_i (\mu_i - t)^2 = O(t^2) \quad \text{for some scalar } t > 0. \tag{3}$$

Since the duality gap is $\sum_i \mu_i$, it suffices to find x and s satisfying the above equation with small enough t. There are many variants of central path methods. We will focus on the version that decreases t and takes a step of μ at the same time. The purpose of moving μ is to maintain the invariant Equation (3) and the purpose of decreasing t is decrease the duality gap, which is roughly nt. One natural way to maintain the invariant Equation (3) is to do a gradient descent step on the energy $\sum_i (\mu_i - t)^2$ defined in Equation (3), namely, moving μ to $\mu - h(\mu - t)$ with step size h.[6]

More generally, say we want to move from μ to $\mu + \delta_\mu$, we approximate the term $(x + \delta_x)_i (s + \delta_s)_i$ by $x_i s_i + x_i \delta_{s,i} + s_i \delta_{x,i}$ and obtain the following system:

$$\begin{aligned} X\delta_s + S\delta_x &= \delta_\mu, \\ A\delta_x &= 0, \\ A^\top \delta_y + \delta_s &= 0, \end{aligned} \tag{4}$$

[6]The classical view of central path method is to take a Newton step on the system Equation (2), which turns out to be same as taking a gradient step on the energy defined in Equation (3). However, our main algorithm will choose a different energy and this gradient descent view is crucial for designing our algorithm.

where $X = \text{diag}(x)$ and $S = \text{diag}(s)$. This equation is the linear approximation of the original goal (moving from μ to $\mu + \delta_\mu$), and that the step is explicitly given by the formula

$$\delta_x = \frac{X}{\sqrt{XS}}(I - P)\frac{1}{\sqrt{XS}}\delta_\mu \text{ and } \delta_s = \frac{S}{\sqrt{XS}}P\frac{1}{\sqrt{XS}}\delta_\mu, \tag{5}$$

where $P = \sqrt{\frac{X}{S}}A^\top \left(A\frac{X}{S}A^\top\right)^{-1} A\sqrt{\frac{X}{S}}$ is an orthogonal projection and the formulas $\frac{X}{\sqrt{XS}}, \frac{X}{S}, \ldots$ are the diagonal matrices of the corresponding vectors.

In turns out that one can decrease t by $1 - \frac{1}{\sqrt{n}}$ a multiplicative factor every iteration while maintaining the invariant Equation (3). This requires $\widetilde{O}(\sqrt{n})$ iterations to converge. Combining this with the inverse maintenance technique [55], this gives a total runtime of $n^{2.5}$. More precisely, the algorithm maintains the invariant $\sum_i (\mu_i - t)^2 = O(t^2)$ by making steps bring μ_i closer to t while taking steps to decrease μ_i uniformly. The progress of the whole algorithm is measured by t, because the duality gap of (x_t, s_t, y_t) is bounded by nt.

2.1.2 Stochastic Central Path Method. This part discusses how to modify the short step central path to decrease the cost per iteration to roughly $n^{\omega-\frac{1}{2}}$. Since our goal is to implement a central path method in sub-quadratic time per iteration, we do not even have the budget to compute Ax every iterations. Therefore, instead of maintaining $(A\frac{X}{S}A^\top)^{-1}$ as shown in previous papers, we will study the problem of maintaining a projection matrix $P = \sqrt{\frac{X}{S}}A^\top(A\frac{X}{S}A^\top)^{-1}A\sqrt{\frac{X}{S}}$ due to the formula of δ_x and δ_s, Equation (5).

However, even if the projection matrix P is given explicitly for free, it is difficult to multiply the dense projection matrix with a dense vector δ_μ in time $o(n^2)$. To avoid moving along a dense δ_μ, we move along an $O(k)$ sparse direction $\widetilde{\delta}_\mu$ defined by

$$\widetilde{\delta}_{\mu,i} = \begin{cases} \delta_{\mu,i}/p_i, \text{ with probability } p_i \overset{\text{def}}{=} k \cdot \left(\frac{\delta_{\mu,i}^2}{\sum_l \delta_{\mu,l}^2} + \frac{1}{n}\right), \\ 0, \qquad \text{else.} \end{cases} \tag{6}$$

The sparse direction is defined so that we are moving in the same direction in expectation ($\mathbf{E}[\widetilde{\delta}_{\mu,i}] = \delta_{\mu,i}$) and that the direction has as small variance as possible ($\mathbf{E}[\widetilde{\delta}_{\mu,i}^2] \leq \frac{\sum_i \delta_{\mu,i}^2}{k}$). If the projection matrix is given explicitly, then we can apply the projection matrix on $\widetilde{\delta}_\mu$ in time $O(nk)$. This article picks $k \sim \sqrt{n}$ and the sum of the cost of projection vector multiplications in the whole algorithm is about $nk^2 = n^2$.

During the whole algorithm, we maintain a projection matrix

$$\overline{P} = \sqrt{\frac{\overline{X}}{\overline{S}}}A^\top \left(A\frac{\overline{X}}{\overline{S}}A^\top\right)^{-1} A\sqrt{\frac{\overline{X}}{\overline{S}}}$$

for vectors $\overline{x}$ and $\overline{s}$ such that $\overline{x}_i$ and $\overline{s}_i$ are multiplicative approximations of x_i and s_i, respectively, for all i. Since we maintain the projection at a nearby point $(\overline{x}, \overline{s})$, our stochastic step $x \leftarrow x + \widetilde{\delta}_x$, $s \leftarrow s + \widetilde{\delta}_s$ and $y \leftarrow y + \widetilde{\delta}_y$ are defined by

$$\begin{aligned} \overline{X}\widetilde{\delta}_s + \overline{S}\widetilde{\delta}_x &= \widetilde{\delta}_\mu, \\ A\widetilde{\delta}_x &= 0, \\ A^\top\widetilde{\delta}_y + \widetilde{\delta}_s &= 0, \end{aligned} \tag{7}$$

which is different from Equation (4) on both sides of the first equation. Note that this system uses $\overline{X}$ and $\overline{S}$, because we have only maintained this projection matrix. The main goal of Section 4 is to show $\overline{X} = \Theta(X)$ and $\overline{S} = \Theta(S)$ is good enough for our interior point method. Similar to Equation (5), Lemma 4.2 shows that

$$\widetilde{\delta}_x = \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}}(I - \overline{P})\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu \text{ and } \widetilde{\delta}_s = \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}}\overline{P}\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu. \tag{8}$$

The previously fastest algorithm involves maintaining the matrix inverse $(A\frac{X}{S}A^\top)^{-1}$ using subspace embedding techniques [10, 41, 50] and leverage score sampling [52]. In this article, we maintain the projection directly using lazy updates.

The key departure from the central path we present is that we can only maintain

$$0.9t \le \mu_i = x_i s_i \le 1.1t, \quad \text{for some } t > 0,$$

instead of μ close to t in ℓ_2 norm. We will further explain the proof in Section 4.1.

2.2 Projection Maintenance via Lazy Update

The projection matrix we maintain is of the form $\sqrt{W}A^\top (AWA^\top)^{-1} A\sqrt{W}$ where $W = \mathrm{diag}(x/s)$. For intuition, we only explain how to maintain the matrix $M_w \stackrel{\text{def}}{=} A^\top (AWA^\top)^{-1}A$ for the short step central path step here. In this case, we have $\sum_i (\frac{w_i^{\text{new}} - w_i}{w_i})^2 = O(1)$ for each step. Given this, there are mainly two extreme cases, w changes uniformly on all coordinates and w changes only on a few coordinates.

If the changes $(\frac{w_i^{\text{new}} - w_i}{w_i})^2$ is uniform across all the coordinates, then $w_i^{\text{new}} = (1 \pm \frac{1}{\sqrt{n}})w_i$ for all i. Since it takes $\sqrt{n}$ steps to change all coordinates by a constant factor and we only need to maintain M_v for some $v_i = \Theta(w_i)$ for all i, we can update the matrix every $\sqrt{n}$ steps. Hence, the average cost per iteration of maintaining the projection matrix is $n^{\omega - \frac{1}{2}}$, which is exactly what we desired.

For the other extreme case when w changes on only a few coordinates, only $\sqrt{n}$ coordinates are changed by a constant factor during all $\sqrt{n}$ iterations. In this case, instead of updating M_w every step, we can compute $M_w h$ online by the Woodbury matrix identity.

Fact 2.2 ([61]). *The Woodbury matrix identity is*

$$(M + UCV)^{-1} = M^{-1} - M^{-1}U(C^{-1} + VM^{-1}U)^{-1}VM^{-1}.$$

Let $S \subset [n]$ denote the set of coordinates that is changed by more than a constant factor and $r = |S|$. Using the identity above, we have that

$$M_{w^{\text{new}}} = M_w - (M_w)_S \left(\Delta_{S,S}^{-1} + (M_w)_{S,S}\right)^{-1} ((M_w)_S)^\top, \tag{9}$$

where $\Delta = \mathrm{diag}(w^{\text{new}} - w)$, $(M_w)_S \in \mathbb{R}^{n \times r}$ is the r columns from S of M_w and $(M_w)_{S,S}, \Delta_{S,S} \in \mathbb{R}^{r \times r}$ are the r rows and columns from S of M_w and Δ.

As long as there are only few coordinates violating $v_i = \Theta(w_i)$, Equation (9) can be applied online efficiently. In another case, we can use Equation (9) instead to update the matrix M_w and the cost is dominated by multiplying a $n \times n$ matrix with a $n \times r$ matrix.

Theorem 2.3 (Rectangular Matrix Multiplication, [31]). *Let the dual exponent of matrix multiplication α be the supremum among all $a \ge 0$ such that it takes $n^{2+o(1)}$ time to multiply an $n \times n$ matrix by an $n \times n^a$ matrix.*

Then, for any $n \geq r$, multiplying an $n \times r$ with an $r \times n$ matrix or $n \times n$ with $n \times r$ takes time

$$n^{2+o(1)} + r^{\frac{\omega-2}{1-\alpha}} n^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)}.$$

Furthermore, we have $\alpha > 0.31389$.

See Lemma A.5 for the origin of the formula. Since the cost of multiplying $n \times n$ matrix by a $n \times 1$ matrix is same as the cost for $n \times n$ with $n \times n^{0.31}$, Equation (9) should be used to update at least $n^{0.31}$ coordinates. In the extreme case only few w_i are changing, we only need to update the matrix $n^{\frac{1}{2}-0.31}$ times during the whole algorithm and each takes n^2 time, and hence the total cost is less than n^ω for the current value of $\omega \sim 2.37$.

In previous papers [22, 33, 34, 42, 44, 57], the matrix is updated in a fixed schedule independent of the input sequence w. This leads to sub-optimal bounds if used in this article. We instead define a potential function to measure the distance between the approximate vector v and the target vector w. When there are less than n^α coordinates of v that is far from w, we are lazy and do not update the matrix. We simply apply the Woodbury matrix identity online. When there are more than n^α coordinates, we update v by a certain greedy step. As in the extreme cases, the worst case for our algorithm is when w changes uniformly across all coordinates and hence the worst case runtime is $n^{\omega-\frac{1}{2}}$ per iteration. We will further explain the potential in Section 5.1.

3 NOTATIONS

For notational convenience, we assume the number of variables $n \geq 10$ and there are no redundant constraints. In particular, this implies that the constraint matrix A is full rank and $n \geq d$.

For a positive integer n, let $[n]$ denote the set $\{1, 2, \ldots, n\}$.

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (respectively, $\gtrsim$) to indicate that $f \leq Cg$ (respectively, $\geq$) for some absolute constant C.

We use $\sinh x$ to denote $\frac{e^x - e^{-x}}{2}$ and $\cosh x$ to denote $\frac{e^x + e^{-x}}{2}$.

For vectors $a, b \in \mathbb{R}^n$ and accuracy parameter $\epsilon \in (0, 1)$, we use $a \approx_\epsilon b$ to denote that $(1-\epsilon)b_i \leq a_i \leq (1+\epsilon)b_i, \forall i \in [n]$. Similarly, for any scalar t, we use $a \approx_\epsilon t$ to denote that $(1-\epsilon)t \leq a_i \leq (1+\epsilon)t, \forall i \in [n]$.

For a vector $x \in \mathbb{R}^n$ and $s \in \mathbb{R}^n$, we use xs to denote a length n vector with the ith coordinate $(xs)_i$ is $x_i \cdot s_i$. Similarly, we extend other scalar operations to vector coordinate-wise.

Given vectors $x, s \in \mathbb{R}^n$, we use X and S to denote the diagonal matrix of those two vectors. We use $\frac{X}{S}$ to denote the diagonal matrix given $(\frac{X}{S})_{i,i} = x_i/s_i$. Similarly, we extend other scalar operations to diagonal matrix diagonal-wise. Note that matrix $\sqrt{\frac{X}{S}} A^\top (A \frac{X}{S} A^\top)^{-1} A \sqrt{\frac{X}{S}}$ is an orthogonal projection matrix.

4 STOCHASTIC CENTRAL PATH METHOD

4.1 Proof Outline

The short step central path method is defined using the approximation $(x + \delta_x)_i (s + \delta_s)_i \sim x_i s_i + x_i \delta_{s,i} + s_i \delta_{x,i}$. This approximate is accurate if $\|X^{-1}\delta_x\|_\infty \leq 1/2$ and $\|S^{-1}\delta_s\|_\infty \leq 1/2$. For the δ_x step, we have

$$X^{-1}\delta_x = \frac{1}{\sqrt{XS}}(I - P)\frac{1}{\sqrt{XS}}\delta_\mu \sim \frac{1}{t}(I - P)\delta_\mu, \tag{10}$$

where we used $x_i s_i \sim t$ for all i.

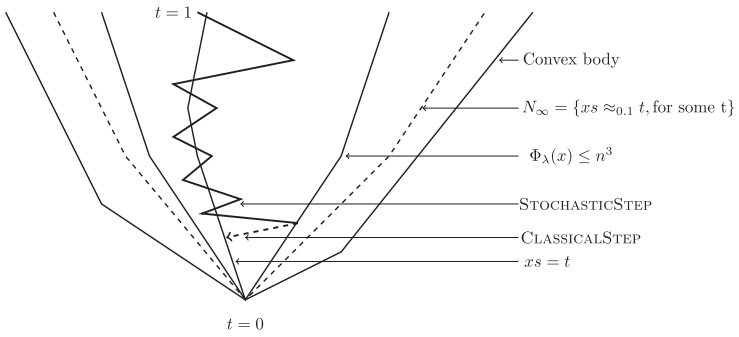

Fig. 1. ClassicalStep happens with n^{-2} probability.

If we know that $\|\delta_\mu\|_2 \le t/4$, then the ℓ_∞ norm can be roughly bounded as follows:

$$\|X^{-1}\delta_x\|_\infty \le \|X^{-1}\delta_x\|_2 \lesssim \frac{1}{t}\|(I-P)\delta_\mu\|_2 \le \frac{1}{t}\|\delta_\mu\|_2 \le 1/2,$$

where we used that $I - P$ is an orthogonal projection matrix. This is the reason why a standard choice of $\delta_{\mu,i}$ is $-ct/\sqrt{n}$ for all i for some small constant c.

For the stochastic step, $\widetilde{\delta}_{\mu,i} \sim -\frac{t}{\sqrt{n}}\frac{n}{k}$ for roughly k coordinates where the term $\frac{n}{k}$ is used to preserve the expectation of the step. Therefore, the ℓ_2 norm of $\widetilde{\delta}_\mu$ is very large ($\|\widetilde{\delta}_\mu\|_2 \sim t\sqrt{\frac{n}{k}}$). After the projection, we have $\|X^{-1}\delta_x\|_2 \sim \frac{1}{t}\|(I-P)\delta_\mu\|_2 \sim \sqrt{\frac{n}{k}}$. Hence, the bound of $\|X^{-1}\delta_x\|_\infty$ using $\|X^{-1}\delta_x\|_2$ is too weak. To improve the bound, we use Chernoff bounds to estimate $\|X^{-1}\delta_x\|_\infty$. To simplify the proof, we use a loop in Algorithm 1 to ensure both the sup norm is always small not just with high probability.

Beside the ℓ_∞ norm bound, the proof sketch in Equation (10) also requires using $x_i s_i \sim t$ for all i. The short step central path proof maintains an invariant that $\sum_i (x_i s_i - t)^2 = O(t^2)$. However, since our stochastic step has a stochastic noise with ℓ_2 norm as large as $t\sqrt{\frac{n}{k}}$, one cannot hope to maintain $x_i s_i$ close to t in ℓ_2 norm. Instead, we follow an idea in References [33, 35] and maintain the following potential

$$\sum_{i=1}^{n} \cosh\left(\lambda\left(\frac{x_i s_i}{t} - 1\right)\right) = n^{O(1)},$$

with $\lambda = \Theta(\log n)$. This potential is a variant of soft-max. Note that the potential bounded by $n^{O(1)}$ implies that $x_i s_i$ is a multiplicative approximation of t. To bound the potential, consider $r_i = \frac{x_i s_i}{t}$ and $\Phi(r)$ be the potential above. Then, we have that

$$\mathbf{E}[\Phi(r^{\text{new}})] \le \Phi(r) + \langle \nabla\Phi(r), \mathbf{E}[r^{\text{new}} - r]\rangle + O(1)\mathbf{E}\left[\|r^{\text{new}} - r\|^2_{\nabla^2\Phi(r)}\right].$$

The first-order term can be bounded efficiently, because $\mathbf{E}[r^{\text{new}} - r]$ is close to the short step central path step. The second term is a variance term that scales like $1/k$ due to the k independent coordinates. Therefore, the potential changed by $1/k \sim 1/\sqrt{n}$ factor each step. Hence, we can maintain it for roughly $\sqrt{n}$ steps.

ALGORITHM 1

1: **procedure** StochasticStep(mp, $x, s, \delta_\mu, k, \epsilon$) ▷ Lemma 4.2, 4.3, 4.8
2: $w \leftarrow \frac{x}{s}, \widetilde{v} \leftarrow$ mp.Update(w) ▷ Algorithm 3
3: $\overline{x} \leftarrow x\sqrt{\frac{\widetilde{v}}{w}}, \overline{s} \leftarrow s\sqrt{\frac{w}{\widetilde{v}}}$ ▷ It guarantees that $\frac{\overline{x}}{\overline{s}} = \widetilde{v}$ and $\overline{xs} = xs$
4: **repeat**
5: Generate $\widetilde{\delta}_\mu$ such that ▷ Compute a sparse direction
6: $\widetilde{\delta}_{\mu,i} \leftarrow \begin{cases} \delta_{\mu,i}/p_i, & \text{with prob. } p_i = \min(1, k \cdot ((\delta_{\mu,i}^2 / \sum_{l=1}^n \delta_{\mu,l}^2) + 1/n)); \\ 0 & \text{else.} \end{cases}$
7: ▷ Compute an approximate step
8: ▷ Find $(\widetilde{\delta}_x, \widetilde{\delta}_s, \widetilde{\delta}_y)$ such that these three equations hold

$$\overline{X}\widetilde{\delta}_s + \overline{S}\widetilde{\delta}_x = \widetilde{\delta}_\mu,$$
$$A\widetilde{\delta}_x = 0,$$
$$A^\top \widetilde{\delta}_y + \widetilde{\delta}_s = 0.$$

9: $p_\mu \leftarrow$ mp.Query($\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu$) ▷ Algorithm 3
10: $\widetilde{\delta}_s \leftarrow \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}} p_\mu$ ▷ According to (11)
11: $\widetilde{\delta}_x \leftarrow \frac{1}{\overline{S}}\widetilde{\delta}_\mu - \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}} p_\mu$ ▷ According to (12)
12: **until** $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \le \frac{1}{100 \log n}$ and $\|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \le \frac{1}{100 \log n}$
13: **return** $(x + \widetilde{\delta}_x, s + \widetilde{\delta}_s)$
14: **end procedure**

To make sure the potential Φ is bounded during the whole algorithm, our step is the mixtures of two steps of the form $\delta_\mu \sim -\frac{t}{\sqrt{n}} - t\frac{\nabla\Phi}{\|\nabla\Phi\|_2}$. The first term is to decrease t and the second term is to decrease Φ.

Since the algorithm is randomized, there is a tiny probability that Φ is large. In that case, we switch to a short step central path method. See Figure 1 and Algorithms 1 and 2. The first part of the proof involves bounding every quantity listed in Table 1. In the second part, we are using these quantities to bound the expectation of Φ.

To decouple the proof in both parts, we will make the following assumption in the first part. It will be verified in the second part.

Assumption 4.1. *Assume the following for the input of the procedure StochasticStep (see Algorithm 1):*

- $xs \approx_{0.1} t$ *with* $t > 0$.
- mp.*Update*(w) *outputs* $\widetilde{v}$ *such that* $w \approx_{\epsilon_{\text{mp}}} \widetilde{v}$ *with* $\epsilon_{\text{mp}} \le 1/40{,}000$.
- $\|\delta_\mu\|_2 \le \epsilon t$ *with* $0 < \epsilon < 1/(40{,}000 \log n)$.
- $k \ge 1{,}000\epsilon\sqrt{n}\log^2 n/\epsilon_{\text{mp}}$.

The data structure mp in both Algorithms 1 and 2 is used to maintain some approximation of the projection matrix. It is formally defined in Section 5. For this section, the only facts we need is that $w \approx_{\epsilon_{\text{mp}}} \widetilde{v}$ stated in the assumption and that the vector mp.Query(w) outputs satisfies line 8 in Algorithm 1.

ALGORITHM 2: Our main algorithm

procedure MAIN(A, b, c, δ) ▷ Theorem 2.1
  $\epsilon \leftarrow \frac{1}{40000 \log n}$, $\epsilon_{\text{mp}} \leftarrow \frac{1}{40000}$, $k \leftarrow \frac{1000\epsilon\sqrt{n}\log^2 n}{\epsilon_{\text{mp}}}$.
  $\lambda \leftarrow 40 \log n$, $\delta \leftarrow \min(\frac{\delta}{2}, \frac{1}{\lambda})$, $a \leftarrow \min(\alpha, 2/3)$.
  Modify the linear program and obtain an initial x and s according to Lemma A.6.
  MAINTAINPROJECTION mp
  mp.INITIALIZE($A, \frac{x}{s}, \epsilon_{\text{mp}}, a$) ▷ Algorithm 3
  $t \leftarrow 1$ ▷ Initialize t
  while $t > \delta^2/(32n^3)$ **do** ▷ We stop once the error is small enough
    $t^{\text{new}} \leftarrow (1 - \frac{\epsilon}{3\sqrt{n}})t$
    $\mu \leftarrow xs$
    $\delta_\mu \leftarrow (\frac{t^{\text{new}}}{t} - 1)xs - \frac{\epsilon}{2} \cdot t^{\text{new}} \cdot \frac{\nabla\Phi_\lambda(\mu/t-1)}{\|\nabla\Phi_\lambda(\mu/t-1)\|_2}$ ▷ Φ_λ is defined in Lemma 4.12
    $(x^{\text{new}}, s^{\text{new}}) \leftarrow$ STOCHASTICSTEP(mp, $x, s, \delta_\mu, k, \epsilon$) ▷ Algorithm 1
    if $\Phi_\lambda(\mu^{\text{new}}/t^{\text{new}} - 1) > n^3$ **then** ▷ When potential function is large
      $(x^{\text{new}}, s^{\text{new}}) \leftarrow$ CLASSICALSTEP(x, s, t^{new}) ▷ Lemma A.2, [57]
      mp.INITIALIZE($A, \frac{x^{\text{new}}}{s^{\text{new}}}, \epsilon_{\text{mp}}, a$) ▷ Restart the data structure
    end if
    $(x, s) \leftarrow (x^{\text{new}}, s^{\text{new}})$, $t \leftarrow t^{\text{new}}$
  end while
  Return an approximate solution of the original linear program according to Lemma A.6.
end procedure

4.2 Bounding Each Quantity of Stochastic Step

First, we give an explicit formula for our step, which will be used in all subsequent calculations.

LEMMA 4.2. *The procedure* STOCHASTICSTEP(mp, $x, s, \delta_\mu, k, \epsilon$) *(see Algorithm 1) finds a solution* $\widetilde{\delta}_x$, $\widetilde{\delta}_s \in \mathbb{R}^n$ *to (7) by the formula*

$$\widetilde{\delta}_x = \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}}(I - \overline{P})\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu, \tag{11}$$

$$\widetilde{\delta}_s = \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}}\overline{P}\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu, \tag{12}$$

with

$$\overline{P} = \sqrt{\frac{\overline{X}}{\overline{S}}}A^\top \left(A\frac{\overline{X}}{\overline{S}}A^\top\right)^{-1} A\sqrt{\frac{\overline{X}}{\overline{S}}}. \tag{13}$$

PROOF. For the first equation of Equations (7), we multiply $A\overline{S}^{-1}$ on both sides,

$$A\overline{S}^{-1}\overline{X}\widetilde{\delta}_s + A\widetilde{\delta}_x = A\overline{S}^{-1}\widetilde{\delta}_\mu.$$

Since the second equation gives $A\widetilde{\delta}_x = 0$, then we know that $A\overline{S}^{-1}\overline{X}\widetilde{\delta}_s = A\overline{S}^{-1}\widetilde{\delta}_\mu$.

Multiplying $A\overline{S}^{-1}\overline{X}$ on both sides of the third equation of Equations (7), we have

$$-A\overline{S}^{-1}\overline{X}A^\top\widetilde{\delta}_y = A\overline{S}^{-1}\overline{X}\widetilde{\delta}_s = A\overline{S}^{-1}\widetilde{\delta}_\mu.$$

Table 1. The Bound of Each Quantity Under Assumption 4.1

Quantity	Bound	Place
$\|\mathbf{E}[s^{-1}\widetilde{\delta}_s]\|_2, \|\mathbf{E}[x^{-1}\widetilde{\delta}_x]\|_2, \|\mathbf{E}[\mu^{-1}\widetilde{\delta}_\mu]\|_2$	$O(\epsilon)$	Part 1, Lemma 4.3
$\|\mathbf{E}[\mu^{-1}(\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu)]\|_2$	$O(\epsilon_{\text{mp}} \cdot \epsilon)$	Part 1, Lemma 4.8
$\|\mathbf{E}[\mu^{-1}(\mu^{\text{new}} - \mu)]\|_2$	$O(\epsilon)$	Part 1, Lemma 4.8
$\mathbf{Var}[s_i^{-1}\widetilde{\delta}_{s,i}], \mathbf{Var}[x_i^{-1}\widetilde{\delta}_{x,i}], \mathbf{Var}[\mu_i^{-1}\widetilde{\delta}_{\mu,i}]$	$O(\epsilon^2/k)$	Part 2, Lemma 4.3
$\mathbf{Var}[\mu_i^{-1}\mu^{\text{new}}]$	$O(\epsilon^2/k)$	Part 2, Lemma 4.8
$\|s^{-1}\widetilde{\delta}_s\|_\infty, \|x^{-1}\widetilde{\delta}_x\|_\infty, \|\mu^{-1}\widetilde{\delta}_\mu\|_\infty$	$O(1/\log n)$	Part 3, Lemma 4.3
$\|\mu^{-1}(\mu^{\text{new}} - \mu)\|_\infty$	$O(1/\log n)$	Part 3, Lemma 4.8

For intuition, think $\epsilon \sim \epsilon_{\text{mp}} \sim 1/10$ and $k \sim \sqrt{n}$.

Thus,

$$\begin{aligned}
\widetilde{\delta}_y &= -(A\overline{S}^{-1}\overline{X}A^\top)^{-1}A\overline{S}^{-1}\widetilde{\delta}_\mu, \\
\widetilde{\delta}_s &= A^\top(A\overline{S}^{-1}\overline{X}A^\top)^{-1}A\overline{S}^{-1}\widetilde{\delta}_\mu, \\
\widetilde{\delta}_x &= \overline{S}^{-1}\widetilde{\delta}_\mu - \overline{S}^{-1}\overline{X}A^\top(A\overline{S}^{-1}\overline{X}A^\top)^{-1}A\overline{S}^{-1}\widetilde{\delta}_\mu.
\end{aligned}$$

Recall we define $\overline{P}$ as Equation (13), then we have

$$\widetilde{\delta}_s = \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}} \cdot \sqrt{\frac{\overline{X}}{\overline{S}}}A^\top\left(A\frac{\overline{X}}{\overline{S}}A^\top\right)^{-1}\sqrt{\frac{\overline{X}}{\overline{S}}} \cdot \frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu = \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}}\overline{P}\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu$$

and

$$\widetilde{\delta}_x = \overline{S}^{-1}\widetilde{\delta}_\mu - \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}} \cdot \sqrt{\frac{\overline{X}}{\overline{S}}}A^\top\left(A\frac{\overline{X}}{\overline{S}}A^\top\right)^{-1}\sqrt{\frac{\overline{X}}{\overline{S}}} \cdot \frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu = \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}}(I - \overline{P})\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu,$$

which are matching Equations (11) and (12).

To see why the StochasticStep outputs $\widetilde{\delta}_x, \widetilde{\delta}_s$ satisfying Equations (11) and (12), we note that

$$p_\mu = \sqrt{\widetilde{V}}A^\top\left(A\frac{\overline{X}}{\overline{S}}A^\top\right)^{-1}A\sqrt{\widetilde{V}}\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu = \overline{P}\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu$$

because of Theorem 5.1. □

Using the explicit formula, we are ready to bound all quantities we needed in the following two subsubsections.

4.2.1 *Bounding $\widetilde{\delta}_s$, $\widetilde{\delta}_x$ and $\widetilde{\delta}_\mu$.*

Lemma 4.3. *Under the Assumption 4.1, the two vectors $\widetilde{\delta}_x$ and $\widetilde{\delta}_s$ found by StochasticStep satisfy:*

1. $\|\mathbf{E}[\overline{s}^{-1}\widetilde{\delta}_s]\|_2 \le 2\epsilon, \|\mathbf{E}[\overline{x}^{-1}\widetilde{\delta}_x]\|_2 \le 2\epsilon, \|\mathbf{E}[s^{-1}\widetilde{\delta}_s]\|_2 \le 2\epsilon, \|\mathbf{E}[x^{-1}\widetilde{\delta}_x]\|_2 \le 2\epsilon, \|\mathbf{E}[\mu^{-1}\widetilde{\delta}_\mu]\|_2 \le 4\epsilon.$
2. $\mathbf{Var}[\frac{\widetilde{\delta}_{s,i}}{\overline{s}_i}] \le \frac{2\epsilon^2}{k}, \mathbf{Var}[\frac{\widetilde{\delta}_{x,i}}{\overline{x}_i}] \le \frac{2\epsilon^2}{k}, \mathbf{Var}[\frac{\widetilde{\delta}_{s,i}}{s_i}] \le \frac{2\epsilon^2}{k}, \mathbf{Var}[\frac{\widetilde{\delta}_{x,i}}{x_i}] \le \frac{2\epsilon^2}{k}, \mathbf{Var}[\frac{\widetilde{\delta}_{\mu,i}}{\mu_i}] \le \frac{8\epsilon^2}{k}.$
3. $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \le \frac{0.01}{\log n}, \|s^{-1}\widetilde{\delta}_s\|_\infty \le \frac{0.02}{\log n}, \|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \le \frac{0.01}{\log n}, \|x^{-1}\widetilde{\delta}_x\|_\infty \le \frac{0.02}{\log n}, \|\mu^{-1}\widetilde{\delta}_\mu\|_\infty \le \frac{0.02}{\log n}.$

Remark 4.4. For notational simplicity, the **E** and **Var** in the proof are for the case without resampling (Line 12). Since the all the additional terms due to resampling are polynomially bounded and since we can set failure probability to an arbitrarily small inverse polynomial (see Claim 4.7), if we took into account the extra variance from resampling, then the proof does not change and the result remains the same.

Proof.

Claim 4.5 (Part 1, Bounding the ℓ_2 Norm of Expectation)

$$\|\mathbf{E}[\overline{s}^{-1}\widetilde{\delta}_s]\|_2 \le 2\epsilon, \|\mathbf{E}[\overline{x}^{-1}\widetilde{\delta}_x]\|_2 \le 2\epsilon, \|\mathbf{E}[s^{-1}\widetilde{\delta}_s]\|_2 \le 2\epsilon, \|\mathbf{E}[x^{-1}\widetilde{\delta}_x]\|_2 \le 2\epsilon, \|\mathbf{E}[\mu^{-1}\widetilde{\delta}_\mu]\|_2 \le 4\epsilon.$$

Proof. For $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty$, we consider the ith coordinate of the vector

$$\overline{s}_i^{-1}\widetilde{\delta}_{s,i} = \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\sum_{j=1}^{n}\overline{P}_{i,j}\frac{\widetilde{\delta}_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}.$$

Then, we have

$$\mathbf{E}\left[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}\right] = \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\sum_{j=1}^{n}\overline{P}_{i,j}\frac{\mathbf{E}[\widetilde{\delta}_{\mu,j}]}{\sqrt{\overline{x}_j\overline{s}_j}} = \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\sum_{j=1}^{n}\overline{P}_{i,j}\frac{\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}.$$

Since $xs \approx_{0.1} t$ and $\|\delta_\mu\|_2 \le \epsilon t$, we have $\|\frac{\delta_\mu}{\sqrt{xs}}\|_2 \le \frac{1.1\epsilon t}{\sqrt{t}}$. Since $\overline{P}$ is an orthogonal projection matrix, we have $\|\overline{P}\frac{\delta_\mu}{\sqrt{\overline{xs}}}\|_2 \le \|\frac{\delta_\mu}{\sqrt{\overline{xs}}}\|_2$. Putting all the above facts and $xs = \overline{xs}$, we can show

$$\begin{aligned}
\left\|\mathbf{E}[\overline{s}^{-1}\widetilde{\delta}_s]\right\|_2^2 &= \sum_{i=1}^{n}\left(\frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\sum_{j=1}^{n}\overline{P}_{i,j}\frac{\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}\right)^2 = \sum_{i=1}^{n}\frac{1}{\overline{x}_i\overline{s}_i}\left(\sum_{j=1}^{n}\overline{P}_{i,j}\frac{\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}\right)^2 \\
&\le \frac{1}{0.9t}\sum_{i=1}^{n}\left(\sum_{j=1}^{n}\overline{P}_{i,j}\frac{\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}\right)^2 = \frac{1}{0.9t}\left\|\overline{P}\frac{\delta_\mu}{\sqrt{\overline{xs}}}\right\|_2^2 \\
&\le \frac{1}{0.9t}\left\|\frac{\delta_\mu}{\sqrt{\overline{xs}}}\right\|_2^2 \le \frac{(1.1)^2}{0.9t}\cdot\frac{(\epsilon t)^2}{t} \le 1.4\epsilon^2,
\end{aligned}$$

which implies that

$$\left\|\mathbf{E}[\overline{s}^{-1}\widetilde{\delta}_s]\right\|_2 \le 1.2\epsilon. \tag{14}$$

Notice that the proof for x is identical to the proof for s, because $(I - \overline{P})$ is also a projection matrix. Since $\overline{s} \approx_{0.1} s$ and $\overline{x} \approx_{0.1} x$, then we can also prove the next two inequalities in the Claim statement.

Now, we are ready to bound $\|\mathbf{E}[\mu^{-1}\widetilde{\delta}_\mu]\|_2$,

$$\|\mathbf{E}[\mu^{-1}\widetilde{\delta}_\mu]\|_2 = \|\mathbf{E}[\overline{s}^{-1}\overline{x}^{-1}(\overline{x}\widetilde{\delta}_s + \overline{s}\widetilde{\delta}_x)]\|_2 \le \|\mathbf{E}[\overline{s}^{-1}\widetilde{\delta}_s]\|_2 + \|\mathbf{E}[\overline{x}^{-1}\widetilde{\delta}_x]\|_2 \le 4\epsilon,$$

by using $\mu = xs = \overline{xs}$ and $\overline{x}\widetilde{\delta}_s + \overline{s}\widetilde{\delta}_x = \widetilde{\delta}_\mu$ from Equation (7). □

Claim 4.6 (Part 2, Bounding the Variance per Coordinate).

$$\mathbf{Var}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}] \le \frac{2\epsilon^2}{k}, \mathbf{Var}[\overline{x}_i^{-1}\widetilde{\delta}_{x,i}] \le \frac{2\epsilon^2}{k}, \mathbf{Var}[s_i^{-1}\widetilde{\delta}_{s,i}] \le \frac{2\epsilon^2}{k}, \mathbf{Var}[x_i^{-1}\widetilde{\delta}_{x,i}] \le \frac{2\epsilon^2}{k}, \mathbf{Var}[\mu_i^{-1}\widetilde{\delta}_{\mu,i}] \le \frac{8\epsilon^2}{k}.$$

Proof. Consider the ith coordinate of the vector

$$\overline{s}_i^{-1}\widetilde{\delta}_{s,i} = \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\sum_{j=1}^{n}\overline{P}_{i,j}\frac{\widetilde{\delta}_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}.$$

For variance of $\overline{s}_i^{-1}\widetilde{\delta}_{s,i}$, we have

$$\begin{aligned}
\mathbf{Var}\left[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}\right] &= \frac{1}{\overline{x}_i\overline{s}_i}\sum_{j=1}^{n}\frac{\overline{P}_{i,j}^2}{\overline{x}_j\overline{s}_j}\mathbf{Var}[\widetilde{\delta}_{\mu,j}] && \text{by all } \widetilde{\delta}_{\mu,j} \text{ are independent}\\
&\le \frac{1}{\overline{x}_i\overline{s}_i}\sum_{j=1}^{n}\frac{\overline{P}_{i,j}^2}{\overline{x}_j\overline{s}_j}\frac{1}{k}\frac{\delta_{\mu,j}^2}{\frac{\delta_{\mu,j}^2}{\sum_{l=1}^{n}\delta_{\mu,l}^2}+\frac{1}{n}} && \text{by Equation (6)}\\
&\le \frac{1}{\overline{x}_i\overline{s}_i}\sum_{j=1}^{n}\frac{\overline{P}_{i,j}^2}{\overline{x}_j\overline{s}_j}\frac{1}{k}\sum_{l=1}^{n}\delta_{\mu,l}^2\\
&\le \frac{1.3}{t^2}\sum_{j=1}^{n}\overline{P}_{i,j}^2\frac{1}{k}\sum_{l=1}^{n}\delta_{\mu,l}^2 \le \frac{1.3\epsilon^2}{k}, && \text{by } \overline{x}_i\overline{s}_i = x_is_i \approx_{1/10} t,
\end{aligned}$$

where we used that $\sum_{j=1}^{n}\overline{P}_{i,j}^2 = \overline{P}_{i,i} \le 1$, $\|\delta_\mu\|_2 \le \epsilon t$ at the end.

The proof for the other three inequalities in the Claim statement are identical to this one. We omit here.

For the variance of $\mu_i^{-1}\widetilde{\delta}_{\mu,i}$,

$$\begin{aligned}
\mathbf{Var}\left[\mu_i^{-1}\widetilde{\delta}_{\mu,i}\right] &= \mathbf{Var}\left[\overline{x}_i^{-1}\overline{s}_i^{-1}(\overline{x}_i\widetilde{\delta}_{s,i}+\overline{s}_i\widetilde{\delta}_{x,i})\right]\\
&\le 2\,\mathbf{Var}\left[\overline{x}_i^{-1}\overline{x}_i\overline{s}_i^{-1}\widetilde{\delta}_{s,i}\right] + 2\,\mathbf{Var}\left[\overline{s}_i^{-1}\overline{s}_i\overline{x}_i^{-1}\widetilde{\delta}_{x,i}\right]\\
&= 2\,\mathbf{Var}\left[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}\right] + 2\,\mathbf{Var}\left[\overline{x}_i^{-1}\widetilde{\delta}_{x,i}\right] \le 8\epsilon^2/k,
\end{aligned}$$

where we used the definition $\mu = xs = \overline{xs}$ and Equation (7) in the first step, the triangle inequality in the second step and, $\mathbf{Var}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}], \mathbf{Var}[\overline{x}_i^{-1}\widetilde{\delta}_{x,i}] \le 2\epsilon^2/k$ at the end. □

Claim 4.7 (Part 3, Bounding the Probability of Success). *Without resampling, the following holds with probability* $1 - 2n\exp(-\frac{0.003k}{\epsilon\sqrt{n}\log n})$*:*

$$\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \le \frac{0.01}{\log n}, \|s^{-1}\widetilde{\delta}_s\|_\infty \le \frac{0.02}{\log n}, \|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \le \frac{0.01}{\log n}, \|x^{-1}\widetilde{\delta}_x\|_\infty \le \frac{0.02}{\log n}, \|\mu^{-1}\widetilde{\delta}_\mu\|_\infty \le \frac{0.02}{\log n}.$$

With resampling, it always holds.

Proof. We can write $\overline{s}_i^{-1}\widetilde{\delta}_{s,i} - \mathbf{E}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}] = \sum_j Y_j$ where Y_j are independent random variables defined by

$$Y_j = \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\overline{P}_{i,j}\frac{\widetilde{\delta}_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}} - \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\overline{P}_{i,j}\frac{\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}.$$

We bound the sum using Bernstein inequality. Note that Y_j are mean 0 and that Claim 4.6 shows that $\sum_{j=1}^{n} \mathbf{E}[Y_j^2] = \mathbf{Var}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}] \le \frac{2\epsilon^2}{k}$. We also need to give an upper bound for Y_j

$$
\begin{aligned}
|Y_j| &= \left| \frac{1}{\sqrt{\overline{x}_i \overline{s}_i}} \overline{P}_{i,j} \left(\frac{\widetilde{\delta}_{\mu,j} - \delta_{\mu,j}}{\sqrt{\overline{x}_j \overline{s}_j}} \right) \right| \\
&\le \frac{1.2}{t} |\widetilde{\delta}_{\mu,j} - \delta_{\mu,j}| && \text{by } |\overline{P}_{i,j}| \le 1, x_i s_i \approx_{1/10} t \\
&\le \frac{1.2}{t} |\delta_{\mu,j}/p_j| && \text{by } \widetilde{\delta}_{\mu,j} \in [0, \delta_{\mu,j}/p_j] \\
&= \frac{1.2}{t} \frac{1}{k} \frac{1}{(\frac{\delta_{\mu,i}}{\sum_{l=1}^{n} \delta_{\mu,l}^2} + \frac{1}{n\delta_{\mu,i}})} && \text{by Equation (6)} \\
&\le \frac{0.6}{t} \frac{1}{k} \left(n \sum_{l=1}^{n} \delta_{\mu,l}^2 \right)^{1/2} && \text{by } a^2 + b^2 \ge 2ab \\
&\le \frac{0.6\epsilon\sqrt{n}}{k} \stackrel{\text{def}}{=} M && \text{by } \|\delta_\mu\|_2 \le \epsilon t.
\end{aligned}
$$

Now, we can apply Bernstein inequality

$$
\begin{aligned}
\Pr\left[\left| \sum_{j=1}^{n} Y_j \right| > b \right] &\le 2 \exp\left(- \frac{b^2/2}{\sum_{j=1}^{n} \mathbf{E}[Y_j^2] + Mb/3} \right) \\
&\le 2 \exp\left(- \frac{b^2/2}{2\epsilon^2/k + (0.6\epsilon\sqrt{n}/k) \cdot b/3} \right).
\end{aligned}
$$

We choose $b = \frac{0.005}{\log n}$ and use $\epsilon \le \frac{1}{400 \log n}$ and $n \ge 10$ to get

$$
\Pr\left[\left| \sum_{j=1}^{n} Y_j \right| \ge \frac{0.05}{\log n} \right] \le 2 \exp\left(- \frac{0.003k}{\epsilon\sqrt{n} \log n} \right).
$$

Since $\|\mathbf{E}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}]\|_2 \le 2\epsilon \le \frac{0.005}{\log n}$, we have that $|\overline{s}_i^{-1}\widetilde{\delta}_{s,i}| \le \frac{0.01}{\log n}$ with probability $1 - 2\exp(-\frac{0.003k}{\epsilon\sqrt{n}\log n})$. Taking a union bound, we have that $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \le \frac{0.01}{\log n}$ with probability $1 - 2n\exp(-\frac{0.003k}{\epsilon\sqrt{n}\log n})$. Similarly, this holds for the other three terms.

Now, the last term follows by the calculation

$$
|\mu_i^{-1}\widetilde{\delta}_{\mu,i}| = |\overline{x}_i^{-1}\overline{s}_i^{-1}(\overline{x}_i\widetilde{\delta}_{s,i} + \overline{s}_i\widetilde{\delta}_{x,i})| = |\overline{s}_i^{-1}\widetilde{\delta}_{s,i}| + |\overline{x}_i^{-1}\widetilde{\delta}_{x,i}| \le \frac{0.02}{\log n}. \qquad \square
$$

$\square$

4.2.2 *Bounding $\mu^{\text{new}} - \mu$.*

Lemma 4.8. *Under the Assumption 4.1, the vector $\mu_i^{\text{new}} \stackrel{\text{def}}{=} (x_i + \widetilde{\delta}_{x,i})(s_i + \widetilde{\delta}_{s,i})$ satisfies*

1. $\|\mathbf{E}[\mu^{-1}(\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu)]\|_2 \le 10\epsilon_{\text{mp}} \cdot \epsilon$ *and* $\|\mathbf{E}[\mu^{-1}(\mu^{\text{new}} - \mu)]\|_2 \le 5\epsilon$.
2. $\mathbf{Var}[\mu_i^{-1}\mu_i^{\text{new}}] \le 50\epsilon^2/k$ *for all* i.
3. $\|\mu^{-1}(\mu^{\text{new}} - \mu)\|_\infty \le \frac{0.021}{\log n}$.

Claim 4.9 (Part 1 of Lemma 4.8).

$$
\|\mathbf{E}[\mu^{-1}(\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu)]\|_2 \le 10\epsilon_{\text{mp}} \cdot \epsilon, \text{ and } \|\mathbf{E}[\mu^{-1}(\mu^{\text{new}} - \mu)]\|_2 \le 5\epsilon.
$$

Proof.

$$\mu^{\text{new}} = (x+\widetilde{\delta}_x)(s+\widetilde{\delta}_s) = \mu + x\widetilde{\delta}_s + s\widetilde{\delta}_x + \widetilde{\delta}_x\widetilde{\delta}_s = \mu + \underbrace{\overline{x}\widetilde{\delta}_s + \overline{s}\widetilde{\delta}_x}_{\widetilde{\delta}_\mu} + \underbrace{(x-\overline{x})\widetilde{\delta}_s + (s-\overline{s})\widetilde{\delta}_x + \widetilde{\delta}_x\widetilde{\delta}_s}_{\epsilon_\mu}.$$

Taking the expectation on both sides, we have

$$\mathbf{E}[\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu] = (x-\overline{x})\mathbf{E}[\widetilde{\delta}_s] + (s-\overline{s})\mathbf{E}[\widetilde{\delta}_x] + \mathbf{E}[\widetilde{\delta}_x\widetilde{\delta}_s].$$

Hence, we have that

$$\begin{aligned}
&\|\mu^{-1}\mathbf{E}[\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu]\|_2 \\
\le\ &\|\mu^{-1}(x-\overline{x})s\cdot s^{-1}\mathbf{E}[\widetilde{\delta}_s]\|_2 + \|\mu^{-1}(s-\overline{s})x\cdot x^{-1}\mathbf{E}[\widetilde{\delta}_x]\|_2 + \|\mu^{-1}\mathbf{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2 \\
\le\ &\|\mu^{-1}(x-\overline{x})s\|_\infty\cdot\|s^{-1}\mathbf{E}[\widetilde{\delta}_s]\|_2 + \|\mu^{-1}(s-\overline{s})x\|_\infty\cdot\|x^{-1}\mathbf{E}[\widetilde{\delta}_x]\|_2 + \|\mu^{-1}\mathbf{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2 \\
\le\ &\epsilon_{\text{mp}}\cdot\|s^{-1}\mathbf{E}[\widetilde{\delta}_s]\|_2 + \epsilon_{\text{mp}}\cdot\|x^{-1}\mathbf{E}[\widetilde{\delta}_x]\|_2 + \|\mu^{-1}\mathbf{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2 \\
\le\ &4\epsilon_{\text{mp}}\cdot\epsilon + \|\mu^{-1}\mathbf{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2, \qquad (15)
\end{aligned}$$

where we used the triangle inequality in the first step, $\|ab\|_2 \le \|a\|_\infty\cdot\|b\|_2$ in the second step, $\|\mu^{-1}(x-\overline{x})s\|_\infty \le \epsilon_{\text{mp}}$ and $\|\mu^{-1}(s-\overline{s})x\|_\infty \le \epsilon_{\text{mp}}$ (since $\overline{x} \approx_{\epsilon_{\text{mp}}} x$, $\overline{s} \approx_{\epsilon_{\text{mp}}} s$) in the third step, and $\|\mathbf{E}[s^{-1}\widetilde{\delta}_s]\|_2 \le 2\epsilon$ and $\|\mathbf{E}[x^{-1}\widetilde{\delta}_x]\|_2 \le 2\epsilon$ (Part 1 of Lemma 4.3) at the end.

To bound the last term, using $\mathbf{E}[\widetilde{\delta}_s] = \delta_s$ and $\mathbf{E}[\widetilde{\delta}_x] = \delta_x$, we note that

$$\mathbf{E}[\widetilde{\delta}_{x,i}\widetilde{\delta}_{s,i}] = \delta_{x,i}\delta_{s,i} + \mathbf{E}[(\widetilde{\delta}_{x,i}-\delta_{x,i})(\widetilde{\delta}_{s,i}-\delta_{s,i})].$$

Hence, we have

$$\begin{aligned}
\|\mu^{-1}\mathbf{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2 &\le \|\mu^{-1}\delta_x\delta_s\|_2 + \left(\sum_{i=1}^n\left(\mathbf{E}\left[x_i^{-1}(\widetilde{\delta}_{x,i}-\delta_{x,i})\cdot s_i^{-1}(\widetilde{\delta}_{s,i}-\delta_{s,i})\right]\right)^2\right)^{1/2} \\
&\le 4\epsilon^2 + \frac{1}{2}\left(\sum_{i=1}^n\left(\mathbf{Var}[x_i^{-1}\widetilde{\delta}_{x,i}] + \mathbf{Var}[s_i^{-1}\widetilde{\delta}_{s,i}]\right)^2\right)^{1/2} \\
&\le 4\epsilon^2 + \frac{1}{2}\left(\sum_{i=1}^n 2(\mathbf{Var}[x_i^{-1}\widetilde{\delta}_{x,i}])^2 + 2(\mathbf{Var}[s_i^{-1}\widetilde{\delta}_{s,i}])^2\right)^{1/2} \\
&\le 4\epsilon^2 + 2\sqrt{n\cdot\epsilon^4/k^2} \le 4\epsilon^2 + 2\epsilon\cdot\epsilon_{\text{mp}} \le 6\epsilon\cdot\epsilon_{\text{mp}}, \qquad (16)
\end{aligned}$$

where we used $\|\mu^{-1}\delta_x\delta_s\|_2 \le \|x^{-1}\delta_x\|_2\cdot\|s^{-1}\delta_s\|_2 \le 4\epsilon^2$ (Part 1 of Lemma 4.3) and $2ab \le a^2+b^2$ in the second step, $(a+b)^2 \le 2a^2+2b^2$ in the third step, $\mathbf{Var}[x_i^{-1}\widetilde{\delta}_{x,i}] \le 2\epsilon^2/k$ and $\mathbf{Var}[s_i^{-1}\widetilde{\delta}_{s,i}] \le 2\epsilon^2/k$ (Part 2 of Lemma 4.3) in the fourth step, and $k \ge \frac{\epsilon\sqrt{n}}{\epsilon_{\text{mp}}}$ at the end.

Combining Equations (15) and (16), we have that

$$\|\mu^{-1}(\mathbf{E}[\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu])\|_2 \le 4\epsilon_{\text{mp}}\cdot\epsilon + \|\mu^{-1}\mathbf{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2 \le 10\epsilon_{\text{mp}}\cdot\epsilon,$$

where we used $\epsilon \le \epsilon_{\text{mp}}$.

From Part 1 of Lemma 4.3, we know that $\|\mu^{-1}\mathbf{E}[\widetilde{\delta}_\mu]\|_2 \le 4\epsilon$. Thus, using triangle inequality, we know

$$\|\mu^{-1}(\mathbf{E}[\mu^{\text{new}} - \mu])\|_2 \le 10\epsilon_{\text{mp}}\cdot\epsilon + 4\epsilon \le 5\epsilon. \qquad \square$$

Claim 4.10 (Part 2 of Lemma 4.8). $\mathbf{Var}[\mu_i^{-1}\mu_i^{\text{new}}] \le 50\epsilon^2/k$ *for all* i.

Proof. Recall that

$$\mu^{\text{new}} = \mu + \widetilde{\delta}_\mu + (x - \overline{x})\widetilde{\delta}_s + (s - \overline{s})\widetilde{\delta}_x + \widetilde{\delta}_x\widetilde{\delta}_s.$$

We can upper bound the variance of $\mu_i^{-1}\mu_i^{\text{new}}$,

$$\begin{aligned}
\mathbf{Var}[\mu_i^{-1}\mu_i^{\text{new}}] &\le 4\,\mathbf{Var}[\mu_i^{-1}\widetilde{\delta}_{\mu,i}] + 4\,\mathbf{Var}[\mu_i^{-1}(x_i - \overline{x}_i)\widetilde{\delta}_{s,i}] + 4\,\mathbf{Var}[\mu_i^{-1}(s_i - \overline{s}_i)\widetilde{\delta}_{x,i}] \\
&\quad + 4\,\mathbf{Var}[\mu_i^{-1}\widetilde{\delta}_{x,i}\widetilde{\delta}_{s,i}] \\
&\le 32\frac{\epsilon^2}{k} + 4\frac{\epsilon^2}{k} + 4\frac{\epsilon^2}{k} + \mathbf{Var}[\mu_i^{-1}\widetilde{\delta}_{x,i}\widetilde{\delta}_{s,i}] \\
&= 40\frac{\epsilon^2}{k} + \mathbf{Var}[x_i^{-1}\widetilde{\delta}_{x,i} \cdot s_i^{-1}\widetilde{\delta}_{s,i}] \\
&\le 40\frac{\epsilon^2}{k} + 2\,\mathbf{Sup}[(x_i^{-1}\widetilde{\delta}_{x,i})^2] \cdot \mathbf{Var}[s_i^{-1}\widetilde{\delta}_{s,i}] + 2\,\mathbf{Sup}[(s_i^{-1}\widetilde{\delta}_{s,i})^2] \cdot \mathbf{Var}[x_i^{-1}\widetilde{\delta}_{x,i}] \\
&\le 40\frac{\epsilon^2}{k} + 2 \cdot \left(\frac{0.02}{\log n}\right)^2 \cdot \frac{\epsilon^2}{k} + 2 \cdot \left(\frac{0.02}{\log n}\right)^2 \cdot \frac{\epsilon^2}{k} \le 50\frac{\epsilon^2}{k},
\end{aligned}$$

where the second step follows by the inequality $\mathbf{Var}[\mu_i^{-1}\widetilde{\delta}_{\mu,i}] \le 8\epsilon^2/k$ (Part 2 of Lemma 4.3),

$$\mathbf{Var}[\mu_i^{-1}(x_i - \overline{x}_i)\widetilde{\delta}_{s,i}] = \mathbf{Var}[x_i^{-1}(x_i - \overline{x}_i)s_i^{-1}\widetilde{\delta}_{s,i}] \le 2\epsilon_{\text{mp}}^2\mathbf{Var}[s_i^{-1}\widetilde{\delta}_{s,i}] \le \epsilon^2/k.$$

and a similar inequality $\mathbf{Var}[\mu_i^{-1}(s_i - \overline{s}_i)\widetilde{\delta}_{x,i}] \le \epsilon^2/k$, the third step follows by the definition $\mu = xs$, the fourth step follows by the inequality $\mathbf{Var}[xy] \le 2\,\mathbf{Sup}[x^2]\mathbf{Var}[y] + 2\,\mathbf{Sup}[y^2]\mathbf{Var}[x]$ (Lemma A.1) with **Sup** denoting the deterministic maximum of the random variable, the fifth step follows by the inequalities $\mathbf{Var}[s_i^{-1}\widetilde{\delta}_{s,i}] \le 2\epsilon^2/k$ and $\mathbf{Var}[x_i^{-1}\widetilde{\delta}_{x,i}] \le 2\epsilon^2/k$ (Part 2 of Lemma 4.3). □

Claim 4.11 (Part 3 of Lemma 4.8). $\|\mu^{-1}(\mu^{\text{new}} - \mu)\|_\infty \le \frac{0.021}{\log n}$.

Proof. We again note that

$$\mu^{\text{new}} = \mu + \widetilde{\delta}_\mu + (x - \overline{x})\widetilde{\delta}_s + (s - \overline{s})\widetilde{\delta}_x + \widetilde{\delta}_x\widetilde{\delta}_s.$$

Hence, we have

$$\begin{aligned}
&|\mu_i^{-1}(\mu_i^{\text{new}} - \mu_i - \widetilde{\delta}_{\mu,i})| \\
&\le |(x - \overline{x})_i\mu_i^{-1}\widetilde{\delta}_{s,i}| + |(s - \overline{s})_i\mu_i^{-1}\widetilde{\delta}_{x,i}| + |\mu_i^{-1}\widetilde{\delta}_{x,i}\widetilde{\delta}_{s,i}| \\
&= |(x - \overline{x})_i x_i^{-1}| \cdot |s_i^{-1}\widetilde{\delta}_{s,i}| + |(s - \overline{s})_i s_i^{-1}| \cdot |x_i^{-1}\widetilde{\delta}_{x,i}| + |x_i^{-1}\widetilde{\delta}_{x,i}| \cdot |s_i^{-1}\widetilde{\delta}_{s,i}| \\
&\le \epsilon_{\text{mp}}|s_i^{-1}\widetilde{\delta}_{s,i}| + \epsilon_{\text{mp}}|x_i^{-1}\widetilde{\delta}_{x,i}| + |s_i^{-1}\widetilde{\delta}_{s,i}||x_i^{-1}\widetilde{\delta}_{x,i}| \\
&\le \epsilon_{\text{mp}} \cdot \frac{0.2}{\log n} + \epsilon_{\text{mp}} \cdot \frac{0.02}{\log n} + \left(\frac{0.02}{\log n}\right)^2 \le \frac{1}{1{,}000\log n},
\end{aligned}$$

where the first step follows by the triangle inequality, the second step follows by the definition $\mu_i = x_i s_i$, the third step follows by the invariants $x \approx_{\epsilon_{\text{mp}}} \overline{x}$ and $s \approx_{\epsilon_{\text{mp}}} \overline{s}$, the fifth step follows by the inequalities $|s_i^{-1}\widetilde{\delta}_{s,i}| \le \frac{0.02}{\log n}$ and $|x_i^{-1}\widetilde{\delta}_{x,i}| \le \frac{0.02}{\log n}$ (Part 3 of Lemma 4.3).

Since we know that $|\mu_i^{-1}\widetilde{\delta}_{\mu,i}| \le \frac{0.02}{\log n}$ (Part 3 of Lemma 4.3), we have

$$|\mu_i^{-1}(\mu_i^{\text{new}} - \mu_i)| \le \frac{1}{1000\log n} + \frac{0.02}{\log n} \le \frac{0.021}{\log n}.$$

□

4.3 Stochastic Central Path

Now, we are ready to prove $x_i s_i \approx_{0.1} t$ during the whole algorithm. As explained in the proof outline (see Section 4.1), we will prove this bound by analyzing the potential $\Phi_\lambda(\mu/t - 1)$, where $\Phi_\lambda(r) = \sum_{i=1}^n \cosh(\lambda r_i)$.

First, we give some basic properties of Φ_λ.

Lemma 4.12 (Basic Properties of Potential Function). *Let $\Phi_\lambda(r) = \sum_{i=1}^n \cosh(\lambda r_i)$ for some $\lambda > 0$. For any vector $r \in \mathbb{R}^n$,*

1. For any vector $\|v\|_\infty \le 1/\lambda$, we have that

$$\Phi_\lambda(r+v) \le \Phi_\lambda(r) + \langle \nabla\Phi_\lambda(r), v\rangle + 2\|v\|^2_{\nabla^2\Phi_\lambda(r)}.$$

2. $\|\nabla\Phi_\lambda(r)\|_2 \ge \frac{\lambda}{\sqrt{n}}(\Phi_\lambda(r) - n)$.

3. $(\sum_{i=1}^n \lambda^2 \cosh^2(\lambda r_i))^{1/2} \le \lambda\sqrt{n} + \|\nabla\Phi_\lambda(r)\|_2$.

Proof. For each $i \in [n]$, we use r_i to denote the ith coordinate of vector r.

Proof of Part 1. Using mean-value forms of Taylor's theorem, we have that

$$\cosh(\lambda(r_i + v_i)) = \cosh(\lambda r_i) + \lambda \sinh(\lambda r_i) v_i + \frac{\lambda^2}{2}\cosh(\zeta_i) v_i^2,$$

where ζ_i is between λr_i and $\lambda(r_i + v_i)$. By definition of cosh and the assumption that $\|v\|_\infty \le \frac{1}{2\lambda}$, we have that

$$\cosh(\zeta_i) = \frac{1}{2}\exp(\zeta_i) + \frac{1}{2}\exp(-\zeta_i) \le \exp(1)\cdot\frac{1}{2}(\exp(\lambda r_i) + \exp(-\lambda r_i)) \le 3\cosh(\lambda r_i).$$

Hence, we have

$$\cosh(\lambda(r_i + v_i)) \le \cosh(\lambda r_i) + \lambda\sinh(\lambda r_i) v_i + 2\lambda^2\cosh(\lambda r_i) v_i^2.$$

Summing over all the coordinates gives

$$\begin{aligned}
&\sum_{i=1}^n \cosh(\lambda(r_i + v_i)) \le \sum_{i=1}^n \left[\cosh(\lambda r_i) + 2\lambda\sinh(\lambda r_i) v_i + \lambda^2\cosh(\lambda r_i) v_i^2\right] \\
\Rightarrow\ &\Phi_\lambda(r+v) \le \Phi_\lambda(r) + \langle\nabla\Phi_\lambda(r), v\rangle + 2\|v\|^2_{\nabla^2\Phi_\lambda(r)}.
\end{aligned}$$

Proof of Part 2. Since $\Phi_\lambda(r) = \sum_{i=1}^n \cosh(\lambda r_i)$, then

$$\nabla\Phi_\lambda(r) = \begin{bmatrix}\lambda\sinh(\lambda r_1) & \lambda\sinh(\lambda r_2) & \cdots & \lambda\sinh(\lambda r_n)\end{bmatrix}^\top.$$

Thus, we can lower bound $\|\nabla\Phi_\lambda(r)\|_2$ in the following way:

$$\begin{aligned}
\|\nabla\Phi_\lambda(r)\|_2 &= \left(\sum_{i=1}^n \lambda^2\sinh^2(\lambda r_i)\right)^{1/2} \\
&= \left(\sum_{i=1}^n \lambda^2(\cosh^2(\lambda r_i) - 1)\right)^{1/2} && \text{by } \cosh^2(y) - \sinh^2(y) = 1, \forall y \\
&\ge \frac{\lambda}{\sqrt{n}}\sum_{i=1}^n \sqrt{\cosh^2(\lambda r_i) - 1} && \text{by } \|\cdot\|_2 \ge \frac{1}{\sqrt{n}}\|\cdot\|_1 \\
&\ge \frac{\lambda}{\sqrt{n}}\sum_{i=1}^n (\cosh(\lambda r_i) - 1) && \text{by } \cosh(\lambda r_i) \ge 1 \\
&= \frac{\lambda}{\sqrt{n}}(\Phi_\lambda(r) - n) && \text{by def of } \Phi(r).
\end{aligned}$$

Proof of Part 3.

$$\left(\sum_{i=1}^{n}\lambda^2\cosh^2(\lambda r_i)\right)^{1/2} = \left(\sum_{i=1}^{n}\lambda^2+\lambda^2\sinh^2(\lambda r_i)\right)^{1/2} \qquad \text{by } \cosh^2(y)-\sinh^2(y)=1, \forall y$$
$$\leq (n\lambda^2)^{1/2} + \left(\sum_{i=1}^{n}\lambda^2\sinh^2(\lambda r_i)\right)^{1/2}$$
$$= \lambda\sqrt{n} + \|\nabla\Phi_\lambda(r)\|_2. \qquad \square$$

The following lemma shows that the potential Φ is decreasing in expectation when Φ is large.

Lemma 4.13. *Under the Assumption 4.1, we have*

$$\mathbf{E}\left[\Phi_\lambda\left(\frac{\mu^{\text{new}}}{t^{\text{new}}}-1\right)\right] \leq \Phi_\lambda\left(\frac{\mu}{t}-1\right) - \frac{\lambda\epsilon}{15\sqrt{n}}\left(\Phi_\lambda\left(\frac{\mu}{t}-1\right)-10n\right).$$

Proof. Let $\epsilon_\mu = \mu^{\text{new}} - \mu - \widetilde{\delta}_\mu$. From the definition, we have

$$\mu^{\text{new}} - t^{\text{new}} = \mu + \widetilde{\delta}_\mu + \epsilon_\mu - t^{\text{new}},$$

which implies

$$\begin{aligned}\frac{\mu^{\text{new}}}{t^{\text{new}}}-1 &= \frac{\mu}{t^{\text{new}}} + \frac{1}{t^{\text{new}}}(\widetilde{\delta}_\mu+\epsilon_\mu) - 1\\ &= \frac{\mu}{t}\frac{t}{t^{\text{new}}} + \frac{1}{t^{\text{new}}}(\widetilde{\delta}_\mu+\epsilon_\mu) - 1\\ &= \frac{\mu}{t} + \frac{\mu}{t}\left(\frac{t}{t^{\text{new}}}-1\right) + \frac{1}{t^{\text{new}}}(\widetilde{\delta}_\mu+\epsilon_\mu) - 1\\ &= \frac{\mu}{t} - 1 + \underbrace{\frac{\mu}{t}\left(\frac{t}{t^{\text{new}}}-1\right) + \frac{1}{t^{\text{new}}}(\widetilde{\delta}_\mu+\epsilon_\mu)}_{v}. \end{aligned} \tag{17}$$

To apply Lemma 4.12 with $r = \mu/t - 1$ and $r + v = \mu^{\text{new}}/t^{\text{new}} - 1$, we first compute the expectation of v

$$\begin{aligned}\mathbf{E}[v] &= \frac{\mu}{t}\left(\frac{t}{t^{\text{new}}}-1\right) + \frac{1}{t^{\text{new}}}(\mathbf{E}[\widetilde{\delta}_\mu] + \mathbf{E}[\epsilon_\mu])\\ &= \frac{\mu}{t}\left(\frac{t}{t^{\text{new}}}-1\right) + \frac{1}{t^{\text{new}}}(\delta_\mu + \mathbf{E}[\epsilon_\mu])\\ &= \frac{\mu}{t}\left(\frac{t}{t^{\text{new}}}-1\right) + \frac{1}{t^{\text{new}}}\left(\left(\left(\frac{t^{\text{new}}}{t}-1\right)\mu - \frac{\epsilon}{2}t^{\text{new}}\frac{\nabla\Phi_\lambda(\mu/t-1)}{\|\nabla\Phi_\lambda(\mu/t-1)\|_2}\right) + \mathbf{E}[\epsilon_\mu]\right)\\ &= -\frac{\epsilon}{2}\frac{\nabla\Phi_\lambda(\mu/t-1)}{\|\nabla\Phi_\lambda(\mu/t-1)\|_2} + \frac{1}{t^{\text{new}}}\mathbf{E}[\epsilon_\mu], \end{aligned} \tag{18}$$

where the third step follows by the definition of δ_μ.

Next, we bound the $\|v\|_\infty$ as follows:

$$\begin{aligned}\|v\|_\infty &\leq \left\|\frac{\mu}{t}\left(\frac{t}{t^{\text{new}}}-1\right)\right\|_\infty + \left\|\frac{1}{t^{\text{new}}}(\widetilde{\delta}_\mu+\epsilon_\mu)\right\|_\infty \leq \frac{\epsilon}{\sqrt{n}} + \frac{\|\mu^{-1}(\mu^{\text{new}}-\mu)\|_\infty}{0.9}\\ &\leq \frac{\epsilon}{\sqrt{n}} + \frac{0.021}{0.9\log n} \leq \frac{1}{\lambda},\end{aligned}$$

where we used Part 3 of Lemma 4.8 and $\epsilon \leq \frac{1}{400\log n}$.

Since $\|v\|_\infty \leq \frac{1}{\lambda}$, we can apply Part 1 of Lemma 4.12 and get

$$\begin{aligned}
&\mathbf{E}[\Phi_\lambda(\mu/t + v - 1)] \\
\leq\ & \Phi_\lambda(\mu/t - 1) + \langle \nabla\Phi_\lambda(\mu/t - 1), \mathbf{E}[v]\rangle + 2\mathbf{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t+v-1)}] \\
=\ & \Phi_\lambda(\mu/t - 1) - \frac{\epsilon}{2}\|\nabla\Phi_\lambda(\mu/t-1)\|_2 + \frac{t}{t^{\mathrm{new}}}\langle \nabla\Phi_\lambda(\mu/t-1), \mathbf{E}[t^{-1}\epsilon_\mu]\rangle + 2\mathbf{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}] \\
\leq\ & \Phi_\lambda(\mu/t - 1) - \frac{\epsilon}{2}\|\nabla\Phi_\lambda(\mu/t-1)\|_2 + \frac{t}{t^{\mathrm{new}}}\|\nabla\Phi_\lambda(\mu/t-1)\|_2 \cdot \|\mathbf{E}[t^{-1}\epsilon_\mu]\|_2 + 2\mathbf{E}[\|[v]\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}] \\
\leq\ & \Phi_\lambda(\mu/t - 1) - \frac{\epsilon}{2}\|\nabla\Phi_\lambda(\mu/t-1)\|_2 + 10\epsilon_{\mathrm{mp}}\cdot\epsilon\|\nabla\Phi_\lambda(\mu/t-1)\|_2 + 2\mathbf{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}],
\end{aligned}$$

where we substituted $\mathbf{E}[v]$ by Equation (18) in the second step, we used $\langle a, b\rangle \leq \|a\|_2 \cdot \|b\|_2$ in the third step, and $\|\mathbf{E}[t^{-1}\epsilon_\mu]\|_2 \leq 10\epsilon_{\mathrm{mp}}\cdot\epsilon$ (from Part 1 of Lemma 4.8 and $\mu \approx_{0.1} t$) at the end.

We still need to bound $\mathbf{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}]$. Before bounding it, we first bound $\mathbf{E}[v_i^2]$,

$$\begin{aligned}
\mathbf{E}[v_i^2] \leq\ & 2\,\mathbf{E}\left[\left(\frac{\mu_i}{t}\left(\frac{t}{t^{\mathrm{new}}} - 1\right)\right)^2\right] + 2\,\mathbf{E}\left[\left(\frac{1}{t^{\mathrm{new}}}(\widetilde{\delta}_{\mu,i} + \widehat{\delta}_{\mu,i})\right)^2\right] \\
\leq\ & \epsilon^2/n + 2.5\,\mathbf{E}\left[((\mu_i^{\mathrm{new}} - \mu_i)/\mu_i)^2\right] \\
=\ & \epsilon^2/n + 2.5\,\mathbf{Var}[(\mu_i^{\mathrm{new}} - \mu_i)/\mu_i] + 2.5(\mathbf{E}[(\mu_i^{\mathrm{new}} - \mu_i)/\mu_i])^2 \\
\leq\ & \epsilon^2/n + 125\epsilon^2/k + 2.5(\mathbf{E}[(\mu_i^{\mathrm{new}} - \mu_i)/\mu_i])^2 \\
\leq\ & 126\epsilon^2/k + 3(\mathbf{E}[(\mu_i^{\mathrm{new}} - \mu_i)/\mu_i])^2, \qquad (19)
\end{aligned}$$

where we used the definition of v (see Equation (17)) in the first step, $\mu \approx_{0.1} t$ and $(t/t^{\mathrm{new}} - 1)^2 \leq \epsilon^2/(4n)$ in the second step, $\mathbf{E}[x^2] = \mathbf{Var}[x] + (\mathbf{E}[x])^2$ in the third step, Part 2 of Lemma 4.8 in the fourth step, and $n \geq k$ at the end.

Now, we are ready to bound $\mathbf{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}]$

$$\begin{aligned}
&\mathbf{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}] \\
=\ & \lambda^2\sum_{i=1}^n \mathbf{E}[\Phi_\lambda(\mu/t-1)_i v_i^2] \\
\leq\ & \lambda^2\sum_{i=1}^n \Phi_\lambda(\mu/t-1)_i \cdot (126\epsilon^2/k + 3(\mathbf{E}[(\mu_i^{\mathrm{new}} - \mu_i)/\mu_i])^2) \\
=\ & 126\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t-1) + 3\lambda^2\sum_{i=1}^n \Phi_\lambda(\mu/t-1)_i \cdot (\mathbf{E}[(\mu_i^{\mathrm{new}} - \mu_i)/\mu_i])^2 \\
\leq\ & 126\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t-1) + 3\lambda\left(\sum_{i=1}^n \lambda^2\Phi_\lambda(\mu/t-1)_i^2\right)^{1/2} \cdot \|\mathbf{E}[\mu^{-1}(\mu^{\mathrm{new}} - \mu)]\|_4^2 \\
\leq\ & 126\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t-1) + 3\lambda\left(\lambda\sqrt{n} + \|\nabla\Phi_\lambda(\mu/t-1)\|_2\right)\cdot(5\epsilon)^2,
\end{aligned}$$

where the first step follows from the fact $\Phi_\lambda(x)_i = \cosh(\lambda x_i)$, the second step follows from Equation (19), the fourth step follows from Cauchy-Schwarz inequality, the fifth step follows from Part 3 of Lemma 4.12 and the fact that $\|\mathbf{E}[\mu^{-1}(\mu^{\mathrm{new}} - \mu)]\|_4^2 \leq \|\mathbf{E}[\mu^{-1}(\mu^{\mathrm{new}} - \mu)]\|_2^2 \leq (5\epsilon)^2$ (Lemma 4.8).

Then,

$$\begin{aligned}
&\mathbf{E}[\Phi_\lambda(\mu/t+v-1)]\\
&\le \Phi_\lambda(\mu/t-1)-\left(\frac{\epsilon}{2}-10\epsilon_{\mathrm{mp}}\cdot\epsilon\right)\|\nabla\Phi_\lambda(\mu/t-1)\|_2+252\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t-1)\\
&\quad+150\lambda^2\epsilon^2\sqrt{n}+150\lambda\epsilon^2\|\Phi_\lambda(\mu/t-1)\|_2\\
&\le \Phi_\lambda(\mu/t-1)-\frac{\epsilon}{3}\|\nabla\Phi_\lambda(\mu/t-1)\|_2+252\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t-1)+150\lambda^2\epsilon^2\sqrt{n}\\
&\le \Phi_\lambda(\mu/t-1)-\frac{\lambda\epsilon}{3\sqrt{n}}(\Phi_\lambda(\mu/t-1)-n)+252\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t-1)+150\lambda^2\epsilon^2\sqrt{n}\\
&\le \Phi_\lambda(\mu/t-1)-\frac{\lambda\epsilon}{3\sqrt{n}}(\Phi_\lambda(\mu/t-1)/5-2n),
\end{aligned}$$

where the second step follows from the inequalities $1{,}000\lambda\epsilon \le 1$ and $1{,}000\epsilon_{\mathrm{mp}} \le 1$, the third step follows from Part 2 of Lemma 4.12, and the last step follows from the inequalities $1{,}000\lambda\epsilon_{\mathrm{mp}} \le \log n$ and $k \ge \frac{\sqrt{n}\epsilon\log n}{\epsilon_{\mathrm{mp}}}$. □

As a corollary, we have the following:

Lemma 4.14. *During the Main algorithm, Assumption 4.1 is always satisfied. Furthermore, the ClassicalStep happens with probability $O(\frac{1}{n^2})$ each step.*

Proof. The second and the fourth assumptions simply follow from the choice of ϵ_{mp} and k.

Let $\Phi^{(k)}$ be the potential at the kth iteration of the Main. The ClassicalStep ensures that $\Phi^{(k)} \le n^3$ at the end of each iteration. By the definition of Φ and the choice of λ in Main, we have that

$$\left\|\frac{xs}{t}-1\right\|_\infty \le \frac{\ln(2n^3)}{\lambda} \le 0.1.$$

This proves the first assumption $xs \approx_{0.1} t$ with $t > 0$.

For the third assumption, we note that

$$\begin{aligned}
\|\delta_\mu\|_2 &= \left\|\left(\frac{t^{\mathrm{new}}}{t}-1\right)xs-\frac{\epsilon}{2}\cdot t^{\mathrm{new}}\cdot\frac{\nabla\Phi_\lambda(\mu/t-1)}{\|\nabla\Phi_\lambda(\mu/t-1)\|_2}\right\|_2\\
&\le \left|\frac{t^{\mathrm{new}}}{t}-1\right|\|xs\|_2+\frac{\epsilon}{2}t^{\mathrm{new}}\\
&\le \frac{\epsilon}{3\sqrt{n}}\cdot 1.1\sqrt{n}t+1.01\cdot\frac{\epsilon}{2}t\le\epsilon t,
\end{aligned}$$

where we used $xs \approx_{0.1} t$ and the formula of t^{new}. Hence, we proved all assumptions in Assumption 4.1.

Now, we bound the probability that ClassicalStep happens. In the beginning of the Main, Lemma A.6 is used to modify the linear program with parameter $\min(\frac{\delta}{2},\frac{1}{\lambda})$. Hence, the initial point x and s satisfies $xs \approx_{1/\lambda} 1$. Therefore, we have $\Phi^{(0)} \le 10n$. Lemma 4.13 shows $\mathbf{E}[\Phi^{(k+1)}] \le (1-\frac{\lambda\epsilon}{15\sqrt{n}})\mathbf{E}[\Phi^{(k)}]+\frac{\lambda\epsilon}{15\sqrt{n}}10n$. By induction, we have that $\mathbf{E}[\Phi^{(k)}] \le 10n$ for all k. Since the potential is positive, Markov inequality shows that for any k, $\Phi^{(k)} \ge n^3$ with probability at most $O(\frac{1}{n^2})$. □

4.4 Analysis of Cost per Iteration

To apply the data structure for projection maintenance (Theorem 5.1), we need to first prove the input vector w does not change too much for each step.

Lemma 4.15. *Let* $x^{\text{new}} = x + \widetilde{\delta}_x$ *and* $s^{\text{new}} = s + \widetilde{\delta}_s$. *Let* $w = \frac{x}{s}$ *and* $w^{\text{new}} = \frac{x^{\text{new}}}{s^{\text{new}}}$. *Then, we have*

$$\sum_{i=1}^{n} \left(\mathbf{E}[\ln w_i^{\text{new}}] - \ln w_i\right)^2 \le 64\epsilon^2, \quad \sum_{i=1}^{n} \left(\mathbf{Var}[\ln w_i^{\text{new}}]\right)^2 \le 1{,}000\epsilon^2.$$

Proof. From the definition, we know that

$$\frac{w_i^{\text{new}}}{w_i} = \frac{1}{s_i^{-1}x_i} \frac{x_i + \widetilde{\delta}_{x,i}}{s_i + \widetilde{\delta}_{s,i}} = \frac{1 + x_i^{-1}\widetilde{\delta}_{x,i}}{1 + s_i^{-1}\widetilde{\delta}_{s,i}}.$$

Part 1. For each $i \in [n]$, we have

$$\begin{aligned}
\mathbf{E}[\ln w_i^{\text{new}}] - \ln w_i &= \mathbf{E}\left[\ln(1 + x_i^{-1}\widetilde{\delta}_{x,i}) - \ln(1 + s_i^{-1}\widetilde{\delta}_{s,i})\right] \\
&\le 2|\mathbf{E}[x_i^{-1}\widetilde{\delta}_{x,i} - s_i^{-1}\widetilde{\delta}_{s,i}]| && \text{by } |s_i^{-1}\widetilde{\delta}_{s,i}|, |x_i^{-1}\widetilde{\delta}_{x,i}| \le 0.2, \text{Lemma 4.3} \\
&\le 2|\mathbf{E}[x_i^{-1}\widetilde{\delta}_{x,i}]| + 2|\mathbf{E}[s_i^{-1}\widetilde{\delta}_{s,i}]| && \text{by triangle inequality.}
\end{aligned}$$

Thus, summing over all the coordinates gives

$$\sum_{i=1}^{n} \left(\mathbf{E}[\ln w_i^{\text{new}}] - \ln w_i\right)^2 \le \sum_{i=1}^{n} 8(\mathbf{E}[x_i^{-1}\widetilde{\delta}_{x,i}])^2 + 8(\mathbf{E}[s_i^{-1}\widetilde{\delta}_{s,i}])^2 \le 64\epsilon^2,$$

where the first step follows by the triangle inequality, the last step follows by the inequalities $\|\mathbf{E}[s^{-1}\widetilde{\delta}_s]\|_2^2, \|\mathbf{E}[x^{-1}\widetilde{\delta}_x]\|_2^2 \le 4\epsilon^2$ (Part 1 of Lemma 4.3).

Part 2. For each $i \in [n]$, we have

$$\begin{aligned}
\mathbf{Var}[w_i^{\text{new}}] &\le \mathbf{E}\left[\left(\ln w_i^{\text{new}} - \ln w_i\right)^2\right] \\
&= \mathbf{E}\left[\left(\ln \frac{1 + x_i^{-1}\widetilde{\delta}_{x,i}}{1 + s_i^{-1}\widetilde{\delta}_{s,i}}\right)^2\right] \\
&\le 2\,\mathbf{E}[(x_i^{-1}\widetilde{\delta}_{x,i} - s_i^{-1}\widetilde{\delta}_{s,i})^2] \\
&\le 2\,\mathbf{E}[2(x_i^{-1}\widetilde{\delta}_{x,i})^2 + 2(s_i^{-1}\widetilde{\delta}_{s,i})^2] \\
&= 4\,\mathbf{E}[(x_i^{-1}\widetilde{\delta}_{x,i})^2] + 4\,\mathbf{E}[(s_i^{-1}\widetilde{\delta}_{s,i})^2] \\
&= 4\,\mathbf{Var}[x_i^{-1}\widetilde{\delta}_{x,i}] + 4(\mathbf{E}[x_i^{-1}\widetilde{\delta}_{x,i}])^2 + 4\,\mathbf{Var}[s_i^{-1}\widetilde{\delta}_{s,i}] + 4(\mathbf{E}[s_i^{-1}\widetilde{\delta}_{s,i}])^2 \\
&\le 16\epsilon^2/k + 4(\mathbf{E}[x_i^{-1}\widetilde{\delta}_{x,i}])^2 + 4(\mathbf{E}[s_i^{-1}\widetilde{\delta}_{s,i}])^2,
\end{aligned}$$

where we used $\mathbf{Var}[x_i^{-1}\widetilde{\delta}_{x,i}], \mathbf{Var}[s_i^{-1}\widetilde{\delta}_{s,i}] \le 2\epsilon^2/k$ (Part 2 of Lemma 4.3) at the end.

Thus, summing over all the coordinates

$$\begin{aligned}
\sum_{i=1}^{n} \left(\mathbf{Var}[w_i^{\text{new}}]\right)^2 &\le \frac{512n\epsilon^4}{k^2} + 64\sum_{i=1}^{n}\left((\mathbf{E}[x_i^{-1}\widetilde{\delta}_{x,i}])^4 + (\mathbf{E}[s_i^{-1}\widetilde{\delta}_{s,i}])^4\right) \\
&\le \frac{512n\epsilon^4}{k^2} + 2048\epsilon^4 \le 1000\epsilon^2,
\end{aligned}$$

where we used $\|\mathbf{E}[s^{-1}\widetilde{\delta}_s]\|_2^2, \|\mathbf{E}[x^{-1}\widetilde{\delta}_x]\|_2^2 \le 4\epsilon^2$ and $k \ge \sqrt{n}\epsilon$ at the end. □

Now, we analyze the cost per iteration in procedure Main. This is a direct application of our projection maintenance result.

Lemma 4.16. *For* $\epsilon \ge \frac{1}{\sqrt{n}}$, *each iteration of* Main *(Algorithm 2) takes*

$$n^{1+a+o(1)} + \epsilon \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)})$$

expected time per iteration in amortized where $0 \le a \le \alpha$ *controls the batch size in the data structure and* α *is the dual exponent of matrix multiplication.*

Proof. Lemma 4.14 shows that ClassicalStep happens with only $O(1/n^2)$ probability each step. Since the cost of each step only takes $\widetilde{O}(n^{2.5})$, the expected cost is only $\widetilde{O}(n^{0.5})$.

Lemma 4.15 shows that the conditions in Theorem 5.1 holds with the parameter $C_1 = O(\epsilon), C_2 = O(\epsilon), \epsilon_{\text{mp}} = \Theta(1)$.

In the procedure StochasticStep, Theorem 5.1 shows that the amortized time per iteration is mainly dominated by two steps:

1. mp.Update(w): $O(\epsilon \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}))$.
2. mp.Query($\frac{1}{\sqrt{XS}}\widetilde{\delta}_\mu$): $O(n \cdot \|\widetilde{\delta}_\mu\|_0 + n^{1+a+o(1)})$.

Combining both running time and using $\mathbf{E}[\|\widetilde{\delta}_\mu\|_0] = O(1+k) = O(\epsilon\sqrt{n}\log^2 n)$ (according to the probability of success in Claim 4.7 and matching Assumption 4.1), we have the result. □

4.5 Main Result

Proof of Theorem 2.1. In the beginning of the Main algorithm, Lemma A.6 is called to modify the linear program. Then, we run the stochastic central path method on this modified linear program.

When the algorithm stops, we obtain a vector x and s such that $xs \approx_{0.1} t$ with $t \le \frac{\delta^2}{32n^3}$. Hence, the duality gap is bounded by $\sum_i x_i s_i \le (\delta/4n)^2$. Lemma A.6 shows how to obtain an approximate solution of the original linear program with the guarantee needed using the x and s we just found.

Since t is decreased by $1 - \frac{\epsilon}{3\sqrt{n}}$ factor each iteration, it takes $O(\frac{\sqrt{n}}{\epsilon} \cdot \log(\frac{n}{\delta}))$ iterations in total. In Lemma 4.16, we proved that each iteration takes

$$n^{1+a+o(1)} + \epsilon \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}),$$

and hence the total runtime is

$$O\left(n^{2.5-a/2+o(1)} + n^{\omega+o(1)} + \frac{n^{1.5+a+o(1)}}{\epsilon}\right) \cdot \log\left(\frac{n}{\delta}\right).$$

Since $\epsilon = \Theta(\frac{1}{\log n})$, the total runtime is

$$O(n^{2.5-a/2+o(1)} + n^{\omega+o(1)} + n^{1.5+a+o(1)}) \cdot \log\left(\frac{n}{\delta}\right).$$

Finally, we note that the optimal choice of a is $\min(\frac{2}{3}, \alpha)$, which gives the promised runtime. □

Using the same proof, but different choice of the parameters, we can analyze the ultra short step stochastic central path method, where each step involves sampling only polylogarithmic coordinates. As we mentioned before, the runtime is still around n^ω.

Corollary 4.17. *Under the same assumption as Theorem 2.1, if we choose* $\epsilon = \Theta(1/\sqrt{n})$ *and* $a = \min(\frac{1}{3}, \alpha)$*, the expected time of Main (Algorithm 2) is*

$$\left(n^{\omega+o(1)} + n^{2.5-\alpha/2+o(1)} + n^{2+1/3+o(1)}\right) \cdot \log\left(\frac{n}{\delta}\right).$$

5 PROJECTION MAINTENANCE

The goal of this section is to prove the following theorem:

Theorem 5.1 (Projection Maintenance). *Given a full rank matrix* $A \in \mathbb{R}^{d\times n}$ *with* $n \ge d$ *and a tolerance parameter* $0 < \epsilon_{\text{mp}} < 1/4$*. Given any positive number* a *such that* $a \le \alpha$ *where* α *is the*

ALGORITHM 3: Projection Maintenance Data Structure

datastructure MaintainProjection ▷ Theorem 5.1

members
- $w \in \mathbb{R}^n$ ▷ Target vector
- $v, \widetilde{v} \in \mathbb{R}^n$ ▷ Approximate vectors $v \approx_{\epsilon_{\text{mp}}} w$ and $\widetilde{v} \approx_{\epsilon_{\text{mp}}} w$
- $A \in \mathbb{R}^{d \times n}$
- $M \in \mathbb{R}^{n \times n}$ ▷ Matrix $M = A^\top (AVA^\top)^{-1} A$
- $\epsilon_{\text{mp}} \in (0, 1/4)$ ▷ Tolerance
- $a \in (0, \alpha]$ ▷ Batch Size n^a for Update

end members

procedure Initialize($A, w, \epsilon_{\text{mp}}, a$) ▷ Lemma 5.3
- $w \leftarrow w$, $v \leftarrow w$, $\epsilon_{\text{mp}} \leftarrow \epsilon_{\text{mp}}$, $A \leftarrow A$, $a \leftarrow a$
- $M \leftarrow A^\top (AVA^\top)^{-1} A$

end procedure

procedure Update(w^{new}) ▷ Lemma 5.4
- $y_i \leftarrow \ln w_i^{\text{new}} - \ln v_i, \forall i \in [n]$
- $r \leftarrow$ the number of indices i such that $|y_i| \geq \epsilon_{\text{mp}}/2$.
- **if** $r < n^a$ **then**
  - $v^{\text{new}} \leftarrow v$
  - $M^{\text{new}} \leftarrow M$
- **else**
  - Let $\pi : [n] \to [n]$ be a sorting permutation such that $|y_{\pi(i)}| \geq |y_{\pi(i+1)}|$
  - **while** $1.5 \cdot r < n$ and $|y_{\pi(\lceil 1.5 \cdot r \rceil)}| \geq (1 - 1/\log n)|y_{\pi(r)}|$ **do**
    - $r \leftarrow \min(\lceil 1.5 \cdot r \rceil, n)$
  - **end while**
  - $v^{\text{new}}_{\pi(i)} \leftarrow \begin{cases} w^{\text{new}}_{\pi(i)} & i \in \{1, 2, \ldots, r\} \\ v_{\pi(i)} & i \in \{r+1, \ldots, n\} \end{cases}$
  - ▷ Compute $M^{\text{new}} = A^\top (AV^{\text{new}} A^\top)^{-1} A$ via Woodbury matrix identity
  - $\Delta \leftarrow \text{diag}(v^{\text{new}} - v)$ ▷ $\Delta \in \mathbb{R}^{n \times n}$ and $\|\Delta\|_0 = r$
  - Let $S \leftarrow \pi([r])$ be the first r indices in the permutation.
  - Let $M_S \in \mathbb{R}^{n \times r}$ be the r columns from S of M.
  - Let $M_{S,S}, \Delta_{S,S} \in \mathbb{R}^{r \times r}$ be the r rows and columns from S of M and Δ.
  - $M^{\text{new}} \leftarrow M - M_S \cdot (\Delta^{-1}_{S,S} + M_{S,S})^{-1} \cdot (M_S)^\top$
- **end if**
- $w \leftarrow w^{\text{new}}$, $v \leftarrow v^{\text{new}}$, $M \leftarrow M^{\text{new}}$
- $\widetilde{v}_i \leftarrow \begin{cases} v_i & \text{if } |\ln w_i - \ln v_i| < \epsilon_{\text{mp}}/2 \\ w_i & \text{otherwise} \end{cases}$
- **return** $\widetilde{v}$

end procedure

procedure Query(h) ▷ Lemma 5.5
- Let $\widetilde{S}$ be the indices i such that $|\ln w_i - \ln v_i| \geq \epsilon_{\text{mp}}/2$.
- **return** $\sqrt{\widetilde{V}} \cdot (M \cdot (\sqrt{\widetilde{V}} \cdot h)) - \sqrt{\widetilde{V}} \cdot (M_{\widetilde{S}} \cdot ((\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} \cdot (M^\top_{\widetilde{S}} \sqrt{\widetilde{V}} h)))$

end procedure

end datastructure

dual exponent of matrix multiplication. There is a deterministic data structure (Algorithm 3) that approximately maintains the projection matrices

$$\sqrt{W}A^\top(AWA^\top)^{-1}A\sqrt{W}$$

for positive diagonal matrices W through the following two operations:

(1) Update(w): Output a vector $\widetilde{v}$ such that for all i,

$$(1-\epsilon_{\text{mp}})\widetilde{v}_i \le w_i \le (1+\epsilon_{\text{mp}})\widetilde{v}_i.$$

(2) Query(h): Output $\sqrt{\widetilde{V}}A^\top(A\widetilde{V}A^\top)^{-1}A\sqrt{\widetilde{V}}h$ for the $\widetilde{v}$ outputted by the last call to Update.

The data structure takes $n^2d^{\omega-2}$ time to initialize and each call of Query(h) takes time

$$n\cdot\|h\|_0 + n^{1+a+o(1)}.$$

Furthermore, if the initial vector $w^{(0)}$ and the (random) update sequence $w^{(1)},\dots,w^{(T)}\in\mathbb{R}^n$ satisfies

$$\sum_{i=1}^n\left(\mathbf{E}[\ln w_i^{(k+1)}]-\ln w_i^{(k)}\right)^2\le C_1^2 \qquad \text{and} \qquad \sum_{i=1}^n(\mathbf{Var}[\ln w_i^{(k+1)}])^2\le C_2^2,$$

with the expectation and variance conditional on $w_i^{(k)}$ for all $k=0,1,\dots,T-1$, then the amortized expected time[7] per call of Update(w) is

$$(C_1/\epsilon_{\text{mp}}+C_2/\epsilon_{\text{mp}}^2)\cdot(n^{\omega-1/2+o(1)}+n^{2-a/2+o(1)}).$$

Remark 5.2. For our linear program algorithm, we have $C_1=O(1/\log n)$, $C_2=O(1/\log n)$ and $\epsilon_{\text{mp}}=\Theta(1)$. See Lemma 4.15.

5.1 Proof Outline

For intuition, we consider the case $C_1=\Theta(1)$, $C_2=\Theta(1)$, and $\epsilon_{\text{mp}}=\Theta(1)$ in this explanation. The correctness of the data structure (Algorithm 3) directly follows from Woodbury matrix identity. (The update rule in Line 34 correctly maintains $M=A^\top(AVA^\top)^{-1}A$). The amortized time analysis is based on a potential function that measures the distance of the approximate vector v and the target vector w. We will show that

- The cost to update the projection M is proportional to the decrease of the potential.
- Each call to query increase the potential by a fixed amount.

Combining both together gives the amortized runtime bound of our data structure.

Now, we explain the definition of the potential. Consider the kth round of the algorithm. For all $i\in[n]$, we define $x_i^{(k)}=\ln w_i^{(k)}-\ln v_i^{(k)}$. Note that $|x_i^{(k)}|$ measures the relative distance between $w_i^{(k)}$ and $v_i^{(k)}$. Our algorithm fixes the indices with largest error $x_i^{(k)}$. To capture the fact that updating in a larger batch is more efficient, we define the potential as a weighted combination of the error where we put more weight to higher $x_i^{(k)}$. Formally, we sort the coordinates of $x^{(k)}$ such that $|x_i^{(k)}|\ge|x_{i+1}^{(k)}|$ and define the potential by

$$\Psi_k=\sum_{i=1}^n g_i\cdot\psi\left(x_i^{(k)}\right),$$

[7] If the input is deterministic, then so is the output and the runtime.

where g_i are positive decreasing numbers to be chosen and ψ is a symmetric ($\psi(x) = \psi(-x)$) positive function that increases on both sides. For intuition, one can think $\psi(x)$ behaves roughly like $|x|$.

Each iteration we update the projection matrix such that the error of $|x_1|, \ldots, |x_r|$ drops from roughly ϵ_{mp} to 0. This decreases the potential of $\psi(x_i^{(k)})$ by $\Omega(\epsilon_{\text{mp}})$ from $i = 1, \ldots, r$. Therefore, the whole potential decreases by $\Omega(\epsilon_{\text{mp}} \sum_{i=1}^r g_i)$. To make the term $\sum_{i=1}^r g_i$ proportional to the time to update a rank r part of the projection matrix, we set

$$g_i = \begin{cases} n^{-a}, & \text{if } i < n^a, \\ i^{\frac{\omega-2}{1-a}-1} n^{-\frac{a(\omega-2)}{1-a}}, & \text{otherwise.} \end{cases} \tag{20}$$

where ω is the exponent of matrix multiplication and a is any positive number less than or equals to the dual exponent of matrix multiplication. Lemma A.4 shows that g is indeed non-increasing and Lemma 5.4 shows that the update time of data-structure is indeed $O(r g_r n^{2+o(1)}) = O(\sum_{i=1}^r g_i n^{2+o(1)})$ for any $r \geq n^a$.

Each call to Update, the expectation of the error vector $x^{(k)}$ moves roughly in an unit ℓ_2 ball. Therefore, the changes of the potential is roughly upper bounded $(\sum_{i=1}^n g_i^2)^{1/2} \approx n^{\omega-5/2}$. Since it takes us $n^{2+o(1)}$ time to decrease the potential by roughly 1 in the update step, the total time is roughly $n^{\omega-1/2}$.

For the case of stochastic central path, we note that the variance of the vector x is quite small. By choosing a smooth potential function ψ (see Equation (21)), we can essentially give the same result as if there is no variance.

5.2 Proof of Theorem 5.1

Now, we give the proof of Theorem 5.1. We will defer some simple calculations into later sections.

Proof of Theorem 5.1.

Proof of Correctness. The definition of $\widetilde{v}$ in Line 37 ensures that $(1 - \epsilon_{\text{mp}})\widetilde{v}_i \leq w_i \leq (1 + \epsilon_{\text{mp}})\widetilde{v}_i$.

Using the Woodbury matrix identity, one can verify that the update rule in Line 34 correctly maintains $M = A^\top (AVA^\top)^{-1} A$. See the deviation of the formula in Lemma 5.3. By the same reasoning, the Line 43 outputs the vector $\sqrt{\widetilde{V}} A^\top (A\widetilde{V}A^\top)^{-1} A\sqrt{\widetilde{V}} h$. This completes the proof of correctness.

Definition of x and y. Consider the kth round of the algorithm. For all $i \in [n]$, we define $x_i^{(k)}$, $x_i^{(k+1)}$ and $y_i^{(k)}$ as follows:

$$x_i^{(k)} = \ln w_i^{(k)} - \ln v_i^{(k)}, y_i^{(k)} = \ln w_i^{(k+1)} - \ln v_i^{(k)}, x_i^{(k+1)} = \ln w_i^{(k+1)} - \ln v_i^{(k+1)}.$$

Note that the difference between $x_i^{(k)}$ and $y_i^{(k)}$ is that w is changing. The difference between $y_i^{(k)}$ and $x_i^{(k+1)}$ is that v is changing.

Assume sorting. Assume the coordinates of vector $x^{(k)} \in \mathbb{R}^n$ are sorted such that $|x_i^{(k)}| \geq |x_{i+1}^{(k)}|$. Let τ and π are permutations such that $|x_{\tau(i)}^{(k+1)}| \geq |x_{\tau(i+1)}^{(k+1)}|$ and $|y_{\pi(i)}^{(k)}| \geq |y_{\pi(i+1)}^{(k)}|$.

Definition of Potential function. Let g be defined in (20). Let $\psi : \mathbb{R} \to \mathbb{R}$ be defined by

$$\psi(x) = \begin{cases} \frac{|x|^2}{\epsilon_{\text{mp}}}, & |x| \in [0, \epsilon_{\text{mp}}/2], \\ \epsilon_{\text{mp}}/2 - \frac{(\epsilon_{\text{mp}} - |x|)^2}{\epsilon_{\text{mp}}}, & |x| \in (\epsilon_{\text{mp}}/2, \epsilon_{\text{mp}}], \\ \epsilon_{\text{mp}}/2, & |x| \in (\epsilon_{\text{mp}}, +\infty). \end{cases} \tag{21}$$

We define the potential at the kth round by

$$\Psi_k = \sum_{i=1}^{n} g_i \cdot \psi\left(x^{(k)}_{\tau_k(i)}\right),$$

where $\tau_k(i)$ is the permutation such that $|x^{(k)}_{\tau_k(i)}| \geq |x^{(k)}_{\tau_k(i+1)}|$.

Bounding the potential.

We can express $\Psi_{k+1} - \Psi_k$ as follows:

$$\begin{aligned}\Psi_{k+1} - \Psi_k &= \sum_{i=1}^{n} g_i \cdot \left(\psi(x^{(k+1)}_{\tau(i)}) - \psi(x^{(k)}_i)\right) \\ &= \sum_{i=1}^{n} g_i \cdot \underbrace{\left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_i)\right)}_{w \text{ move}} - \sum_{i=1}^{n} g_i \cdot \underbrace{\left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})\right)}_{v \text{ move}}. \end{aligned} \tag{22}$$

Now, using Lemmas 5.6 and 5.9, and the fact that $\Psi_0 = 0$ and $\Psi_T \geq 0$, with Equation (22), we get

$$\begin{aligned} 0 \leq \Psi_T - \Psi_0 &= \sum_{k=0}^{T-1} (\Psi_{k+1} - \Psi_k) \\ &\leq \sum_{k=0}^{T-1} \left(O(C_1 + C_2/\epsilon_{\text{mp}}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega-5/2}) - \Omega(\epsilon_{\text{mp}} r_k g_{r_k} / \log n)\right) \\ &= T \cdot O(C_1 + C_2/\epsilon_{\text{mp}}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega-5/2}) - \sum_{k=1}^{T} \Omega(\epsilon_{\text{mp}} r_k g_{r_k} / \log n), \end{aligned}$$

where the third step follows by Lemmas 5.6 and 5.9 and r_k is the number of coordinates we update during that iteration.

Therefore, we get

$$\sum_{k=1}^{T} r_k g_{r_k} = O\left(T \cdot (C_1/\epsilon_{\text{mp}} + C_2/\epsilon_{\text{mp}}^2) \cdot \log^{3/2} n \cdot (n^{\omega-5/2} + n^{-a/2})\right).$$

Proof of running time. See the Section 5.3. □

5.3 Initialization Time, Update Time, Query Time

To formalize the amortized runtime proof, we first analyze the initialization time (Lemma 5.3), update time (Lemma 5.4), and query time (Lemma 5.5) of our projection maintenance data-structure.

Lemma 5.3 (Initialization Time). *The initialization time of data-structure MaintainProjection (Algorithm 3) is $O(n^2 d^{\omega-2})$.*

Proof. Given matrix $A \in \mathbb{R}^{d \times n}$ and diagonal matrix $V \in \mathbb{R}^{n \times n}$, computing $A^\top (AVA^\top)^{-1} A$ takes $O(n^2 d^{\omega-2})$. □

Lemma 5.4 (Update Time). *The update time of data-structure MaintainProjection (Algorithm 3) is $O(r g_r n^{2+o(1)})$ where r is the number of indices we updated in v.*

Proof. Let $A_S \in \mathbb{R}^{d\times r}$ be the r columns from S of A. From kth query to $(k+1)$th query, we have

$$\begin{aligned}
&A^\top (AV^{(k+1)}A^\top)^{-1}A \\
=&A^\top (A(V^{(k)}+\Delta)A^\top)^{-1}A \\
=&A^\top \left((AV^{(k)}A^\top)^{-1} - (AV^{(k)}A^\top)^{-1}A_S(\Delta_{S,S}^{-1} + A_S^\top (AV^{(k)}A^\top)^{-1}A_S)^{-1}A_S^\top (AV^{(k)}A^\top)^{-1}\right)A \\
=&A^\top (AV^{(k)}A^\top)^{-1}A - A^\top (AV^{(k)}A^\top)^{-1}A_S(\Delta_{S,S}^{-1} + A_S^\top (AV^{(k)}A^\top)^{-1}A_S)^{-1}A_S^\top (AV^{(k)}A^\top)^{-1}A \\
=&M^{(k)} - M_S^{(k)}(\Delta_{S,S}^{-1} + M_{S,S}^{(k)})^{-1}(M_S^{(k)})^\top,
\end{aligned}$$

where the second step follows by Woodbury matrix identity and the last step follows by the definition of $M^{(k)} \in \mathbb{R}^{n\times n}$.

Thus, the update rule of matrix $M^{(k+1)} \in \mathbb{R}^{n\times n}$ can be written as

$$M^{(k+1)} = M^{(k)} - M_S^{(k)}(\Delta_{S,S}^{-1} + (M^{(k)})_{S,S})^{-1}(M_S^{(k)})^\top.$$

The updates in round k can be splitted into four parts:

(1) Adding two $r\times r$ matrices takes $O(r^2)$ time.
(2) Computing the inverse of an $r\times r$ matrix takes $O(r^{\omega+o(1)})$ time.
(3) Computing the matrix multiplication of a $n\times r$ and $r\times n$ matrix takes $O(rg_r\cdot n^{2+o(1)})$ time where we used that $r \geq n^a$ (Lemma 2.3).
(4) Adding two $n\times n$ matrices together takes $O(n^2)$ time.

Hence, the total cost is

$$O(r^2 + r^{\omega+o(1)} + rg_r\cdot n^{2+o(1)} + n^2) = O(r^2 + r^{\omega+o(1)} + rg_r\cdot n^{2+o(1)}) = O(rg_r\cdot n^{2+o(1)}),$$

where we used $rg_r \geq 1$ for all $r \geq n^a$ in the first step. □

Lemma 5.5 (Query Time). *The query time of data-structure* MaintainProjection *(Algorithm 3) is* $O(n\cdot \|h\|_0 + n^{1+a+o(1)})$.

Proof. Let $\widetilde{\Delta}$ satisfy $\widetilde{V} = V + \widetilde{\Delta}$. Let $\widetilde{S} \subset [n]$ denote the support of $\widetilde{\Delta}$ and then $|\widetilde{S}| \leq n^a$. Let $\widetilde{r}$ denote $|\widetilde{S}|$. We abuse the notation here, $\widetilde{\Delta}$ denotes both $n\times n$ diagonal matrix and a length n vector.

Using Woodbury matrix identity and definition of M, the same proof as Update time (Lemma 5.4) shows

$$A^\top (A\widetilde{V}A^\top)^{-1}A = M + M_{\widetilde{S}}\left(\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}}\right)^{-1} M_{\widetilde{S}}^\top,$$

where $\widetilde{\Delta}_{\widetilde{S}\times\widetilde{S}}$ has size $\widetilde{r}\times\widetilde{r}$, $M_{\widetilde{S},\widetilde{S}}$ has size $\widetilde{r}\times\widetilde{r}$ and $M_{\widetilde{S}}$ has size $n\times\widetilde{r}$.

To compute $\sqrt{\widetilde{V}}A^\top (A\widetilde{V}A^\top)^{-1}A\sqrt{\widetilde{V}}h$, we just need to compute

$$\sqrt{\widetilde{V}}M\sqrt{\widetilde{V}}h + \sqrt{\widetilde{V}}M_{\widetilde{S}}(\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1}M_{\widetilde{S}}^\top\sqrt{\widetilde{V}}h.$$

Note the running time of computing the first term of the above equation only takes $O(n\cdot\|h\|_0)$ time.

Next, we analyze the cost of computing the second term of the above equation. It contains several parts:

(1) Computing $\widetilde{M}_{\widetilde{S}}^\top\cdot(\sqrt{\widetilde{V}}\cdot h) \in \mathbb{R}^{\widetilde{r}}$ takes $\widetilde{r}\|h\|_0$ time.
(2) Computing $(\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} \in \mathbb{R}^{\widetilde{r}\times\widetilde{r}}$ that is the inverse of a $\widetilde{r}\times\widetilde{r}$ matrix takes $\widetilde{r}^{\omega+o(1)}$ time.

(3) Computing matrix-vector multiplication between $\widetilde{r} \times \widetilde{r}$ matrix $((\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1})$ and $\widetilde{r} \times 1$ vector $(\widetilde{M}^{\top}_{\widetilde{S}} \sqrt{\widetilde{V}} h)$ takes $O(\widetilde{r}^2)$ time.
(4) Computing matrix-vector multiplication between $n \times \widetilde{r}$ matrix $(M_{\widetilde{S}})$ and $\widetilde{r} \times 1$ vector $((\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} M^{\top}_{\widetilde{S}} \sqrt{\widetilde{V}} h)$ takes $O(n\widetilde{r})$ time.
(5) Computing the entry-wise product of two n vectors takes $O(n)$ time.

Thus, overall the running time is

$$O(\widetilde{r}\|h\|_0 + \widetilde{r}^{\omega+o(1)} + \widetilde{r}^2 + n\widetilde{r} + n) = O(\widetilde{r}^{\omega+o(1)} + n\widetilde{r}) = O(n^{a\cdot\omega+o(1)} + n^{1+a}).$$

Finally, we note that $\omega \leq 3 - \alpha \leq 3 - a$ (Lemma A.4) and hence $a \cdot \omega \leq a(3-a) \leq 1 + a$. Therefore, the runtime is $n^{1+a+o(1)}$. □

5.4 Bounding w Move

The goal of this section is to prove Lemma 5.6.

Lemma 5.6 (w move). *We have*

$$\sum_{i=1}^{n} g_i \cdot \mathbf{E}\left[\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_i)\right] \leq O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega-5/2}).$$

Proof. Observe that since the errors $|x^{(k)}_i|$ are sorted in descending order, and $\psi(x)$ is symmetric and non-decreasing function for $x \geq 0$, thus $\psi(x^{(k)}_i)$ is also in decreasing order. In addition, note that g is decreasing, we have

$$\sum_{i=1}^{n} g_i \psi\left(x^{(k)}_{\pi(i)}\right) \leq \sum_{i=1}^{n} g_i \psi\left(x^{(k)}_i\right). \tag{23}$$

Hence, the first term in Equation (22) can be upper bounded as follows:

$$\begin{aligned}
\mathbf{E}\left[\sum_{i=1}^{n} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_i)\right)\right] &\leq \mathbf{E}\left[\sum_{i=1}^{n} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_{\pi(i)})\right)\right] \qquad \text{by Equation (23)} \\
&= \sum_{i=1}^{n} g_i \cdot \mathbf{E}[\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_{\pi(i)})] \\
&= O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega-5/2}) \text{ by Lemma 5.7.}
\end{aligned}$$

Thus, we complete the proof of w move Lemma. □

It remains to prove the following Lemma.

Lemma 5.7.

$$\sum_{i=1}^{n} g_i \cdot \mathbf{E}[\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_{\pi(i)})] = O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega-5/2}).$$

Proof. We separate the term into two:

$$\begin{aligned}
\sum_{i=1}^{n} g_i \cdot \mathbf{E}[\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_{\pi(i)})] = &\sum_{i=1}^{n} g_{\pi^{-1}(i)} \cdot \mathbf{E}[\psi(y^{(k)}_i) - \psi(\mathbf{E}[y^{(k)}_i])] \\
&+ \sum_{i=1}^{n} g_{\pi^{-1}(i)} \cdot (\psi(\mathbf{E}[y^{(k)}_i]) - \psi(x^{(k)}_i)).
\end{aligned}$$

For the first term, Mean value theorem shows that

$$\begin{aligned}\psi(y_i^{(k)}) - \psi(\mathbf{E}[y_i^{(k)}]) &= \psi'(\mathbf{E}[y_i^{(k)}])(y_i^{(k)} - \mathbf{E}[y_i^{(k)}]) + \frac{1}{2}\psi''(\zeta)(y_i^{(k)} - \mathbf{E}[y_i^{(k)}])^2 \\ &\le \psi'(\mathbf{E}[y_i^{(k)}])(w_i^{(k+1)} - \mathbf{E}[w_i^{(k+1)}]) + \frac{L_2}{2}\left(w_i^{(k+1)} - \mathbf{E}[w_i^{(k+1)}]\right)^2,\end{aligned}$$

where $L_2 = \max_x \psi''(x)$. Let $\gamma_i = \mathbf{Var}[\ln w_i^{(k+1)}]$. Summing over i and taking conditional expectation given $w^{(k)}$ on both sides, we get

$$\begin{aligned}\sum_{i=1}^n g_{\pi^{-1}(i)}\mathbf{E}[\psi(y_i^{(k)}) - \psi(\mathbf{E}[y_i^{(k)}])] &\le \sum_{i=1}^n g_{\pi^{-1}(i)}\psi'(\mathbf{E}[y_i^{(k)}])\mathbf{E}[w_i^{(k+1)} - \mathbf{E}[w_i^{(k+1)}]] + \frac{L_2}{2}\sum_{i=1}^n g_{\pi^{-1}(i)}\gamma_i \\ &= \frac{L_2}{2}\cdot\sum_{i=1}^n g_{\pi^{-1}(i)}\gamma_i \\ &\le \frac{L_2}{2}\cdot\|g\|_2\cdot\left(\sum_{i=1}^n \gamma_i^2\right)^{1/2} \\ &\le \frac{L_2}{2}\cdot C_2\cdot\|g\|_2.\end{aligned}$$

For the second term, we define $\beta_i = \mathbf{E}[\ln w_i^{(k+1)}] - \ln w_i^{(k)}$. Lipschitz constant of ψ shows that

$$\begin{aligned}\sum_{i=1}^n g_{\pi^{-1}(i)}(\psi(\mathbf{E}[y_i^{(k)}]) - \psi(x_i^{(k)})) &\le L_1\cdot\sum_{i=1}^n g_{\pi^{-1}(i)}|\mathbf{E}[y_i^{(k)}] - x_i^{(k)}| \\ &= L_1\cdot\sum_{i=1}^n g_{\pi^{-1}(i)}|\beta_i| \\ &\le L_1\cdot C_1\cdot\|g\|_2,\end{aligned}$$

where we used that $\sum_{i=1}^n \beta_i^2 \le C_1^2$.

Now, combining both terms and using that $L_1 = O(1), L_2 = O(1/\epsilon_{\mathrm{mp}})$ (from part 4 of Lemma 5.10) and $\|g\|_2 \le \sqrt{\log n}\cdot O(n^{-a/2} + n^{\omega-5/2})$ (from Lemma 5.8), we have that

$$\sum_{i=1}^n g_i\cdot\mathbf{E}[\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\pi(i)}^{(k)})] \le O(C_1 + C_2/\epsilon_{\mathrm{mp}})\cdot\sqrt{\log n}\cdot(n^{-a/2} + n^{\omega-5/2}). \qquad \square$$

Lemma 5.8.

$$\left(\sum_{i=1}^n g_i^2\right)^{1/2} \le \sqrt{\log n}\cdot O(n^{-a/2} + n^{\omega-5/2}).$$

Proof. Since function g behaves differently when $i \le n^a$ and $i > n^a$. We split the sum into two parts.

For the first part, we have

$$\sum_{i=1}^{n^a} g_i^2 = \sum_{i=1}^{n^a} n^{-2a} = n^{-a}.$$

For the second part, we have

$$\sum_{i=n^a}^n g_i^2 = \sum_{i=n^a}^n i^{\frac{2(\omega-2)}{1-a}-2}n^{-\frac{2a(\omega-2)}{1-a}} = \sum_{i=n^a}^n \frac{1}{i}\cdot i^{\frac{2(\omega-2)}{1-a}-1}n^{-\frac{2a(\omega-2)}{1-a}}.$$

Note that

$$\max_{i\in[n^a,n]} i^{\frac{2(\omega-2)}{1-a}-1}n^{-\frac{2a(\omega-2)}{1-a}} = \max(n^{a\frac{2(\omega-2)}{1-a}-a}n^{-\frac{2a(\omega-2)}{1-a}}, n^{\frac{2(\omega-2)}{1-a}-1}n^{-\frac{2a(\omega-2)}{1-a}}) = \max(n^{-a}, n^{2\omega-5}).$$

Thus, the second part is

$$\sum_{i=n^a}^{n} g_i^2 \le \sum_{i=n^a}^{n} \frac{1}{i}\cdot\max(n^{-a}, n^{2\omega-5}) = O(\log n)\cdot\max(n^{-a}, n^{2\omega-5}).$$

Combining the first part and the second part completes the proof. □

5.5 Bounding v Move

Lemma 5.9 (v move). *We have*

$$\sum_{i=1}^{n} g_i\cdot\left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})\right) \ge \Omega(\epsilon_{\mathrm{mp}} r_k g_{r_k}/\log n).$$

Proof. We first understand some simple facts that are useful in the later proof. Note that from the definition of $x_i^{(k+1)}$, we know that $x^{(k+1)}$ has r_k coordinates are 0 and hence $\|y^{(k)} - x^{(k+1)}\|_0 = r_k$. The difference between those vectors is, for the largest r_k coordinates in $y^{(k)}$, we erase them in $x^{(k+1)}$. Then for each $i \in [n-r_k]$, $x^{(k+1)}_{\tau(i)} = y^{(k)}_{\pi(i+r_k)}$. For convenience, we define $y^{(k)}_{\pi(n+i)} = 0, \forall i \in [r_k]$.

We split the proof into two cases.

Case 1. We exit the while loop when $1.5r_k \ge n$.

Let u^* denote the largest u s.t. $|y^{(k)}_{\pi(u)}| \ge \epsilon_{\mathrm{mp}}/2$. If $u^* = r_k$, then we have that $|y^{(k)}_{\pi(r_k)}| \ge \epsilon_{\mathrm{mp}}/2 \ge \epsilon_{\mathrm{mp}}/100$. Otherwise, the condition of the loop shows that

$$|y^{(k)}_{\pi(r_k)}| \ge (1-1/\log n)^{\log_{1.5} r_k - \log_{1.5} u^*}|y^{(k)}_{\pi(u^*)}| \ge (1-1/\log n)^{\log_{1.5} n}\epsilon_{\mathrm{mp}}/2 \ge \epsilon_{\mathrm{mp}}/100,$$

where we used that $n \ge 4$.

According to definition of $x^{(k+1)}_{\tau(i)}$, we have

$$\begin{aligned}\sum_{i=1}^{n} g_i(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})) &= \sum_{i=1}^{n} g_i(\psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k)}_{\pi(i+r_k)})) \ge \sum_{i=n/3+1}^{n} g_i(\psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k)}_{\pi(i+r_k)}))\\ &\ge \sum_{i=n/3+1}^{n} g_i(\psi(y^{(k)}_{\pi(i)})) \ge \sum_{i=n/3+1}^{2n/3} g_i\psi(\epsilon_{\mathrm{mp}}/100) \ge \Omega(r_k g_{r_k}\epsilon_{\mathrm{mp}}),\end{aligned}$$

where the first step follows from $x^{(k+1)}_{\tau(i)} = y^{(k)}_{\pi(i+r_k)}$, the second step follows from the facts that $\psi(|x|)$ is non-decreasing (part 2 of Lemma 5.10) and $|y^{(k)}_{\pi(i)}|$ is non-increasing, the third step follows from $1.5r_k > n$ and hence $\psi(y^{(k)}_{\pi(i+r_k)}) = 0$ for $i \ge n/3+1$, the fourth step follows from the facts ψ is non-decreasing and $|y^{(k)}_{\pi(i)}| \ge |y^{(k)}_{\pi(r_k)}| \ge \epsilon_{\mathrm{mp}}/100$ for all $i < 2n/3$, and the last step follows by the fact g is decreasing and part 3 of Lemma 5.10.

Case 2. We exit the while loop when $1.5r_k < n$ and $|y^{(k)}_{\pi(1.5r_k)}| < (1-1/\log n)|y^{(k)}_{\pi(r_k)}|$.

By the same argument as Case 1, we have that $|y^{(k)}_{\pi(r_k)}| \ge \epsilon_{\mathrm{mp}}/100$. Part 3 of Lemma 5.10 together with the fact

$$|y^{(k)}_{\pi(1.5r)}| < \min(\epsilon_{\mathrm{mp}}/2, |y^{(k)}_{\pi(r)}|\cdot(1-1/\log n)),$$

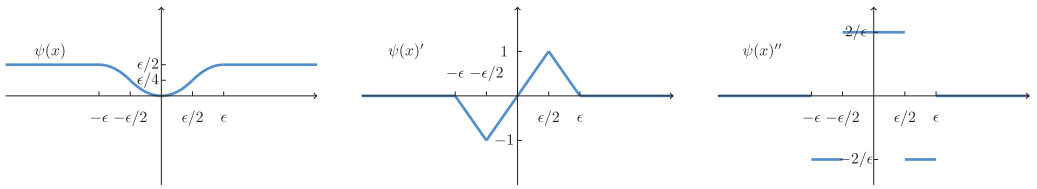

Fig. 2. $\psi(x)$, $\psi(x)'$ and $\psi(x)''$. For $\epsilon_{\mathrm{mp}} \in (0,1)$.

shows that

$$\psi(|y^{(k)}_{\pi(1.5r)}|) - \psi(|y^{(k)}_{\pi(r)}|) = \Omega(\epsilon_{\mathrm{mp}}/\log n). \tag{24}$$

Putting it all together, we have

$$\begin{aligned}
&\sum_{i=1}^{n} g_i \cdot (\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})) \\
=& \sum_{i=1}^{n} g_i \cdot (\psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k)}_{\pi(i+r_k)})) && \text{by } x^{(k+1)}_{\tau(i)} = y^{(k)}_{\pi(i+r_k)} \\
\geq& \sum_{i=r_k/2}^{r_k} g_i \cdot (\psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k)}_{\pi(i+r_k)})) && \text{by } \psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k)}_{\pi(i+r_k)}) \geq 0 \\
\geq& \sum_{i=r_k/2}^{r_k} g_i \cdot (\psi(y^{(k)}_{\pi(r_k)}) - \psi(y^{(k)}_{\pi(1.5r_k)})) \\
\geq& \sum_{i=r_k/2}^{r_k} g_i \cdot \Omega\left(\frac{\epsilon_{\mathrm{mp}}}{\log n}\right) && \text{by (24)} \\
\geq& \sum_{i=r_k/2}^{r_k} g_{r_k} \cdot \Omega\left(\frac{\epsilon_{\mathrm{mp}}}{\log n}\right) && \text{by } g_i \text{ is decreasing} \\
=& \Omega(\epsilon_{\mathrm{mp}} r_k g_{r_k}/\log n),
\end{aligned}$$

where the third step follows by the facts $|y^{(k)}_{\pi(i)}|$ is decreasing and ψ is non-decreasing (from part 2 of Lemma 5.10). □

5.6 Potential Function ψ

Lemma 5.10 (Properties of Function ψ). *Let function ψ be defined in Equation (21). Then function ψ satisfies the following properties:*

1. *Symmetric ($\psi(-x) = \psi(x)$) and $\psi(0) = 0$;*
2. *$\psi(|x|)$ is non-decreasing;*
3. *$|\psi'(x)| = \Omega(1), \forall |x| \in [0.01\epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{mp}}]$;*
4. *$L_1 \stackrel{\text{def}}{=} \max_x \psi'(x) = 1$ and $L_2 \stackrel{\text{def}}{=} \max_x \psi''(x) = 1/\epsilon_{\mathrm{mp}}$.*

Proof. We can see that

$$\psi(x)' = \begin{cases} \frac{2|x|}{\epsilon_{\mathrm{mp}}}, & |x| \in [0, \epsilon_{\mathrm{mp}}/2], \\ \frac{2(\epsilon_{\mathrm{mp}} - |x|)}{\epsilon_{\mathrm{mp}}}, & |x| \in (\epsilon_{\mathrm{mp}}/2, \epsilon_{\mathrm{mp}}], \\ 0, & |x| \in (\epsilon_{\mathrm{mp}}, +\infty), \end{cases} \quad \text{and} \quad \psi(x)'' = \begin{cases} \frac{2}{\epsilon_{\mathrm{mp}}}, & x \in [0, \epsilon_{\mathrm{mp}}/2] \cup [-\epsilon_{\mathrm{mp}}, -\epsilon_{\mathrm{mp}}/2], \\ -\frac{2}{\epsilon_{\mathrm{mp}}}, & x \in (\epsilon_{\mathrm{mp}}/2, \epsilon_{\mathrm{mp}}] \cup [-\epsilon_{\mathrm{mp}}/2, 0], \\ 0, & |x| \in (\epsilon_{\mathrm{mp}}, +\infty). \end{cases}$$

From the $\psi(x)'$ and $\psi(x)''$, it is not hard to see that ψ satisfies the properties needed. □

A APPENDIX

Lemma A.1. *Let x and y denote (possibly dependent) random variables such that $|x| \leq c_x$ and $|y| \leq c_y$ almost surely. Then, we have*

$$\mathbf{Var}[xy] \leq 2c_x^2 \cdot \mathbf{Var}[y] + 2c_y^2 \cdot \mathbf{Var}[x].$$

Proof. Recall that $\mathbf{Var}[xy] \leq \mathbf{E}[(xy - t)^2]$ for any scalar t. Hence,

$$\begin{aligned}\mathbf{Var}[xy] &\leq \mathbf{E}[(xy - \mathbf{E}[x]\mathbf{E}[y])^2] = \mathbf{E}[(xy - x\mathbf{E}[y] + x\mathbf{E}[y] - \mathbf{E}[x]\mathbf{E}[y])^2] \\ &\leq 2\,\mathbf{E}[(xy - x\mathbf{E}[y])^2] + 2\,\mathbf{E}[(x\mathbf{E}[y] - \mathbf{E}[x]\mathbf{E}[y])^2] \\ &\leq 2c_x^2 \cdot \mathbf{Var}[y] + 2c_y^2 \cdot \mathbf{Var}[x].\end{aligned}$$

□

Lemma A.2 ([57]). *Given a matrix $A \in \mathbb{R}^{d\times n}$, vectors $b \in \mathbb{R}^d, c \in \mathbb{R}^n$. Suppose $x, s, y \in \mathbb{R}^n$ satisfy that $xs \approx_{0.1} t$, $Ax = b$ and $A^\top y + s = c$ for some $t > 0$. For any $\epsilon \in (0, 1/2]$, in $\widetilde{O}(n^{2.5}\log(n/\epsilon))$ time, we can find vectors $x^{\text{new}}, s^{\text{new}} \in \mathbb{R}^n$ and $y^{\text{new}} \in \mathbb{R}^d$ such that*

$$\begin{aligned}\|x^{\text{new}}s^{\text{new}} - t\|_2 &\leq \epsilon, \\ Ax^{\text{new}} &= b, \\ A^\top y^{\text{new}} + s &= c.\end{aligned}$$

Remark A.3. Instead of using the algorithm in Reference [57], one can also run our algorithm with $k = n$ for $O(\sqrt{n}\log n)$ iterations. Since $k = n$, there is no randomness involved and hence Φ will decrease deterministically to $O(n)$.

Lemma A.4. $\omega \leq 3 - \alpha$.

Proof. We consider a $n \times n$ matrix A multiply another $n \times n$ matrix B. We split A into $n^{1-\alpha}$ fat matrices where each of them has size $n^\alpha \times n$. Since ω is the best exponent of matrix multiplication, thus we know

$$n^{\omega+o(1)} \leq n^{1-\alpha} \cdot n^{2+o(1)}.$$

Taking $n \to \infty$, this implies $\omega \leq 3 - \alpha$. □

Lemma A.5 (Rectangular Matrix Multiplication). *For any $n \geq r$, multiplying an $n \times r$ with an $r \times n$ matrix or $n \times n$ with $n \times r$ takes time*

$$n^{2+o(1)} + r^{\frac{\omega-2}{1-\alpha}} n^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)}.$$

Proof. The cost for multiplying a $n \times n$ and a $n \times r$ matrix is the same as multiplying a $n \times r$ and a $r \times n$ matrix [45, page 51]. So, we focus on the later case.

For the case $r \leq n^\alpha$, it follows from the rectangular matrix multiplication result in Reference [31].

For the case $r \geq n^\alpha$, we let $k = (n/r)^{\frac{1}{1-\alpha}}$. We can view the problem as multiplying a $k \times k^\alpha$ and a $k^\alpha \times k$ block matrices and each block has size $\frac{n}{k} \times \frac{n}{k}$ size. Therefore, the total cost is

$$k^{2+o(1)} \times \left(\frac{n}{k}\right)^{\omega+o(1)} = r^{\frac{\omega-2}{1-\alpha}} n^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)}.$$

□

Lemma A.6. *Let $A \in \mathbb{R}^{d\times n}$, $b \in \mathbb{R}^d$ and $c \in \mathbb{R}^n$. For a matrix A, we define $\|A\|_1$ to be $\sum_{i,j} |A_{i,j}|$. Given a linear program $\min_{Ax=b, x\geq 0} c^\top x$ with n variables and d constraints. Assume that*

1. Diameter of the polytope : For any $x \geq 0$ with $Ax = b$, we have that $\|x\|_\infty \leq R$.

2. Lipschitz constant of the linear program : $\|c\|_\infty \leq L$.

For any $\delta \in (0,1]$, the modified linear program $\min_{\overline{A}\overline{x}=\overline{b},\overline{x}\geq 0} \overline{c}^\top \overline{x}$, with

$$\overline{A} = \begin{bmatrix} A & 0 & \frac{1}{R}b - A1_n \\ 1_n^\top & 1 & 0 \end{bmatrix} \in \mathbb{R}^{(d+1)\times(n+2)}, \quad \overline{b} = \begin{bmatrix} \frac{1}{R}b \\ n+1 \end{bmatrix} \in \mathbb{R}^{d+1} \text{ and, } \quad \overline{c} = \begin{bmatrix} \frac{\delta}{L}\cdot c \\ 0 \\ 1 \end{bmatrix} \in \mathbb{R}^{n+2}$$

satisfies the following:

1. $\overline{x} = \begin{bmatrix} 1_n \\ 1 \\ 1 \end{bmatrix} \in \mathbb{R}^{n+2}$, $\overline{y} = \begin{bmatrix} 0_d \\ -1 \end{bmatrix} \in \mathbb{R}^{d+1}$, *and* $\overline{s} = \begin{bmatrix} 1_n + \frac{\delta}{L}\cdot c \\ 1 \\ 1 \end{bmatrix} \in \mathbb{R}^{n+2}$ *are feasible primal dual vectors.*
2. *For any feasible primal dual vectors $(\overline{x},\overline{y},\overline{s}) \in \mathbb{R}^{(n+2)\times(d+1)\times(n+2)}$ with duality gap $\leq \delta^2$, the vector $\widehat{x} = R\cdot \overline{x}_{1:n} \in \mathbb{R}^n$ ($\overline{x}_{1:n}$ is the first n coordinates of x) is an approximate solution to the original linear program in the following sense*

$$\begin{aligned} c^\top \widehat{x} &\leq \min_{Ax=b,\, x\geq 0} c^\top x + LR\cdot\delta, \\ \|A\widehat{x} - b\|_1 &\leq 4n\delta\cdot(R\|A\|_1 + \|b\|_1), \\ \widehat{x} &\geq 0. \end{aligned}$$

Proof. **Part 1.** For the first result, straightforward calculations show that $(\overline{x},\overline{y},\overline{s}) \in \mathbb{R}^{(n+2)\times(d+1)\times(n+2)}$ are feasible, i.e.,

$$\overline{A}\overline{x} = \begin{bmatrix} A & 0 & \frac{1}{R}b - A1_n \\ 1_n^\top & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1_n \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{R}b \\ n+1 \end{bmatrix} = \overline{b}$$

and

$$\begin{aligned} \overline{A}^\top \overline{y} + \overline{s} &= \begin{bmatrix} A^\top & 1_n \\ 0 & 1 \\ \frac{1}{R}b^\top - 1_n^\top A^\top & 0 \end{bmatrix} \cdot \begin{bmatrix} 0_d \\ -1 \end{bmatrix} + \begin{bmatrix} 1_n + \frac{\delta}{L}\cdot c \\ 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -1_n \\ -1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1_n + \frac{\delta}{L}\cdot c \\ 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{\delta}{L}\cdot c \\ 0 \\ 1 \end{bmatrix} \\ &= \overline{c}. \end{aligned}$$

Part 2. For the second result, we let

$$\mathrm{OPT} = \min_{Ax=b,\, x\geq 0} c^\top x, \text{ and, } \overline{\mathrm{OPT}} = \min_{\overline{A}\overline{x}=\overline{b},\,\overline{x}\geq 0} \overline{c}^\top \overline{x}.$$

For any optimal $x \in \mathbb{R}^n$ in the original LP, we consider the following $\overline{x} \in \mathbb{R}^{n+2}$:

$$\overline{x} = \begin{bmatrix} \frac{1}{R}x \\ n + 1 - \frac{1}{R}\sum_{i=1}^n x_i \\ 0 \end{bmatrix} \tag{25}$$

and $\overline{c} \in \mathbb{R}^{n+2}$

$$\overline{c} = \begin{bmatrix} \frac{\delta}{L} \cdot c^\top \\ 0 \\ 1 \end{bmatrix}. \tag{26}$$

We want to argue that $\overline{x} \in \mathbb{R}^{n+2}$ is feasible in the modified LP. It is obvious that $\overline{x} \geq 0$, it remains to show $\overline{A}\overline{x} = \overline{b} \in \mathbb{R}^{d+1}$. We have

$$\overline{A}\overline{x} = \begin{bmatrix} A & 0 & \frac{1}{R}b - A1_n \\ 1_n^\top & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{R}x \\ n+1-\frac{1}{R}\sum_{i=1}^n x_i \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{R}Ax \\ n+1 \end{bmatrix} = \begin{bmatrix} \frac{1}{R}b \\ n+1 \end{bmatrix} = \overline{b},$$

where the third step follows from $Ax = b$, and the last step follows from the definition of $\overline{b}$.

Therefore, using the definition of $\overline{x}$ in Equation (25), we have that

$$\overline{\mathrm{OPT}} \leq \overline{c}^\top \overline{x} = \begin{bmatrix} \frac{\delta}{L} \cdot c^\top & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{R}x \\ n+1-\frac{1}{R}\sum_{i=1}^n x_i \\ 0 \end{bmatrix} = \frac{\delta}{LR} \cdot c^\top x = \frac{\delta}{LR} \cdot \mathrm{OPT}, \tag{27}$$

where the first step follows from the fact that modified program is a minimization problem, the second step follows from the definitions of $\overline{x} \in \mathbb{R}^{n+2}$ (Equation (25)) and $\overline{c} \in \mathbb{R}^{n+2}$ (Equation (26)), the last step follows from the fact that $x \in \mathbb{R}^n$ is an optimal solution in the original linear program.

Given a feasible $(\overline{x}, \overline{y}, \overline{s}) \in \mathbb{R}^{(n+2)\times(d+1)\times(n+2)}$ with duality gap δ^2. Write $\overline{x} = \begin{bmatrix} \overline{x}_{1:n} \\ \tau \\ \theta \end{bmatrix} \in \mathbb{R}^{n+2}$ for some $\tau \geq 0, \theta \geq 0$. We can compute $\overline{c}^\top \overline{x}$, which is $\frac{\delta}{L} \cdot c^\top \overline{x}_{1:n} + \theta$. Then, we have

$$\frac{\delta}{L} \cdot c^\top \overline{x}_{1:n} + \theta \leq \overline{\mathrm{OPT}} + \delta^2 \leq \frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2, \tag{28}$$

where the first step follows from definition of duality gap, the last step follows from Equation (27).

Hence, we can upper bound the OPT of the transformed program as follows:

$$c^\top \widehat{x} = R \cdot c^\top \overline{x}_{1:n} = \frac{LR}{\delta} \cdot \frac{\delta}{L} c^\top \overline{x}_{1:n} \leq \frac{RL}{\delta}\left(\frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2\right) = \mathrm{OPT} + LR \cdot \delta,$$

where the first step follows by $\widehat{x} = R \cdot \overline{x}_{1:n}$, the third step follows by Equation (28).

Note that

$$\frac{\delta}{L} c^\top \overline{x}_{1:n} \geq -\frac{\delta}{L}\|c\|_\infty \|\overline{x}_{1:n}\|_1 = -\frac{\delta}{L}\|c\|_\infty \left\|\frac{1}{R}x\right\|_1 \geq -\frac{\delta}{L}\|c\|_\infty \frac{n}{R}\|x\|_\infty \geq -\delta n, \tag{29}$$

where the second step follows from the definition of $\overline{x} \in \mathbb{R}^{n+2}$, and the last step follows from $\|c\|_\infty \leq L$ and $\|x\|_\infty \leq R$.

We can upper bound the θ in the following sense,

$$\theta \leq \frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2 + \delta n \leq 2n\delta + \delta^2 \leq 4n\delta, \tag{30}$$

where the first step follows from Equations (28) and (29), the second step follows by $\mathrm{OPT} = \min_{Ax=b, x\geq 0} c^\top x \leq nLR$ (because $\|c\|_\infty \leq L$ and $\|x\|_\infty \leq R$), and the last step follows from $\delta \leq 1 \leq n$.

The constraint in the new polytope shows that

$$A\overline{x}_{1:n} + \left(\frac{1}{R}b - A1_n\right)\theta = \frac{1}{R}b.$$

Using $\widehat{x} = Rx_{1:n} \in \mathbb{R}^n$, we have

$$A\frac{1}{R}\widehat{x} + \left(\frac{1}{R}b - A1_n\right)\theta = \frac{1}{R}b.$$

Rewriting it, we have $A\widehat{x} - b = (RA1_n - b)\theta \in \mathbb{R}^d$, and hence

$$\|A\widehat{x} - b\|_1 = \|(RA1_n - b)\theta\|_1 \leq \theta(\|RA1_n\|_1 + \|b\|_1) \leq \theta \cdot (R\|A\|_1 + \|b\|_1) \leq 4n\delta \cdot (R\|A\|_1 + \|b\|_1),$$

where the second step follows from the triangle inequality, the third step follows from $\|A1_n\|_1 \leq \|A\|_1$ (because the definition of entry-wise ℓ_1 norm), and the last step follows from Equation (30).

Thus, we complete the proof. □

B GENERALIZED PROJECTION MAINTENANCE

Given the usefulness of projection maintenance, we state a more general version of Theorem 5.1 for future use.

Theorem B.1. *Assume the following about the cost of matrix operations:*

- *In $O(t_k)$ time, we can multiply a $n \times n$ and a $n \times k$ matrix, and we can multiply a $n \times k$ and a $k \times n$ matrix.*
- *In $O(s_n)$ time, we can invert a $n \times n$ matrix, and we can multiply a $n \times n$ and a $n \times n$ matrix.*
- *t_k/k is decreasing k.*

Given a matrix $A \in \mathbb{R}^{d \times n}$ with $n \geq d$, a tolerance parameter $0 < \epsilon_{\text{mp}} < 1/4$ and $k^* \in [n]$, there is a deterministic data structure that approximately maintains the projection matrices

$$\sqrt{W}A^\top(AWA^\top)^{-1}A\sqrt{W}$$

and the inverse matrices $(AWA^\top)^{-1} \in \mathbb{R}^{d \times d}$ for positive diagonal matrices $W \in \mathbb{R}^{n \times n}$ through the following operations:

- *Update(w): Output a vector $\widetilde{v}$ such that for all i,*

$$(1 - \epsilon_{\text{mp}})\widetilde{v}_i \leq w_i \leq (1 + \epsilon_{\text{mp}})\widetilde{v}_i.$$

- *Query$_1(h)$: Output $\sqrt{\widetilde{V}}A^\top(A\widetilde{V}A^\top)^{-1}A\sqrt{\widetilde{V}}h \in \mathbb{R}^n$ for the $\widetilde{v}$ outputted by the last call to Update.*
- *Query$_2(h)$: Output $(A\widetilde{V}A^\top)^{-1}h \in \mathbb{R}^d$ for the $\widetilde{v}$ outputted by the last call to Update.*
- *Insert(a, w_a): Insert a column a into A, a weight w_a into w.*
- *Delete(a): Delete a column a from A and its corresponding weight from w.*
- *Output$()$: Output $\sqrt{\widetilde{V}}A^\top(A\widetilde{V}A^\top)^{-1}A\sqrt{\widetilde{V}} \in \mathbb{R}^{n \times n}$ and $(A\widetilde{V}A^\top)^{-1} \in \mathbb{R}^{d \times d}$ for the $\widetilde{v}$ outputted by the last call to Update.*

Suppose that the number of columns is $O(n)$ during the whole algorithm and that for any call of Update, we have

$$\sum_{i=1}^{n}\left(\mathbf{E}[\ln w_i] - \ln(w_i^{\text{old}})\right)^2 \leq C_1^2 \qquad \text{and} \qquad \sum_{i=1}^{n}(\mathbf{Var}[\ln(w_i)])^2 \leq C_2^2,$$

where w is the input of call, w^{old} is the weight before the call, and the expectation and variance is conditional on w_i^{old}. Then, we have that:

- *The data structure takes $O(s_n + t_n)$ time to initialize.*
- *Each call of Query takes time $O(n \cdot \|h\|_0 + s_{k^*} + nk^*)$.*
- *Each call of Output$()$ takes $O(t_{k^*})$ time.*
- *Each call of Insert and Delete takes $O(n^2)$ time.*

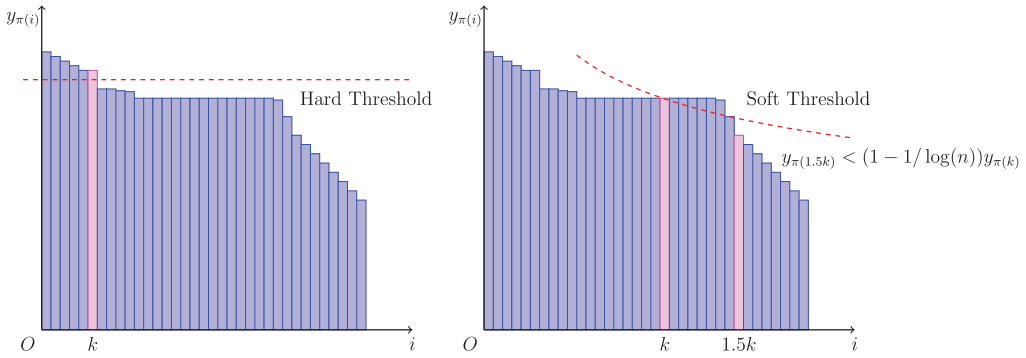

Fig. 3. In this figure, we illustrate the naive hard threshold and the soft threshold. The x-axis represents the sorted n coordinates, and the y-axis represents the errors $y_{\pi(i)}$. All the coordinates smaller than the threshold k are updated. In the left figure, we choose the hard threshold k as the smallest coordinates such that $y_{\pi(k)} \leq \epsilon_{\mathrm{mp}}$. In the right figure, we choose the soft threshold k as the smallest coordinates that satisfies both $y_{\pi(k)} \leq \epsilon_{\mathrm{mp}}$ and $y_{\pi(1.5k)} < (1 - 1/\log(n))y_{\pi(k)}$.

- *Each call of* Update *takes*

$$O\left((C_1/\epsilon_{\mathrm{mp}} + C_2/\epsilon_{\mathrm{mp}}^2) \cdot \left(\frac{t_{k^*}^2}{k^*} + \sum_{i=k^*}^{n} \frac{t_i^2}{i^2}\right)^{1/2} \cdot \log n\right)$$

expected time in amortized.

Proof. The proof is essentially the same as Theorem 5.1. The way to maintain $A\widetilde{V}A^\top \in \mathbb{R}^{d\times d}$ is almost identical to $\sqrt{\widetilde{V}}A^\top(A\widetilde{V}A^\top)^{-1}A\sqrt{\widetilde{V}}h \in \mathbb{R}^n$. Updating both matrices under insertion and deletion can be done via Sherman Morrison formula in $O(n^2)$ time. We note that these updates does not increase our potential and hence it does not affect the amortized cost for Update. Finally, to bound the runtime using t_k and s_n instead of α and ω, we use the same potential $\Psi_k = \sum_{i=1}^n g_i\psi(x_i^{(k)})$ with a new definition of g:

$$g_i = \begin{cases} t_i/i, & \text{if } i \geq k^*, \\ t_{k^*}/k^*, & \text{otherwise.} \end{cases}$$

Now, the update time in Lemma 5.4 becomes $O(rg_r)$. The rest of the proof is identical. In particular, Lemma 5.8 becomes

$$\|g\|_2 = \left(\frac{t_{k^*}}{k^*} + \sum_{i=k^*}^{n} \frac{t_i^2}{i^2}\right)^{1/2},$$

and hence Lemma 5.7 gives the bound

$$O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \left(\frac{t_{k^*}}{k^*} + \sum_{i=k^*}^{n} \frac{t_i^2}{i^2}\right)^{1/2}. \qquad \square$$

ACKNOWLEDGMENTS

We thank Sébastien Bubeck and Aaron Sidford for helpful discussions. We thank Rasmus Kyng for bringing up the question and providing some fixes in the proof of projection maintenance. We thank Josh Alman for some useful discussions about matrix multiplication. We thank Swati Padmanabhan for writing suggestions. We thank Eric Price for the suggestion of the title of this

article. We thank Shunhua Jiang and Hengjie Zhang for drawing several beautiful pictures. Finally, we thank anonymous STOC and JACM reviewers for their detailed feedback.

REFERENCES

[1] Josh Alman. 2019. Limits on the universal method for matrix multiplication. In *Proceedings of the 34th Computational Complexity Conference (CCC'19)*.

[2] Josh Alman and Virginia Vassilevska Williams. 2018. Limits on all known (and some unknown) approaches to matrix multiplication. In *Proceedings of the IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS'18)*. IEEE.

[3] Andris Ambainis, Yuval Filmus, and François Le Gall. 2015. Fast matrix multiplication: Limitations of the Coppersmith-Winograd method. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC'15)*. ACM, 585–593.

[4] Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. 2020. Training (overparametrized) neural networks in near-linear time. Retrieved from https://arxiv.org/pdf/2006.11648.pdf.

[5] Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, and Yuanzhi Li. 2018. An homotopy method for ℓ_p regression provably beyond self-concordance and in input-sparsity time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'18)*. ACM, 1130–1137.

[6] Diptarka Chakraborty, Lior Kamma, and Kasper Green Larsen. 2018. Tight cell probe bounds for succinct boolean matrix-vector multiplication. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'18)*. 1297–1306.

[7] Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. 2020. MONGOOSE: A learnable LSH framework for efficient neural network training. In *OpenReview.net*. Retrieved from https://openreview.net/forum?id=wWK7yXkULyh.

[8] Matthias Christandl, François Le Gall, Vladimir Lysikov, and Jeroen Zuiddam. 2020. Barriers for rectangular matrix multiplication. In *Proceedings of the Computational Complexity Conference (CCC'20)*. Retrieved from https://arXiv.org/pdf/2003.03019.pdf.

[9] Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. 2011. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC'11)*. ACM, 273–282.

[10] Kenneth L. Clarkson and David P. Woodruff. 2013. Low rank approximation and regression in input sparsity time. In *Proceedings of the Symposium on Theory of Computing Conference (STOC'13)*. https://arxiv.org/pdf/1207.6365, 81–90.

[11] Michael B. Cohen, Jonathan Kelner, Rasmus Kyng, John Peebles, Richard Peng, Anup B. Rao, and Aaron Sidford. 2018. Solving directed laplacian systems in nearly-linear time through sparse LU factorizations. In *Proceedings of the IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS'18)*. IEEE, 898–909.

[12] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. 2014. Solving sdd linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC'14)*. ACM, 343–352.

[13] Michael B. Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. 2016. Geometric median in nearly linear time. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC'16)*. ACM, 9–21.

[14] Michael B. Cohen, Aleksander Mądry, Piotr Sankowski, and Adrian Vladu. 2017. Negative-weight shortest paths and unit capacity minimum cost flow in $\widetilde{O}(m^{10/7} \log W)$ time. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*. SIAM, 752–771.

[15] Michael B. Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. 2017. Matrix scaling and balancing via box constrained Newton's method and interior point methods. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*. IEEE, 902–913.

[16] Don Coppersmith and Shmuel Winograd. 1987. Matrix multiplication via arithmetic progressions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC'87)*. ACM, 1–6.

[17] Alexander Munro Davie and Andrew James Stothers. 2013. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh Section A: Mathematics* 143, 2 (2013), 351–369.

[18] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. 2015. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC'15)*. 21–30.

[19] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. 2020. A faster interior point method for semidefinite programming. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS'20)*.

[20] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. 2020. An improved cutting plane method for convex optimization, convex-concave games and its applications. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC'20)*. 944–953. Retrieved from https://arxiv.org/pdf/2004.04250.pdf.

[21] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. 2020. Faster dynamic matrix inverse for faster LPs. Retrieved from https://arxiv.org/2004.07470.

[22] Narendra Karmarkar. 1984. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC'84)*. ACM, 302–311.

[23] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. 2013. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC'13)*. ACM, 911–920.

[24] Ioannis Koutis, Gary L. Miller, and Richard Peng. 2010. Approaching optimality for solving SDD linear systems. In *Proceedings of the IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS'10)*. IEEE, 235–244.

[25] Ioannis Koutis, Gary L. Miller, and Richard Peng. 2011. A nearly-m log n time solver for sdd linear systems. In *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS'11)*. IEEE, 590–598.

[26] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. 2016. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC'16)*. ACM, 842–850.

[27] Rasmus Kyng, Richard Peng, Robert Schwieterman, and Peng Zhang. 2018. Incomplete nested dissection. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'18)*. ACM, 404–417.

[28] Rasmus Kyng and Sushant Sachdeva. 2016. Approximate gaussian elimination for laplacians-fast, sparse, and simple. In *Proceedings of the IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS'16)*. IEEE, 573–582.

[29] Kasper Green Larsen and Ryan Williams. 2017. Faster online matrix-vector multiplication. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*. SIAM, 2182–2189.

[30] François Le Gall. 2014. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC'14)*. ACM, 296–303.

[31] Francois Le Gall and Florent Urrutia. 2018. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*. SIAM, 1029–1046.

[32] Yin Tat Lee and Aaron Sidford. 2013. Path finding I: Solving linear programs with $\widetilde{O}(\sqrt{\text{rank}})$ linear system solves. Retrieved from https://arXiv:1312.6677.

[33] Yin Tat Lee and Aaron Sidford. 2014. Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *Proceedings of the IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS'14)*. IEEE, 424–433.

[34] Yin Tat Lee and Aaron Sidford. 2015. Efficient inverse maintenance and faster algorithms for linear programming. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS'15)*. IEEE, 230–249.

[35] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. 2015. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science (FOCS'15)*. IEEE, 1049–1065.

[36] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. 2019. Solving empirical risk minimization in the current matrix multiplication time. In *Proceedings of the Conference on Learning Theory (COLT'19)*. 2140–2157. Retrieved from https://arxiv.org/pdf/1905.04447.pdf.

[37] S Cliff Liu, Zhao Song, and Hengjie Zhang. 2020. Breaking the n-pass barrier: A streaming algorithm for maximum weight bipartite matching. Retrieved from https://arXiv:2009.06106.

[38] Aleksander Madry. 2013. Navigating central path with electrical flows: From flows to matchings, and back. In *Proceedings of the IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS'13)*. IEEE, 253–262.

[39] Aleksander Madry. 2016. Computing maximum flow with augmenting electrical flows. In *Proceedings of the IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS'16)*. IEEE, 593–602.

[40] Nimrod Megiddo. 2012. *Progress in Mathematical Programming: Interior-Point and Related Methods*. Springer Science & Business Media.

[41] Jelani Nelson and Huy L. Nguyên. 2013. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Proceedings of the IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS'13)*. IEEE, 117–126. Retrieved from https://arxiv.org/pdf/1211.1002.

[42] Yurii Nesterov and Arkadii Nemirovskii. 1994. *Interior-point Polynomial Algorithms in Convex Programming*. Vol. 13. Siam.

[43] Yu Nesterov and Arkadi Nemirovsky. 1989. Self-concordant functions and polynomial-time methods in convex programming. *Technical Report, Central Economic and Mathematic Institute, USSR Academy of Science*.

[44] Yu Nesterov and Arkadi Nemirovsky. 1991. Acceleration and parallelization of the path-following interior point method for a linearly constrained convex quadratic problem. *SIAM J. Optimiz.* 1, 4 (1991), 548–564.

[45] Victor Pan. 1984. How to multiply matrices faster. *Lecture Notes Comput. Sci.* 179 (1984).

[46] Jiming Peng, Cornelis Roos, and Tamás Terlaky. 2002. Self-regular functions and new search directions for linear and semidefinite optimization. *Math. Program.* 93, 1 (2002), 129–171.
[47] James Renegar. 1988. A polynomial-time algorithm, based on Newton's method, for linear programming. *Math. Program.* 40, 1–3 (1988), 59–93.
[48] James Renegar. 2001. *A Mathematical View of Interior-point Methods in Convex Optimization*. Vol. 3. Siam.
[49] Cornelis Roos, Tamás Terlaky, and J.-Ph. Vial. 2005. *Interior Point Methods for Linear Optimization*. Springer Science & Business Media.
[50] Tamás Sarlós. 2006. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 143–152.
[51] Zhao Song and Zheng Yu. 2020. Oblivious sketching-based central path method for solving linear programming problems. In *OpenReview.net*. Retrieved from https://openreview.net/forum?id=fGiKxvF-eub.
[52] Daniel A. Spielman and Nikhil Srivastava. 2011. Graph sparsification by effective resistances. *SIAM J. Comput.* 40, 6 (2011), 1913–1926.
[53] Daniel A. Spielman and Shang-Hua Teng. 2004. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'04)*. ACM, 81–90.
[54] Tamás Terlaky. 2013. *Interior Point Methods of Mathematical Programming*. Vol. 5. Springer Science & Business Media.
[55] Pravin M. Vaidya. 1987. An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$ arithmetic operations. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC'87)*. ACM, 29–38.
[56] Pravin M. Vaidya. 1989. A new algorithm for minimizing convex functions over convex sets. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS'89)*. IEEE, 338–343.
[57] Pravin M. Vaidya. 1989. Speeding-up linear programming using fast matrix multiplication. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS'89)*. IEEE, 332–337.
[58] Jan van den Brand. 2020. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*. SIAM, 259–278.
[59] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. 2020. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC'20)*. 775–788.
[60] Virginia Vassilevska Williams. 2012. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC'12)*. ACM, 887–898.
[61] Max A. Woodbury. 1950. Inverting modified matrices. *Memo. Report* 42, 106 (1950), 336.
[62] Stephen J. Wright. 1997. *Primal-dual Interior-point Methods*. Vol. 54. SIAM.
[63] Yinyu Ye. 1997. *Interior Point Algorithms: Theory and Analysis*. Springer.
[64] Yinyu Ye, Michael J. Todd, and Shinji Mizuno. 1994. An $O(\sqrt{nL})$-iteration homogeneous and self-dual linear programming algorithm. *Math. Oper. Res.* 19, 1 (1994), 53–67.

Received June 2019; revised September 2020; accepted September 2020