Faster Dynamic Matrix Inverse for Faster LPs

Shunhua Jiang* Zhao Song† Omri Weinstein‡ Hengjie Zhang§

Abstract

Motivated by recent Linear Programming solvers, we design dynamic data structures for maintaining the inverse of an $n \times n$ real matrix under *low-rank* updates, with polynomially faster amortized running time. Our data structure is based on a recursive application of the Woodbury-Morrison identity for implementing *cascading* low-rank updates, combined with recent sketching technology. Our techniques and amortized analysis of multi-level partial updates, may be of broader interest to dynamic matrix problems.

This data structure leads to the fastest known LP solver for general (dense) linear programs, improving the running time of the recent algorithms of (Cohen et al.'19, Lee et al.'19, Brand'20) from $O^*(n^{2+\max\{\frac{1}{6},\omega-2,\frac{1-\alpha}{2}\}})$ to $O^*(n^{2+\max\{\frac{1}{18},\omega-2,\frac{1-\alpha}{2}\}})$, where ω and α are the fast matrix multiplication exponent and its dual. Hence, under the common belief that $\omega \approx 2$ and $\alpha \approx 1$, our LP solver runs in $O^*(n^{2.055})$ time instead of $O^*(n^{2.16})$.

*sj3005@columbia.edu. Columbia University. Research supported by NSF CAREER award CCF-1844887.
†zhaos@ias.edu. Princeton University and Institute for Advanced Study.
‡omri@cs.columbia.edu. Columbia University. Research supported by NSF CAREER award CCF-1844887.
§hz2613@columbia.edu. Columbia University. Research supported by NSF CAREER award CCF-1844887.

1 Introduction

Dynamic matrix inverse problems ask to maintain the inverse of an $n \times n$ matrix M over some field (say $\mathbb{R}$), when M undergoes a long sequence of row/column updates. This data structure problem arises in many important TCS applications, such as directed reachability in dynamic graphs and maintaining the eigenvalues and rank of a matrix [San04], empirical risk minimization [LSZ19], routing and electrical-flow computation (inverting Laplacians) [ST04, Mad13], and in fastest known linear programming solvers [LS15, CLS19, BLSS20]. While many variants of this problem have been considered over the years (depending on the specific application), the most common one requires the data structure to support *vector-queries*, i.e., computing $\textsc{Query}(h) = M^{-1}h$ where h comes from some restricted[1] family of $\mathbb{R}^n$, under a long sequence of *low-rank* updates (e.g., row/column changes to M). Recomputing the inverse from scratch upon each update in this setting incurs a daunting computational overhead, and therefore the goal is to optimize the tradeoff between the (amortized) update time t_u and query time t_q (or the total running time). The first dynamic data structure for this problem, tracing back to the 1950's [Woo49, Woo50] shows how to *explicitly* maintain the inverse of M in $O(n^2)$ time, which is already nontrivial. Substantial improvements to this data structure were developed more recently, showing that *row-queries* (which are equivalent to answering $(M^\top)^{-1}h$ for any 1-*sparse* vector h) under row/column updates, can be done in $\max\{t_u, t_q\} = O(n^{1.529})$ time [San04, SM10, BNS19]. Brand et al. [BNS19] also showed a certain conditional $\Omega(n^{1.5})$ worst-case lower bound for the *exact* version of this problem, which is important for some of the aforementioned applications (e.g., maintaining graph properties such as directed reachability).

In this work we consider a more challenging dynamic inverse problem: The data structure needs to support low-rank updates of the form $\textsc{Update}(u, v) = M + uv^\top$ where u, v come from some fixed set of vectors of size $O(n)$, and needs to answer inverse queries $M^{-1}h^{(i)}$ under a *slowly changing* vector sequence ($\|h^{(i)} - h^{(i-1)}\|_0 \leq O(1)$). In contrast to row-queries (a special case $\|h^{(i)}\|_0 = 1$), here the online query vectors h may be arbitrarily dense, but their *differences* are sparse.[2] An important special case of this dynamic problem is "projection maintenance" [CLS19], where updates to M take the form $\textsc{Update}(D) = M + ADA^\top$ where A is an arbitrary fixed matrix and D is a sparse diagonal matrix (hence $\text{rank}(ADA^\top) \leq \text{rank}(D)$ is small and can be therefore written as $\sum_{i=1}^{rank(D)} A_{i_1} A_{i_2}^\top$).

Nevertheless, an important difference between this work and the aforementioned ones on dynamic inverse maintenance (e.g., [San04, SM10, BNS19]), is allowing *approximate* answers, i.e., the data structure needs to compute $M^{-1}h$ up to some small relative ℓ_∞ error. This enables the use of randomized tools from *sketching and sparse recovery* literature [GLPS10, CW13, LNNT16, LSZ19]. Another point of departure is our use of very heavy *amortization*, augmenting the approach of [CLS19, LSZ19, Bra20] with a novel algebraic technique and more sophisticated potential analysis (based on "high order" martingales). A recurring theme in dynamic data structures (both upper and lower bounds) is that amortized analysis is a different ballgame compared to worst-case performance – Some classic examples are the amortized analysis of fully-dynamic undirected connectivity [ST85] and its matching amortized cell-probe lower bound [PD06], which required substantially new techniques (and another decade) compared to the worst-case lower bound [FS89]. In contrast to dynamic graph problems, whose amortized complexity has been studied extensively, its counterpart in *dynamic matrix problems* is far less understood ([HKNS15, BNS19]), and we believe our work sheds further light on the power of amortization.

[1] Indeed, supporting *arbitrary* online queries $h \in \mathbb{R}^n$ is conjectured to be impossible in truly sub-quadratic time even in the *static* case where M remains fixed, see the "oMV Conjecture" [HKNS15].

[2] Note that sparsity of Δh is *not* equivalent to sparsity of queries h themselves: If M were *fixed*, then by linearity, computing $M^{-1}(\Delta h)$ would indeed suffice, but here M is dynamically changing so this standard trick doesn't work.

Dynamic inverse in linear programming The primary application and motivation of this work is the role of dynamic matrix inverse data structures in speeding up *interior-point methods* (IPMs) for solving linear programs (LPs) in close to matrix-multiplication time.

Linear programming is one of the cornerstones of algorithm design and convex optimization, dating back to as early as Fourier in 1827. LPs are the key toolbox for (literally hundreds of) approximation algorithms, and a standard subroutine in convex optimization problems. Dantzig's 1947 *simplex algorithm* [Dan47] was the first proposed solution for general LPs with n variables and d constraints ($\min_{Ax=b,x\geq 0} c^\top x$). Despite its impressive performance in practice, however, the simplex algorithm turned out to have exponential worst-case running time (Klee and Minty [KM72]). The first *polynomial time* algorithm for general LPs was only developed in 1980, when Khachiyan [Kha80] introduced the *Elliposid method*, and showed that it runs in $O(n^6)$ time. Unfortunately, this algorithm is very slow in practice compared to the simplex algorithm, raising a quest for LP solvers which are efficient in both theory and practice.

This was the primary motivation behind the development of *interior point methods* (IPMs), which uses a primal-dual gradient descent approach to iteratively converge to a feasible solution (Karmarkar, [Kar84]). An appealing feature of IPM methods for solving LPs is that they are not only guaranteed to run fast in theory, but also in practice [Str87]. In 1989, Vaidya proposed an $O(n^{2.5})$ LP solver based on a specific implementation of IPMs, known as the *Central Path algorithm* [Vai87, Vai89]. Vaidya already observed that the main bottleneck of this algorithm boils down to a dynamic data structure problem of maintaining the *inverse matrix* M associated with the central path equations (see Section 3), under a sequence of updates of the form $(M+A\Delta A^\top)^{-1}$, where Δ is a sparse diagonal matrix and A is the fixed LP constraint matrix. Since Δ is sparse, each "gradient descent" iteration of this algorithm induces a *low-rank* update to M, hence it is conceivable to avoid recomputing the inverse matrix from scratch—which would naively cost n^ω time per iteration—and gain substantially from amortization. This data structure problem was the centerpiece of the recent line of developments on IPM solvers [CLS19, LSZ19, Bra20], which focused on designing faster dynamic inverse structures for implementing the Central Path algorithm.

The fastest known algorithm for general (dense) LPs, based on this approach, is due to Cohen, Lee and Song [CLS19], whose running time is

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

where $\omega < 2.37$ is the fast matrix-multiplication exponent and $\alpha > 0.31$ is the dual matrix multiplication exponent.[3] Note that n^ω is the minimal time for merely inverting a matrix, i.e., finding *any* feasible solution ($Ax = b$) to the LP, hence it seems quite remarkable that solving the full optimization problem ($\min_{Ax=b,x\geq 0} c^\top x$) may be done at virtually no extra cost. It is widely believed that $\omega \approx 2$ [CKSU05, Wil12] and as such, $\alpha \approx 1$ (though the only formal connection between these constants is $\omega + (\omega/2)\alpha \leq 3$ [CGLZ20]). Assuming indeed that $\omega < 2 + 1/6 \approx 2.166$ and $\alpha > 0.66$, the runtime of the aforementioned algorithms is $n^{2.166}$. Whether the additive $n^{2.166}$ term can be improved or completely removed was explicitly posed as an open question in [CLS19] and [Son19].

Our main result is an affirmative answer to this open question, asserting that LPs can be solved in matrix multiplication time for nearly any value of ω (i.e., so long as $\omega > 2.055$). We design an improved LP solver which runs in time $O^*(n^\omega + n^{2.5-\alpha/2} + n^{2+1/18})$. In the most notable (ideal) case that $\omega \approx 2$ and $\alpha \approx 1$, our algorithm runs in $O^*(n^{2.055})$ time, instead of $O^*(n^{2.166})$ time of previous IPM algorithms [CLS19, LSZ19, Bra20]. More precisely:

[3] The dual exponent α is defined as the asymptotically maximum number $a \leq 1$ s.t multiplying an $n \times n^a$ matrix by an $n^a \times n$ matrix can be done is $n^{2+o(1)}$ time. The current best lower bound is $\alpha > 0.31389$ [GU18].

Theorem 1.1 (Main result, Informal statement of Theorem 4.1)**.** *Let* $\min_{Ax=b,x\geq 0} c^\top x$ *be a linear program where* $A \in \mathbb{R}^{d\times n}$ *and* $d = \Omega(n)$*. Then for any accuracy parameter* $\delta \in (0,1)$*, there is a randomized algorithm that solves the LP in expected time*

$$O^*(n^\omega + n^{2.5-\alpha/2} + n^{2+1/18}) \cdot \log(n/\delta).$$

We achieve this result by designing a more efficient projection maintenance data structure, speeding up both the update and query times of previous algorithms. This is done via a new algebraic framework for bootstrapping lazy updates (described next), combined with randomized compression techniques and a sophisticated amortized analysis of the underlying dynamic process.

Organization In Section 2 we provide a high-level description of our main technique, which is the centerpiece of this paper. Section 3 contains some necessary background and brief overview of previous related work. In Section 4, we provide a detailed technical overview of the proof of Theorem 1.1. This 10-page streamlined overview should be understood as an extended abstract of our entire result, deferring technical proofs and calculations to the Appendix.

Appendix Organization. Section A contains preliminaries and notation. In Section B we provide an analysis of the Stochastic Central Path algorithm, postponing output-feasibility issues to Section I. We put preliminary part of our data structure in Section C. We present our full "cascading data structure" and prove its correctness in Section D, and we analyze its running time in Sections E, F. Finally, Section G contains the final runtime analysis of our LP solver using the full data structure. In Section H, we present more details for Section 2.

2 Bootstrapping low-rank updates via cascading lazy updates

One of the main new ideas of this work is "bootstrapping"[4] the lazy updates technique of [CLS19] by repeated application of the Woodburry-Morrison identity (Lemma A.2), allowing faster low-rank operations via *cascading lazy updates*. For ease of presentation, let us focus on the following simplified dynamic data structure problem, which we henceforth call *low-rank inverse maintenance* problem (Definition. H.5). The data structure is initially given a full rank matrix $M^{(0)} = M \in \mathbb{R}^{n\times n}$, and a fixed vector $h \in \mathbb{R}^n$. The data structure needs to support a sequence of rank-one updates and vector-queries as follows. In the t-th UPDATE operation, we are given two vectors $u^{(t)}, v^{(t)} \in \mathbb{R}^n$ and a real number $c^{(t)} \in \mathbb{R}$, and need to perform a rank-1 update $M^{(t)} = M^{(t-1)} + c^{(t)} \cdot u^{(t)}(v^{(t)})^\top$. A QUERY operation asks to calculate $x = (M^{(T)})^{-1} \cdot h$, where T is the number of updates in the sequence so far. Note that this setting easily captures rank-k updates invoking k consecutive rank-1 updates.

Achieving sub-quadratic update and query times for this problem requires some restriction on the update vectors $u^{(t)}, u^{(t)}$ (we show in Section H that otherwise it would break the oMV Conjecture [HKNS15]). Our technique requires one reasonable assumption, namely, that all updates $u^{(t)}$ and $v^{(t)}$ come from a fixed set $|S| = O(n)$. As noted in the introduction, LP projection maintenance is a special case where $M = (ADA^\top)$, A is the fixed LP matrix and D is a diagonal matrix which is changing slowly under ℓ_0 norm. Thus, a sparse update Δ_i to D corresponds to $M \leftarrow M + \Delta_i \cdot A_i A_i^\top$.

[4]This term refers to a general approach for speeding up dynamic algorithms by repeating a certain technique several times recursively [San04, BNS19]. We remark that [BNS19] uses a different kind of bootstrapping to speed up exact inverse maintenance under simpler row-updates and row-queries (via "one more leve" of FMM). In particular, [BNS19] has nothing to do with *dynamic LU-decompositions* nor recursion on Woodburry's identity. Nevertheless, it is noteworthy that the bottleneck in both works is maintaining certain matrix products for > 2 "levels" of recursion.

We also remark that the assumption that the query vector h is fixed throughout the sequence—while natural in many streaming applications—is only for simplicity of exposition: Our data structure will actually support an online sequence of slowly-changing queries h (i.e., $\|h^{(t)} - h^{(t-1)}\|_0 = o(n)$). Handling sparse updates to h turns out to be much easier than low-rank updates to M, hence we focus on the latter task.

Our technique for solving the *low-rank inverse maintenance* problem is based on an algorithmic generalization of Woodbury's identity to $K > 1$ "levels" (to be explained below), allowing for *recursive lazy updates* of these K levels using different thresholds. The basic idea is to (dynamically) group updates into K "epochs" $0 \le t_1 \le \cdots \le t_{K-1} \le t_K = T$. The first "level" maintains the first epoch t_1 and $M^{(t_1)}$. Similarly, in level $k \in \{2, \cdots, K\}$ we group all updates $u^{(t)}, v^{(t)}, c^{(t)}$ for which $t \in (t_{k-1}, t_k]$, and partition the update sequence in terms of these epochs. More formally, let $U_k, V_k \in \mathbb{R}^{n \times (t_k - t_{k-1})}$ and the diagonal matrix $C_k \in \mathbb{R}^{(t_k - t_{k-1}) \times (t_k - t_{k-1})}$ be, respectively, the concatenation of all u_i, v_i and c_i in the kth epoch, so that $\sum_{i=t_{k-1}+1}^{t_k} c_i \cdot u_i v_i^\top = U_k C_k V_k^\top$. Let r_k be defined as the rank of $U_k C_k V_k^\top$, the epochs are maintained under the invariant that $r_k \le n_k$, where $n = n_1 \gg n_2 \gg \cdots \gg n_K \ge 1$ are predefined thresholds that decrease exponentially and can later be optimized. In this terminology, for any $h \in \mathbb{R}^n$, the query answer

$$x := \left(M^{(t_1)} + \sum_{i=t_1+1}^{T} c_i u_i v_i^\top \right)^{-1} h$$

can be equivalently re-written as the following linear system (by adding 'dummy' variables ξ_j):

$$\begin{bmatrix} x \\ \xi_2 \\ \xi_3 \\ \xi_4 \\ \vdots \\ \xi_K \end{bmatrix} = \left(\underbrace{\begin{bmatrix} M^{(t_1)} & U_2 & U_3 & U_4 & \cdots & U_K \\ V_2^\top & -C_2^{-1} & 0 & 0 & \cdots & 0 \\ V_3^\top & 0 & -C_3^{-1} & 0 & \cdots & 0 \\ V_4^\top & 0 & 0 & -C_4^{-1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ V_K^\top & 0 & 0 & 0 & \cdots & -C_K^{-1} \end{bmatrix}}_{D} \right)^{-1} \cdot \begin{bmatrix} h \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{1}$$

Note that this equation is precisely Woodbury's identity, written in a K-block matrix form. Indeed, Woodbury's identity is derived as the solution x to the linear system

$$\begin{bmatrix} x \\ \xi \end{bmatrix} = \begin{bmatrix} M^{(t_1)} & U \\ V^\top & -C^{-1} \end{bmatrix}^{-1} \cdot \begin{bmatrix} h \\ 0 \end{bmatrix},$$

where here ξ, U, V, C denote the concatenation of ξ_k, U_k, V_k, C_k respectively, over all $k \in \{2, \cdots, K\}$. We now explain how the above generalization leads to an efficient data structure for implementing low-rank updates.

LU-decomposition At update time, the data structure will maintain an LU-decomposition (lower-upper triangular factorization) of the matrix D in (1).

$$D = L \cdot U, \tag{2}$$

where L and U are both K-block triangular matrices (which are uniquely defined by the Gaussian-elimination algorithm and imposing the diagonals of L to be identity matrices, see Eq.(70) for an example of the case $K = 3$). As we will explain in the next paragraph, such decomposition is useful since the inverse of triangular matrices (L, U) can be maintained efficiently. This means that the (slowly changing) query answers $U^{-1}L^{-1}[h, 0]^\top = D^{-1}[h, 0]^\top = [x, \xi]^\top$ can be maintained efficiently with respect to the updated D.

Cascading lazy updates and query The key part of our data structure is performing lazy updates recursively w.r.t different thresholds, to speed up the amortized runtime. Recall that in the latest T-th update, we are given u_T, v_T, c_T. In order to solve the forthcoming queries using this framework, we need to update the epoch in the bottom level t_K to reflect the up-to-date $M^{(T)}$, by updating its corresponding tuple U_K, V_K, C_K and maintain the LU-decomposition. If the update $t_K \leftarrow T$ violates the invariant $r_K \leq n_K$ (Recall $r_k = t_k - t_{k-1}$ is the rank of the update $U_k C_k V_k^\top$ in the k-th epoch), we "cascade" the update to the next level by updating $t_{K-1} \leftarrow t_K \leftarrow T$ and also updating $U_{K-1}, V_{K-1}, C_{K-1}$ to include U_k, V_K, C_K, and recurse on the next level t_{K-2} and so on, until the threshold invariant is restored. A *crucial observation* in the implementation of cascading lazy updates and maintaining the LU-decomposition is that when level k gets updated, the upper-left $(k-1) \times (k-1)$ blocks of L and U, which is the dominating part compared to the entire matrix, *does not change*. Since L and U are triangular matrices, the upper-left $(k-1) \times (k-1)$ blocks of L^{-1} and U^{-1} *also* remain intact. If we write the new L^{-1}, U^{-1} as $(L^{-1})^{\text{new}} = L^{-1} + \Delta L$ and $(U^{-1})^{\text{new}} = U^{-1} + \Delta U$, the non-zero part of ΔL (ΔU) is lower (upper) triangular of $n \times n_k$ submatrices that reside on the bottom (right). Thus, the query answer can be maintained by computing $(U^{-1})^{\text{new}}(L^{-1})^{\text{new}}[h, 0]^\top = (U^{-1} + \Delta U)(L^{-1} + \Delta L)[h, 0]^\top$. (The "heavy" part $U^{-1}L^{-1}[h, 0]^\top$ can be reused, and the remaining components are relatively cheap to calculate).

This argument implies that, at level k, we can afford time proportional to its size $r_k \leq n_k$ to rebuild the lower-upper triangular matrix L and U on their changed part along with the vector $U^{-1}L^{-1}[h, 0]^\top$, to perform fast queries. We remark that recomputing $\Delta L, \Delta U$ requires certain preprocessing which is where we exploit the assumption that updates u, v are from a fixed set.

Application to the LP setting. We now explain how the above framework can be successfully applied to the LP setting, i.e., to efficiently implement IPM algorithms. As explained in the next section, the goal of each iteration in IPM solvers is to (re-)calculate an approximate matrix-vector product r of the following form, given new disposition vectors w^{appr} and h^{appr} (see Algorithm 1):

$$r := P(w^{\text{appr}}) \cdot h^{\text{appr}} = \sqrt{W^{\text{appr}}} A^\top (A W^{\text{appr}} A^\top)^{-1} A \sqrt{W^{\text{appr}}} \cdot h^{\text{appr}}.$$

We can use our cascading lazy updates technique to maintain the middle term $(AW^{\text{appr}}A^\top)^{-1} \cdot h$, where for simplicity we assume here that h is some fixed vector. Indeed, letting w^{appr} denote the j-th iteration update $w^{(j)}$, recalculating $(AW^{\text{appr}}A^\top)^{-1} \cdot h$ corresponds to maintaining a sequence of $T_j := \|w^{(j)} - w^{(j-1)}\|_0$ of updates u_i, v_i, c_i followed by one query such that $\sum_{i=1}^{T_j} c_i \cdot u_i v_i^\top = A(W^{(j)} - W^{(j-1)})A^\top$.

Pictorially, the cascading lazy updates process resembles the following "chasing game": Child number t_{k-1} is chasing its friend t_k. Once the distance between them is too large, t_{k-1} updates his position to the position of t_k (Figure 1). This process generalizes [CLS19], in which there's only one child t_1 chasing its friend $t_2 = T$. To analyze the amortized cost of this process, we exploit the following special features of the LP problem:

1. The updates $u^{(t)}, v^{(t)}$ is some row of the original (fixed) LP matrix A.

2. w^{appr} is slowly changing in each IPM iteration, which imposes nontrivial *sparsity guarantees* that can be used to determine the cascading thresholds $\{n_k\}_{k=1}^K$: Informally speaking, since w^{appr} is roughly a martingale, the rank r_k of epoch k will be typically far less than its boundary condition ($r_k \ll t_k - t_{k-1}$) – It takes about $\sqrt{n_k}$ LP-iterations for epoch k to exceed the threshold n_k, in which case we need to update epoch $k-1$ and compute ΔL, ΔU which are of size $n \times n_{k-1}$. (see Sections 4.2, F for more details).

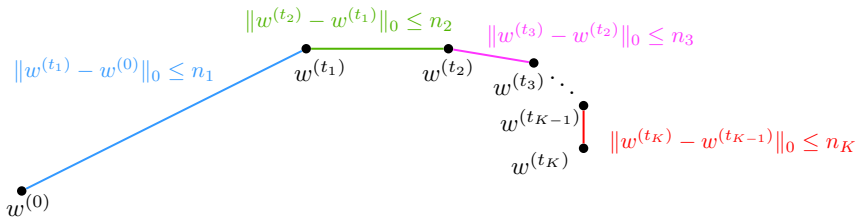


Figure 1: The cascading lazy updates process with per-level invariants $\|w^{(t_k)} - w^{(t_k-1)}\|_0 \le n_k$. Updates become more expensive but less frequent as we move down the levels.

3. The robustness of the Central Path algorithm implies that queries $M^{-1} \cdot h$ can tolerate small relative ℓ_∞ error. This allows the use of randomized compression (left-sketching $R^\top R \cdot U^{-1}L^{-1}[h,0]^\top$) to further reduce the running time (see Section 4.1 for technique overview).

Therefore, assuming ideal matrix-multiplication constants ($\omega = 2$, $\alpha = 1$) and that ΔL and ΔU can be recomputed in time linear in their sparsity $O(n \cdot n_k)$ for *every level* $k \in [K]$, the vector $(L^{-1})^{\text{new}}[h,0]^\top = (L^{-1} + \Delta L)[h,0]^\top$ can be maintained in $O(n \cdot n_k)$ time. Furthermore, the solution

$$\begin{aligned}(U^{-1})^{\text{new}}(L^{-1})^{\text{new}}[h,0]^\top &= (U^{-1} + \Delta U)(L^{-1} + \Delta L)[h,0]^\top \\ &= U^{-1}L^{-1}[h,0]^\top + \Delta U\left((L^{-1} + \Delta L)[h,0]^\top\right) + (U^{-1}\Delta L)[h,0]^\top\end{aligned}$$

can be maintained in the same $O(n \cdot n_k)$ time since the non-zero part of ΔL is an $(n_k \times n)$ block. The bottom K-th level is a special one, as every level k except the last one involves extra pre-computation to support the next $(k+1)$-th level. As we show, with some extra (non-trivial) effort, this fact enables maintaining the K-th level in only $O(n_{K-1} \cdot n_K)$ time instead of the brute-force $O(n \cdot n_K)$ time. (Our algorithm implements $K = 3$ levels of this framework. By convention, in later sections we refer to the third level as the "query level"; The first and second levels are referred as "two-level" updates).

In conclusion, the ideal time per LP-iteration according to our framework is proportional to

$$\sum_{k=1}^{K-1} \underbrace{(n \cdot n_k/\sqrt{n_{k+1}})}_{K-1 \text{ levels}} + \underbrace{n_{K-1} \cdot n_K}_{K\text{-th level}}, \tag{3}$$

where $n \cdot n_k/\sqrt{n_{k+1}}$ is the amortized update cost of level $k \in [K-1]$, and $n_{K-1} \cdot n_K$ is the update cost of the last level K. Carefully balancing the terms by setting geometrically decreasing thresholds n_k's (see Claim H.6), this runtime is optimized to be

$$\widetilde{O}(K) \cdot n^{2+\frac{1}{6 \cdot (2^{K-1}-1)}} = \widetilde{O}(K) \cdot n^{2+O(2^{-K})}, \tag{4}$$

using the fact that the LP algorithm (see Algorithm 1) has $\widetilde{O}(\sqrt{n})$ iterations. The formal calculation can be found in Section H.3. Note that when $K = 2$, this running time is precisely $O^*(n^{2+1/6})$, matching [CLS19, LSZ19, Bra20]'s results. For $K = 3$ levels, this matches our result $O^*(n^{2+1/18})$.

Achieving the runtime predicted by Eq. (4) for $K = \omega(1)$ levels would show that LPs can be solved in the ideal time $O^*(n^\omega)$. The main challenge in implementing our technique beyond $K > 3$ levels is maintaining the LU-decomposition of (2) in the desired ($\sim n \cdot n_k$) time, and indeed doing so even for $K = 3$ is highly nontrivial, as this paper shows. While completing this ambitious program is beyond the reach of this paper, we believe the cascading lazy updates framework may have applications in future developments in LP and SDP solvers, cutting plane methods, and more generally for dynamic inverse problems, hence it is worthwhile presenting it at its full generality.

3 Background

This section provides a brief overview of recent developments in LP solvers, the optimization framework of IPMs and its relation to dynamic inverse problems.

3.1 Recent developments in LP solvers

The recent work of [CLS19] improved the $O(n^{2.5})$ LP algorithm of [Vai89] to[5]

$$O^*(n^\omega + n^{2.5-a/2} + n^{1.5+a}), \tag{5}$$

where $a \leq \alpha$ is a tunable parameter, and ω, α are the fast matrix multiplication exponent and its dual, respectively. Note that for the current values of $\omega \approx 2.38$ and $\alpha \approx 0.31$, this running time is already $O^*(n^\omega)$. However, under common belief that $\omega = 2$ and $\alpha = 1$ [CKSU05, Wil12], the running time is $O^*(n^{2+1/6})$, hence there is still a polynomial gap to the ideal running time $O^*(n^2)$.

The three main ingredients in [CLS19]'s algorithm are: (i) considering a *stochastic* version of the Central Path algorithm (see Algorithm 1 below), and then leveraging the *robustness* of this algorithm to design a more efficient matrix maintenance data structure via *subsampling* (sparsification of the "gradient" vector) yielding $o(n^2)$ query time per iteration. (ii) *Lazy updates*: Delaying updates to the projection matrix (associated with the central path equation) via "soft thresholding" and analyzing their amortized performance via martingale-based potential analysis. (iii) Using fast rectangular matrix multiplication to gain extra speedup. Our data structure will also take advantage of these building blocks.

3.2 Optimization: The stochastic central-path algorithm

We use a similar optimization framework as that of [CLS19] (see Section 2.1 for a more detailed explanation and context). Roughly speaking, the Central Path (CP) algorithm maintains a primal-dual pair of vectors, $x^{(i)}$ and $s^{(i)}$, and iteratively shrinks the *duality gap* $\mu^{(i)} := \sum_{j=1}^n x_j^{(i)} s_j^{(i)}$ by $\sim (1 - 1/\sqrt{n})$ in each iteration, until converging to a feasible point ($\mu^{(i)} \approx 0$). Hence, the Central Path algorithm has a total of $O(\sqrt{n})$ iterations. In matrix notation, this algorithm essentially boils down to implementing the following iterative algorithm [CLS19]:

Algorithm 1 Stochastic Central Path

1: $i \leftarrow 1$, initialize $x, s \in \mathbb{R}^n$
2: **while** $i < \sqrt{n}$ **do** ▷ In each iteration, we hope $\mu \approx t$
3: $\quad t \leftarrow t \cdot (1 - 1/\sqrt{n})$ ▷ target decrease of duality gap
4: $\quad \mu \leftarrow x \cdot s$ ▷ actual decrease in duality gap
5: $\quad$ Compute δ_μ based on $-\frac{\mu}{\sqrt{n}}$ and the gradient $-\nabla\Psi(\mu/t - 1)$.
6: $\quad P \leftarrow \sqrt{\frac{X}{S}} A^\top (A \frac{X}{S} A^\top)^{-1} A \sqrt{\frac{X}{S}}$ ▷ *matrix inverse*, matrix-matrix mult.
7: $\quad \delta_x \leftarrow \frac{X}{\sqrt{XS}}(I - P)\frac{1}{\sqrt{XS}}\delta_\mu$, $\delta_s \leftarrow \frac{S}{\sqrt{XS}} P \frac{1}{\sqrt{XS}}\delta_\mu$ ▷ matrix-vector mult.
8: $\quad x \leftarrow x + \delta_x$, $s \leftarrow s + \delta_s$, $i \leftarrow i + 1$
9: **end while**

Here, $X = \mathrm{diag}(x)$, $S = \mathrm{diag}(s)$ are the primal and dual vectors, and $A \in \mathbb{R}^{n \times n}$ is the (fixed) LP constraint matrix. Ψ is a potential function measuring how close μ is from t (the "target" duality

[5] The running time is actually $O^*(n^\omega + n^{2.5-a/2} + n^{1.5+a} + n^{\omega a+0.5} + n^{2a+0.5})$, and the $n^{\omega a+0.5}$ and $n^{2a+0.5}$ terms also come from query time. But they are dominated by other terms. In our paper we improved not only the n^{1+a} term, but also these other two terms.

gap), and the vector δ_μ has two purposes: decreasing μ by a $(1-1/\sqrt{n})$ factor while keeping the potential function bounded. The vectors δ_x, δ_s compute the disposition of the primal and dual vectors in each iteration. P is an orthogonal *projection matrix* ($P^2 = P$ and $P = P^\top$), and the formulas $\frac{X}{\sqrt{XS}}$, $\frac{S}{\sqrt{XS}}$, $\frac{1}{\sqrt{XS}}$, and $\frac{X}{S} \in \mathbb{R}^{n\times n}$ are the diagonal matrices of the corresponding vectors.

A key observation in [CLS19] is that this algorithm is *robust to small perturbations* along the central path: Denoting by w the vector x/s, and by h the vector $\frac{\delta_\mu}{\sqrt{XS}}$, [CLS19] shows that in the above algorithm, is enough to approximately maintain $w^{\text{appr}} \approx_{\epsilon_{\text{mp}}} w, h^{\text{appr}} \approx_{\epsilon_{\text{mp}}} h$, where $\epsilon_{\text{mp}} < 1/4$ and $\approx_{\epsilon_{\text{mp}}}$ denotes coordinate-wise approximation.

3.3 Data structures: Projection maintenance

The main bottleneck of Algorithm 1 is to efficiently maintain the approximate projection matrix

$$P(w^{\text{appr}}) = \sqrt{W^{\text{appr}}} A^\top (A W^{\text{appr}} A^\top)^{-1} A \sqrt{W^{\text{appr}}}, \tag{6}$$

recalculating the queries $r := P(w^{\text{appr}})h$ on line 7, where $h := \delta_\mu/\sqrt{XS}$. There are $O(\sqrt{n})$ iterations.

Lazy updates. It was already observed in [Vai89] that since each iteration only changes the ℓ_2 mass of w by a small amount (which can be turned into an ℓ_0 sparsity guarantee by "rounding" and absorbing a small error), most of the time the queries can be answered efficiently by computing the *low-rank* incremental change in P, amortizing away the rare cases where too many coordinates of w have changed, which are handled using brute force fast matrix multiplication. As noted above, [CLS19] further used the power of fast rectangular matrix multiplication: By definition of α, for any threshold parameter $a \le \alpha$, the complexity of multiplying an $n \times n^a$ rectangular matrix by an $n^a \times n$ rectangular matrix is the same as multiplying an $n \times 1$ vector with a $1 \times n$ vector, so they only update P when *at least* n^a *coordinates* of w have changed. [CLS19] further make a clever use of *soft thresholding* on n^a, which combined with a potential function analysis, yields an amortized $O(n^{\omega-1/2} + n^{2-a/2})$ update time per iteration. Note that the $n^{2-a/2}$ term needs to be balanced with query time. [LSZ19] and [Bra20] both follow the same update scheme.

Fast queries. Computing the queries $r = P(w^{\text{appr}})h$ in each iteration from scratch takes n^2 time since the projection matrix may be very dense. The three papers [CLS19, LSZ19, Bra20] proposed different techniques for speeding up this matrix-vector multiplication to $o(n^2)$. In [CLS19, LSZ19], the authors use the idea of "iterating and sketching" [Son19], an adaptive version of the classic "sketch and solve" paradigm [CW13]. In [Bra20], the author maintains the query answer r directly, exploiting the observation that the vector h is also slowly changing (and not just updates w). Both techniques ([CLS19, Bra20]) are essentially using *sparsification* of the vector h, hence involve a "right hand side" linear operation. In contrast, [LSZ19] uses a "left-hand side" operation by *sketching the projection matrix* itself, effectively making it smaller.

Sampling on the right [CLS19]. Here the idea is to apply a $O(\sqrt{n})$-sparse *diagonal sampling matrix* $D \in \mathbb{R}^{n\times n}$ on the *right* hand side of the maintained matrix:

$$\sqrt{W^{\text{appr}}} A^\top (A W^{\text{appr}} A^\top)^{-1} A \sqrt{W^{\text{appr}}} \underbrace{D}_{\text{sample right}} h.$$

After this sampling, the vector Dh becomes $O(\sqrt{n})$-sparse, so computing the multiplication of a matrix with Dh takes $O(n^{1.5})$ time. Also, since the rank of the change in W is guaranteed to be at most n^a, there is also a $O(n^{1+a})$ term in the query time.

Sketching on the left [LSZ19]. The idea here is to apply a (subsampled) Hadamard/Fourier transform matrix [LDFU13, PSW17] $R \in \mathbb{R}^{\sqrt{n} \times n}$, by *sketching on the left* hand side:

$$\underbrace{R^\top R}_{\text{sketch left}} \sqrt{W^{\text{appr}}} A^\top (A W^{\text{appr}} A^\top)^{-1} A \sqrt{W^{\text{appr}}} h.$$

In this way the matrix RM has size $\sqrt{n} \times n$,[6] so multiplying this matrix with a vector takes $O(n^{1.5})$ time. So the final query time is also $O(n^{1.5} + n^{1+a})$. [LSZ19] algorithm is also maintaining some extra vectors, e.g., the explicit/implicit version of x, s.

Maintaining query answers [Bra20]. Brand observed that is possible to maintain not only the inverse matrix $P(w^{\text{appr}})$ via lazy updates, but also the previously computed query answers r, by observing that the vector h it also changes slowly. In each iteration, the new r is computed as:

$$r + \sqrt{W^{\text{appr}}} A^\top (A W^{\text{appr}} A^\top)^{-1} A \sqrt{W^{\text{appr}}} \underbrace{\Delta h}_{\text{change in } h}.$$

[Bra20] chooses a similar sparsity threshold n^a for the vector and updates the maintained r when Δh exceeds n^a, so Δh is guaranteed to be n^a-sparse at query time. As such, the query time is $O(n^{1+a})$. It is noteworthy that this algorithm is deterministic, as it avoids sketching/sampling altogether. Indeed, the main motivation of [Bra20] was derandomizing [CLS19].

Our approach. We show how to break the $O(n^{1+a})$ barrier for query time, by *combining* both left-hand and right-hand linear transformations on P, together with the *cascading lazy updates* technique from Section 2. Such combination is needed to further "compress" the matrix-vector multiplication when (re)calculating r. It turns out that using two sources of randomness—sampling on the right [CLS19] + sketching on the left [LSZ19]—obliterates the error analysis (which needs to be controlled to ensure convergence), and this is intuitively where [Bra20]'s deterministic alternative is useful for us as a substitute to right-hand-sampling.

4 Detailed Technical Overview

This section provides a detailed, streamlined technical overview of the proof of Theorem 1.1, which we restate formally below. This section should be understood as a self-contained extended abstract of our entire algorithm. Formal proofs of all technical claims can be found in the Appendix.

Theorem 4.1 (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 $\delta \in (0, 1]$, $\textsc{Main}(A, b, c, \delta, a, \widetilde{a})$ *(Algorithm 17) outputs* $x \geq 0$ *such that*

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

in expected time

$$O(n^{\omega + o(1)} + n^{2.5 - a/2 + o(1)} + n^{1.5 + a - \widetilde{a}/2 + o(1)} + n^{0.5 + a + (\omega - 1)\widetilde{a}}) \cdot \log(n/\delta)$$

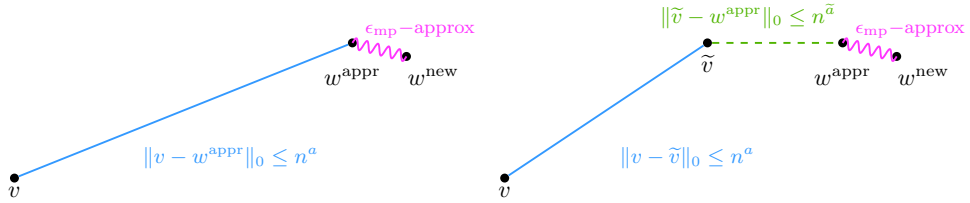
for any $0 < a \leq \alpha$ *and* $0 < \widetilde{a} \leq \alpha a$*. In particular, so long as the constants of fast matrix multiplication satisfy* $\omega > 2.055$ *and* $\alpha > 5 - 2\omega$*, general LPs can be solved in* $O(n^{\omega + o(1)})$ *time. In the ideal case that* $\omega = 2$ *and* $\alpha = 1$*, the running time is* $n^{2 + 1/18} = n^{2.055}$*.*

[6] Intuitively, $O(\sqrt{n})$ rows is the minimal sketch size one can hope for, as this ensures that with $O(\sqrt{n})$ CP iterations, every coordinate $i \in [n]$ has a constant chance to be sampled

The first two terms n^ω and $n^{2.5-a/2}$ of our running time are the same as [CLS19], stemming from the amortized cost of lazy updates (for $K = 1$ levels). The $n^{1.5+a-\widetilde{a}/2}$ term comes from the amortized cost of our cascading lazy updates algorithm for the $K = 2$nd level update. Finally, the $n^{0.5+a+(\omega-1)\widetilde{a}}$ term is the worst-case cost of the query algorithm. We note that a prerequisite for achieving the runtime of Theorem 1.1 is removing the explicit n^{1+a} term as well as the two (implicit) $n^{a\omega}$, n^{2a} terms in the query time of all previous works. Below we elaborate on how this is achieved step by step, as shown in Table 1.

Statement	Technique	Time	Ideal	Choice
[CLS19]	Sec. 3	$n^{2.5-a/2} + n^{0.5+2a} + n^{0.5+\omega a} + n^{1.5+a}$	$n^{2+1/6}$	$a = 2/3$
Thm. 4.2	Sec. 4.1.1	$n^{2.5-a/2} + n^{0.5+2a} + n^{0.5+\omega a}$	$n^{2+1/10}$	$a = 4/5$
Thm. 4.3	Sec. 4.1.2	$n^{2.5-a/2} + n^{0.5+2a}$	$n^{2+1/10}$	$a = 4/5$
Thm. 4.1	Sec. 4.1.3	$n^{2.5-a/2} + n^{1.5+a-\widetilde{a}/2} + n^{0.5+a+(\omega-1)\widetilde{a}}$	$n^{2+1/18}$	$a = 8/9, \widetilde{a} = 2/3$

Table 1: We ignore the n^ω term, and also ignore all the $n^{o(1)}$ terms. **Ideal** denotes the resulting running time when $\omega = 2$ and $\alpha = 1$. (Note that the current values are $\omega \sim 2.38$ and $\alpha \sim 0.31$).



(a) CLS19 (b) Ours

Figure 2: (a): In each iteration, we are given w^{new} which is changing slowly. The algorithm will find w^{appr} such that w^{appr} is coordinate-wise ϵ_{mp} close to w^{new}(pink wave line), and it is also close to v in ℓ_0 norm(blue solid line). (b): Based on (a), we add an intermediate level $\widetilde{v}$, such that in the query, $\|v - \widetilde{v}\|_0 \le n^a$(blue solid line) and $\|\widetilde{v} - w^{\text{appr}}\|_0 \le n^{\widetilde{a}}$(green dashed line).

We note that, although our better running time is achieved by breaking the three bottlenecks of the *query* time of previous works, it actually requires non-trivial design of both the *query* part and the *update* part. Indeed, our data structure involves an entirely new update subroutine for second-level of updates.

In the next Section 4.1, we walk through the techniques for removing the three Query bottlenecks one by one, while introducing the data structure members that must be maintained in order to make queries faster. At a high level, these members correspond to maintaining the components appearing in the LU-decomposition part of our cascading framework for $K = 2$ levels (described in Section 2, see Figure 2 for illustration). In Section 4.2 we describe and analyze the Update algorithm of our data strucutre. This part uses a combination of "soft thresholding" for two levels of updates (Figure 6), capturing the cascading lazy updates process. We remark that this type of analysis may be of independent interest to "bootstrapping" techniques in dynamic matrix problems.

4.1 Our Query Algorithm

Our dynamic data structure relies on the following three main ingredients, for removing each of the three bottlenecks shown in Table 1 respectively:

1. "Compressing" the projection matrix using linear operations on both sides (Section 4.1.1): We show that by combining a sketching on the left to reduce the matrix size from $n \times n$ to $\sqrt{n} \times n$,

with the direct maintenance of the previous query answer that makes the new query vector sparse, already breaks the n^{1+a} query time barrier of previous works.

2. Faster queries through the "cascading lazy updates" framework (Section 4.1.2): The core technique of previous works is exploiting the fact that the inverse matrix and the query vector are both changing slowly over iterations, hence the data structure can maintain all intermediate values from previous iterations, and only re-compute the differences. As explained in the last part of Section 2, we show that even the *"derivative"* of updates and queries is slowly changing (these are the "sparsity thresholds" we allude to in Section 2). It is therefore natural to maintain a second-level of values ("changes" of the inverse matrix and the query vector), where each new iteration only computes the changes in these second-level values. We update both levels according to the *cascading lazy updates* framework (recall Figure 1). We show this technique removes the $n^{a\omega}$ term in the query time of previous data structures (Table 1).

3. Maintaining the components of the second-level structure efficiently (Section 4.1.3): This is done by essentially recursing the approach of maintaining the first-level to the second-level, by carefully designing the maintained objects (matrix-products, vectors, sets). This removes the n^{2a} term in the query time, which in fact appeared in *multiple* places in previous algorithms.

We now turn to a detailed description of the query algorithm, using [CLS19] as a baseline. Recall that each iteration of the CP algorithm generates two vectors w and h from the previous outputs of the data structure. The data structure needs to re-calculate

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

By robustness of the stochastic CP algorithm, it suffices to output an approximated value

$$r = \underbrace{(R)^\top}_{\text{de-sketch}} \cdot \underbrace{R\sqrt{W^{\text{appr}}}A^\top(AW^{\text{appr}}A^\top)^{-1}A\sqrt{W^{\text{appr}}}h^{\text{appr}}}_{\text{sketch}},$$

where for some fixed parameter $\epsilon_{\text{mp}} < 1/4$ the data structure guarantees that $w^{\text{appr}} \approx_{\epsilon_{\text{mp}}} w$ and $h^{\text{appr}} \approx_{\epsilon_{\text{mp}}} h$. We use the same sketching matrix R as that of [LSZ19] which satisfies $\mathbb{E}[R^\top R] = I$ and a guarantee that the variance and ℓ_∞ error of $(R^\top RPh - Ph)$ are both small. R has size $\sqrt{n} \times n$ – this value is natural, since we have $\sim \sqrt{n}$ CP iterations, hence using $o(\sqrt{n})$ linear measurements would fail to even detect changes in all n coordinates. Also, the size of R allows us to pre-batch $O(\sqrt{n})$ copies of sketching matrices in the beginning, which only takes $O(n^\omega)$ time.

As in [CLS19, Bra20], our data structure maintains a member v that serves as a proxy for w, and a member g that serves as a proxy for h. If the new value w^{appr} is too far from v, then in addition to computing the new displacement r, the data structure also updates v along with all of its members that depend on v. An analogous scheme is used for g w.r.t h^{appr}. Since the new values w^{appr} and h^{sample} show up in three places in the output r, from now on we will refer to these three places as the left part, the middle part, and the right part, and label them as $a^{\text{new}} \in \mathbb{R}^{\sqrt{n}\times n}$, $b^{\text{new}} \in \mathbb{R}^{n\times n}$, $c^{\text{new}} \in \mathbb{R}^n$ for ease of presentation. Accordingly, we use $a \in \mathbb{R}^{\sqrt{n}\times n}$, $b \in \mathbb{R}^{n\times n}$, $c \in \mathbb{R}^n$ to denote the values that depend on v and g:

$$\underbrace{R\sqrt{W^{\text{appr}}}}_{a^{\text{new}}}\underbrace{A^\top(AW^{\text{appr}}A^\top)^{-1}A}_{b^{\text{new}}}\underbrace{\sqrt{W^{\text{appr}}}h^{\text{appr}}}_{c^{\text{new}}} \quad , \quad \underbrace{R\sqrt{V}}_{a}\underbrace{A^\top(AVA^\top)^{-1}A}_{b}\underbrace{\sqrt{V}g}_{c}.$$

We also denote $\partial a := a^{\text{new}} - a$, $\partial b := b^{\text{new}} - b$, $\partial c := c^{\text{new}} - c$.

For a tunable parameter $a \in (0, \alpha]$, the worst case query time per iteration of [CLS19]'s data structure for implementing this process is $t_q = n^{2a} + n^{\omega a} + n^{1+a}$, the three terms come from the

cost of recomputing the query r. In the remainder of this subsection, we describe how this query time can be improved to

$$t_q = n^{a+(\omega-1)\widetilde{a}} \tag{7}$$

for any $a \in (0, \alpha]$ and $\widetilde{a} \in (0, \alpha a]$. We show this in three steps, removing the terms n^{1+a}, $n^{\omega a}$, and n^{2a} one by one.

4.1.1 Technique for removing n^{1+a}

We combine the "sketching on the left" technique of [LSZ19] and the "query maintenance" technique of [Bra20] to remove this term. In [Bra20] the query is computed as

$$r = \underbrace{\sqrt{W^{\mathrm{appr}}}}_{a^{\mathrm{new}}}\Big(\underbrace{\beta_2}_{bc} + \underbrace{M}_{b}\underbrace{\big(\sqrt{W^{\mathrm{appr}}}h^{\mathrm{appr}} - \sqrt{V}g\big)}_{\partial c} + \underbrace{\big(-M_S(\Delta_{S,S}^{-1} + M_{S,S})^{-1}(M_S)^\top\big)}_{\partial b}\underbrace{\sqrt{W^{\mathrm{appr}}}h^{\mathrm{appr}}}_{c^{\mathrm{new}}}\Big),$$

where $M := A^\top(AVA^\top)^{-1}A$, $\beta_2 := M\sqrt{V}g \in \mathbb{R}^n$ are members that the data structure maintains, $\Delta := W^{\mathrm{appr}} - V$, and $S := \mathrm{supp}(w^{\mathrm{appr}} - v)$. The subscript M_S means taking the sub matrix of columns of M in the set S. Note that this output is $a^{\mathrm{new}}(bc + b\cdot\partial c + \partial b\cdot c^{\mathrm{new}}) = a^{\mathrm{new}}b^{\mathrm{new}}c^{\mathrm{new}}$.

The n^{1+a} term shows up in three places when computing different terms:

- In $a^{\mathrm{new}}\cdot b\cdot\partial c$, multiplying a $n\times n$ matrix M with a n^a-sparse vector $\big(\sqrt{W^{\mathrm{appr}}}h^{\mathrm{appr}} - \sqrt{V}g\big)$.
- In $a^{\mathrm{new}}\cdot\partial b\cdot c^{\mathrm{new}}$, multiplying a $n\times n^a$ matrix M_S with a $n^a\times 1$ vector $(\Delta_{S,S}^{-1}+M_{S,S})^{-1}(M_S)^\top\sqrt{W^{\mathrm{appr}}}h^{\mathrm{appr}}$.
- In $a^{\mathrm{new}}\cdot\partial b\cdot c^{\mathrm{new}}$, multiplying a $n^a\times n$ matrix $(M_S)^\top$ with a $n\times 1$ vector $\sqrt{W^{\mathrm{appr}}}h^{\mathrm{appr}}$.

The last one is easy, and we deal it by splitting $c^{\mathrm{new}} \in \mathbb{R}^n$ into $c + \partial c$ again:

$$(M_S)^\top\underbrace{\sqrt{W^{\mathrm{appr}}}h^{\mathrm{appr}}}_{c^{\mathrm{new}}} = (\beta_2)_S + (M_S)^\top\underbrace{\big(\sqrt{W^{\mathrm{appr}}}h^{\mathrm{appr}} - \sqrt{V}g\big)}_{\partial c}.$$

Since the ∂c term is n^a-sparse, this computation only takes n^{2a} time now.

We deal with the first two n^{1+a} terms by adding the sketching matrix R on the left, and splitting a^{new} into $a + \partial a$. Aside from maintaining $M = A^\top(AVA^\top)^{-1}A$ and $\beta_2 = M\sqrt{V}g$, we further maintain their sketched versions:

$$Q = R\sqrt{V}M \in \mathbb{R}^{\sqrt{n}\times n}, \quad \beta_1 = R\sqrt{V}\beta_2 \in \mathbb{R}^{\sqrt{n}}.$$

In addition to the temporary variables $S = \mathrm{supp}(w^{\mathrm{appr}} - v)$ and $\Delta = W^{\mathrm{appr}} - V$, we also define $\Gamma := \sqrt{W^{\mathrm{appr}}} - \sqrt{V}$. We have the guarantee that Δ, Γ and $(\sqrt{W^{\mathrm{appr}}}h^{\mathrm{appr}} - \sqrt{V}g)$ are all n^a-sparse and $|S| \le n^a$, because otherwise the algorithm would first update V and g. The query part of our new data structure becomes

$$r_1 := \underbrace{\beta_1}_{abc}, \qquad r_2 := \underbrace{(Q_S}_{ab} + \underbrace{R\Gamma M)}_{\partial a\cdot b}\underbrace{\big(\sqrt{W^{\mathrm{appr}}}h^{\mathrm{appr}} - \sqrt{V}g\big)}_{\partial c}, \qquad r_3 := \underbrace{R\Gamma}_{\partial a}\cdot\underbrace{\beta_2}_{bc}$$

$$r_4 := \underbrace{(Q_S + R\Gamma\cdot\overbrace{M_S}^{\partial b})(-(\Delta_{S,S}^{-1} + M_{S,S})^{-1})\Big((M_S)^\top\cdot\big(\sqrt{W^{\mathrm{appr}}}h^{\mathrm{appr}} - \sqrt{V}g\big) + \beta_{2,S}\Big)}_{(M_S)^\top\cdot c^{\mathrm{new}}}$$

Note that this output is

$$r := \underbrace{abc}_{r_1} + \underbrace{(a+\partial a)b\cdot\partial c}_{r_2} + \underbrace{\partial a\cdot bc}_{r_3} + \underbrace{a^{\text{new}}\cdot\partial b\cdot c^{\text{new}}}_{r_4} = a^{\text{new}}b^{\text{new}}c^{\text{new}}.$$

Recall that the n^{1+a} term stemmed from multiplying an $n\times n$ matrix by an n^a-sparse vector when computing $a^{\text{new}}\cdot b\cdot\partial c$. We split this term as $a^{\text{new}}\cdot b\cdot\partial c := a\cdot b\cdot\partial c + \partial a\cdot b\cdot\partial c$ (see the formula for r_2). The data structure will now maintain $ab := Q_S$, whose size is only $\sqrt{n}\times n^a$, hence computing $ab\cdot\partial c$ only takes $n^{1/2+a} < n^{1.5}$ time now. Note that when computing $\partial a\cdot b\cdot\partial c$, the $n\times n$ matrix M is "sandwiched" by a n^a-sparse diagonal matrix Γ on the left and a n^a-sparse vector $(\sqrt{W^{\text{appr}}}h^{\text{appr}} - \sqrt{V}g)$ on the right, thus computing the product $\Gamma M(\sqrt{W^{\text{appr}}}h^{\text{appr}} - \sqrt{V}g)$ takes n^{2a} time.

The last n^{1+a} bottleneck from [CLS19] is removed in a similar way, yielding the following intermediate result:

Theorem 4.2 (Informal, first improvement)**.** *For any $a\le\alpha$, there is a randomized algorithm for solving general LPs in expected time $O^*(n^\omega + n^{2.5-a/2} + n^{0.5+a\omega})$.*

Note that for $\omega = 2$ and $\alpha = 1$, this approach already yields an improved $n^{2+1/10}$ LP algorithm.

4.1.2 Technique for removing $n^{\omega a}$

The $n^{\omega a}$ term in previous data structures came from computing the inverse of an $n^a\times n^a$ matrix $(\Delta_{S,S}^{-1}+M_{S,S})$, and this inverse is still present in the intermediate algorithm described in Section 4.1.1 (see the formula for r_4). This is where the *cascading lazy updates* technique comes useful – we shall remove the $n^{\omega a}$ by showing how to implement it for $K = 2$ "levels". We now provide the details of this data structure.

Once again, the main observation is that since we've already computed $(\Delta_{S,S}^{-1} + M_{S,S})^{-1}$ in previous iterations, we do not need to re-compute it from scratch. Instead, we only need to compute the difference between the new inverse matrix and the old one. More concretely, we maintain a second-level data structure member $\widetilde{v}$. $\widetilde{v}$ keeps a *closer* distance with w^{appr} than v, and is therefore updated more frequently. We update v whenever $\|\widetilde{v}-v\|_0 > n^a$ (for some $a\le\alpha$) and update $\widetilde{v}$ whenever $\|w^{\text{appr}}-\widetilde{v}\| > n^{\widetilde{a}}$ (for some $\widetilde{a}\le\alpha a$). By abuse of notation, we define

$$S := \text{supp}(\widetilde{v}-v),\quad S^{\text{new}} := \text{supp}(w^{\text{appr}}-v),\quad \partial S := \text{supp}(w^{\text{appr}}-\widetilde{v}). \tag{8}$$

In this overview we can think of $S^{\text{new}} = S\cup\partial S$. Later we also handle the possibly non-empty set $S' = (S\cup\partial S)\backslash S^{\text{new}}$, but the key ideas are the same. The updates guarantee that in the query we always have that $|S^{\text{new}}|\le n^a$ and $|\partial S|\le n^{\widetilde{a}}$. Our data structure maintains a second-level member

$$B := (\Delta_{S,S}^{-1} + M_{S,S})^{-1}\in\mathbb{R}^{n^a\times n^a}.$$

Observe that the new matrix $((\Delta^{\text{new}})^{-1}_{S^{\text{new}},S^{\text{new}}} + M_{S^{\text{new}},S^{\text{new}}})$ only differs from $B^{-1} = (\Delta_{S,S}^{-1}+M_{S,S})$ on entries in $S^{\text{new}}\times\partial S$ and $\partial S\times S^{\text{new}}$. (See the left part of Figure 3.)

So we have the following decomposition that can be computed in $O(n^{\widetilde{a}+a})$ time:

$$U'CU^\top = ((\Delta^{\text{new}})^{-1}_{S^{\text{new}},S^{\text{new}}} + M_{S^{\text{new}},S^{\text{new}}}) - (\Delta_{S,S}^{-1} + M_{S,S}),$$

where $U', U\in\mathbb{R}^{n^a\times n^{\widetilde{a}}}$, and $C\in\mathbb{R}^{n^{\widetilde{a}}\times n^{\widetilde{a}}}$. In fact U' and U are both constructed by taking a submatrix from M and concatenate it with two identity matrices (see Figure 3), this will be useful in the next section where we remove the n^{2a} term. Now we can use Woodbury identity to compute

$$((\Delta^{\text{new}})^{-1}_{S^{\text{new}},S^{\text{new}}} + M_{S^{\text{new}},S^{\text{new}}})^{-1} = (B^{-1} + U'CU^\top)^{-1} = B - BU'(C^{-1} + U^\top BU')^{-1}U^\top B.$$

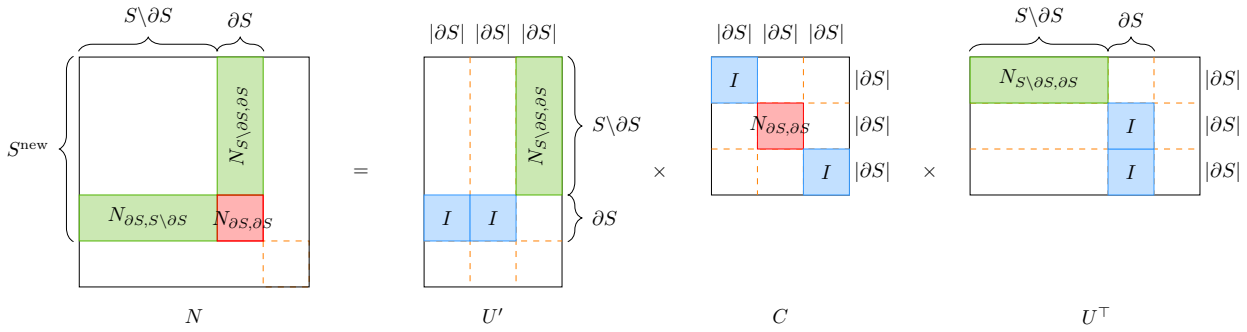


Figure 3: An illustration of the construction of $U', C, U^\top$. N is defined as $((\Delta^{\text{new}})^{-1}_{S^{\text{new}},S^{\text{new}}} + M_{S^{\text{new}},S^{\text{new}}}) - (\Delta^{-1}_{S,S} + M_{S,S})$. I denotes the identity matrix.

The most expensive part of this formula is to compute the multiplication $U^\top B$ and BU', and the inverse $(C^{-1} + U^\top BU')^{-1}$. Since $\widetilde{a} \leq \alpha a$, using fast rectangular matrix multiplication, multiplying a $n^{\widetilde{a}} \times n^a$ matrix $U^\top$ with a $n^a \times n^a$ matrix B takes $O(n^{2a})$ time. Computing BU' takes the same time. Computing the inverse of a $n^{\widetilde{a}} \times n^{\widetilde{a}}$ matrix $(C^{-1} + U^\top BU')$ takes $n^{\widetilde{a}\omega} = \mathcal{T}_{\text{mat}}(n^{\widetilde{a}}, n^{\widetilde{a}}, n^{\widetilde{a}}) \leq \mathcal{T}_{\text{mat}}(n^a, n^a, n^{\widetilde{a}}) = n^{2a}$. All other parts of the query algorithm remain the same as in Section 4.1.1. Hence, so far, the running time is upper bounded by $O(n^{2a})$:

Theorem 4.3 (informal, second improvement)**.** *For any $a \leq \alpha$, there is a randomized algorithm for solving general LPs in expected time $O^*(n^\omega + n^{2.5-a/2} + n^{0.5+2a})$.*

We remark that the second-level members B and the local variables U, C and U' that the data structure maintains, correspond to the variables in the cascading lazy update framework of Section 2: The matrix B here is exactly the same inverse B as defined in Section 2 for $K = 2$ (see Eq.(70)). In the same vein, the block C^{-1} here corresponds to the term $-C_2^{-1} - V_2^\top M^{-1} U_2$, $U^\top$ corresponds to $V_2^\top M^{-1} U_1$, and U' corresponds to $V_1^\top M^{-1} U_2$ of Eq.(70). That said, since Section 2 deals with a simplified version of our actual inverse problem, we will need to maintain several other ad-hoc members to achieve the claimed running time. We turn to describe those next.

4.1.3 Technique for removing n^{2a}

Now we show how to remove the n^{2a} term from the current algorithm as described in Section 4.1.1, with the inverse matrix of r_4 computed as described in Section 4.1.2). The n^{2a} term appears in the computation of both r_2 and r_4. Since removing the n^{2a} terms in r_4 is more difficult, we mainly focus on the r_4 term in this section. In analogy to the second-level member $\widetilde{v}$, we also maintain $\widetilde{g}$, and update it whenever $\|h^{\text{appr}} - \widetilde{g}\|_0 > n^{\widetilde{a}}$. Similar to the definition of $S^{\text{new}}, S, \partial S$ (Eq. 8), we shall use the following three variants of notations:

$$\begin{aligned}
\Delta^{\text{new}} &= W^{\text{appr}} - V, & \Delta &= \widetilde{V} - V, & \partial\Delta &= W^{\text{appr}} - \widetilde{V}, \\
\Gamma^{\text{new}} &= \sqrt{W^{\text{appr}}} - \sqrt{V}, & \Gamma &= \sqrt{\widetilde{V}} - \sqrt{V}, & \partial\Gamma &= \sqrt{W^{\text{appr}}} - \sqrt{\widetilde{V}}, \\
\xi^{\text{new}} &= \sqrt{W^{\text{appr}}} h^{\text{appr}} - \sqrt{V} g, & \xi &= \sqrt{\widetilde{V}} \widetilde{g} - \sqrt{V} g, & \partial\xi &= \sqrt{W^{\text{appr}}} h^{\text{appr}} - \sqrt{\widetilde{V}} \widetilde{g}.
\end{aligned}$$

Intuitively, for $X \in \{S, \Delta, \Gamma, \xi\}$, X^{new} represents the difference between w^{appr} and the first-level proxy v, this is the real "first derivative", and it is what we need to compute the output; X represents the difference between the first-level proxy v and the second-level proxy $\widetilde{v}$, this is what we maintain

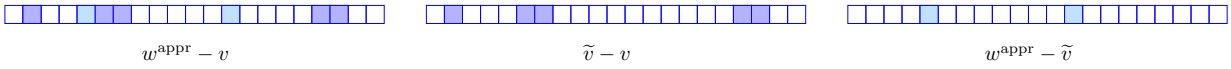


Figure 4: The three variants of notations. The left vector $w^{\text{appr}} - v$ corresponds to the notations X^{new}, and $|\operatorname{supp}(w^{\text{appr}} - v)| \leq n^a$. The middle vector $\widetilde{v} - v$ corresponds to the notations X, and $|\operatorname{supp}(\widetilde{v} - v)| \leq n^a$. The right vector $w^{\text{appr}} - \widetilde{v}$ corresponds to the notations ∂X, and $|\operatorname{supp}(w^{\text{appr}} - \widetilde{v})| \leq n^{\widetilde{a}}$.

in the data structure, and we can think of it as an out-of-date "first derivative"; ∂X represents the difference between w^{appr} and the second-level proxy $\widetilde{v}$. This term is very sparse, and should be thought of as the "second derivative". The actual "first derivative" is the sum of the outdated "first derivative" plus the "second derivative". We can therefore split X^{new} as $X + \partial X$ when computing the output. In this way we exploits both the maintained members of the data structure and the *sparsity* of ∂X, compares to computing X^{new} from scratch.

We now turn to formalize the above intuition. Our data structure maintains the second-level members $S \subseteq [n], \Delta \in \mathbb{R}^{n \times n}, \Gamma \in \mathbb{R}^{n \times n}, \xi \in \mathbb{R}^n$. By design, our update algorithm ensures that

$$|S^{\text{new}}|, \|\Delta^{\text{new}}\|_0, \|\Gamma^{\text{new}}\|_0, \|\xi^{\text{new}}\|_0, |S|, \|\Delta\|_0, \|\Gamma\|_0, \|\xi\|_0 \leq n^a; \quad |\partial S|, \|\partial\Delta\|_0, \|\partial\Gamma\|_0, \|\partial\xi\|_0 \leq n^{\widetilde{a}}.$$

Now, recall that from Section 4.1.1 and Section 4.1.2, r_4 is computed as follows

$$r_4 = -\Big(Q_{S^{\text{new}}} + R\underbrace{\Gamma^{\text{new}} M_{S^{\text{new}}}}_{3}\Big) \cdot \Big(\underbrace{BU'}_{2}(C^{-1} + U^\top \underbrace{BU'}_{2})^{-1}U^\top - I\Big) \cdot \underbrace{B\Big((\beta_2)_{S^{\text{new}}} + (M_{S^{\text{new}}})^\top \xi^{\text{new}}\Big)}_{1},$$

where $B = ((\Delta_{S,S})^{-1} + M_{S,S})^{-1}$ and $U'CU^\top = ((\Delta^{\text{new}})^{-1}_{S^{\text{new}},S^{\text{new}}} + M_{S^{\text{new}},S^{\text{new}}})^{-1} - B$. Observe that in the above formula the n^{2a} term appears in the following places:

1. Multiplying a $n^a \times n^a$ matrix B with a $n^a \times 1$ vector $(\beta_2)_{S^{\text{new}}} + (M_{S^{\text{new}}})^\top \xi^{\text{new}}$.
2. Multiplying a $n^a \times n^a$ matrix B with a $n^a \times n^{\widetilde{a}}$ matrix U'.
3. Multiplying a n^a-sparse diagonal matrix Γ^{new} with a $n \times n^a$ matrix $M_{S^{\text{new}}}$ and then with a $n^a \times 1$ vector that comes from later terms.

To remove these n^{2a} terms, we maintain additional second-level data structure members (precomputed matrix products):

$$\begin{aligned} r_4 = &-\Big(Q_{S^{\text{new}}} + \underbrace{R\Gamma M_S}_{F\text{: member}} + R(\Gamma M_{\partial S\setminus S} + \partial\Gamma M_{S^{\text{new}}})\Big) \cdot \Big((\underbrace{BU'}_{U^{\text{tmp}}}(C^{-1} + U^\top \underbrace{BU'}_{U^{\text{tmp}}})^{-1}U^\top) - I\Big) \cdot \\ &\Big(\underbrace{B(\beta_2)_S + B(M_S)^\top \cdot \xi}_{\gamma_1\text{: member}} + B(\beta_2)_{\partial S\setminus S} + B(M_{\partial S\setminus S})^\top \xi^{\text{new}} + \underbrace{B(M_S)^\top}_{E\text{: member}} \partial\xi\Big) \end{aligned}$$

Each of the previous n^{2a} terms are removed as follows.

1. We maintain $\gamma_1 := B(\beta_2)_S + B(M_S)^\top \xi \in \mathbb{R}^{n^a}$ so that we only need to compute the difference

$$(B(\beta_2)_{S^{\text{new}}} + B(M_{S^{\text{new}}})^\top \xi^{\text{new}}) - \gamma_1 = B(\beta_2)_{\partial S\setminus S} + B(M_{\partial S\setminus S})^\top \xi^{\text{new}} + B(M_S)^\top \partial\xi.$$

   Since $(\beta_2)_{\partial S\setminus S}$ is $n^{\widetilde{a}}$-sparse, and $(M_{\partial S\setminus S})^\top$ only has $n^{\widetilde{a}}$ non-zero rows, the first two terms $B(\beta_2)_{\partial S\setminus S}$ and $B(M_{\partial S\setminus S})^\top \xi^{\text{new}}$ can both be computed in $O(n^{a+\widetilde{a}})$ time. For the third term, we also maintain $E := B(M_S)^\top \in \mathbb{R}^{n^a \times n^a}$. Since $\partial\xi$ is also $n^{\widetilde{a}}$-sparse, it takes $O(n^{a+\widetilde{a}})$ time to compute this term as well.

2. The construction of U' has the following property: $BU' = [B_{(\partial S \setminus S)}, B_{\partial S}, BM_{S,(\partial S \setminus S)}]$. Since we already maintain $E := B(M_S)^\top \in \mathbb{R}^{n^a \times n}$, we define a local variable $U^{\text{tmp}} := [B_{(\partial S \setminus S)}, B_{\partial S}, E_{(\partial S \setminus S)}] \in \mathbb{R}^{n^a \times 3n^{\widetilde{a}}}$. Then $U^{\text{tmp}} = BU'$, and it can be computed in the same time as its size, which is $O(n^{a+\widetilde{a}})$.

3. We maintain $F := R\Gamma M_S \in \mathbb{R}^{\sqrt{n} \times n^a}$, and the difference is $R\Gamma^{\text{new}} M_{S^{\text{new}}} - F = R(\Gamma M_{\partial S \setminus S} + \partial\Gamma M_{S^{\text{new}}})$. Multiplying F with a $n^a \times 1$ vector that comes from later terms only takes $n^{1/2+a} < n^{1.5}$ time. Also since $M_{\partial S \setminus S}$ only has $n^{\widetilde{a}}$ non-empty columnns, and $\partial\Gamma$ is $n^{\widetilde{a}}$-sparse, multiplying $(\Gamma M_{\partial S \setminus S} + \partial\Gamma M_{S^{\text{new}}})$ with the $n^a \times 1$ vector that comes from later terms takes $n^{a+\widetilde{a}}$ time.

A full list of all the second-level members that we maintain is as follows:

$$1.S = \text{supp}(\widetilde{v} - v), \quad 2.\Delta = \widetilde{V} - V, \quad 3.\Gamma = \sqrt{\widetilde{V}} - \sqrt{V}, \quad 4.\xi = \sqrt{\widetilde{V}}\widetilde{g} - \sqrt{V}g,$$
$$5.B = (\Delta_{S,S}^{-1} + M_{S,S})^{-1}, \quad 6.E = B(M_S)^\top, \quad 7.F = R\Gamma M_S, \quad 8.\gamma_1 = B(\beta_2)_S + B(M_S)^\top \xi.$$

Now whenever our data structure needs to multiply an $n^a \times n^a$ matrix with a $n^a \times 1$ vector, it is always the case that either the vector is $n^{\widetilde{a}}$-sparse, or the matrix only has $n^{\widetilde{a}}$ rows, so in both cases this operation takes $O(n^{a+\widetilde{a}})$ time. We also avoid multiplying the $n^a \times n^a$ matrix B with the $n^{\widetilde{a}} \times n^a$ matrix U' directly by maintaining $E = B(M_S)^\top$, and we can now extract $U^{\text{tmp}} = BU'$ from E efficiently. But still we need to multiply a $n^{\widetilde{a}} \times n^a$ matrix $U^\top$ with a $n^a \times n^{\widetilde{a}}$ matrix U^{tmp}, which takes time $\mathcal{T}_{\text{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}) \leq n^{a-\widetilde{a}} \cdot \mathcal{T}_{\text{mat}}(n^{\widetilde{a}}, n^{\widetilde{a}}, n^{\widetilde{a}}) = n^{a+(\omega-1)\widetilde{a}}$. This is our final running time for query, as presented in the last line of Table 1.

Finally we remark that removing the last n^{2a} term that stems from computing r_2, can be done in an analogous way to the usage of γ_1 for r_4 (i.e., by maintaining an additional member $\gamma_2 := \Gamma M \xi$). We omit the formal details here.

Thus we finished the proof of how to get the $O(n^{a+(\omega-1)\widetilde{a}})$ query time of Theorem 4.1.

4.2 Our Update Algorithm

Bounding the running time of our two-level update scheme requires both algorithmic modifications and a more sophisticated amortized analysis than that of [CLS19], in order to capture the cascading lazy updates process (as a random process under the sketching of the CP). This section is organized as follows. In Section 4.2.1 we describe the four update subroutines for maintaining the two-level members for both w and h, and present their worst-case running time per call in Table 2. These subroutines correspond to maintaining the LU-decomposition of the cascading updates algorithm for a $K = 3$ block matrix, described in Section 2 (see Figure 2). Section 4.2.2 describes the main algorithm deciding when to call each of these four subroutines, using "two-level soft thresholding" (see Figure 6). In order to synchronize two levels of soft thresholding, we introduce a new Adjust function. Finally, Section 4.2.3 describes a potential-based amortized analysis of our update algorithm, and the final amortized running time of the four update subroutines is presented in Table 3.

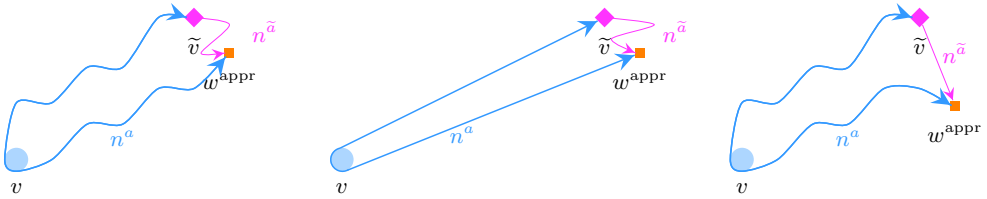
4.2.1 Cascading updates subroutines

We now describe the four subroutines required to efficiently implement the cascading lazy updates process for $K = 3$ levels as described in the previous section. Recall our data structure maintains two levels of proxies v and $\widetilde{v}$ for the input w: v is the first-level proxy, and $\widetilde{v}$ is the second-level proxy. v keeps a larger distance of n^a with w and is updated less frequently, while $\widetilde{v}$ keeps a smaller distance of $n^{\widetilde{a}}$ with w and is updated more frequently. We define two subroutines to update v and $\widetilde{v}$ (see Figure 5 for illustration).

Level	Name	Time per call	Rank/Sparsity	Comment
1	Matrix	$\mathcal{T}_{\text{mat}}(n, n, k)$	$k := \|v^{\text{new}} - v\|_0$	Update v and $\widetilde{v}$ if $\|w^{\text{appr}} - v\|_0 \geq n^a$
2	PartialMatrix	$\mathcal{T}_{\text{mat}}(n, n^a, \widetilde{k})$	$\widetilde{k} := \|\widetilde{v}^{\text{new}} - \widetilde{v}\|_0$	Update $\widetilde{v}$ if $\|w^{\text{appr}} - \widetilde{v}\|_0 \geq n^{\widetilde{a}}$
1	Vector	$pn + n^{2a}$	$p := \|g^{\text{new}} - g\|_0$	Update g and $\widetilde{g}$ if $\|h^{\text{appr}} - g\|_0 \geq n^a$
2	PartialVector	$\widetilde{p}n + n^{2a}$	$\widetilde{p} := \|\widetilde{g}^{\text{new}} - \widetilde{g}\|_0$	Update $\widetilde{g}$ if $\|h^{\text{appr}} - \widetilde{g}\|_0 \geq n^{\widetilde{a}}$

Table 2: Four update procedures

The first subroutine is PartialMatrixUpdate, which corresponds to second-level updates we alluded to in Section 2: so long as the rank of the updates is smaller than the second level threshold $n^{\widetilde{a}}$, it suffices to only update the second level members as defined in Section 4.1.3. These members are relatively cheap to update, and thus PartialMatrixUpdate has a cost of $\mathcal{T}_{\text{mat}}(n, n^a, \widetilde{k})$ per call (third column of Table 2). When the algorithm has exceeded the allowable changes in w, we cascade to the first level update subroutine MatrixUpdate. This subroutine must update all data structure members, and consequently is more expensive: it has has a cost of $\mathcal{T}_{\text{mat}}(n, n, k)$ per call (third column of Table 2). The subroutines VectorUpdate and PartialVectorUpdate play an analogous role for updating h. We proceed to describe when to execute each of these subroutines.



(a). No update. (b). Matrix update. (c). Par. Matrix update.

Figure 5: An illustration of the two types of updates in our Update algorithm. The blue leash connects v with $\widetilde{v}$ and w^{appr} and is of length n^a. The pink leash connects $\widetilde{v}$ with w^{appr} and is of length $n^{\widetilde{a}}$. (a) When all leashes are loose, no update is performed. (b) The leash on v is tight. In this case we call MatrixUpdate. (c) The leash on $\widetilde{v}$ is tight. In this case we call PartialMatrixUpdate.

4.2.2 Synchronizing two-level soft thresholding

For simplicity, we only focus on MatrixUpdate and PartialMatrixUpdate (VectorUpdate and PartialVectorUpdate are analogous).

In order to bound the amortized cost of the two level updates, we use the "soft thresholding" implicit[7] in [CLS19] to determine when to invoke each update. The basic idea is to use a smooth threshold (see Algorithm 7) to approximate the discrete threshold shown in the last column of Table 2. Soft thresholding ensures that there is a gap between errors of coordinates that are updated and the errors of other coordinates, and is essential to guarantee a proper decrease in potential. Our update algorithm invokes this subroutine SoftThreshold twice – once for $\widetilde{v}$ and once for v (see Figure 6). We now explain the 3 main components of combining the two SoftThreshold subroutines (see Algorithm 2):

Restoring threshold gaps via Adjust. Our algorithm first computes $\widetilde{v}^{\text{new}}$ to update all the coordinates i for which $|w_i^{\text{new}}/\widetilde{v}_i - 1|$ exceeds the threshold ϵ_{mp}. It then calls the Adjust function

[7]The soft thresholding is proposed by [CLS19], and note that they embed it inside their update function since they only have one update function. Since we use it four times, we give it an explicit name SoftThreshold.

Algorithm 2 When to execute MatrixUpdate and PartialMatrixUpdate

$\widetilde{v}^{\text{new}} \leftarrow \textsc{SoftThreshold}(y \leftarrow |w^{\text{new}}/\widetilde{v} - 1|, w^{\text{new}}, \widetilde{v}, \epsilon_{\text{mp}}, n^{\widetilde{a}})$
$\textsc{Adjust}(\widetilde{v}^{\text{new}}, \epsilon_{\text{mp}}/(100 \log n))$
if $\|\widetilde{v}^{\text{new}} - v\|_0 \geq n^a$ **then**
    $v^{\text{new}} \leftarrow \textsc{SoftThreshold}(y \leftarrow |w^{\text{new}}/v - 1| + |w^{\text{new}}/\widetilde{v} - 1|, w^{\text{new}}, v, \frac{\epsilon_{\text{mp}}}{100 \log^2 n}, n^a)$
    $w^{\text{appr}} \leftarrow v^{\text{new}}$
    $\textsc{MatrixUpdate}()$
else
    $w^{\text{appr}} \leftarrow \widetilde{v}^{\text{new}}$
    if $\|\widetilde{v}^{\text{new}} - \widetilde{v}\|_0 \geq n^{\widetilde{a}}$ **then**
        $\textsc{PartialMatrixUpdate}()$
    end if
end if

to restore all updated coordinates $\widetilde{v}_i^{\text{new}}$ whose new value $\widetilde{v}_i^{\text{new}}$ is within a distance of $\epsilon_{\text{mp}}/(100 \log n)$ from v_i, back to the original value v_i. Since $\widetilde{v}^{\text{new}}$ is the new value of $\widetilde{v}$, in this way we ensure that $|\widetilde{v}_i - v_i| > \epsilon_{\text{mp}}/(100 \log n)$ for all $i \in \text{supp}(\widetilde{v} - v)$. Hence when $\|\widetilde{v} - v\|_0$ exceeds its threshold, there is a large enough decrease in our potential function ($\approx \|\widetilde{v} - v\|_0 \cdot \epsilon_{\text{mp}}/(100 \log n)$) for "charging" the update cost (of $v \leftarrow \widetilde{v}$). The purpose of using a smaller threshold-error of $\epsilon_{\text{mp}}/(100 \log n)$ here ensures that even when $\widetilde{v}_i^{\text{new}}$ (which is the new value of $\widetilde{v}$) is v_i instead of w_i^{new}, the error still decreases by at least a $(1 - 1/\log n)$ factor after updating $\widetilde{v} \leftarrow \widetilde{v}^{\text{new}}$.

Synchronizing the error function. When updating v, we define the error as $|w^{\text{new}}/v - 1| + |w^{\text{new}}/\widetilde{v} - 1|$ which is a function of both v and $\widetilde{v}$. This is because as long as *one* of v_i and $\widetilde{v}_i$ is too far from w_i^{new}, we need to update both variables to be the same as w_i^{new} .

Two error thresholds. When updating v, we use a smaller error threshold of $\epsilon_{\text{mp}}/(100 \log^2 n)$ than that of the Adjust threshold. This is because Adjust guarantees that $v_i - \widetilde{v}_i \geq \epsilon_{\text{mp}}/(100 \log n)$ on all coordinates i for which $v_i \neq \widetilde{v}_i$, hence using an even smaller threshold when updating v ensures that all such coordinates are counted as "error larger than threshold". As such, after MatrixUpdate, $\widetilde{v}$ is set back to be the same as v on all coordinates.

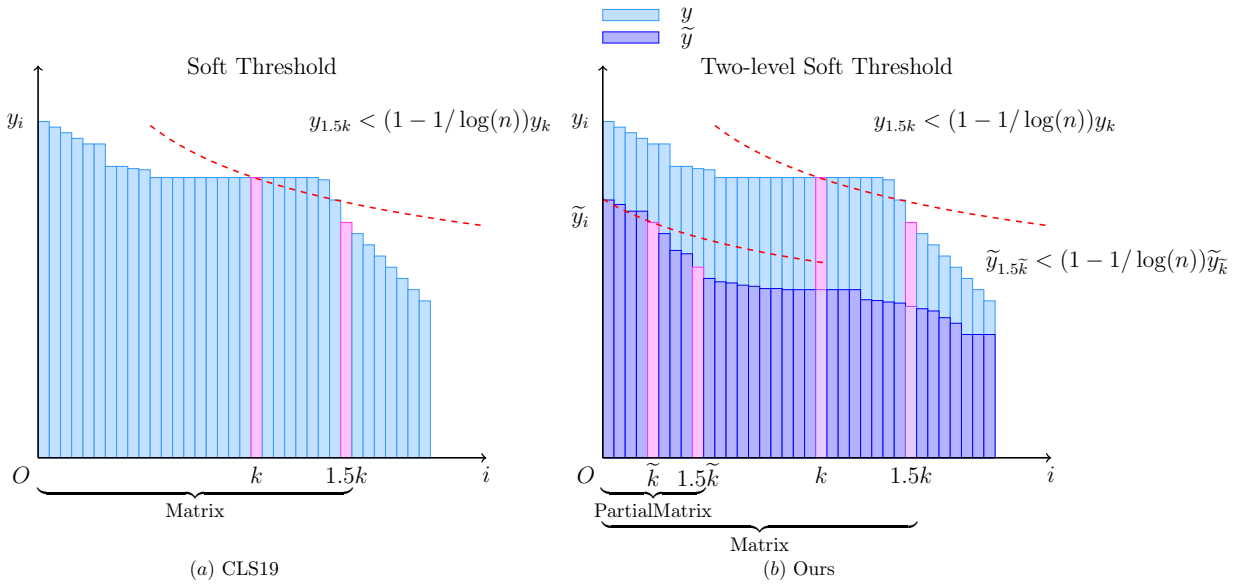
4.2.3 Amortized analysis based on high-order martingales

As noted in Section 2, the main source of amortization in the update algorithm comes from the fact that the maintained variables in all levels are changing slowly. More formally, the j-th iteration of the CP algorithm calls our data structure with the following inputs: A vector $w^{(j+1)} \in \mathbb{R}^n$ and a vector $h^{(j+1)} \in \mathbb{R}^n$ satisfying the following relative error bounds (where the randomness is over the sketching matrix used to generate $w^{(j+1)}$ and $h^{(j+1)}$):

$$\|\mathbb{E}[w^{(j+1)}|w^{(j)}]/w^{(j)} - 1\|_2 \leq O(1), \quad \|\mathbb{E}[h^{(j+1)}|h^{(j)}]/h^{(j)} - 1\|_2 \leq O(1). \tag{9}$$

The "evolution" of $w^{(j)}$ and $h^{(j)}$ is essentially a martingale process with the above guarantee. Informally, this is true because we are using *independent* random sketches in each iteration. We analyze these random processes by defining four different potential functions to capture our four different update subroutines, as shown in Table 3. We next elaborate on the careful design of potentials and what they aim to measure.

1. MatrixUpdate. The potential function for this subroutine is defined in row 1 of Table 3. Instead of splitting the change in the potential function into "w move" and "v move" as [CLS19],



(a) CLS19 (b) Ours

Figure 6: Two-level vs. one-level soft thresholding: (a). [CLS19] has a single level update scheme, implemented via one level of soft thresholding. (b). Our Update algorithm has two cascading subroutines (MatrixUpdate, PartialMatrixUpdate), each requires its own potential function and soft threshold.

Level	**Name**	$\Phi \in \mathbb{R}$	$\phi \in \mathbb{R}^n$, $i > n^a$ (resp. $i > n^{\widetilde{a}}$)	**Amortized**
1	Matrix	$\sum_{i=1}^n \phi_i \cdot (\lvert w_i/v_i - 1\rvert + \lvert w_i/\widetilde{v}_i - 1\rvert)$	$\phi_i = i^{\frac{\omega-2}{1-a}-1} \cdot n^{-\frac{a(\omega-2)}{1-a}}$	$n^{2-a/2} + n^{\omega-1/2}$
2	PartialMatrix	$\sum_{i=1}^n \phi_i \cdot (\lvert w_i/\widetilde{v}_i - 1\rvert)$	$\phi_i = i^{\frac{a(\omega-2)}{a-\widetilde{a}}-1} \cdot n^{-\frac{a\widetilde{a}(\omega-2)}{a-\widetilde{a}}}$	$n^{1+a-\widetilde{a}/2}$
1	Vector	$\sum_{i=1}^n \phi_i \cdot (\lvert h_i/g_i - 1\rvert + \lvert h_i/\widetilde{g}_i - 1\rvert)$	$\phi_i = 1$	$n^{1.5}$
2	PartialVector	$\sum_{i=1}^n \phi_i \cdot (\lvert h_i/\widetilde{g}_i - 1\rvert)$	$\phi_i = i^{-1}$	$n^{1.5} + n^{2a-\widetilde{a}/2}$

Table 3: The four different potential functions Φ are shown in the third column. We always assume that the coordinates are sorted in decreasing order, e.g., for PartialMatrix, we assume that $|w_i/\widetilde{v}_i - 1| \geq |w_{i+1}/\widetilde{v}_{i+1} - 1|$. The vector $\phi \in \mathbb{R}^n$ is non-increasing (in i): for level-1 subroutines, $\phi_i := n^{-a}$ when $i \leq n^a$, and for level-2 subroutines, $\phi_i := n^{-\widetilde{a}}$ for $i \leq n^{\widetilde{a}}$. The definition of ϕ_i for $i > n^a$ (level 1) or $n^{\widetilde{a}}$ (level 2) are shown in the fourth column. The vectors ϕ are designed to upper bound the worst-case running time of the update procedures. The last column shows the amortized cost of our four update subroutines.

we split the change into a "w move" part and a "*v and $\widetilde{v}$ move*" part:

$$\Phi_{j+1}^{\mathrm{mat}} - \Phi_j^{\mathrm{mat}} = \ (w \text{ move}) - (v \text{ and } \widetilde{v} \text{ move})$$

where the "w move" part measures how much the potential can increase due to the input changes from $w^{(j)}$ to $w^{(j+1)}$, and the "v and $\widetilde{v}$ move" part measures how much we can decrease the potential by updating $v^{(j)}, \widetilde{v}^{(j)}$ to $v^{(j+1)}, \widetilde{v}^{(j+1)}$. The formal details can be found in Section F.

Using Eq. (9) which upper bounds the expected relative error of $w^{(j+1)}$ with $w^{(j)}$, it is possible to upper bound the "w move" term by $O(\|\phi\|_2) = O(n^{\omega-5/2} + n^{-a/2})$.

When entering MatrixUpdate, for some coordinate $i \in [k]$ (recall that $k := \|v^{\mathrm{new}} - v\|_0$, see Table 2), by design $v_i^{(j+1)}$ and $\widetilde{v}_i^{(j+1)}$ are both reset to $w_i^{(j+1)}$, hence the potential $\phi_i(|w_i^{(j+1)}/v_i^{(j+1)} - 1| + |w_i^{(j+1)}/\widetilde{v}_i^{(j+1)} - 1|)$ decreases to 0. This fact can be used to show that the "v move" term in Φ decreases by at least

$$\Omega(k \cdot \phi_k) \geq n^{-2} \cdot \mathcal{T}_{\mathrm{mat}}(n, n, k).$$

We also prove that PartialMatrixUpdate can only further decrease the "v move" term. Since the "v and $\widetilde{v}$ move" term is upper bounded by the "w move" term, and the cost per call of MatrixUpdate is $\mathcal{T}_{\mathrm{mat}}(n, n, k)$, the amortized cost of MatrixUpdate per iteration is bounded by $O(n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)})$.

2. PartialMatrixUpdate. The potential function for this subroutine is defined in row 2 of Table 3. Once again, we split the change in the potential function, this time into a "w move" part and a "$\widetilde{v}$ move" part:

$$\Phi_{j+1} - \Phi_j = (w \text{ move}) - (\widetilde{v} \text{ move})$$

The "w move" term is upper bounded by $O(\|\phi\|_2) = O(n^{a\omega-5\widetilde{a}/2} + n^{-\widetilde{a}/2})$. To lower bound the "$\widetilde{v}$ move" term, we observe that when entering PartialMatrixUpdate, for any $i \in [\widetilde{k}]$ (recall that $\widetilde{k} := \|\widetilde{v}^{\mathrm{new}} - \widetilde{v}\|_0$, see Table 2), the term $|w_i^{\mathrm{new}}/\widetilde{v}_i^{\mathrm{new}} - 1|$ is decreased by at least a factor of $(1 - 1/\log n)$. This is where we use the guarantees (and smaller threshold parameter) of Adjust and SoftThreshold for v. Using this, we show that the "$\widetilde{v}$ move" term decreases by at least

$$\Omega(\widetilde{k} \cdot \phi_{\widetilde{k}}) \geq n^{-1-a} \cdot \mathcal{T}_{\mathrm{mat}}(n, n^a, \widetilde{k}).$$

Therefore, the amortized cost of PartialMatrixUpdate is $O(n^{1+(\omega+1)a-5\widetilde{a}/2} + n^{1+a-\widetilde{a}/2}) = O(n^{1+a-\widetilde{a}/2})$ since the cost per call of PartialMatrixUpdate is $\mathcal{T}_{\mathrm{mat}}(n, n^a, \widetilde{k})$.

3. VectorUpdate. The potential function for this subroutine is defined in row 3 of Table 3. Note that the dominating term in the worst-case cost (per call) of this subroutine is the pn term (recall $p := \|g^{\mathrm{new}} - g\|_0$, see Table 2). Again, after amortization in each iteration we have

$$\sqrt{n} = \|\phi\|_2 \geq \Omega(p \cdot \phi_p) \geq n^{-1}(pn).$$

Therefore, the amortized cost of VectorUpdate per iteration is $O(n^{1.5})$.

4. PartialVectorUpdate. The potential function for this subroutine is defined in row 4 of Table 3. Here, the dominating term in the worst-case cost (per call) is the n^{2a} term (Table 2). Since PartialVectorUpdate is invoked only when $\widetilde{p} \geq n^{\widetilde{a}}$ (recall that $\widetilde{p} := \|\widetilde{g}^{\mathrm{new}} - \widetilde{g}\|_0$, see Table 2), the amortized cost of the j-th iteration is $n^{2a} \cdot \mathbf{1}_{\widetilde{p}_j > n^{\widetilde{a}}}$. Once again, after amortization in each iteration we have

$$n^{-\widetilde{a}} = \|\phi\|_2 \geq \Omega(p \cdot \phi_p) \geq \mathbf{1}_{\widetilde{p}_j > n^{\widetilde{a}}}.$$

Thus the amortized cost of VectorUpdate per iteration is $n^{2a} \cdot O(\|\phi\|_2) = O(n^{2a-\widetilde{a}/2})$.

4.3 Putting it all together

Combining the query time $t_q = n^{a+(\omega-1)\widetilde{a}}$ (Eq. (7) in Section 4.1) and the update time $t_u = n^{\omega-1/2} + n^{2-a/2} + n^{1+a-\widetilde{a}/2}$ (see the last column of Table 3), and since there are $O(\sqrt{n})$ iterations in total, we have that the final running time of our LP algorithm is

$$\sqrt{n} \cdot (t_q + t_u) = n^{0.5+a+(\omega-1)\widetilde{a}} + n^{\omega} + n^{2.5-a/2} + n^{1.5+a-\widetilde{a}/2},$$

matching the statement of Theorem 4.1.

Also note that in the ideal case where $\omega = 2$ and $\alpha = 1$, we can choose $a = \frac{8}{9}$ and $\widetilde{a} = \frac{2}{3}$, and this leads to an $O(n^{2+1/18})$ algorithm.

References

[BCRL79] D. Bini, M. Capovani, F. Romani, and G. Lotti. $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication. 8(5):234–235, 1979.

[Ber24] Sergei Bernstein. On a modification of chebyshev's inequality and of the error formula of laplace. *Ann. Sci. Inst. Sav. Ukraine, Sect. Math*, 1(4):38–49, 1924.

[BLSS20] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *STOC*. https://arxiv.org/pdf/2002.02304.pdf, 2020.

[BNS19] Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. Dynamic matrix inverse: Improved algorithms and matching conditional lower bounds. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 456–480. IEEE, 2019.

[Bra20] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 259–278. SIAM, 2020.

[CGLZ20] Matthias Christandl, François Le Gall, Vladimir Lysikov, and Jeroen Zuiddam. Barriers for fast rectangular matrix multiplication. In *arXiv preprint*. https://arxiv.org/pdf/2003.03019.pdf, 2020.

[CKSU05] Henry Cohn, Robert Kleinberg, Balazs Szegedy, and Christopher Umans. Group-theoretic algorithms for matrix multiplication. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 379–388. IEEE, 2005.

[CLS19] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *STOC*. https://arxiv.org/pdf/1810.07896, 2019.

[Cop82] Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM Journal on Computing*, 11(3):467–471, 1982.

[Cop97] Don Coppersmith. Rectangular matrix multiplication revisited. *Journal of Complexity*, 13(1):42–49, 1997.

[CW82] Don Coppersmith and Shmuel Winograd. On the asymptotic complexity of matrix multiplication. *SIAM Journal on Computing*, 11(3):472–492, 1982.

[CW87] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing (STOC)*, pages 1–6. ACM, 1987.

[CW13] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference (STOC)*, pages 81–90. https://arxiv.org/pdf/1207.6365, 2013.

[Dan47] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1947.

[FS89] Michael Fredman and Michael Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 345–354, 1989.

[GLPS10] Anna C Gilbert, Yi Li, Ely Porat, and Martin J Strauss. Approximate sparse recovery: optimizing time and measurements. *SIAM Journal on Computing 2012 (A preliminary version of this paper appears in STOC 2010)*, 41(2):436–453, 2010.

[GU18] Francois Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pages 1029–1046. https://arxiv.org/pdf/1708.05622.pdf, 2018.

[HKNS15] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 21–30, 2015.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing (STOC)*, pages 302–311. ACM, 1984.

[Kha80] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[KM72] Victor Klee and George J Minty. How good is the simplex algorithm. *Inequalities*, 3(3):159–175, 1972.

[KN12] Daniel M. Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. In *SODA*, pages 1195–1206, 2012.

[LDFU13] Yichao Lu, Paramveer Dhillon, Dean P Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in neural information processing systems*, pages 369–377, 2013.

[LG14] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation (ISSAC)*, pages 296–303. ACM, https://arxiv.org/pdf/1401.7714.pdf, 2014.

[LNNT16] Kasper Green Larsen, Jelani Nelson, Huy L Nguyen, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–70. IEEE, https://arxiv.org/pdf/1604.01357, 2016.

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 424–433. https://arxiv.org/pdf/1312.6677.pdf, https://arxiv.org/pdf/1312.6713.pdf, 2014.

[LS15] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 230–249. https://arxiv.org/pdf/1503.01752.pdf, 2015.

[LSZ19] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*. https://arxiv.org/pdf/1905.04447, 2019.

[Mad13] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 253–262. IEEE, 2013.

[NN94] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994.

[Pan78] V Ya Pan. Strassen's algorithm is not optimal trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations. In *19th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 166–176. IEEE, 1978.

[PD06] Mihai Patrascu and Erik D Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM Journal on Computing*, 35(4):932–963, 2006.

[PSW17] Eric Price, Zhao Song, and David P. Woodruff. Fast regression with an ℓ_∞ guarantee. In *International Colloquium on Automata, Languages, and Programming (ICALP)*. https://arxiv.org/pdf/1705.10723.pdf, 2017.

[Ren88] James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, 1988.

[Rom82] Francesco Romani. Some properties of disjoint sums of tensors related to matrix multiplication. *SIAM Journal on Computing*, 11(2):263–267, 1982.

[San04] Piotr Sankowski. Dynamic transitive closure via dynamic matrix inverse. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 509–517. IEEE, 2004.

[Sar06] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[Sch81] Arnold Schönhage. Partial and total matrix multiplication. *SIAM Journal on Computing*, 10(3):434–455, 1981.

[SM10] Piotr Sankowski and Marcin Mucha. Fast dynamic transitive closure with lookahead. *Algorithmica*, 56(2):180, 2010.

[Son19] Zhao Song. *Matrix Theory : Optimization, Concentration and Algorithms*. PhD thesis, The University of Texas at Austin, 2019.

[ST85] Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *Journal of the ACM (JACM)*, 32(3):652–686, 1985.

[ST04] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 81–90, 2004.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[Str86] Volker Strassen. The asymptotic spectrum of tensors and the exponent of matrix multiplication. In *27th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 49–54. IEEE, 1986.

[Str87] Gilbert Strang. Karmarkar's algorithm and its place in applied mathematics. *The Mathematical Intelligencer*, 9(2):4–10, 1987.

[Str91] Volker Strassen. Degeneration and complexity of bilinear maps: some asymptotic spectra. *J. reine angew. Math*, 413:127–180, 1991.

[Vai87] Pravin M Vaidya. An algorithm for linear programming which requires $O(((m+n)n^2 + (m+n)^{1.5}n)L)$ arithmetic operations. In *28th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 1987.

[Vai89] Pravin M Vaidya. Speeding-up linear programming using fast matrix multiplication. In *30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 332–337. IEEE, 1989.

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 887–898. ACM, 2012.

[Woo49] Max A Woodbury. The stability of out-input matrices. *Chicago, IL*, 9, 1949.

[Woo50] Max A Woodbury. Inverting modified matrices. ., 1950.

[YTM94] Yinyu Ye, Michael J. Todd, and Shinji Mizuno. An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. *Math. Oper. Res.*, 19(1):53–67, 1994.

Contents

1 Introduction **1**

2 Bootstrapping low-rank updates via cascading lazy updates **3**

3 Background **7**
- 3.1 Recent developments in LP solvers . . . 7
- 3.2 Optimization: The stochastic central-path algorithm . . . 7
- 3.3 Data structures: Projection maintenance . . . 8

4 Detailed Technical Overview **9**
- 4.1 Our Query Algorithm . . . 10
  - 4.1.1 Technique for removing n^{1+a} . . . 12
  - 4.1.2 Technique for removing $n^{\omega a}$ . . . 13
  - 4.1.3 Technique for removing n^{2a} . . . 14
- 4.2 Our Update Algorithm . . . 16
  - 4.2.1 Cascading updates subroutines . . . 16
  - 4.2.2 Synchronizing two-level soft thresholding . . . 17
  - 4.2.3 Amortized analysis based on high-order martingales . . . 18
- 4.3 Putting it all together . . . 20

References **21**

A Preliminaries **27**

B Optimization **29**
- B.1 Definitions . . . 29
- B.2 Facts . . . 32
- B.3 Bounding δ_s, δ_x, δ_t, δ_Φ and δ_μ . . . 34
- B.4 Bounding $\overline{\mu}^{\mathrm{new}} - \overline{\mu}$ . . . 38
- B.5 Potential martingale . . . 41
- B.6 Bounding the movement of $\overline{w}$ . . . 44
- B.7 Bounding the movement of $\overline{\mu}$ . . . 45
- B.8 One step of central path . . . 47

C Data structure : preliminary **48**
- C.1 Preliminary and Definitions . . . 48
- C.2 Facts . . . 50
- C.3 Main result . . . 55

D Data structure : correctness **56**
- D.1 Correctness of Query . . . 63
- D.2 Correctness of UpdateV and UpdateG . . . 68
- D.3 Correctness of MatrixUpdate . . . 69
- D.4 Correctness of PartialMatrixUpdate . . . 71
- D.5 Correctness of VectorUpdate . . . 73
- D.6 Correctness of PartialVectorUpdate . . . 74
- D.7 Correctness of Initialize . . . 76

E Data structure : time per call **76**
E.1 Sparsity guarantees . . . 76
E.2 Running time of Query . . . 78
E.3 Running time of MatrixUpdate . . . 83
E.4 Running time of PartialMatrixUpdate . . . 85
E.5 Running time of VectorUpdate . . . 89
E.6 Running time of PartialVectorUpdate . . . 90
E.7 Running time of Initialize . . . 92

F Data structure : amortized time **92**
F.1 Definitions and Preliminaries . . . 92
F.2 Facts based on Adjust and two level of SoftThreshold . . . 93
F.3 Amortized analysis for MatrixUpdate . . . 99
F.3.1 Definitions . . . 99
F.3.2 Main result . . . 100
F.3.3 w move . . . 101
F.3.4 $v, \widetilde{v}$ move . . . 104
F.3.5 ℓ_2-norm of g . . . 105
F.4 Amortized analysis for PartialMatrixUpdate . . . 106
F.4.1 Definitions . . . 106
F.4.2 Main result . . . 106
F.4.3 w move . . . 108
F.4.4 $\widetilde{v}$ move . . . 108
F.4.5 ℓ_2-norm of g . . . 110
F.5 Amortized analysis for VectorUpdate . . . 110
F.6 Amortized analysis for PartialVectorUpdate . . . 111
F.7 Potential function ψ . . . 112

G Combining data structure with optimization **113**

H Multi-level with more details **117**
H.1 LU-decomposition of Woodbury identity when $K = 3$ . . . 117
H.2 Online low-rank inverse and the oMV conjecture. . . . 117
H.3 Optimizing the parameters of Eq. (3) . . . 119

I A feasible algorithm **119**
I.1 Analysis . . . 122
I.2 Correctness of feasible algorithm . . . 126
I.3 Bounding x and $\overline{x}$ . . . 130
I.4 Running time of feasible data structure . . . 132

J History of Matrix Multiplication and LP **135**

Appendix

A Preliminaries

Throughout this paper when considering the linear program $\min_{Ax=b,x\geq 0} c^\top x$, we always assume that the input matrix $A \in \mathbb{R}^{d\times n}$ is full-rank, and $d = \Omega(n)$, $d \leq n$. For convenience we also assume that $n \geq 10$. In our algorithm we use the standard transformation of LP by [YTM94] such that it is easy to obtain the initial x and s for the transformed LP. We refer the readers to [YTM94] and [CLS19] for more details.

Standard notations. For a positive integer n, we denote $[n] = \{1, 2, \cdots, n\}$.

For a positive integer n, we use I_n to denote the identity matrix of size $n \times n$. We use standard definitions of hyperbolic functions $\sinh(x) = \frac{e^x - e^{-x}}{2}$, $\cosh(x) = \frac{e^x + e^{-x}}{2}$.

For a vector $v \in \mathbb{R}^n$, we use the standard definition of ℓ_p norms: $\forall p \geq 1$, $\|v\|_p = (\sum_{i=1}^n |v_i|^p)^{1/p}$. Specially, $\|v\|_\infty = \max_{i\in[n]} |v_i|$.

A vector $v \in \mathbb{R}^n$ is called $k-$sparse if at most k entries in v is non-zero.

For any random variable x, we use $\mathbb{E}[x]$ to denote its expectation, and we use $\mathbf{Var}[x]$ to denote its variance. We define $\mathbf{Sup}[x]$ to be the deterministic maximum of x.

Approximations. For any function f, we define $\widetilde{O}(f) = f \cdot \operatorname{poly}\log(f)$, and $O^*(f) = f \cdot f^{o(1)}$.

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

Coordinate-wise operations. For a vector $x \in \mathbb{R}^n$ and $s \in \mathbb{R}^n$, we denote $xs \in \mathbb{R}^n$ as a length-n vector whose i-th coordinate is $(xs)_i = x_i s_i$, $\forall i \in [n]$. Similarly, we also define other scalar operations on vectors as coordinate-wise operations.

For a scalar function $f : \mathbb{R} \to \mathbb{R}$ and a vector $x \in \mathbb{R}^n$, we define $f(x) = [f(x_1), f(x_2), \ldots, f(x_n)]^\top$.

Upper case as diagonal matrix. Given vectors $x, s \in \mathbb{R}^n$, we use $X \in \mathbb{R}^{n\times n}$ and $S \in \mathbb{R}^{n\times n}$ to denote the diagonal matrix of those two vectors. We use X/S to denote the diagonal matrix there the i-th entry on the diagonal is $(X/S)_{i,i} = x_i/s_i$, $\forall i \in [n]$. Similarly, we extend other scalar operations to diagonal matrix.

Matrix and vectors with subscripts. For any matrix $M \in \mathbb{R}^{m\times n}$ where $m > 1$ and $n > 1$, and any subset $S \subseteq [n]$, we define $M_S \in \mathbb{R}^{m\times|S|}$ to be the submatrix of M that only has columns in S.

For any subsets $S_1 \subseteq [m], S_2 \subseteq [n]$, we also define $M_{S_1,S_2} \in \mathbb{R}^{|S_1|\times|S_2|}$ to be the submatrix of M that only has rows in S_1 and columns in S_2.

For any vector $v \in \mathbb{R}^{n\times 1}$ where $n > 1$, and any subset $S \subseteq [n]$, we define $v_S \in \mathbb{R}^{|S|\times 1}$ to be the subvector of M that only has the entries in S.

Fast Matrix Multiplication. We use ω to denote the exponent of matrix multiplication, which is defined as the infimum number such that the multiplication of two $n \times n$ matrices can be done in $O(n^\omega)$ time. We use α to denote the the dual exponent of matrix multiplication, which is defined as the supremum over all $a \geq 0$ such that the multiplication of a $n \times n^a$ matrix with another $n^a \times n$ matrix can be done in $O(n^{2+o(1)})$ time.

We denote $\mathcal{T}_{\mathrm{mat}}(n,k,r)$ to be the time needed to multiply a $n \times k$ matrix with a $k \times r$ matrix. $\mathcal{T}_{\mathrm{mat}}(n,k,r)$ has the following property.

Lemma A.1 ([Str91, GU18, CGLZ20]).

$$\mathcal{T}_{\mathrm{mat}}(n,k,r) = O(\mathcal{T}_{\mathrm{mat}}(n,r,k)) = O(\mathcal{T}_{\mathrm{mat}}(k,n,r)).$$

Woodbury identity. We use the Woodbury matrix identity to calculate the inverse of a matrix M under low-rank updates.

Fact A.2 (Woodbury matrix identity, [Woo49, Woo50]). *For matrices $M \in \mathbb{R}^{n\times n}$, $U \in \mathbb{R}^{n\times k}$, $C \in \mathbb{R}^{k\times k}$, $V \in \mathbb{R}^{k\times n}$,*

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

Probability tools. We use the following well-known probability tools.

Lemma A.3 (Bernstein Inequality, [Ber24]). *Let $X_1, \cdots, X_n$ be independent zero-mean random variables. Suppose that $|X_i| \leq M$ almost surely, for all i. Then, for all $t > 0$,*

$$\Pr\left[\sum_{i=1}^n X_i > t\right] \leq \exp\left(-\frac{t^2/2}{\sum_{j=1}^n \mathbb{E}[X_j^2] + Mt/3}\right).$$

Central Path Method. The stochastic central path method of [CLS19] consider the following LP with $A \in \mathbb{R}^{d\times n}$: $\min_{Ax=b, x\geq 0} c^\top x$, and its dual: $\max_{A^\top y \leq c} b^\top y$. By defining $s \in \mathbb{R}^n$ as the slack variables, the optimal solution of the above LP must satisfy the following constraints:

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

where $\sum_{i=1}^n x_i s_i$ is the duality gap, and the other three equations are the feasibility constraints. In the central path method, the duality gap is parameterized by t that decreases by a factor of $1 - O(\frac{1}{\sqrt{n}})$ in each iteration. Let $\mu := xs$. [CLS19] designed a potential function $\Psi(\mu/t - 1)$ such that as long as $\Psi(\mu/t-1) \leq \mathrm{poly}(n)$, $\mu \approx t$ is satisfied. The change δ_μ of μ has two parts: a term $-\frac{\mu}{\sqrt{n}}$ to decrease μ, and a term $-\nabla\Psi(\mu/t-1)$ to bound the potential function.

Given δ_μ, the changes δ_x and δ_s should satisfy the following constraints (the second-order term $\delta_x \cdot \delta_s$ is ignored):

$$\begin{aligned} X\delta_s + S\delta_x &= \delta_\mu, \\ A\delta_x &= 0, \\ A^\top \delta_y + \delta_s &= 0. \end{aligned} \tag{10}$$

The unique solution of the above linear equations is

$$\begin{aligned} \delta_x &= \frac{X}{\sqrt{XS}}(I-P)\frac{1}{\sqrt{XS}}\delta_\mu, \\ \delta_s &= \frac{S}{\sqrt{XS}}P\frac{1}{\sqrt{XS}}\delta_\mu, \end{aligned}$$

where $P = \sqrt{\frac{X}{S}}A^\top(A\frac{X}{S}A^\top)^{-1}A\sqrt{\frac{X}{S}}$ is a projection matrix. Thus the stochastic central path method is summarized as the Algorithm 1 presented in Section 3.2.

Lemma	Section	Comment
Lemma B.16	Section B.3	Bounding $\delta_x, \delta_s, \delta_\mu, \delta_t, \delta_\Phi$
Lemma B.21	Section B.4	Bounding $\mu^{\mathrm{new}} - \mu$
Lemma B.27	Section B.5	Bounding the expectation of potential function
Lemma B.28	Section B.6	Bounding the movement of w
Lemma B.29	Section B.7	Bounding the movement of μ

Table 4: Summary of this section

B Optimization

In this section we provide the error analysis of central path method. The meaning of x, s, and the deviation of δ_x, δ_s are standard (see the Central Path Method paragraph of Section A).

The proof of this whole paper is induction based. The induction hypothesis ensures that at the beginning of each iteration Assumption B.5 is satisfied. In this section we use this induction hypothesis to prove the guarantees of the central path method. Later in Section G we combine the central path guarantees with the data structure guarantees to prove that Assumption B.5 is still satisfied at the end of each iteration. More specifically in this section we prove the following:

1. Two guarantees Lemma B.28 and B.29 that are needed by the data structure given in Section D. Then the properties of the data structure prove that Part 1 of Assumption B.5 is still satisfied.

2. An upper bound on the potential function (Lemma B.27) which proves that Part 2 of Assumption B.5 is still satisfied.

B.1 Definitions

Some of the definitions are used as variables in the algorithm, some of the definitions are used only for analysis, and some of the definitions are used for both.

Assumption B.1. *We use the following two error parameters ϵ and ϵ_{mp}:*

$$\epsilon \in (0, 10^{-4}), \text{ and } \epsilon_{\mathrm{mp}} \in (0, 10^{-4}).$$

We use the same potential function as [CLS19, LSZ19, Bra20],

Definition B.2 (Potential function)**.** *For a parameter $\lambda > \log n$, we define function $\Phi_\lambda : \mathbb{R}^n \to \mathbb{R}$:*

$$\Phi_\lambda(r) := \sum_{i=1}^{n} \cosh(\lambda r_i).$$

Definition B.3 (Overline version of parameters)**.** *At the beginning of each iteration, we have $t \in \mathbb{R}$, and $\overline{x}, \overline{s} \in \mathbb{R}^n$ from last iteration. $t^{\mathrm{new}} \in \mathbb{R}$ is the new value of t. We define $\overline{w} \in \mathbb{R}^n$ and $\overline{\mu} \in \mathbb{R}^n$:*

$$\overline{w} := \overline{x}/\overline{s}, \ \overline{\mu} := \overline{x} \cdot \overline{s}.$$

We define overline version of projection matrix $\overline{P} \in \mathbb{R}^{n \times n}$:

$$\overline{P} := \sqrt{\overline{W}} A^\top (A \overline{W} A^\top)^{-1} A \sqrt{\overline{W}}.$$

The change in $\overline{\mu}$ consists of two parts $\overline{\delta}_t$ and $\overline{\delta}_\Phi$:

$$\overline{\delta}_t := \Big(\frac{t^{\mathrm{new}}}{t} - 1\Big)\overline{\mu}, \quad \overline{\delta}_\Phi := -\frac{\epsilon}{2} t^{\mathrm{new}} \cdot \frac{\nabla \Phi_\lambda(\overline{\mu}/t - 1)}{\|\nabla \Phi_\lambda(\overline{\mu}/t - 1)\|_2}, \quad \overline{\delta}_\mu := \overline{\delta}_t + \overline{\delta}_\Phi.$$

The changes to $\overline{x}$ and $\overline{s}$ is computed from $\overline{\delta}_\mu$:

$$\overline{\delta}_x := \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}}(I - \overline{P})\frac{1}{\sqrt{\overline{X}\overline{S}}}\overline{\delta}_\mu, \quad \overline{\delta}_s := \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}}\overline{P}\frac{1}{\sqrt{\overline{X}\overline{S}}}\overline{\delta}_\mu.$$

The data structure will take $\overline{w} \in \mathbb{R}^n$ and $\overline{\mu} \in \mathbb{R}^n$ as inputs, and output some $\widetilde{w} \in \mathbb{R}^n$ and $\widetilde{\mu} \in \mathbb{R}^n$ such that $\widetilde{w} \approx_{\epsilon_{\mathrm{mp}}} \overline{w}$, $\widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$.

Definition B.4 (Tilde version of parameters)**.** *For a given $\widetilde{w} \in \mathbb{R}^n$ and $\widetilde{\mu} \in \mathbb{R}^n$ that is returned by the data structure, we define $\widetilde{x} \in \mathbb{R}^n$ and $\widetilde{s} \in \mathbb{R}^n$:*

$$\widetilde{x} := \sqrt{\widetilde{w}\widetilde{\mu}}, \quad \widetilde{s} := \sqrt{\widetilde{\mu}/\widetilde{w}}.$$

Note that $\widetilde{x}/\widetilde{s} = \widetilde{w}$, and $\widetilde{x}\widetilde{s} = \widetilde{\mu}$. We define tilde version of projection matrix $\widetilde{P} \in \mathbb{R}^{n \times n}$:

$$\widetilde{P} := \sqrt{\widetilde{W}} A^\top (A \widetilde{W} A^\top)^{-1} A \sqrt{\widetilde{W}}.$$

Further, we define $\widetilde{\delta}_t$, $\widetilde{\delta}_\Phi$, $\widetilde{\delta}_\mu \in \mathbb{R}^n$:

$$\widetilde{\delta}_t := (\frac{t^{\mathrm{new}}}{t} - 1)\widetilde{\mu}, \quad \widetilde{\delta}_\Phi := -\frac{\epsilon}{2} t^{\mathrm{new}} \cdot \frac{\nabla \Phi_\lambda(\widetilde{\mu}/t - 1)}{\|\nabla \Phi_\lambda(\widetilde{\mu}/t - 1)\|_2}, \quad \widetilde{\delta}_\mu := \widetilde{\delta}_t + \widetilde{\delta}_\Phi.$$

And we define $\widetilde{\delta}_x$, $\widetilde{\delta}_s \in \mathbb{R}^n$:

$$\widetilde{\delta}_x := \frac{\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}}(I - \widetilde{P})\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu, \quad \widetilde{\delta}_s := \frac{\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu.$$

Given these definitions, we state the following important assumptions. We assume they are satisfied in the beginning of each iteration, and use them to prove the correctness of the algorithm in this iteration. Later we will verify that these assumptions are always satisfied by induction on iterations. Note that the second assumption is true if the potential function is bounded by $\mathrm{poly}(n)$.

Assumption B.5. *We make the following assumptions:*

1. *$\widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$, $\widetilde{w} \approx_{\epsilon_{\mathrm{mp}}} \overline{w}$, where $\widetilde{\mu}$ and $\widetilde{w}$ are returned by the data structure, and $\overline{\mu}$, $\overline{w}$ are the input to the data structure.*
2. *$\overline{\mu} \approx_{0.1} t$.*

We further make the following definitions that make use of sketching matrices:

Definition B.6 (Hat version of parameters)**.** *Let $b = o(n)$. For any $\widetilde{x} \in \mathbb{R}^n$, $\widetilde{s} \in \mathbb{R}^n$, $\widetilde{P} \in \mathbb{R}^{n \times n}$, $\widetilde{\delta}_\mu \in \mathbb{R}^n$, and a sketching matrix $R \in \mathbb{R}^{b \times n}$, we define $\widehat{\delta}_x$, $\widehat{\delta}_s \in \mathbb{R}^n$:*

$$\widehat{\delta}_x := \frac{\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}}(I - R^\top R \widetilde{P})\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu, \quad \widehat{\delta}_s := \frac{\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}} R^\top R \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu.$$

Remark B.7. *In our case, $R \in \mathbb{R}^{b \times n}$ is a subsampled randomized Hadamard matrix. See more details in Definition B.10.*

Definition B.8 (New versions of definition)**.** *We use a superscript "new" to denote the corresponding variables at the beginning of the next iteration. Specifically, we define $t^{\mathrm{new}} \in \mathbb{R}$ as follows:*

$$t^{\mathrm{new}} := (1 - \frac{\epsilon}{3\sqrt{n}})t.$$

We define $\overline{\mu}^{\mathrm{new}}, \overline{w}^{\mathrm{new}} \in \mathbb{R}^n$ as follows:

$$\overline{\mu}^{\mathrm{new}} := (\overline{x} + \widehat{\delta}_x) \cdot (\overline{s} + \widehat{\delta}_s), \quad \overline{w}^{\mathrm{new}} := (\overline{x} + \widehat{\delta}_x)/(\overline{s} + \widehat{\delta}_s).$$

The following facts directly follow from the definitions and Assumption B.5.

Fact B.9. *We have the following properties:*

1. $\widetilde{X}\widehat{\delta}_s + \widetilde{S}\widehat{\delta}_x = \widetilde{\delta}_\mu = \widetilde{\delta}_t + \widetilde{\delta}_\Phi$,
2. $\widetilde{x} \approx_{2\epsilon_{\mathrm{mp}}} \overline{x}$, $\widetilde{s} \approx_{2\epsilon_{\mathrm{mp}}} \overline{s}$,
3. $\|\widetilde{\delta}_t\|_2 \leq 0.5\epsilon t$, $\|\widetilde{\delta}_\Phi\|_2 \leq 0.5\epsilon t$, $\|\widetilde{\delta}_\mu\|_2 \leq \epsilon t$.

Proof. **Part 1.** From the definition of $\widehat{\delta}_x$ and $\widehat{\delta}_s$ (Definition B.6) we have that

$$\begin{aligned}\widetilde{X}\widehat{\delta}_s + \widetilde{S}\widehat{\delta}_x &= \frac{\widetilde{X}\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}} R^\top R \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \widetilde{\delta}_\mu + \frac{\widetilde{S}\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}} (I - R^\top R \widetilde{P}) \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \widetilde{\delta}_\mu \\ &= \frac{\widetilde{X}\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}} I \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \widetilde{\delta}_\mu = \widetilde{\delta}_\mu = \widetilde{\delta}_t + \widetilde{\delta}_\Phi.\end{aligned}$$

Part 2. By part 1 and part 2 of Assumption B.5, we have that $\widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$ and $\widetilde{w} \approx_{\epsilon_{\mathrm{mp}}} \overline{w}$, so we have

$$\widetilde{x} = \sqrt{\widetilde{w} \cdot \widetilde{\mu}} \approx_{2\epsilon_{\mathrm{mp}}} \sqrt{\overline{w} \cdot \overline{\mu}} = \sqrt{(\overline{x}/\overline{s}) \cdot (\overline{x} \cdot \overline{s})} = \overline{x},$$

where the first step follows from Definition B.4, the second step follows from the fact that if $a \approx_{\epsilon_{\mathrm{mp}}} a'$ and $b \approx_{\epsilon_{\mathrm{mp}}} b'$, then $ab \approx_{2\epsilon_{\mathrm{mp}}} a'b'$, and the third step follows from Definition B.3.

Using a similar argument, we also have that $\widetilde{s} \approx_{2\epsilon_{\mathrm{mp}}} \overline{s}$.

Part 3. We upper bound $\|\widetilde{\delta}_t\|_2$ as follows:

$$\|\widetilde{\delta}_t\|_2 = \Big\|\Big(\frac{t^{\mathrm{new}}}{t} - 1\Big)\widetilde{\mu}\Big\|_2 = \Big\|\frac{\epsilon}{3\sqrt{n}}\widetilde{\mu}\Big\|_2 \leq (1 + \epsilon_{\mathrm{mp}})\Big\|\frac{\epsilon}{3\sqrt{n}}\overline{\mu}\Big\|_2 \leq 1.1(1 + \epsilon_{\mathrm{mp}})\Big\|\frac{\epsilon t}{3\sqrt{n}}\mathbf{1}\Big\|_2 \leq 0.5\epsilon t.$$

where the first step follows from the definition of $\widetilde{\delta}_t$ (Definition B.4), the second step follows from the definition of t^{new} (Definition B.8), the third step follows from $\widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$ (Part 1 of Assumption B.5), the forth step follows from $\overline{\mu} \approx_{0.1} t$ (Part 2 of Assumption B.5), and the last step follows from $\epsilon_{\mathrm{mp}} \leq 10^{-4}$ (Assumption B.1).

Then we upper bound $\|\widetilde{\delta}_\Phi\|_2$ as follows:

$$\|\widetilde{\delta}_\Phi\|_2 = \|\frac{\epsilon}{2}t^{\mathrm{new}} \cdot \frac{\nabla\Phi_\lambda(\widetilde{\mu}/t - 1)}{\|\nabla\Phi_\lambda(\widetilde{\mu}/t - 1)\|_2}\|_2 = \frac{\epsilon}{2}t^{\mathrm{new}} = \frac{\epsilon}{2}(1 - \frac{\epsilon}{3\sqrt{n}})t \leq 0.5\epsilon t,$$

where the first step follows from the definition of $\widetilde{\delta}_\Phi$ (Definition B.4), and the third step follows from the definition of t^{new} (Definition B.8).

Finally, we can use triangle inequality to upper bound

$$\|\widetilde{\delta}_\mu\|_2 \leq \|\widetilde{\delta}_t\|_2 + \|\widetilde{\delta}_\Phi\|_2 \leq 0.5\epsilon t + 0.5\epsilon t \leq \epsilon t.$$

□

B.2 Facts

Random sketching matrices are usually used to give subspace embedding and approximate matrix product. Instead of using subspace embedding [Sar06] and approximate matrix product [KN12], LP solver requires a different version of embedding, it was from [PSW17] implicitly and defined in [LSZ19] explicitly.

Definition B.10 (Coordinate-wise embedding)**.** *Let Π denote a distribution on $b \times n$ matrices R. We say Π is an (α, β, δ)-coordinate-wise embedding if for any fixed vector $h \in \mathbb{R}^n$, the following properties hold:*

1. $\mathop{\mathbb{E}}_{R\sim\Pi}[R^\top Rh] = h$,
2. $\mathop{\mathbb{E}}_{R\sim\Pi}[(R^\top Rh)_i^2] \leq h_i^2 + \frac{\alpha}{b}\|h\|_2^2$,
3. $\Pr_{R\sim\Pi}\left[|(R^\top Rh)_i - h_i| > \|h\|_2 \frac{\beta}{\sqrt{b}}\right] \leq \delta$.

Lemma B.11 (Lemma A.1 in [CLS19])**.** *Let x and y be (possibly dependent) random variables such that $|x| \leq c_x$ and $|y| \leq c_y$ almost surely. Then, we have*

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

Fact B.12 (Gradient and Hessian of potential function)**.** *Let $\Phi_\lambda(r) = \sum_{i=1}^n \cosh(\lambda r_i)$ for some $\lambda > 0$. The gradient and the Hessian of $\Phi_\lambda(r)$ are*

$$\begin{aligned}\nabla\Phi_\lambda(r) =& \ \lambda \cdot \big(\sinh(\lambda r_1), \sinh(\lambda r_2), \cdots, \sinh(\lambda r_n)\big)^\top \in \mathbb{R}^n,\\ \nabla^2\Phi_\lambda(r) =& \ \mathrm{diag}\left(\lambda^2\cosh(\lambda r_1), \lambda^2\cosh(\lambda r_2), \cdots, \lambda^2\cosh(\lambda r_n)\right) \in \mathbb{R}^{n\times n}.\end{aligned}$$

Lemma B.13 (Basic properties of potential function, Lemma 4.12 in [CLS19])**.** *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 $\|v\|_\infty \leq 1/\lambda$, we have that

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

2. $\|\nabla\Phi_\lambda(r)\|_2 \geq \frac{\lambda}{\sqrt{n}}(\Phi_\lambda(r) - n)$.
3. $\left(\sum_{i=1}^n \lambda^2\cosh^2(\lambda r_i)\right)^{1/2} \leq \lambda\sqrt{n} + \|\nabla\Phi_\lambda(r)\|_2$.

Lemma B.14 (Appendix E in [LSZ19])**.** *Let $R \in \mathbb{R}^{b\times n}$ denote a subsample randomized Hadamard transform, then it gives $(\alpha = 1, \beta = O(\log(n/\delta)), \delta)$-Coordinate-wise Embedding (Definition B.10).*

We state another property of potential function

Lemma B.15 (Basic properties of potential function, general version of Lemma 5.14 of [Bra20])**.** *Let $\Phi_\lambda(r) = \sum_{i=1}^n \cosh(\lambda r_i)$ for some $\lambda > 0$. If $\|v\|_\infty \leq 1/(30\lambda)$, then we have*

$$\left\langle \nabla\Phi_\lambda(r), -\frac{\nabla\Phi_\lambda(r+v)}{\|\nabla\Phi_\lambda(r+v)\|_2}\right\rangle \leq -0.9\|\nabla\Phi_\lambda(r)\|_2 + 0.1\lambda\sqrt{n}.$$

The proof is very similar to that of [Bra20], we put it here for completeness.

Proof. We have

$$\begin{aligned}\langle \nabla \Phi_\lambda(r), -\nabla \Phi_\lambda(r+v)\rangle =& -\langle \nabla \Phi_\lambda(r+v), \nabla \Phi_\lambda(r+v)\rangle + \langle \nabla \Phi_\lambda(r+v) - \nabla \Phi_\lambda(r), \nabla \Phi_\lambda(r+v)\rangle \\ \leq& -\|\nabla \Phi_\lambda(r+v)\|_2^2 + \|\nabla \Phi_\lambda(r) - \nabla \Phi_\lambda(r+v)\|_2 \cdot \|\nabla \Phi_\lambda(r+v)\|_2\end{aligned}$$

where the last step follows from $\langle a, b\rangle \leq \|a\|_2 \cdot \|b\|_2$. Then we have that

$$\begin{aligned}&\Big\langle \nabla \Phi_\lambda(r), -\frac{\nabla \Phi_\lambda(r+v)}{\|\nabla \Phi_\lambda(r+v)\|_2}\Big\rangle \\ \leq& -\|\nabla \Phi_\lambda(r+v)\|_2 + \|\nabla \Phi_\lambda(r) - \nabla \Phi_\lambda(r+v)\|_2 \\ \leq& -\Big(\|\nabla \Phi_\lambda(r)\|_2 - \|\nabla \Phi_\lambda(r) - \nabla \Phi_\lambda(r+v)\|_2\Big) + \|\nabla \Phi_\lambda(r) - \nabla \Phi_\lambda(r+v)\|_2 \\ \leq& -\|\nabla \Phi_\lambda(r)\|_2 + 2\|\nabla \Phi_\lambda(r) - \nabla \Phi_\lambda(r+v)\|_2, \end{aligned} \tag{11}$$

where the second step follows from the fact that $\|a\|_2 \geq \|b\|_2 - \|b-a\|_2$.

Now we need to upper bound the norm $\|\nabla \Phi_\lambda(r) - \nabla \Phi_\lambda(r+v)\|_2$. Note that $\nabla \Phi_\lambda(x)_i = \lambda \sinh(\lambda x_i)$ and $\sinh(x) = (e^x - e^{-x})/2$. We have the following property for $|\sinh(x+y) - \sinh(x)|$:

$$\begin{aligned}|\sinh(x+y) - \sinh(x)| &= |e^x \cdot e^y - e^{-x}\cdot e^{-y} - (e^x - e^{-x})|/2 \\ &= |e^x \cdot (e^y - 1) + e^{-x} \cdot (1 - e^{-y})|/2 \\ &\leq (e^x \cdot |e^y - 1| + e^{-x} \cdot |1 - e^{-y}|)/2 \\ &\leq (e^x + e^{-x})/2 \cdot \max\{|e^y - 1|, |1 - e^{-y}|\} \\ &\leq (e^x + e^{-x})/2 \cdot (e^{|y|} - 1) = \cosh(x)(e^{|y|} - 1),\end{aligned} \tag{12}$$

where the first and the last step follows from the definitions of sinh and cosh, the third step follows from the triangle inequality of absolute values, and the fifth step follows from $e^y + e^{-y} \geq 2$.

Thus we can upper bound the difference as follows

$$\begin{aligned}\|\nabla \Phi_\lambda(r) - \nabla \Phi_\lambda(r+v)\|_2 &= \lambda \|\sinh(\lambda(r+v)) - \sinh(\lambda r)\|_2 \\ &\leq \lambda \|\cosh(\lambda r)(e^{|\lambda v|} - 1)\|_2 \\ &\leq \lambda \|\cosh(\lambda r)\|_2 (e^{\lambda \|v\|_\infty} - 1) \\ &\leq (\lambda\sqrt{n} + \|\nabla \Phi_\lambda(r)\|_2)(e^{\lambda\|v\|_\infty} - 1),\end{aligned} \tag{13}$$

where the second step follows from Eq. (12), the third step follows from the fact that $\|a \cdot b\|_2 \leq \|a\|_2 \|b\|_\infty$, and the last step follows from Part 3 of Lemma B.13.

We then have

$$(e^{\lambda\|v\|_\infty} - 1) < e^{1/30} - 1 \leq 0.05, \tag{14}$$

where the first step follows from $\|v\|_\infty \leq 1/(30\lambda)$.

Finally, this allows us to obtain

$$\begin{aligned}\Big\langle \nabla \Phi_\lambda(r), -\frac{\nabla \Phi_\lambda(r+v)}{\|\nabla \Phi_\lambda(r+v)\|_2}\Big\rangle \leq& -\|\nabla \Phi_\lambda(r)\|_2 + 2\|\nabla \Phi_\lambda(r) - \nabla \Phi_\lambda(r+v)\|_2 \\ <& -\|\nabla \Phi_\lambda(r)\|_2 + 0.1(\lambda\sqrt{n} + \|\nabla \Phi_\lambda(r)\|_2) \\ \leq& -0.9\|\nabla \Phi_\lambda(r)\|_2 + 0.1\lambda\sqrt{n}\end{aligned}$$

where the first step follows from Eq. (11), and the second step follows from Eq. (13) and Eq. (14). □

Quantity	Bound	Stated in Lem. B.16	Used by Lem. B.21
$\|\overline{s}^{-1}\widetilde{\delta}_s\|_2$, $\|\overline{x}^{-1}\widetilde{\delta}_x\|_2$	ϵ	Part 1	Part 1
$\|\mathbb{E}[\overline{s}^{-1}\widehat{\delta}_s]\|_2$, $\|\mathbb{E}[\overline{x}^{-1}\widehat{\delta}_x]\|_2$	ϵ	Part 1	Part 1
$\|\overline{\mu}^{-1}(\widetilde{\delta}_t - \overline{\delta}_t)\|_2$	$\epsilon_{\mathrm{mp}} \cdot \epsilon$	Part 1	Part 1
$\|\overline{\mu}^{-1}(\overline{\delta}_t + \widetilde{\delta}_\Phi)\|_2$	ϵ	Part 1	Part 4
$\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}]$, $\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}]$	ϵ^2/b	Part 2	Part 1, 2
$\|\overline{x}^{-1}(\overline{x} - \widetilde{x})\|_\infty$, $\|\overline{s}^{-1}(\overline{s} - \widetilde{s})\|_\infty$	ϵ_{mp}	Part 3	/
$\|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty$, $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty$	ϵ	Part 3	/
$\|\overline{\mu}^{-1}\widetilde{\delta}_\mu\|_\infty$	ϵ	Part 3	Part 3
$\|\overline{x}^{-1}\widehat{\delta}_x\|_\infty$, $\|\overline{s}^{-1}\widehat{\delta}_s\|_\infty$	ϵ	Part 4	Part 2, 3

Table 5: Summary of Lemma B.16. We ignore the constants. Note that the Part 4 (last row of this table) holds with probability $1 - 1/\operatorname{poly}(n)$ and requires $b \geq 1000 \log^2 n$.

B.3 Bounding δ_s, δ_x, δ_t, δ_Φ and δ_μ

The goal of this section is to prove Lemma B.16.

Lemma B.16 (A deep version of Lemma 4.3 in [CLS19])**.** *Under Assumption B.5, and given that $b \geq 1000 \log^2 n$, we have the following:*

1. $\|\overline{s}^{-1}\widetilde{\delta}_s\|_2 \leq 2\epsilon$, $\|\overline{x}^{-1}\widetilde{\delta}_x\|_2 \leq 2\epsilon$,
   $\|\mathbb{E}[\overline{s}^{-1}\widehat{\delta}_s]\|_2 \leq 2\epsilon$, $\|\mathbb{E}[\overline{x}^{-1}\widehat{\delta}_x]\|_2 \leq 2\epsilon$,
   $\|\overline{\mu}^{-1}(\widetilde{\delta}_t - \overline{\delta}_t)\|_2 \leq \epsilon_{\mathrm{mp}} \cdot \epsilon$,
   $\|\overline{\mu}^{-1}(\overline{\delta}_t + \widetilde{\delta}_\Phi)\|_2 \leq 5\epsilon$.
2. $\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}] \leq 2\epsilon^2/b$, $\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] \leq 2\epsilon^2/b$.
3. $\|\overline{x}^{-1}(\overline{x} - \widetilde{x})\|_\infty \leq 2\epsilon_{\mathrm{mp}}$, $\|\overline{s}^{-1}(\overline{s} - \widetilde{s})\|_\infty \leq 2\epsilon_{\mathrm{mp}}$,
   $\|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \leq 2\epsilon$, $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \leq 2\epsilon$,
   $\|\overline{\mu}^{-1}\widetilde{\delta}_\mu\|_\infty \leq 5\epsilon$.
4. $\|\overline{x}^{-1}\widehat{\delta}_x\|_\infty \leq 3\epsilon$, $\|\overline{s}^{-1}\widehat{\delta}_s\|_\infty \leq 3\epsilon$ *hold with probability* $1 - 1/n^4$.

Claim B.17 (Part 1 of Lemma B.16, bounding the ℓ_2 norm)**.**

(1) $\|\overline{s}^{-1}\widetilde{\delta}_s\|_2 \leq 2\epsilon$, $\|\overline{x}^{-1}\widetilde{\delta}_x\|_2 \leq 2\epsilon$,
(2) $\|\mathbb{E}[\overline{s}^{-1}\widehat{\delta}_s]\|_2 \leq 2\epsilon$, $\|\mathbb{E}[\overline{x}^{-1}\widehat{\delta}_x]\|_2 \leq 2\epsilon$,
(3) $\|\overline{\mu}^{-1}(\widetilde{\delta}_t - \overline{\delta}_t)\|_2 \leq \epsilon_{\mathrm{mp}} \cdot \epsilon$,
(4) $\|\overline{\mu}^{-1}(\overline{\delta}_t + \widetilde{\delta}_\Phi)\|_2 \leq 5\epsilon$.

Proof. **Proof of (1).** We first upper bound the ℓ_2 norm of $\widetilde{P}\frac{\widetilde{\delta}_\mu}{\sqrt{\widetilde{X}\widetilde{S}}}$ in the following way:

$$\begin{aligned}\left\|\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\right\|_2 \leq \left\|\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\right\|_2 &\leq \mathbf{Sup}\left[\frac{1}{\sqrt{\widetilde{\mu}}}\right] \cdot \|\widetilde{\delta}_\mu\|_2 \leq \mathbf{Sup}\left[\frac{1}{\sqrt{(1-\epsilon_{\mathrm{mp}})\overline{\mu}}}\right] \cdot \epsilon t \\ &\leq \frac{1}{\sqrt{0.9(1-\epsilon_{\mathrm{mp}})t}} \cdot \epsilon t \;\leq\; 1.1\epsilon\sqrt{t}, \end{aligned} \tag{15}$$

where the first step holds since $\widetilde{P}$ is an orthogonal projection matrix, the second step is because $\widetilde{x}\widetilde{s} = \widetilde{\mu}$ (Definition B.4) and $\|a \cdot b\|_2 \leq \mathbf{Sup}[a]\|b\|_2$, the third step follows from $\widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$ (Part 1 of

Assumption B.5) and $\|\widetilde{\delta}_\mu\|_2 \le \epsilon t$ (Part 3 of Fact B.9), the fourth step follows from $\overline{\mu} \approx_{0.1} t$ (Part 2 of Assumption B.5), and the last step follows from $\epsilon_{\mathrm{mp}} \le 10^{-4}$ (Assumption B.1).

Then we can upper bound $\|\overline{s}^{-1}\widetilde{\delta}_s\|_2$ as follows:

$$\|\overline{s}^{-1}\widetilde{\delta}_s\|_2 = \left\|\frac{\overline{S}^{-1}\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\right\|_2 \le \mathbf{Sup}\left[\frac{\overline{S}^{-1}\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}}\right]\left\|\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\right\|_2 \le \frac{1+2\epsilon_{\mathrm{mp}}}{\sqrt{(1-\epsilon_{\mathrm{mp}})0.9t}}\cdot\left\|\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\right\|_2$$
$$\le \frac{1+2\epsilon_{\mathrm{mp}}}{\sqrt{(1-\epsilon_{\mathrm{mp}})0.9t}}\cdot 1.1\epsilon\sqrt{t} \le 2\epsilon,$$

where the first step follows by definition of $\widetilde{\delta}_s$ (Definition B.4), the second step follows from $\|a\cdot b\|_2 \le \mathbf{Sup}[a]\cdot\|b\|_2$, the third step follows from $\widetilde{s} \approx_{2\epsilon_{\mathrm{mp}}} \overline{s}$ (Part 2 of Fact B.9) and $\widetilde{x}\widetilde{s} = \widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu} \approx_{0.1} t$ (Definition B.4, Assumption B.5), the fourth step follows from Eq. (15), the last step follows from $\epsilon_{\mathrm{mp}} \le 10^{-4}$ (Assumption B.1).

The proof for $\|\overline{x}^{-1}\widetilde{\delta}_x\|_2 \le 2\epsilon$ is similar since $I - \widetilde{P}$ is also an orthogonal projection matrix.

Proof of (2). Note that from Lemma B.14 and definition of $\widehat{\delta}_s$ and $\widetilde{\delta}_s$ we have $\mathbb{E}[\widehat{\delta}_s] = \widetilde{\delta}_s$, therefore,

$$\|\mathbb{E}[\overline{s}^{-1}\widehat{\delta}_s]\|_2 = \|\overline{s}^{-1}\widetilde{\delta}_s\|_2 \le 2\epsilon.$$

Similarly, we can prove $\|\overline{x}^{-1}\widetilde{\delta}_x\|_2 \le 2\epsilon$.

Proof of (3).

$$\|\overline{\mu}^{-1}(\widetilde{\delta}_t - \overline{\delta}_t)\|_2 = \left\|\overline{\mu}^{-1}\left(\left(\frac{t^{\mathrm{new}}}{t}-1\right)\widetilde{\mu} - \left(\frac{t^{\mathrm{new}}}{t}-1\right)\overline{\mu}\right)\right\|_2 = \left\|\overline{\mu}^{-1}\frac{\epsilon}{3\sqrt{n}}(\widetilde{\mu}-\overline{\mu})\right\|_2 \le \epsilon_{\mathrm{mp}}\cdot\epsilon,$$

where the first step is by definition of $\widetilde{\delta}_t$ and $\overline{\delta}_t$ (Definition B.4 and B.3), the second step is by $\frac{t^{\mathrm{new}}}{t} - 1 = \frac{(1-\epsilon/3\sqrt{n})t}{t} - 1 = -\frac{\epsilon}{3\sqrt{n}}$, and the last step is by $\widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$ (Part 1 of Assumption B.5).

Proof of (4). We use triangle inequality to upper bound

$$\|\overline{\mu}^{-1}(\overline{\delta}_t + \widetilde{\delta}_\Phi)\|_2 = \|\overline{\mu}^{-1}((\overline{\delta}_t - \widetilde{\delta}_t) + (\widetilde{\delta}_t + \widetilde{\delta}_\Phi))\|_2 \le \|\overline{\mu}^{-1}(\overline{\delta}_t - \widetilde{\delta}_t)\|_2 + \|\overline{\mu}^{-1}(\widetilde{\delta}_t + \widetilde{\delta}_\Phi)\|_2.$$

The first term is upper bounded in Part (3): $\|\overline{\mu}^{-1}(\widetilde{\delta}_t - \overline{\delta}_t)\|_2 \le \epsilon_{\mathrm{mp}}\cdot\epsilon$.

For the second term, we have

$$\|\overline{\mu}^{-1}(\widetilde{\delta}_t + \widetilde{\delta}_\Phi)\|_2 = \|\overline{\mu}^{-1}\widetilde{\delta}_\mu\|_2 = \|\overline{\mu}^{-1}(\widetilde{x}\widetilde{\delta}_s + \widetilde{s}\widetilde{\delta}_x)\|_2 \le \|\overline{\mu}^{-1}\widetilde{x}\widetilde{\delta}_s\|_2 + \|\overline{\mu}^{-1}\widetilde{s}\widetilde{\delta}_x\|_2$$
$$\le (1+2\epsilon_{\mathrm{mp}})\|\overline{s}^{-1}\widetilde{\delta}_s\|_2 + (1+2\epsilon_{\mathrm{mp}})\|\overline{x}^{-1}\widetilde{\delta}_x\|_2 \le 4\epsilon(1+2\epsilon_{\mathrm{mp}}) \le 5\epsilon,$$

where the first step follows from $\widetilde{\delta}_\mu = \widetilde{\delta}_t + \widetilde{\delta}_\Phi$ (Definition B.4), the second step follows from

$$\widetilde{x}\widetilde{\delta}_s + \widetilde{s}\widetilde{\delta}_x = \frac{\widetilde{S}\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}}(I-\widetilde{P})\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu + \frac{\widetilde{X}\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu = \widetilde{\delta}_\mu,$$

the third step follows from triangle inequality, the forth step follows from $\widetilde{x} \approx_{2\epsilon_{\mathrm{mp}}} \overline{x}$, $\widetilde{s} \approx_{2\epsilon_{\mathrm{mp}}} \overline{s}$ (Part 2 of Fact B.9) and $\overline{x}\cdot\overline{s} = \overline{\mu}$ (Definition B.3), the fifth step follows from Part (1) that $\|\overline{x}^{-1}\widetilde{\delta}_x\|_2 \le 2\epsilon$ and $\|\overline{s}^{-1}\widetilde{\delta}_s\|_2 \le 2\epsilon$, and the sixth step follows from $\epsilon_{\mathrm{mp}} \le 10^{-4}$ (Assumption B.1). □

Claim B.18 (Part 2 of Lemma B.16, bounding the variance per coordinate)**.**

$$\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}] \le 2\epsilon^2/b, \ \mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] \le 2\epsilon^2/b.$$

Proof. For each $i \in [n]$, we can rewrite the expectation of $\overline{s}_i^{-1}\widehat{\delta}_{s,i}$ as follows:

$$\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] = \mathbb{E}\left[\frac{\widetilde{s}_i}{\overline{s}_i\sqrt{\widetilde{x}_i\widetilde{s}_i}}\left(R^\top R\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\right)_i\right] = \frac{\widetilde{s}_i}{\overline{s}_i\sqrt{\widetilde{x}_i\widetilde{s}_i}}\left(\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\right)_i = \overline{s}_i^{-1}\widetilde{\delta}_{s,i}, \tag{16}$$

where the first step follows from the definition of $\widehat{\delta}_s$ (Definition B.6), the second step follows from the property of matrix R (Part 1 of Definition B.10 and Lemma B.14), and the third step follows from the definition of $\widetilde{\delta}_s$ (Definition B.4).

We then upper bound the expectation of $(\overline{s}_i^{-1}\widehat{\delta}_{s,i})^2$ as follows:

$$\begin{aligned}\mathbb{E}[(\overline{s}_i^{-1}\widehat{\delta}_{s,i})^2] &= \frac{\widetilde{s}_i^2}{\overline{s}_i^2\cdot\widetilde{x}_i\widetilde{s}_i}\mathbb{E}\left[(R^\top R\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu)_i^2\right] \le \frac{\widetilde{s}_i^2}{\overline{s}_i^2\cdot\widetilde{x}_i\widetilde{s}_i}\left((\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu)_i^2 + \frac{1}{b}\left\|\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\right\|_2^2\right)\\ &= (\overline{s}_i^{-1}\widetilde{\delta}_{s,i})^2 + \frac{\widetilde{s}_i^2}{\overline{s}_i^2\cdot\widetilde{x}_i\widetilde{s}_i}\cdot\frac{1}{b}\left\|\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\right\|_2^2,\end{aligned} \tag{17}$$

where the first step follows from definition of $\widehat{\delta}_s$ (Definition B.6), the second step follows from Part 2 of Definition B.10, and the third step follows from the definition of $\widetilde{\delta}_s$ (Definition B.4).

Now we have

$$\begin{aligned}\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] =\ & \mathbb{E}[(\overline{s}_i^{-1}\widehat{\delta}_{s,i})^2] - (\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}])^2 = \frac{1}{b}\frac{\widetilde{s}_i^2}{\overline{s}_i^2\cdot\widetilde{x}_i\widetilde{s}_i}\|\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\|_2^2 \le \frac{1}{b}\frac{(1+2\epsilon_{\mathrm{mp}})^2}{\widetilde{\mu}_i}\|\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\|_2^2\\ \le\ & \frac{1}{b}\frac{(1+2\epsilon_{\mathrm{mp}})^2}{\widetilde{\mu}_i}(1.1\epsilon)^2 t \le \frac{1}{b}(1+2\epsilon_{\mathrm{mp}})^2(1.1\epsilon)^2(1.1+\epsilon_{\mathrm{mp}}) \le \frac{2\epsilon^2}{b},\end{aligned}$$

where the first step follows from definition of variance, the second step follows from Eq. (16) and Eq. (17), the third step follows from $\widetilde{s}_i \approx_{2\epsilon_{\mathrm{mp}}} \overline{s}_i$ (Part 2 of Fact B.9) and $\widetilde{\mu} = \widetilde{x}\cdot\widetilde{s}$ (Definition B.4), the forth step follows from $\|\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\|_2 \le 1.1\epsilon\sqrt{t}$ (Eq. (15)), and the sixth step follows from $\widetilde{\mu} \approx_{0.1+\epsilon_{\mathrm{mp}}} t$ (Part 1 and 2 of Assumption B.5), the last step follows from $\epsilon_{\mathrm{mp}} \le 10^{-4}$ (Assumption B.1).

The other part that $\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}] \le 2\epsilon^2/b$ follows from a similar argument. □

Claim B.19 (Part 3 of Lemma B.16, bounding the infinity norm)**.**

(1) $\|\overline{x}^{-1}(\overline{x}-\widetilde{x})\|_\infty \le 2\epsilon_{\mathrm{mp}},\ \|\overline{s}^{-1}(\overline{s}-\widetilde{s})\|_\infty \le 2\epsilon_{\mathrm{mp}}$,

(2) $\|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \le 2\epsilon,\ \|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \le 2\epsilon$,

(3) $\|\overline{\mu}^{-1}\widetilde{\delta}_\mu\|_\infty \le 5\epsilon$.

Proof. **Proof of (1).** From Part 2 of Fact B.9, we have that $\widetilde{x} \approx_{2\epsilon_{\mathrm{mp}}} \overline{x}$ and $\widetilde{s} \approx_{2\epsilon_{\mathrm{mp}}} \overline{s}$. Therefore $\|\overline{x}^{-1}(\overline{x}-\widetilde{x})\|_\infty \le 2\epsilon_{\mathrm{mp}}$, $\|\overline{s}^{-1}(\overline{s}-\widetilde{s})\|_\infty \le 2\epsilon_{\mathrm{mp}}$,

Proof of (2). From Part 1 of Claim B.17, we have $\|\overline{x}^{-1}\widetilde{\delta}_x\|_2 \le 2\epsilon$. Therefore, $\|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \le \|\overline{x}^{-1}\widetilde{\delta}_x\|_2 \le 2\epsilon$. Similarly, we have $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \le \|\overline{s}^{-1}\widetilde{\delta}_s\|_2 \le 2\epsilon$.

Proof of (3). Now, the last term follows by

$$\begin{aligned}|\overline{\mu}_i^{-1}\widetilde{\delta}_{\mu,i}| &= |\overline{x}_i^{-1}\overline{s}_i^{-1}(\widetilde{x}_i\widetilde{\delta}_{s,i} + \widetilde{s}_i\widetilde{\delta}_{x,i})| \le (1+2\epsilon_{\mathrm{mp}})|\overline{s}_i^{-1}\widetilde{\delta}_{s,i}| + (1+2\epsilon_{\mathrm{mp}})|\overline{x}_i^{-1}\widetilde{\delta}_{x,i}|\\ &\le (1+2\epsilon_{\mathrm{mp}})2\epsilon + (1+2\epsilon_{\mathrm{mp}})2\epsilon = 5\epsilon,\end{aligned}$$

where the first step is by $\widetilde{x}\widetilde{\delta}_s + \widetilde{s}\widetilde{\delta}_x = \widetilde{\delta}_\mu$ (Definition B.4), the second step is by $\overline{x} \approx_{2\epsilon_{\mathrm{mp}}} \widetilde{x}$ and $\overline{s} \approx_{2\epsilon_{\mathrm{mp}}} \widetilde{s}$ (Part 2 of Fact B.9), the third step follows from Part (2) that $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \le 2\epsilon$ and $\|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \le 2\epsilon$, and the last step follows from $\epsilon_{\mathrm{mp}} \le 10^{-4}$ (Assumption B.1). □

Claim B.20 (Part 4 of Lemma B.16, bounding the infinity norm with high probability)**.** *Given* $b \geq 1000 \log^2 n$,[8] *we have*

$$\|\overline{x}^{-1}\widehat{\delta}_x\|_\infty \leq 3\epsilon, \ \ \|\overline{s}^{-1}\widehat{\delta}_s\|_\infty \leq 3\epsilon,$$

holds with probability $1 - 1/n^4$.

Proof. By triangle inequality, we have

$$\|\overline{s}^{-1}\widehat{\delta}_s\|_\infty \leq \|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty + \|\overline{s}^{-1}(\widehat{\delta}_s - \widetilde{\delta}_s)\|_\infty.$$

The first term is upper bounded by $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \leq 2\epsilon$ (Part 2 of Claim B.19). The second part involves randomness, therefore we need to prove that it holds with high probability. Note that $\widehat{\delta}_s$ is the unbiased estimation of $\widetilde{\delta}_s$, i.e. $\mathbb{E}[\widehat{\delta}_s] = \widetilde{\delta}_s$. We have

$$\widehat{\delta}_s - \widetilde{\delta}_s = \frac{\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}}\left(R^\top R\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu - \widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu\right) = \frac{\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}}\left(R^\top Rh - h\right), \tag{18}$$

where the first steps is by definition of $\widehat{\delta}_s$(Definition B.6) and $\widetilde{\delta}_s$ (Definition B.4), and in the second step we define $h := \widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu$. And by Eq.(15), we have $\|h\|_2 \leq 1.1\epsilon\sqrt{t}$.

Definition B.10 and Lemma B.14 guarantee that for any vector $h \in \mathbb{R}^n$, a subsample randomized Hadamard transform matrix $R \in \mathbb{R}^{b\times n}$ satisfies

$$\Pr_R\left[|(R^\top Rh)_i - h_i| > \|h\|_2 \cdot \frac{\log(n/\delta)}{\sqrt{b}}\right] \leq \delta.$$

In every iteration we use a fresh subsample Hadamard matrix R which is independent of h, therefore we can apply this bound using the same h and failure probability $\delta = 1/n^4$, and we have that with probability at least $1 - 1/n^4$, $|(R^\top Rh)_i - h_i| \leq \frac{5.5\epsilon\sqrt{t}\log n}{\sqrt{b}}$. Therefore,

$$\begin{aligned}\left|\overline{s}_i^{-1}(\widehat{\delta}_s - \widetilde{\delta}_s)_i\right| &= \left|\frac{\overline{s}_i^{-1}\widetilde{s}_i}{\sqrt{\widetilde{x}_i\widetilde{s}_i}}\left(R^\top Rh_i - h_i\right)\right| \leq \left|\frac{1+2\epsilon_{\mathrm{mp}}}{\sqrt{0.9(1-\epsilon_{\mathrm{mp}})t}}\left(R^\top Rh_i - h_i\right)\right| \\ &\leq \left|\frac{1+2\epsilon_{\mathrm{mp}}}{\sqrt{0.9(1-\epsilon_{\mathrm{mp}})t}}\frac{5.5\epsilon\sqrt{t}\log n}{\sqrt{b}}\right| \ \leq \ \epsilon,\end{aligned} \tag{19}$$

where the first step is by Eq. (18), the second step is because $\widetilde{s} \approx_{2\epsilon_{\mathrm{mp}}} \overline{s}$ (Part 2 of Fact B.9) and $\widetilde{x}\widetilde{s} = \widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu} \approx_{0.1} t$ (Part 1 and 2 of Assumption B.5), and the third step is by the upper bound on $|(R^\top Rh)_i - h_i|$, the last step follows by $b \geq 1000\log^2 n$ and $\epsilon_{\mathrm{mp}} \leq 10^{-4}$ (Assumption B.1).

Finally, we have

$$\|\overline{s}^{-1}\widehat{\delta}_s\|_\infty \leq \ \|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty + \|\overline{s}^{-1}(\widehat{\delta}_s - \widetilde{\delta}_s)\|_\infty \leq \ 2\epsilon + \|\overline{s}^{-1}(\widehat{\delta}_s - \widetilde{\delta}_s)\|_\infty \leq \ 3\epsilon,$$

where the second step is by $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \leq 2\epsilon$ (Part 2 of Claim B.19), the third step is by Eq.(19).

Similarly, we can show $\|\overline{x}^{-1}\widehat{\delta}_x\|_\infty \leq 3\epsilon$ with probability $1 - 1/n^4$. □

Quantity	Bound	Part	Prob.	Use Lem. B.16
$\|\mathbb{E}[\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu} - \overline{\delta}_t - \widetilde{\delta}_\Phi)]\|_2$	$\epsilon_{\mathrm{mp}}\epsilon + \epsilon^2 + \epsilon^2\sqrt{n}/b$	Part 1	1	Part 1,2
$\mathbf{Var}[\overline{\mu}_i^{-1}\overline{\mu}_i^{\mathrm{new}}]$	$\epsilon_{\mathrm{mp}}^2\epsilon^2/b + \epsilon^4/b$	Part 2	$1 - 1/\operatorname{poly}(n)$	Part 2,4
$\|\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu})\|_\infty$	ϵ	Part 3	$1 - 1/\operatorname{poly}(n)$	Part 3,4
$\|\mathbb{E}[\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu})]\|_2$	$\epsilon + \epsilon^2\sqrt{n}/b$	Part 4	1	Part 1

Table 6: Summary of Lemma B.21. We ignore the constants.

B.4 Bounding $\overline{\mu}^{\mathrm{new}} - \overline{\mu}$

The goal of this section is to prove Lemma B.21.

Lemma B.21 (A deep version of Lemma 4.8 in [CLS19])**.** *Let $\overline{\mu}$ and $\overline{\mu}^{\mathrm{new}}$ be defined as that of Definition B.3 and Definition B.8: $\overline{\mu} = \overline{x} \cdot \overline{s}$, and $\overline{\mu}^{\mathrm{new}} = (\overline{x} + \widehat{\delta}_x)(\overline{s} + \widehat{\delta}_s)$. We have*
1. $\|\mathbb{E}[\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu} - \overline{\delta}_t - \widetilde{\delta}_\Phi)]\|_2 \leq 9\epsilon_{\mathrm{mp}}\epsilon + 4\epsilon^2 + 2\epsilon^2\sqrt{n}/b$,
2. $\mathbf{Var}[\overline{\mu}_i^{-1}\overline{\mu}_i^{\mathrm{new}}] \leq 16\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 320\epsilon^4/b$ *holds with probability at least* $1 - 1/\operatorname{poly}(n)$ *for all* $i \in [n]$,
3. $\|\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu})\|_\infty \leq 6\epsilon$,
4. $\|\mathbb{E}[\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu})]\|_2 \leq 6\epsilon + 2\epsilon^2\sqrt{n}/b$.

Claim B.22 (Part 1 of Lemma B.21)**.** $\|\mathbb{E}[\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu} - \overline{\delta}_t - \widetilde{\delta}_\Phi)]\|_2 \leq 9\epsilon_{\mathrm{mp}}\epsilon + 4\epsilon^2 + 2\epsilon^2\sqrt{n}/b$.

Proof. From the definition of $\overline{\mu}^{\mathrm{new}}$, we have

$$\begin{aligned}
\overline{\mu}^{\mathrm{new}} =& (\overline{x} + \widehat{\delta}_x)(\overline{s} + \widehat{\delta}_s) = \overline{\mu} + \overline{x}\widehat{\delta}_s + \overline{s}\widehat{\delta}_x + \widehat{\delta}_x\widehat{\delta}_s \\
=& \overline{\mu} + (\widetilde{x}\widehat{\delta}_s + \widetilde{s}\widehat{\delta}_x) + (\overline{x} - \widetilde{x})\widehat{\delta}_s + (\overline{s} - \widetilde{s})\widehat{\delta}_x + \widehat{\delta}_x\widehat{\delta}_s \\
=& \overline{\mu} + (\widetilde{\delta}_t + \widetilde{\delta}_\Phi) + (\overline{x} - \widetilde{x})\widehat{\delta}_s + (\overline{s} - \widetilde{s})\widehat{\delta}_x + \widehat{\delta}_x\widehat{\delta}_s \\
=& \overline{\mu} + (\overline{\delta}_t + \widetilde{\delta}_\Phi) + (\widetilde{\delta}_t - \overline{\delta}_t) + (\overline{x} - \widetilde{x})\widehat{\delta}_s + (\overline{s} - \widetilde{s})\widehat{\delta}_x + \widehat{\delta}_x\widehat{\delta}_s, \qquad (20)
\end{aligned}$$

where in the forth step we use the fact $\widetilde{x}\widehat{\delta}_s + \widetilde{s}\widehat{\delta}_x = \widetilde{\delta}_\mu = \widetilde{\delta}_t + \widetilde{\delta}_\Phi$ (Part 1 of Fact B.9). Subtracting $\overline{\mu} + (\overline{\delta}_t + \widetilde{\delta}_\Phi)$ on both sides and taking the expectation, we have

$$\mathbb{E}[\overline{\mu}^{\mathrm{new}} - \overline{\mu} - \overline{\delta}_t - \widetilde{\delta}_\Phi] = (\widetilde{\delta}_t - \overline{\delta}_t) + (\overline{x} - \widetilde{x})\,\mathbb{E}[\widehat{\delta}_s] + (\overline{s} - \widetilde{s})\,\mathbb{E}[\widehat{\delta}_x] + \mathbb{E}[\widehat{\delta}_x\widehat{\delta}_s].$$

Hence, we have that

$$\begin{aligned}
& \|\overline{\mu}^{-1}\,\mathbb{E}[\overline{\mu}^{\mathrm{new}} - \overline{\mu} - \overline{\delta}_t - \widetilde{\delta}_\Phi]\|_2 \\
\leq & \|\overline{\mu}^{-1}(\widetilde{\delta}_t - \overline{\delta}_t)\|_2 + \|\overline{\mu}^{-1}(\overline{x} - \widetilde{x})\overline{s} \cdot \overline{s}^{-1}\,\mathbb{E}[\widehat{\delta}_s]\|_2 + \|\overline{\mu}^{-1}(\overline{s} - \widetilde{s})\overline{x} \cdot \overline{x}^{-1}\,\mathbb{E}[\widehat{\delta}_x]\|_2 + \|\overline{\mu}^{-1}\,\mathbb{E}[\widehat{\delta}_x\widehat{\delta}_s]\|_2 \\
\leq & \epsilon_{\mathrm{mp}} \cdot \epsilon + \|\overline{\mu}^{-1}(\overline{x} - \widetilde{x})\overline{s} \cdot \overline{s}^{-1}\,\mathbb{E}[\widehat{\delta}_s]\|_2 + \|\overline{\mu}^{-1}(\overline{s} - \widetilde{s})\overline{x} \cdot \overline{x}^{-1}\,\mathbb{E}[\widehat{\delta}_x]\|_2 + \|\overline{\mu}^{-1}\,\mathbb{E}[\widehat{\delta}_x\widehat{\delta}_s]\|_2 \\
\leq & \epsilon_{\mathrm{mp}} \cdot \epsilon + \|\overline{\mu}^{-1}(\overline{x} - \widetilde{x})\overline{s}\|_\infty \cdot \|\overline{s}^{-1}\,\mathbb{E}[\widehat{\delta}_s]\|_2 + \|\overline{\mu}^{-1}(\overline{s} - \widetilde{s})\overline{x}\|_\infty \cdot \|\overline{x}^{-1}\,\mathbb{E}[\widehat{\delta}_x]\|_2 + \|\overline{\mu}^{-1}\,\mathbb{E}[\widehat{\delta}_x\widehat{\delta}_s]\|_2 \\
\leq & \epsilon_{\mathrm{mp}} \cdot \epsilon + 2\epsilon_{\mathrm{mp}} \cdot \|\overline{s}^{-1}\,\mathbb{E}[\widehat{\delta}_s]\|_2 + 2\epsilon_{\mathrm{mp}} \cdot \|\overline{x}^{-1}\,\mathbb{E}[\widehat{\delta}_x]\|_2 + \|\overline{\mu}^{-1}\,\mathbb{E}[\widehat{\delta}_x\widehat{\delta}_s]\|_2 \\
\leq & 9\epsilon_{\mathrm{mp}} \cdot \epsilon + \|\overline{\mu}^{-1}\,\mathbb{E}[\widehat{\delta}_x\widehat{\delta}_s]\|_2, \qquad (21)
\end{aligned}$$

where the first step follows by triangle inequality, the second step follows by Part 1 of Lemma B.16, the third step follows by $\|ab\|_2 \leq \|a\|_\infty \cdot \|b\|_2$, the forth step follows by $\|\overline{\mu}^{-1}(\overline{x} - \widetilde{x})\overline{s}\|_\infty \leq 2\epsilon_{\mathrm{mp}}$ and $\|\overline{\mu}^{-1}(\overline{s} - \widetilde{s})\overline{x}\|_\infty \leq 2\epsilon_{\mathrm{mp}}$ (since $\widetilde{x} \approx_{2\epsilon_{\mathrm{mp}}} \overline{x}$, $\widetilde{s} \approx_{2\epsilon_{\mathrm{mp}}} \overline{s}$ by Part 2 of Fact B.9, and $\overline{\mu} = \overline{x} \cdot \overline{s}$

[8] This assumption is added in Part 3 of Assumption B.26.

by Definition B.3), the last step follows by $\|\mathbb{E}[\overline{s}^{-1}\widehat{\delta}_s]\|_2 \le 2\epsilon$ and $\|\mathbb{E}[\overline{x}^{-1}\widehat{\delta}_x]\|_2 \le 2\epsilon$ (Part 1 of Lemma B.16).

To bound the last term of Eq. (21), using $\mathbb{E}[\widehat{\delta}_s] = \widetilde{\delta}_s$ and $\mathbb{E}[\widehat{\delta}_x] = \widetilde{\delta}_x$, we have that

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

Hence, we have

$$\begin{aligned}
\|\overline{\mu}^{-1}\,\mathbb{E}[\widehat{\delta}_x\widehat{\delta}_s]\|_2 &\le \|\overline{\mu}^{-1}\widetilde{\delta}_x\widetilde{\delta}_s\|_2 + \left(\sum_{i=1}^n \left(\mathbb{E}\left[\overline{x}_i^{-1}(\widehat{\delta}_{x,i} - \widetilde{\delta}_{x,i})\cdot \overline{s}_i^{-1}(\widehat{\delta}_{s,i} - \widetilde{\delta}_{s,i})\right]\right)^2\right)^{1/2} \\
&\le 4\epsilon^2 + \frac{1}{2}\left(\sum_{i=1}^n \left(\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}] + \mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}]\right)^2\right)^{1/2} \\
&\le 4\epsilon^2 + \frac{1}{2}\left(\sum_{i=1}^n 2(\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}])^2 + 2(\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}])^2\right)^{1/2} \\
&\le 4\epsilon^2 + 2\sqrt{n\cdot\epsilon^4/b^2} = 4\epsilon^2 + 2\epsilon^2\sqrt{n}/b, \qquad (22)
\end{aligned}$$

where the first step follows from triangle inequality and $\overline{\mu} = \overline{x}\overline{s}$, the second step follows by $\|\overline{\mu}^{-1}\widetilde{\delta}_x\widetilde{\delta}_s\|_2 \le \|\overline{x}^{-1}\widetilde{\delta}_x\|_2\cdot\|\overline{s}^{-1}\widetilde{\delta}_s\|_2 \le 4\epsilon^2$ (Part 1 of Lemma B.16) and $2ab \le a^2+b^2$, the third step follows by $(a+b)^2 \le 2a^2+2b^2$, the fourth step follows by $\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}] \le 2\epsilon^2/b$ and $\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] \le 2\epsilon^2/b$ (Part 2 of Lemma B.16).

Finally, we have that

$$\|\overline{\mu}^{-1}(\mathbb{E}[\overline{\mu}^{\mathrm{new}} - \overline{\mu} - \overline{\delta}_t - \widetilde{\delta}_\Phi])\|_2 \le\ 9\epsilon_{\mathrm{mp}}\epsilon + \|\overline{\mu}^{-1}\,\mathbb{E}[\widehat{\delta}_x\widehat{\delta}_s]\|_2 \le\ 9\epsilon_{\mathrm{mp}}\epsilon + 4\epsilon^2 + 2\epsilon^2\sqrt{n}/b.$$

where the first step follows from Eq. (21), and the last step follows from Eq. (22). □

Claim B.23 (Part 4 of Lemma B.21)**.** *We have*

$$\|\,\mathbb{E}[\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu})]\|_2 \le 6\epsilon + 2\epsilon^2\sqrt{n}/b.$$

Proof. From Part 1 of Lemma B.16, we know that $\|\overline{\mu}^{-1}(\overline{\delta}_t + \widetilde{\delta}_\Phi)\|_2 \le 5\epsilon$. Thus using triangle inequality and Part 1 of Lemma B.21, we know

$$\begin{aligned}
\|\overline{\mu}^{-1}(\mathbb{E}[\overline{\mu}^{\mathrm{new}} - \overline{\mu}])\|_2 &\le \|\overline{\mu}^{-1}(\mathbb{E}[\overline{\mu}^{\mathrm{new}} - \overline{\mu} - \overline{\delta}_t - \widetilde{\delta}_\Phi])\|_2 + \|\overline{\mu}^{-1}(\overline{\delta}_t + \widetilde{\delta}_\Phi)\|_2 \\
&\le 9\epsilon_{\mathrm{mp}}\epsilon + 4\epsilon^2 + 2\epsilon^2\sqrt{n}/b + 5\epsilon \le 6\epsilon + 2\epsilon^2\sqrt{n}/b,
\end{aligned}$$

where the last step follows by $\epsilon_{\mathrm{mp}} < 10^{-4}$ and $\epsilon < 10^{-4}$ (Assumption B.1). □

Claim B.24 (Part 2 of Lemma B.21)**.** $\mathbf{Var}[\overline{\mu}_i^{-1}\overline{\mu}_i^{\mathrm{new}}] \le 16\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 320\epsilon^4/b$ *holds with probability at least* $1 - 1/\operatorname{poly}(n)$ *for all* $i \in [n]$.

Proof. Recall that we showed in Eq. (20) that

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

We compute the variance of each of the terms in this formula. For $(\overline{x} - \widetilde{x})\widehat{\delta}_s$ we have

$$\mathbf{Var}[\overline{\mu}_i^{-1}(\overline{x}_i - \widetilde{x}_i)\widehat{\delta}_{s,i}] = \mathbf{Var}[\overline{x}_i^{-1}(\overline{x}_i - \widetilde{x}_i)\overline{s}_i^{-1}\widehat{\delta}_{s,i}] \le 4\epsilon_{\mathrm{mp}}^2\,\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] \le\ 8\epsilon_{\mathrm{mp}}^2\epsilon^2/b \qquad (23)$$

where the second step is by $\overline{x} \approx_{2\epsilon_{\mathrm{mp}}} \widetilde{x}$ (Part 2 of Fact B.9), and the third step is by $\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] \le 2\epsilon^2/b$ (Part 2 of Lemma B.16).

And similarly for $(\overline{s} - \widetilde{s})\widehat{\delta}_x$ we can show

$$\mathbf{Var}[\overline{\mu}_i^{-1}(\overline{s}_i - \widetilde{s}_i)\widehat{\delta}_{x,i}] \le 8\epsilon_{\mathrm{mp}}^2\epsilon^2/b. \tag{24}$$

Now we can upper bound the variance of $\overline{\mu}_i^{-1}\overline{\mu}_i^{\mathrm{new}}$,

$$\begin{aligned}
\mathbf{Var}[\overline{\mu}_i^{-1}\overline{\mu}_i^{\mathrm{new}}] &\le 4\,\mathbf{Var}[\overline{\mu}_i^{-1}\widetilde{\delta}_{\mu,i}] + 4\,\mathbf{Var}[\overline{\mu}_i^{-1}(\overline{x}_i - \widetilde{x}_i)\widehat{\delta}_{s,i}] + 4\,\mathbf{Var}[\overline{\mu}_i^{-1}(\overline{s}_i - \widetilde{s}_i)\widehat{\delta}_{x,i}] + 4\,\mathbf{Var}[\overline{\mu}_i^{-1}\widehat{\delta}_{x,i}\widehat{\delta}_{s,i}] \\
&\le 4 \cdot 0 + 8\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 8\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 4\,\mathbf{Var}[\overline{\mu}_i^{-1}\widehat{\delta}_{x,i}\widehat{\delta}_{s,i}] \\
&= 16\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 4\,\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i} \cdot \overline{s}_i^{-1}\widehat{\delta}_{s,i}] \\
&\le 16\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 8\,\mathbf{Sup}[(\overline{x}_i^{-1}\widehat{\delta}_{x,i})^2] \cdot \mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] + 8\,\mathbf{Sup}[(\overline{s}_i^{-1}\widehat{\delta}_{s,i})^2] \cdot \mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}] \\
&\le 16\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 8 \cdot (3\epsilon)^2 \cdot \frac{2\epsilon^2}{b} + 8 \cdot (3\epsilon)^2 \cdot \frac{2\epsilon^2}{b} \\
&\le 16\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 320\epsilon^4/b,
\end{aligned}$$

where the first step follows from triangle inequality and the fact that $\mathbf{Var}[1] = 0$, the second step follows by $\mathbf{Var}[\overline{\mu}_i^{-1}\widetilde{\delta}_{\mu,i}] = 0$ (since $\overline{\mu}_i^{-1}$ and $\widetilde{\delta}_{\mu,i}$ don't involve randomness) and plugging in Eq. (23) and Eq. (24), the third step follows by $\overline{\mu} = \overline{x} \cdot \overline{s}$ (Definition B.3), the fourth step follows by $\mathbf{Var}[xy] \le 2\,\mathbf{Sup}[x^2]\,\mathbf{Var}[y] + 2\,\mathbf{Sup}[y^2]\,\mathbf{Var}[x]$ (Lemma B.11) with $\mathbf{Sup}$ denoting the deterministic maximum of the random variable, the fifth step follows by $\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] \le 2\epsilon^2/b$ and $\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}] \le 2\epsilon^2/b$ (Part 2 of Lemma B.16) and $\|\overline{x}^{-1}\widehat{\delta}_x\|_\infty \le 3\epsilon$ and $\|\overline{s}^{-1}\widehat{\delta}_s\|_\infty \le 3\epsilon$ (Part 4 of Lemma B.16). □

Claim B.25 (Part 3 of Lemma B.21)**.** $\|\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu})\|_\infty \le 6\epsilon$ *holds with probability at least* $1 - 1/\operatorname{poly}(n)$.

Proof. We again note that from Eq. (20) we have

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

Hence, we have that with probability at least $1 - 1/n^4$ the following is true:

$$\begin{aligned}
|\overline{\mu}_i^{-1}(\overline{\mu}_i^{\mathrm{new}} - \overline{\mu}_i - \widetilde{\delta}_{\mu,i})| &\le |(\overline{x} - \widetilde{x})_i\overline{\mu}_i^{-1}\widehat{\delta}_{s,i}| + |(\overline{s} - \widetilde{s})_i\overline{\mu}_i^{-1}\widehat{\delta}_{x,i}| + |\overline{\mu}_i^{-1}\widehat{\delta}_{x,i}\widehat{\delta}_{s,i}| \\
&= |(\overline{x} - \widetilde{x})_i\overline{x}_i^{-1}| \cdot |\overline{s}_i^{-1}\widehat{\delta}_{s,i}| + |(\overline{s} - \widetilde{s})_i\overline{s}_i^{-1}| \cdot |\overline{x}_i^{-1}\widehat{\delta}_{x,i}| + |\overline{x}_i^{-1}\widehat{\delta}_{x,i}| \cdot |\overline{s}_i^{-1}\widehat{\delta}_{s,i}| \\
&\le 2\epsilon_{\mathrm{mp}}|\overline{s}_i^{-1}\widehat{\delta}_{s,i}| + 2\epsilon_{\mathrm{mp}}|\overline{x}_i^{-1}\widehat{\delta}_{x,i}| + |\overline{x}_i^{-1}\widehat{\delta}_{x,i}| \cdot |\overline{s}_i^{-1}\widehat{\delta}_{s,i}| \\
&\le 2\epsilon_{\mathrm{mp}} \cdot 3\epsilon + 2\epsilon_{\mathrm{mp}} \cdot 3\epsilon + (3\epsilon)^2 \\
&\le 20\epsilon_{\mathrm{mp}} \cdot \epsilon + 10\epsilon^2,
\end{aligned} \tag{25}$$

where the first step follows by triangle inequality, the second step follows by $\overline{\mu}_i = \overline{x}_i\overline{s}_i$ (Definition B.3), the third step follows by $\overline{x} \approx_{2\epsilon_{\mathrm{mp}}} \widetilde{x}$ and $\overline{s} \approx_{2\epsilon_{\mathrm{mp}}} \widetilde{s}$ (Part 2 of Fact B.9), the forth step follows by $|\overline{s}_i^{-1}\widehat{\delta}_{s,i}| \le 3\epsilon$ and $|\overline{x}_i^{-1}\widehat{\delta}_{x,i}| \le 3\epsilon$ holds with $1 - 1/n^4$ (Part 4 of Lemma B.16).

Finally, we have

$$\begin{aligned}
|\overline{\mu}_i^{-1}(\overline{\mu}_i^{\mathrm{new}} - \overline{\mu}_i)| &\le |\overline{\mu}_i^{-1}(\overline{\mu}_i^{\mathrm{new}} - \overline{\mu}_i - \widetilde{\delta}_{\mu,i})| + |\overline{\mu}_i^{-1}\widetilde{\delta}_{\mu,i}| \le 20\epsilon_{\mathrm{mp}} \cdot \epsilon + 10\epsilon^2 + |\overline{\mu}_i^{-1}\widetilde{\delta}_{\mu,i}| \\
&\le 20\epsilon_{\mathrm{mp}} \cdot \epsilon + 10\epsilon^2 + 5\epsilon \;\le\; 6\epsilon,
\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from Eq.(25), and the third step follows from $|\overline{\mu}_i^{-1}\widetilde{\delta}_{\mu,i}| \le 5\epsilon$ (Part 3 of Lemma B.16), and fourth step follows from $\epsilon, \epsilon_{\mathrm{mp}} \le 10^{-4}$ (Assumption B.1). □

Notation	ϵ	ϵ_{mp}	λ	b
Choice	$10^{-7}/\log n$	$10^{-5}/\log n$	$40\log n$	$10^{22}\sqrt{n}\log^{10} n$

Table 7: Choice of ϵ, ϵ_{mp}, λ and b that satisfies all constraints in Assumption B.5 and Assumption B.26. These parameters are assigned in MAIN procedure (Algorithm 17). Later they are used to prove Theorem G.3.

B.5 Potential martingale

We first state the constraints of the parameters.

Assumption B.26. *Let parameters $b, \lambda, \epsilon, \epsilon_{\mathrm{mp}}$ satisfying the following constraints:*

1. $b \geq 20000 \cdot (\lambda\epsilon_{\mathrm{mp}}^2\epsilon\sqrt{n} + \epsilon^3\sqrt{n})$,
2. $\lambda \geq 30\log n$,
3. $b \geq 1000\log^2 n$,
4. $\frac{1}{30\lambda} \geq \frac{\epsilon}{\sqrt{n}} + 8\epsilon$,
5. $b \geq 20000\epsilon\sqrt{n}$,
6. $\lambda\epsilon < 10^{-5}$,
7. $\lambda \leq 60\log n$,
8. $1.2\epsilon_{\mathrm{mp}} < 1/30\lambda$.

Now we are ready to prove the main lemma for bounding the potential function. The goal of this section is to prove Lemma B.27.

Lemma B.27 (A deep version of Lemma 4.13 in [CLS19])**.** *Under the Assumptions B.1, B.5, and B.26, we have*

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

Proof. Let $\epsilon_\mu = \overline{\mu}^{\mathrm{new}} - \overline{\mu} - \overline{\delta}_t - \widetilde{\delta}_\Phi$. From this definition, we have

$$\overline{\mu}^{\mathrm{new}} - t^{\mathrm{new}} = \overline{\mu} + \overline{\delta}_t + \widetilde{\delta}_\Phi + \epsilon_\mu - t^{\mathrm{new}},$$

which implies

$$\begin{aligned}\frac{\overline{\mu}^{\mathrm{new}}}{t^{\mathrm{new}}} - 1 &= \frac{\overline{\mu}}{t^{\mathrm{new}}} + \frac{1}{t^{\mathrm{new}}}(\overline{\delta}_t + \widetilde{\delta}_\Phi + \epsilon_\mu) - 1 = \frac{\overline{\mu}}{t} + \frac{\overline{\mu}}{t}(\frac{t}{t^{\mathrm{new}}} - 1) + \frac{1}{t^{\mathrm{new}}}(\overline{\delta}_t + \widetilde{\delta}_\Phi + \epsilon_\mu) - 1 \\ &= \frac{\overline{\mu}}{t} - 1 + \underbrace{\frac{\overline{\mu}}{t}(\frac{t}{t^{\mathrm{new}}} - 1) + \frac{1}{t^{\mathrm{new}}}(\overline{\delta}_t + \widetilde{\delta}_\Phi + \epsilon_\mu)}_{v}. \end{aligned} \tag{26}$$

To apply Lemma B.13 with $r = \overline{\mu}/t - 1$ and $r + v = \overline{\mu}^{\mathrm{new}}/t^{\mathrm{new}} - 1$, we first compute $\mathbb{E}[v]$:

$$\begin{aligned}\mathbb{E}[v] &= \frac{\overline{\mu}}{t}(\frac{t}{t^{\mathrm{new}}} - 1) + \frac{1}{t^{\mathrm{new}}}(\overline{\delta}_t + \widetilde{\delta}_\Phi + \mathbb{E}[\epsilon_\mu]) \\ &= \frac{\overline{\mu}}{t}(\frac{t}{t^{\mathrm{new}}} - 1) + \frac{1}{t^{\mathrm{new}}}\left((\frac{t^{\mathrm{new}}}{t} - 1)\overline{\mu} - \frac{\epsilon}{2}t^{\mathrm{new}}\frac{\nabla\Phi_\lambda(\widetilde{\mu}/t - 1)}{\|\nabla\Phi_\lambda(\widetilde{\mu}/t - 1)\|_2} + \mathbb{E}[\epsilon_\mu]\right) \\ &= -\frac{\epsilon}{2}\frac{\nabla\Phi_\lambda(\widetilde{\mu}/t - 1)}{\|\nabla\Phi_\lambda(\widetilde{\mu}/t - 1)\|_2} + \frac{1}{t^{\mathrm{new}}}\mathbb{E}[\epsilon_\mu], \end{aligned} \tag{27}$$

where the second step follows by definition of $\overline{\delta}_t$ (Definition B.3) and $\widetilde{\delta}_\Phi$ (Definition B.4).

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

$$\begin{aligned}\|v\|_\infty \leq & \left\|\frac{\overline{\mu}}{t}(\frac{t}{t^{\mathrm{new}}}-1)\right\|_\infty + \left\|\frac{1}{t^{\mathrm{new}}}(\overline{\mu}^{\mathrm{new}}-\overline{\mu})\right\|_\infty \leq \frac{\epsilon}{\sqrt{n}} + \frac{\|\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}}-\overline{\mu})\|_\infty}{0.9} \\ \leq & \frac{\epsilon}{\sqrt{n}} + 8\epsilon \leq \frac{1}{30\lambda},\end{aligned}$$

where the second step follows from $t^{\mathrm{new}} = (1-\epsilon/(3\sqrt{n}))\cdot t$ (Definition B.8) and $\overline{\mu} \approx_{0.1} t$ (Part 2 of Assumption B.5), the third step follows from Part 3 of Lemma B.21, and the last step follows from Part 4 of Assumption B.26 that $\frac{1}{30\lambda} \geq \frac{\epsilon}{\sqrt{n}} + 8\epsilon$.

Since $\|v\|_\infty \leq \frac{1}{30\lambda}$, we can apply Part 1 of Lemma B.13 and get

$$\begin{aligned}\mathbb{E}[\Phi_\lambda(\overline{\mu}/t+v-1)] \leq & \Phi_\lambda(\overline{\mu}/t-1) + \langle \nabla\Phi_\lambda(\overline{\mu}/t-1), \mathbb{E}[v]\rangle + 2\,\mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\overline{\mu}/t-1)}] \\ = & \Phi_\lambda(\overline{\mu}/t-1) + \underbrace{\Big(-\frac{\epsilon}{2}\Big\langle \nabla\Phi_\lambda(\overline{\mu}/t-1), \frac{\nabla\Phi_\lambda(\widetilde{\mu}/t-1)}{\|\nabla\Phi_\lambda(\widetilde{\mu}/t-1)\|_2}\Big\rangle\Big)}_{a_1} \\ & + \underbrace{\frac{t}{t^{\mathrm{new}}}\langle \nabla\Phi_\lambda(\overline{\mu}/t-1), \mathbb{E}[t^{-1}\epsilon_\mu]\rangle}_{a_2} + \underbrace{2\,\mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\overline{\mu}/t-1)}]}_{a_3}, \end{aligned} \tag{28}$$

where the second step follows by Eq. (27).

We have $\|(\widetilde{\mu}-\overline{\mu})/t\| \leq 1.1\epsilon_{\mathrm{mp}} \leq \frac{1}{30\lambda}$ since $\widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$ and $\overline{\mu} \approx_{0.1} t$ (Assumption B.5) and $1.2\epsilon_{\mathrm{mp}} < 1/30\lambda$ (Part 8 of Assumption B.26). So we can use Lemma B.15 and let $r \leftarrow \overline{\mu}/t-1$ and $v \leftarrow (\widetilde{\mu}-\overline{\mu})/t-1$ in the lemma statement to upper bound the a_1 term in Eq. (28):

$$a_1 \leq -0.45\epsilon\|\nabla\Phi_\lambda(\overline{\mu}/t-1)\|_2 + 0.1\lambda\epsilon\sqrt{n}. \tag{29}$$

We upper bound a_2 term in Eq. (28) as follows:

$$\begin{aligned}a_2 = & \frac{t}{t^{\mathrm{new}}}\langle \nabla\Phi_\lambda(\overline{\mu}/t-1), \mathbb{E}[t^{-1}\epsilon_\mu]\rangle \leq \frac{t}{t^{\mathrm{new}}}\|\nabla\Phi_\lambda(\overline{\mu}/t-1)\|_2 \cdot \|\,\mathbb{E}[t^{-1}\epsilon_\mu]\|_2 \\ \leq & 1.1\|\nabla\Phi_\lambda(\overline{\mu}/t-1)\|_2 \cdot \|\,\mathbb{E}[t^{-1}\epsilon_\mu]\|_2 \leq 2(9\epsilon_{\mathrm{mp}}\epsilon + 4\epsilon^2 + 2\epsilon^2\sqrt{n}/b)\|\nabla\Phi_\lambda(\overline{\mu}/t-1)\|_2 \end{aligned} \tag{30}$$

where the second step follows by $\langle a,b\rangle \leq \|a\|_2\cdot\|b\|_2$, the third step follows from definition of t^{new} (Definition B.8), the forth step follows by $\|\,\mathbb{E}[\overline{\mu}^{-1}\epsilon_\mu]\|_2 \leq 9\epsilon_{\mathrm{mp}}\epsilon + 4\epsilon^2 + 2\epsilon^2\sqrt{n}/b$ (Part 1 of Lemma B.21) and $\overline{\mu} \approx_{0.1} t$ (Part 2 of Assumption B.5).

We still need to bound $a_3 = 2\,\mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\overline{\mu}/t-1)}]$ term in Eq. (28). Before bounding it, we first bound $\mathbb{E}[v_i^2]$,

$$\begin{aligned}\mathbb{E}[v_i^2] \leq & 2\,\mathbb{E}\left[\left(\frac{\overline{\mu}_i}{t}(\frac{t}{t^{\mathrm{new}}}-1)\right)^2\right] + 2\,\mathbb{E}\left[\left(\frac{1}{t^{\mathrm{new}}}(\overline{\mu}_i^{\mathrm{new}}-\overline{\mu}_i)\right)^2\right] \leq \epsilon^2/n + 3\,\mathbb{E}\left[((\overline{\mu}_i^{\mathrm{new}}-\overline{\mu}_i)/\overline{\mu}_i)^2\right] \\ = & \epsilon^2/n + 3\,\mathbf{Var}[(\overline{\mu}_i^{\mathrm{new}}-\overline{\mu}_i)/\overline{\mu}_i] + 3(\mathbb{E}[(\overline{\mu}_i^{\mathrm{new}}-\overline{\mu}_i)/\overline{\mu}_i])^2 \\ \leq & \epsilon^2/n + 40\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 1000\epsilon^4/b + 3(\mathbb{E}[(\overline{\mu}_i^{\mathrm{new}}-\overline{\mu}_i)/\overline{\mu}_i])^2, \end{aligned} \tag{31}$$

where the first step follows by definition of v (see Eq. (26)), the second step follows by $\overline{\mu} \approx_{0.1} t$ (Part 2 of Assumption B.5) and $(t/t^{\mathrm{new}}-1)^2 \leq \epsilon^2/(4n)$ (Definition B.8), the third step follows by $\mathbb{E}[x^2] = \mathbf{Var}[x] + (\mathbb{E}[x])^2$, the fourth step follows by Part 2 of Lemma B.21.

Now, we are ready to bound $a_3/2 = \mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\overline{\mu}/t-1)}]$:

$$\mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\overline{\mu}/t-1)}]$$

$$
\begin{aligned}
&= \lambda^2 \sum_{i=1}^n \mathbb{E}[\Phi_\lambda(\overline{\mu}/t-1)_i v_i^2] \\
&\leq \lambda^2 \sum_{i=1}^n \Phi_\lambda(\overline{\mu}/t-1)_i \cdot (\epsilon^2/n + 40\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 1000\epsilon^4/b + 3(\mathbb{E}[(\overline{\mu}_i^{\mathrm{new}} - \overline{\mu}_i)/\overline{\mu}_i])^2 \\
&= (\epsilon^2/n + 40\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 1000\epsilon^4/b)\lambda^2 \cdot \Phi_\lambda(\overline{\mu}/t-1) \\
&\quad + 3\lambda^2 \sum_{i=1}^n \Phi_\lambda(\overline{\mu}/t-1)_i \cdot (\mathbb{E}[(\overline{\mu}_i^{\mathrm{new}} - \overline{\mu}_i)/\overline{\mu}_i])^2, \qquad (32)
\end{aligned}
$$

where the first step follows by defining $\Phi_\lambda(x)_i = \cosh(\lambda x_i)$, the second step follows from Eq. (31).

For the second term in Eq. (32), we can upper bound it in the following way:

$$
\begin{aligned}
3\lambda^2 \sum_{i=1}^n \Phi_\lambda(\overline{\mu}/t-1)_i \cdot (\mathbb{E}[(\overline{\mu}_i^{\mathrm{new}} - \overline{\mu}_i)/\overline{\mu}_i])^2 &\leq 3\lambda \Big(\sum_{i=1}^n \lambda^2 \Phi_\lambda(\overline{\mu}/t-1)_i^2 \Big)^{1/2} \cdot \| \mathbb{E}[\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu})] \|_4^2 \\
&\leq 3\lambda \left(\lambda\sqrt{n} + \|\nabla \Phi_\lambda(\overline{\mu}/t-1)\|_2 \right) \cdot (6\epsilon + 2\epsilon^2\sqrt{n}/b)^2 \\
&\leq 3\lambda \left(\lambda\sqrt{n} + \|\nabla \Phi_\lambda(\overline{\mu}/t-1)\|_2 \right) \cdot (100\epsilon^2 + 10\epsilon^4 n/b^2) \qquad (33)
\end{aligned}
$$

where the first step follows from Cauchy-Schwarz inequality, the second step follows from Part 3 of Lemma B.13 and the fact that $\| \mathbb{E}[\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu})] \|_4^2 \leq \| \mathbb{E}[\overline{\mu}^{-1}(\overline{\mu}^{\mathrm{new}} - \overline{\mu})] \|_2^2 \leq (6\epsilon + 2\epsilon^2\sqrt{n}/b)^2$ (Part 4 of Lemma B.21), the last step follows by $(a+b)^2 \leq 2a^2 + 2b^2$.

Combining Eq. (32) and Eq. (33), we have

$$
\begin{aligned}
a_3 &= 2\,\mathbb{E}[\|v\|^2_{\nabla^2 \Phi_\lambda(\overline{\mu}/t-1)}] \\
&\leq 2(\epsilon^2/n + 40\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 1000\epsilon^4/b)\lambda^2 \cdot \Phi_\lambda(\overline{\mu}/t-1) \\
&\quad + 6\lambda \left(\lambda\sqrt{n} + \|\nabla \Phi_\lambda(\overline{\mu}/t-1)\|_2 \right) \cdot (100\epsilon^2 + 10\epsilon^4 n/b^2) \qquad (34)
\end{aligned}
$$

Then, loading Eq. (29), (30), (34) back into Eq. (28)

$$
\begin{aligned}
\mathbb{E}[\Phi_\lambda(\overline{\mu}/t + v - 1)] &\leq \Phi_\lambda(\overline{\mu}/t-1) + a_1 + a_2 + a_3 \\
&\leq \Phi_\lambda(\overline{\mu}/t-1) + (\text{Eq. (29)}) + (\text{Eq. (30)}) + (\text{Eq. (34)}) \\
&= \Phi_\lambda(\overline{\mu}/t-1) + \Phi_\lambda(\overline{\mu}/t-1) \cdot (b_{1,\Phi} + b_{2,\Phi} + b_{3,\Phi}) \\
&\quad + \|\nabla \Phi_\lambda(\overline{\mu}/t-1)\|_2 \cdot (b_{1,\nabla} + b_{2,\nabla} + b_{3,\nabla}) \\
&\quad + \lambda\epsilon\sqrt{n} \cdot (b_{1,\sqrt{n}} + b_{2,\sqrt{n}} + b_{3,\sqrt{n}})
\end{aligned}
$$

where we define terms that come from a_1:

$$
b_{1,\Phi} = 0, \quad b_{1,\nabla} = -0.45\epsilon, \quad b_{1,\sqrt{n}} = 0.1,
$$

and terms that come from a_2:

$$
b_{2,\Phi} = 0, \quad b_{2,\nabla} = 2(9\epsilon_{\mathrm{mp}}\epsilon + 4\epsilon^2 + 2\epsilon^2\sqrt{n}/b) = \epsilon \cdot 2(9\epsilon_{\mathrm{mp}} + 4\epsilon + 2\epsilon\sqrt{n}/b), \quad b_{2,\sqrt{n}} = 0,
$$

and terms that come from a_3:

$$
\begin{aligned}
b_{3,\Phi} &= 2(\epsilon^2/n + 40\epsilon_{\mathrm{mp}}^2\epsilon^2/b + 1000\epsilon^4/b)\lambda^2 = (\lambda\epsilon/\sqrt{n}) \cdot 2(\lambda\epsilon/\sqrt{n} + 40\epsilon_{\mathrm{mp}}^2\lambda\epsilon\sqrt{n}/b + 1000\lambda\epsilon^3\sqrt{n}/b), \\
b_{3,\nabla} &= 6\lambda(100\epsilon^2 + 10\epsilon^4 n/b^2) = \epsilon \cdot 6(100\lambda\epsilon + 10\lambda\epsilon \cdot \epsilon^2 n/b^2),
\end{aligned}
$$

$b_{3,\sqrt{n}} = 6(100\lambda\epsilon + 10\lambda\epsilon^3 n/b^2)$.

Note that, if $b \geq 20000\epsilon\sqrt{n}$ (Part 5 of Assumption B.26), $\epsilon_{\mathrm{mp}} < 1/1000$ (Assumption B.1), $\lambda\epsilon < 10^{-5}$ (Part 6 of Assumption B.26), we have

$$b_{1,\nabla} + b_{2,\nabla} + b_{3,\nabla} < -0.4\epsilon \tag{35}$$

Thus, using Eq. (35) and Part 2 of Lemma B.13, we have

$$\begin{aligned}
\mathbb{E}[\Phi_\lambda(\overline{\mu}/t + v - 1)] \leq\ & \Phi_\lambda(\overline{\mu}/t - 1) + \Phi_\lambda(\overline{\mu}/t - 1) \cdot (b_{1,\Phi} + b_{2,\Phi} + b_{3,\Phi}) \\
& + \|\nabla\Phi_\lambda(\overline{\mu}/t - 1)\|_2 \cdot (-0.4\epsilon) \\
& + \lambda\epsilon\sqrt{n} \cdot (b_{1,\sqrt{n}} + b_{2,\sqrt{n}} + b_{3,\sqrt{n}}) \\
\leq\ & \Phi_\lambda(\overline{\mu}/t - 1) + \Phi_\lambda(\overline{\mu}/t - 1) \cdot (b_{1,\Phi} + b_{2,\Phi} + b_{3,\Phi}) \\
& + \frac{\lambda}{\sqrt{n}}(\Phi_\lambda(\overline{\mu}/t - 1) - n) \cdot (-0.4\epsilon) \\
& + \lambda\epsilon\sqrt{n} \cdot (b_{1,\sqrt{n}} + b_{2,\sqrt{n}} + b_{3,\sqrt{n}}) \\
=\ & \Phi_\lambda(\overline{\mu}/t - 1) \cdot \underbrace{(1 + b_{1,\Phi} + b_{2,\Phi} + b_{3,\Phi} - 0.4\lambda\epsilon/\sqrt{n})}_{c_\Phi} \\
& + \lambda\epsilon\sqrt{n} \cdot \underbrace{(b_{1,\sqrt{n}} + b_{2,\sqrt{n}} + b_{3,\sqrt{n}} + 0.4)}_{c_{\sqrt{n}}},
\end{aligned}$$

where the first step follows from Eq. (35) and the second step follows from Part 2 of Lemma B.13.

If $b \geq 20000 \cdot (\lambda\epsilon_{\mathrm{mp}}^2\epsilon\sqrt{n} + \epsilon^3\sqrt{n})$ (Part 1 of Assumption B.26) and $\lambda\epsilon < 10^{-5}$ (Part 6 of Assumption B.26), we have $c_\Phi \leq 1 - 0.2\lambda\epsilon/\sqrt{n}$.

If $\lambda\epsilon < 10^{-5}$ (Part 6 of Assumption B.26) and $b \geq 20000\epsilon\sqrt{n}$ (Part 5 of Assumption B.26), we have $c_{\sqrt{n}} \leq 0.6$.

Thus, we obtain

$$\begin{aligned}
\mathbb{E}[\Phi_\lambda(\overline{\mu}/t + v - 1)] &\leq \Phi_\lambda(\overline{\mu}/t - 1) \cdot (1 - 0.2\lambda\epsilon/\sqrt{n}) + 0.6\lambda\epsilon\sqrt{n} \\
&\leq \Phi_\lambda(\overline{\mu}/t - 1) - \frac{\lambda\epsilon}{15\sqrt{n}}(\Phi_\lambda(\overline{\mu}/t - 1) - 10n).
\end{aligned}$$

□

B.6 Bounding the movement of $\overline{w}$

The goal of this section is to prove Lemma B.28

Lemma B.28 (Bounding the movement of $\overline{w}$)**.** *Let $\overline{x}^{\mathrm{new}} = \overline{x} + \widehat{\delta}_x$, $\overline{s}^{\mathrm{new}} = \overline{s} + \widehat{\delta}_s$, $\overline{w} = \frac{\overline{x}}{\overline{s}}$, and $\overline{w}^{\mathrm{new}} = \frac{\overline{x}^{\mathrm{new}}}{\overline{s}^{\mathrm{new}}}$ (same as Definition B.8). Let b denote the size of sketching matrix. Then we have*

1. $\sum_{i=1}^n (\mathbb{E}[\overline{w}_i^{\mathrm{new}}]/\overline{w}_i - 1)^2 \leq 100\epsilon^2$,
2. $\sum_{i=1}^n \left(\mathbb{E}\left[(\overline{w}_i^{\mathrm{new}}/\overline{w}_i - 1)^2\right]\right)^2 \leq 10^3 \cdot \epsilon^4 n/b^2 + 4 \cdot 10^4 \cdot \epsilon^4$,
3. $|\overline{w}_i^{\mathrm{new}}/\overline{w}_i - 1| \leq 10\epsilon$.

Proof. From the definition, we know that

$$\frac{\overline{w}_i^{\text{new}}}{\overline{w}_i} = \frac{1}{\overline{s}_i^{-1}\overline{x}_i}\frac{\overline{x}_i + \widehat{\delta}_{x,i}}{\overline{s}_i + \widehat{\delta}_{s,i}} = \frac{1+\overline{x}_i^{-1}\widehat{\delta}_{x,i}}{1+\overline{s}_i^{-1}\widehat{\delta}_{s,i}}.$$

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

$$\begin{aligned}\frac{\mathbb{E}[\overline{w}_i^{\text{new}}]}{\overline{w}_i} - 1 = \mathbb{E}\left[\frac{1+\overline{x}_i^{-1}\widehat{\delta}_{x,i}}{1+\overline{s}_i^{-1}\widehat{\delta}_{s,i}}\right] - 1 &= \mathbb{E}\left[\frac{\overline{x}_i^{-1}\widehat{\delta}_{x,i} - \overline{s}_i^{-1}\widehat{\delta}_{s,i}}{1+\overline{s}_i^{-1}\widehat{\delta}_{s,i}}\right] \le 2|\,\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i} - \overline{s}_i^{-1}\widehat{\delta}_{s,i}]| \\ &\le 2|\,\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}]| + 2|\,\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}]|,\end{aligned}$$

where the third step follows from $|\overline{s}_i^{-1}\widehat{\delta}_{s,i}| \le 3\epsilon$ (Part 4 of Lemma B.16), the last step follows from triangle inequality. Then summing over all the coordinates we have

$$\sum_{i=1}^n (\mathbb{E}[\overline{w}_i^{\text{new}}]/\overline{w}_i - 1)^2 \le \sum_{i=1}^n 8(\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}])^2 + 8(\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}])^2 \le 100\epsilon^2.$$

where the first step follows by $(a+b)^2 \le 2a^2+2b^2$, the last step follows by $\|\,\mathbb{E}[\overline{s}^{-1}\widehat{\delta}_s]\|_2^2, \|\,\mathbb{E}[\overline{x}^{-1}\widehat{\delta}_x]\|_2^2 \le 4\epsilon^2$ (Part 1 of Lemma B.16).

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

$$\begin{aligned}\mathbb{E}\left[\left(\frac{\overline{w}_i^{\text{new}}}{\overline{w}_i} - 1\right)^2\right] = \mathbb{E}\left[\left(\frac{\overline{x}_i^{-1}\widehat{\delta}_{x,i} - \overline{s}_i^{-1}\widehat{\delta}_{s,i}}{1+\overline{s}_i^{-1}\widehat{\delta}_{s,i}}\right)^2\right] &\le 2\,\mathbb{E}[(\overline{x}_i^{-1}\widehat{\delta}_{x,i} - \overline{s}_i^{-1}\widehat{\delta}_{s,i})^2] \\ &\le 2\,\mathbb{E}[2(\overline{x}_i^{-1}\widehat{\delta}_{x,i})^2 + 2(\overline{s}_i^{-1}\widehat{\delta}_{s,i})^2] = 4\,\mathbb{E}[(\overline{x}_i^{-1}\widehat{\delta}_{x,i})^2] + 4\,\mathbb{E}[(\overline{s}_i^{-1}\widehat{\delta}_{s,i})^2] \\ &= 4\,\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}] + 4(\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}])^2 + 4\,\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] + 4(\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}])^2 \\ &\le 16\epsilon^2/b + 4(\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}])^2 + 4(\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}])^2,\end{aligned}$$

where the last step follows by $\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}], \mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] \le 2\epsilon^2/b$ (Part 2 of Lemma B.16).

Thus summing over all the coordinates

$$\begin{aligned}\sum_{i=1}^n \left(\mathbb{E}\left[(\overline{w}_i^{\text{new}}/\overline{w}_i - 1)^2\right]\right)^2 &\le 10^3\epsilon^4 n/b^2 + 64\sum_{i=1}^n \left((\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}])^4 + (\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}])^4\right) \\ &\le 10^3\epsilon^4 n/b^2 + 4\cdot 10^4\cdot\epsilon^4,\end{aligned}$$

where the last step follows by $\|\,\mathbb{E}[\overline{s}^{-1}\widehat{\delta}_s]\|_2^2, \|\,\mathbb{E}[\overline{x}^{-1}\widehat{\delta}_x]\|_2^2 \le 4\epsilon^2$(Part 1 of Lemma B.16).

Part 3. For each $i \in [n]$

$$\left|\frac{\overline{w}_i^{\text{new}}}{\overline{w}_i} - 1\right| = \left|\frac{1+\overline{x}_i^{-1}\widehat{\delta}_{x,i}}{1+\overline{s}_i^{-1}\widehat{\delta}_{s,i}} - 1\right| \le \left|\frac{1+3\epsilon}{1-3\epsilon} - 1\right| \le 10\epsilon,$$

where the second step follows by $|\overline{x}_i^{-1}\widehat{\delta}_{x,i}| \le 3\epsilon$ and $|\overline{s}_i^{-1}\widehat{\delta}_{s,i}| \le 3\epsilon$ (Part 4 of Lemma B.16). □

B.7 Bounding the movement of $\overline{\mu}$

The goal of this section is to prove Lemma B.29

Lemma B.29 (Bounding the movement of $\overline{\mu}$)**.** *Let* $\overline{x}^{\text{new}} = \overline{x} + \widehat{\delta}_x$, $\overline{s}^{\text{new}} = \overline{s} + \widehat{\delta}_s$, $\overline{\mu} = \overline{x} \cdot \overline{s}$, *and* $\overline{\mu}^{\text{new}} = \overline{x}^{\text{new}} \cdot \overline{s}^{\text{new}}$. *Let* b *denote the size of sketching matrix. Then we have*

1. $\sum_{i=1}^{n} (\mathbb{E}[\overline{\mu}_i^{\text{new}}]/\overline{\mu}_i - 1)^2 \leq 100\epsilon^2$,
2. $\sum_{i=1}^{n} \left(\mathbb{E}\left[(\overline{\mu}_i^{\text{new}}/\overline{\mu}_i - 1)^2 \right] \right)^2 \leq 4 \cdot 10^4 \epsilon^4 n/b^2 + 10^5 \cdot \epsilon^4$,
3. $|\overline{\mu}_i^{\text{new}}/\overline{\mu}_i - 1| \leq 10\epsilon$.

Proof. From the definition, we know that

$$\overline{\mu}_i^{\text{new}}/\overline{\mu}_i = (\overline{x}_i \overline{s}_i)^{-1} \cdot (\overline{x}_i + \widehat{\delta}_{x,i})(\overline{s}_i + \widehat{\delta}_{s,i}) = (1 + \overline{x}_i^{-1}\widehat{\delta}_{x,i})(1 + \overline{s}_i^{-1}\widehat{\delta}_{s,i}).$$

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

$$\begin{aligned}
\mathbb{E}[\overline{\mu}_i^{\text{new}}]/\overline{\mu}_i - 1 = {} & \mathbb{E}[(1 + \overline{x}_i^{-1}\widehat{\delta}_{x,i})(1 + \overline{s}_i^{-1}\widehat{\delta}_{s,i})] - 1 = \mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i} + (1 + \overline{x}_i^{-1}\widehat{\delta}_{x,i}) \cdot \overline{s}_i^{-1}\widehat{\delta}_{s,i}] \\
& \leq 2\,\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i} + \overline{s}_i^{-1}\widehat{\delta}_{s,i}] = 2\,\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}] + 2\,\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}]
\end{aligned}$$

where the third step follows by $|\overline{x}_i^{-1}\widehat{\delta}_{x,i}| \leq 3\epsilon$ (Part 4 of Lemma B.16).

Thus, summing over all the coordinates gives us

$$\sum_{i=1}^{n} (\mathbb{E}[\overline{\mu}_i^{\text{new}}]/\overline{\mu}_i - 1)^2 \leq \sum_{i=1}^{n} 8(\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}])^2 + 8(\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}])^2 \leq 100\epsilon^2,$$

where the first step follows by $(a+b)^2 \leq 2a^2 + 2b^2$, the last step is by $\| \mathbb{E}[\overline{x}^{-1}\widehat{\delta}_x] \|_2^2, \| \mathbb{E}[\overline{s}^{-1}\widehat{\delta}_s] \|_2^2 \leq 4\epsilon^2$ (Part 1 of Lemma B.16).

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

$$\begin{aligned}
\mathbb{E}[(\overline{\mu}_i^{\text{new}}/\overline{\mu}_i - 1)^2] = {} & \mathbb{E}[(\overline{x}_i^{-1}\widehat{\delta}_{x,i} + (1 + \overline{x}_i^{-1}\widehat{\delta}_{x,i}) \cdot \overline{s}_i^{-1}\widehat{\delta}_{s,i})^2] \\
& \leq 4\,\mathbb{E}[(\overline{x}_i^{-1}\widehat{\delta}_{x,i} + \overline{s}_i^{-1}\widehat{\delta}_{s,i})^2] \\
& \leq 4\,\mathbb{E}[2(\overline{x}_i^{-1}\widehat{\delta}_{x,i})^2 + 2(\overline{s}_i^{-1}\widehat{\delta}_{s,i})^2] \\
& = 8\,\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}] + 8(\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}])^2 + 8\,\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] + 8(\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}])^2 \\
& \leq 32\epsilon^2/b + 8(\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}])^2 + 8(\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}])^2,
\end{aligned}$$

where the second step follows by $|\overline{x}_i^{-1}\widehat{\delta}_{x,i}| \leq 3\epsilon$ (Part 4 of Lemma B.16), and the last step follows by $\mathbf{Var}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}]$, $\mathbf{Var}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}] \leq 2\epsilon^2/b$ (Part 2 of Lemma B.16).

Thus summing over all the coordinates

$$\begin{aligned}
\sum_{i=1}^{n} \left(\mathbb{E}[(\overline{\mu}_i^{\text{new}}/\overline{\mu}_i - 1)^2] \right)^2 & \leq 4 \cdot 10^4 \epsilon^4 n/b^2 + 256 \sum_{i=1}^{n} \left((\mathbb{E}[\overline{x}_i^{-1}\widehat{\delta}_{x,i}])^4 + (\mathbb{E}[\overline{s}_i^{-1}\widehat{\delta}_{s,i}])^4 \right) \\
& \leq 4 \cdot 10^4 \epsilon^4 n/b^2 + 10^5 \epsilon^4
\end{aligned}$$

where the second step follows by $\| \mathbb{E}[\overline{x}^{-1}\widehat{\delta}_x] \|_2^2$, $\| \mathbb{E}[\overline{s}^{-1}\widehat{\delta}_s] \|_2^2 \leq 4\epsilon^2$ (Part 1 of Lemma B.16).

Part 3. For each $i \in [n]$,

$$|\overline{\mu}_i^{\text{new}}/\overline{\mu}_i - 1| = |(1 + \overline{x}_i^{-1}\widehat{\delta}_{x,i})(1 + \overline{s}_i^{-1}\widehat{\delta}_{s,i}) - 1| \leq |(1 + 3\epsilon)^2 - 1| \leq 10\epsilon,$$

where the second step follows by $|\overline{x}_i^{-1}\widehat{\delta}_{x,i}| \leq 3\epsilon$ and $|\overline{s}_i^{-1}\widehat{\delta}_{s,i}| \leq 3\epsilon$ (Part 4 of Lemma B.16). □

Algorithm 3 One step central path

procedure OneStepCentralPath($\mathrm{mp}_t, \mathrm{mp}_\Phi, \overline{x}, \overline{s}, t, t^{\mathrm{new}}$) ▷ Lemma B.30, Lemma B.37
  $\overline{w} \leftarrow \overline{x}/\overline{s}$
  $\overline{\mu} \leftarrow \overline{x}\overline{s}$
  $(\widetilde{w}, \widetilde{g}_t, p_t) \leftarrow \mathrm{mp}_t.\textsc{UpdateQuery}(\overline{w}, \overline{\mu})$ ▷ Algorithm 8
  ▷ mp_t works with function $f_t(x) = \sqrt{x}$
  ▷ $\widetilde{w} \approx_{\epsilon_{\mathrm{mp}}} \overline{w}$, $\widetilde{g}_t \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$, $p_t = P(\widetilde{w}) f_t(\widetilde{g}_t)$
  $(\widetilde{w}, \widetilde{g}_\Phi, p_\Phi) \leftarrow \mathrm{mp}_\Phi.\textsc{UpdateQuery}(\overline{w}, \overline{\mu}/t)$ ▷ Algorithm 8
  ▷ mp_Φ works with function $f_\Phi(x) = \nabla\Phi(x-1)/\sqrt{x}$
  ▷ $\widetilde{w} \approx_{\epsilon_{\mathrm{mp}}} \overline{w}$, $\widetilde{g}_\Phi \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}/t$, $p_\Phi = P(\widetilde{w}) f_\Phi(\widetilde{g}_\Phi)$
  ▷ Two data structures will return the same $\widetilde{w}$
  $q_\Phi \leftarrow f_\Phi(\widetilde{g}_\Phi)$
  $\widetilde{\mu} \leftarrow \widetilde{g}_\Phi \cdot t$
  $\widetilde{x} \leftarrow \sqrt{\widetilde{\mu}\widetilde{w}}$
  $\widetilde{s} \leftarrow \sqrt{\widetilde{\mu}/\widetilde{w}}$ ▷ $\widetilde{x}$ and $\widetilde{s}$ satisfies $\widetilde{\mu} = \widetilde{x}\widetilde{s}$ and $\widetilde{w} = \widetilde{x}/\widetilde{s}$
  $\widetilde{\delta}_t \leftarrow (\frac{t^{\mathrm{new}}}{t} - 1)\widetilde{\mu}$
  $\widetilde{\delta}_\Phi \leftarrow -\frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{\sqrt{\widetilde{\mu}/t} \cdot q_\Phi}{\|\nabla\Phi_\lambda(\widetilde{\mu}/t-1)\|_2}$
  $\widetilde{\delta}_\mu \leftarrow \widetilde{\delta}_t + \widetilde{\delta}_\Phi$
  $p_\mu \leftarrow (\frac{t^{\mathrm{new}}}{t} - 1)p_t - \frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{p_\Phi}{\sqrt{t}\|\nabla\Phi_\lambda(\widetilde{\mu}/t-1)\|_2}$
  $\widehat{\delta}_s \leftarrow \frac{\widetilde{s}}{\sqrt{\widetilde{\mu}}} p_\mu$
  $\widehat{\delta}_x \leftarrow \frac{1}{\widetilde{s}}\widetilde{\delta}_\mu - \frac{\widetilde{x}}{\sqrt{\widetilde{\mu}}} p_\mu$
  return $(\widehat{\delta}_x, \widehat{\delta}_s)$
end procedure

B.8 One step of central path

The central path method is implemented as Algorithm 3. In this section we prove that the output of this algorithm indeed matches the definitions of previous sections. First note that the $\overline{x}$, $\overline{s}$, $\overline{w}$, and $\overline{\mu}$ matches Definition B.3, and $\widetilde{x}$, $\widetilde{s}$, $\widetilde{w}$, $\widetilde{\mu}$ matches Definition B.4.

Lemma B.30 (Correctness of one step central path)**.** *The $\widehat{\delta}_s$ and $\widehat{\delta}_x$ returned by Algorithm 3 matches the definition in Definition B.6, that*

$$\widehat{\delta}_x = \frac{\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}}(I - (R[l])^\top R[l]\widetilde{P})\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu, \qquad \widehat{\delta}_s = \frac{\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}}(R[l])^\top R[l]\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu,$$

where l is the parameter maintained in the data structure, note that the two data structures mp_t and mp_Φ use the same l and $R[l]$.

We have the following claims from Algorithm 3.

Claim B.31. *$\widetilde{\delta}_t$ (Line 15) matches Definition B.4, that $\widetilde{\delta}_t = (\frac{t^{\mathrm{new}}}{t} - 1)\widetilde{\mu}$.*

Claim B.32. *$\widetilde{\delta}_\Phi$ (Line 16) matches Definition B.4, that $\widetilde{\delta}_\Phi = -\frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{\nabla\Phi_\lambda(\widetilde{\mu}/t-1)}{\|\nabla\Phi_\lambda(\widetilde{\mu}/t-1)\|_2}$.*

Proof. We have

$$\widetilde{\delta}_\Phi = -\frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{\sqrt{\widetilde{\mu}/t} \cdot q_\Phi}{\|\nabla\Phi_\lambda(\widetilde{\mu}/t-1)\|_2} = -\frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{\sqrt{\widetilde{\mu}/t} \cdot f_\Phi(\widetilde{g}_\Phi)}{\|\nabla\Phi_\lambda(\widetilde{\mu}/t-1)\|_2} = -\frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{\nabla\Phi_\lambda(\widetilde{\mu}/t-1)}{\|\nabla\Phi_\lambda(\widetilde{\mu}/t-1)\|_2},$$

where the first step is by definition of $\widetilde{\delta}_\Phi$ (Line 16of Algorithm 3), the second step is by definition of q_Φ (Line 11 of Algorithm 3), the third step is by definition of f_Φ (Line 11 of Algorithm 17) and definition of $\widetilde{\mu}$ (Line 12 of Algorithm 3) which implies $\widetilde{g}_\phi = \widetilde{\mu}/t$. □

Claim B.33. *$\widetilde{\delta}_\mu$ (Line 17) matches Definition B.4, that $\widetilde{\delta}_\mu = \widetilde{\delta}_t + \widetilde{\delta}_\Phi$.*

Claim B.34. *p_μ (Line 18) satisfies $p_\mu = (R[l])^\top R[l] \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \widetilde{\delta}_\mu$, where $\widetilde{P}$ is defined in Definition B.4.*

Proof.

$$\begin{aligned}
p_\mu = & \left(\frac{t^{\mathrm{new}}}{t} - 1 \right) p_t - \frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{p_\Phi}{\sqrt{t} \|\nabla(\Phi_\lambda(\widetilde{\mu}/t - 1)\|_2} \\
= & \left(\frac{t^{\mathrm{new}}}{t} - 1 \right) (R[l])^\top R[l] \widetilde{P} \sqrt{\widetilde{\mu}} + \frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{(R[l])^\top R[l] \widetilde{P} \nabla \Phi_\lambda(\widetilde{\mu}/t - 1)/\sqrt{\widetilde{\mu}/t}}{\sqrt{t} \|\nabla(\Phi_\lambda(\widetilde{\mu}/t - 1)\|_2} \\
= & (R[l])^\top R[l] \widetilde{P} \frac{1}{\sqrt{\widetilde{\mu}}} \left(\frac{t^{\mathrm{new}}}{t} - 1 \right) \widetilde{\mu} + \frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{(R[l])^\top R[l] \widetilde{P} \nabla \Phi_\lambda(\widetilde{\mu}/t - 1)/\sqrt{\widetilde{\mu}}}{\|\nabla(\Phi_\lambda(\widetilde{\mu}/t - 1)\|_2} \\
= & (R[l])^\top R[l] \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \left(\left(\frac{t^{\mathrm{new}}}{t} - 1 \right) \widetilde{\mu} + \frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{\nabla \Phi_\lambda(\widetilde{\mu}/t - 1)}{\|\nabla(\Phi_\lambda(\widetilde{\mu}/t - 1)\|_2} \right) \\
= & (R[l])^\top R[l] \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} (\widetilde{\delta}_t + \widetilde{\delta}_\Phi) \quad = \quad (R[l])^\top R[l] \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \widetilde{\delta}_\mu,
\end{aligned}$$

where the first step is by definition of p_μ (Line 18), the second step is by definitions of p_t (Line 4) and p_Φ (Line 7) and the correctness of UpdateQuery(Part 2 of Theorem D.6), the fourth step is by $\widetilde{\mu} = \widetilde{x}\widetilde{s}$ (Line 13 and 14), and the last two steps are by definitions of $\widetilde{\delta}_t$, $\widetilde{\delta}_\Phi$ and $\widetilde{\delta}_\mu$ (Definition B.4). □

Claim B.35. *$\widehat{\delta}_s$ (Line 19) matches Definition B.6, that $\widehat{\delta}_s = \frac{\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}} (R[l])^\top R[l] \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \widetilde{\delta}_\mu$.*

Proof. $\widehat{\delta}_s = \frac{\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}} p_\mu = \frac{\widetilde{S}}{\sqrt{\widetilde{X}\widetilde{S}}} (R[l])^\top R[l] \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \widetilde{\delta}_\mu$ by Claim B.34. □

Claim B.36. *$\widehat{\delta}_x$ (Line 20) matches Definition B.6, that $\widehat{\delta}_x = \frac{\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}} (I - (R[l])^\top R[l] \widetilde{P}) \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \widetilde{\delta}_\mu$.*

Proof. $\widehat{\delta}_x = \frac{1}{\widetilde{s}} \widetilde{\delta}_\mu - \frac{\widetilde{x}}{\sqrt{\mu}} p_\mu = \frac{\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \widetilde{\delta}_\mu - \frac{\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}} (R[l])^\top R[l] \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \widetilde{\delta}_\mu$ by Claim B.34. □

Proof of Lemma B.30. Combine Claim B.35 and B.36. □

We also have the following lemma about the running time of Algorithm 3:

Lemma B.37 (Running time of one step central path)**.** *The cost of every operation except data structure calls in Algorithm 3 is linear in n, so the bottleneck is the two calls to the data structure.*

C Data structure : preliminary

C.1 Preliminary and Definitions

For ease of presentation, we define the following $\mathcal{L}$ operator that extends the size of a matrix. This $\mathcal{L}$ operator determines the way that our algorithm stores matrices, and executes matrix additions and multiplications.

Procedure	Algorithm	Type	Correctness	Time/Call	Amortized
Initialize	Algorithm 5	public	Lemma D.33	Lemma E.37	/
UpdateQuery	Algorithm 8	public	Theorem D.6	/	Theorem C.9
Query	Algorithm 12	private	Lemma D.7	Lemma E.3	/
UpdateV	Algorithm 9	private	Lemma D.13	/	/
UpdateG	Algorithm 10	private	Lemma D.14	/	/
MatrixUpdate	Algorithm 13	private	Lemma D.17	Lemma E.12	Lemma F.19
PartialMatrixUpdate	Algorithm 14	private	Lemma D.21	Lemma E.18	Lemma F.30
VectorUpdate	Algorithm 15	private	Lemma D.25	Lemma E.27	Lemma F.35
PartialVectorUpdate	Algorithm 16	private	Lemma D.29	Lemma E.33	Lemma F.36
ComputeLocalVariables	Algorithm 11	private	Definition D.2	Remark D.3	/

Table 8: Summary of the improved data structure. **Amortized** denotes the "amortized time".

Definition C.1 (Operator $\mathcal{L}_c$, $\mathcal{L}_r$, $\mathcal{L}$)**.** *The operator $\mathcal{L}_c$ can only be applied to some sub-columns of a matrix. For a matrix M_S where $M \in \mathbb{R}^{k_1 \times k_2}$, $S \subseteq [k_2]$ with $|S| \leq 6n^a$, $\mathcal{L}_c[M_S]$ means to store the matrix M_S in a $k_1 \times 6n^a$ block by appending extra 0s. The algorithm executes $\mathcal{L}_c$ operator in the following way:*

1. *Addition: The $\mathcal{L}_c$ operator supports storing two disjoint submatrices of the same matrix in the same block. For a matrix $M \in \mathbb{R}^{k_1 \times k_2}$ and two subsets $S_1, S_2 \subseteq [k_2]$ with $S_1 \cap S_2 = \emptyset$ and $|S_1 \cup S_2| \leq 6n^a$, $\mathcal{L}_c[M_{S_1}] + \mathcal{L}_c[M_{S_2}] := \mathcal{L}_c[M_{S_1 \cup S_2}]$.*
2. *Multiplication: When we multiply a matrix $\mathcal{L}_c[M_S]$ with column subscript S with another matrix (or vector) $(B_S)^\top$ with row subscript S, if their subscripts are the same, the algorithm will align columns of $\mathcal{L}_c[M_S]$ and rows of $(B_S)^\top$ before doing multiplication.*

In the same way, we define $\mathcal{L}_r$ as the row operator, and we define $\mathcal{L} = \mathcal{L}_r \circ \mathcal{L}_c$.

Similarly, we define $\mathcal{L}_*$ that extends a square matrix to $6n^a$ by appending an identity matrix. The motivation of appending identity matrix instead of appending 0s is to let the matrix inverse being well-defined.

Definition C.2 (Operator $\mathcal{L}_*$)**.** *For any square matrix $M \in \mathbb{R}^{k \times k}$ and $S \subseteq [k]$ with $|S| \leq 6n^a$, we define the operator $\mathcal{L}_*$ such that $\mathcal{L}_*[M_{S,S}]$ is stored in a $6n^a \times 6n^a$ block by appending 1 in the diagonal (so we have $6n^a - |S|$ extra 1s) and appending 0 otherwise. We do the same alignment as what we did for $\mathcal{L}$. The extra 1s are also involved in addition and multiplication.*

Note that in our algorithm whenever we use $\mathcal{L}_r$, $\mathcal{L}_c$, $\mathcal{L}$ or $\mathcal{L}_*$ on a matrix, we always ensure that the size of the extended rows or columns is no larger than $6n^a$. From their definitions, we directly have the following properties.

Remark C.3. *The operators $\mathcal{L}_r$, $\mathcal{L}_c$, $\mathcal{L}$ and $\mathcal{L}_*$ satisfy the following properties:*

1. ***Non-zero entries:** For any $A \in \mathbb{R}^{k_1 \times k_2}$ and two subsets $S_1, S_2 \subseteq [k_2]$ where $S_1 \subseteq S_2$ and $|S_2| \leq 6n^a$, if A only has non-zero entries on columns in S_1, then*

$$\mathcal{L}_c[A_{S_2}] = \mathcal{L}_c[A_{S_1}],$$
$$\mathcal{L}_r[(A_{S_2})^\top] = \mathcal{L}_r[(A_{S_1})^\top].$$

2. ***Addition:*** *For any $A \in \mathbb{R}^{k_1 \times k_2}$ and two subsets $S_1, S_2 \subseteq [k_2]$ with $S_1 \cap S_2 = \emptyset$ and $|S_1 \cup S_2| \leq 6n^a$,*

$$\mathcal{L}_c[A_{S_1}] + \mathcal{L}_c[A_{S_2}] = \mathcal{L}_c[A_{S_1 \cup S_2}],$$
$$\mathcal{L}_r[(A_{S_1})^\top] + \mathcal{L}_r[(A_{S_2})^\top] = \mathcal{L}_r[(A_{S_1 \cup S_2})^\top].$$

3. ***Multiplication 1:*** *For any $A \in \mathbb{R}^{k_2 \times k_1}, B \in \mathbb{R}^{k_2 \times k_3}$, and $S_1 \subseteq [k_1]$, $S_2 \subseteq [k_3]$ where $|S_1|, |S_2| \leq 6n^a$, we have*

$$\mathcal{L}_r[(A_{S_1})^\top \cdot B] = \mathcal{L}_r[(A_{S_1})^\top] \cdot B,$$
$$\mathcal{L}_c[A^\top \cdot B_{S_2}] = A^\top \cdot \mathcal{L}_c[B_{S_2}].$$

4. ***Multiplication 2:*** *For any $A \in \mathbb{R}^{k_1 \times k_2}, B \in \mathbb{R}^{k_3 \times k_2}$, $C \in \mathbb{R}^{k_2 \times k_2}$, and $S_1, S_2 \subseteq [k_2]$ where $S_1 \subseteq S_2$ and $|S_2| \leq 6n^a$, we have*

$$\mathcal{L}_c[A_{S_1}] \cdot \mathcal{L}_r[(B_{S_2})^\top] = A_{S_1} \cdot (B_{S_1})^\top$$
$$\mathcal{L}_c[A_{S_2}] \cdot \mathcal{L}_r[(B_{S_1})^\top] = A_{S_1} \cdot (B_{S_1})^\top$$
$$\mathcal{L}_c[A_{S_1}] \cdot \mathcal{L}_*[C_{S_1,S_1}] \cdot \mathcal{L}_r[(B_{S_1})^\top] = A_{S_1} \cdot C_{S_1,S_1} \cdot (B_{S_1})^\top.$$

5. ***Inverse:*** *For any $C \in \mathbb{R}^{k_2 \times k_2}$, and $S \subseteq [k_2]$ where $|S| \leq 6n^a$, we have*

$$\mathcal{L}_*[(C_{S,S})^{-1}] = (\mathcal{L}_*[C_{S,S}])^{-1}.$$

C.2 Facts

We first prove the following facts that are the cornerstones of the correctness of our data structure. The first lemma shows that we can efficiently decompose low-rank matrices with certain structure.

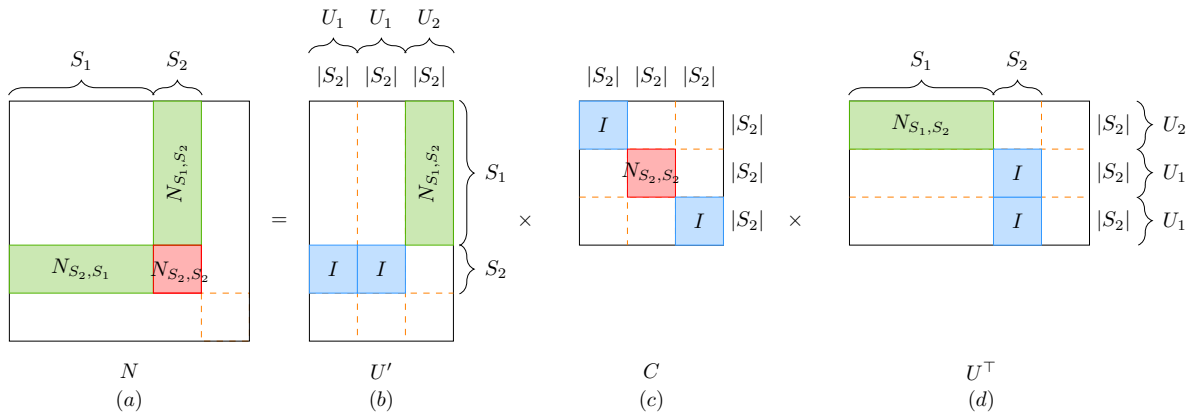


Figure 7: A visualization of the decomposition $N = U'CU^\top$ constructed by the DECOMPOSE function. (See Lemma C.4.)

Lemma C.4 ($U'CU^\top$ Decomposition)**.** *If all the non-zero entries of the $6n^a \times 6n^a$ symmetric matrix N can be split into three parts: N_{S_1,S_2}, N_{S_2,S_1}, and N_{S_2,S_2}, where $S_1, S_2 \subseteq [n]$, S_1 and S_2 are disjoint, and $|S_1 \cup S_2| \leq 2n^a$. Then there exist matrices $U', U \in \mathbb{R}^{6n^a \times 3|S_2|}, C \in \mathbb{R}^{3|S_2| \times 3|S_2|}$ such that the following decomposition holds:*

$$U'CU^\top = N.$$

Also, we can compute this decomposition in $O(|S_1| \cdot |S_2|)$ time. We define a function DECOMPOSE() *such that* DECOMPOSE$(N) = (U', C, U)$.

Proof. We explicitly give a construction of U', C and U. See Figure 7(a) for an illustration of the structure of N, and see Figure 7(b),(c),(d) for an illustration of the construction of U', C, and U.

First note that from Part 1 and 2 of Remark C.3, we have

$$N = \mathcal{L}[N_{S_1,S_2}] + \mathcal{L}[N_{S_2,S_1}] + \mathcal{L}[N_{S_2,S_2}].$$

We define $U_1 \in \mathbb{R}^{6n^a \times |S_2|}$ such that $((U_1^\top)_{S_2})^\top = I_{|S_2|}$ and all other entries of U_1 are 0. We let $U_2 = \mathcal{L}_r[N_{S_1,S_2}] \in \mathbb{R}^{6n^a \times |S_2|}$. We construct U' as $U' = [U_1, U_1, U_2]$, note that U' has size $6n^a \times 3|S_2|$. And we construct U as $U = [U_2, U_1, U_1]$, note that U also has size $6n^a \times 3|S_2|$.

We construct C as $\begin{bmatrix} I_{|S_2|} & 0 & 0 \\ 0 & N_{S_2,S_2} & 0 \\ 0 & 0 & I_{|S_2|} \end{bmatrix}$. Note that C has size $3|S_2| \times 3|S_2|$.

It is easy to check that this decomposition is correct:

$$\begin{aligned} U'CU^\top &= \begin{bmatrix} U_1 & U_1 & U_2 \end{bmatrix} \cdot \begin{bmatrix} I_{|S_2|} & 0 & 0 \\ 0 & N_{S_2,S_2} & 0 \\ 0 & 0 & I_{|S_2|} \end{bmatrix} \cdot \begin{bmatrix} U_2^\top \\ U_1^\top \\ U_1^\top \end{bmatrix} \\ &= U_1 U_2^\top + U_1 \cdot N_{S_2,S_2} \cdot U_1^\top + U_2 U_1^\top \\ &= \mathcal{L}[((U_1^\top)_{S_2})^\top \cdot U_2^\top] + \mathcal{L}[((U_1^\top)_{S_2})^\top \cdot N_{S_2,S_2} \cdot (U_1^\top)_{S_2}] + \mathcal{L}[U_2 \cdot (U_1^\top)_{S_2}] \\ &= \mathcal{L}[U_2^\top] + \mathcal{L}[N_{S_2,S_2}] + \mathcal{L}[U_2] \\ &= \mathcal{L}[N_{S_2,S_1}] + \mathcal{L}[N_{S_2,S_2}] + \mathcal{L}[N_{S_1,S_2}] = N, \end{aligned}$$

where the third step follows from the fact that U_1 only has non-zero entries on the rows in S_2 and Part 1 of Remark C.3, the fourth step follows from $((U_1^\top)_{S_2})^\top = (U_1)_{S_2,S_2} = I$, the fifth step follows from U_2 only has one block of non-zero entries: $(U_2)_{S_1,S_2} = N_{S_1,S_2}$ and Part 1 of Remark C.3.

Finally, since U', C, U are all constructed by copying certain entries of N, the running time of this decomposition is the sum of the sizes of the three matrices. Thus we can compute this decomposition in $O(|S_1| \cdot |S_2|)$ time. □

The next lemma shows that a particular matrix satisfies the constraints of the previous lemma.

Lemma C.5 (Structure of the change in inverse matrix)**.** *For $v, \widetilde{v}, w^{\text{appr}} \in \mathbb{R}^n$, and a symmetric matrix $M \in \mathbb{R}^{n \times n}$, let $S = \text{supp}(\widetilde{v} - v)$, $\partial S = \text{supp}(w^{\text{appr}} - \widetilde{v})$, $S^{\text{new}} = \text{supp}(w^{\text{appr}} - v)$, and $S' = (S \cup \partial S) \backslash S^{\text{new}}$. Let $\Delta = \widetilde{V} - V$, $\Delta^{\text{new}} = W^{\text{appr}} - V$. Let $N = \mathcal{L}_*[(\Delta^{\text{new}}_{S^{\text{new}},S^{\text{new}}})^{-1} + M_{S^{\text{new}},S^{\text{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}]$.*

Then the non-zero entries of N can be split into three parts: $N_{(S\backslash\partial S),\partial S}$, $N_{\partial S,(S\backslash\partial S)}$, and $N_{\partial S,\partial S}$, as shown in Figure 8(a). And $N_{(S\backslash\partial S),\partial S}$ has the following structure (as shown in Figure 8(b)):

- *For columns in $\partial S \backslash S$, $N_{(S\backslash\partial S),(\partial S\backslash S)} = M_{(S\backslash\partial S),(\partial S\backslash S)}$.*
- *For columns in S', $N_{(S\backslash\partial S),S'} = -M_{(S\backslash\partial S),S'}$.*
- *For other columns, $N_{(S\backslash\partial S),(S\cap\partial S)\backslash S'} = 0$.*

Proof. Note that N is a symmetric matrix. It is easy to see that $S^{\text{new}} \subseteq S \cup \partial S$ and $S' \subseteq S \cap \partial S$. We also have the following observations:

- $\forall i \in S \backslash \partial S$, $v_i \neq \widetilde{v}_i$, and $\widetilde{v}_i = w_i^{\text{appr}}$.

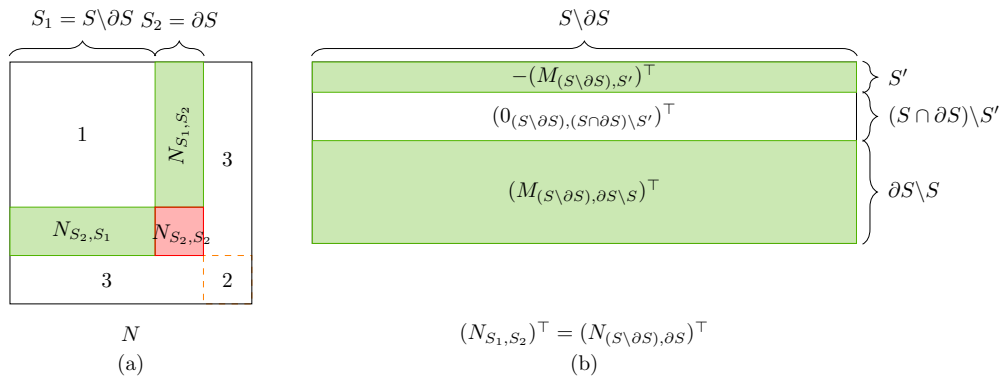


Figure 8: A visualization of the matrices involved in DECOMPOSE function. Figure (a) illustrates the structure of the input matrix N (see Lemma C.4 and Lemma C.5). Figure (b) illustrates the structure of $N_{S_1,S_2} = N_{(S\backslash\partial S),\partial S}$. (See Lemma C.5.) For clarity we show its transpose here.

- $\forall i \in \partial S\backslash S$, $v_i = \widetilde{v}_i$, and $\widetilde{v}_i \neq w_i^{\text{appr}}$.
- $\forall i \in S'$, $v_i \neq \widetilde{v}_i$, and $v_i = w_i^{\text{appr}}$.
- $\forall i \in (S \cap \partial S)\backslash S'$, $v_i \neq \widetilde{v}_i$, $v_i \neq w_i^{\text{appr}}$, and $\widetilde{v}_i \neq w_i^{\text{appr}}$.

Using the above observations and the definition of $\mathcal{L}_*$, N has the following properties:

- $\forall i, j \in S\backslash\partial S$, we have $N_{i,j} = (\Delta_{i,j}^{\text{new}} + M_{i,j}) - (\Delta_{i,j} + M_{i,j}) = 0$, where the second step follows from the fact that $\Delta_{i,i}^{\text{new}} = w_i^{\text{appr}} - v_i = \widetilde{v}_i - v_i = \Delta_{i,i}$. This means the block with label 1 in Figure 8(a) only has zeroes.
- $\forall i, j \notin S \cup \partial S$, if $i = j$, we have $N_{i,j} = 1 - 1 = 0$, otherwise we have $N_{i,j} = 0 - 0 = 0$. This means the block with label 2 in Figure 8(a) only has zeroes.
- $\forall i \in S \cup \partial S, j \notin S \cup \partial S$, we have $N_{i,j} = 0 - 0 = 0$, then we also have $N_{j,i} = N_{i,j} = 0$. This means the two blocks with label 3 in Figure 8(a) only has zeroes.

Thus we prove the first statement of this lemma: the non-zero entries of N can be split into three parts: $N_{(S\backslash\partial S),\partial S}$, $N_{\partial S,(S\backslash\partial S)}$, and $N_{\partial S,\partial S}$. Using the above observations we also have the following properties for entries in $N_{(S\backslash\partial S),\partial S}$. For any $i \in S\backslash\partial S$, note that $i \in S$ and $i \in S^{\text{new}}$.

- $\forall j \in (S\cap\partial S)\backslash S'$, note that $i \neq j$, and $j \in S$ and $j \in S^{\text{new}}$. So $N_{i,j} = (0+M_{i,j})-(0+M_{i,j}) = 0$.
- $\forall j \in S'$, note that $i \neq j$, and $j \in S$ and $j \notin S^{\text{new}}$. So $N_{i,j} = (0+0) - (0+M_{i,j}) = -M_{i,j}$.
- $\forall j \in \partial S\backslash S$, note that $i \neq j$, and $j \notin S$ and $j \in S^{\text{new}}$. So $N_{i,j} = (0+M_{i,j}) - (0+0) = M_{i,j}$.

Thus $N_{(S\backslash\partial S),\partial S}$ has the structure as described in lemma statement. □

The next lemma shows that we can use Woodbury identity together with the previous $U'CU^\top$ decomposition to efficiently maintain the inverse of a matrix.

Lemma C.6 (Correctness of B using Woodbury Identity)**.** *For $v, \widetilde{v}, w^{\text{appr}} \in \mathbb{R}^n$, and a symmetric matrix $M \in \mathbb{R}^{n\times n}$, let $S = \text{supp}(\widetilde{v} - v)$, $\partial S = \text{supp}(w^{\text{appr}} - \widetilde{v})$, $S^{\text{new}} = \text{supp}(w^{\text{appr}} - v)$, and $S' = (S \cup \partial S)\backslash S^{\text{new}}$. Let $\Delta = \widetilde{V} - V$, $\Delta^{\text{new}} = W^{\text{appr}} - V$. Let $B = \mathcal{L}_*[(\Delta_{S,S}^{-1} + M_{S,S})^{-1}]$, and let*

$$N = \mathcal{L}_*[(\Delta_{S^{\text{new}},S^{\text{new}}}^{\text{new}})^{-1} + M_{S^{\text{new}},S^{\text{new}}}] - \mathcal{L}_*[\Delta_{S,S}^{-1} + M_{S,S}].$$

Suppose $|S \cup \partial S| \leq 2n^a$*, and let* $(U', C, U) := \textsc{Decompose}(N)$*, where* $\textsc{Decompose}()$ *is the function of Lemma C.4, note that* $U', U \in \mathbb{R}^{6n^a \times (3|\partial S|)}, C \in \mathbb{R}^{(3|\partial S|)\times(3|\partial S||)}$*, then we have*

$$B - BU'(C^{-1} + U^\top BU')^{-1}U^\top B = \mathcal{L}_*[\big((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}}\big)^{-1}].$$

Proof. From Lemma C.5, we know that the non-zero entries of N can be split into three parts: $N_{(S\backslash\partial S),\partial S}$, $N_{\partial S,(S\backslash\partial S)}$, and $N_{\partial S,\partial S}$. $S\backslash\partial S$ and ∂S are disjoint, and $|S \cup \partial S| \leq 2n^a$, so N satisfies the requirements of Lemma C.4. And in the lemma statement of Lemma C.4, S_1 corresponds to $S\backslash\partial S$ here, S_2 corresponds to ∂S here. Thus U', C, U are well-defined, and we have $U'CU^\top = N$.

Also note that from the property of $\mathcal{L}_*$ operator (Part 5 of Remark C.3) we have

$$B = \mathcal{L}_*[(\Delta^{-1}_{S,S} + M_{S,S})^{-1}] = (\mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}])^{-1}.$$

Using Woodbury identity (Fact A.2), we have that

$$\begin{aligned}
\mathcal{L}_*[\big((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}}\big)^{-1}] &= \big(\mathcal{L}_*[(\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}}]\big)^{-1} \\
&= \Big(\mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}] + U'CU^\top\Big)^{-1} \\
&= \Big(B^{-1} + U'CU^\top\Big)^{-1} \\
&= B - BU'(C^{-1} + U^\top BU')^{-1}U^\top B
\end{aligned}$$

where the first step follows from Part 5 of Remark C.3, the second step follows from $U'CU^\top = N$, the third step follows from the fact that $B = (\mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}])^{-1}$, and the forth step follows from Woodbury identity. □

We can exploit the fact that the output matrices U and U' of $\textsc{Decompose}$ resembles the input matrix, and we have the following corollary:

Corollary C.7 (Correctness of U^{tmp})**.** *Given* $v, \widetilde{v}, w^{\mathrm{appr}} \in \mathbb{R}^n$*, and a symmetric matrix* $M \in \mathbb{R}^{n\times n}$*. Let* S*,* ∂S*,* S^{new}*,* S'*,* Δ*,* Δ^{new}*,* B*,* N*,* U*,* U'*, and* C *be defined the same way as in Lemma C.5 and C.6. Let* $E = B \cdot \mathcal{L}_r[(M_S)^\top]$*. Define* $\partial E \in \mathbb{R}^{6n^a\times|\partial S|}$ *such that*

$$\begin{aligned}
(\partial E)_{(\partial S\backslash S)} &= E_{(\partial S\backslash S)} - B_{(\partial S\cap S)}M_{(\partial S\cap S),(\partial S\backslash S)} \\
(\partial E)_{S'} &= -E_{S'} + B_{\partial S\cap S}M_{(\partial S\cap S),S'}
\end{aligned}$$

and other entries of ∂E *are all zero. Define* $U^{\mathrm{tmp}} \in \mathbb{R}^{6n^a\times 3|\partial S|}$ *as*

$$U^{\mathrm{tmp}} = [B_{\partial S},\ B_{\partial S},\ \partial E],$$

then we have $U^{\mathrm{tmp}} = BU'$*. Note that* ∂E *is the same one as defined on Line 12 and 12 of Algorithm 12, and* U^{tmp} *is the same one as defined on Line 15.*

Proof. From Lemma C.5 we know that N can be split into three parts: $N_{(S\backslash\partial S),\partial S}$, $N_{\partial S,(S\backslash\partial S)}$, and $N_{\partial S,\partial S}$. From the proof of Lemma C.4, we have that $U' = [U_1, U_1, U_2]$, where

1. $U_1 \in \mathbb{R}^{6n^a\times|\partial S|}$ such that $((U_1^\top)_{\partial S})^\top = I_{|\partial S|}$ and all other entries are 0,
2. $U_2 \in \mathbb{R}^{6n^a\times|\partial S|}$ and $U_2 = \mathcal{L}_r[N_{(S\backslash\partial S),\partial S}]$.

Since $BU' = [BU_1, BU_1, BU_2]$, it suffices to prove that $BU_1 = B_{\partial S}$ and $BU_2 = \partial E$. We have

$$BU_1 = B_{\partial S} \cdot ((U_1^\top)_{\partial S})^\top = B_{\partial S},$$

where the first step follows from U_1 only has non-zero rows in ∂S, and the second step follows from $((U_1^\top)_{\partial S})^\top = I_{|\partial S|}$. For BU_2 we first prove the following:

$$\begin{aligned} E_{(\partial S \setminus S)} &= B \cdot \mathcal{L}_r[M_{S,(\partial S \setminus S)}] = \ B \cdot \mathcal{L}_r[M_{(\partial S \cap S),(\partial S \setminus S)}] + B \cdot \mathcal{L}_r[M_{(S \setminus \partial S),(\partial S \setminus S)}] \\ &= B_{(\partial S \cap S)} \cdot M_{(\partial S \cap S),(\partial S \setminus S)} + B \cdot \mathcal{L}_r[M_{(S \setminus \partial S),(\partial S \setminus S)}], \end{aligned}$$

where the first step follows from $E = B \cdot \mathcal{L}_r[(M_S)^\top]$, the second step follows from Part 2 of Remark C.3, and the third step follows from Part 4 of Remark C.3. So we have

$$B \cdot \mathcal{L}_r[M_{(S \setminus \partial S),(\partial S \setminus S)}] = E_{(\partial S \setminus S)} - B_{(\partial S \cap S)} \cdot M_{(\partial S \cap S),(\partial S \setminus S)},$$

and similarly we also have $B \cdot \mathcal{L}_r[M_{(S \setminus \partial S),S'}] = E_{S'} - B_{(\partial S \cap S)} \cdot M_{(\partial S \cap S),S'}$.

Thus combining with the structure of $N_{(S \setminus \partial S),\partial S}$ proved in Lemma C.5, we have

1. For columns in $\partial S \setminus S$,

$$(BU_2)_{\partial S \setminus S} = B \cdot \mathcal{L}_r[N_{(S \setminus \partial S),(\partial S \setminus S)}] = B \cdot \mathcal{L}_r[M_{(S \setminus \partial S),(\partial S \setminus S)}] = E_{(\partial S \setminus S)} - B_{(\partial S \cap S)} \cdot M_{(\partial S \cap S),(\partial S \setminus S)}.$$

2. For columns in S',

$$(BU_2)_{S'} = B \cdot \mathcal{L}_r[N_{(S \setminus \partial S),S'}] = B \cdot \mathcal{L}_r[-M_{(S \setminus \partial S),S'}] = -E_{S'} + B_{(\partial S \cap S)} \cdot M_{(\partial S \cap S),S'}.$$

3. For all other columns, $(BU_2)_{(S \cap \partial S) \setminus S'} = 0$.

Thus we have $BU_2 = \partial E$, and therefore $BU' = U^{\mathrm{tmp}}$. □

We also have the following lemma that shows how to use Woodbury identity to update the inverse of a matrix directly.

Lemma C.8 (Correctness of M using Woodbury Identity)**.** *Let $A \in \mathbb{R}^{n \times n}$, and let $\widetilde{V}, V \in \mathbb{R}^{n \times n}$ be two diagonal matrices. Let $M = A^\top (AVA^\top)^{-1} A \in \mathbb{R}^{n \times n}$, and let $\Delta = \widetilde{V} - V \in \mathbb{R}^{n \times n}$, let $S = \mathrm{supp}(\widetilde{v} - v) \subseteq [n]$, then we have*

$$M - M_S \cdot \left((\Delta_{S,S})^{-1} + M_{S,S}\right)^{-1} \cdot (M_S)^\top = A^\top (A\widetilde{V}A^\top)^{-1} A \tag{36}$$

Proof. We have

$$\begin{aligned} (A\widetilde{V}A^\top)^{-1} &= (A(V+\Delta)A^\top)^{-1} = (AVA^\top + A\Delta A^\top)^{-1} = (AVA^\top + A_S \Delta_{S,S} (A_S)^\top)^{-1} \\ &= (AVA^\top)^{-1} - (AVA^\top)^{-1} \cdot A_S \cdot \left((\Delta_{S,S})^{-1} + (A_S)^\top (AVA^\top)^{-1} A_S\right)^{-1} \cdot (A_S)^\top \cdot (AVA^\top)^{-1} \\ &= (AVA^\top)^{-1} - (AVA^\top)^{-1} \cdot A_S \cdot \left((\Delta_{S,S})^{-1} + M_{S,S}\right)^{-1} \cdot (A_S)^\top \cdot (AVA^\top)^{-1}, \end{aligned}$$

where the first step follows from the definition of Δ, the third step follows from Δ only has non-zero entries on (i,i)-th entries where $i \in S$, the fourth step follows from Woodbury identity, and the fifth step follows from the definition of M. Then from the definition of M we have Eq. (36). □

Notation	MatrixUpdate	P.MatrixUpdate	VectorUpdate	P.VectorUpdate
v	$\surd$			
$\widetilde{v}$	$\surd$	$\surd$		
g			$\surd$	
$\widetilde{g}$			$\surd$	$\surd$
M	$\surd$			
Q	$\surd$			
β_1	$\surd$		$\surd$	
β_2	$\surd$		$\surd$	
S	$\surd$	$\surd$		
T			$\surd$	$\surd$
Δ	$\surd$	$\surd$		
Γ	$\surd$	$\surd$		
ξ	$\surd$	$\surd$	$\surd$	$\surd$
B	$\surd$	$\surd$		
E	$\surd$	$\surd$		
F	$\surd$	$\surd$		
γ_1	$\surd$	$\surd$	$\surd$	$\surd$
γ_2	$\surd$	$\surd$	$\surd$	$\surd$
Goal	$v, \widetilde{v}$	$\widetilde{v}$	$g, \widetilde{g}$	$\widetilde{g}$
Algorithm	Algorithm 13	Algorithm 14	Algorithm 15	Algorithm 16
Correctness	Lemma D.17	Lemma D.21	Lemma D.25	Lemma D.29
Time	Lemma E.12	Lemma E.18	Lemma E.27	Lemma E.33

Table 9: Summary of things got changed over different updates. List of members in Algorithm 4.

C.3 Main result

The goal of this section is to present Theorem C.9.

Theorem C.9 (Main data structure theorem)**.** *Given a full rank matrix $A \in \mathbb{R}^{d \times n}$ with $d \leq n$, two error parameters $0 < \epsilon_{\mathrm{mp}} < 1/4$ and $\epsilon_{\mathrm{far}} \leq \frac{\epsilon_{\mathrm{mp}}}{100 \log n}$, two threshold parameters $a \leq \alpha$ and $\widetilde{a} \leq \alpha \cdot a$ where α is the dual exponent of matrix multiplication, a parameter of sketching size $b \in (0,1)$. Let $f : \mathbb{R} \to \mathbb{R}$ be some pre-defined function which can be computed in $O(1)$ time, and extend the definition of f on vector v with $f(v)_i := f(v_i)$. Let ω denote the exponent of matrix multiplication. There is a data structure (in Algorithm 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16) that approximately maintains the vector*

$$\sqrt{W} A^\top (A W A^\top)^{-1} A \sqrt{W} f(h).$$

The data structure uses $n^{2+o(1)}$ space and supports the following operations:

1. Initialize$(f, \epsilon_{\mathrm{mp}}, b, L, A, w_0, h_0, R)$*: Initialize all data structure members including $L = n^{1-b+o(1)}$ sketching matrices $R_1, R_2, \ldots, R_L \in \mathbb{R}^{n^b \times n}$. This operation takes $O(n^\omega)$ time.*
2. UpdateQuery(w, h)*: On the j-th call to this function, it outputs the following three vectors (see Theorem D.6):*
   (a) *A vector $w^{\mathrm{appr}} \in \mathbb{R}^n$ such that $w^{\mathrm{appr}} \approx_{\epsilon_{\mathrm{mp}}} w$.*
   (b) *A vector $h^{\mathrm{appr}} \in \mathbb{R}^n$ such that $h^{\mathrm{appr}} \approx_{\epsilon_{\mathrm{mp}}} h$.*
   (c) *A vector $r \in \mathbb{R}^n$ such that*

$$r = R_l^\top R_l \sqrt{W^{\mathrm{appr}}} A^\top (A W^{\mathrm{appr}} A^\top) A \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}).$$

Furthermore, if the initial vectors $(w^{(0)}, h^{(0)})$ *and the update sequence* $(w^{(1)}, h^{(1)}), \ldots, (w^{(T)}, h^{(T)}) \in (\mathbb{R}^n, \mathbb{R}^n)$ *satisfy the following constraints:*

1. $$\sum_{i=1}^n \left(\frac{\mathbb{E}[w_i^{(j+1)}] - w_i^{(j)}}{w_i^{(j)}} \right)^2 \leq C_1^2, \ \sum_{i=1}^n \left(\mathbb{E}\left[\left(\frac{w_i^{(j+1)} - w_i^{(j)}}{w_i^{(j)}} \right)^2 \right] \right)^2 \leq C_2^2, \ \left| \frac{w_i^{(j+1)} - w_i^{(j)}}{w_i^{(j)}} \right| \leq C_3,$$

2. $$\sum_{i=1}^n \left(\frac{\mathbb{E}[\mu_i^{(j+1)}] - \mu_i^{(j)}}{\mu_i^{(j)}} \right)^2 \leq C_4^2, \ \sum_{i=1}^n \left(\mathbb{E}\left[\left(\frac{\mu_i^{(j+1)} - \mu_i^{(j)}}{\mu_i^{(j)}} \right)^2 \right] \right)^2 \leq C_5^2, \ \left| \frac{\mu_i^{(j+1)} - \mu_i^{(j)}}{\mu_i^{(j)}} \right| \leq C_6,$$

for $j = 0, 1, \ldots, T-1$*, where the expectation is conditioned on* $w^{(j)}$ *for Part 1, and conditioned on* $h^{(j)}$ *for Part 2, and the parameters* $C_1, C_2, C_3, C_4, C_5, C_6$ *satisfy that* $C_1, C_2, C_4, C_5 > 0$ *and* $0 < C_3, C_6 \leq \frac{1}{4}$*. Then the worst-case running time of* Query *per iteration is*

$$O^*(\mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}) + n^{1+b}),$$

and the expected amortized running time per iteration of the other procedures are

1. MatrixUpdate*:* $O^*\Big((C_1 \epsilon_{\mathrm{mp}} / \epsilon_{\mathrm{far}}^2 + C_2 / \epsilon_{\mathrm{far}}^2) \cdot (n^{2-a/2} + n^{\omega - 1/2}) \Big)$,
2. PartialMatrixUpdate*:* $O^*\Big((C_1 / \epsilon_{\mathrm{mp}} + C_2 / \epsilon_{\mathrm{mp}}^2) \cdot (n^{1+a-\widetilde{a}/2} + n^{1+(\omega-3/2)a}) \Big)$,
3. VectorUpdate*:* $O^*\Big((C_4 \epsilon_{\mathrm{mp}} / \epsilon_{\mathrm{far}}^2 + C_5 / \epsilon_{\mathrm{far}}^2) \cdot n^{1.5} \Big)$,
4. PartialVectorUpdate*:* $O^*\Big((C_4 \epsilon_{\mathrm{mp}} / \epsilon_{\mathrm{far}}^2 + C_5 / \epsilon_{\mathrm{far}}^2) \cdot (n^{1.5} + n^{2a-\widetilde{a}/2}) \Big)$,

where O^* *notation hides all* $n^{o(1)}$ *terms.*

Proof. The properties of the output of UpdateQuery follow from Theorem D.6. The running time of Initialize is by Lemma E.37, the running time of Query is by Lemma E.3. The amortized running time of MatrixUpdate is by Lemma F.19, the amortized running time of PartialMatrixUpdate is by Lemma F.30, the amortized running time of VectorUpdate is by Lemma F.35, and the amortized running time of PartialVectorUpdate is by Lemma F.36. □

We prove the correctness of the data structure in Section D. We give the worst-case analysis of the running time per call for all procedures in Section E, and the amortized analysis in Section F.

D Data structure : correctness

The purpose of this section is to show the correctness of our data structure, stated in Theorem D.6. We start with the invariants that we maintain for data structure members.

Assumption D.1 (Invariants)**.** *The following invariants are maintained in the data structure:*

Algorithm 4 Data structure : members

data structure ▷ Theorem C.9

members ▷ Table 9

- $A \in \mathbb{R}^{d \times n}$
- Function $f : \mathbb{R} \to \mathbb{R}$
- $\epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}} \in \mathbb{R}$
- $v,\ \widetilde{v},\ g,\ \widetilde{g} \in \mathbb{R}^n$
- $a, \widetilde{a}, b \in (0, 1]$
- $L \in \mathbb{N}$ ▷ number of sketching matrices
- $l \in \mathbb{N}$ ▷ count of iterations
- $\forall i \in [L],\ R_i \in \mathbb{R}^{n^b \times n}$
- $R = [R_1^\top, R_2^\top, \cdots, R_L^\top]^\top \in \mathbb{R}^{n^{1+o(1)} \times n}$ ▷ We have the guarantee that $Ln^b = n^{1+o(1)}$
- ▷ Below are the invariant variables
- $M \in \mathbb{R}^{n \times n}$
- $Q \in \mathbb{R}^{n^{1+o(1)} \times n}$
- $\beta_1 \in \mathbb{R}^{n+o(1)}$
- $\beta_2 \in \mathbb{R}^n$
- Set $S \subseteq [n]$
- Set $T \subseteq [n]$
- $\Delta \in \mathbb{R}^{n \times n}$
- $\Gamma \in \mathbb{R}^{n \times n}$
- $\xi \in \mathbb{R}^n$
- $B \in \mathbb{R}^{6n^a \times 6n^a}$
- $F \in \mathbb{R}^{n^{1+o(1)} \times 6n^a}$
- $E \in \mathbb{R}^{6n^a \times n}$
- $\gamma_1 \in \mathbb{R}^{6n^a}$
- $\gamma_2 \in \mathbb{R}^n$

end members

end data structure

1. $M = A^\top (AVA^\top)^{-1} A$,
2. $Q = R\sqrt{V}M$,
3. $\beta_1 = Q\sqrt{V}f(g)$,
4. $\beta_2 = M\sqrt{V}f(g)$,
5. $S = \mathrm{supp}(\widetilde{v} - v)$,
6. $T = \mathrm{supp}(\widetilde{g} - g)$,
7. $\Delta = \widetilde{V} - V$,
8. $\Gamma = \sqrt{\widetilde{V}} - \sqrt{V}$,
9. $\xi = \sqrt{\widetilde{V}}f(\widetilde{g}) - \sqrt{V}f(g)$,
10. $B = \mathcal{L}_*[(\Delta_{S,S}^{-1} + M_{S,S})^{-1}]$,
11. $E = B \cdot \mathcal{L}_r[(M_S)^\top]$,
12. $F = R\Gamma \cdot \mathcal{L}_c[M_S]$,
13. $\gamma_1 = B \cdot \mathcal{L}_r[\beta_{2,S}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi$,
14. $\gamma_2 = \Gamma M \cdot \xi$.

We will prove that if these invariants are true before we enter a procedure, they are still true when the proecedure returns. Thus the correctness of the invariants can be proved by induction.

The following local variables are used in procedures Query (Algorithm 12), MatrixUpdate (Algorithm 13), and PartialMatrixUpdate (Algorithm 14). For clarity of the presentation, we write their definitions here.

Definition D.2 (Local variables)**.** *Given inputs w^{appr} and h^{appr}, we define these local variables:*

Algorithm 5 Data structure : INITIALIZE()

data structure ▷ Theorem C.9

procedure INITIALIZE($f, \epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b, L, A, w_0, h_0, R$) ▷ Lemma D.33, Lemma E.37

- $f \leftarrow f$ ▷ $f : \mathbb{R} \to \mathbb{R}$
- $\epsilon_{\mathrm{mp}} \leftarrow \epsilon_{\mathrm{mp}}$ ▷ $\epsilon_{\mathrm{mp}} \in (0,1)$
- $\epsilon_{\mathrm{far}} \leftarrow \epsilon_{\mathrm{far}}$ ▷ $\epsilon_{\mathrm{far}} \in (0,1)$
- $a \leftarrow a$ ▷ $a \in (0,1]$
- $\widetilde{a} \leftarrow \widetilde{a}$ ▷ $\widetilde{a} \in (0,1]$
- $b \leftarrow b$ ▷ $b \in (0,1]$
- $L \leftarrow L$ ▷ $L \in \mathbb{N}$
- $A \leftarrow A$ ▷ $A \in \mathbb{R}^{d \times n}$
- $R \leftarrow R$ ▷ $R = [R_1^\top, R_2^\top, \cdots, R_L^\top]^\top \in \mathbb{R}^{n^{1+o(1)} \times n}$
- $l \leftarrow 1$ ▷ count of iterations
- $v \leftarrow \widetilde{v} \leftarrow w_0$ ▷ $v, \widetilde{v} \in \mathbb{R}^n$
- $g \leftarrow \widetilde{g} \leftarrow h_0$ ▷ $g, \widetilde{g} \in \mathbb{R}^n$
- ▷ Below are the invariant variables
- $M \leftarrow A^\top (AVA^\top)^{-1} A$ ▷ $M \in \mathbb{R}^{n \times n}$
- $Q \leftarrow R\sqrt{V}M$ ▷ $Q \in \mathbb{R}^{n^{1+o(1)} \times n}$
- $\beta_1 \leftarrow Q\sqrt{V}f(g)$ ▷ $\beta_1 \in \mathbb{R}^{n^{1+o(1)}}$
- $\beta_2 \leftarrow M\sqrt{V}f(g)$ ▷ $\beta_2 \in \mathbb{R}^n$
- $S \leftarrow \emptyset$ ▷ $S \subseteq [n]$
- $T \leftarrow \emptyset$ ▷ $T \subseteq [n]$
- $\Delta \leftarrow 0$ ▷ $\Delta \in \mathbb{R}^{n \times n}$ is a diagonal matrix
- $\Gamma \leftarrow 0$ ▷ $\Gamma \in \mathbb{R}^{n \times n}$ is a diagonal matrix
- $\xi \leftarrow 0$ ▷ $\xi \in \mathbb{R}^n$
- $B \leftarrow I$ ▷ $B \in \mathbb{R}^{6n^a \times 6n^a}$
- $E \leftarrow 0$ ▷ $E \in \mathbb{R}^{6n^a \times n}$
- $F \leftarrow 0$ ▷ $F \in \mathbb{R}^{n^{1+o(1)} \times 6n^a}$
- $\gamma_1 \leftarrow 0$ ▷ $\gamma_1 \in \mathbb{R}^{6n^a}$
- $\gamma_2 \leftarrow 0$ ▷ $\gamma_2 \in \mathbb{R}^n$

end procedure

end data structure

Algorithm 6 Data structure : ADJUST()

data structure

▷ This procedure doesn't use any members in the memory of data structure.

procedure ADJUST($\widetilde{v}^{\mathrm{tmp}}, \widetilde{v}, v, \epsilon_{\mathrm{far}}$)

▷ $\widetilde{v}^{\mathrm{tmp}}$ is the temporary new update of $\widetilde{v}$. $\widetilde{v}^{\mathrm{tmp}}$ is adjusted to $\widetilde{v}^{\mathrm{adj}}$.

- $\widetilde{v}^{\mathrm{adj}} \leftarrow \widetilde{v}^{\mathrm{tmp}}$
- **for** $i = 1$ to n **do**
  - **if** $\widetilde{v}_i^{\mathrm{tmp}} \neq \widetilde{v}_i$ and $\widetilde{v}_i^{\mathrm{tmp}} \in [(1-\epsilon_{\mathrm{far}})v_i, (1+\epsilon_{\mathrm{far}})v_i]$ **then**
    - $\widetilde{v}_i^{\mathrm{adj}} \leftarrow v_i$
  - **end if**
- **end for**
- **return** $\widetilde{v}^{\mathrm{adj}}$

end procedure

▷ If in coordinate i, $\widetilde{v}_i^{\mathrm{tmp}} \neq \widetilde{v}_i$ and $\widetilde{v}_i^{\mathrm{tmp}}$ is close to v_i, then $\widetilde{v}_i^{\mathrm{tmp}}$ should move back to v_i.

end data structure

Algorithm 7 Data structure : SoftThreshold()

data structure
▷ This procedure doesn't use any members in the memory of data structure.
procedure SoftThreshold($y, w^{\mathrm{new}}, v, \epsilon, n^a$)
  Let $\pi : [n] \to [n]$ be a sorting permutation such that $y_{\pi(i)} \geq y_{\pi(i+1)}$
  $k \leftarrow$ the number of indices i such that $y_i \geq \epsilon$
  if $k \geq n^a$ **then**
    repeat
      $k \leftarrow \min\{\lceil 1.5k \rceil, n\}$
    until $k = n$ or $y_{\pi(k)} < (1 - 1/\log n) \cdot y_{\pi(k/1.5)}$
  end if
  $v^{\mathrm{new}}_{\pi(i)} \leftarrow \begin{cases} w^{\mathrm{new}}_{\pi(i)}, & i \in \{1, 2, \cdots, k\}; \\ v_{\pi(i)}, & i \in \{k+1, \cdots, n\}. \end{cases}$
  return v^{new}, k
end procedure

end data structure

Algorithm 8 Data structure : UpdateQuery()

data structure ▷ Theorem C.9

procedure UpdateQuery(w^{new}, h^{new}) ▷ Theorem D.6
  $w^{\mathrm{appr}}, k, \widetilde{k} \leftarrow$ UpdateV(w^{new}) ▷ Algorithm 9, k and $\widetilde{k}$ are only used for analysis.
  $h^{\mathrm{appr}}, p, \widetilde{p} \leftarrow$ UpdateG(h^{new}) ▷ Algorithm 10, p and $\widetilde{p}$ are only used for analysis.
  $r \leftarrow$ Query($w^{\mathrm{appr}}, h^{\mathrm{appr}}$) ▷ Algorithm 12, Lemma D.7, Lemma E.3
  ▷ Compute $r = R[l]^\top R[l] \sqrt{W^{\mathrm{appr}}} A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$
  return $w^{\mathrm{appr}}, h^{\mathrm{appr}}, r$
end procedure

end data structure

1. $\partial\Delta \leftarrow W^{\mathrm{appr}} - \widetilde{V}$,
2. $\partial\Gamma \leftarrow \sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}$,
3. $\partial\xi \leftarrow \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \sqrt{\widetilde{V}} f(\widetilde{g})$,
4. $\partial S \leftarrow \mathrm{supp}(w^{\mathrm{appr}} - \widetilde{v})$,
5. $\Delta^{\mathrm{new}} \leftarrow \Delta + \partial\Delta$,
6. $\Gamma^{\mathrm{new}} \leftarrow \Gamma + \partial\Gamma$,
7. $\xi^{\mathrm{new}} \leftarrow \xi + \partial\xi$,
8. $S^{\mathrm{new}} \leftarrow \mathrm{supp}(w^{\mathrm{appr}} - v)$,
9. $S' \leftarrow (S \cup \partial S) \backslash S^{\mathrm{new}}$.

Remark D.3 (Compute local variables). *The private procedure* ComputeLocalVariables *(Algorithm 11) computes these local variables (defined in Definition D.2) correctly.*

Remark D.4 (Temporary variables). *All variables with super-script* "tmp" *are temporary local variables that are only used in update procedures.*

Remark D.5 (Properties of S' and S^{new}). *Note that $S^{\mathrm{new}} \subseteq S \cup \partial S$, and $S' \subseteq S \cap \partial S$.*

In this section we prove the following main theorem using lemmas proved in later sections.

Theorem D.6 (Correctness of UpdateQuery). *On the j-th call to the procedure* UpdateQuery *(Algorithm 8), the output satisfies the following:*

Algorithm 9 Data structure : UpdateV()

1: **data structure** ▷ Theorem C.9
2:
3: **procedure** UpdateV(w^{new}) ▷ Return $(w^{\mathrm{appr}}, k, \widetilde{k})$. Lemma D.13
4: $\widetilde{v}^{\mathrm{tmp}}, \widetilde{k} \leftarrow \textsc{SoftThreshold}(y_i \leftarrow \psi(w_i^{\mathrm{new}}/\widetilde{v}_i - 1), w^{\mathrm{new}}, \widetilde{v}, \epsilon_{\mathrm{mp}}/2, n^{\widetilde{a}})$ ▷ Algorithm 7
5: $\widetilde{v}^{\mathrm{new}} \leftarrow \textsc{Adjust}(\widetilde{v}^{\mathrm{tmp}}, \widetilde{v}, v, \epsilon_{\mathrm{far}})$ ▷ Algorithm 6
6: **if** $|\mathrm{supp}(\widetilde{v}^{\mathrm{new}} - v)| \geq n^a$ **then**
7: $v^{\mathrm{new}}, k \leftarrow \textsc{SoftThreshold}(y_i \leftarrow (\psi(w_i^{\mathrm{new}}/v_i - 1) + \psi(w_i^{\mathrm{new}}/\widetilde{v}_i - 1)), w^{\mathrm{new}}, v, \frac{\epsilon_{\mathrm{far}}^2}{32\epsilon_{\mathrm{mp}}}, n^a)$
8: ▷ If $|\mathrm{supp}(\widetilde{v}^{\mathrm{new}} - v)| \geq n^a$, then $k \geq n^a$. See Fact F.10
9: $\textsc{MatrixUpdate}(v^{\mathrm{new}})$ ▷ Algorithm 13, Lemma D.17, Lemma E.12, Lemma F.19
10: ▷ Update $v, \widetilde{v}$ to be v^{new}.
11: ▷ Update invariants M, Q, β_1, β_2, S, Δ, Γ, ξ, B, γ_1, γ_2, E, F.
12: **return** $(v^{\mathrm{new}}, k, 0)$
13: **else**
14: **if** $|\mathrm{supp}(\widetilde{v}^{\mathrm{new}} - \widetilde{v})| \geq n^{\widetilde{a}}$ **then**
15: ▷ If $|\mathrm{supp}(\widetilde{v}^{\mathrm{new}} - \widetilde{v})| \geq n^{\widetilde{a}}$, then $\widetilde{k} \geq n^{\widetilde{a}}$. See Fact F.11.
16: $\textsc{PartialMatrixUpdate}(\widetilde{v}^{\mathrm{new}})$ ▷ Algorithm 14, Lemma D.21, Lemma E.18, Lemma F.30
17: ▷ Update $\widetilde{v}$ to be $\widetilde{v}^{\mathrm{new}}$.
18: ▷ Update invariants S, Δ, Γ, ξ, B, γ_1, γ_2, E, F.
19: **return** $(\widetilde{v}^{\mathrm{new}}, 0, \widetilde{k})$
20: **end if**
21: **end if**
22: **return** $(\widetilde{v}^{\mathrm{new}}, 0, 0)$
23: **end procedure**
24:
25: **end data structure**

1. $w^{\mathrm{appr}} \approx_{\epsilon_{\mathrm{mp}}} w^{\mathrm{new}}$, $h^{\mathrm{appr}} \approx_{\epsilon_{\mathrm{mp}}} h^{\mathrm{new}}$,
2. $r = R[l]^\top R[l] \sqrt{W^{\mathrm{appr}}} M^{\mathrm{new}} \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$, *where* $M^{\mathrm{new}} = A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A$.

Proof. **Part 1.** The output w^{appr} is returned by UpdateV, it can be v^{new} returned from Line 12, or it can be $\widetilde{v}^{\mathrm{new}}$ returned from Line 19 and 22 (in Algorithm 9).

In the case of $w^{\mathrm{appr}} = v^{\mathrm{new}}$, the properties of v^{new} are given in Fact F.7. According to Part 2 and 3 of Fact F.7, there exists a permutation $\pi : [n] \to [n]$ and a number k such that $\forall i \in \pi([k])$, $v_i^{\mathrm{new}} = w_i^{(j+1)}$ and $\forall i \notin \pi([k])$, $v_i^{\mathrm{new}} \approx_{\epsilon_{\mathrm{mp}}} w_i^{(j+1)}$, where $w^{(j+1)}$ is defined as w^{new} in the j-th iteration. So $v^{\mathrm{new}} \approx_{\epsilon_{\mathrm{mp}}} w^{\mathrm{new}}$ in this case.

If $w^{\mathrm{appr}} = \widetilde{v}^{\mathrm{new}}$, the properties of $\widetilde{v}^{\mathrm{new}}$ are given in Fact F.6. According to Part 3 and 4 of Fact F.6, there exists a permutation $\pi : [n] \to [n]$ and a number $\widetilde{k}$ such that $\forall i \in \pi([\widetilde{k}])$, $\widetilde{v}_i^{\mathrm{new}} \approx_{\epsilon_{\mathrm{far}}} w_i^{(j+1)}$ and $\forall i \notin \pi([\widetilde{k}])$, $\widetilde{v}_i^{\mathrm{new}} \approx_{\epsilon_{\mathrm{mp}}} w_i^{(j+1)}$, where $w^{(j+1)}$ is defined as w^{new} in the j-th iteration. Using the assumption that $\epsilon_{\mathrm{far}} < \epsilon_{\mathrm{mp}}$, we get $w^{\mathrm{appr}} \approx_{\epsilon_{\mathrm{mp}}} w^{\mathrm{new}}$.

$h^{\mathrm{appr}} \approx_{\epsilon_{\mathrm{mp}}} h^{\mathrm{new}}$ follows by similar reasons.

Part 2. First we prove by induction that all invariants of Assumption D.1 hold all the time. In the beginning, the data structure calls Initialize. By Lemma D.33, all invariants hold.

In the following iterations, the data structure is only accessed via calls to its procedure UpdateQuery by OneStepCentralPath (Line 4 and 7 in Algorithm 3). UpdateQuery calls UpdateV, UpdateG and Query(Line 4, 5, 6 in Algorithm 8). The procedure Query does not modify any data structure member, so it won't violate any invarint. By Part 2 of Lemma D.13 and D.14, if all invariants are satisfied before entering the procedure UpdateV (or UpdateG),

Algorithm 10 Data structure : UpdateG()

data structure ▷ Theorem C.9

procedure UpdateG(h^{new}) ▷ Return $(h^{\mathrm{appr}}, p, \widetilde{p})$. Lemma D.14
  $\widetilde{g}^{\mathrm{tmp}}, \widetilde{p} \leftarrow \textsc{SoftThreshold}(y_i \leftarrow \psi(h_i^{\mathrm{new}}/\widetilde{g}_i - 1), h^{\mathrm{new}}, \widetilde{g}, \epsilon_{\mathrm{mp}}/2, n^{\widetilde{a}})$ ▷ Algorithm 7
  $\widetilde{g}^{\mathrm{new}} \leftarrow \textsc{Adjust}(\widetilde{g}^{\mathrm{tmp}}, \widetilde{g}, g, \epsilon_{\mathrm{far}})$ ▷ Algorithm 6
  if $|\mathrm{supp}(\widetilde{g}^{\mathrm{new}} - g)| \geq n^a$ **then**
    $g^{\mathrm{new}}, p \leftarrow \textsc{SoftThreshold}(y_i \leftarrow (\psi(h_i^{\mathrm{new}}/g_i - 1) + \psi(h_i^{\mathrm{new}}/\widetilde{g}_i - 1)), h^{\mathrm{new}}, g, \frac{\epsilon_{\mathrm{far}}^2}{32\epsilon_{\mathrm{mp}}}, n^a)$
      ▷ Similarly, if $|\mathrm{supp}(\widetilde{g}^{\mathrm{new}} - g)| > n^a$, then $p > n^a$.
    VectorUpdate(g^{new}) ▷ Algorithm 15, Lemma D.25, Lemma E.27, Lemma F.35
      ▷ Update $g, \widetilde{g}$ to be g^{new}. Update invariants β_1, β_2, ξ, γ_1, γ_2, T.
    return (g^{new}, p, 0)
  else
    if $|\mathrm{supp}(\widetilde{g}^{\mathrm{new}} - \widetilde{g})| \geq n^{\widetilde{a}}$ **then**
      ▷ Similarly, if $|\mathrm{supp}(\widetilde{g}^{\mathrm{new}} - \widetilde{g})| > n^{\widetilde{a}}$, then $\widetilde{p} > n^{\widetilde{a}}$.
      PartialVectorUpdate($\widetilde{g}^{\mathrm{new}}$) ▷ Algorithm 16, Lemma D.29, Lemma E.33, Lemma F.36
        ▷ Update $\widetilde{g}$ to be $\widetilde{g}^{\mathrm{new}}$. Update invariants ξ, γ_1, γ_2, T.
      return ($\widetilde{g}^{\mathrm{new}}$, 0, $\widetilde{p}$)
    end if
  end if
  return ($\widetilde{g}^{\mathrm{new}}$, 0, 0)
end procedure

end data structure

Algorithm 11 Data structure : ComputeLocalVariables()

data structure ▷ Theorem C.9

procedure ComputeLocalVariables(w^{appr}, h^{appr})
  $\partial\Delta \leftarrow W^{\mathrm{appr}} - \widetilde{V}$
  $\partial\Gamma \leftarrow \sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}$
  $\partial S \leftarrow \mathrm{supp}(w^{\mathrm{appr}} - \widetilde{v})$
  $\Delta^{\mathrm{new}} \leftarrow \Delta + \partial\Delta$
  $\Gamma^{\mathrm{new}} \leftarrow \Gamma + \partial\Gamma$
  $S^{\mathrm{new}} \leftarrow \mathrm{supp}(w^{\mathrm{appr}} - v)$
  $S' \leftarrow (S \cup \partial S) \backslash S^{\mathrm{new}}$
    ▷ If the input h^{appr} is null, we don't need to compute the following two local variables.
  $\partial\xi \leftarrow \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \sqrt{\widetilde{V}} f(\widetilde{g})$
  $\xi^{\mathrm{new}} \leftarrow \xi + \partial\xi$
  return $(\partial\Delta, \partial\Gamma, \partial\xi, \partial S, \Delta^{\mathrm{new}}, \Gamma^{\mathrm{new}}, \xi^{\mathrm{new}}, S^{\mathrm{new}}, S')$
end procedure

end data structure

then after executing the procedure UpdateV (or UpdateG), all invariants are still satisfied. By induction, all invariants of Assumption D.1 hold all the time.

Next we prove that before entering Query, we always have $|S \cup \partial S| \leq 2n^a$, so that all $\mathcal{L}$, $\mathcal{L}_c$, $\mathcal{L}_r$, $\mathcal{L}_*$ operators are well-defined. Note that

$$\begin{aligned} S = {} & \mathrm{supp}(\widetilde{v} - v) && \text{(by Part 5 of Assumption D.1),} \\ \partial S = {} & \mathrm{supp}(w^{\mathrm{appr}} - \widetilde{v}) && \text{(by Part 4 of Definition D.2).} \end{aligned}$$

Algorithm 12 Data structure : QUERY()

data structure ▷ Theorem C.9

procedure QUERY($w^{\mathrm{appr}}, h^{\mathrm{appr}}$) ▷ Lemma D.7, Lemma E.3

$\partial\Delta, \partial\Gamma, \partial\xi, \partial S, \Delta^{\mathrm{new}}, \_, \_, S^{\mathrm{new}}, S' \leftarrow$ COMPUTELOCALVARIABLES($w^{\mathrm{appr}}, \widetilde{g}^{\mathrm{new}}$) ▷ Algorithm 11

$r_1 \leftarrow \beta_1[l]$ ▷ $r_1 \in \mathbb{R}^{n^b}$

$r_2 \leftarrow Q[l]\xi + R[l]\gamma_2 + R[l]\partial\Gamma M(\xi + \partial\xi) + \big(Q[l] + R[l]\Gamma M\big)\partial\xi$ ▷ $r_2 \in \mathbb{R}^{n^b}$

$r_3 \leftarrow R[l](\Gamma + \partial\Gamma)\beta_2$ ▷ $r_3 \in \mathbb{R}^{n^b}$

$\partial\gamma \leftarrow B \cdot (\mathcal{L}_r[(\beta_2)_{\partial S \setminus S}] - \mathcal{L}_r[(\beta_2)_{S'}]) + B \cdot (\mathcal{L}_r[(M_{\partial S \setminus S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) \cdot (\xi + \partial\xi) + E \cdot \partial\xi$

▷ local variable $\partial\gamma \in \mathbb{R}^{6n^a}$

$(U', C, U) \leftarrow$ DECOMPOSE$\Big(\mathcal{L}_*[(\Delta^{\mathrm{new}}_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}}, S^{\mathrm{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}]\Big)$

▷ DECOMPOSE is defined in Lemma C.4. $U', U \in \mathbb{R}^{6n^a \times 3|\partial S|}$, $C \in \mathbb{R}^{3|\partial S| \times 3|\partial S|}$

$\partial E \leftarrow E_{\partial S} - B_{(\partial S \cap S)} \cdot M_{(\partial S \cap S), \partial S}$

$(\partial E)_{S'} \leftarrow -(\partial E)_{S'}, \quad (\partial E)_{(S \cap \partial S) \setminus S'} \leftarrow 0$ ▷ local variable $\partial E \in \mathbb{R}^{6n^a \times |\partial S|}$

$U^{\mathrm{tmp}} \leftarrow [B_{\partial S}, B_{\partial S}, \partial E]$

▷ local variable $U^{\mathrm{tmp}} \in \mathbb{R}^{6n^a \times 3|\partial S|}$, $U^{\mathrm{tmp}} = BU'$ (Corollary C.7)

$\gamma^{\mathrm{tmp}} \leftarrow U^{\mathrm{tmp}}(C^{-1} + U^\top U^{\mathrm{tmp}})^{-1} U^\top \cdot (\gamma_1 + \partial\gamma)$ ▷ local variable, $\gamma^{\mathrm{tmp}} \in \mathbb{R}^{6n^a}$

$r_4 \leftarrow \Big(\mathcal{L}_c[(Q[l])_{S^{\mathrm{new}}}] + F[l] + R[l]\Gamma(\mathcal{L}_c[M_{\partial S \setminus S}] - \mathcal{L}_c[M_{S'}]) + R[l]\partial\Gamma\mathcal{L}_c[M_{S^{\mathrm{new}}}]\Big)(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)$

$r \leftarrow R[l]^\top(r_1 + r_2 + r_3 + r_4)$ ▷ $r \in \mathbb{R}^n$

$l \leftarrow l + 1$

return r

end procedure

end data structure

Algorithm 13 Data structure : MATRIXUPDATE()

data structure ▷ Theorem C.9

procedure MATRIXUPDATE(w^{appr}) ▷ Lemma D.17, Lemma E.12, Lemma F.19

$\_, \_, \_, \_, \Delta^{\mathrm{new}}, \Gamma^{\mathrm{new}}, \_, S^{\mathrm{new}}, \_ \leftarrow$ COMPUTELOCALVARIABLES($w^{\mathrm{appr}}, \_$) ▷ Algorithm 11

$M^{\mathrm{tmp}} \leftarrow M - M_{S^{\mathrm{new}}} \cdot ((\Delta^{\mathrm{new}}_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} \cdot (M_{S^{\mathrm{new}}})^\top$

$Q^{\mathrm{tmp}} \leftarrow Q + R(\Gamma^{\mathrm{new}} M^{\mathrm{tmp}}) + R\sqrt{V}(M^{\mathrm{tmp}} - M)$

$\beta_1^{\mathrm{tmp}} \leftarrow Q^{\mathrm{tmp}}\sqrt{W^{\mathrm{appr}}} f(g)$

$\beta_2^{\mathrm{tmp}} \leftarrow M^{\mathrm{tmp}}\sqrt{W^{\mathrm{appr}}} f(g)$

$\xi^{\mathrm{tmp}} \leftarrow \sqrt{W^{\mathrm{appr}}}(f(\widetilde{g}) - f(g))$

▷ We start to refresh variables in the memory of data structure

$Q \leftarrow Q^{\mathrm{tmp}}$, $M \leftarrow M^{\mathrm{tmp}}$

$\beta_1 \leftarrow \beta_1^{\mathrm{tmp}}$, $\beta_2 \leftarrow \beta_2^{\mathrm{tmp}}$, $\xi \leftarrow \xi^{\mathrm{tmp}}$

$v \leftarrow \widetilde{v} \leftarrow w^{\mathrm{appr}}$

$B \leftarrow I$, $F \leftarrow 0$, $E \leftarrow 0$

$S \leftarrow \emptyset$, $\Delta \leftarrow \Gamma \leftarrow 0$, $\gamma_1 \leftarrow \gamma_2 \leftarrow 0$

end procedure

end data structure

By Part 1 of Lemma D.13, we have $\|w^{\mathrm{appr}} - \widetilde{v}\|_0 \leq n^{\widetilde{a}}$, so $|\partial S| \leq n^{\widetilde{a}}$. By Corollary D.15, we have $\|\widetilde{v} - v\|_0 \leq n^a$, so $|S| \leq n^a$. Therefore

$$|S \cup \partial S| \leq |S| + |\partial S| \leq n^a + n^{\widetilde{a}} \leq 2n^a,$$

where we use the fact that $\widetilde{a} \leq a$.

Algorithm 14 Data structure : PartialMatrixUpdate().

data structure ▷ Theorem C.9

procedure PartialMatrixUpdate(w^{appr}) ▷ Lemma D.21, Lemma E.18, Lemma F.30

  $\_, \partial\Gamma, \_, \partial S, \Delta^{\mathrm{new}}, \Gamma^{\mathrm{new}}, \_, S^{\mathrm{new}}, \_ \leftarrow$ ComputeLocalVariables($w^{\mathrm{appr}}, \_$) ▷ Algorithm 11

  $(U', C, U) \leftarrow \textsc{Decompose}\Big(\mathcal{L}_*[(\Delta^{\mathrm{new}}_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}}, S^{\mathrm{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}]\Big)$

  ▷ Decompose is defined in Lemma C.4

  $B^{\mathrm{tmp}} \leftarrow B - BU'(C^{-1} + U^\top BU')^{-1}U^\top B$

  $F^{\mathrm{tmp}} \leftarrow F + R\Gamma \cdot (\mathcal{L}_c[M_{\partial S \setminus S}] - \mathcal{L}_c[M_{S'}]) + R\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}]$

  $E^{\mathrm{tmp}} \leftarrow E + B^{\mathrm{tmp}}(\mathcal{L}_r[(M_{\partial S \setminus S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) - BU'(C^{-1} + U^\top BU')^{-1}U^\top E$

  $\xi^{\mathrm{tmp}} \leftarrow \sqrt{W^{\mathrm{appr}}} f(\widetilde{g}) - \sqrt{V} f(g)$

  $\gamma_1^{\mathrm{tmp}} \leftarrow B^{\mathrm{tmp}} \cdot \mathcal{L}_r[\beta_{2,S^{\mathrm{new}}}] + B^{\mathrm{tmp}} \cdot \mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top]\xi^{\mathrm{tmp}}$

  $\gamma_2^{\mathrm{tmp}} \leftarrow \gamma_2 + (\Gamma + \partial\Gamma)M(\sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}})f(\widetilde{g}) + \partial\Gamma M(\sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g))$

  ▷ We start to refresh variables in the memory of data structure

  $B \leftarrow B^{\mathrm{tmp}}$, $F \leftarrow F^{\mathrm{tmp}}$, $E \leftarrow E^{\mathrm{tmp}}$

  $\xi \leftarrow \xi^{\mathrm{tmp}}$ $\gamma_1 \leftarrow \gamma_1^{\mathrm{tmp}}$, $\gamma_2 \leftarrow \gamma_2^{\mathrm{tmp}}$

  $\widetilde{v} \leftarrow w^{\mathrm{appr}}$, $S \leftarrow S^{\mathrm{new}}$, $\Delta \leftarrow \Delta^{\mathrm{new}}$, $\Gamma \leftarrow \Gamma^{\mathrm{new}}$

end procedure

end data structure

Algorithm 15 Data structure : VectorUpdate().

data structure ▷ Theorem C.9

procedure VectorUpdate(h^{appr}) ▷ Lemma D.25, Lemma E.27, Lemma F.35

  $\beta_1^{\mathrm{tmp}} \leftarrow \beta_1 + Q\sqrt{V}(f(h^{\mathrm{appr}}) - f(g))$

  $\beta_2^{\mathrm{tmp}} \leftarrow \beta_2 + M\sqrt{V}(f(h^{\mathrm{appr}}) - f(g))$

  $\xi^{\mathrm{tmp}} \leftarrow (\sqrt{\widetilde{V}} - \sqrt{V}) f(h^{\mathrm{appr}})$

  $\gamma_1^{\mathrm{tmp}} \leftarrow B \cdot \mathcal{L}_r[(\beta_2^{\mathrm{tmp}})_S] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi^{\mathrm{tmp}}$

  $\gamma_2^{\mathrm{tmp}} \leftarrow \Gamma M \cdot \xi^{\mathrm{tmp}}$

  ▷ We start to refresh variables in the memory of data structure

  $\beta_1 \leftarrow \beta_1^{\mathrm{tmp}}$, $\beta_2 \leftarrow \beta_2^{\mathrm{tmp}}$, $\xi \leftarrow \xi^{\mathrm{tmp}}$, $\gamma_1 \leftarrow \gamma_1^{\mathrm{tmp}}$, $\gamma_2 \leftarrow \gamma_2^{\mathrm{tmp}}$

  $g \leftarrow \widetilde{g} \leftarrow h^{\mathrm{appr}}$,

  $T \leftarrow \emptyset$

end procedure

end data structure

Now the two conditions of Lemma D.7 are both satisfied, so we have

$$r = R[l]^\top R[l] \sqrt{W^{\mathrm{appr}}} M^{\mathrm{new}} \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}),$$

where $M^{\mathrm{new}} = A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A$. □

D.1 Correctness of Query

In this section we follow the notation of the procedure Query (Algorithm 12). Note that v, $\widetilde{v}$, g, $\widetilde{g}$, M, Q, R, β_1, β_2, γ_1, γ_2, B, E, F, Δ, Γ, S are all members of the data structure (See Algorithm 4). Query (Algorithm 12) takes w^{appr} and h^{appr} as input, and uses the inputs and members of the data structure to compute the following local variables: $\partial\Delta$, $\partial\Gamma$, $\partial\xi$, ∂S, $\partial\gamma$, Δ^{new}, S^{new}, S', U',

Algorithm 16 Data structure : PartialVectorUpdate().

data structure ▷ Theorem C.9

procedure PartialVectorUpdate(h^{appr}) ▷ Lemma D.29, Lemma E.33, Lemma F.36

  $\xi^{\mathrm{tmp}} \leftarrow \sqrt{\widetilde{V}} f(h^{\mathrm{appr}}) - \sqrt{V} f(g)$

  $\gamma_1^{\mathrm{tmp}} \leftarrow \gamma_1 + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \sqrt{\widetilde{V}} \big(f(h^{\mathrm{appr}}) - f(\widetilde{g})\big)$

  $\gamma_2^{\mathrm{tmp}} \leftarrow \gamma_2 + \Gamma M \sqrt{\widetilde{V}} \big(f(h^{\mathrm{appr}}) - f(\widetilde{g})\big)$

  ▷ We start to refresh variables in the memory of data structure

  $\xi \leftarrow \xi^{\mathrm{tmp}}$, $\gamma_1 \leftarrow \gamma_1^{\mathrm{tmp}}$, $\gamma_2 \leftarrow \gamma_2^{\mathrm{tmp}}$

  $T \leftarrow \mathrm{supp}(h^{\mathrm{appr}} - g)$

  $\widetilde{g} \leftarrow \widetilde{g}^{\mathrm{new}}$

end procedure

end data structure

Procedure	Lemma	Section
UpdateQuery	Theorem D.6	–
Query	Lemma D.7	Section D.1
UpdateV	Lemma D.13	Section D.2
UpdateG	Lemma D.14	Section D.2
MatrixUpdate	Lemma D.17	Section D.3
PartialMatrixUpdate	Lemma D.21	Section D.4
VectorUpdate	Lemma D.25	Section D.5
PartialVectorUpdate	Lemma D.29	Section D.6
Initialize	Lemma D.33	Section D.7

Table 10: Summary of the section that proves the correctness of the data structure.

C, U, ∂E, U^{tmp}, γ^{tmp}, r_1, r_2, r_3, r_4. Finally, Query (Algorithm 12) outputs r. The goal of this section is to prove Lemma D.7 which gives a close-form formula of the output r.

Lemma D.7 (Correctness of Query)**.** *Before entering* Query *(Algorithm 12), if we have the following two guarantees: $|S \cup \partial S| \leq 2n^a$, and all the invariants of Assumption D.1 are satisfied, then the output r is*

$$r = R[l]^\top R[l] \sqrt{W^{\mathrm{appr}}} M^{\mathrm{new}} \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}),$$

where $M^{\mathrm{new}} = A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A$.

This lemma is proved in Claim D.12 using the following:

1. $r_1 = Q[l] \sqrt{V} f(g)$ (Claim D.8)
2. $r_2 = (Q[l] + R[l](\Gamma + \partial\Gamma) M) \cdot (\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \sqrt{V} f(g))$ (Claim D.9)
3. $r_3 = R[l](\Gamma + \partial\Gamma) M \sqrt{V} f(g)$ (Claim D.10)
4. $r_4 = -R[l] \sqrt{W^{\mathrm{appr}}} M_{S^{\mathrm{new}}} ((\Delta^{\mathrm{new}}_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} (M_{S^{\mathrm{new}}})^\top \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$ (Claim D.11)

Now we prove these claims one by one. In the following we assume that $|S \cup \partial S| \leq 2n^a$ and all the invariants of Assumption D.1 are satisfied. Note that when $|S \cup \partial S| \leq 2n^a$, all of the $\mathcal{L}$, $\mathcal{L}_c$, $\mathcal{L}_r$, and $\mathcal{L}_*$ are well-defined, and the Decompose function is also well-defined.

Claim D.8 (Close-form formula for r_1)**.** *We have $r_1 = Q[l] \sqrt{V} f(g)$.*

Proof. From the assignment of r_1 (Line 5 of Algorithm 12), we have

$$r_1 = \beta_1[l] = Q[l]\sqrt{V}f(g).$$

where the last step follows from $\beta_1 = Q\sqrt{V}f(g)$ (Part 2 of Assumption D.1). □

Claim D.9 (Close-form formula for r_2)**.** *We have*

$$r_2 = (Q[l] + R[l](\Gamma + \partial\Gamma)M) \cdot (\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}}) - \sqrt{V}f(g)).$$

Proof. From the assignment of r_2 (Line 6 of Algorithm 12), we have

$$\begin{aligned}
r_2 &= Q[l]\xi + R[l]\gamma_2 + R[l]\partial\Gamma M(\xi + \partial\xi) + (Q[l] + R[l]\Gamma M)\partial\xi \\
&= (Q[l] + R[l]\Gamma M)\xi + R[l]\partial\Gamma M(\xi + \partial\xi) + (Q[l] + R[l]\Gamma M)\partial\xi \\
&= \big(Q[l] + R[l](\Gamma + \partial\Gamma)M\big) \cdot (\xi + \partial\xi) \\
&= \big(Q[l] + R[l](\Gamma + \partial\Gamma)M\big) \cdot (\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}}) - \sqrt{V}f(g)),
\end{aligned}$$

where the second step follows from $\gamma_2 = \Gamma M \cdot \xi$ (Part 14 of Assumption D.1), the third step follows from merging terms, and the fourth step follows from the invariant $\xi = \sqrt{\widetilde{V}}f(\widetilde{g}) - \sqrt{V}f(g)$ (Part 9 of Assumption D.1) and the definition $\partial\xi = \sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}}) - \sqrt{\widetilde{V}}f(\widetilde{g})$ (Part 3 of Definition D.2). □

Claim D.10 (Close-form formula for r_3)**.** *We have* $r_3 = R[l](\Gamma + \partial\Gamma)M\sqrt{V}f(g)$.

Proof. From the assignment of r_3 (Line 7 of Algorithm 12), we have

$$r_3 = R[l](\Gamma + \partial\Gamma)\beta_2 = R[l](\Gamma + \partial\Gamma)M\sqrt{V}f(g),$$

where the second step follows from the invariant $\beta_2 = M\sqrt{V}f(g)$ (Part 4 of Assumption D.1). □

Claim D.11 (Close-form formula for r_4)**.** *We have*

$$r_4 = -R[l]\sqrt{W^{\mathrm{appr}}}M_{S^{\mathrm{new}}} \cdot ((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} \cdot (M_{S^{\mathrm{new}}})^\top \sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}}).$$

Proof. First note that the left part of r_4 is

$$\begin{aligned}
&\mathcal{L}_c[(Q[l])_{S^{\mathrm{new}}}] + F[l] + R[l]\Gamma \cdot (\mathcal{L}_c[M_{\partial S\setminus S}] - \mathcal{L}_c[M_{S'}]) + R[l]\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}] \\
&= \mathcal{L}_c[(Q[l])_{S^{\mathrm{new}}}] + R[l]\Gamma \cdot \mathcal{L}_c[M_S] + R[l]\Gamma \cdot (\mathcal{L}_c[M_{\partial S\setminus S}] - \mathcal{L}_c[M_{S'}]) + R[l]\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}] \\
&= \mathcal{L}_c[(Q[l])_{S^{\mathrm{new}}}] + R[l]\Gamma \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}] + R[l]\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}] \\
&= \mathcal{L}_c[(Q[l])_{S^{\mathrm{new}}}] + R[l](\Gamma + \partial\Gamma) \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}] \\
&= R[l]\sqrt{V}\mathcal{L}_c[M_{S^{\mathrm{new}}}] + R[l](\Gamma + \partial\Gamma) \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}] \\
&= R[l]\sqrt{W^{\mathrm{appr}}}\mathcal{L}_c[M_{S^{\mathrm{new}}}], \qquad (37)
\end{aligned}$$

where the first step follows from $F = R\Gamma \cdot \mathcal{L}_c[M_S]$ (Part 12 of Assumption D.1), the second step follows from $S' = (S \cup \partial S)\setminus S^{\mathrm{new}}$ (Part 9 of Definition D.2) and thus $\mathcal{L}_c[M_{S'}] + \mathcal{L}_c[M_{S^{\mathrm{new}}}] = \mathcal{L}_c[M_{S\cup\partial S}] = \mathcal{L}_c[M_S] + \mathcal{L}_c[M_{\partial S\setminus S}]$ by Part 2 of Remark C.3, the fourth step follows from $Q = R\sqrt{V}M$ (Part 2 of Assumption D.1) and Part 3 of Remark C.3, and the fifth step follows from $\Gamma = \sqrt{\widetilde{V}} - \sqrt{V}$ (Part 8 of Assumption D.1) and $\partial\Gamma = \sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}$ (Part 2 of Assumption D.2).

We also have

$$\gamma_1 + \partial\gamma = B \cdot \mathcal{L}_r[(\beta_2)_S] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi +$$

$$
\begin{aligned}
& B \cdot (\mathcal{L}_r[(\beta_2)_{\partial S \backslash S}] - \mathcal{L}_r[(\beta_2)_{S'}]) + B \cdot (\mathcal{L}_r[(M_{\partial S \backslash S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) \cdot (\xi + \partial\xi) + E \cdot \partial\xi \\
= & \underbrace{B \cdot \mathcal{L}_r[(\beta_2)_S] + B \cdot (\mathcal{L}_r[(\beta_2)_{\partial S \backslash S}] - \mathcal{L}_r[(\beta_2)_{S'}])}_{a_1} + \\
& \underbrace{B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi + B \cdot (\mathcal{L}_r[(M_{\partial S \backslash S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) \cdot (\xi + \partial\xi) + E \cdot \partial\xi}_{a_2}, \qquad (38)
\end{aligned}
$$

where the first step follows from $\gamma_1 = B \cdot \mathcal{L}_r[(\beta_2)_S] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi$ (Part 13 of Assumption D.1) and the assignment of $\partial\gamma$ on Line 8 of Algorithm 12, the second step follows from changing the order of terms. And

$$
a_1 = B \cdot \mathcal{L}_r[(\beta_2)_{S^{\text{new}}}], \qquad (39)
$$

which follows from $S' = (S \cup \partial S) \backslash S^{\text{new}}$ (Part 9 of Definition D.2) and thus $\mathcal{L}_c[(\beta_2)_{S'}] + \mathcal{L}_c[(\beta_2)_{S^{\text{new}}}] = \mathcal{L}_c[(\beta_2)_{S \cup \partial S}] = \mathcal{L}_c[(\beta_2)_S] + \mathcal{L}_c[(\beta_2)_{\partial S \backslash S}]$ by Part 2 of Remark C.3. And also

$$
\begin{aligned}
a_2 &= B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi + B \cdot (\mathcal{L}_r[(M_{\partial S \backslash S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) \cdot (\xi + \partial\xi) + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \partial\xi \\
&= B \cdot \Big(\mathcal{L}_r[(M_S)^\top] + \mathcal{L}_r[(M_{\partial S \backslash S})^\top] - \mathcal{L}_r[(M_{S'})^\top] \Big) \cdot (\xi + \partial\xi) \\
&= B \cdot \mathcal{L}_r[(M_{S^{\text{new}}})^\top] \cdot (\xi + \partial\xi), \qquad (40)
\end{aligned}
$$

where the first step follows from $E = B \cdot \mathcal{L}_r[(M_S)^\top]$, the second step follows from merging terms, and the third step follows from $S' = (S \cup \partial S) \backslash S^{\text{new}}$ (Part 9 of Definition D.2) and thus $\mathcal{L}_r[(M_{S'})^\top] + \mathcal{L}_r[(M_{S^{\text{new}}})^\top] = \mathcal{L}_r[(M_{S \cup \partial S})^\top] = \mathcal{L}_r[(M_S)^\top] + \mathcal{L}_r[(M_{\partial S \backslash S})^\top]$ by Part 2 of Remark C.3.

Combining Eq. (38), (39), and (40) together, we have

$$
\begin{aligned}
\gamma_1 + \partial\gamma &= B \cdot \mathcal{L}_r[(\beta_2)_{S^{\text{new}}}] + B \cdot \mathcal{L}_r[(M_{S^{\text{new}}})^\top] \cdot (\xi + \partial\xi) \\
&= B \cdot \mathcal{L}_r[(M_{S^{\text{new}}})^\top] \sqrt{V} f(g) + B \cdot L_r[(M_{S^{\text{new}}})^\top] \cdot (\sqrt{W^{\text{appr}}} f(h^{\text{appr}}) - \sqrt{V} f(g)) \\
&= B \cdot L_r[(M_{S^{\text{new}}})^\top] \cdot \sqrt{W^{\text{appr}}} f(h^{\text{appr}}), \qquad (41)
\end{aligned}
$$

where the second step follows from $\beta_2 = M\sqrt{V} f(g)$ (Part 4 of Assumption D.1) and using Part 3 of Remark C.3, and $\xi + \partial\xi = \sqrt{W^{\text{appr}}} f(h^{\text{appr}}) - \sqrt{V} f(g)$ (Part 9 of Assumption D.1 and Part 3 of Definition D.2), and the third step follows from merging terms.

Therefore,

$$
\begin{aligned}
\gamma^{\text{tmp}} - \gamma_1 - \partial\gamma = & - (I - U^{\text{tmp}} (C^{-1} + U^\top U^{\text{tmp}})^{-1} U^\top) \cdot (\gamma_1 + \partial\gamma) \\
= & - (I - BU' (C^{-1} + U^\top BU')^{-1} U^\top) \cdot (\gamma_1 + \partial\gamma) \\
= & - (I - BU' (C^{-1} + U^\top BU')^{-1} U^\top) \cdot B \cdot L_r[(M_{S^{\text{new}}})^\top] \cdot \sqrt{W^{\text{appr}}} f(h^{\text{appr}}) \\
= & - \mathcal{L}_*[((\Delta^{\text{new}}_{S^{\text{new}}, S^{\text{new}}}) + M_{S^{\text{new}}, S^{\text{new}}})^{-1}] \cdot L_r[(M_{S^{\text{new}}})^\top] \cdot \sqrt{W^{\text{appr}}} f(h^{\text{appr}}), \qquad (42)
\end{aligned}
$$

where the first step is by assignment of γ^{tmp} on Line 16 of Algorithm 12, the second step is by $U^{\text{tmp}} = BU'$(Corollary C.7), the third step follows by Eq. (41), the fourth step is by $B - BU(C^{-1} + U^\top BU)^{-1} U^\top B = \mathcal{L}_*[((\Delta^{\text{new}}_{S^{\text{new}}, S^{\text{new}}}) + M_{S^{\text{new}}, S^{\text{new}}})^{-1}]$ (Lemma C.6),

Then from the assignment of r_4 on Line 17 of Algorithm 12, we have

$$
\begin{aligned}
r_4 &= \Big(\mathcal{L}_c[(Q[l])_{S^{\text{new}}}] + F[l] + R[l]\Gamma \cdot (\mathcal{L}_c[M_{\partial S \backslash S}] - \mathcal{L}_c[M_{S'}]) + R[l]\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\text{new}}}] \Big) \cdot (\gamma^{\text{tmp}} - \gamma_1 - \partial\gamma) \\
&= R[l] \sqrt{W^{\text{appr}}} \mathcal{L}_c[M_{S^{\text{new}}}] \cdot (\gamma^{\text{tmp}} - \gamma_1 - \partial\gamma)
\end{aligned}
$$

$$= -R[l]\sqrt{W^{\mathrm{appr}}}\mathcal{L}_c[M_{S^{\mathrm{new}}}]\cdot\mathcal{L}_*[((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})+M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}]\cdot L_r[(M_{S^{\mathrm{new}}})^\top]\cdot\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}})$$
$$= -R[l]\sqrt{W^{\mathrm{appr}}}M_{S^{\mathrm{new}}}\cdot((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})+M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}\cdot(M_{S^{\mathrm{new}}})^\top\cdot\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}}), \quad (43)$$

where the second step follows from Eq. (37), the third step follows from Eq.(42), the fourth step follows from the property of $\mathcal{L}$ operators (Part 4 of Remark C.3). □

Claim D.12 (Close-form formula for r)**.**

$$r = R[l]^\top R[l]\sqrt{W^{\mathrm{appr}}}M^{\mathrm{new}}\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}}),$$

where $M^{\mathrm{new}} = A^\top(AW^{\mathrm{appr}}A^\top)^{-1}A$.

Proof. From Claim D.8, we have $r_1 = Q[l]\sqrt{V}f(g)$.

From Claim D.9, we have $r_2 = (Q[l]+R[l](\Gamma+\partial\Gamma)M)\cdot(\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}})-\sqrt{V}f(g))$.

From Claim D.10, we have $r_3 = R[l](\Gamma+\partial\Gamma)M\sqrt{V}f(g)$.

From Claim D.11, we have

$$r_4 = -R[l]\sqrt{W^{\mathrm{appr}}}M_{S^{\mathrm{new}}}\cdot((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}+M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}\cdot(M_{S^{\mathrm{new}}})^\top\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}}).$$

The proof sketch is as follows: we first compute r_1+r_3, then compute $(r_1+r_3)+r_2$, and finally we compute $(r_1+r_2+r_3)+r_4$. First we compute r_1+r_3 as follows:

$$r_1+r_3 = Q[l]\sqrt{V}f(g)+R[l](\Gamma+\partial\Gamma)M\sqrt{V}f(g) = R[l]\sqrt{V}M\sqrt{V}f(g)+R[l](\Gamma+\partial\Gamma)M\sqrt{V}f(g)$$
$$= R[l](\sqrt{V}+(\Gamma+\partial\Gamma))M\sqrt{V}f(g) = R[l]\sqrt{W^{\mathrm{appr}}}M\sqrt{V}f(g), \quad (44)$$

where the first step follows from Claim D.8 and D.10, the second step follows from $Q = R\sqrt{V}M$ (Part 2 of Assumption D.1), the third step follows from merging terms, and the fourth step follows from $\Gamma+\partial\Gamma = (\sqrt{\widetilde{V}}-\sqrt{V})+(\sqrt{W^{\mathrm{appr}}}-\sqrt{\widetilde{V}}) = \sqrt{W^{\mathrm{appr}}}-\sqrt{V}$, ($\Gamma$ from Part 8 in Assumption D.1, $\partial\Gamma$ from Part 2 in Definition D.2).

Secondly, we can compute $(r_1+r_3)+r_2$,

$$\begin{aligned}(r_1+r_3)+r_2 &= R[l]\sqrt{W^{\mathrm{appr}}}M\sqrt{V}f(g)+(Q[l]+R[l](\Gamma+\partial\Gamma)M)(\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}})-\sqrt{V}f(g))\\ &= R[l]\sqrt{W^{\mathrm{appr}}}M\sqrt{V}f(g)+(R[l]\sqrt{V}M+R[l](\Gamma+\partial\Gamma)M)(\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}})-\sqrt{V}f(g))\\ &= R[l]\sqrt{W^{\mathrm{appr}}}M\sqrt{V}f(g)+R[l]\sqrt{W^{\mathrm{appr}}}M(\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}})-\sqrt{V}f(g))\\ &= R[l]\sqrt{W^{\mathrm{appr}}}M\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}}), \quad (45)\end{aligned}$$

where the first step follows from Eq. (44) and Claim D.9, the second step follows from $Q = R\sqrt{V}M$ (Part 2 of Assumption D.1), the third step follows from $\Gamma+\partial\Gamma = (\sqrt{\widetilde{V}}-\sqrt{V})+(\sqrt{W^{\mathrm{appr}}}-\sqrt{\widetilde{V}}) = \sqrt{W^{\mathrm{appr}}}-\sqrt{V}$ (Γ from Part 8 in Assumption D.1, $\partial\Gamma$ from Part 2 in Definition D.2), and the fourth step follows from merging terms.

Finally, we can compute $r_1+r_2+r_3+r_4$.

$$\begin{aligned}(r_1+r_2+r_3)+r_4 &= R[l]\sqrt{W^{\mathrm{appr}}}M\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}})-\\ &\quad R[l]\sqrt{W^{\mathrm{appr}}}M_{S^{\mathrm{new}}}\left((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}+M_{S^{\mathrm{new}},S^{\mathrm{new}}}\right)^{-1}(M_{S^{\mathrm{new}}})^\top\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}})\\ &= R[l]\sqrt{W^{\mathrm{appr}}}\left(M-M_{S^{\mathrm{new}}}\left((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}+M_{S^{\mathrm{new}},S^{\mathrm{new}}}\right)^{-1}(M_{S^{\mathrm{new}}})^\top\right)\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}})\\ &= R[l]\sqrt{W^{\mathrm{appr}}}M^{\mathrm{new}}\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}}), \quad (46)\end{aligned}$$

where the first step follows from Eq. (45) and Claim D.11, the second step follows from merging terms, and the third step follows from Lemma C.8 (by setting the parameters in the lemma statement as $\Delta\leftarrow\Delta^{\mathrm{new}}$, $S\leftarrow S^{\mathrm{new}}$, $\widetilde{v}\leftarrow w^{\mathrm{appr}}$) and the definition that $M^{\mathrm{new}} = A^\top(AW^{\mathrm{appr}}A^\top)^{-1}A$.

Therefore, from the assignment of r on Line 18 of Algorithm 12, we have

$$r = R[l]^\top(r_1+r_2+r_3+r_4) = R[l]^\top R[l]\sqrt{W^{\mathrm{appr}}}M^{\mathrm{new}}\sqrt{W^{\mathrm{appr}}}f(h^{\mathrm{appr}}).$$

□

D.2 Correctness of UpdateV and UpdateG

Lemma D.13 (Correctness of UpdateV)**.** *After executing the procedure* UpdateV*, the following properties are satisfied:*

1. $\|w^{\text{appr}} - v\|_0 \le n^a$, $\|w^{\text{appr}} - \widetilde{v}\|_0 \le n^{\widetilde{a}}$.
2. *If all invariants of Assumption D.1 are satisfied before entering the procedure* UpdateV *(Algorithm 9), then all invariants are still satisfied after* UpdateV*.*

Proof. **Part 1.** The procedure UpdateV could exit in three places: Line 22, 19 and 12. We discuss them case by case.

(a. Line 22**).** w^{appr} is assigned to be $\widetilde{v}^{\text{new}}$. The algorithm avoids the if branches on Line 6 and the if branch on Line 14. Therefore, both of the conditions of the if branches are false, so we have $\|\widetilde{v}^{\text{new}} - v\|_0 < n^a$ and $\|\widetilde{v}^{\text{new}} - \widetilde{v}\|_0 < n^{\widetilde{a}}$.

(b. Line 19**).** w^{appr} is assigned to be $\widetilde{v}^{\text{new}}$. The algorithm avoids the if branch on Line 6, so we have $\|\widetilde{v}^{\text{new}} - v\|_0 < n^a$. Then the algorithm enters procedure PartialMatrixUpdate (see Line 16) to update $\widetilde{v} \leftarrow \widetilde{v}^{\text{new}}$ (see Line 16 of Algorithm 14), so $\|\widetilde{v}^{\text{new}} - \widetilde{v}\|_0 = 0 < n^{\widetilde{a}}$.

(c. Line 12**).** w^{appr} is assigned to be v^{new}. The algorithm enters the procedure MatrixUpdate (see Line 9) to update $v \leftarrow \widetilde{v} \leftarrow w^{\text{appr}}$ (see Line 13 of Algorithm 13). Thus $\|v^{\text{new}} - v\|_0 = 0 < n^a$ and $\|v^{\text{new}} - \widetilde{v}\|_0 = 0 < n^{\widetilde{a}}$.

Part 2. We first prove that $|S \cup \partial S| \le 2n^a$ is satisfied if we enter PartialMatrixUpdate. Note that the input w^{appr} of PartialMatrixUpdate is $\widetilde{v}^{\text{new}}$ (Line 16), so $\partial S = \text{supp}(w^{\text{appr}} - \widetilde{v}) = \text{supp}(\widetilde{v}^{\text{new}} - \widetilde{v})$ (Part 4 of Definition D.2). We have

$$\begin{aligned} |S \cup \partial S| &= |S| + |\partial S \backslash S| \le n^a + |\partial S \backslash S| = n^a + |\{i \in [n] : v_i = \widetilde{v}_i, \widetilde{v}_i^{\text{new}} \ne \widetilde{v}_i\}| \\ &= n^a + |\{i \in [n] : \widetilde{v}_i^{\text{new}} \ne v_i\}| \ \le \ 2n^a, \end{aligned}$$

where the second step follows from $|S| \le n^a$ which is a direct implication of Part 1 of this lemma (see proof of Corollary D.15), the third step follows from $S = \text{supp}(v - \widetilde{v})$ (Part 5 of Assumption D.1) and $\partial S = \text{supp}(\widetilde{v}^{\text{new}} - \widetilde{v})$, and the fifth step follows from $|\text{supp}(\widetilde{v}^{\text{new}} - v)| < n^a$ since the if-clause of Line 6 of UpdateV (Algorithm 9) has to be false to enter PartialMatrixUpdate.

In procedure UpdateV, the data structure members are modified only by procedure MatrixUpdate (on Line 9) and procedure PartialMatrixUpdate (on Line 16). Since all invariants are satisfied before entering MatrixUpdate or PartialMatrixUpdate, and $|S \cup \partial S| \le 2n^a$ is also satisfied before entering PartialMatrixUpdate, from Lemma D.17 and Lemma D.21 we know that all the invariants are still satisfied after MatrixUpdate and PartialMatrixUpdate. □

Lemma D.14 (Correctness of UpdateG)**.** *After executing the procedure* UpdateG*, the following properties are satisfied:*

1. $\|h^{\text{appr}} - g\|_0 \le n^a$, $\|h^{\text{appr}} - \widetilde{g}\|_0 \le n^{\widetilde{a}}$.
2. *If all invariants of Assumption D.1 are satisfied before entering the procedure* UpdateG *(Algorithm 10), then all invariants are still satisfied after* UpdateG*.*

Proof. **Part 1.** The proof is analogous to that of Part 1 of Lemma D.14.

Part 2. In procedure UpdateG, the data structure members are modified only by procedure VectorUpdate (see Line 9) and procedure PartialVectorUpdate (see Line 15). So this directly follows from the fact that all the invariants are satisfied after VectorUpdate (Lemma D.25) and PartialVectorUpdate (Lemma D.29). □

By the same reasoning, we immediately have the following corollary:

Corollary D.15 (Sparsity of $\widetilde{v} - v$ and $\widetilde{g} - g$)**.** *Throughout the algorithm the following is always satisfied:*

$$\|\widetilde{v} - v\|_0 \leq n^a, \ \|\widetilde{g} - g\|_0 \leq n^a.$$

Corollary D.16 (Sparsity guarantees when entering the procedures)**.** *Let k and $\widetilde{k}$ be the output returned by* UpdateV *(Line 4 in Algorithm 8). Let p, $\widetilde{p}$ be the output returned by* UpdateG *(Line 5 in Algorithm 8).*

1. *When entering the procedure* MatrixUpdate *(Algorithm 13), we have*

$$\|w^{\mathrm{appr}} - v\|_0 = k.$$

2. *When entering the procedure* PartialMatrixUpdate *(Algorithm 14), we have*

$$\|w^{\mathrm{appr}} - v\|_0 \leq n^a, \ \|w^{\mathrm{appr}} - \widetilde{v}\|_0 = \widetilde{k} \leq 2n^a.$$

3. *When entering the procedure* VectorUpdate *(Algorithm 15), we have*

$$\|w^{\mathrm{appr}} - v\|_0 \leq n^a, \ \|w^{\mathrm{appr}} - \widetilde{v}\|_0 \leq n^{\widetilde{a}}, \ \|h^{\mathrm{appr}} - g\|_0 = p.$$

4. *When entering the procedure* PartialVectorUpdate *(Algorithm 16), we have*

$$\|w^{\mathrm{appr}} - v\|_0 \leq n^a, \ \|w^{\mathrm{appr}} - \widetilde{v}\|_0 \leq n^{\widetilde{a}}, \ \|h^{\mathrm{appr}} - g\|_0 \leq n^a, \ \|h^{\mathrm{appr}} - \widetilde{g}\|_0 = \widetilde{p} \leq 2n^a.$$

5. *When entering the procedure* Query *(Algorithm 12), we have*

$$\|w^{\mathrm{appr}} - v\|_0 \leq n^a, \ \|w^{\mathrm{appr}} - \widetilde{v}\|_0 \leq n^{\widetilde{a}}, \ \|h^{\mathrm{appr}} - g\|_0 \leq n^a, \ \|h^{\mathrm{appr}} - \widetilde{g}\|_0 \leq n^{\widetilde{a}}.$$

Proof. All line number mentioned in the proof is in UpdateV (Algorithm 9).
Part 1. MatrixUpdate is entered in Line 9, and its input w^{appr} is v^{new}, which is defined on Line 7. So $\|w^{\mathrm{appr}} - v\|_0 = \|v^{\mathrm{new}} - v\|_0 = k$ directly follows from the definition of k.
Part 2. PartialMatrixUpdate is entered in Line 16, and its input w^{appr} is $\widetilde{v}^{\mathrm{new}}$, which is defined on Line 4. $\|w^{\mathrm{appr}} - v\|_0 \leq n^a$ is because the algorithm bypass the if-branch in Line 6. And $\|w^{\mathrm{appr}} - \widetilde{v}\|_0 = \|\widetilde{v}^{\mathrm{new}} - v\|_0 = \widetilde{k}$ directly follows from the definition of $\widetilde{k}$. And $\|w^{\mathrm{appr}} - \widetilde{v}\|_0 \leq \|w^{\mathrm{appr}} - v\|_0 + \|v - \widetilde{v}\|_0 \leq 2n^a$ follows from triangle inequality and Corollary D.15.
Part 3,4. The guarantee that $\|w^{\mathrm{appr}} - v\|_0 \leq n^a, \ \|w^{\mathrm{appr}} - \widetilde{v}\|_0 \leq n^{\widetilde{a}}$ is from Part 1 of Lemma D.13. The remaining proof is the same as Part 1 and Part 2.
Part 5. This directly follows from Part 1 of Lemma D.13 and Part 1 of Lemma D.14. □

D.3 Correctness of MatrixUpdate

Lemma D.17 (Correctness of MatrixUpdate)**.** *If all invariants of Assumption D.1 are satisfied before entering the procedure* MatrixUpdate *(Algorithm 13), then after the procedure* MatrixUpdate *we have the following guarantees:*

1. $v = \widetilde{v} = w^{\mathrm{appr}}$.

2. *g and $\widetilde{g}$ both remain the same.*

3. *All invariants of Assumption D.1 are still satisfied.*

Part 1 and 2 are proved in Claim D.18, and Part 3 is proved in Claim D.20 by using Claim D.19.

Claim D.18 (Part 1 and 2 of Lemma D.17)**.** *After the procedure* MatrixUpdate *(Algorithm 13), we have $v = \widetilde{v} = w^{\mathrm{appr}}$, and g and $\widetilde{g}$ remain the same.*

Proof. This follows directly from the value assignment of v and $\widetilde{v}$ on Line 13 of Algorithm 13, and the fact that g and $\widetilde{g}$ are not modified by Algorithm 13. □

Claim D.19. *In the procedure* MatrixUpdate *(Algorithm 13), before we refresh variables in the memory of data structure, we have the following:*

1. $M^{\mathrm{tmp}} = A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A$,
2. $Q^{\mathrm{tmp}} = R\sqrt{W^{\mathrm{appr}}} M^{\mathrm{tmp}}$,
3. $\beta_1^{\mathrm{tmp}} = Q^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}} f(g)$,
4. $\beta_2^{\mathrm{tmp}} = M^{\mathrm{new}} \sqrt{W^{\mathrm{appr}}} f(g)$,
5. $\xi^{\mathrm{tmp}} = \sqrt{W^{\mathrm{appr}}} f(\widetilde{g}) - \sqrt{W^{\mathrm{appr}}} f(g)$.

Proof. **Part 1.** On Line 5 of Algorithm 13 we assigned M^{tmp} as

$$M^{\mathrm{tmp}} = M - M_{S^{\mathrm{new}}} \cdot ((\Delta^{\mathrm{new}}_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} \cdot (M_{S^{\mathrm{new}}})^\top.$$

Since $M = A^\top (A V A^\top)^{-1} A$ (Part 1 of Assumption D.1), $\Delta^{\mathrm{new}} = W^{\mathrm{appr}} - V$ (Part 7 of Assumption D.1), and $S = \mathrm{supp}(w^{\mathrm{appr}} - v)$ (Part 5 of Assumption D.1), from Lemma C.8 we have

$$M^{\mathrm{tmp}} = A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A.$$

Part 2. We have

$$\begin{aligned} Q^{\mathrm{tmp}} &= Q + R(\Gamma^{\mathrm{new}} M^{\mathrm{tmp}}) + R\sqrt{V}(M^{\mathrm{tmp}} - M) = R\sqrt{V} M + R(\Gamma^{\mathrm{new}} M^{\mathrm{tmp}}) + R\sqrt{V}(M^{\mathrm{tmp}} - M) \\ &= R(\Gamma^{\mathrm{new}} + \sqrt{V}) M^{\mathrm{tmp}} = R(\Gamma + \partial\Gamma + \sqrt{V}) M^{\mathrm{tmp}} \\ &= R((\sqrt{\widetilde{V}} - \sqrt{V}) + (\sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}) + \sqrt{V}) M^{\mathrm{tmp}} = R\sqrt{W^{\mathrm{appr}}} M^{\mathrm{tmp}}, \end{aligned}$$

where the first step follows from the assigned value for Q^{tmp} on Line 6 of Algorithm 13, the second step follows from $Q = R\sqrt{V} M$ (Part 2 of Assumption D.1), the third step is by merging terms, the fourth step follows from $\Gamma^{\mathrm{new}} = \Gamma + \partial\Gamma$ (Part 6 of Definition D.2), the fifth step follows from $\Gamma = \sqrt{\widetilde{V}} - \sqrt{V}$ (Part 8 of Assumption D.1) and $\partial\Gamma = \sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}$ (Part 2 of Definition D.2), and the last step is by merging terms.

Part 3, 4 and 5. These directly follow from the assignment of β_1^{tmp} on Line 7 of Algorithm 13, the assignment of β_2^{tmp} on Line 8, and the assignment of ξ^{tmp} on Line 9. □

Claim D.20 (Part 3 of Lemma D.17)**.** *All invariants of Assumption D.1 are satisfied after the procedure* MatrixUpdate *(Algorithm 13).*

Proof. First note that g, $\widetilde{g}$, and T all remain the same after MatrixUpdate. Note that v and $\widetilde{v}$ are both assigned the value w^{appr} (Line 13 of Algorithm 13). Then using Claim D.19, and since we assigned M^{tmp} to M, Q^{tmp} to Q, β_1^{tmp} to β_1, β_2^{tmp} to β_2, and ξ^{tmp} to ξ, we have

$$M = A^\top (AVA^\top)^{-1} A, \qquad Q = R\sqrt{V}M,$$
$$\beta_1 = Q\sqrt{V}f(g), \qquad \beta_2 = M\sqrt{V}f(g),$$
$$\xi = \sqrt{\widetilde{V}}f(\widetilde{g}) - \sqrt{V}f(g).$$

We prove the other invariants directly from the assignment on Line 15 of Algorithm 13:

$$S = \emptyset = \mathrm{supp}(\widetilde{v} - v), \qquad \Delta = 0 = \widetilde{V} - V,$$
$$\Gamma = 0 = \sqrt{\widetilde{V}} - \sqrt{V}, \qquad B = I = \mathcal{L}_*[(\Delta_{S,S}^{-1} + M_{S,S})^{-1}],$$
$$\gamma_1 = 0 = B \cdot \mathcal{L}_r[\beta_{2,S}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi, \qquad \gamma_2 = 0 = \Gamma M \cdot \xi,$$
$$E = 0 = B \cdot \mathcal{L}_r[(M_\emptyset)^\top] = B \cdot \mathcal{L}_r[(M_S)^\top], \qquad F = 0 = R\Gamma \cdot \mathcal{L}_c[M_\emptyset] = R\Gamma \cdot \mathcal{L}_c[M_S].$$

□

D.4 Correctness of PartialMatrixUpdate

Lemma D.21 (Correctness of PartialMatrixUpdate)**.** *If all invariants of Assumption D.1 are satisfied before entering the procedure* PartialMatrixUpdate *(Algorithm 14), and $|S \cup \partial S| \leq 2n^a$, then after the procedure* PartialMatrixUpdate *we have the following guarantees:*

1. *$\widetilde{v} = w^{\text{appr}}$, and v remains the same.*
2. *g, $\widetilde{g}$ both remain the same.*
3. *All invariants of Assumption D.1 are still satisfied.*

Note that since we have the guarantee that $|S \cup \partial S| \leq 2n^a$, all $\mathcal{L}_r, \mathcal{L}_c, \mathcal{L}_*$ operators that appear in the procedure PartialMatrixUpdate are well-defined. And the Decompose function used in PartialMatrixUpdate is also well-defined.

Part 1 and 2 are proved in Claim D.22, and Part 3 is proved in Claim D.24 by using Claim D.23.

Claim D.22 (Part 1 and 2 of Lemma D.21)**.** *After the procedure* PartialMatrixUpdate *(Algorithm 14), we have $\widetilde{v} = w^{\text{appr}}$, and v, g, $\widetilde{g}$ all remain the same.*

Proof. $\widetilde{v} = w^{\text{appr}}$ follows directly from the value assignment of $\widetilde{v}$ on Line 16 of Algorithm 14.

The procedure PartialMatrixUpdate (Algorithm 14) does not modify v, g, $\widetilde{g}$, so they all remain the same. □

Claim D.23. *In the procedure* PartialMatrixUpdate *(Algorithm 14) before we refresh variables in the memory of the data structure, we have the following:*

1. $B^{\text{tmp}} = \mathcal{L}_*[((\Delta^{\text{new}}_{S^{\text{new}},S^{\text{new}}})^{-1} + M_{S^{\text{new}},S^{\text{new}}})^{-1}]$,
2. $\xi^{\text{tmp}} = \sqrt{W^{\text{appr}}} f(\widetilde{g}) - \sqrt{V} f(g)$,
3. $\gamma_1^{\text{tmp}} = B^{\text{tmp}} \cdot \mathcal{L}_r[\beta_{2,S^{\text{new}}}] + B^{\text{tmp}} \cdot \mathcal{L}_r[(M_{S^{\text{new}}})^\top] \cdot \xi^{\text{tmp}}$,
4. $\gamma_2^{\text{tmp}} = \Gamma^{\text{new}} M \cdot \xi^{\text{tmp}}$,

5. $F^{\text{tmp}} = R\Gamma^{\text{new}} \cdot \mathcal{L}_c[M_{S^{\text{new}}}]$,

6. $E^{\text{tmp}} = B^{\text{tmp}} \cdot \mathcal{L}_r[(M_{S^{\text{new}}})^\top]$.

Proof. **Part 1.** On Line 7 of Algorithm 14, we assigned B^{tmp} as

$$B^{\text{tmp}} \leftarrow B - BU'(C^{-1} + U^\top BU')^{-1}U^\top B,$$

where $(U', C, U) = \textsc{Decompose}(\mathcal{L}_*[(\Delta^{\text{new}}_{S^{\text{new}},S^{\text{new}}})^{-1} + M_{S^{\text{new}},S^{\text{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}])$, then using Lemma C.6 we have that $B^{\text{tmp}} = \mathcal{L}_*[((\Delta^{\text{new}}_{S^{\text{new}},S^{\text{new}}})^{-1} + M_{S^{\text{new}},S^{\text{new}}})^{-1}]$.

Part 2 and 3. Part 2 directly follows from the assignment of ξ^{tmp} on Line 10 of Algorithm 14.

And Part 3 directly follows from the assignment of γ_1^{tmp} on Line 11 of Algorithm 14.

Part 4. We have

$$\begin{aligned}
\gamma_2^{\text{tmp}} &= \gamma_2 + (\Gamma + \partial\Gamma)M(\sqrt{W^{\text{appr}}} - \sqrt{\widetilde{V}})f(\widetilde{g}) + \partial\Gamma M(\sqrt{\widetilde{V}}f(\widetilde{g}) - \sqrt{V}f(g)) \\
&= \Gamma M \cdot (\sqrt{\widetilde{V}}f(\widetilde{g}) - \sqrt{V}f(g)) + (\Gamma + \partial\Gamma)M(\sqrt{W^{\text{appr}}} - \sqrt{\widetilde{V}})f(\widetilde{g}) + \partial\Gamma M(\sqrt{\widetilde{V}}f(\widetilde{g}) - \sqrt{V}f(g)) \\
&= \Gamma M \cdot (\sqrt{W^{\text{appr}}}f(\widetilde{g}) - \sqrt{V}f(g)) + \partial\Gamma M(\sqrt{W^{\text{appr}}}f(\widetilde{g}) - \sqrt{V}f(g)) \\
&= (\Gamma + \partial\Gamma)M \cdot (\sqrt{W^{\text{appr}}}f(\widetilde{g}) - \sqrt{V}f(g)) \\
&= \Gamma^{\text{new}} M \cdot (\sqrt{W^{\text{appr}}}f(\widetilde{g}) - \sqrt{V}f(g)) \\
&= \Gamma^{\text{new}} M \cdot \xi^{\text{tmp}},
\end{aligned}$$

where the first step follows from the assignment of γ_2^{new} (Line 12 of Algorithm 14), the second step follows from $\gamma_2 = \Gamma M \xi = \Gamma M(\sqrt{\widetilde{V}}f(\widetilde{g}) - \sqrt{V}f(g))$ (Part 14 and 9 of Assumption D.1), the third and the fourth step both follow from merging terms, the fifth step follows from $\Gamma^{\text{new}} = \Gamma + \partial\Gamma$ (Part 2 of Definition D.2), and the sixth step follows from the Part 2 of this claim.

Part 5.

$$\begin{aligned}
F^{\text{tmp}} &= F + R\Gamma \cdot (\mathcal{L}_c[M_{\partial S \backslash S}] - \mathcal{L}_c[M_{S'}]) + R\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\text{new}}}] \\
&= R\Gamma \cdot \mathcal{L}_c[M_S] + R\Gamma \cdot (\mathcal{L}_c[M_{\partial S \backslash S}] - \mathcal{L}_c[M_{S'}]) + R\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\text{new}}}] \\
&= R\Gamma \cdot \mathcal{L}_c[M_{S^{\text{new}}}] + R\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\text{new}}}] \\
&= R\Gamma^{\text{new}} \cdot \mathcal{L}_c[M_{S^{\text{new}}}],
\end{aligned}$$

where the first step follows from the assignment of F^{tmp}(Line 8 of Algorithm 14), the second step follows from $F = R\Gamma \cdot \mathcal{L}_c[M_S]$ (Part 12 of Assumption D.1), the third step follows from $S' = (S \cup \partial S)\backslash S^{\text{new}}$ (Part 9 of Definition D.2) and thus $\mathcal{L}_c[M_{S'}] + \mathcal{L}_c[M_{S^{\text{new}}}] = \mathcal{L}_c[M_{S \cup \partial S}] = \mathcal{L}_c[M_S] + \mathcal{L}_c[M_{\partial S \backslash S}]$ by Part 2 of Remark C.3, and the fourth step follows from $\Gamma^{\text{new}} = \Gamma + \partial\Gamma$ (Part 6 of Definition D.2).

Part 6

$$\begin{aligned}
E^{\text{tmp}} &= E + B^{\text{tmp}}(\mathcal{L}_r[(M_{\partial S \backslash S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) - BU'(C^{-1} + U^\top BU')^{-1}U^\top E \\
&= B \cdot \mathcal{L}_r[(M_S)^\top] + B^{\text{tmp}}(\mathcal{L}_r[(M_{\partial S \backslash S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) - BU'(C^{-1} + U^\top BU')^{-1}U^\top B \cdot \mathcal{L}_r[(M_S)^\top] \\
&= (B - BU'(C^{-1} + U^\top BU')^{-1}U^\top B) \cdot \mathcal{L}_r[(M_S)^\top] + B^{\text{tmp}}(\mathcal{L}_r[(M_{\partial S \backslash S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) \\
&= B^{\text{tmp}}\mathcal{L}_r[(M_S)^\top] + B^{\text{tmp}}(\mathcal{L}_r[(M_{\partial S \backslash S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) \\
&= B^{\text{tmp}}\mathcal{L}_r[(M_{S^{\text{new}}})^\top]
\end{aligned}$$

where the first step follows from the assignment of E^{tmp}(Line 9 of Algorithm 14), the second step follows from $E = B \cdot \mathcal{L}_r[(M_S)^\top]$ (Part 11 of Assumption D.1), the fourth step follows from the

definition of B^{tmp}, and the fifth step follows from $S' = (S \cup \partial S) \backslash S^{\text{new}}$ (Part 9 of Definition D.2) and thus $\mathcal{L}_r[(M_{S'})^\top] + \mathcal{L}_r[(M_{S^{\text{new}}})^\top] = \mathcal{L}_r[(M_{S \cup \partial S})^\top] = \mathcal{L}_r[(M_S)^\top] + \mathcal{L}_r[(M_{\partial S \backslash S})^\top]$ by Part 2 of Remark C.3, and the fourth step follows from $\Gamma^{\text{new}} = \Gamma + \partial\Gamma$ (Part 6 of Definition D.2). □

Claim D.24 (Part 3 of Lemma D.21)**.** *All invariants of Assumption D.1 are satisfied after the procedure* PartialMatrixUpdate *(Algorithm 14).*

Proof. First note that the value of v, g, $\widetilde{g}$, T, M, Q, β_1 and β_2 all remain the same after the procedure PartialMatrixUpdate (Algorithm 14). Also note that $\widetilde{v}$ is assigned the value w^{appr} (Line 16 in Algorithm 14).

From the assignment of S, Δ, and Γ on Line 16 of Algorithm 14, we have

$$\begin{aligned} S = S^{\text{new}} &= \mathrm{supp}(w^{\text{appr}} - v) = \mathrm{supp}(\widetilde{v} - v), \\ \Delta = \Delta^{\text{new}} &= W^{\text{appr}} - V = \widetilde{V} - V, \\ \Gamma = \Gamma^{\text{new}} &= \sqrt{W^{\text{appr}}} - \sqrt{V} = \sqrt{\widetilde{V}} - \sqrt{V}. \end{aligned}$$

Then using Claim D.23, and since we assigned B^{tmp} to B, ξ^{tmp} to ξ, γ_1^{new} to γ_1, and γ_2^{new} to γ_2, E^{tmp} to E, F^{tmp} to F, we have

$$\begin{aligned} B &= \mathcal{L}_*[((\Delta_{S,S})^{-1} + M_{S,S})^{-1}], & \xi &= \sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g), \\ \gamma_1 &= B \cdot \mathcal{L}_r[\beta_{2,S}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi, & \gamma_2 &= \Gamma M \cdot \xi, \\ E &= B \cdot \mathcal{L}_r[(M_S)^\top], & F &= R\Gamma \cdot \mathcal{L}_c[M_S]. \end{aligned}$$

□

D.5 Correctness of VectorUpdate

Lemma D.25 (Correctness of VectorUpdate)**.** *If all invariants of Assumption D.1 are satisfied before entering the procedure* VectorUpdate *(Algorithm 15), then after the procedure* VectorUpdate *we have the following guarantees:*

1. *v, $\widetilde{v}$ both remain the same.*
2. *$g = \widetilde{g} = h^{\text{appr}}$.*
3. *All invariants of Assumption D.1 are still satisfied.*

First note that from Corollary D.15 we have that $|S| \le n^a$, so all $\mathcal{L}_r$ operators that appear in the procedure VectorUpdate are well-defined.

Part 1 and 2 are proved in Claim D.26, and Part 3 is proved in Claim D.28 by using Claim D.27.

Claim D.26 (Part 1 and 2 of Lemma D.25)**.** *After the procedure* VectorUpdate *(Algorithm 15), v and $\widetilde{v}$ both remain the same, and $g = \widetilde{g} = h^{\text{appr}}$.*

Proof. The procedure VectorUpdate does not modify v or $\widetilde{v}$, so they both remain the same. And $g = \widetilde{g} = h^{\text{appr}}$ follows directly from the value assignment of g and $\widetilde{g}$ on Line 11 of Algorithm 15. □

Claim D.27. *In the procedure* VectorUpdate *(Algorithm 15) before we refresh variables in the memory of the data structure, we have the following:*

1. $\beta_1^{\text{tmp}} = Q\sqrt{V}f(h^{\text{appr}})$,
2. $\beta_2^{\text{tmp}} = M\sqrt{V}f(h^{\text{appr}})$,
3. $\xi^{\text{tmp}} = (\sqrt{\widetilde{V}} - \sqrt{V})f(h^{\text{appr}})$,
4. $\gamma_1^{\text{tmp}} = B \cdot \mathcal{L}_r[\beta_{2,S}^{\text{tmp}}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi^{\text{tmp}}$,
5. $\gamma_2^{\text{tmp}} = \Gamma M \cdot \xi^{\text{tmp}}$.

Proof. **Part 1.** From the assignment of β_1^{tmp} on Line 4 of Algorithm 15, we have

$$\beta_1^{\text{tmp}} = \beta_1 + Q\sqrt{V}(f(h^{\text{appr}}) - f(g)) = Q\sqrt{V}f(g) + Q\sqrt{V}(f(h^{\text{appr}}) - f(g)) = Q\sqrt{V}f(h^{\text{appr}}),$$

where the second step follows from $\beta_1 = Q\sqrt{V}f(g)$ (Part 3 of Assumption D.1).
Part 2. From the assignment of β_2^{tmp} on Line 5 of Algorithm 15, we have

$$\beta_2^{\text{tmp}} = \beta_2 + M\sqrt{V}(f(h^{\text{appr}}) - f(g)) = M\sqrt{V}f(g) + M\sqrt{V}(f(h^{\text{appr}}) - f(g)) = M\sqrt{V}f(h^{\text{appr}}),$$

where the second step follows from $\beta_2 = M\sqrt{V}f(g)$ (Part 4 of Assumption D.1).
Part 3, 4 and 5. These directly follow from the assignment of ξ^{tmp} (Line 6), γ_1^{tmp} (Line 7), and γ_2^{tmp} (Line 8) in Algorithm 15. □

Claim D.28 (Part 3 of Lemma D.25)**.** *All invariants of Assumption D.1 are satisfied after the procedure* VectorUpdate *(Algorithm 15).*

Proof. First note that v, $\widetilde{v}$, M, Q, B, Δ, Γ, and S all remain the same after the procedure VectorUpdate. Also note that g and $\widetilde{g}$ are both assigned the value h^{appr} (Line 11 of Algorithm 15).

Then using Claim D.27, and since we assigned β_1^{tmp} to β_1, β_2^{tmp} to β_2, ξ^{tmp} to ξ, γ_1^{tmp} to γ_1, and γ_2^{tmp} to γ_2, we have

$$\beta_1 = Q\sqrt{V}f(g), \qquad \beta_2 = M\sqrt{V}f(g),$$
$$\gamma_1 = B \cdot \mathcal{L}_r[\beta_{2,S}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi, \qquad \gamma_2 = \Gamma M \cdot \xi,$$
$$\xi = (\sqrt{\widetilde{V}} - \sqrt{V})f(h^{\text{appr}}) = \sqrt{\widetilde{V}}f(\widetilde{g}) - \sqrt{V}f(g).$$

Also, from the assignment of T on Line 12 of Algorithm 15, we have $T = \emptyset = \text{supp}(\widetilde{g} - g)$. □

D.6 Correctness of PartialVectorUpdate

Lemma D.29 (Correctness of PartialVectorUpdate)**.** *If all invariants of Assumption D.1 are satisfied before entering the procedure* PartialVectorUpdate *(Algoritm 16), then after the procedure* PartialVectorUpdate *we have the following guarantees:*

1. *v, $\widetilde{v}$ both remain the same.*
2. *$\widetilde{g} = h^{\text{appr}}$, and g remains the same.*
3. *All invariants of Assumption D.1 are still satisfied.*

First note that from Corollary D.15 we have that $|S| \le n^a$, so all $\mathcal{L}_r$ operators that appear in the procedure PartialVectorUpdate are well-defined.

Part 1 and 2 are proved in Claim D.30, and Part 3 is proved in Claim D.32 by using Claim D.31.

Claim D.30 (Part 1 and 2 of Lemma D.29)**.** *After the procedure* PartialVectorUpdate *(Algorithm 16), we have $\widetilde{g} = h^{\text{appr}}$, and v, $\widetilde{v}$, g all remain the same.*

Proof. $\widetilde{g} = h^{\text{appr}}$ follows directly from the value assignment of $\widetilde{g}$ on Line 10 of Algorithm 16.

The procedure PartialVectorUpdate (Algorithm 16) does not modify v, $\widetilde{v}$, g, so they all remain the same. □

Claim D.31. *In the procedure* PartialVectorUpdate *(Algorithm 16) before we refresh variables in the memory of the data structure, we have the following:*

1. $\xi^{\text{tmp}} = \sqrt{\widetilde{V}} f(h^{\text{appr}}) - \sqrt{V} f(g)$,
2. $\gamma_1^{\text{tmp}} = B \cdot \mathcal{L}_r[\beta_{2,S}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi^{\text{tmp}}$,
3. $\gamma_2^{\text{tmp}} = \Gamma M \cdot \xi^{\text{tmp}}$.

Proof. **Part 1.** This directly follows from the assignment of ξ^{tmp} on Line 4 of Algorithm 16.
Part 2. From the assignment of γ_1^{tmp} on Line 5 of Algorithm 16, we have

$$\begin{aligned}
\gamma_1^{\text{tmp}} &= \gamma_1 + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \sqrt{\widetilde{V}}\big(f(h^{\text{appr}}) - f(\widetilde{g})\big) \\
&= B \cdot \mathcal{L}_r[\beta_{2,S}] + B \cdot \mathcal{L}_r[(M_S)^\top](\sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g)) + B \cdot \mathcal{L}_r[(M_S)^\top]\sqrt{\widetilde{V}}\big(f(h^{\text{appr}}) - f(\widetilde{g})\big) \\
&= B \cdot \mathcal{L}_r[\beta_{2,S}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \Big((\sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g)) + \sqrt{\widetilde{V}}(f(h^{\text{appr}}) - f(\widetilde{g}))\Big) \\
&= B \cdot \mathcal{L}_r[\beta_{2,S}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot (\sqrt{\widetilde{V}} f(h^{\text{appr}}) - \sqrt{V} f(g)) \\
&= B \cdot \mathcal{L}_r[\beta_{2,S}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi^{\text{tmp}},
\end{aligned}$$

where the second step follows from the invariant of γ_1 (Part 13 in Assumption D.1), the third and the fourth steps follow from merging terms, and the last step follows from Part 1 of this lemma.
Part 3. From the assignment of γ_2^{tmp} on Line 6 of Algorithm 16, we have

$$\begin{aligned}
\gamma_2^{\text{tmp}} &= \gamma_2 + \Gamma M \sqrt{\widetilde{V}}\big(f(h^{\text{appr}}) - f(\widetilde{g})\big) = \Gamma M(\sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g)) + \Gamma M \sqrt{\widetilde{V}}\big(f(h^{\text{appr}}) - f(\widetilde{g})\big) \\
&= \Gamma M(\sqrt{\widetilde{V}} f(h^{\text{appr}}) - \sqrt{V} f(g)) = \Gamma M \cdot \xi^{\text{tmp}},
\end{aligned}$$

where the second step follows from the invariant of γ_2 (Part 14 in Assumption D.1), the third step follows from merging terms, and the last step follows from Part 1 of this lemma. □

Claim D.32 (Part 3 of Lemma D.29)**.** *All invariants of Assumption D.1 are satisfied after the procedure* PartialVectorUpdate *(Algorithm 16).*

Proof. First note that v, $\widetilde{v}$, g, M, Q, B, Δ, Γ, S, β_1, and β_2 all remain the same after the procedure VectorUpdate. Also note that $\widetilde{g}$ is assigned the value h^{appr} (Line 10 of Algorithm 16).

Then using Claim D.27, and since we assigned ξ^{tmp} to ξ, γ_1^{tmp} to γ_1, and γ_2^{tmp} to γ_2, we have

$$\begin{aligned}
&\gamma_1 = B \cdot \mathcal{L}_r[\beta_{2,S}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi, && \gamma_2 = \Gamma M \cdot \xi, \\
&\xi = \sqrt{\widetilde{V}} f(h^{\text{appr}}) - \sqrt{V} f(g) = \sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g).
\end{aligned}$$

Finally, from the assignment of T on Line 9, we have $T = \text{supp}(h^{\text{appr}} - g) = \text{supp}(\widetilde{g} - g)$. □

D.7 Correctness of INITIALIZE

Lemma D.33 (Correctness of INITIALIZE)**.** *When initialized, all the invariants of the data structure members stated in Assumption D.1 are satisfied.*

Proof. Since in the beginning v and $\widetilde{v}$ are both assigned the value w_0, and g and $\widetilde{g}$ are both assigned the value h_0, it is obvious that $S = \emptyset$, $T = \emptyset$, $\Delta = 0$, $\Gamma = 0$, $\xi = 0$, $\gamma_1 = 0$, $\gamma_2 = 0$ all satisfy their invariant requirement of Assumption D.1. Also, $B = I = \mathcal{L}_*[0]$ also satisfies the invariant requirement. Finally note that the initial assignment of M, Q, β_1, β_2 directly satisfy their invariant requirement of Assumption D.1. □

E Data structure : time per call

In Section E we provide a worst-case analysis of the running time per call for the five major procedures MATRIXUPDATE, PARTIALMATRIXUPDATE, VECTORUPDATE, PARTIALVECTORUPDATE, QUERY. We prove the amortized running time of these procedures later in Section F. In Section E, we ignore the running time of adding two vectors since it is only $O(n)$.

Procedure	Time per Call	Amortized Time	Lemma
QUERY	$\mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}) + n^{1+b}$	$\mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}) + n^{1+b}$	Lemma E.3
MATRIXUPDATE	$\mathcal{T}_{\mathrm{mat}}(k, n, n)$	$\widetilde{O}(n^{\omega-1/2} + n^{2-a/2})$	Lemma E.12, F.19
PARTIALMATRIXUPDATE	$\mathcal{T}_{\mathrm{mat}}(\widetilde{k}, n^a, n)$	$\widetilde{O}(n^{1+(\omega-3/2)a} + n^{1+a-\widetilde{a}/2})$	Lemma E.18, F.30
VECTORUPDATE	$pn + n^{2a}$	$\widetilde{O}(n^{1.5})$	Lemma E.27, F.35
PARTIALVECTORUPDATE	$\widetilde{p}n^a + n^{2a}$	$\widetilde{O}(n^{2a-\widetilde{a}/2})$	Lemma E.33, F.36

Table 11: Time for different procedures. Summary of Section E and Section F.

E.1 Sparsity guarantees

We first present some sparsity bounds that will be useful in the time analysis.

Procedure	$\|w^{\mathrm{appr}} - v\|_0$	$\|w^{\mathrm{appr}} - \widetilde{v}\|_0$	$\|h^{\mathrm{appr}} - g\|_0$	$\|h^{\mathrm{appr}} - \widetilde{g}\|_0$	$\|\widetilde{v} - v\|_0$	$\|\widetilde{g} - g\|_0$
MATRIXUPDATE	$= k$	/	/	/	$\leq n^a$	$\leq n^a$
P.MATRIXUPDATE	$\leq n^a$	$= \widetilde{k} \leq 2n^a$	/	/	$\leq n^a$	$\leq n^a$
VECTORUPDATE	$\leq n^a$	$\leq n^{\widetilde{a}}$	$= p$	/	$\leq n^a$	$\leq n^a$
P.VECTORUPDATE	$\leq n^a$	$\leq n^{\widetilde{a}}$	$\leq n^a$	$= \widetilde{p} \leq 2n^a$	$\leq n^a$	$\leq n^a$
QUERY	$\leq n^a$	$\leq n^{\widetilde{a}}$	$\leq n^a$	$\leq n^{\widetilde{a}}$	$\leq n^a$	$\leq n^a$

Table 12: Sparsity guarantees of w^{appr}, $\widetilde{v}$, v, h^{appr}, $\widetilde{g}$, g when entering the procedures (Part 1 of Lemma E.1). We say some vector $x \in \mathbb{R}^n$ is k-sparse, it means that $\mathrm{supp}(x) = k$.

Lemma E.1 (Sparsity guarantees)**.** *The members of data structure presented in Table 12 and Table 13 all follow the invariants of Assumption D.1, and the local variables presented in Table 13 all follow the definition of Definition D.2. We have the following sparsity guarantees.*

1. *When entering the procedures* MATRIXUPDATE*,* PARTIALMATRIXUPDATE*,* VECTORUPDATE*,* PARTIALVECTORUPDATE*, and* QUERY*, we have the sparsity bounds on* $\|w^{\mathrm{appr}} - v\|_0$*,* $\|w^{\mathrm{appr}} - \widetilde{v}\|_0$*,* $\|h^{\mathrm{appr}} - g\|_0$*,* $\|h^{\mathrm{appr}} - \widetilde{g}\|_0$*,* $\|\widetilde{v} - v\|_0$*, and* $\|\widetilde{g} - g\|_0$ *as presented in Table 12.*

Procedure	$\|\Delta + \partial\Delta\|_0$, $\|\Gamma + \partial\Gamma\|_0$	$\|\partial\Delta\|_0$, $\|\partial\Gamma\|_0$, $\|\partial S\|$	$\|\xi + \partial\xi\|_0$	$\|\partial\xi\|_0$	$\|\Delta\|_0$, $\|\xi\|_0$, $\|\Gamma\|_0$, $\|S\|$	$\|S \cup \partial S\|$
MatrixUpdate	$= k$	/	/	/	$\leq n^a$	$\leq 3k$
P.MatrixUpdate	$\leq n^a$	$= \widetilde{k} \leq 2n^a$	/	/	$\leq n^a$	$\leq 3n^a$
VectorUpdate	$\leq n^a$	$\leq n^{\widetilde{a}}$	$= p$	/	$\leq n^a$	/
P.VectorUpdate	$\leq n^a$	$\leq n^{\widetilde{a}}$	$\leq n^a$	$= \widetilde{p} \leq 2n^a$	$\leq n^a$	/
Query	$\leq n^a$	$\leq n^{\widetilde{a}}$	$\leq n^a$	$\leq n^{\widetilde{a}}$	$\leq n^a$	$\leq 2n^a$

Table 13: Sparsity guarantees of other local variables (Definition D.2) when entering the procedures (Part 2 of Lemma E.1). We say some vector $x \in \mathbb{R}^n$ is k-sparse, it means that $\mathrm{supp}(x) = k$.

2. *When entering the procedures* MatrixUpdate, PartialMatrixUpdate, VectorUpdate, PartialVectorUpdate, *and* Query, *we have the sparsity bounds on* $\Delta + \partial\Delta$, $\Gamma + \partial\Gamma$, $S \cup \partial S$, $\partial\Delta$, $\partial\Gamma$, ∂S, $\xi + \partial\xi$, $\partial\xi$, Δ, Γ, S, ξ *as presented in Table 13.*

Proof. **Part 1.** The first four columns follow from Corollary D.16, and the last two columns follow from Corollary D.15.

Part 2. For Col. 1, we have

$$\begin{aligned}\|\Delta + \partial\Delta\|_0 &= \|W^{\mathrm{appr}} - V\|_0 = \|w^{\mathrm{appr}} - v\|_0,\\ \|\Gamma + \partial\Gamma\|_0 &= \|\sqrt{W^{\mathrm{appr}}} - \sqrt{V}\|_0 = \|w^{\mathrm{appr}} - v\|_0,\end{aligned}$$

the bounds of this column then follow from Col. 1 of Table 12.

For Col. 2, we have

$$\begin{aligned}\|\partial\Delta\|_0 &= \|W^{\mathrm{appr}} - \widetilde{V}\|_0 = \|w^{\mathrm{appr}} - \widetilde{v}\|_0,\\ \|\partial\Gamma\|_0 &= \|\sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}\|_0 = \|w^{\mathrm{appr}} - \widetilde{v}\|_0,\\ |\partial S| &= \|W^{\mathrm{appr}} - \widetilde{V}\|_0 = \|w^{\mathrm{appr}} - \widetilde{v}\|_0,\end{aligned}$$

the bounds of this column then follow from Col. 2 of Table 12.

For Col. 3 we have

$$\begin{aligned}\|\xi + \partial\xi\|_0 &= \|\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \sqrt{V} f(g)\|_0 \leq \max\{\|\sqrt{W^{\mathrm{appr}}} - \sqrt{V}\|_0, \|f(h^{\mathrm{appr}}) - f(g)\|_0\}\\ &= \max\{\|w^{\mathrm{appr}} - v\|_0, \|h^{\mathrm{appr}} - g\|_0\},\end{aligned}$$

the bounds of this column then follow from Col. 1 and Col. 3 of Table 12 and the fact $p \geq n^a$ (since we have the equivalent fact for p as of Fact F.10 for k).

For Col. 4 we have

$$\begin{aligned}\|\partial\xi\|_0 &= \|\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \sqrt{\widetilde{V}} f(\widetilde{g})\|_0 \leq \max\{\|\sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}\|_0, \|f(h^{\mathrm{appr}}) - f(\widetilde{g})\|_0\}\\ &= \max\{\|w^{\mathrm{appr}} - \widetilde{v}\|_0, \|h^{\mathrm{appr}} - \widetilde{g}\|_0\},\end{aligned}$$

the bounds of this column then follow from Col. 2 and Col. 4 of Table 12 and the fact $\widetilde{p} \geq n^{\widetilde{a}}$ (since we have the equivalent fact for $\widetilde{p}$ as of Fact F.11 for $\widetilde{k}$).

For Col. 5 we have

$$\begin{aligned}\|\Delta\|_0 &= \|\widetilde{V} - V\|_0 = \|\widetilde{v} - v\|_0,\\ \|\Gamma\|_0 &= \|\sqrt{\widetilde{V}} - \sqrt{V}\|_0 = \|\widetilde{v} - v\|_0,\end{aligned}$$

$$|S| = |\operatorname{supp}(\widetilde{v} - v)| = \|\widetilde{v} - v\|_0,$$
$$\|\xi\|_0 = \|\sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g)\|_0 \leq \max\{\|\widetilde{v} - v\|_0, \|\widetilde{g} - g\|_0\},$$

the bounds of this column then follow from Col. 5 and Col. 6 of Table 12.

For the first part (MatrixUpdate) of Col. 6, we have

$$\begin{aligned} |S \cup \partial S| &\leq |S| + |\partial S| \leq |\operatorname{supp}(v - \widetilde{v})| + |\operatorname{supp}(w^{\mathrm{appr}} - \widetilde{v})| \\ &\leq |\operatorname{supp}(v - \widetilde{v})| + |\operatorname{supp}(w^{\mathrm{appr}} - v)| + |\operatorname{supp}(v - \widetilde{v})| \leq n^a + k + n^a \leq 3k, \end{aligned}$$

where the first and the third steps both follow from triangle inequality, the second step follows from $S = \operatorname{supp}(v - \widetilde{v})$ (part 5 Assumption D.1) and $\partial S = \operatorname{supp}(w^{\mathrm{appr}} - \widetilde{v})$ (Part 4 of Definition D.2), the fourth step follows from Col. 1 and Col. 5 of Table 12, the last step follows from the fact that $k \geq n^a$ (Fact F.10).

For the second part (PartialMatrixUpdate) of Col. 6, we have

$$|S \cup \partial S| \leq |S| + |\partial S| \leq |\operatorname{supp}(v - \widetilde{v})| + |\operatorname{supp}(w^{\mathrm{appr}} - \widetilde{v})| \leq n^a + 2n^a = 3n^a,$$

where the first two steps are the same as previous inequality, and the third step follows from Col. 2 and Col. 5 of Table 12.

For the fifth part (Query) of Col. 6, we have

$$|S \cup \partial S| \leq |S| + |\partial S| \leq |\operatorname{supp}(v - \widetilde{v})| + |\operatorname{supp}(w^{\mathrm{appr}} - \widetilde{v})| \leq n^a + n^{\widetilde{a}} \leq 2n^a,$$

where the first two steps are the same as previous inequality, the third step follows from Col. 2 and Col. 5 of Table 12. □

Procedure	**Lemma**	**Section**
Query	Lemma E.3	Section E.2
MatrixUpdate	Lemma E.12	Section E.3
PartialMatrixUpdate	Lemma E.18	Section E.4
VectorUpdate	Lemma E.27	Section E.5
PartialVectorUpdate	Lemma E.33	Section E.6
Initialize	Lemma E.37	Section E.7

Table 14: Summary of the section that proves running time per call.

E.2 Running time of Query

The goal of this section is to prove Lemma E.3. We will use the following sparsity guarantees that are proved in Lemma E.1.

Fact E.2 (Sparsity guarantees for Query)**.** *When entering* Query *we have the following sparsity guarantee (from Table 13):*

1. $\|\Gamma + \partial\Gamma\|_0 \leq n^a$,
2. $\|\partial\Gamma\|_0 \leq n^{\widetilde{a}}$,
3. $\|\Gamma\|_0 \leq n^a$,
4. $\|\xi + \partial\xi\|_0 \leq n^a$,
5. $\|\partial\xi\|_0 \leq n^{\widetilde{a}}$,
6. $|\partial S| \leq n^{\widetilde{a}}$,
7. $|S| \leq n^a$.

Lemma E.3 (Running time of QUERY)**.** *In procedure* QUERY *(Algorithm 12), it takes*

1. $O(n^{1+b}+n^{a+\widetilde{a}})$ *time to compute*
$$r_2 \leftarrow Q[j]\xi + R[j]\gamma_2 + R[j]\partial\Gamma M(\xi+\partial\xi) + \big(Q[j]+R[j]\Gamma M\big)\partial\xi,$$
2. $O(n^{1+b})$ *time to compute*
$$r_3 \leftarrow R[j](\Gamma+\partial\Gamma)\beta_2,$$
3. $O(n^{a+\widetilde{a}})$ *time to compute*
$$\partial\gamma \leftarrow B\cdot\mathcal{L}_r[(\beta_2)_{\partial S\setminus S}] + B\cdot\mathcal{L}_r[(M_{\partial S\setminus S})^\top]\cdot(\xi+\partial\xi) + E\cdot\partial\xi,$$
4. $O(n^{a+\widetilde{a}})$ *time to compute*
$$(U', C, U) \leftarrow \textsc{Decompose}\Big(\mathcal{L}_*[(\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}]\Big),$$
5. $O(\mathcal{T}_{\mathrm{mat}}(n^a, n^{\widetilde{a}}, n^{\widetilde{a}}))$ *time to compute*
$$E^{\mathrm{tmp}} \leftarrow E_{\partial S} - B_{(\partial S\setminus S)}M_{(\partial S\setminus S),\partial S},\ E^{\mathrm{tmp}}_{S'} \leftarrow -E^{\mathrm{tmp}}_{S'},\ E^{\mathrm{tmp}}_{(S\cap\partial S)\setminus S'} \leftarrow 0,\ \textit{and}$$
$$U^{\mathrm{tmp}} \leftarrow [B_{\partial S}, B_{\partial S}, E^{\mathrm{tmp}}],$$
6. $O(\mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}))$ *time to compute*
$$\gamma^{\mathrm{tmp}} \leftarrow U^{\mathrm{tmp}}(C^{-1} + U^\top U^{\mathrm{tmp}})^{-1}U^\top\cdot(\gamma_1+\partial\gamma),$$
7. $O(n^{\widetilde{a}+a}+n^{1+b})$ *time to compute*
$$r_4 \leftarrow \Big(\mathcal{L}_c[(Q[j])_{S^{\mathrm{new}}}] + F[j] + R[j]\Gamma\cdot(\mathcal{L}_c[M_{\partial S\setminus S}] - \mathcal{L}_c[M_{S'}]) + R[j]\partial\Gamma\cdot\mathcal{L}_c[M_{S^{\mathrm{new}}}]\Big)\cdot(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma).$$

Overall, the time to compute r (which is $r_1+r_2+r_3+r_4$) is
$$O(n^{1+b} + \mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}})).$$

Claim E.4 (Part 1 of Lemma E.3)**.** *In procedure* QUERY *(Algorithm 12), it takes $O(n^{1+b}+n^{a+\widetilde{a}})$ time to compute*
$$r_2 \leftarrow Q[j]\xi + R[j]\gamma_2 + \underbrace{R[j]\partial\Gamma M(\xi+\partial\xi)}_{a_1} + \underbrace{\big(Q[j]+R[j]\Gamma M\big)\partial\xi}_{a_2}.$$

Proof. The running time of this step can be split into the following parts:

The first part is to compute $Q[j]\xi$ by multiplying a $n^b\times n$ matrix $Q[j]$ with a $n\times 1$ vector ξ. It takes $O(n^{1+b})$ time. The second part is to compute $R[j]\gamma_2$ by multiplying a $n^b\times n$ matrix $R[j]$ with a $n\times 1$ vector γ_2. It takes $O(n^{1+b})$ time.

The third part is to compute a_1 as follows:

1. We multiply a $n^{\widetilde{a}}$-sparse $n\times n$ diagonal matrix $\partial\Gamma$ (Part 2 of Fact E.2) with a $n\times n$ matrix M and then with a n^a-sparse $n\times 1$ vector $(\xi+\partial\xi)$ (Part 4 of Fact E.2). It takes $O(n^{a+\widetilde{a}})$ time.

2. We multiply a $n^b \times n$ matrix $R[j]$ and a $n \times 1$ vector $(\partial\Gamma M(\xi + \partial\xi))$. It takes $O(n^{1+b})$ time.

So computing a_1 takes $O(n^{a+\widetilde{a}} + n^{1+b})$ time in total.

The fourth part is to compute a_2 as follows:

1. We multiply a n^a-sparse $n \times n$ diagonal matrix Γ (Part 3 of Fact E.2) with a $n \times n$ matrix M and then with a $n^{\widetilde{a}}$-sparse $n \times 1$ vector $\partial\xi$ (Part 5 of Fact E.2). It takes $O(n^{a+\widetilde{a}})$ time.

2. We multiply a $n^b \times n$ matrix $R[j]$ with a $n \times 1$ vector $(\Gamma M \partial\xi)$. It takes $O(n^{1+b})$ time.

3. We multiply a $n^b \times n$ matrix $Q[j]$ and a $n^{\widetilde{a}}$-sparse $n \times 1$ vector $\partial\xi$ (Part 5 of Fact E.2). It takes $O(n^{\widetilde{a}+b})$ time.

So computing a_2 takes $O(n^{a+\widetilde{a}} + n^{1+b})$ time in total since $\widetilde{a} \leq 1$.

Thus the total running time is $O(n^{1+b} + n^{a+\widetilde{a}})$. □

Claim E.5 (Part 2 of Lemma E.3)**.** *In procedure* QUERY *(Algorithm 12), it takes $O(n^{1+b})$ time to compute $r_3 \leftarrow R[j](\Gamma + \partial\Gamma)\beta_2$.*

Proof. The running time of this step can be split into the following parts.

1. We multiply a n^a-sparse $n \times n$ diagonal matrix $(\Gamma + \partial\Gamma)$ (Part 1 of Fact E.2) and a $n \times 1$ vector β_2. It takes $O(n^a)$ time.

2. We multiply a $n^b \times n$ matrix $R[j]$ and a $n \times 1$ vector $((\Gamma + \partial\Gamma)\beta_2)$. It takes $O(n^{1+b})$ time.

The total running time is $O(n^a + n^{1+b}) = O(n^{1+b})$ since $a \leq 1$. □

Claim E.6 (Part 3 of Lemma E.3)**.** *In procedure* QUERY *(Algorithm 12), it takes $O(n^{a+\widetilde{a}})$ time to compute $\partial\gamma \leftarrow B \cdot \mathcal{L}_r[(\beta_2)_{\partial S \backslash S}] + B \cdot \mathcal{L}_r[(M_{\partial S \backslash S})^\top] \cdot (\xi + \partial\xi) + E \cdot \partial\xi$.*

Proof. The running time of this step can be split into the following parts.

1. We multiply a $6n^a \times 6n^a$ matrix B with a $n^{\widetilde{a}}$-sparse $6n^a \times 1$ vector $\mathcal{L}_r[(\beta_2)_{\partial S \backslash S}]$ (since $|\partial S \backslash S| \leq |\partial S| \leq n^{\widetilde{a}}$ by Part 6 of Fact E.2). It takes $O(n^{a+\widetilde{a}})$ time.

2. We multiply a $6n^a \times n$ matrix $\mathcal{L}_r[(M_{\partial S \backslash S})^\top]$ that only has $n^{\widetilde{a}}$ non-zero rows (since $|\partial S \backslash S| \leq |\partial S| \leq n^{\widetilde{a}}$ by Part 6 of Fact E.2) with a n^a-sparse $n \times 1$ vector $(\xi + \partial\xi)$ (Part 4 of Fact E.2). It takes $O(n^{a+\widetilde{a}})$ time.

3. We multiply a $6n^a \times 6n^a$ matrix B with a $n^{\widetilde{a}}$-sparse $6n^a \times 1$ vector $(\mathcal{L}_r[(M_{\partial S \backslash S})^\top](\xi + \partial\xi))$ (since $|\partial S \backslash S| \leq |\partial S| \leq n^{\widetilde{a}}$ by Part 6 of Fact E.2). It takes $O(n^{a+\widetilde{a}})$ time.

4. We multiply a $6n^a \times n$ matrix E with a $n^{\widetilde{a}}$-sparse vector $\partial\xi$ (Part 5 of Fact E.2). It takes $O(n^{a+\widetilde{a}})$ time.

Thus the total running time is $O(n^{a+\widetilde{a}})$. □

Claim E.7 (Part 4 of Lemma E.3)**.** *In procedure* QUERY *(Algorithm 12), it takes $O(n^{a+\widetilde{a}})$ time to compute*

$$(U', C, U) \leftarrow \textsc{Decompose}\Big(\mathcal{L}_*[(\Delta^{\mathrm{new}}_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}}, S^{\mathrm{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}]\Big).$$

And the size of the computed matrices are $U', U \in \mathbb{R}^{n^a \times c}$, $C \in \mathbb{R}^{c \times c}$, where $c \leq O(n^{\widetilde{a}})$.

Proof. For ease of notation, we denote $N := \mathcal{L}_*[(\Delta^{\text{new}}_{S^{\text{new}},S^{\text{new}}})^{-1} + M_{S^{\text{new}},S^{\text{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}]$.

From Lemma C.5 we know that the non-zero entries of N can be split into three parts: $N_{\partial S,(S\setminus\partial S)}$, $N_{(S\setminus\partial S),\partial S}$, and $N_{\partial S,\partial S}$. Thus we don't need to compute N explicitly, but only compute the non-zero entries of N, which takes $O(|\partial S| \cdot |S\setminus\partial S| + |\partial S| \cdot |\partial S|) = O(n^{a+\widetilde{a}})$ time (from Part 7 and Part 6 of Fact E.2 and since $\widetilde{a} \le a$).

Then N satisfies the requirement of Lemma C.4 with $S_1 = S\setminus\partial S$, $S_2 = \partial S$, so the function DECOMPOSE can be computed in $O(n^a|S_2|) = O(n^a|\partial S|) = O(n^{a+\widetilde{a}})$ time (Part 6 of Fact E.2). Also using Lemma C.4, we know that the computed matrix C has size $c \times c = (3|S_2|) \times (3|S_2|) = (3|\partial S|) \times (3|\partial S|)$, thus $c \le O(n^{\widetilde{a}})$ (Part 6 of Fact E.2).

Thus the total running time is $O(n^{a+\widetilde{a}})$. □

Claim E.8 (Part 5 of Lemma E.3)**.** *In procedure* QUERY *(Algorithm 12), it takes* $O(\mathcal{T}_{\text{mat}}(n^a, n^{\widetilde{a}}, n^{\widetilde{a}}))$ *time to compute*

$$E^{\text{tmp}} \leftarrow E_{\partial S} - B_{(\partial S\cap S)}M_{(\partial S\cap S),\partial S}; \quad E^{\text{tmp}}_{S'} \leftarrow -E^{\text{tmp}}_{S'}; \quad E^{\text{tmp}}_{(S\cap\partial S)\setminus S'} \leftarrow 0; \quad U^{\text{tmp}} \leftarrow [B_{\partial S}, B_{\partial S}, E^{\text{tmp}}].$$

Proof. To compute the initial $E^{\text{tmp}} \leftarrow E_{\partial S} - B_{(\partial S\cap S)}M_{(\partial S\cap S),\partial S}$, we need to multiply a $6n^a \times |\partial S\cap S|$ matrix $B_{(\partial S\cap S)}$ with a $|\partial S\cap S| \times |\partial S|$ matrix $M_{(\partial S\cap S),\partial S}$, and this takes $\mathcal{T}_{\text{mat}}(6n^a, |\partial S\cap S|, |\partial S|) = O(\mathcal{T}_{\text{mat}}(n^a, n^{\widetilde{a}}, n^{\widetilde{a}}))$ time (Part 6 of Fact E.2).

Finally note that the other two steps $E^{\text{tmp}}_{S'} \leftarrow -E^{\text{tmp}}_{S'}$ and $E^{\text{tmp}}_{(S\cap\partial S)\setminus S'} \leftarrow 0$ to adjust E^{tmp} takes the same time as the size of E^{tmp}, which is $n^a \times |\partial S| = O(n^{a+\widetilde{a}})$ (Part 6 of Fact E.2). Computing U^{tmp} only needs to copy entries from the already computed B and E^{tmp}, so it also takes the same time as the size of U^{tmp}, which is $n^a \times 3|\partial S| = O(n^{a+\widetilde{a}})$. Thus the total running time is

$$O(\mathcal{T}_{\text{mat}}(n^a, n^{\widetilde{a}}, n^{\widetilde{a}}) + n^{a+\widetilde{a}}) = O(\mathcal{T}_{\text{mat}}(n^a, n^{\widetilde{a}}, n^{\widetilde{a}})).$$

□

Claim E.9 (Part 6 of Lemma E.3)**.** *In procedure* QUERY *(Algorithm 12), it takes* $O(\mathcal{T}_{\text{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}))$ *time to compute*

$$\gamma^{\text{tmp}} \leftarrow U^{\text{tmp}} \underbrace{(C^{-1} + U^\top U^{\text{tmp}})^{-1}}_{N} U^\top \cdot (\gamma_1 + \partial\gamma).$$

Proof. First note that from Lemma E.7 we have that the sizes of U' and U are $6n^a \times 3|\partial S|$, and the size of C is $3|\partial S| \times 3|\partial S|$. From the assignment of U^{tmp} on Line 15, we know that the size of U^{tmp} is also $6n^a \times 3|\partial S|$. The running time of this step can be split into the following parts:

The first part is to compute the $3|\partial S| \times 3|\partial S|$ matrix N.

1. Computing the inverse of a $3|\partial S| \times 3|\partial S|$ matrix C takes $O(n^{\widetilde{a}\omega})$ time (Part 6 of Fact E.2).
2. We multiply a $3|\partial S| \times 6n^a$ rectangular matrix $U^\top$ with a $6n^a \times 3|\partial S|$ matrix U^{tmp}, which takes $\mathcal{T}_{\text{mat}}(3|\partial S|, 6n^a, 3|\partial S|) = O(\mathcal{T}_{\text{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}))$ time ($|\partial S| \le n^{\widetilde{a}}$ from Part 6 of Fact E.2).
3. We add a $3|\partial S| \times 3|\partial S|$ matrix C^{-1} with a $3|\partial S| \times 3|\partial S|$ matrix $U^\top U^{\text{tmp}}$ and calculate its inverse. It takes $O(|\partial S|^\omega) = O(n^{\widetilde{a}\omega})$ time (Part 6 of Fact E.2).

Thus in total computing N takes time $O(\mathcal{T}_{\text{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}) + n^{\widetilde{a}\omega}) = O(\mathcal{T}_{\text{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}))$, sinnce $\widetilde{a} \le a$ and thus $n^{\widetilde{a}\omega} = \mathcal{T}_{\text{mat}}(n^{\widetilde{a}}, n^{\widetilde{a}}, n^{\widetilde{a}}) \le \mathcal{T}_{\text{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}})$.

The second part is to compute the $6n^a \times 1$ vector γ^{tmp}.

1. We multiply the $3|\partial S| \times 6n^a$ matrix $U^\top$ with a $6n^a \times 1$ vector $(\gamma_1 + \partial\gamma)$. This takes $O(n^a \cdot |\partial S|) = O(n^{a+\widetilde{a}})$ time (Part 6 of Fact E.2).

2. We multiply the $3|\partial S| \times 3|\partial S|$ matrix N with a $3|\partial S| \times 1$ vector $U^\top(\gamma_1 + \partial\gamma)$. This takes $O(|\partial S|^2) = O(n^{2\widetilde{a}})$ time (Part 6 of Fact E.2).

3. We multiply the $6n^a \times 3|\partial S|$ matrix U^{tmp} with a $3|\partial S| \times 1$ vector $NU^\top(\gamma_1 + \partial\gamma)$. This takes $O(n^a \cdot |\partial S|) = O(n^{a+\widetilde{a}})$ time (Part 6 of Fact E.2).

Thus this part takes $O(n^{a+\widetilde{a}})$ time since $\widetilde{a} \leq a$.

Thus the total time to compute γ^{tmp} is

$$O(\mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}) + n^{a+\widetilde{a}}) = O(\mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}})).$$

□

Claim E.10 (Part 7 of Lemma E.3)**.** *In procedure* QUERY *(Algorithm 12), it takes* $O(n^{\widetilde{a}+a} + n^{1+b})$ *time to compute*

$$r_4 \leftarrow \left(\mathcal{L}_c[(Q[j])_{S^{\mathrm{new}}}] + F[j] + R[j]\Gamma \cdot (\mathcal{L}_c[M_{\partial S \setminus S}] - \mathcal{L}_c[M_{S'}]) + R[j]\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}]\right) \cdot (\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma).$$

Proof. The running time can be split into the following parts:

1. We multiply a $n^{\widetilde{a}}$-sparse $n \times n$ diagonal matrix $\partial\Gamma$ (Part 2 of Fact E.2) with a $n \times 6n^a$ matrix $\mathcal{L}_c[M_{S^{\mathrm{new}}}]$ with a $6n^a \times 1$ vector $(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)$. It takes $O(n^{\widetilde{a}+a})$ time.

2. We multiply a $n^b \times n$ matrix $R[j]$ with a $n \times 1$ vector $(\partial\Gamma\mathcal{L}_c[M_{S^{\mathrm{new}}}](\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma))$. It takes $O(n^{1+b})$ time.

3. We multiply a n^a-sparse $n \times n$ diagonal matrix Γ (Part 3 of Fact E.2) with a $n \times 6n^a$ matrix $\mathcal{L}_c[M_{\partial S \setminus S}]$ that only has $n^{\widetilde{a}}$ non-zero columns (since $|\partial S \setminus S| \leq |\partial S| \leq n^{\widetilde{a}}$ by Part 6 of Fact E.2) and then with a $6n^a \times 1$ vector $(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)$. It takes $O(n^{a+\widetilde{a}})$ time.

4. We multiply a $n^b \times n$ matrix $R[j]$ with a $n \times 1$ vector $(\Gamma(\mathcal{L}_c[M_{\partial S \setminus S}] - \mathcal{L}_c[M_{S'}])(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma))$. It takes $O(n^{1+b})$ time.

5. We multiply a $n^b \times 6n^a$ matrix $F[j]$ with a $6n^a \times 1$ vector $(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)$. It takes $O(n^{1+b})$ time.

6. We multiply a $n^b \times 6n^a$ matrix $\mathcal{L}_c[(Q[j])_{S^{\mathrm{new}}}]$ with a $6n^a \times 1$ vector $(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)$. It takes $O(n^{b+a})$ time.

Thus this part takes $O(n^{\widetilde{a}+a} + n^{1+b} + n^{b+a}) = O(n^{\widetilde{a}+a} + n^{1+b})$ time, since $a \leq 1$.

Thus, the total running time to compute r_4 is $O(n^{\widetilde{a}+a} + n^{1+b})$. □

Proof of Lemma E.3. Summing over the time to compute r_2, r_3, (U', C, U), $\partial\gamma$, $r_{4,1}$, $r_{4,2}$, $r_{4,3}$, U^{tmp}, and $r_{4,4}$, the total running time of computing r is

$$O(n^{1+b} + n^{a+\widetilde{a}} + n^{a+b} + \mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}})) = O(n^{1+b} + \mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}})),$$

which follows by $a \leq 1$ and $n^{a+\widetilde{a}} \leq \mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}))$. □

E.3 Running time of MatrixUpdate

The goal of this section is to prove Lemma E.12. We will use the following sparsity guarantees that are proved in Lemma E.1.

Fact E.11 (Sparsity guarantees for MatrixUpdate)**.** *When entering* MatrixUpdate *we have the following sparsity guarantee (from Table 13):*

1. $|S^{\mathrm{new}}| \leq |S \cup \partial S| \leq 3k$,
2. $\|\Gamma^{\mathrm{new}}\|_0 = \|\Gamma + \partial\Gamma\|_0 = k$.

Lemma E.12 (Running time of MatrixUpdate)**.** *In the procedure* MatrixUpdate *(in Algorithm 13) it takes*

1. $O(k^{\omega+o(1)} + \mathcal{T}_{\mathrm{mat}}(n,k,n))$ *time to compute*
$$M^{\mathrm{tmp}} \leftarrow M - M_{S^{\mathrm{new}}} \cdot ((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}(M_{S^{\mathrm{new}}})^\top,$$
2. $O(k^{\omega+o(1)} + \mathcal{T}_{\mathrm{mat}}(n^{1+o(1)},k,n))$ *time to compute*
$$Q^{\mathrm{tmp}} \leftarrow Q + R(\Gamma^{\mathrm{new}} M^{\mathrm{tmp}}) + R\sqrt{V}(M^{\mathrm{tmp}} - M),$$
3. $O(n^{2+o(1)})$ *time to compute*
$$\beta_1 \leftarrow Q^{\mathrm{tmp}}\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}),$$
4. $O(n^2)$ *time to compute*
$$\beta_2 \leftarrow M^{\mathrm{tmp}}\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}).$$

Further, it takes $O(n)$ time to do all other computation. So the overall running time of the procedure MatrixUpdate *is $O(\mathcal{T}_{\mathrm{mat}}(n^{1+o(1)},k,n))$.*

Claim E.13 (Part 1 of Lemma E.12)**.** *In procedure* MatrixUpdate*, it takes $O(k^{\omega+o(1)}+\mathcal{T}_{\mathrm{mat}}(n,k,n))$ time to compute*
$$M^{\mathrm{tmp}} \leftarrow M - M_{S^{\mathrm{new}}} \cdot ((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}(M_{S^{\mathrm{new}}})^\top.$$

Proof. The running time of computing M^{new} can be split into the following parts:

1. Computing the inverse of a $O(k) \times O(k)$ matrix $((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}})$ (the size follows from $|S^{\mathrm{new}}| = O(k)$, see Part 1 of Fact E.11), this part takes $O(k^{\omega+o(1)})$ time.
2. Computing the multiplication of a $n \times O(k)$ matrix $M_{S^{\mathrm{new}}}$ with a $O(k) \times O(k)$ matrix $((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}$, this part takes $O(\mathcal{T}_{\mathrm{mat}}(n,k,k))$ time.
3. Computing the multiplication of a $n \times O(k)$ matrix $M_{S^{\mathrm{new}}}((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}$ with a $O(k) \times n$ matrix $(M_{S^{\mathrm{new}}})^\top$, this part takes $O(\mathcal{T}_{\mathrm{mat}}(n,k,n))$ time.
4. Subtracting a $n \times n$ matrix $M_{S^{\mathrm{new}}} \cdot ((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}(M_{S^{\mathrm{new}}})^\top$ from the $n \times n$ matrix M, this part takes $O(n^2)$ time.

So in total computing M^{tmp} takes time

$$O(k^{\omega+o(1)} + \mathcal{T}_{\text{mat}}(n,k,k) + \mathcal{T}_{\text{mat}}(n,k,n) + n^2) = O(k^{\omega+o(1)} + \mathcal{T}_{\text{mat}}(n,k,n)).$$

□

Claim E.14 (Part 2 of Lemma E.12)**.** *In* MatrixUpdate*, it takes* $O(k^{\omega+o(1)} + \mathcal{T}_{\text{mat}}(n^{1+o(1)}, k, n))$ *time to compute*

$$Q^{\text{tmp}} \leftarrow Q + \underbrace{R(\Gamma^{\text{new}} M^{\text{tmp}})}_{N_1} + \underbrace{R\sqrt{V}(M^{\text{tmp}} - M)}_{N_2}.$$

Proof. The running time of computing Q^{tmp} can be split into the following parts:

The first part is to compute the $n \times n$ matrix N_1:

1. Multiplying a k-sparse $n \times n$ diagonal matrix Γ^{new} (Part 2 of Fact E.11) with a $n \times n$ matrix M^{tmp} takes $O(kn)$ time.
2. Multiplying a $n^{1+o(1)} \times n$ matrix Q^{tmp} with a k-row-sparse $n \times n$ matrix $(\Gamma^{\text{new}} M^{\text{tmp}})$ (Part 1 of Fact) takes $O(\mathcal{T}_{\text{mat}}(n^{1+o(1)}, k, n))$ time.

So in total computing N_1 takes $O(\mathcal{T}_{\text{mat}}(n^{1+o(1)}, k, n))$ time.

The second part is to compute the $n \times n$ matrix N_2. By the definition of M^{tmp} we have

$$N_2 = R\sqrt{V}(M^{\text{tmp}} - M) = R\sqrt{V} M_{S^{\text{new}}} \cdot \underbrace{((\Delta^{\text{new}}_{S^{\text{new}},S^{\text{new}}})^{-1} + M_{S^{\text{new}},S^{\text{new}}})^{-1} (M_{S^{\text{new}}})^\top}_{N_3}.$$

And we computes N_2 as follows:

1. Multiplying a $n^{1+o(1)} \times n$ matrix R with a $n \times n$ diagonal matrix $\sqrt{V}$ takes $O(n^{2+o(1)})$ time.
2. Multiplying a $n^{1+o(1)} \times n$ matrix $R\sqrt{V}$ with a $n \times O(k)$ matrix $M_{S^{\text{new}}}$ (Part 1 of Fact E.11) takes $O(\mathcal{T}_{\text{mat}}(n^{1+o(1)}, n, k))$ time.
3. The $O(k) \times n$ matrix N_3 is already computed when we were computing M^{tmp} (Claim E.13), and this takes $O(k^{\omega+o(1)} + \mathcal{T}_{\text{mat}}(k,k,n))$ time.
4. Multiplying a $n^{1+o(1)} \times O(k)$ matrix $(R\sqrt{V} M_{S^{\text{new}}})$ with a $O(k) \times n$ matrix N_3 takes $O(\mathcal{T}_{\text{mat}}(n^{1+o(1)}, k, n))$ time.

So in total computing Q^{tmp} takes time

$$O(n^{2+o(1)} + k^{\omega+o(1)} + \mathcal{T}_{\text{mat}}(k,k,n) + \mathcal{T}_{\text{mat}}(n^{1+o(1)}, k, n)) = O(k^{\omega+o(1)} + \mathcal{T}_{\text{mat}}(n^{1+o(1)}, k, n)).$$

□

Claim E.15 (Part 3 of Lemma E.12)**.** *In* MatrixUpdate*, it takes* $O(n^{2+o(1)})$ *time to compute*

$$\beta_1 \leftarrow Q^{\text{tmp}} \sqrt{W^{\text{appr}}} f(h^{\text{appr}}).$$

Proof. To compute β_1, we first multiply a $n^{1+o(1)} \times n$ matrix Q^{tmp} with a $n \times n$ diagonal matrix $\sqrt{W^{\text{appr}}}$, which takes $O(n^{2+o(1)})$ time. Then we multiply a $n^{1+o(1)} \times n$ matrix $Q^{\text{tmp}}\sqrt{W^{\text{appr}}}$ with a $n \times 1$ vector, which takes $O(n^{2+o(1)})$ time. So in total computing β_1 takes time $O(n^{2+o(1)})$. □

Claim E.16 (Part 4 of Lemma E.12)**.** *In procedure* MatrixUpdate*, it takes* $O(n^2)$ *time to compute*

$$\beta_2 \leftarrow M^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}).$$

Proof. To compute β_2, we first multiply a $n \times n$ matrix M^{tmp} with a $n \times n$ diagonal matrix $\sqrt{W^{\mathrm{appr}}}$, which takes $O(n^2)$ time. Then we multiply a $n^1 \times n$ matrix $M^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}}$ with a $n \times 1$ vector, which takes $O(n^2)$ time. So in total computing β_1 takes time $O(n^2)$. □

Proof of Lemma E.12. **Overall time.** The running time of procedure MatrixUpdate is $O(k^{\omega+o(1)} + \mathcal{T}_{\mathrm{mat}}(n^{1+o(1)}, k, n) + n^{2+o(1)}) = O(\mathcal{T}_{\mathrm{mat}}(n^{1+o(1)}, k, n))$. □

E.4 Running time of PartialMatrixUpdate

The goal of this section is to prove Lemma E.18. We will use the following sparsity guarantees that are proved in Lemma E.1.

Fact E.17 (Sparsity guarantees for PartialMatrixUpdate)**.** *When entering* PartialMatrixUpdate *(Algorithm 14) we have the following sparsity guarantee (from Table 12 and Table 13):*

1. $\|\sqrt{W^{\mathrm{appr}}} - \sqrt{V}\|_0 \leq n^a$,
2. $\|\partial\Gamma\|_0 \leq \widetilde{k} \leq 2n^a$,
3. $\|\Gamma + \partial\Gamma\|_0 \leq n^a$,
4. $\|f(\widetilde{g}) - f(g)\|_0 \leq n^a$,
5. $\|\xi\|_0 \leq n^a$,
6. $|S| \leq n^a$,
7. $|\partial S| \leq \widetilde{k} \leq 2n^a$.

Lemma E.18 (Running time of PartialMatrixUpdate)**.** *In the procedure* PartialMatrixUpdate *(Algorithm 14) it takes*

1. $O(\widetilde{k} n^a)$ *time to compute*
$$(U', C, U) \leftarrow \textsc{Decompose}\Big(\mathcal{L}_*[(\Delta^{\mathrm{new}}_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}}, S^{\mathrm{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}]\Big),$$
2. $O(\mathcal{T}_{\mathrm{mat}}(\widetilde{k}, n^a, n^a))$ *time to compute*
$$B^{\mathrm{tmp}} \leftarrow B - BU'(C^{-1} + U^\top BU')^{-1} U^\top B,$$
3. $O(n)$ *time to compute*
$$\xi^{\mathrm{tmp}} \leftarrow \sqrt{W^{\mathrm{appr}}} f(\widetilde{g}) - \sqrt{V} f(g),$$
4. $O(n^{2a})$ *time to compute*
$$\gamma_1^{\mathrm{tmp}} \leftarrow B^{\mathrm{tmp}} \cdot \mathcal{L}_r[\beta_{2, S^{\mathrm{new}}}] + B^{\mathrm{tmp}} \cdot \mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top] \xi^{\mathrm{tmp}},$$
5. $O(\widetilde{k} n^a)$ *time to compute*
$$\gamma_2^{\mathrm{tmp}} \leftarrow \gamma_2 + (\Gamma + \partial\Gamma) M(\sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}) f(\widetilde{g}) + \partial\Gamma M(\sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g)).$$
6. $O(\mathcal{T}_{\mathrm{mat}}(n^{1+o(1)}, n^a, \widetilde{k}))$ *time to compute*
$$F^{\mathrm{tmp}} \leftarrow F + R\Gamma \cdot (\mathcal{L}_c[M_{\partial S \setminus S}] - \mathcal{L}_c[M_{S'}]) + R\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}].$$

7. $O(\mathcal{T}_{\mathrm{mat}}(n, n^a, \widetilde{k}))$ *time to compute*

$$E^{\mathrm{tmp}} = E + B^{\mathrm{tmp}}(\mathcal{L}_r[(M_{\partial S \setminus S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) - BU'(C^{-1} + U^\top BU')^{-1}U^\top E.$$

Further, it takes $O(n)$ time to do all other computation. So the overall running time of the procedure PartialMatrixUpdate *is $O(\mathcal{T}_{\mathrm{mat}}(n^{1+o(1)}, n^a, \widetilde{k}))$.*

Claim E.19 (Part 1 of Lemma E.18)**.** *In the procedure* PartialMatrixUpdate *(Algorithm 14), it takes $O(\widetilde{k}n^a)$ time to compute*

$$(U', C, U) \leftarrow \textsc{Decompose}\Big(\mathcal{L}_*[(\Delta^{\mathrm{new}}_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}}, S^{\mathrm{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}]\Big).$$

And the size of the computed matrices are $U', U \in \mathbb{R}^{6n^a \times c}$, $C \in \mathbb{R}^{c \times c}$, where $c = O(\widetilde{k})$.

Proof. The analysis for the Decompose function in PartialMatrixUpdate is the same as the analysis in Query (Claim E.7). We still have $|S| \leq n^a$ (Part 6 of Fact E.17), but the bound on ∂S is different and now we have $|\partial S| = \widetilde{k} \leq 2n^a$ (Part 7 of Fact E.17). So $c = O(|\partial S|) = O(\widetilde{k})$.

And to compute (U', C, U) it takes $O(\widetilde{k}n^a)$ time. □

Claim E.20 (Part 2 of Lemma E.18)**.** *In the procedure* PartialMatrixUpdate *(Algorithm 14), it takes $O(\mathcal{T}_{\mathrm{mat}}(\widetilde{k}, n^a, n^a))$ time to compute*

$$B^{\mathrm{tmp}} \leftarrow B - BU' \underbrace{(C^{-1} + U^\top BU')^{-1}}_{N} U^\top B.$$

Proof. The running time of computing $B^{\mathrm{tmp}} \in \mathbb{R}^{6n^a \times 6n^a}$ can be split into the following parts.

The first part is to compute $N \in \mathbb{R}^{c \times c}$ as follows:

1. Multiplying a $c \times 6n^a$ matrix $U^\top$ with a $6n^a \times 6n^a$ matrix B takes $O(\mathcal{T}_{\mathrm{mat}}(c, n^a, n^a))$ time.
2. Multiplying a $c \times 6n^a$ matrix $(U^\top B)$ with a $6n^a \times c$ matrix U' takes $O(\mathcal{T}_{\mathrm{mat}}(c, n^a, c))$ time.
3. Computing the inverse of a $c \times c$ matrix $(C^{-1} + U^\top BU')$ takes $O(c^{\omega+o(1)})$ time.

So the total running time to compute N is $O(\mathcal{T}_{\mathrm{mat}}(c, n^a, n^a) + \mathcal{T}_{\mathrm{mat}}(c, n^a, c) + c^{\omega+o(1)}) = O(\mathcal{T}_{\mathrm{mat}}(\widetilde{k}, n^a, n^a))$ (since $c = O(\widetilde{k}) \leq O(n^a)$ by Claim E.19), and $\widetilde{k} \leq 2n^a$.

The second part is to compute $BU'NU^\top B \in \mathbb{R}^{6n^a \times 6n^a}$ as follows:

1. Multiplying a $6n^a \times 6n^a$ matrix B with a $6n^a \times c$ matrix U' takes $O(\mathcal{T}_{\mathrm{mat}}(n^a, n^a, c))$ time.
2. Multiplying a $c \times 6n^a$ matrix $U^\top$ with a $6n^a \times 6n^a$ matrix B takes $O(\mathcal{T}_{\mathrm{mat}}(c, n^a, n^a))$ time.
3. Multiplying a $6n^a \times c$ matrix BU' with a $c \times c$ matrix N takes $O(\mathcal{T}_{\mathrm{mat}}(n^a, c, c))$ time.
4. Multiplying a $6n^a \times c$ matrix $BU'N$ with a $c \times 6n^a$ matrix $U^\top B$ takes $O(\mathcal{T}_{\mathrm{mat}}(n^a, c, n^a))$ time.

So the total running time to compute $BU'NU^\top B$ is $O(\mathcal{T}_{\mathrm{mat}}(n^a, n^a, c) + \mathcal{T}_{\mathrm{mat}}(n^a, c, c)) = O(\mathcal{T}_{\mathrm{mat}}(n^a, n^a, \widetilde{k}))$ (since $c = O(\widetilde{k}) \leq O(n^a)$ by Claim E.19), and $\widetilde{k} \leq 2n^a$.

Finally, subtracting $BU'NU^\top B$ from B takes $O(n^{2a})$ time since both matrices has size $6n^a \times 6n^a$.

So in total computing $B^{\mathrm{tmp}} \in \mathbb{R}^{6n^a \times 6n^a}$ takes time $O(\mathcal{T}_{\mathrm{mat}}(\widetilde{k}, n^a, n^a) + n^{2a}) = O(\mathcal{T}_{\mathrm{mat}}(\widetilde{k}, n^a, n^a))$. □

Claim E.21 (Part 3 of Lemma E.18)**.** *In the procedure* PartialMatrixUpdate *(Algorithm 14), it takes $O(n)$ time to compute $\xi^{\mathrm{tmp}} \leftarrow \sqrt{W^{\mathrm{appr}}} f(\widetilde{g}) - \sqrt{V} f(g)$. And the computed vector $\xi^{\mathrm{tmp}} \in \mathbb{R}^{n \times 1}$ is n^a-sparse.*

Proof. The $O(n)$ running time follows trivially from the fact that $\sqrt{W^{\mathrm{appr}}}$ and $\sqrt{V}$ are $n \times n$ diagonal matrices, and $f(\widetilde{g})$ and $f(g)$ are $n \times 1$ vectors. We also have

$$\|\xi^{\mathrm{tmp}}\|_0 = \|\sqrt{W^{\mathrm{appr}}} f(\widetilde{g}) - \sqrt{V} f(g)\|_0 = \max\{\|\sqrt{W^{\mathrm{appr}}} - \sqrt{V}\|_0, \|f(\widetilde{g}) - f(g)\|_0\} \leq n^a,$$

which follows from Part 1 and 4 of Fact E.17. □

Claim E.22 (Part 4 of Lemma E.18)**.** *In the procedure* PartialMatrixUpdate *(Algorithm 14), it takes $O(n^{2a})$ time to compute $\gamma_1^{\mathrm{tmp}} \leftarrow B^{\mathrm{tmp}} \cdot \mathcal{L}_r[\beta_{2,S^{\mathrm{new}}}] + B^{\mathrm{tmp}} \cdot \mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top] \cdot \xi^{\mathrm{tmp}}$.*

Proof. The running time of computing γ_1^{tmp} can be split into the following parts:

1. Multiplying a $6n^a \times 6n^a$ matrix B^{tmp} with a $6n^a \times 1$ vector $\mathcal{L}_r[\beta_{2,S^{\mathrm{new}}}]$ takes $O(n^{2a})$ time.
2. Multiplying a $6n^a \times n$ matrix $\mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top]$ with a n^a-sparse $n \times 1$ vector ξ^{tmp} (Claim E.21) takes $O(n^{2a})$ time.
3. Multiplying a $6n^a \times 6n^a$ matrix B^{tmp} with a $6n^a \times 1$ vector $\mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top]\xi^{\mathrm{tmp}}$ takes $O(n^{2a})$ time.

So in total computing γ_1^{tmp} takes $O(n^{2a})$ time. □

Claim E.23 (Part 5 of Lemma E.18)**.** *In the procedure* PartialMatrixUpdate *(Algorithm 14), it takes $O(\widetilde{k} n^a)$ time to compute*

$$\gamma_2^{\mathrm{tmp}} \leftarrow \gamma_2 + (\Gamma + \partial\Gamma) M \underbrace{(\sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}) f(\widetilde{g})}_{a_1} + \partial\Gamma M \underbrace{(\sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g))}_{a_2}.$$

Proof. First note that the $n \times 1$ vectors a_1 and a_2 can both be computed in $O(n)$ time. Also,

$$\|a_1\|_0 = \|(\sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}) f(\widetilde{g})\|_0 \leq \min\{\|\sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}}\|_0, \|f(\widetilde{g})\|_0\} \leq \widetilde{k},$$

where the last step follows from Part 1 of Fact E.17. And

$$\|a_2\|_0 = \|\sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g)\|_0 = \|\xi\|_0 \leq n^a,$$

where the last step follows from Part 5 of Fact E.17.

The running time of computing γ_2^{tmp} can be split into the following parts:

1. Multiplying a n^a-sparse $n \times n$ diagonal matrix $(\Gamma + \partial\Gamma)$ (Part 3 of Fact E.17) with a $n \times n$ matrix M and then with a $\widetilde{k}$-sparse $n \times 1$ vector a_1 takes $O(\widetilde{k} n^a)$ time.
2. Multiplying a $\widetilde{k}$-sparse $n \times n$ diagonal matrix $\partial\Gamma$ (Part 2 of Fact E.17) with a $n \times n$ matrix M and then with a n^a-sparse $n \times 1$ vector a_2 takes $O(\widetilde{k} n^a)$ time.

So in total computing γ_2^{tmp} takes time $O(\widetilde{k} n^a)$. □

Claim E.24 (Part 6 of Lemma E.18)**.** *In the procedure* PartialMatrixUpdate *(Algorithm 14), it takes $O(\mathcal{T}_{\mathrm{mat}}(n^{1+o(1)}, n^a, \widetilde{k}))$ time to compute*

$$F^{\mathrm{tmp}} \leftarrow F + R\Gamma \cdot (\mathcal{L}_c[M_{\partial S \setminus S}] - \mathcal{L}_c[M_{S'}]) + R\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}].$$

Proof. By sparsity guarantee(Part 7, 6 of Fact E.17), we have $|\partial S| + |S'| \leq |\partial S| \leq \widetilde{k} \leq 2n^a$. The running time of computing F^{tmp} can be split into the following parts:

1. Multiplying a $n^{1+o(1)} \times n$ matrix R with a n^a-sparse $n \times n$ diagonal matrix Γ and then with a $\widetilde{k}$-column-sparse $n \times 6n^a$ matrix $\mathcal{L}_c[M_{\partial S \setminus S}] - \mathcal{L}_c[M_{S'}]$ takes $O(\mathcal{T}_{\mathrm{mat}}(n^{1+o(1)}, n^a, \widetilde{k}))$ time.

2. Multiplying a $n^{1+o(1)} \times n$ matrix R with a $\widetilde{k}$-sparse $n \times n$ diagonal matrix $\partial\Gamma$ and then with a $n \times 6n^a$ matrix $\mathcal{L}_c[M_{S^{\mathrm{new}}}]$ takes $O(\mathcal{T}_{\mathrm{mat}}(n^{1+o(1)}, \widetilde{k}, n^a)$ time.

3. Adding two matrices $R\Gamma(\mathcal{L}_c[M_{\partial S \setminus S}] - \mathcal{L}_c[M_{S'}])$, $R\partial\Gamma\mathcal{L}_c[M_{S^{\mathrm{new}}}]$ on the current stored matrix F takes $O(n^{1+o(1)+a})$ time.

So in total computing F^{tmp} takes $O(\mathcal{T}_{\mathrm{mat}}(n^{1+o(1)}, n^a, \widetilde{k}) + n^{1+o(1)+a}) = O(\mathcal{T}_{\mathrm{mat}}(n^{1+o(1)}, n^a, \widetilde{k}))$ time since $O(n^{1+o(1)+a}) \leq O(\mathcal{T}_{\mathrm{mat}}(n^{1+o(1)}, n^a, \widetilde{k}))$. □

Claim E.25 (Part 7 of Lemma E.18)**.** *In the procedure* PartialMatrixUpdate *(Algorithm 14), it takes $O(\mathcal{T}_{\mathrm{mat}}(n, n^a, \widetilde{k}))$ time to compute*

$$E^{\mathrm{tmp}} \leftarrow E + \underbrace{B^{\mathrm{tmp}}(\mathcal{L}_r[(M_{\partial S \setminus S})^\top] - \mathcal{L}_r[(M_{S'})^\top])}_{N_2} - \underbrace{BU' \underbrace{(C^{-1} + U^\top BU')^{-1}}_{N} U^\top E}_{N_1}$$

Proof. By sparsity guarantee(Part 7, 6 of Fact E.17), we have $|\partial S| \leq \widetilde{k} \leq 2n^a$, $|S| \leq n^a$. So $|S^{\mathrm{new}}| \leq |S \cup \partial S| \leq |S| + |\partial S| \leq 3n^a$. The running time of computing F^{tmp} can be split into the following parts:

First we compute $N_1 \in \mathbb{R}^{6n^a \times n}$.

1. We already compute the time of computing N in Claim E.20. It takes $O(\mathcal{T}_{\mathrm{mat}}(c, n^a, n^a))$ time.

2. Multiplying a $6n^a \times 6n^a$ matrix B with a $6n^a \times c$ matrix U' takes $O(\mathcal{T}_{\mathrm{mat}}(n^a, n^a, c))$ time.

3. Multiplying a $6n^a \times c$ matrix BU' with a $c \times c$ matrix N takes $O(\mathcal{T}_{\mathrm{mat}}(n^a, c, c))$ time.

4. Multiplying a $c \times 6n^a$ matrix $U^\top$ with a $6n^a \times n$ matrix E takes $O(\mathcal{T}_{\mathrm{mat}}(c, n^a, n))$ time.

5. Multiplying a $6n^a \times c$ matrix $BU'N$ with a $c \times n$ matrix $U^\top E$ takes time $O(\mathcal{T}_{\mathrm{mat}}(n^a, c, n))$.

Next, we compute $N_2 \in \mathbb{R}^{6n^a \times n}$. Multiplying a $6n^a \times 6n^a$ matrix B^{tmp} with a $\widetilde{k}$-row-sparse $6n^a \times n$ matrix $(\mathcal{L}_r[(M_{\partial S \setminus S})^\top] - \mathcal{L}_r[(M_{S'})^\top])$ takes $O(\mathcal{T}_{\mathrm{mat}}(n^a, \widetilde{k}, n)$ time.

So in total computing E^{tmp} takes time

$$\begin{aligned} & O(\mathcal{T}_{\mathrm{mat}}(c, n^a, n^a) + \mathcal{T}_{\mathrm{mat}}(n^a, n^a, c) + \mathcal{T}_{\mathrm{mat}}(n^a, c, c) + \mathcal{T}_{\mathrm{mat}}(c, n^a, n) + \mathcal{T}_{\mathrm{mat}}(n^a, c, n)) \\ = {} & O(\mathcal{T}_{\mathrm{mat}}(n, n^a, c)) = O(\mathcal{T}_{\mathrm{mat}}(n, n^a, \widetilde{k})), \end{aligned}$$

where the first step follows since $c = O(\widetilde{k}) \leq O(n^a)$ by Claim E.19), and $\widetilde{k} \leq 2n^a$. □

Proof of Lemma E.18. **Overall time.** So in total the running time of procedure PartialMatrixUpdate (Algorithm 14) is

$$O(\widetilde{k}n^a + \mathcal{T}_{\mathrm{mat}}(\widetilde{k}, n^a, n^a) + n + n^{2a} + \mathcal{T}_{\mathrm{mat}}(n, n^a, \widetilde{k})) = O(\mathcal{T}_{\mathrm{mat}}(n, n^a, \widetilde{k})).$$

□

E.5 Running time of VectorUpdate

The goal of this section is to prove Lemma E.27. We will use the following sparsity guarantees that are proved in Lemma E.1.

Fact E.26 (Sparsity guarantees for VectorUpdate)**.** *When entering* VectorUpdate *(Algorithm 15) we have the following sparsity guarantee (from Table 12 and Table 13):*

1. $\|\Gamma\|_0 = \|\sqrt{\widetilde{V}} - \sqrt{V}\|_0 \leq n^a$,
2. $\|f(h^{\mathrm{appr}}) - f(g)\|_0 = p$.

Lemma E.27 (Running time of VectorUpdate)**.** *The procedure* VectorUpdate *(Algorithm 15) takes*

1. $O(pn^{1+o(1)})$ *time to compute*
$$\beta_1^{\mathrm{tmp}} \leftarrow \beta_1 + Q\sqrt{V}(f(h^{\mathrm{appr}}) - f(g)), \quad \beta_2^{\mathrm{tmp}} \leftarrow \beta_2 + M\sqrt{V}(f(h^{\mathrm{appr}}) - f(g)),$$
2. $O(n)$ *time to compute*
$$\xi^{\mathrm{tmp}} \leftarrow (\sqrt{\widetilde{V}} - \sqrt{V})f(h^{\mathrm{appr}}),$$
3. $O(n^{2a})$ *time to compute*
$$\gamma_1^{\mathrm{tmp}} \leftarrow B \cdot \mathcal{L}_r[\beta_{2,S}^{\mathrm{tmp}}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi^{\mathrm{tmp}},$$
4. $O(n^{2a})$ *time to compute*
$$\gamma_2^{\mathrm{tmp}} \leftarrow \Gamma M \cdot \xi^{\mathrm{tmp}}.$$

Overall, the running time of procedure VectorUpdate *is*
$$O(n^{2a} + pn^{1+o(1)}).$$

Claim E.28 (Part 1 of Lemma E.27)**.** *In the procedure* VectorUpdate *(Algorithm 15), it takes $O(pn^{1+o(1)})$ time to compute*
$$\beta_1^{\mathrm{tmp}} \leftarrow \beta_1 + Q\sqrt{V}(f(h^{\mathrm{appr}}) - f(g)), \quad \beta_2^{\mathrm{tmp}} \leftarrow \beta_2 + M\sqrt{V}(f(h^{\mathrm{appr}}) - f(g)).$$

Proof. We compute β_1^{tmp} as follows.

1. Multiplying a $n \times n$ diagonal matrix $\sqrt{V}$ with a p-sparse vector $f(h^{\mathrm{appr}}) - f(g)$ (Part 2 of Fact E.26) takes $O(p)$ time, and the resulting vector $\sqrt{V}(f(h^{\mathrm{appr}}) - f(g))$ is also p-sparse.
2. Multiplying a $n^{1+o(1)} \times n$ matrix Q with a p-sparse vector $\sqrt{V}(f(h^{\mathrm{appr}}) - f(g))$ takes $O(pn^{1+o(1)})$ time.

So the total running time to compute β_1^{tmp} is $O(pn^{1+o(1)})$.

Following the same reason and note that M has size $n \times n$, we can compute β_2^{tmp} in $O(pn) \leq O(pn^{1+o(1)})$ time. □

Claim E.29 (Part 2 of Lemma E.27)**.** *In the procedure* VectorUpdate *(Algorithm 15), it takes $O(n)$ time to compute $\xi^{\mathrm{tmp}} \leftarrow (\sqrt{\widetilde{V}} - \sqrt{V})f(h^{\mathrm{appr}})$. And the computed $n \times 1$ vector ξ^{tmp} is n^a-sparse.*

Proof. ξ^{tmp} is computed by multiplying a $n \times n$ diagonal matrix $(\sqrt{\widetilde{V}} - \sqrt{V})$ with a $n \times 1$ vector $f(h^{\mathrm{appr}})$, and this takes $O(n)$ time. We also have

$$\|\xi^{\mathrm{tmp}}\|_0 = \|(\sqrt{\widetilde{V}} - \sqrt{V}) f(h^{\mathrm{appr}})\|_0 \leq \min\{\|\sqrt{\widetilde{V}} - \sqrt{V}\|_0, \|f(h^{\mathrm{appr}})\|_0\} \leq n^a,$$

where the last step follows from Part 1 of Fact E.26. □

Claim E.30 (Part 3 of Lemma E.27)**.** *In the procedure* VectorUpdate *(Algorithm 15), it takes* $O(n^{2a})$ *time to compute* $\gamma_1^{\mathrm{tmp}} \leftarrow B \cdot \mathcal{L}_r[\beta_{2,S}^{\mathrm{tmp}}] + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \xi^{\mathrm{tmp}}$.

Proof. The running time of computing γ_1^{tmp} can be split into the following parts:

1. Multiplying a $6n^a \times 6n^a$ matrix B with a $6n^a \times 1$ vector $\mathcal{L}_r[\beta_{2,S}^{\mathrm{tmp}}]$ takes $O(n^{2a})$ time.
2. Multiplying a $6n^a \times n$ matrix $\mathcal{L}_r[(M_S)^\top]$ with a n^a-sparse $n \times 1$ vector ξ^{tmp} (Claim E.29) takes $O(n^{2a})$ time.
3. Multiplying a $6n^a \times 6n^a$ matrix B with a $6n^a \times 1$ vector $\mathcal{L}_r[(M_S)^\top]\xi^{\mathrm{tmp}}$ takes $O(n^{2a})$ time.

The total running time to compute γ_1^{tmp} is $O(n^{2a})$. □

Claim E.31 (Part 4 of Lemma E.27)**.** *In the procedure* VectorUpdate *(Algorithm 15), it takes* $O(n^{2a})$ *time to compute* $\gamma_2^{\mathrm{tmp}} \leftarrow \Gamma M \cdot \xi^{\mathrm{tmp}}$.

Proof. We compute $\gamma_2^{\mathrm{tmp}} \in \mathbb{R}^n$ by multiplying a n^a-sparse $n \times n$ diagonal matrix Γ (Part 1 of Fact E.26) with a $n \times n$ matrix M and then with a n^a-sparse $n \times 1$ vector ξ^{tmp} ((Claim E.29)), which takes $O(n^{2a})$ time. □

Proof of Lemma E.27. Overall, the total running time to compute γ_1^{tmp} is $O(pn^{1+o(1)} + n^{2a}) = O(pn^{1+o(1)})$, since we always have $p \geq n^a$. □

E.6 Running time of PartialVectorUpdate

The goal of this section is to prove Lemma E.33. We will use the following sparsity guarantees that are proved in Lemma E.1.

Fact E.32 (Sparsity guarantees for PartialVectorUpdate)**.** *When entering* PartialVectorUpdate *(Algorithm 16) we have the following sparsity guarantee (from Table 12 and Table 13):*

1. $\|\Gamma\|_0 \leq n^a$,
2. $\|f(h^{\mathrm{appr}}) - f(\widetilde{g})\|_0 = \widetilde{p} \leq 2n^a$.

Lemma E.33 (Running time of PartialVectorUpdate)**.** *In the procedure* PartialVectorUpdate *(in Algorithm 16) it takes*

1. $O(n)$ *time to compute*

$$\xi^{\mathrm{tmp}} \leftarrow \sqrt{\widetilde{V}} f(h^{\mathrm{appr}}) - \sqrt{V} f(g),$$

2. $O(n^{2a} + \widetilde{p}n^a)$ *time to compute*

$$\gamma_1^{\mathrm{tmp}} \leftarrow \gamma_1 + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \sqrt{\widetilde{V}}\big(f(h^{\mathrm{appr}}) - f(\widetilde{g})\big),$$

3. $O(\widetilde{p}n^a)$ time to compute

$$\gamma_2^{\mathrm{tmp}} \leftarrow \gamma_2 + \Gamma M \sqrt{\widetilde{V}}\big(f(h^{\mathrm{appr}}) - f(\widetilde{g})\big).$$

Overall, the procedure PartialVectorUpdate *takes $O(n^{2a} + \widetilde{p}n^a)$ time.*

Claim E.34 (Part 1 of Lemma E.33)**.** *In the procedure* PartialVectorUpdate *(Algorithm 16), it takes $O(n)$ time to compute $\xi^{\mathrm{tmp}} \leftarrow \sqrt{\widetilde{V}} f(h^{\mathrm{appr}}) - \sqrt{V} f(g)$.*

Proof. ξ^{tmp} is computed by multiplying the $n \times n$ diagonal matrices $\sqrt{\widetilde{V}}$ and $\sqrt{V}$ with the $n \times 1$ vectors $f(h^{\mathrm{appr}})$ and $f(g)$ respectively, and this takes $O(n)$ time. □

Claim E.35 (Part 2 of Lemma E.33)**.** *In the procedure* PartialVectorUpdate *(Algorithm 16), it takes $O(n^{2a} + \widetilde{p}n^a)$ time to compute $\gamma_1^{\mathrm{tmp}} \leftarrow \gamma_1 + B \cdot \mathcal{L}_r[(M_S)^\top] \cdot \sqrt{\widetilde{V}}\big(f(h^{\mathrm{appr}}) - f(\widetilde{g})\big)$.*

Proof. The running time of computing γ_1^{tmp} can be split into the following parts:

1. Multiplying a $n \times n$ diagonal matrix $\sqrt{\widetilde{V}}$ with a $\widetilde{p}$-sparse $n \times 1$ vector $(f(h^{\mathrm{appr}}) - f(\widetilde{g}))$ (Part 2 of Fact E.32) takes $O(\widetilde{p})$ time, and the resulting vector $\sqrt{\widetilde{V}}(f(h^{\mathrm{appr}}) - f(\widetilde{g}))$ is also $\widetilde{p}$-sparse.
2. Multiplying a $6n^a \times n$ matrix $\mathcal{L}_r[(M_S)^\top]$ with a $\widetilde{p}$-sparse $n \times 1$ vector $\sqrt{\widetilde{V}}(f(h^{\mathrm{appr}}) - f(\widetilde{g}))$ takes $O(\widetilde{p}n^a)$ time.
3. Multiplying a $6n^a \times 6n^a$ matrix B with a $6n^a \times 1$ vector $\mathcal{L}_r[(M_S)^\top]\sqrt{\widetilde{V}}(f(h^{\mathrm{appr}}) - f(\widetilde{g}))$ takes $O(n^{2a})$ time.
4. Adding two $n \times 1$ vectors together takes $O(n)$ time.

So in total the running time to compute γ_1^{tmp} is $O(n^{2a} + \widetilde{p}n^a)$. □

Claim E.36 (Part 3 of Lemma E.33)**.** *In the procedure* PartialVectorUpdate *(Algorithm 16), it takes $O(\widetilde{p}n^a)$ time to compute $\gamma_2^{\mathrm{tmp}} \leftarrow \gamma_2 + \Gamma M \sqrt{\widetilde{V}}\big(f(h^{\mathrm{appr}}) - f(\widetilde{g})\big)$.*

Proof. The running time of computing γ_2^{tmp} can be split into the following parts:

1. Multiplying a $n \times n$ diagonal matrix $\sqrt{\widetilde{V}}$ with a $\widetilde{p}$-sparse $n \times 1$ vector $(f(h^{\mathrm{appr}}) - f(\widetilde{g}))$ (Part 2 of Fact E.32) takes $O(\widetilde{p})$ time, and the resulting vector $\sqrt{\widetilde{V}}(f(h^{\mathrm{appr}}) - f(\widetilde{g}))$ is also $\widetilde{p}$-sparse.
2. Multiplying a n^a-sparse $n \times n$ diagonal matrix Γ (Part 1 of Fact E.32) with a $n \times n$ matrix M and then with a $\widetilde{p}$-sparse $n \times 1$ vector $\sqrt{\widetilde{V}}(f(h^{\mathrm{appr}}) - f(\widetilde{g}))$ takes $O(\widetilde{p}n^a)$ time.
3. Adding two $n \times 1$ vectors together takes $O(n)$ time.

So in total the running time to compute γ_2^{tmp} is $O(\widetilde{p}n^a)$. □

Proof of Lemma E.33. Overall, the running time of PartialVectorUpdate (Algorithm 16) is $O(n^{2a} + \widetilde{p}n^a)$. □

E.7 Running time of Initialize

The goal of this section is to prove Lemma E.37.

Lemma E.37 (Running time of Initialize)**.** *The procedure* Initialize *(in Algorithm 5) takes* $O(n^{\omega+o(1)})$ *time.*

Proof. The bottleneck in Initialize is to compute $M = A^\top (AVA^\top)^{-1} A$ and $Q = R\sqrt{V}M$. Other operations take at most $O(n^{2+o(1)})$ time.

Computing M involves matrix multiplication of two $n \times n$ matrix, and inversion of a $n \times n$ matrix. Both of them can be done in $O(n^{\omega+o(1)})$ time.

Computing Q involves matrix multiplication of $n^{1+o(1)} \times n$ matrix with a $n \times n$ matrix. This can be done in $O(n^{\omega+o(1)})$ time.

Therefore, the total running time is $O(n^{\omega+o(1)})$. □

F Data structure : amortized time

In this section we provide an amortized analysis of the four procedures MatrixUpdate, PartialMatrixUpdate, VectorUpdate, and PartialVectorUpdate. Our amortized analysis builds upon the amortized analysis of [CLS19].

F.1 Definitions and Preliminaries

Our algorithm makes T calls to the procedure UpdateQuery. For clarity, we define some notations with superscripts that represent the number of iterations. For the input diagonal matrix W and its approximations V and $\widetilde{V}$, we define the following notations:

Definition F.1 (Definitions of sequences $\{w^{(j)}\}_{j=0}^T$, $\{v^{(j)}\}_{j=0}^T$ and $\{\widetilde{v}^{(j)}\}_{j=0}^T$)**.** *When initializing, we use* $v^{(0)}$ *and* $\widetilde{v}^{(0)}$ *to denote the initial values of the data structure members* v *and* $\widetilde{v}$*. Note that* $v^{(0)} = \widetilde{v}^{(0)} = w^{(0)}$*.*

In the j*-th iteration, we use* $w^{(j+1)}$ *to denote the input* w^{new} *of* UpdateQuery*. Since the procedure* Update *might update both* v *and* $\widetilde{v}$*, we use* $v^{(j+1)}$ *and* $\widetilde{v}^{(j+1)}$ *to denote the new values of* v *and* $\widetilde{v}$ *respectively.*

We define similar notations for the input query vector h and its approximations g and $\widetilde{g}$:

Definition F.2 (Definitions of sequences $\{h^{(j)}\}_{j=0}^T$, $\{g^{(j)}\}_{j=0}^T$ and $\{\widetilde{g}^{(j)}\}_{j=0}^T$)**.** *When initializing, we use* $g^{(0)}$ *and* $\widetilde{g}^{(0)}$ *to denote the initial values of the data structure members* g *and* $\widetilde{g}$*. Note that* $g^{(0)} = \widetilde{g}^{(0)} = h^{(0)}$*.*

In the j*-th iteration, we use* $h^{(j+1)}$ *to denote the input* h^{new} *of* UpdateQuery*. Since the procedure* Update *might update both* g *and* $\widetilde{g}$*, we use* $g^{(j+1)}$ *and* $\widetilde{g}^{(j+1)}$ *to denote the new values of* g *and* $\widetilde{g}$ *respectively.*

The following four notations will be helpful to prove the amortized cost of the procedures:

Definition F.3 (Definition of k, $\widetilde{k}$, p, and $\widetilde{p}$)**.** *In the* j*-th iteration, we define the following notations:*

1. k_j *and* $\widetilde{k}_j$ *denote outputs* k *and* $\widetilde{k}$ *of* UpdateV *on Line 4 of* UpdateQuery *(Algorithm 8).*
2. p_j *and* $\widetilde{p}_j$ *denote outputs* p *and* $\widetilde{p}$ *of* UpdateG *on Line 5 of* UpdateQuery *(Algorithm 8),*

We also define a function ψ that approximates absolute function $|x|$ around 0. It will be used to define the potential functions for amortized analysis.

Definition F.4 (Definition of ψ function)**.** *Let $\epsilon_{\mathrm{mp}} \in (0, 1/10)$. Let $\psi : \mathbb{R} \to \mathbb{R}$ be defined by*

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

We also make the following assumption about the values of ϵ_{mp} and ϵ_{far}:

Assumption F.5. *The error parameters satisfy $0 < \epsilon_{\mathrm{mp}} < 1/10$ and $0 < \epsilon_{\mathrm{far}} < \frac{\epsilon_{\mathrm{mp}}}{100 \log n}$.*

F.2 Facts based on Adjust and two level of SoftThreshold

The following facts are direct results from the algorithms, and they are proved solely based on the algorithms. They will be useful when proving the lemmas of amortized running time.

Fact F.6 (Characterization of $\widetilde{k}$ (Line 4), $\widetilde{v}^{\mathrm{tmp}}$ (Line 4), $\widetilde{v}^{\mathrm{new}}$ (Line 5) of UpdateV)**.** *In the j-th iteration, $\forall i \in [n]$, let $y_i = |w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1|$. Let $\pi : [n] \to [n]$ be the sorting such that $y_{\pi(i)} \geq y_{\pi(i+1)}$. We have the following guarantees from the description of procedures* SoftThreshold *(Algorithm 7) and* Adjust *(Algorithm 6).*

1. $\{i \in [n] : \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1) \geq \epsilon_{\mathrm{mp}}/2\} \subseteq \pi([\widetilde{k}])$
2. $\widetilde{v}_i^{\mathrm{tmp}} = \begin{cases} w_i^{(j+1)}, & \text{if } i \in \pi([\widetilde{k}]); \\ \widetilde{v}_i^{(j)}, & \text{otherwise.} \end{cases}$
3. $\widetilde{v}_i^{\mathrm{new}} = \begin{cases} v_i^{(j)}, & \text{if } i \in \pi([\widetilde{k}]) \text{ and } w_i^{(j+1)} \in [(1-\epsilon_{\mathrm{far}})v_i^{(j)}, (1+\epsilon_{\mathrm{far}})v_i^{(j)}]; \\ w_i^{(j+1)}, & \text{if } i \in \pi([\widetilde{k}]) \text{ but } w_i^{(j+1)} \notin [(1-\epsilon_{\mathrm{far}})v_i^{(j)}, (1+\epsilon_{\mathrm{far}})v_i^{(j)}]; \\ \widetilde{v}_i^{(j)}, & \text{otherwise.} \end{cases}$
4. $\forall i \notin \pi([\widetilde{k}]),\ |w_i^{(j+1)}/\widetilde{v}_i^{\mathrm{new}} - 1| < \epsilon_{\mathrm{mp}}$.

Proof. **Part 1 and 2.** In Line 4 in Procedure UpdateV(Algorithm 9), we create $\widetilde{v}^{\mathrm{tmp}}, \widetilde{k}$ by calling procedure SoftThreshold$(y_i \leftarrow \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1), w^{(j+1)}, \widetilde{v}^{(j)}, \epsilon_{\mathrm{mp}}/2, n^{\widetilde{a}})$.

From the initial assignment of $\widetilde{k}$ on Line 5 of SoftThreshold(Algorithm 7), and the fact that the repeat-loop (Line 7 to 9) only increases k, we know that for any i such that $y_i = \psi((w_i^{\mathrm{new}} - \widetilde{v}_i)/\widetilde{v}_i) \geq \epsilon_{\mathrm{mp}}/2$, $i \in \pi([\widetilde{k}])$.

By how we calculate $\widetilde{v}^{\mathrm{tmp}}$ in SoftThreshold (Line 11 of Algorithm 7), Part 2 follows directly.
Part 3. We get $\widetilde{v}^{\mathrm{new}}$ by calling procedure Adjust $(\widetilde{v}^{\mathrm{tmp}}, \widetilde{v}^{(j)}, v^{(j)}, \epsilon_{\mathrm{far}})$ (Line 5 in Procedure UpdateV, Algorithm 9). By Part 2 and the rule of how we calculate $\widetilde{v}_i^{\mathrm{new}}$(see Line 5 to 8 in Procedure Adjust(Algorithm 6)):

$$\begin{aligned} &\widetilde{v}_i^{\mathrm{new}} \leftarrow \widetilde{v}_i^{\mathrm{tmp}} \\ &\textbf{if} \quad \widetilde{v}_i^{\mathrm{tmp}} \neq \widetilde{v}_i^{(j)} \quad \text{and} \quad \widetilde{v}_i^{\mathrm{tmp}} \in [(1-\epsilon_{\mathrm{far}})v_i^{(j)}, (1+\epsilon_{\mathrm{far}})v_i^{(j)}] \quad \textbf{then} \\ &\quad \widetilde{v}_i^{\mathrm{new}} \leftarrow v_i^{(j)} \end{aligned}$$

It is easy to see that for $i \notin \pi([\widetilde{k}])$, $\widetilde{v}_i^{\mathrm{tmp}} = \widetilde{v}_i^{(j)}$, so $\widetilde{v}_i^{\mathrm{new}} = \widetilde{v}_i^{\mathrm{tmp}} = \widetilde{v}_i^{(j)}$. And for $i \in \pi([\widetilde{k}])$, if $w_i^{(j+1)} \in [(1-\epsilon_{\mathrm{far}})v_i^{(j)}, (1+\epsilon_{\mathrm{far}})v_i^{(j)}]$, then $\widetilde{v}_i^{\mathrm{new}}$ is adjusted to be $v_i^{(j)}$, otherwise $\widetilde{v}_i^{\mathrm{new}} = \widetilde{v}_i^{\mathrm{tmp}} = w_i^{(j+1)}$.

Part 4. From Part 3, $\widetilde{v}_i^{\text{new}}$ has three possible values. If $\widetilde{v}_i^{\text{new}} = w_i^{(j+1)}$, we have $|w_i^{(j+1)}/\widetilde{v}_i^{\text{new}} - 1| = 0$.

If $\widetilde{v}_i^{\text{new}} = v_i^{(j)}$, we have $w_i^{(j+1)} \in [(1-\epsilon_{\text{far}})v_i^{(j)}, (1+\epsilon_{\text{far}})v_i^{(j)}]$. So $|w_i^{(j+1)}/\widetilde{v}_i^{\text{new}} - 1| < \epsilon_{\text{far}} < \epsilon_{\text{mp}}$ (Assumption F.5).

If $\widetilde{v}_i^{\text{new}} = \widetilde{v}_i^{(j)}$, we have $i \notin \pi([\widetilde{k}])$, then by Part 1 we know that $\psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1) < \epsilon_{\text{mp}}/2$. By definition of ψ function (Definition F.4), if $\psi(x) < \epsilon_{\text{mp}}/2$, we have $|x| < \epsilon_{\text{mp}}$. □

Fact F.7 (Characterization of k (Line 7), v^{new} (Line 7) of UpdateV)**.** *In the j-th iteration, $\forall i \in [n]$, let $y_i = \psi(w_i^{(j+1)}/v_i^{(j)} - 1) + \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1)$. Let $\pi : [n] \to [n]$ be the sorting such that $y_{\pi(i)} \geq y_{\pi(i+1)}$. We have the following guarantees from the description of procedure* SoftThreshold *(Algorithm 7).*

1. $\left\{ i \in [n] : \psi(w_i^{(j+1)}/v_i^{(j)} - 1) + \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1) \geq \epsilon_{\text{far}}^2/(32\epsilon_{\text{mp}}) \right\} \subseteq \pi([k])$
2. $v_i^{\text{new}} = \begin{cases} w_i^{(j+1)}, & \text{if } i \in \pi([k]); \\ v_i^{(j)}, & \text{otherwise.} \end{cases}$
3. $\forall i \notin \pi([k]),\ |w_i^{(j+1)}/v_i^{\text{new}} - 1| < \epsilon_{\text{mp}}$.

Proof. **Part 1 and 2.** From Line 7 of UpdateV (Algorithm 9) $v^{\text{new}} \in \mathbb{R}^n$ is the output of

$$\text{SoftThreshold}(y_i \leftarrow \psi(w_i^{(j+1)}/v_i^{(j)} - 1) + \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1), w^{(j+1)}, v^{(j)}, \frac{\epsilon_{\text{far}}^2}{32\epsilon_{\text{mp}}}, n^a).$$

The statements then follow from a similar proof as that of Part 1 and 2 of Fact F.6.

Part 3. From Part 2, v_i^{new} has two possible values. If $v_i^{\text{new}} = w_i^{(j+1)}$, we have $|w_i^{(j+1)}/v_i^{\text{new}} - 1| = 0$.

If $v_i^{\text{new}} = v_i^{(j)}$, we have $i \notin \pi([k])$, then by Part 1 we know that $\psi(w_i^{(j+1)}/v_i^{(j)} - 1) < \epsilon_{\text{far}}^2/(32\epsilon_{\text{mp}}) < \epsilon_{\text{mp}}/2$ (Assumption F.5). Then $|w_i^{(j+1)}/v_i^{(j)} - 1| < \epsilon_{\text{mp}}$ by Definition F.4. □

Fact F.8 ($\widetilde{v}$ is far away from v)**.** *For any $j \in \{0, ..., T\}$, for any $i \in [n]$, either $v_i^{(j)} = \widetilde{v}_i^{(j)}$, or $|\widetilde{v}_i^{(j)}/v_i^{(j)} - 1| > \epsilon_{\text{far}}$.*

Proof. We prove it by induction. When initializing, we set $v^{(0)} = \widetilde{v}^{(0)}$ (Line 14 of Initialize, Algorithm 5), so the statement is true when $j = 0$.

In later iterations, v and $\widetilde{v}$ can only be modified by procedure UpdateV. And in UpdateV (Algorithm 9), the if branch on Line 6 ensures that we can enter at most one of MatrixUpdate or PartialMatrexUpdate. Now we analyze iteration j by looking into these two cases.

Case 1. When we enter Procedure MatrixUpdate, we always set $v \leftarrow v^{\text{new}}$ and $\widetilde{v} \leftarrow v^{\text{new}}$ (Line 13 of Algorithm 13). So $v^{(j+1)} = \widetilde{v}^{(j+1)}$, and the statement holds in this case.

Case 2. When we enter Procedure PartialMatrexUpdate, we know we did not enter Procedure MatrixUpdate due to the else-if branch, and we do not modify v in PartialMatrexUpdate, so v does not change, i.e., $v^{(j+1)} = v^{(j)}$.

And for $\widetilde{v}$, we set $\widetilde{v}^{(j+1)} \leftarrow \widetilde{v}^{\text{new}}$ (Line 16 in Algorithm 14), where $\widetilde{v}^{\text{new}}$ is defined on Line 5 of Procedure UpdateV(Algorithm 9). By Part 3 of Fact F.6 we have

$$\widetilde{v}_i^{(j+1)} \leftarrow \widetilde{v}_i^{\text{new}} = \begin{cases} v_i^{(j)}, & \text{if } i \in \pi([\widetilde{k}]) \text{ and } w_i^{(j+1)} \in [(1-\epsilon_{\text{far}})v_i^{(j)}, (1+\epsilon_{\text{far}})v_i^{(j)}]; \\ w_i^{(j+1)}, & \text{if } i \in \pi([\widetilde{k}]) \text{ but } w_i^{(j+1)} \notin [(1-\epsilon_{\text{far}})v_i^{(j)}, (1+\epsilon_{\text{far}})v_i^{(j)}]; \\ \widetilde{v}_i^{(j)}, & \text{otherwise.} \end{cases}$$

In the first case, we have $\widetilde{v}_i^{(j+1)} = v_i^{(j)} = v_i^{(j+1)}$, and the lemma statement holds.

In the second case, we have $|\widetilde{v}_i^{(j+1)}/v_i^{(j+1)}-1| = |w_i^{(j+1)}/v_i^{(j)}-1| > \epsilon_{\mathrm{far}}$, and the lemma statement is also true.

In the third case we have $\widetilde{v}^{(j+1)} = \widetilde{v}^{(j)}$. The lemma statement is true by induction hypothesis. □

The following corollary follows from the above fact and triangle inequality:

Corollary F.9. *For any $j \in \{0,\dots,T\}$ and any $i \in [n]$, if $v_i^{(j)} \neq \widetilde{v}_i^{(j)}$, then $\forall x \in \mathbb{R}$, we have*

$$|\frac{x - v_i^{(j)}}{v_i^{(j)}}| + |\frac{x - \widetilde{v}_i^{(j)}}{\widetilde{v}_i^{(j)}}| \geq \epsilon_{\mathrm{far}}/2.$$

Proof. Since $v_i^{(j)} \neq \widetilde{v}_i^{(j)}$, from Fact F.8, we have

$$|\widetilde{v}_i^{(j)}/v_i^{(j)} - 1| > \epsilon_{\mathrm{far}}. \tag{47}$$

We consider the following two cases depend on the value of $|\widetilde{v}_i^{(j)}/v_i^{(j)}|$:

Case 1, $|\widetilde{v}_i^{(j)}/v_i^{(j)}| \leq 1$: We have

$$|\frac{x - \widetilde{v}_i^{(j)}}{\widetilde{v}_i^{(j)}}| + |\frac{x - v_i^{(j)}}{v_i^{(j)}}| = |\frac{x - \widetilde{v}_i^{(j)}}{v_i^{(j)}}| \cdot |\frac{v_i^{(j)}}{\widetilde{v}_i^{(j)}}| + |\frac{x - v_i^{(j)}}{v_i^{(j)}}| \geq |\frac{x - \widetilde{v}_i^{(j)}}{v_i^{(j)}}| + |\frac{x - v_i^{(j)}}{v_i^{(j)}}| \geq |\frac{\widetilde{v}_i^{(j)} - v_i^{(j)}}{v_i^{(j)}}| > \epsilon_{\mathrm{far}}.$$

where the second step follows from the assumption of Case 1 that $|\widetilde{v}_i^{(j)}/v_i^{(j)}| \leq 1$, the third step follows from triangle inequality, and the fourth step follows from Eq. (47).

Case 2, $|\widetilde{v}_i^{(j)}/v_i^{(j)}| > 1$: We have

$$|\frac{\widetilde{v}_i^{(j)} - v_i^{(j)}}{\widetilde{v}_i^{(j)}}| = \frac{|(\widetilde{v}_i^{(j)} - v_i^{(j)})/v_i^{(j)}|}{|\widetilde{v}_i^{(j)}/v_i^{(j)}|} \geq \frac{|(\widetilde{v}_i^{(j)} - v_i^{(j)})/v_i^{(j)}|}{|(\widetilde{v}_i^{(j)} - v_i^{(j)})/v_i^{(j)}| + 1} \geq \frac{\epsilon_{\mathrm{far}}}{\epsilon_{\mathrm{far}} + 1} \geq \epsilon_{\mathrm{far}}/2.$$

where the second step follows from triangle inequality, the third step follows from Eq. (47) and the fact that function $\frac{x}{x+1}$ is monotonically increasing, and the last step follows from $\epsilon_{\mathrm{far}} \leq 1$.

Then similar as Case 1 we have

$$|\frac{x - \widetilde{v}_i^{(j)}}{\widetilde{v}_i^{(j)}}| + |\frac{x - v_i^{(j)}}{v_i^{(j)}}| = |\frac{x - \widetilde{v}_i^{(j)}}{\widetilde{v}_i^{(j)}}| + |\frac{x - v_i^{(j)}}{\widetilde{v}_i^{(j)}}| \cdot |\frac{\widetilde{v}_i^{(j)}}{v_i^{(j)}}| \geq |\frac{x - \widetilde{v}_i^{(j)}}{\widetilde{v}_i^{(j)}}| + |\frac{x - v_i^{(j)}}{\widetilde{v}_i^{(j)}}| \geq |\frac{\widetilde{v}_i^{(j)} - v_i^{(j)}}{\widetilde{v}_i^{(j)}}| > \epsilon_{\mathrm{far}}/2.$$

□

Fact F.10 (Lower bound on k_j)**.** *In the j-th iteration, we have that either $k_j = 0$, or $k_j \geq n^a$.*

Proof. Recall that k_j is the second returned value by UpdateV on Line 4 of UpdateQuery (Algorithm 8). If in UpdateV (Algorithm 9) we do not enter the if-branch on Line 6, then by the return clauses on Line 19 and Line 22, we have that $k_j = 0$.

Now we only need to prove that when we enter the if-branch on Line 6 of UpdateV (Algorithm 9), the returned value $k_j = k \geq n^a$. When the if-clause on Line 6 is true, we have

$$|\operatorname{supp}(\widetilde{v}^{\mathrm{new}} - v^{(j)})| \geq n^a, \tag{48}$$

where $\widetilde{v}^{\mathrm{new}}$ is defined on Line 5. $\forall i \in [n]$, let $y_i = \psi(w_i^{(j+1)}/v_i^{(j)} - 1) + \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1)$. Let $\pi : [n] \to [n]$ be the sorting such that $y_{\pi(i)} \geq y_{\pi(i+1)}$. From Part 1 of Fact F.7, we have

$$\left\{ i \in [n] : \psi(w_i^{(j+1)}/v_i^{(j)} - 1) + \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1) \geq \epsilon_{\mathrm{far}}^2/(32\epsilon_{\mathrm{mp}}) \right\} \subseteq \pi([k]). \tag{49}$$

Now it suffices to prove

$$\operatorname{supp}(\widetilde{v}^{\text{new}} - v^{(j)}) \subseteq \left\{ i : \psi(w_i^{(j+1)}/v_i^{(j)} - 1) + \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1) \geq \epsilon_{\text{far}}^2/(32\epsilon_{\text{mp}}) \right\}, \tag{50}$$

because then we would have

$$\begin{aligned} k_j = k = |\pi([k])| &\geq \left| \left\{ i : \psi(w_i^{(j+1)}/v_i^{(j)} - 1) + \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1) \geq \epsilon_{\text{far}}^2/(32\epsilon_{\text{mp}}) \right\} \right| \\ &\geq |\operatorname{supp}(\widetilde{v}^{\text{new}} - v^{(j)})| \geq n^a, \end{aligned}$$

where the first step follows from the definition of k_j, the third step follows from Eq. (49), the fourth step follows from Eq. (50), and the fifth step follows from Eq. (48).

Now it remains to prove Eq. (50). We first prove that

$$\operatorname{supp}(\widetilde{v}^{\text{new}} - v^{(j)}) \subseteq \left\{ i : |w_i^{(j+1)}/v_i^{(j)} - 1| + |w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1| \geq \epsilon_{\text{far}}/2 \right\}.$$

Using Part 3 of Fact F.6 we know that $\widetilde{v}_i^{\text{new}}$ can be $v_i^{(j)}$, $w_i^{(j+1)}$, or $\widetilde{v}_i^{(j)}$. So for any $i \in \operatorname{supp}(\widetilde{v}^{\text{new}} - v^{(j)})$, since $\widetilde{v}_i^{\text{new}} \neq v_i^{(j)}$, we know that $\widetilde{v}_i^{\text{new}}$ is either $\widetilde{v}_i^{(j)}$ or $w_i^{(j+1)}$. We consider these two cases:

1. $\widetilde{v}_i^{\text{new}} = \widetilde{v}_i^{(j)} \neq v_i^{(j)}$. Using Corollary F.9, and plugging in $x \leftarrow w_i^{(j+1)}$ we directly have that $|w_i^{(j+1)}/v_i^{(j)} - 1| + |w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1| \geq \epsilon_{\text{far}}/2$.
2. $\widetilde{v}_i^{\text{new}} = w_i^{(j+1)} \neq v_i^{(j)}$. By Part 3 of Fact F.6 we have $w_i^{(j+1)} \notin [(1-\epsilon_{\text{far}})v_i^{(j)}, (1+\epsilon_{\text{far}})v_i^{(j)}]$, which then gives us $|w_i^{(j+1)}/v_i^{(j)} - 1| \geq \epsilon_{\text{far}} \geq \epsilon_{\text{far}}/2$.

Thus we always have that $\forall i \in \operatorname{supp}(\widetilde{v}^{\text{new}} - v^{(j)})$, $|w_i^{(j+1)}/v_i^{(j)} - 1| + |w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1| \geq \epsilon_{\text{far}}/2$. So at least one of the following is true:

$$|w_i^{(j+1)}/v_i^{(j)} - 1| \geq \epsilon_{\text{far}}/4 \quad \text{or} \quad |w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1| \geq \epsilon_{\text{far}}/4.$$

Without loss of generality, assume $|w_i^{(j+1)}/v_i^{(j)} - 1| \geq \frac{\epsilon_{\text{far}}}{4}$. We have

$$\psi(w_i^{(j+1)}/v_i^{(j)} - 1) + \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1) \geq \psi(w_i^{(j+1)}/v_i^{(j)} - 1) \geq \psi(\epsilon_{\text{far}}/4) = \epsilon_{\text{far}}^2/(32\epsilon_{\text{mp}}).$$

where the first two steps follow from ψ is non-negative and non-decreasing, and the third step follows from $\psi(\epsilon_{\text{far}}/4) = \epsilon_{\text{far}}^2/(32\epsilon_{\text{mp}})$ (see Definition F.4 of ψ). Thus this proves Eq. (50). □

Fact F.11 (Lower bound on $\widetilde{k}_j$)**.** *In the j-th iteration, we have that either $\widetilde{k}_j = 0$, or $\widetilde{k}_j \geq n^{\widetilde{a}}$.*

Proof. Recall that $\widetilde{k}_j$ is the third returned value by UpdateV on Line 4 of UpdateQuery (Algorithm 8). If in UpdateV (Algorithm 9) we do not enter the else-if branch on Line 14, then by the return clauses on Line 12 and Line 22, we have that $\widetilde{k}_j = 0$.

Now we only need to prove that when we enter the else-if branch on Line 14 of UpdateV (Algorithm 9), the returned value $\widetilde{k}_j = \widetilde{k} \geq n^{\widetilde{a}}$. When the if-clause on Line 14 is true, we have

$$|\operatorname{supp}(\widetilde{v}^{\text{new}} - \widetilde{v}^{(j)})| \geq n^{\widetilde{a}}.$$

From Part 3 of Fact F.6, we have $|\operatorname{supp}(\widetilde{v}^{\text{new}} - \widetilde{v}^{(j)})| \subseteq \pi([\widetilde{k}])$. Therefore,

$$\widetilde{k}_j = \widetilde{k} \geq |\operatorname{supp}(\widetilde{v}^{\text{new}} - \widetilde{v}^{(j)})| \geq n^{\widetilde{a}}.$$

□

Fact F.12 (Characterization of MatrixUpdate)**.** *Assume Assumption F.5 is true. In the j-th iteration, $\forall i \in [n]$, let $y_i = \psi(w_i^{(j+1)}/v_i^{(j)} - 1) + \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1)$. Let $\pi : [n] \to [n]$ be the sorting such that $y_{\pi(i)} \geq y_{\pi(i+1)}$. If $k_j > 0$, we have the following:*

1. *k_j satisfies that either $k_j = n$ or $y_{\pi(k_j)} < (1 - 1/\log n) \cdot y_{\pi(k_j/1.5)}$.*
2. *$y_{\pi(k_j)} \geq \epsilon_{\mathrm{far}}^2/(3200\epsilon_{\mathrm{mp}})$.*
3. *After the procedure* MatrixUpdate*,*
$$\begin{cases} \widetilde{v}_{\pi(i)}^{(j+1)} = v_{\pi(i)}^{(j+1)} = w_{\pi(i)}^{(j+1)}, & \forall i \leq k_j; \\ \widetilde{v}_{\pi(i)}^{(j+1)} = v_{\pi(i)}^{(j+1)} = v_{\pi(i)}^{(j)} = \widetilde{v}_{\pi(i)}^{(j)}, & \forall i > k_j. \end{cases}$$
4. *The running time of* MatrixUpdate *in the jth iteration is $\mathcal{T}_{\mathrm{mat}}(k_j, n^{1+o(1)}, n)$.*

Proof. **Part 1 and Part 2.** Note that as long as $k_j > 0$, k_j is the k computed in Line 7 in Procedure UpdateV(Algorithm 9). By Fact F.10 and $k_j \neq 0$, we have $k_j \geq n^a$. Therefore, when calculating k using one call of SoftThreshold (Line 7 in Algorithm 9), we must have entered the repeat-until branch (Line 7 to 9 in Algorithm 7). So k_j must satisfy the end condition of the repeat-loop that either $k_j = n$ or $y_{\pi(k_j)} < (1 - 1/\log n) \cdot y_{\pi(k_j/1.5)}$. This finishes the proof of Part 1.

Further, let k^* denote the largest index such that $y_{\pi(k^*)} \geq \epsilon_{\mathrm{far}}^2/(32\epsilon_{\mathrm{mp}})$. We have

$$y_{\pi(k_j)} \geq (1 - 1/\log n)^{\log_{1.5} k_j - \log_{1.5} k^*} \cdot y_{\pi(k^*)} \geq (1 - 1/\log n)^{\log_{1.5} n} \cdot \frac{\epsilon_{\mathrm{far}}^2}{32\epsilon_{\mathrm{mp}}} \geq \frac{\epsilon_{\mathrm{far}}^2}{3200\epsilon_{\mathrm{mp}}},$$

where the first step follows from the repeat-loop (Line 8 in Algorithm 7), the second step follows from $\log_{1.5} k_j - \log_{1.5} k^* \leq \log_{1.5} n$ and $y_{\pi(k^*)} \geq \epsilon_{\mathrm{far}}^2/(32\epsilon_{\mathrm{mp}})$, and the last step follows from the fact that for $n \geq 4$, $(1 - 1/\log n)^{\log_{1.5} n} \geq 1/100$. This finishes the proof of Part 2.

Part 3. From Line 13 of MatrixUpdate (Algorithm 13) we have that $v^{(j+1)} = \widetilde{v}^{(j+1)} = v^{\mathrm{new}}$. Using Part 2 of Fact F.7, we have that

$$\begin{cases} v_{\pi(i)}^{\mathrm{new}} = w_{\pi(i)}^{(j+1)}, & \forall i \leq k_j; \\ v_{\pi(i)}^{\mathrm{new}} = v_{\pi(i)}^{(j)}, & \forall i > k_j, \end{cases} \tag{51}$$

so we have

$$\begin{cases} \widetilde{v}_{\pi(i)}^{(j+1)} = v_{\pi(i)}^{(j+1)} = w_{\pi(i)}^{(j+1)}, & \forall i \leq k_j; \\ \widetilde{v}_{\pi(i)}^{(j+1)} = v_{\pi(i)}^{(j+1)} = v_{\pi(i)}^{(j)}, & \forall i > k_j. \end{cases}$$

Note that by Part 1, either $k_j = n$ or $y_{\pi(k_j)} < (1 - 1/\log n) \cdot y_{\pi(k_j/1.5)}$. If $k_j = n$, there is no $i > k_j$, so the proof is already finished.

Otherwise, for any $i > k_j$, we prove that $\widetilde{v}_{\pi(i)}^{(j)} = v_{\pi(i)}^{(j)}$ by contradiction. If $\widetilde{v}_{\pi(i)}^{(j)} \neq v_{\pi(i)}^{(j)}$, using Corollary F.9 and plugging in $x \leftarrow w_{\pi(i)}^{(j+1)}$, we have $|w_{\pi(i)}^{(j+1)}/v_{\pi(i)}^{(j)} - 1| + |w_{\pi(i)}^{(j+1)}/\widetilde{v}_{\pi(i)}^{(j)} - 1| \geq \epsilon_{\mathrm{far}}/2$, so at least one of the following is true:

$$|w_{\pi(i)}^{(j+1)}/v_{\pi(i)}^{(j)} - 1| \geq \epsilon_{\mathrm{far}}/4 \quad \text{or} \quad |w_{\pi(i)}^{(j+1)}/\widetilde{v}_{\pi(i)}^{(j)} - 1| \geq \epsilon_{\mathrm{far}}/4.$$

Thus we have

$$y_{\pi(i)} = \psi(w_{\pi(i)}^{(j+1)}/v_{\pi(i)}^{(j)} - 1) + \psi(w_{\pi(i)}^{(j+1)}/\widetilde{v}_{\pi(i)}^{(j)} - 1) \geq \psi(\epsilon_{\mathrm{far}}/4) = \epsilon_{\mathrm{far}}^2/(32\epsilon_{\mathrm{mp}}),$$

where the first two steps follow from ψ is non-negative and non-decreasing, and the third step follows from $\psi(\epsilon_{\text{far}}/4) = \epsilon_{\text{far}}^2/(32\epsilon_{\text{mp}})$ (see Definition F.4 of ψ).

Then we have $y_{\pi(k_j)} \geq y_{\pi(i)} \geq \epsilon_{\text{far}}^2/(32\epsilon_{\text{mp}})$ since y is decreasingly sorted according to π and $i > k_j$. But from the initial assignment of k_j on Line 5 of SoftThreshold(Algorithm 7), and that k_j can only strictly increase afterwards, we have that $y_{\pi(k_j)} < \epsilon_{\text{far}}^2/(32\epsilon_{\text{mp}})$. This leads to contradiction. Thus we have $\widetilde{v}_{\pi(i)}^{(j+1)} = v_{\pi(i)}^{(j+1)} = v_{\pi(i)}^{(j)} = \widetilde{v}_{\pi(i)}^{(j)}$, $\forall i > k_j$.

Part 4. This directly follows from Lemma E.12 which proves the running time of MatrixUpdate per call. □

Fact F.13 (Characterization of PartialMatrixUpdate)**.** *Assume Assumption F.5 is true. In the j-th iteration, $\forall i \in [n]$, let $y_i = \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1)$. Let $\pi : [n] \to [n]$ be the sorting such that $y_{\pi(i)} \geq y_{\pi(i+1)}$. If $\widetilde{k}_j > 0$, we have the following:*

1. *$\widetilde{k}_j$ satisfies that either $\widetilde{k}_j = n$ or $y_{\pi(\widetilde{k}_j)} < (1 - 1/\log n) \cdot y_{\pi(\widetilde{k}_j/1.5)}$.*
2. *$y_{\pi(\widetilde{k}_j)} \geq \epsilon_{\text{mp}}/100$.*
3. *After the procedure PartialMatrixUpdate, $\forall i \in \pi([\widetilde{k}_j])$, $\widetilde{v}_i^{(j+1)}$ satisfies*
$$\psi(w_i^{(j+1)}/\widetilde{v}_i^{(j+1)} - 1) \leq \epsilon_{\text{mp}}/(200 \log n), \tag{52}$$
and $\forall i \notin \pi([\widetilde{k}_j])$, $\widetilde{v}_i^{(j+1)} = \widetilde{v}_i^{(j)}$. Also, $\forall i \in [n]$, $v_i^{(j+1)} = v_i^{(j)}$.
4. *The running time of PartialMatrixUpdate in the j-th iteration is $\mathcal{T}_{\text{mat}}(\widetilde{k}_j, n^a, n^a)$.*

Proof. **Part 1 and 2.** Note that as long as $\widetilde{k}_j > 0$, $\widetilde{k}_j$ is the $\widetilde{k}$ computed in Line 4 in Procedure UpdateV (Algorithm 9). By Fact F.11, we have $\widetilde{k} \geq n^{\widetilde{a}}$. So by a similar argument as that of the proof Part 1 and 2 of Fact F.12, we can prove Part 1 and 2 of this fact.

Part 3. Because we do not modify v in PartialMatrixUpdate, so $\forall i \in [n]$, $v_i^{(j+1)} = v_i^{(j)}$.

In procedure PartialMatrixUpdate, we set $\widetilde{v}^{(j+1)} \leftarrow \widetilde{v}^{\text{new}}$ (Line 16 of Algorithm 14), where $\widetilde{v}^{\text{new}}$ is created by one call to SoftThreshold (Line 4 in UpdateV, Algorithm 8). By Part 3 of Fact F.6, we have

$$\widetilde{v}_i^{\text{new}} \leftarrow \begin{cases} v_i^{(j)}, & \text{if } i \in \pi([\widetilde{k}]) \text{ and } w_i^{(j+1)} \in [(1-\epsilon_{\text{far}})v_i^{(j)}, (1+\epsilon_{\text{far}})v_i^{(j)}]; \\ w_i^{(j+1)}, & \text{if } i \in \pi([\widetilde{k}]) \text{ but } w_i^{(j+1)} \notin [(1-\epsilon_{\text{far}})v_i^{(j)}, (1+\epsilon_{\text{far}})v_i^{(j)}]; \\ \widetilde{v}_i^{(j)}, & \text{otherwise.} \end{cases}$$

For $i \notin \pi([\widetilde{k}_j])$, $\widetilde{v}_i^{(j+1)} = \widetilde{v}_i^{(j)}$.

For $i \in \pi([\widetilde{k}_j])$, if $\widetilde{v}_i^{(j+1)} = \widetilde{v}_i^{\text{new}} = w_i^{(j+1)}$, Eq. (52) is trivially true since $\psi(w_i^{(j+1)}/\widetilde{v}_i^{(j+1)} - 1) = 0$.

Otherwise $\widetilde{v}_i^{(j+1)} = \widetilde{v}_i^{\text{new}} = v_i^{(j)}$ and $w_i^{(j+1)} \in [(1-\epsilon_{\text{far}})v_i^{(j)}, (1+\epsilon_{\text{far}})v_i^{(j)}]$, we have:

$$\psi(w_i^{(j+1)}/\widetilde{v}_i^{(j+1)} - 1) = \psi(w_i^{(j+1)}/v_i^{(j)} - 1) \leq \psi(\epsilon_{\text{far}}) = \epsilon_{\text{far}}^2/(2\epsilon_{\text{mp}}) \leq (\epsilon_{\text{mp}}/(200 \log n)),$$

where the last step is by $\epsilon_{\text{far}} \leq \epsilon_{\text{mp}}/(100 \log n)$ (Assumption F.5).

Part 4. This directly follows from Lemma E.18 which proves the running time of PartialMatrixUpdate per call. □

Corollary F.14. *Assume Assumption F.5 is true. In the j-th iteration, if we enter PartialMatrixUpdate, we must have $\forall i \in [n]$, $\psi(w_i^{(j+1)}/\widetilde{v}_i^{(j+1)} - 1) \leq \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1)$.*

Proof. If we enter PartialMatrixUpdate, we must have $\widetilde{k}_j > 0$. By Part 2,3 of Fact F.13, there is some permutation π such that $\forall i \in \pi([\widetilde{k}_j])$,

$$\psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1) \geq \epsilon_{\mathrm{mp}}/100 \quad \text{and} \quad \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j+1)} - 1) \leq \epsilon_{\mathrm{mp}}/(200 \log n) < \epsilon_{\mathrm{mp}}/100.$$

And also by Part 3 of Fact F.13, $\forall i \notin \pi([\widetilde{k}_j])$, $\widetilde{v}_i^{(j+1)} = \widetilde{v}_i^{(j)}$. Therefore, $\forall i \notin \pi([\widetilde{k}_j])$, we have that $\psi(w_i^{(j+1)}/\widetilde{v}_i^{(j+1)} - 1) = \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1)$. □

F.3 Amortized analysis for MatrixUpdate

F.3.1 Definitions

Definition F.15 (x and y for MatrixUpdate)**.** *For any $j \in \{0, 1, \ldots, T-1\}$, and for any $i \in [n]$, we define $x_i^{(j)}$ and $y_i^{(j)}$ as follows:*

$$x_i^{(j)} := \psi(w_i^{(j)}/v_i^{(j)} - 1) + \psi(w_i^{(j)}/\widetilde{v}_i^{(j)} - 1), \qquad y_i^{(j)} := \psi(w_i^{(j+1)}/v_i^{(j)} - 1) + \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1),$$

where $v^{(j)}$, $\widetilde{v}^{(j)}$ and $w^{(j)}$ are defined as of Definition F.1.

Note that the difference between $x_i^{(j)}$ and $y_i^{(j)}$ is that w changes from $w^{(j)}$ to $w^{(j+1)}$. The difference between $y_i^{(j)}$ and $x_i^{(j+1)}$ is that v and $\widetilde{v}$ changes from $v^{(j)}, \widetilde{v}^{(j)}$ to $v^{(j+1)}, \widetilde{v}^{(j+1)}$.

For convenience, we define permutations of the coordinates that are sorted according to x or y.

Definition F.16 (Sorting permutations for MatrixUpdate)**.** *For any $j \in \{0, 1, \ldots, T\}$, let τ_j be the permutation such that $x_{\tau_j(i)}^{(j)} \geq x_{\tau_j(i+1)}^{(j)}$, and let π_j be the permutation such that $y_{\pi_j(i)}^{(j)} \geq y_{\pi_j(i+1)}^{(j)}$.*

When it is clear from the context that we are only arguing about the j-th iteration, for simplicity we assume the coordinates of vector $x^{(j)} \in \mathbb{R}^n$ are sorted such that $x_i^{(j)} \geq x_{i+1}^{(j)}$. And we use τ and π to denote the permutations such that $x_{\tau(i)}^{(j+1)} \geq x_{\tau(i+1)}^{(j+1)}$ and $y_{\pi(i)}^{(j)} \geq y_{\pi(i+1)}^{(j)}$.

Definition F.17 (g for MatrixUpdate)**.** *For some $a \leq \alpha$, where α is the dual exponent of matrix multiplication, we define $g \in \mathbb{R}^n$ as follows:*

$$g_i = \begin{cases} n^{-a}, & \text{if } i \leq n^a; \\ i^{\frac{\omega-2}{1-a}-1} \cdot n^{-\frac{a(\omega-2)}{1-a}}, & \text{if } i \in (n^a, n]. \end{cases}$$

Note that g is non-increasing. For all $k_j \in (n^a, n]$, $(k_j \cdot g_{k_j} n^2) = k_j^{\frac{\omega-2}{1-a}} \cdot n^{2-\frac{a(\omega-2)}{1-a}}$ is an upper bound of the running time $\mathcal{T}_{\mathrm{mat}}(n, n, k_j)$ of multiplying a $n \times n$ matrix with a $n \times k_j$ matrix. For more details please refer to [GU18].

Definition F.18 (Potential function Φ for MatrixUpdate)**.** *We define the potential function in the j-th iteration as*

$$\Phi_j = \sum_{i=1}^{n} g_i \cdot x_{\tau_j(i)}^{(j)}.$$

Note that we always have $\Phi_j \geq 0$ since $\forall i$, g_i and $x_i^{(j)}$ are both non-negative.

F.3.2 Main result

Lemma F.19 (Amortized time for MatrixUpdate)**.** *Let sequences* $\{w^{(j)}\}_{j=0}^T$, $\{v^{(j)}\}_{j=0}^T$, $\{\widetilde{v}^{(j)}\}_{j=0}^T$ *be defined as of Definition F.1, let* k_j *be defined as of Definition F.3, and let* $\{x^{(j)}\}_{j=0}^T$, $\{y^{(j)}\}_{j=0}^T$, g, Φ *be defined as of Definition F.15, F.17 and F.18. If we further have the condition that the input sequence satisfies the following:* $\forall j \in \{0, ..., T-1\}$

$$\sum_{i=1}^n (\mathbb{E}[w_i^{(j+1)}|w^{(j)}]/w_i^{(j)} - 1)^2 \leq C_1^2, \quad \sum_{i=1}^n (\mathbb{E}[(w_i^{(j+1)}/w_i^{(j)} - 1)^2 \mid w^{(j)}])^2 \leq C_2^2, \quad |w_i^{(j+1)}/w_i^{(j)} - 1| \leq 1/4.$$

Then, we have that in expectation

$$\frac{1}{T}\sum_{j=1}^T k_j g_{k_j} = O\Big((\frac{C_1\epsilon_{\mathrm{mp}}}{\epsilon_{\mathrm{far}}^2} + \frac{C_2}{\epsilon_{\mathrm{far}}^2}) \cdot \log n \cdot \|g\|_2\Big).$$

Further, combining with Lemma E.12, the expected amortized running time per iteration of MatrixUpdate *is*

$$O\Big((\frac{C_1\epsilon_{\mathrm{mp}}}{\epsilon_{\mathrm{far}}^2} + \frac{C_2}{\epsilon_{\mathrm{far}}^2}) \cdot (n^{2-a/2} + n^{\omega-1/2}) \log n\Big).$$

Proof. First note that in the j-th iteration, the value of the potential Φ_j depends on $w^{(j)}$, $v^{(j)}$ and $\widetilde{v}^{(j)}$. And the value of $v^{(j)}$ and $\widetilde{v}^{(j)}$ are affected by both MatrixUpdate and PartialMatrixUpdate. We upper bound how much the potential function can increase due to changing $w^{(j)}$ to $w^{(j+1)}$ in Section F.3.3, and we also lower bound how much the potential function can decrease because of changing $v^{(j)}$ to $v^{(j+1)}$ and $\widetilde{v}^{(j)}$ to $\widetilde{v}^{(j+1)}$ in Section F.3.4.

In the beginning $v^{(0)} = \widetilde{v}^{(0)} = w^{(0)}$, so $\Phi_0 = 0$. Also note that $\Phi_j \geq 0, \forall j \in [T]$. Thus we have

$$\begin{aligned}
0 \leq \mathbb{E}[\Phi_T] - \Phi_0 = & \sum_{j=0}^{T-1} \mathbb{E}[\Phi_{j+1} - \Phi_j] = \sum_{j=0}^{T-1}\sum_{i=1}^n g_i \cdot \mathbb{E}\left[x_{\tau(i)}^{(j+1)} - x_i^{(j)}\right] \\
= & \sum_{j=0}^{T-1}\Bigg(\sum_{i=1}^n g_i \cdot \underbrace{\mathbb{E}\left[y_{\pi(i)}^{(j)} - x_i^{(j)}\right]}_{w \text{ move}} - \sum_{i=1}^n g_i \cdot \underbrace{\mathbb{E}\left[y_{\pi(i)}^{(j)} - x_{\tau(i)}^{(j+1)}\right]}_{v,\widetilde{v} \text{ move}}\Bigg) \\
\leq & \sum_{j=0}^{T-1}\Big(O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \|g\|_2 - \Omega\left(\epsilon_{\mathrm{far}}^2 \cdot k_j \cdot g_{k_j}/(\epsilon_{\mathrm{mp}} \log n)\right)\Big) \\
= & \, T \cdot O(C_1 + C_2/\epsilon_{\mathrm{mp}})\|g\|_2 - \sum_{j=1}^T \Omega\left(\epsilon_{\mathrm{far}}^2 \cdot k_j \cdot g_{k_j}/(\epsilon_{\mathrm{mp}} \log n)\right),
\end{aligned}$$

where the second step follows from splitting terms and the fact that Φ_0 is deterministic, the third step follows from the definition of Φ (Definition F.18), the fourth step follows from splitting terms, the fifth step follows from Lemma F.20 which states that $\forall w^{(j)}, v^{(j)}, \widetilde{v}^{(j)}$, we have

$$\sum_{i=1}^n g_i \cdot \mathbb{E}\left[y_{\pi(i)}^{(j)} - x_i^{(j)} \;\middle|\; w^{(j)}, v^{(j)}, \widetilde{v}^{(j)}\right] \leq O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \|g\|_2,$$

then this upper bound also holds for unconditional expectation, the fifth step also follows from Lemma F.24 which states that $\sum_{i=1}^n g_i \cdot (y_{\pi(i)}^{(j)} - x_{\tau(i)}^{(j+1)}) \geq \Omega(\epsilon_{\mathrm{far}}^2 \cdot k_j \cdot g_{k_j}/(\epsilon_{\mathrm{mp}} \log n))$.

Therefore, we have

$$\frac{1}{T}\sum_{j=1}^{T} k_j g_{k_j} \leq O\Big((\frac{C_1\epsilon_{\mathrm{mp}}}{\epsilon_{\mathrm{far}}^2} + \frac{C_2}{\epsilon_{\mathrm{far}}^2}) \cdot \log n \cdot \|g\|_2\Big).$$

Using Lemma E.12, we have that the expected amortized running time per iteration of MATRIXUPDATE is

$$\begin{aligned}\frac{1}{T}\sum_{j=1}^{T} \mathcal{T}_{\mathrm{mat}}(n,n,k_j) \leq \frac{n^2}{T}\sum_{j=1}^{T} k_j g_{k_j} &= O\Big((\frac{C_1\epsilon_{\mathrm{mp}}}{\epsilon_{\mathrm{far}}^2} + \frac{C_2}{\epsilon_{\mathrm{far}}^2}) \cdot n^2 \log n \cdot \|g\|_2\Big) \\ &= O\Big((\frac{C_1\epsilon_{\mathrm{mp}}}{\epsilon_{\mathrm{far}}^2} + \frac{C_2}{\epsilon_{\mathrm{far}}^2}) \cdot (n^{2-a/2} + n^{\omega-1/2}) \log n\Big),\end{aligned}$$

where the first step follows from the definition of g which gives that $\mathcal{T}_{\mathrm{mat}}(n,n,k_j) \leq n^2 k_j g_{k_j}$, and the third step follows from Lemma F.25 that $\|g\|_2 = O(n^{-a/2} + n^{\omega-5/2})$. □

F.3.3 w move

The goal of this section is to prove Lemma F.20.

Lemma F.20 (w move)**.** *In the j-th iteration, for any possible values $w^{(j)}$, $v^{(j)}$, and $\widetilde{v}^{(j)}$, we have*

$$\sum_{i=1}^{n} g_i \cdot \mathbb{E}\left[y_{\pi(i)}^{(j)} - x_i^{(j)} \;\middle|\; w^{(j)}, v^{(j)}, \widetilde{v}^{(j)}\right] \leq O(C_1 + C_2/\epsilon_{\mathrm{mp}})\|g\|_2. \tag{53}$$

Proof. For simplicity, in this proof we write $\mathbb{E}[\cdot]$ as a shorthand of $\mathbb{E}[\cdot|w^{(j)}, v^{(j)}, \widetilde{v}^{(j)}]$.

Observe that since the non-negative values $x_i^{(j)}$ are sorted in descending order, and g is also non-increasing, we have

$$\sum_{i=1}^{n} g_i x_{\pi(i)}^{(j)} \leq \sum_{i=1}^{n} g_i x_i^{(j)}. \tag{54}$$

We then have

$$\begin{aligned}&\sum_{i=1}^{n} g_i \cdot \mathbb{E}[y_{\pi(i)}^{(j)} - x_i^{(j)}] \leq \sum_{i=1}^{n} g_i \cdot \mathbb{E}[y_{\pi(i)}^{(j)} - x_{\pi(i)}^{(j)}] \\ =& \sum_{i=1}^{n} g_i \cdot \mathbb{E}[\psi(w_{\pi(i)}^{(j+1)}/v_{\pi(i)}^{(j)} - 1) + \psi(w_{\pi(i)}^{(j+1)}/\widetilde{v}_{\pi(i)}^{(j)} - 1)] - \sum_{i=1}^{n} g_i \cdot \mathbb{E}[\psi(w_{\pi(i)}^{(j)}/v_{\pi(i)}^{(j)} - 1) + \psi(w_{\pi(i)}^{(j)}/\widetilde{v}_{\pi(i)}^{(j)} - 1)] \\ =& \sum_{i=1}^{n} g_i \cdot \mathbb{E}[\psi(w_{\pi(i)}^{(j+1)}/v_{\pi(i)}^{(j)} - 1) - \psi(w_{\pi(i)}^{(j)}/v_{\pi(i)}^{(j)} - 1)] + \sum_{i=1}^{n} g_i \cdot \mathbb{E}[\psi(w_{\pi(i)}^{(j+1)}/\widetilde{v}_{\pi(i)}^{(j)} - 1) - \psi(w_{\pi(i)}^{(j)}/\widetilde{v}_{\pi(i)}^{(j)} - 1)]\end{aligned}$$

where the first step follows from Eq.(54), the second step follows from the definitions of $x^{(j)}$ and $y^{(j)}$ (Definition F.15).

Now $\sum_{i=1}^{n} g_i \cdot \mathbb{E}[y_{\pi(i)}^{(j)} - x_i^{(j)}] \leq O(C_1 + C_2/\epsilon_{\mathrm{mp}})\|g\|_2$ directly follows from Lemma F.21 and Lemma F.22. □

It remains to prove the following two lemmas.

Lemma F.21. *Under Assumption F.5, in the j-th iteration, for any $w^{(j)}$, $v^{(j)}$, and $\widetilde{v}^{(j)}$ we have*

$$\sum_{i=1}^{n} g_i \cdot \mathbb{E}[\psi(w_{\pi(i)}^{(j+1)}/v_{\pi(i)}^{(j)} - 1) - \psi(w_{\pi(i)}^{(j)}/v_{\pi(i)}^{(j)} - 1) \mid w^{(j)}, v^{(j)}, \widetilde{v}^{(j)}] = O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \|g\|_2.$$

Proof. For simplicity, in this proof we write $\mathbb{E}[\cdot]$ as a shorthand of $\mathbb{E}[\cdot|w^{(j)}, v^{(j)}, \widetilde{v}^{(j)}]$. And we also define x and y as

$$y_i = w_i^{(j+1)}/v_i^{(j)} - 1, \quad x_i = w_i^{(j)}/v_i^{(j)} - 1, \tag{55}$$

and they are only used in this proof. Then the lemma statement becomes

$$\sum_{i=1}^{n} g_i \cdot \mathbb{E}[\psi(y_{\pi(i)}) - \psi(x_{\pi(i)})] = O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \|g\|_2.$$

Let I be the set of indices such that $|x_i| \le 1$. We separate the term into two:

$$\sum_{i=1}^{n} g_i \cdot \mathbb{E}[\psi(y_{\pi(i)}) - \psi(x_{\pi(i)})] = \sum_{i\in I} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i) - \psi(x_i)] + \sum_{i\in I^c} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i) - \psi(x_i)].$$

Case 1: Terms from I. Mean value theorem shows that for any y_i, there exist ζ such that

$$\begin{aligned}\psi(y_i) - \psi(x_i) &= \psi'(x_i)(y_i - x_i) + \frac{1}{2}\psi''(\zeta)(y_i - x_i)^2 \\ &\le \psi'(x_i) \cdot \frac{w_i^{(j+1)} - w_i^{(j)}}{v_i^{(j)}} + \frac{L_2}{2} \cdot (\frac{w_i^{(j+1)} - w_i^{(j)}}{v_i^{(j)}})^2,\end{aligned}$$

where the second step follows from plugging in the definition of x_i and y_i in Eq. (55), and letting $L_2 = \max_x \psi''(x)$. Taking conditional expectation (over $w^{(j)}$, $v^{(j)}$, and $\widetilde{v}^{(j)}$) on both sides, we get

$$\begin{aligned}\mathbb{E}[\psi(y_i) - \psi(x_i)] &\le \psi'(x_i) \cdot \frac{\mathbb{E}[w_i^{(j+1)}] - w_i^{(j)}}{v_i^{(j)}} + \frac{L_2}{2}\frac{1}{(v_i^{(j)})^2}\mathbb{E}[(w_i^{(j+1)} - w_i^{(j)})^2] \\ &= \psi'(x_i) \cdot \frac{w_i^{(j)}}{v_i^{(j)}} \cdot \beta_i + \frac{L_2}{2}\frac{(w_i^{(j)})^2}{(v_i^{(j)})^2}\gamma_i,\end{aligned}$$

where β_i and γ_i are defined as $\beta_i = \mathbb{E}[w_i^{(j+1)}]/w_i^{(j)} - 1$, $\gamma_i = \mathbb{E}[(w_i^{(j+1)}/w_i^{(j)} - 1)^2]$.

This then gives us

$$\sum_{i\in I} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i) - \psi(x_i)] \le \sum_{i\in I} g_{\pi^{-1}(i)} \cdot \psi'(x_i) \cdot \frac{w_i^{(j)}}{v_i^{(j)}} \cdot \beta_i + \sum_{i\in I} g_{\pi^{-1}(i)} \cdot \frac{L_2}{2} \cdot \frac{(w_i^{(j)})^2}{(v_i^{(j)})^2} \cdot \gamma_i. \tag{56}$$

For the term $w_i^{(j)}/v_i^{(j)}$, we note that for $i \in I$, we have

$$|w_i^{(j)}/v_i^{(j)}| = |x_i + 1| \le |x_i| + 1 \le 2, \tag{57}$$

where the first step follows the definition of x_i in Eq. (55), the second step follows from triangle inequality, and the third step follows from $|x_i| \le 1$ for $i \in I$.

Using this, we can bound the first term of Eq. (56) by

$$\sum_{i\in I} g_{\pi^{-1}(i)} \cdot \psi'(x_i) \cdot \frac{w_i^{(j)}}{v_i^{(j)}} \cdot \beta_i \leq \Big(\sum_{i\in I} (g_{\pi^{-1}(i)} \cdot \psi'(x_i) \cdot \frac{w_i^{(j)}}{v_i^{(j)}})^2 \cdot \sum_{i\in I} \beta_i^2 \Big)^{1/2}$$
$$\leq \Big(\sum_{i\in I} (2L_1 \cdot g_{\pi^{-1}(i)})^2 \cdot \sum_{i\in I} \beta_i^2 \Big)^{1/2}$$
$$\leq O(L_1) \cdot \Big(\sum_{i=1}^n g_i^2 \cdot C_1^2 \Big)^{1/2} = O(C_1 L_1 \|g\|_2), \tag{58}$$

where $L_1 = \max_x |\psi'(x)|$, the first step follows by Cauchy-Schwarz inequality, and the second step follows by $|\psi'(x_i)| \leq L_1$ and $|w_i^{(j)}/v_i^{(j)}| \leq 2$ by Eq.(57), and the third step follows from $\sum_{i=1}^n \beta_i^2 \leq C_1^2$ from the lemma statement of Lemma F.19.

For the second term of Eq. (56), we have

$$\sum_{i\in I} g_{\pi^{-1}(i)} \cdot \frac{L_2}{2} \cdot \frac{(w_i^{(j)})^2}{(v_i^{(j)})^2} \cdot \gamma_i \leq O(L_2) \cdot \sum_{i=1}^n g_{\pi^{-1}(i)} \cdot \gamma_i \leq O(L_2) \cdot \|g\|_2 \cdot \|\gamma\|_2 = O(C_2 L_2 \|g\|_2), \tag{59}$$

where the first step follows from $|w_i^{(j)}/v_i^{(j)}| \leq 2$ by Eq.(57), the second step follows from Cauchy-Schwarz inequality, and the third step follows from $\sum_{i=1}^n \gamma_i^2 \leq C_2^2$ from the lemma statement of Lemma F.19.

Now, plugging the bound of Eq. (58) and Eq. (59) into Eq. (56) and using that $L_1 = O(1)$, $L_2 = O(1/\epsilon_{\mathrm{mp}})$ (from Part 4 of Lemma F.37), we have that

$$\sum_{i\in I} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i) - \psi(x_i)] \leq O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \|g\|_2.$$

Case 2: Terms from I^c. For all $i \in I^c$, we have $|x_i| > 1$. Note that $\psi(x)$ is a constant for $x \geq 2\epsilon_{\mathrm{mp}}$, and from Assumption F.5 we assume that $\epsilon_{\mathrm{mp}} \leq 1/4$. Therefore, if $|y_i| \geq 1/2$, we have that $\psi(y_i) - \psi(x_i) = 0$. Hence, we only need to consider the $i \in I^c$ such that $|y_i| < 1/2$. For these i, we have that

$$|y_i - x_i| \geq |x_i| - |y_i| > 1/2, \tag{60}$$

where the first step follows by triangle inequality, and the second step follows from the assumptions $|x_i| > 1$ and $|y_i| < 1/2$. We also have

$$|y_i - x_i| = \Big|(w_i^{(j+1)} - w_i^{(j)})/v_i^{(j)}\Big| = \Big|w_i^{(j+1)}/v_i^{(j)}\Big| \cdot \Big|(w_i^{(j+1)} - w_i^{(j)})/w_i^{(j+1)}\Big| \leq \frac{3}{2}\Big|1 - w_i^{(j)}/w_i^{(j+1)}\Big|, \tag{61}$$

where the first step follows from the definition of y_i and x_i in Eq. (55), the third step follows from $|y_i| = |w_i^{(j+1)}/v_i^{(j)} - 1| < 1/2$ and thus $|w_i^{(j+1)}/v_i^{(j)}| < 3/2$.

Combining Eq. (60) and Eq. (61), we have that $|1 - w_i^{(j)}/w_i^{(j+1)}| > 1/3$ and hence $|w_i^{(j)}/w_i^{(j+1)}| < 2/3$ or $|w_i^{(j)}/w_i^{(j+1)}| > 4/3$, which then gives us $|w_i^{(j+1)}/w_i^{(j)} - 1| > 1/4$, but this contradicts with the lemma statement of Lemma F.19, so $|y_i| < 1/2$ is impossible.

Hence, we have

$$\sum_{i\in I^c} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i) - \psi(x_i)] = 0.$$

Combining both cases, we have the result. □

Lemma F.22. *In the j-th iteration, for any $w^{(j)}$, $v^{(j)}$, and $\widetilde{v}^{(j)}$ we have*

$$\sum_{i=1}^{n} g_i \cdot \mathbb{E}\left[\psi(w_{\pi(i)}^{(j+1)}/\widetilde{v}_{\pi(i)}^{(j)} - 1) - \psi(w_{\pi(i)}^{(j)}/\widetilde{v}_{\pi(i)}^{(j)} - 1) \;\middle|\; w^{(j)}, v^{(j)}, \widetilde{v}^{(j)}\right] = O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \|g\|_2.$$

Proof. The proof of this lemma is exactly the same as that of Lemma F.21, just replace all v with $\widetilde{v}$ in the proof of Lemma F.21. □

Note that in the proof of Lemma F.21 we do not have any requirement on v, so in fact we have the following more generalized lemma.

Lemma F.23 (Generalized "w move" lemma)**.** *Let $\{w^{(j)}\}_{j=0}^{T}$ be a random sequence that satisfies $\forall j \in \{0, ..., T-1\}$,*

$$\sum_{i=1}^{n} \Big(\mathbb{E}[w_i^{(j+1)}|w^{(j)}]/w_i^{(j)} - 1\Big)^2 \le C_1^2, \quad \sum_{i=1}^{n} \Big(\mathbb{E}[(w_i^{(j+1)}/w_i^{(j)} - 1)^2 \mid w^{(j)}]\Big)^2 \le C_2^2, \quad |w_i^{(j+1)}/w_i^{(j)} - 1| \le 1/4.$$

Let $\{v^{(j)}\}_{j=0}^{T}$ be another random sequence such that $\forall j \in [T]$, $v^{(j)}$ only depends on $w^{(j)}$ and $v^{(j-1)}$. Let ψ be defined as of Definition F.4, where the parameter $\epsilon_{\mathrm{mp}} < 1/4$. Let $g \in \mathbb{R}^n$ be a sequence that is non-increasing, i.e., $g_1 \ge g_2 \ge \cdots \ge g_n$. Let $\pi_j : [n] \to [n]$ be the sorting $\psi(w_{\pi(i)}^{(j+1)}/v_{\pi(i)}^{(j)} - 1) \ge \psi(w_{\pi(i+1)}^{(j+1)}/v_{\pi(i+1)}^{(j)} - 1)$.

Then $\forall j \in \{0, ..., T-1\}$, in the j-th iteration, for any $w^{(j)}$ and $v^{(j)}$ we have

$$\sum_{i=1}^{n} g_i \cdot \mathbb{E}\left[\psi(w_{\pi_j(i)}^{(j+1)}/v_{\pi_j(i)}^{(j)} - 1) - \psi(w_{\pi_j(i)}^{(j)}/v_{\pi_j(i)}^{(j)} - 1) \;\middle|\; w^{(j)}, v^{(j)}\right] = O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \|g\|_2.$$

F.3.4 $v, \widetilde{v}$ move

The goal of this section is to prove Lemma F.24.

Lemma F.24 ($v, \widetilde{v}$ move)**.** *In the j-th iteration, we have,*

$$\sum_{i=1}^{n} g_i \cdot (y_{\pi(i)}^{(j)} - x_{\tau(i)}^{(j+1)}) \ge \Omega\Big(\frac{\epsilon_{\mathrm{far}}^2 \cdot k_j \cdot g_{k_j}}{\epsilon_{\mathrm{mp}} \log n}\Big),$$

in which $\pi, \tau : [n] \to [n]$ are permutations such that $y_{\pi(i)}^{(j)} \ge y_{\pi(i+1)}^{(j)}$ and $x_{\tau(i)}^{(j+1)} \ge x_{\tau(i+1)}^{(j+1)}$.

Proof. **Case 1, $k_j = 0$.** When $k_j = 0$, we didn't enter the MatrixUpdate procedure, so $v^{(j+1)} = v^{(j)}$. If we further enter the PartialMatrixUpdate procedure and change $\widetilde{v}$, we have that $\forall i \in [n]$,

$$y_i^{(j)} - x_i^{(j+1)} = \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j)} - 1) - \psi(w_i^{(j+1)}/\widetilde{v}_i^{(j+1)} - 1) \ge 0,$$

where the first step follows by the definition of $x_i^{(j+1)}$ and $y_i^{(j)}$ (Definition F.15) and the fact that v do not change, and the last step follows by Corollary F.14.

Thus $y_i^{(j)} \ge x_i^{(j+1)}$, $\forall i \in [n]$. Since g_i and $y_{\pi(i)}^{(j)}$ are both non-increasing, we have

$$\sum_{i=1}^{n} g_i \cdot (y_{\pi(i)}^{(j)} - x_{\tau(i)}^{(j+1)}) \ge \sum_{i=1}^{n} g_i \cdot (y_{\tau(i)}^{(j)} - x_{\tau(i)}^{(j+1)}) \ge 0 = \Omega(\epsilon_{\mathrm{far}}^2 \cdot k_j \cdot g_{k_j}/(\epsilon_{\mathrm{mp}} \log n)),$$

where the last step follows by $k_j = 0$.

Case 2, $k_j \neq 0$. When $k_j \neq 0$, we must have entered MatrixUpdate, and by Fact F.10, we must have $k_j \geq n^a$. By Part 3 of Fact F.12, the difference between $x^{(j+1)}$ and $y^{(j)}$ is that in coordinates $i \in \pi([k_j])$, $x_i^{(j+1)}$ is cleared to 0, and in other coordinates $x_i^{(j+1)}$ is the same with $y_i^{(j)}$. So we have

$$\sum_{i=1}^{n} g_i \cdot (y_{\pi(i)}^{(j)} - x_{\tau(i)}^{(j+1)}) = \sum_{i=1}^{n} g_i \cdot (y_{\pi(i)}^{(j)} - y_{\pi(i+k_j)}^{(j)}). \tag{62}$$

Note that when the subscripts are out of range, we define $y_{\pi(n+1)}^{(j)} = \cdots = y_{\pi(n+k_j)}^{(j)} = 0$.

Part 2 of Fact F.12 shows that

$$y_{\pi(k_j)}^{(j)} \geq \epsilon_{\mathrm{far}}^2/(3200\epsilon_{\mathrm{mp}}). \tag{63}$$

Part 1 of Fact F.12 shows that either $k_j = n$ or $y_{\pi(k_j)}^{(j)} < (1 - 1/\log n) y_{\pi(k_j/1.5)}^{(j)}$. If $k_j = n$, we let $L = k_j = n$, otherwise we let $L = k_j/1.5$. The L we choose always satisfied that for all $i \in [L]$,

$$y_{\pi(i)}^{(j)} - y_{\pi(i+k_j)}^{(j)} \geq y_{\pi(L)}^{(j)} - y_{\pi(1+k_j)}^{(j)} \geq \epsilon_{\mathrm{far}}^2/(3200\epsilon_{\mathrm{mp}} \log n), \tag{64}$$

where the first step follows by $y_{\pi(i)}^{(j)}$ is non-increasing, the second step is true because:

1. In the case of $k_j = n$, we have $y_{\pi(L)}^{(j)} = y_{\pi(k_j)}^{(j)} \geq \frac{\epsilon_{\mathrm{far}}^2}{3200\epsilon_{\mathrm{mp}}}$ by Eq. (63) and $y_{\pi(k_j+1)}^{(j)} = y_{\pi(n+1)}^{(j)} = 0$.
2. In the case of $y_{\pi(k_j)}^{(j)} < (1 - 1/\log n) y_{\pi(k_j/1.5)}^{(j)}$, we have

   $$y_{\pi(L)}^{(j)} - y_{\pi(1+k_j)}^{(j)} \geq y_{\pi(k_j/1.5)}^{(j)} - y_{\pi(k_j)}^{(j)} \geq y_{\pi(k_j/1.5)}^{(j)}/\log n \geq \epsilon_{\mathrm{far}}^2/(3200\epsilon_{\mathrm{mp}} \log n),$$

   where the second step follows from the inequality of $y_{\pi(k_j)}^{(j)}$, and the third step follows from Eq. (63) and the fact that $y_{\pi(i)}^{(j)}$ is non-increasing.

Putting it all together, we have

$$\begin{aligned}
\sum_{i=1}^{n} g_i \cdot (y_{\pi(i)}^{(j)} - x_{\tau(i)}^{(j+1)}) &\geq \sum_{i=1}^{n} g_i \cdot (y_{\pi(i)}^{(j)} - y_{\pi(i+k_j)}^{(j)}) \geq \sum_{i=1}^{L} g_i \cdot (y_{\pi(i)}^{(j)} - y_{\pi(i+k_j)}^{(j)}) \\
&\geq \sum_{i=1}^{L} g_i \cdot \Big(\frac{\epsilon_{\mathrm{far}}^2}{3200\epsilon_{\mathrm{mp}} \log n}\Big) = \Omega\Big(\frac{\epsilon_{\mathrm{far}}^2 \cdot k_j \cdot g_{k_j}}{\epsilon_{\mathrm{mp}} \log n}\Big),
\end{aligned}$$

where the first step is by Eq. (62), the second step follows from $y_{\pi(i)}^{(j)}$ is non-increasing and thus all terms ≥ 0, the third step is by Eq. (64), and the last step follows from $g_L \geq g_{k_j}$ and $L = \Omega(k_j)$. □

F.3.5 ℓ_2-norm of g

Lemma F.25 (ℓ_2-norm of g)**.** *$g \in \mathbb{R}^n$ (Definition F.17) satisfies $\|g\|_2 = O(n^{-a/2} + n^{\omega-5/2})$.*

Proof. For $i \leq n^a$, we have $\sum_{i=1}^{n^a} g_i^2 = \sum_{i=1}^{n^a} n^{-2a} = n^{-a}$.

For $i > n^a$, note that there exists $i \in [n]$ such that $i > n^a$ implies $a < 1$, so we have

$$\begin{aligned}
\sum_{i=n^a+1}^{n} g_i^2 &= \sum_{i=n^a+1}^{n} i^{\frac{2(\omega-2)}{1-a}-2} \cdot n^{-\frac{2a(\omega-2)}{1-a}} = O(1) \cdot \int_{n^{a+1}}^{n} x^{\frac{2(\omega-2)}{1-a}-2} \cdot n^{-\frac{2a(\omega-2)}{1-a}} \mathrm{d}x \\
&= O(1) \cdot \max\{n^{\frac{2(\omega-2)}{1-a}-1}, n^{\frac{2a(\omega-2)}{1-a}-a}\} \cdot n^{-\frac{2a(\omega-2)}{1-a}} = O(n^{2\omega-5} + n^{-a}).
\end{aligned}$$

Therefore, we have $\|g\|_2 = O(n^{-a/2} + n^{\omega-5/2})$. □

F.4 Amortized analysis for PartialMatrixUpdate

F.4.1 Definitions

Definition F.26 (x and y for PartialMatrixUpdate)**.** *For any $j \in \{0, 1, \ldots, T-1\}$, and for any $i \in [n]$, we define $x_i^{(j)}$ and $y_i^{(j)}$ as follows:*

$$x_i^{(j)} := \psi(w_i^{(j)} / \widetilde{v}_i^{(j)} - 1), \quad y_i^{(j)} := \psi(w_i^{(j+1)} / \widetilde{v}_i^{(j)} - 1),$$

where $v^{(j)}$, $\widetilde{v}^{(j)}$ and $w^{(j)}$ are defined as of Definition F.1.

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

Definition F.27 (Sorting permutations for PartialMatrixUpdate)**.** *For any $j \in \{0, 1, \ldots, T\}$, let τ_j be the permutation that $x^{(j)}_{\tau_j(i)} \geq x^{(j)}_{\tau_j(i+1)}$, and let π_j be the permutation that $y^{(j)}_{\pi(i)} \geq y^{(j)}_{\pi(i+1)}$.*

When it is clear from the context that we are only arguing about the j-th iteration, for simplicity we assume the coordinates of vector $x^{(j)} \in \mathbb{R}^n$ are sorted such that $x_i^{(j)} \geq x_{i+1}^{(j)}$. And we use τ and π to denote the permutations such that $x^{(j+1)}_{\tau(i)} \geq x^{(j+1)}_{\tau(i+1)}$ and $y^{(j)}_{\pi(i)} \geq y^{(j)}_{\pi(i+1)}$.

Definition F.28 (g for PartialMatrixUpdate)**.** *For some $\widetilde{a} \leq \alpha \cdot a$, where α is the dual exponent of matrix multiplication, and a is the parameter for* MatrixUpdate*, we define $g \in \mathbb{R}^n$ as follows:*

$$g_i = \begin{cases} n^{-\widetilde{a}}, & \text{if } i \leq n^{\widetilde{a}}; \\ i^{\frac{a(\omega-2)}{a-\widetilde{a}}-1} \cdot n^{-\frac{a\widetilde{a}(\omega-2)}{a-\widetilde{a}}}, & \text{if } i \in (n^{\widetilde{a}}, n^a]; \\ 0 & \text{if } i > n^a. \end{cases}$$

Note that g is non-increasing. For all $\widetilde{k}_j \in (n^{\widetilde{a}}, n^a]$, $(\widetilde{k}_j \cdot g_{\widetilde{k}_j} n^{2a}) = \widetilde{k}_j^{\frac{a(\omega-2)}{a-\widetilde{a}}} \cdot n^{2a - \frac{a\widetilde{a}(\omega-2)}{a-\widetilde{a}}}$ is an upper bound of $\mathcal{T}_{\mathrm{mat}}(n^a, n^a, \widetilde{k}_j)$ of multiplying a $n^a \times n^a$ matrix with a $n^a \times \widetilde{k}_j$ matrix. For more details please refer to [GU18].

Definition F.29 (Potential function Φ for PartialMatrixUpdate)**.** *We define the potential function in the j-th iteration as*

$$\Phi_j = \sum_{i=1}^n g_i \cdot x^{(j)}_{\tau_j(i)}.$$

Note that we always have $\Phi_j \geq 0$ since $\forall i$, g_i and $x_i^{(j)}$ are both non-negative.

F.4.2 Main result

Lemma F.30 (Amortized time for PartialMatrixUpdate)**.** *Let sequences $\{w^{(j)}\}_{j=0}^T$, $\{v^{(j)}\}_{j=0}^T$, $\{\widetilde{v}^{(j)}\}_{j=0}^T$ be defined as of Definition F.1, let $\widetilde{k}_j$ be defined as of Definition F.3, and let $\{x^{(j)}\}_{j=0}^T$, $\{y^{(j)}\}_{j=0}^T$, g, Φ be defined as of Definition F.15, F.28 and F.29. If we further have the condition that the input sequence satisfies the following: $\forall j \in \{0, ..., T-1\}$*

$$\sum_{i=1}^n (\mathbb{E}[w_i^{(j+1)} | w^{(j)}] / w_i^{(j)} - 1)^2 \leq C_1^2, \quad \sum_{i=1}^n (\mathbb{E}[(w_i^{(j+1)} / w_i^{(j)} - 1)^2 \mid w^{(j)}])^2 \leq C_2^2, \quad |w_i^{(j+1)} / w_i^{(j)} - 1| \leq 1/4.$$

Then, we have that in expectation

$$\frac{1}{T}\sum_{j=1}^{T}\widetilde{k}_j g_{\widetilde{k}_j} = O\Big((C_1/\epsilon_{\mathrm{mp}} + C_2/\epsilon_{\mathrm{mp}}^2)\cdot \log n \cdot \|g\|_2\Big).$$

Further, combining with Lemma E.18, the expected amortized running time per iteration of PartialMatrixUpdate *is*

$$O\Big((C_1/\epsilon_{\mathrm{mp}} + C_2/\epsilon_{\mathrm{mp}}^2)\cdot (n^{1+a-\widetilde{a}/2} + n^{1+(\omega-3/2)a})\log n\Big).$$

Proof. Similar to the proof of Lemma F.19, we upper bound how much the potential function can increase due to changing $w^{(j)}$ to $w^{(j+1)}$ (in Section F.4.3), and also lower bound how much the potential function can decrease because of changing $\widetilde{v}^{(j)}$ to $\widetilde{v}^{(j+1)}$ (in Section F.4.4).

Similar to Lemma F.19, we have

$$\begin{aligned}
0 \le \mathbb{E}[\Phi_T] - \Phi_0 = & \sum_{j=0}^{T-1}\Bigg(\sum_{i=1}^{n} g_i \cdot \underbrace{\mathbb{E}\left[y_{\pi(i)}^{(j)} - x_i^{(j)}\right]}_{w \text{ move}} - \sum_{i=1}^{n} g_i \cdot \underbrace{\mathbb{E}\left[y_{\pi(i)}^{(j)} - x_{\tau(i)}^{(j+1)}\right]}_{\widetilde{v} \text{ move}}\Bigg) \\
\le & \sum_{j=0}^{T-1}\Big(O(C_1 + C_2/\epsilon_{\mathrm{mp}})\cdot\|g\|_2 - \Omega(\epsilon_{\mathrm{mp}}\widetilde{k}_j g_{\widetilde{k}_j}/\log n)\Big) \\
= & \, T\cdot O(C_1 + C_2/\epsilon_{\mathrm{mp}})\|g\|_2 - \sum_{j=1}^{T}\Omega(\epsilon_{\mathrm{mp}}\widetilde{k}_j g_{\widetilde{k}_j}/\log n),
\end{aligned}$$

where the third step follows from Lemma F.31 which states that $\forall w^{(j)}, v^{(j)}, \widetilde{v}^{(j)}$, we have

$$\sum_{i=1}^{n} g_i \cdot \mathbb{E}\left[y_{\pi(i)}^{(j)} - x_i^{(j)} \;\middle|\; w^{(j)}, v^{(j)}, \widetilde{v}^{(j)}\right] \le O(C_1 + C_2/\epsilon_{\mathrm{mp}})\cdot\|g\|_2,$$

then this upper bound also holds for unconditional expectation, the third step also follows from Lemma F.33 which states that $\sum_{i=1}^{n} g_i \cdot \left(y_{\pi(i)}^{(j)} - x_{\tau(i)}^{(j+1)}\right) \ge \Omega(\epsilon_{\mathrm{mp}}\widetilde{k}_j g_{\widetilde{k}_j}/\log n)$.

Therefore, we have

$$\frac{1}{T}\sum_{j=1}^{T}\widetilde{k}_j g_{\widetilde{k}_j} = O\Big((C_1/\epsilon_{\mathrm{mp}} + C_2/\epsilon_{\mathrm{mp}}^2)\cdot \log n \cdot \|g\|_2\Big).$$

Using Lemma E.18, we have that the expected amortized running time per iteration of PartialMatrixUpdate is

$$\begin{aligned}
\frac{1}{T}\sum_{j=1}^{T}\mathcal{T}_{\mathrm{mat}}(n, n^a, \widetilde{k}_j) \le & \frac{1}{T}\sum_{j=1}^{T} n^{1-a}\cdot\mathcal{T}_{\mathrm{mat}}(n^a, n^a, \widetilde{k}_j) \le \frac{n^{1+a}}{T}\sum_{j=1}^{T}\widetilde{k}_j g_{\widetilde{k}_j} \\
= & \, O\Big((C_1/\epsilon_{\mathrm{mp}} + C_2/\epsilon_{\mathrm{mp}}^2)\cdot n^{1+a}\log n\cdot\|g\|_2\Big) \\
= & \, O\Big((C_1/\epsilon_{\mathrm{mp}} + C_2/\epsilon_{\mathrm{mp}}^2)\cdot (n^{1+a-\widetilde{a}/2} + n^{1+(\omega-3/2)a})\log n\Big),
\end{aligned}$$

where the first step follows from the fact that we can always divide a $n \times n^a$ matrix into n^{1-a} copies of $n^a \times n^a$ matrices, the second step follows from Definition F.28 of g which gives that

$\mathcal{T}_{\mathrm{mat}}(n^a, n^a, \widetilde{k}_j) \leq n^{2a} \cdot \widetilde{k}_j g_{\widetilde{k}_j}$ for $n^{\widetilde{a}} \leq \widetilde{k}_j \leq n^a$, and we indeed have $\widetilde{k}_j \geq n^{\widetilde{a}}$ (Fact F.11) and $\widetilde{k}_j \leq 2n^a$ (Lemma E.1) when entering PartialMatrixUpdate. (We ignore the 2 factor since it can only increase the final amortized time by a constant factor.) The fourth step follows from Lemma F.34 that $\|g\|_2 = O(n^{-\widetilde{a}/2} + n^{a\omega - 5a/2})$.

□

F.4.3 w move

The goal of this section is to prove Lemma F.31.

Lemma F.31 (w move)**.** *In the j-th iteration, for any possible values $w^{(j)}$, $v^{(j)}$, and $\widetilde{v}^{(j)}$, we have*

$$\sum_{i=1}^n g_i \cdot \mathbb{E}\left[y^{(j)}_{\pi(i)} - x^{(j)}_i \;\middle|\; w^{(j)}, v^{(j)}, \widetilde{v}^{(j)} \right] \leq O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \|g\|_2. \tag{65}$$

Proof. For simplicity, in this proof we write $\mathbb{E}[\,\cdot\,]$ as a shorthand of $\mathbb{E}[\,\cdot \mid w^{(j)}, v^{(j)}, \widetilde{v}^{(j)}\,]$.

Similar to the proof of Lemma F.20, we have

$$\sum_{i=1}^n g_i \cdot \mathbb{E}[y^{(j)}_{\pi(i)} - x^{(j)}_i] \leq \sum_{i=1}^n g_i \cdot \mathbb{E}[y^{(j)}_{\pi(i)} - x^{(j)}_{\pi(i)}] = \sum_{i=1}^n g_i \cdot \mathbb{E}[\psi(w^{(j+1)}_{\pi(i)}/\widetilde{v}^{(j)}_{\pi(i)} - 1) - \psi(w^{(j)}_{\pi(i)}/\widetilde{v}^{(j)}_{\pi(i)} - 1)]$$

where the first step follows from that the non-negative values $x^{(j)}_i$ are sorted in descending order, and g is also non-increasing, the second step follows from the definitions of $x^{(j)}$ and $y^{(j)}$ (Definition F.26.

Now $\sum_{i=1}^n g_i \cdot \mathbb{E}[y^{(j)}_{\pi(i)} - x^{(j)}_i] \leq O(C_1 + C_2/\epsilon_{\mathrm{mp}})\|g\|_2$ directly follows from Lemma F.32. □

It remains to prove the following Lemma.

Lemma F.32. *In the j-th iteration, for any $w^{(j)}$, $v^{(j)}$, and $\widetilde{v}^{(j)}$ we have*

$$\sum_{i=1}^n g_i \cdot \mathbb{E}\left[\psi(w^{(j+1)}_{\pi(i)}/\widetilde{v}^{(j)}_{\pi(i)} - 1) - \psi(w^{(j)}_{\pi(i)}/\widetilde{v}^{(j)}_{\pi(i)} - 1) \;\middle|\; w^{(j)}, v^{(j)}, \widetilde{v}^{(j)} \right] = O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \|g\|_2.$$

Proof. The proof of this lemma is exactly the same as that of Lemma F.21, just replace all v with $\widetilde{v}$ in the proof of Lemma F.21. □

F.4.4 $\widetilde{v}$ move

The goal of this section is to prove Lemma F.33.

Lemma F.33 ($\widetilde{v}$ move)**.** *In the j-th iteration, we have,*

$$\sum_{i=1}^n g_i \cdot (y^{(j)}_{\pi(i)} - x^{(j+1)}_{\tau(i)}) \geq \Omega(\epsilon_{\mathrm{mp}} \widetilde{k}_j g_{\widetilde{k}_j} / \log n).$$

Proof. **Case 1.** If we enter the MatrixUpdate procedure, we have $\widetilde{k}_j = 0$ since we won't enter the else branch in Line 13 of UpdateV (Algorithm 9). From Part 3 of Fact F.12, we know that $\forall i \leq k_j$, $\widetilde{v}^{(j+1)}_{\pi(i)} = w^{(j+1)}_{\pi(i)}$, and $\forall i > k_j$, $\widetilde{v}^{(j+1)}_{\pi(i)} = \widetilde{v}^{(j)}_{\pi(i)}$. Therefore, $\forall i \in [n]$,

$$x^{(j+1)}_i = \psi(w^{(j+1)}_i/\widetilde{v}^{(j+1)}_i - 1) \leq \psi(w^{(j+1)}_i/\widetilde{v}^{(j)}_i - 1) = y^{(j)}_i,$$

where the first and the third steps follow by the definition of $x^{(j+1)}$ and $y^{(j)}$ (Definition F.26). This means $y_i^{(j)} \geq x_i^{(j+1)}$, $\forall i \in [n]$. Since g_i and $y^{(j)}_{\pi(i)}$ are both non-increasing, we have

$$\sum_{i=1}^n g_i \cdot (y^{(j)}_{\pi(i)} - x^{(j+1)}_{\tau(i)}) \geq \sum_{i=1}^n g_i \cdot (y^{(j)}_{\tau(i)} - x^{(j+1)}_{\tau(i)}) \geq 0 = \Omega(\epsilon_{\mathrm{mp}} \widetilde{k}_j g_{\widetilde{k}_j} / \log n),$$

where the last step follows by $\widetilde{k}_j = 0$.

Case 2. If we do not enter both the MatrixUpdate and the PartialMatrixUpdate procedure, nothing happens and $x^{(j+1)}$ is the same as $y^{(j)}$, this statement also holds.

Case 3. Now we only need to consider the case where we enter the PartialMatrixUpdate procedure. By Part 3 of Fact F.13, $x^{(j+1)}$ satisfies that in coordinates $i \in \pi([\widetilde{k}_j])$, $x_i^{(j+1)} \leq \frac{\epsilon_{\mathrm{mp}}}{200 \log n}$, and in coordinates $i \notin \pi([\widetilde{k}_j])$, $x^{(j+1)}$ is the same with $y^{(j)}$. So we decompose $x^{(j+1)}$ into two pieces $x^{(j+1)} = x_1 + x_2$, where x_1 copies the values on coordinates $i \in \pi([\widetilde{k}_j])$ and has 0 on other coordinates, and x_2 copies the values on coordinates $i \notin \pi([\widetilde{k}_j])$ and has 0 on other coordinates. And when the subscripts are out of range, we define $y^{(j)}_{\pi(n+1)} = \cdots = y^{(j)}_{\pi(n+k_j)} = 0$. We have

$$\begin{aligned}
\sum_{i=1}^n g_i \cdot (y^{(j)}_{\pi(i)} - x^{(j+1)}_{\tau(i)}) = & \sum_{i=1}^n g_i \cdot (y^{(j)}_{\pi(i)} - x_{1,\tau(i)} - x_{2,\tau(i)}) \geq \sum_{i=1}^n g_i \cdot (y^{(j)}_{\pi(i)} - x_{2,\tau(i)}) - \sum_{i=1}^{\widetilde{k}_j} g_i \cdot \frac{\epsilon_{\mathrm{mp}}}{200 \log n} \\
= & \sum_{i=1}^n g_i \cdot (y^{(j)}_{\pi(i)} - y^{(j)}_{\pi(i+\widetilde{k}_j)}) - \sum_{i=1}^{\widetilde{k}_j} g_i \cdot \frac{\epsilon_{\mathrm{mp}}}{200 \log n} \\
\geq & \sum_{i=1}^n g_i \cdot (y^{(j)}_{\pi(i)} - y^{(j)}_{\pi(i+\widetilde{k}_j)}) - 1.5 \sum_{i=1}^{\widetilde{k}_j/1.5} g_i \cdot \frac{\epsilon_{\mathrm{mp}}}{200 \log n}, \qquad (66)
\end{aligned}$$

where the second step follows by $x_{1\tau(i)} \leq \frac{\epsilon_{\mathrm{mp}}}{200 \log n}$ for $\tau(i) \in \pi([\widetilde{k}_j])$ and $x_{1\tau(i)} = 0$ for $\tau(i) \notin \pi([\widetilde{k}_j])$, the third step follows by $x_{2,\tau(i)} = 0$ for $\tau(i) \in \pi([\widetilde{k}_j])$ and $x_{2,\tau(i)} = y^{(j)}_{\tau(i)}$ for $i \notin \pi([\widetilde{k}_j])$, and the last step follows by g is non-increasing.

Part 2 of Fact F.13 shows that

$$y^{(j)}_{\pi(\widetilde{k}_j)} \geq \epsilon_{\mathrm{mp}}/100 \qquad (67)$$

Part 1 of Fact F.13 shows that either $\widetilde{k}_j = n$ or $y^{(j)}_{\pi(\widetilde{k}_j)} < (1 - 1/\log n) \cdot y^{(j)}_{\pi(\widetilde{k}_j/1.5)}$. If $\widetilde{k}_j = n$, we let $L = \widetilde{k}_j = n$, otherwise we let $L = \widetilde{k}_j/1.5$. The L we choose always satisfies that for all $i \in [L]$,

$$y^{(j)}_{\pi(i)} - y^{(j)}_{\pi(i+\widetilde{k}_j)} \geq \ y^{(j)}_{\pi(L)} - y^{(j)}_{\pi(1+\widetilde{k}_j)} \geq \ \epsilon_{\mathrm{mp}}/(100 \log n), \qquad (68)$$

where the first step follows by $y^{(j)}_{\pi(i)}$ is non-increasing, the second step is true because:

1. In the case of $\widetilde{k}_j = n$, we have $y^{(j)}_{\pi(L)} = y^{(j)}_{\pi(\widetilde{k}_j)} \geq \epsilon_{\mathrm{mp}}/100$ by Eq. (67) and $y^{(j)}_{\pi(\widetilde{k}_j+1)} = y^{(j)}_{\pi(n+1)} = 0$.
2. In the case of $y^{(j)}_{\pi(\widetilde{k}_j)} < (1 - 1/\log n) \cdot y^{(j)}_{\pi(\widetilde{k}_j/1.5)}$, we have

$$y^{(j)}_{\pi(L)} - y^{(j)}_{\pi(1+\widetilde{k}_j)} \geq y^{(j)}_{\pi(\widetilde{k}_j/1.5)} - y^{(j)}_{\pi(\widetilde{k}_j)} \geq y^{(j)}_{\pi(\widetilde{k}_j)} / \log n \geq \epsilon_{\mathrm{mp}}/(100 \log n),$$

where the second step follows from the inequality of $y^{(j)}_{\pi(\widetilde{k}_j)}$, and the third step follows from Eq. (67).

Putting it all together, we have

$$\begin{aligned}
\sum_{i=1}^n g_i \cdot (y_{\pi(i)}^{(j)} - x_{\tau(i)}^{(j+1)}) \geq & \sum_{i=1}^n g_i \cdot (y_{\pi(i)}^{(j)} - y_{\pi(i+\widetilde{k}_j)}^{(j)}) - 1.5 \sum_{i=1}^{\widetilde{k}_j/1.5} g_i \cdot \frac{\epsilon_{\mathrm{mp}}}{200 \log n} \\
\geq & \sum_{i=1}^L g_i \cdot (y_{\pi(i)}^{(j)} - y_{\pi(i+\widetilde{k}_j)}^{(j)}) - 1.5 \sum_{i=1}^{\widetilde{k}_j/1.5} g_i \cdot \frac{\epsilon_{\mathrm{mp}}}{200 \log n} \\
\geq & \sum_{i=1}^L g_i \cdot (y_{\pi(i)}^{(j)} - y_{\pi(i+\widetilde{k}_j)}^{(j)} - 1.5 \frac{\epsilon_{\mathrm{mp}}}{200 \log n}) \\
\geq & \sum_{i=1}^L g_i \cdot (\frac{\epsilon_{\mathrm{mp}}}{100 \log n} - 1.5 \frac{\epsilon_{\mathrm{mp}}}{200 \log n}) = \Omega(\epsilon_{\mathrm{mp}} \cdot \widetilde{k}_j \cdot g_{\widetilde{k}_j} / \log n).
\end{aligned}$$

where first step is by Eq. (66), the second step follows from $y_{\pi(i)}^{(j)}$ is non-increasing and thus all terms ≥ 0, the third step follows from $L \geq \widetilde{k}_j/1.5$, the forth step is by Eq. (68), and the last step follows from $g_L \geq g_{\widetilde{k}_j}$ and $L = \Omega(\widetilde{k}_j)$. □

F.4.5 ℓ_2-norm of g

Lemma F.34. *$g \in \mathbb{R}^n$ (Definition F.28) satisfies $\|g\|_2 = O(n^{-\widetilde{a}/2} + n^{a\omega - 5a/2})$.*

Proof. The proof is same as that of Lemma F.25. We use n^a to replace n, and $n^{\widetilde{a}}$ to replace n^a. □

F.5 Amortized analysis for VectorUpdate

Lemma F.35 (Amortized time for VectorUpdate)**.** *Let sequences $\{h^{(j)}\}_{j=0}^T$, $\{g^{(j)}\}_{j=0}^T$, $\{\widetilde{g}^{(j)}\}_{j=0}^T$ be defined as of Definition F.2, and let p_j be defined as of Definition F.3. If we further have the condition that the input sequence satisfies the following: $\forall j \in \{0, ..., T-1\}$*

$$\sum_{i=1}^n (\mathbb{E}[h_i^{(j+1)} | h^{(j)}]/h_i^{(j)} - 1)^2 \leq C_4^2, \quad \sum_{i=1}^n (\mathbb{E}[(h_i^{(j+1)}/h_i^{(j)} - 1)^2 \mid h^{(j)}])^2 \leq C_5^2, \quad |h_i^{(j+1)}/h_i^{(j)} - 1| \leq 1/4.$$

Then, we have that in expectation

1. $\frac{1}{T} \sum_{j=1}^T p_j n^{1+o(1)} = O\big((C_4 \epsilon_{\mathrm{mp}}/\epsilon_{\mathrm{far}}^2 + C_5/\epsilon_{\mathrm{far}}^2) \cdot \log n \cdot n^{1.5+o(1)}\big)$,
2. $\frac{1}{T} \sum_{j=1}^T n^{2a} \cdot \mathbf{1}_{p_j > 0} = O\big((C_4 \epsilon_{\mathrm{mp}}/\epsilon_{\mathrm{far}}^2 + C_5/\epsilon_{\mathrm{far}}^2) \cdot \log n \cdot n^{1.5a}\big)$.

Further, combining with Lemma E.27, the expected amortized running time per iteration of VectorUpdate *is*

$$O\big((C_4 \epsilon_{\mathrm{mp}}/\epsilon_{\mathrm{far}}^2 + C_5/\epsilon_{\mathrm{far}}^2) \cdot \log n \cdot n^{1.5+o(1)}\big).$$

Proof. **Part 1.** For the first equation, we define $g \in \mathbb{R}^n$ to be $g_i = 1$, $\forall i \in [n]$. Note that g is non-increasing, $n^{1+o(1)} \cdot (p_j g_{p_j}) = p_j n^{1+o(1)}$, and $\|g\|_2 = \sqrt{n}$. Then we can use the same argument as Lemma F.19 for MatrixUpdate to prove that

$$\frac{1}{T} \sum_{j=1}^T p_j n^{1+o(1)} = O\big((C_4 \epsilon_{\mathrm{mp}}/\epsilon_{\mathrm{far}}^2 + C_5/\epsilon_{\mathrm{far}}^2) \cdot \log n \cdot n^{1+o(1)} \cdot \|g\|_2\big)$$

$$= O\big((C_4\epsilon_{\mathrm{mp}}/\epsilon_{\mathrm{far}}^2 + C_5/\epsilon_{\mathrm{far}}^2)\cdot \log n \cdot n^{1.5+o(1)}\big).$$

Part 2. For the second equation, we define $g \in \mathbb{R}^n$ to be

$$g_i = \begin{cases} n^{-a}, & i \le n^a, \\ i^{-1}, & n^a < i < n. \end{cases}$$

Note that g is non-increasing, and $n^{2a}\cdot \mathbf{1}_{p_j>0} = n^{2a}\cdot \mathbf{1}_{p_j \ge n^a} \le n^{2a}\cdot (p_j g_{p_j})$ since analogous to Fact F.10 we have that either $p_j = 0$ or $p_j \ge n^a$. We also have

$$\|g\|_2^2 \le \frac{n^a}{n^{2a}} + \int_{n^a}^n x^{-2}\mathrm{d}x = n^{-a} + n^{-a} - n^{-1} = O(n^{-a}).$$

Then we can use the same argument as Lemma F.19 for MatrixUpdate to prove that

$$\frac{1}{T}\sum_{j=1}^T n^{2a}\cdot \mathbf{1}_{p_j>0} = O\big((C_4/\epsilon_{\mathrm{mp}} + C_5/\epsilon_{\mathrm{mp}}^2)\log n \cdot n^{2a}\|g\|_2\big) = O\big((C_4/\epsilon_{\mathrm{mp}} + C_5/\epsilon_{\mathrm{mp}}^2)\cdot \log n \cdot n^{1.5a}\big).$$

Combine Part 1 and Part 2. Using Lemma E.27 and note that $n^{1.5a} \le n^{1.5+o(1)}$, we have that the expected amortized running time per iteration of VectorUpdate is

$$\frac{1}{T}\sum_{j=1}^T (p_j n^{1+o(1)} + n^{2a}\mathbf{1}_{p_j>0}) = O\big((C_4/\epsilon_{\mathrm{mp}} + C_5/\epsilon_{\mathrm{mp}}^2)\cdot \log n \cdot n^{1.5+o(1)}\big).$$

□

F.6 Amortized analysis for PartialVectorUpdate

Lemma F.36 (Amortized time for PartialVectorUpdate)**.** *Let sequences $\{h^{(j)}\}_{j=0}^T$, $\{g^{(j)}\}_{j=0}^T$, $\{\widetilde{g}^{(j)}\}_{j=0}^T$ be defined as of Definition F.2, and let $\widetilde{p}_j$ be defined as of Definition F.3. If we further have the condition that the input sequence satisfies the following: $\forall j \in \{0,...,T-1\}$*

$$\sum_{i=1}^n (\mathbb{E}[h_i^{(j+1)}|h^{(j)}]/h_i^{(j)} - 1)^2 \le C_4^2, \quad \sum_{i=1}^n (\mathbb{E}[(h_i^{(j+1)}/h_i^{(j)} - 1)^2 \mid h^{(j)}])^2 \le C_5^2, \quad |h_i^{(j+1)}/h_i^{(j)} - 1| \le 1/4.$$

Then, we have that in expectation

1. $\frac{1}{T}\sum_{j=1}^T \widetilde{p}_j n^{1+o(1)} = O\big((C_4/\epsilon_{\mathrm{mp}} + C_5/\epsilon_{\mathrm{mp}}^2)\cdot \log n \cdot n^{1.5+o(1)}\big)$,
2. $\frac{1}{T}\sum_{j=1}^T n^{2a}\cdot \mathbf{1}_{\widetilde{p}_j>0} = O\big((C_4/\epsilon_{\mathrm{mp}} + C_5/\epsilon_{\mathrm{mp}}^2)\cdot \log n \cdot n^{2a-\widetilde{a}/2}\big)$.

Further, combining with Lemma E.33, the expected amortized running time per iteration of PartialVectorUpdate *is*

$$O\big((C_4\epsilon_{\mathrm{mp}}/\epsilon_{\mathrm{far}}^2 + C_5/\epsilon_{\mathrm{far}}^2)\cdot \log n \cdot (n^{1.5+o(1)} + n^{2a-\widetilde{a}/2})\big).$$

Proof. First note that we always have $\widetilde{p}_j \le 2n^a$ by Lemma E.1.
Part 1. For the first equation, we define $g \in \mathbb{R}^n$ to be $g_i = 1$, $\forall i \in [2n^a]$, and $g_i = 0$, $\forall i \notin [2n^a]$. Note that g is non-increasing, $n^{a+o(1)}\cdot(\widetilde{p}_j g_{\widetilde{p}_j}) = \widetilde{p}_j n^{a+o(1)}$, and $\|g\|_2 = \sqrt{2n^a}$. Then we can use the same argument as Lemma F.30 for PartialMatrixUpdate to prove that

$$\frac{1}{T}\sum_{j=1}^T \widetilde{p}_j n^{a+o(1)} = O\big((C_4/\epsilon_{\mathrm{mp}} + C_5/\epsilon_{\mathrm{mp}}^2)\log n \cdot n^{a+o(1)}\cdot \|g\|_2\big) = O\big((C_4/\epsilon_{\mathrm{mp}} + C_5/\epsilon_{\mathrm{mp}}^2)\cdot \log n \cdot n^{1.5a+o(1)}\big).$$

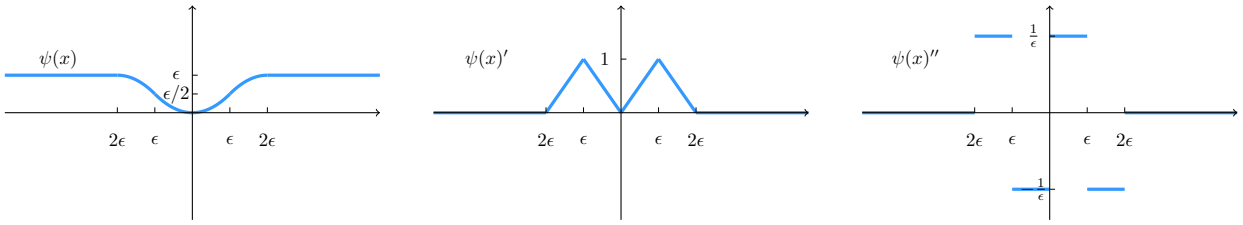


Figure 9: $\psi(x)$, $\psi(x)'$ and $\psi(x)''$. For $\epsilon_{\mathrm{mp}} \in (0,1)$, and for simplicity we use ϵ in the figures.

Part 2. For the second equation, we let $g \in \mathbb{R}^n$ be

$$g_i = \begin{cases} n^{-\widetilde{a}}, & i \leq n^{\widetilde{a}}, \\ i^{-1}, & n^{\widetilde{a}} < i \leq 2n^a, \\ 0, & i > 2n^a. \end{cases}$$

Note that g is non-increasing, and $n^{2a} \cdot \mathbf{1}_{p_j>0} = n^{2a} \cdot \mathbf{1}_{p_j \geq n^{\widetilde{a}}} \leq n^{2a} \cdot (p_j g_{p_j})$ since analogous to Fact F.11 we have that either $\widetilde{p}_j = 0$ or $\widetilde{p}_j \geq n^{\widetilde{a}}$. We also have

$$\|g\|_2^2 = \frac{n^{\widetilde{a}}}{n^{2\widetilde{a}}} + \int_{n^{\widetilde{a}}}^{2n^a} x^{-2} \mathrm{d}x = n^{-\widetilde{a}} + n^{-\widetilde{a}} - n^{-a}/2 = O(n^{-\widetilde{a}}).$$

Then we can use the same argument as the Lemma F.30 for PartialMatrixUpdate to prove that

$$\frac{1}{T} \sum_{j=1}^{T} n^{2a} \cdot \mathbf{1}_{\widetilde{p}_j>0} = O\big((C_4/\epsilon_{\mathrm{mp}} + C_5/\epsilon_{\mathrm{mp}}^2) \log n \cdot n^{2a} \|g\|_2\big) = O\big((C_4/\epsilon_{\mathrm{mp}} + C_5/\epsilon_{\mathrm{mp}}^2) \log n \cdot n^{2a-\widetilde{a}/2}\big).$$

Combine Part 1 and Part 2. Using Lemma E.33 and note that $n^{1.5a} \leq n^{1.5}$, we have that the expected amortized running time per iteration of PartialVectorUpdate is

$$\frac{1}{T} \sum_{j=1}^{T} (p_j n^{1+o(1)} + n^{2a} \cdot \mathbf{1}_{p_j>0}) = O\big((C_4/\epsilon_{\mathrm{mp}} + C_5/\epsilon_{\mathrm{mp}}^2) \cdot \log n \cdot (n^{1.5+o(1)} + n^{2a-\widetilde{a}/2})\big).$$

□

F.7 Potential function ψ

Lemma F.37 (Properties of function ψ, Lemma 5.10 of [CLS19])**.** *Let function ψ be defined as of Definition F.4. Then function ψ satisfies the following properties:*

1. *Symmetric: $\psi(-x) = \psi(x)$ and $\psi(0) = 0$;*
2. *$\psi(|x|)$ is non-decreasing;*
3. *$|\psi'(x)| = \Omega(1), \forall |x| \leq 1.5\epsilon_{\mathrm{mp}}$;*
4. *$L_1 := \max_x \psi'(x) = 1$ and $L_2 := \max_x \psi''(x) = 1/\epsilon_{\mathrm{mp}}$;*
5. *$\psi(x)$ is a constant for $|x| \geq 2\epsilon_{\mathrm{mp}}$.*

Proof. We can see that

$$\psi(x)' = \begin{cases} \frac{|x|}{\epsilon_{\mathrm{mp}}} & |x| \in [0, \epsilon_{\mathrm{mp}}] \\ \frac{2\epsilon_{\mathrm{mp}} - |x|}{\epsilon_{\mathrm{mp}}} & |x| \in (\epsilon_{\mathrm{mp}}, 2\epsilon_{\mathrm{mp}}] \\ 0 & x \in (2\epsilon_{\mathrm{mp}}, +\infty) \end{cases} \quad \text{and} \quad \psi(x)'' = \begin{cases} \frac{1}{\epsilon_{\mathrm{mp}}} & x \in [0, \epsilon_{\mathrm{mp}}] \cup [-2\epsilon_{\mathrm{mp}}, -\epsilon_{\mathrm{mp}}] \\ -\frac{1}{\epsilon_{\mathrm{mp}}} & x \in (\epsilon_{\mathrm{mp}}, 2\epsilon_{\mathrm{mp}}] \cup [-\epsilon_{\mathrm{mp}}, 0] \\ 0 & x \in (2\epsilon_{\mathrm{mp}}, +\infty) \end{cases}$$

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

Algorithm 17 Main algorithm

procedure MAIN($A, b, c, \delta, a, \widetilde{a}$) ▷ Theorem G.3
▷ A, b, c are inputs of LP
▷ δ is the accuracy parameter
  $\epsilon_{\mathrm{mp}} \leftarrow 10^{-5}/\log n$, $\epsilon_{\mathrm{far}} \leftarrow \epsilon_{\mathrm{mp}}/100\log n$, $\epsilon \leftarrow 10^{-7}/\log n$
  $\lambda \leftarrow 40\log n$, $b_{\mathrm{sketch}} \leftarrow 10^{12}\sqrt{n}\log^8 n/\epsilon_{\mathrm{mp}}^2$, $L_{\mathrm{sketch}} \leftarrow n^{1/2+o(1)}$
  $\delta \leftarrow \min\{\frac{\delta}{2}, \frac{1}{\lambda}\}$
  Create L_{sketch} sketching matrices $R_1, R_2, \cdots, R_{L_{\mathrm{sketch}}} \in \mathbb{R}^{b_{\mathrm{sketch}} \times n}$ ▷ Lemma B.14
  Let $R = [R_1^\top, R_2^\top, \cdots, R_{L_{\mathrm{sketch}}}^\top]^\top$
  Modify the linear program and obtain an initial x and s.
  Let mp_t, mp_Φ be projection maintenance data structures.
  Let $f_t : x \mapsto \sqrt{x}$, and $f_\Phi : x \mapsto \lambda \sinh(\lambda(x-1))/\sqrt{x}$
  $t \leftarrow 1$
  mp_t.INITIALIZE($f_t, \epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b_{\mathrm{sketch}}, L_{\mathrm{sketch}}, A, \frac{x}{s}, xs, R$) ▷ Algorithm 5
  mp_Φ.INITIALIZE($f_\Phi, \epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b_{\mathrm{sketch}}, L_{\mathrm{sketch}}, A, \frac{x}{s}, \frac{xs}{t}, R$) ▷ Algorithm 5
  while $t > \delta^2/(32n^3)$ **do**
    $t^{\mathrm{new}} \leftarrow (1 - \frac{\epsilon}{3\sqrt{n}})t$
    repeat
      $\delta_x, \delta_s \leftarrow$ ONESTEPCENTRALPATH($\mathrm{mp}_t, \mathrm{mp}_\Phi, x, s, t, t^{\mathrm{new}}$) ▷ Algorithm 3
      if the L_{sketch} sketching matrices are used up **then**
        re-initialize mp_t and mp_Φ with new skeching matrices.
      end if
    until $\|x^{-1}\delta_x\|_\infty \le 3\epsilon$, $\|s^{-1}\delta_s\|_\infty \le 3\epsilon$
    $x^{\mathrm{new}} \leftarrow x + \delta_x$, $s^{\mathrm{new}} \leftarrow s + \delta_s$
    if $\Phi_\lambda(x^{\mathrm{new}}s^{\mathrm{new}}/t - 1) > n^3$ **then**
      $(x^{\mathrm{new}}, s^{\mathrm{new}}) \leftarrow$ CLASSICALSTEP(x, s, t^{new}) ▷ Use the central path step of [Vai89].
      Construct sketching matrices R similar as before.
      mp_t.INITIALIZE($f_1, \epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b_{\mathrm{sketch}}, L_{\mathrm{sketch}}, A, \frac{x^{\mathrm{new}}}{s^{\mathrm{new}}}, x^{\mathrm{new}}s^{\mathrm{new}}, R$) ▷ Algorithm 5
      mp_Φ.INITIALIZE($f_2, \epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b_{\mathrm{sketch}}, L_{\mathrm{sketch}}, A, \frac{x^{\mathrm{new}}}{s^{\mathrm{new}}}, \frac{x^{\mathrm{new}}s^{\mathrm{new}}}{t}, R$) ▷ Algorithm 5
    end if
    $x \leftarrow x^{\mathrm{new}}$, $s \leftarrow s^{\mathrm{new}}$, $t \leftarrow t^{\mathrm{new}}$
  end while
  Return an approximate solution of the original linear program
end procedure

G Combining data structure with optimization

In this section we combine the results of optimization and data structure to prove Theorem 4.1.

Lemma G.1. *During the Main algorithm (Algorithm 17), we have the following guarantees:*

1. *Assumption B.1, B.26, and F.5 about the error parameters are always satisfied.*
2. *Assumption B.5 that $\widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$, $\widetilde{w} \approx_{\epsilon_{\mathrm{mp}}} \overline{w}$, and $\overline{\mu} \approx_{0.1} t$ is always satisfied.*
3. *The* CLASSICALSTEP *(line 25 of Algorithm 17) is executed with probability at most $\frac{10}{n^2}$ in each iteration.*
4. *In expectation the repeat-loop on line 17 of Algorithm 17 is executed at most 2 times.*

Notation	ϵ	ϵ_{mp}	ϵ_{far}	λ	b
Choice	$10^{-7}/\log n$	$10^{-5}/\log n$	$10^{-7}/\log^2 n$	$40\log n$	$10^{22}\sqrt{n}\log^{10} n$

Table 15: Extension of Table 7. Summary of choice of ϵ, ϵ_{mp}, ϵ_{far}, λ and b. Assigned in Main procedure (Algorithm 17). They are used to prove Theorem G.3.

Proof. **Part 1.** After plugging in the parameters in Table 15, it is straightforward to see that the constraints stated in Assumption B.1, B.26, and F.5 are all satisfied.

Part 2. The assumption that $\widetilde{\mu} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$ follows from Part (b) of the correctness of UpdateQuery in Theorem C.9 that $h^{\mathrm{appr}} \approx_{\epsilon_{\mathrm{mp}}} h^{\mathrm{new}}$, and the assumption that $\widetilde{w} \approx_{\epsilon_{\mathrm{mp}}} \overline{w}$ follows from Part (a) of the correctness of UpdateQuery in Theorem C.9 that $w^{\mathrm{appr}} \approx_{\epsilon_{\mathrm{mp}}} w^{\mathrm{new}}$.

Finally, whenever $\Phi_\lambda(xs/t-1) > n^3$, the main algorithm runs the procedure ClassicalStep, and the (x,s) returned by ClassicalStep is guaranteed to satisfy $xs \approx_{0.01} t$ (see [Vai89]). Also, if $\Phi_\lambda(xs/t-1) \leq n^3$, we have that $e^{\lambda|x_i s_i/t-1|} \leq n^3$, and since $\lambda \geq 30\log n$ (Part 2 of Assumption B.26), we have $|x_i s_i/t - 1| \leq 0.1$. Thus $\overline{\mu} \approx_{0.1} t$ is always satisfied as well.

Part 3. Let $\Phi^{(i)} = \Phi_\lambda(\frac{x^{(i)}s^{(i)}}{t^{(i)}} - 1)$ denote the value of the potential function in the i-th iteration. We use induction to prove $\mathbb{E}[\Phi^{(i)}] \leq 10n$, for all i. In the beginning of the main algorithm, $x_i s_i = 1 = t$, $\forall i \leq n$. Therefore in the base case, $\Phi^{(0)} = n < 10n$. If the algorithm executes ClassicalStep in the i-th iteration, ClassicalStep outputs x and s that $xs \approx_{0.01} t$, and since $\lambda \leq 60\log n$ (Part 7 of Assumption B.26), $\Phi^{(i)} \leq 10n$. If the algorithm doesn't execute ClassicalStep, Lemma B.27 gives us $\mathbb{E}[\Phi^{(i)}] \leq (1 - \frac{\lambda\epsilon}{15\sqrt{n}})\,\mathbb{E}[\Phi^{(i-1)}] + \frac{\lambda\epsilon}{15\sqrt{n}} 10n$. Therefore $\mathbb{E}[\Phi^{(i)}] \leq 10n$ since we have $\mathbb{E}[\Phi^{(i-1)}] \leq 10n$ from induction hypothesis. Then using Markov's inequality we have $\Pr[\Phi^{(k)} > n^3] \leq 10/n^2$. Thus the ClassicalStep on line 25 of Algorithm 17 is executed with probability at most $10/n^2$ in each iteration.

Part 4. From Part 4 of Lemma B.16 we have that $\|x^{-1}\delta_x\|_\infty > 3\epsilon$ and $\|s^{-1}\delta_s\|_\infty > 3\epsilon$ each happens with probability at most $1/n^4$. Thus in expectation the repeat-loop on line 17 of Algorithm 17 is executed at most 2 times. □

Lemma G.2. *For $\epsilon \in (0, 1/10000)$, $\epsilon_{\mathrm{mp}} \in (0, 1/10000)$, and $\epsilon_{\mathrm{far}} = \epsilon_{\mathrm{mp}}/100\log n$, each iteration of* Main *(Algorithm 17) takes*

$$O^*\Big((\epsilon/\epsilon_{\mathrm{mp}}) \cdot (n^{\omega-1/2} + n^{2-a/2} + n^{1+a-\widetilde{a}/2}) + n^{(\omega-1)\widetilde{a}+a} + n^{1+b}\Big)$$

expected amortized time per iteration, where ω is the exponent of matrix multiplication, α is the dual exponent of matrix multiplication, $0 \leq a \leq \alpha$ and $0 \leq \widetilde{a} \leq a\alpha$ are the thresholds used by the data structure, and n^b is the sketching size. O^* notation hides all $n^{o(1)}$ terms.

Proof. Part 3 of Lemma G.1 shows that in each iteration ClassicalStep is executed with probability at most $O(1/n^2)$. Since the cost of ClassicalStep is $O(n^{2.5})$, the amortized cost of executing ClassicalStep is $O(n^{0.5})$ for one iteration.

Part 4 of Lemma G.1 shows that in expectation OneStepCentralPath is executed at most 2 times in each iteration. So now we only need to bound the running time of the procedure OneStepCentralPath (Algorithm 3). In the procedure OneStepCentralPath, we call the procedure UpdateQuery of the data structures (Algorithm 8) two times. Since the time analysis of these two data structure is the same, we are going to focus on one of them. Also note that the running time of UpdateQuery is the sum of that of MatrixUpdate, PartialMatrixUpdate, VectorUpdate, PartialVectorUpdate, and Query, so we analyze them one by one.

From what we proved in Lemma B.28 and Lemma B.29, we have $C_1 = C_4 = \Theta(\epsilon)$ and $C_2 = C_5 = \Theta(\epsilon^2)$ and $C_3 = C_6 = \Theta(\epsilon) < 1/4$.

By Theorem C.9 and plugging in $\epsilon_{\mathrm{far}} = \epsilon_{\mathrm{mp}}/100 \log n$, the expected amortized cost per iteration of the following procedures are as follows:

$$\begin{aligned}
\textsc{MatrixUpdate} &= O^*((C_1\epsilon_{\mathrm{mp}}/\epsilon_{\mathrm{far}}^2 + C_2/\epsilon_{\mathrm{far}}^2) \cdot (n^{2-a/2} + n^{\omega-1/2})) \\
&= O^*((\epsilon/\epsilon_{\mathrm{mp}}) \cdot (n^{2-a/2} + n^{\omega-1/2})) \\
\textsc{PartialMatrixUpdate} &= O^*((C_1/\epsilon_{\mathrm{mp}} + C_2/\epsilon_{\mathrm{mp}}^2) \cdot (n^{1+a-\widetilde{a}/2} + n^{1+(\omega-3/2)a})) \\
&= O^*((\epsilon/\epsilon_{\mathrm{mp}}) \cdot (n^{1+a-\widetilde{a}/2} + n^{1+(\omega-3/2)a})) \\
\textsc{VectorUpdate} &= O^*((C_4\epsilon_{\mathrm{mp}}/\epsilon_{\mathrm{far}}^2 + C_5/\epsilon_{\mathrm{far}}^2) \cdot n^{1.5}) \\
&= O^*((\epsilon/\epsilon_{\mathrm{mp}}) \cdot n^{1.5}) \\
\textsc{PartialVectorUpdate} &= O^*((C_4\epsilon_{\mathrm{mp}}/\epsilon_{\mathrm{far}}^2 + C_5/\epsilon_{\mathrm{far}}^2) \cdot (n^{1.5} + n^{2a-\widetilde{a}/2})) \\
&= O^*((\epsilon/\epsilon_{\mathrm{mp}}) \cdot (n^{1.5} + n^{2a-\widetilde{a}/2})) \\
\textsc{Query} &= O^*(\mathcal{T}_{\mathrm{mat}}(n^{\widetilde{a}}, n^a, n^{\widetilde{a}}) + n^{1+b}) \\
&= O^*(n^{(\omega-1)\widetilde{a}+a} + n^{1+b}).
\end{aligned}$$

So the overall expected amortized cost of one iteration is

$$\begin{aligned}
&\textsc{MatrixUpdate} + \textsc{PartialMatrixUpdate} \\
&+ \textsc{VectorUpdate} + \textsc{PartialVectorUpdate} + \textsc{Query} \\
&= O^*\Big(\underbrace{(\epsilon/\epsilon_{\mathrm{mp}}) \cdot (n^{2-a/2} + n^{\omega-1/2})}_{\textsc{MatrixUpdate}} + \underbrace{(\epsilon/\epsilon_{\mathrm{mp}}) \cdot (n^{1+a-\widetilde{a}/2} + n^{1+(\omega-3/2)a})}_{\textsc{PartialMatrixUpdate}} \\
&+ \underbrace{(\epsilon/\epsilon_{\mathrm{mp}}) \cdot n^{1.5}}_{\textsc{VectorUpdate}} + \underbrace{(\epsilon/\epsilon_{\mathrm{mp}}) \cdot (n^{1.5} + n^{2a-\widetilde{a}/2})}_{\textsc{PartialVectorUpdate}} + \underbrace{n^{(\omega-1)\widetilde{a}+a} + n^{1+b}}_{\textsc{Query}} \Big) \\
&= O^*\Big((\epsilon/\epsilon_{\mathrm{mp}}) \cdot (n^{\omega-1/2} + n^{2-a/2} + n^{1+a-\widetilde{a}/2}) + n^{(\omega-1)\widetilde{a}+a} + n^{1+b} \Big),
\end{aligned}$$

where in the last step we use $\omega \geq 2$, $\widetilde{a} \leq a \leq 1$ and $\omega - 1/2 = 1 + (\omega - 3/2) \geq 1 + (\omega - 3/2)a$. □

Now we are ready to prove the main theorem of this paper.

Theorem G.3 (Restate Theorem 4.1, Main result, third improvement)**.** *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 $\delta \in (0, 1]$, $\textsc{Main}(A, b, c, \delta)$ (Algorithm 17) outputs $x \geq 0$ such that

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

in expected time

$$\widetilde{O}(n^{\omega+o(1)} + n^{2.5-a/2+o(1)} + n^{1.5+a-\widetilde{a}/2+o(1)} + n^{0.5+a+(\omega-1)\widetilde{a}}) \cdot \log(n/\delta)$$

where ω is the exponent of matrix multiplication, α is the dual exponent of matrix multiplication, and $0 < a \leq \alpha$.

In the ideal case when $\omega = 2$ and $\alpha = 1$. The running time is $\widetilde{O}(n^{2+1/18})$. For general $2 \leq \omega \leq 3$ and $0 \leq \alpha \leq 1$, the running time is $O^*(n^\omega + n^{2.5-a/2} + n^{(8+\sqrt{19})/6}) = O^*(n^\omega + n^{2.5-a/2} + n^{2.06})$.

Proof. We use the parameters of Table 15 to prove the theorem. Since t is decreasing by a $(1-\frac{\epsilon}{3\sqrt{n}})$ factor, the Main algorithm will take $O(\epsilon^{-1}\sqrt{n}\log(n/\delta))$ iterations in total.

Thus the total running time is

$$\begin{aligned}
&\#\text{iterations} \cdot \text{cost per iteration} \\
&= O(\epsilon^{-1}\sqrt{n}\log(n/\delta)) \cdot O^*\Big((\epsilon/\epsilon_{\text{mp}}) \cdot (n^{\omega-1/2} + n^{2-a/2} + n^{1+a-\widetilde{a}/2}) + n^{(\omega-1)\widetilde{a}+a} + n^{1+b}\Big) \\
&= O^*\Big(\epsilon_{\text{mp}}^{-1}(n^{\omega} + n^{2.5-a/2} + n^{1.5+a-\widetilde{a}/2}) + \epsilon^{-1}(n^{0.5+(\omega-1)\widetilde{a}+a} + n^{1.5+b})\Big) \cdot \log(n/\delta).
\end{aligned}$$

By plugging in the parameters $\epsilon = O(1/\log n)$, $\epsilon_{\text{mp}} = O(1/\log n)$, and $b = \sqrt{n}\log^{10} n$ (see Table 15), the above running time becomes

$$O^*\Big(n^{\omega} + n^{2.5-a/2} + n^{1.5+a-\widetilde{a}/2} + n^{0.5+(\omega-1)\widetilde{a}+a}\Big) \cdot \log(n/\delta), \tag{69}$$

and recall that parameters a and $\widetilde{a}$ need to satisfy that $a \le \alpha$ and $\widetilde{a} \le \alpha a$.

Therefore, in the ideal case where $\omega = 2$ and $\alpha = 1$, we can choose $a = \frac{8}{9}$ and $\widetilde{a} = \frac{2}{3}$, and we have $2.5 - a/2 = 1.5 + a - \widetilde{a}/2 = 0.5 + (\omega-1)\widetilde{a} + a = 2 + 1/18$, so the above running time simplifies to

$$O^*(n^{2+1/18+o(1)}) \cdot \log(n/\delta).$$

For general ω and α, the parameters are optimized as follows:

$$a = \begin{cases} \alpha, & \text{if } \alpha \le \frac{4w}{3(2w-1)}, \\ \frac{4w}{3(2w-1)}, & \text{o.w.} \end{cases} \qquad \widetilde{a} = \begin{cases} \min\{\alpha^2, \frac{2}{2\omega-1}\}, & \text{if } \alpha \le \frac{4\omega}{3(2\omega-1)}, \\ \frac{2}{2\omega-1}, & \text{o.w.} \end{cases}$$

Here we prove the final running time by discussing two cases.

1. In the first case where $\alpha \le \frac{4\omega}{3(2\omega-1)}$, we have $a = \alpha$ and $\widetilde{a} \le \alpha^2 = \alpha a$.

   If $\widetilde{a} = \alpha^2$, then $1.5 + \alpha - \widetilde{a}/2 = 1.5 + \alpha - \alpha^2/2 \le 2.5 - \alpha/2$ since $\alpha \le 1$.

   If $\widetilde{a} = \frac{2}{2\omega-1}$, then $1.5 + \alpha - \widetilde{a}/2 = 1.5 + \alpha - 1/(2\omega-1) \le 2.5 - \alpha/2$, since $\alpha \le \frac{4\omega}{3(2\omega-1)}$ and $\frac{4\omega}{3(2\omega-1)}$ is the value that balances the two terms. Thus the following inequality holds:

   $$n^{1.5+a-\widetilde{a}/2} \le n^{2.5-a/2}.$$

   We also have $0.5 + (\omega-1)\widetilde{a} + a \le 1.5 + a - \widetilde{a}/2$, since $\widetilde{a} \le \frac{2}{2\omega-1}$ and $\frac{2}{2\omega-1}$ is the value that balances these two terms. Thus the following inequality also holds:

   $$n^{0.5+(\omega-1)\widetilde{a}+a} \le n^{1.5+a-\widetilde{a}/2}.$$

   Therefore the running time of Eq. (69) is dominated by $O(n^{\omega} + n^{2.5-\alpha/2})$ in the first case.

2. In the second case where $\alpha > \frac{4\omega}{3(2\omega-1)}$, we have $a = \frac{4\omega}{3(2\omega-1)} \le \alpha$, and $\widetilde{a} = \frac{2}{2\omega-1} \le (\frac{4\omega}{3(2\omega-1)})^2 \le \alpha a$, where the second step follows since $\omega \ge 2$. With these parameters, we have that

   $$2.5 - a/2 = 1.5 + a - \widetilde{a}/2 = 0.5 + (\omega-1)\widetilde{a} + a = \frac{13}{6} - \frac{1}{3(2\omega-1)}.$$

   Therefore in the second case the running time of Eq. (69) is dominated by

   $$O(n^{\omega} + n^{\frac{13}{6} - \frac{1}{3(2\omega-1)}}) \le O(n^{\omega} + n^{(8+\sqrt{19})/6}),$$

   where $(8+\sqrt{19})/6 \approx 2.0598 \le 2.06$ is the solution of equation $\omega = \frac{13}{6} - \frac{1}{3(2\omega-1)}$.

Thus the running time in Eq. (69) is always upper bounded by

$$O^*\Big(n^\omega + n^{2.5-\alpha/2} + n^{(8+\sqrt{19})/6}\Big) \cdot \log(n/\delta).$$

□

H Multi-level with more details

In this section we provide more details for Section 2.

H.1 LU-decomposition of Woodbury identity when $K = 3$

The LU-decomposition of matrix $D = \begin{bmatrix} M & U_2 & U_3 \\ V_2^\top & -C_2^{-1} & 0 \\ V_3^\top & 0 & -C_3^{-1} \end{bmatrix}$ where the diagonal blocks of L are identity matrices is

$$\begin{aligned} D &= \begin{bmatrix} I & 0 & 0 \\ V_2^\top M^{-1} & I & 0 \\ V_3^\top M^{-1} & 0 & I \end{bmatrix} \cdot \begin{bmatrix} M & U_2 & U_3 \\ 0 & -C_2^{-1} - V_2^\top M^{-1} U_2 & -V_2^\top M^{-1} U_3 \\ 0 & -V_3^\top M^{-1} U_2 & -C_3^{-1} - V_3^\top M^{-1} U_3 \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} I & 0 & 0 \\ V_2^\top M^{-1} & I & 0 \\ V_3^\top M^{-1} & 0 & I \end{bmatrix} \cdot \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & -V_3^\top M^{-1} U_2 B^{-1} & I \end{bmatrix}}_{L} \\ &\quad \cdot \underbrace{\begin{bmatrix} M & U_2 & U_3 \\ 0 & B & -V_2^\top M^{-1} U_3 \\ 0 & 0 & -C_3^{-1} - V_3^\top M^{-1} U_3 - V_3^\top M^{-1} U_2 B^{-1} V_2^\top M^{-1} U_3 \end{bmatrix}}_{U}, \end{aligned} \tag{70}$$

where $B := -C_2^{-1} - V_2^\top M^{-1} U_2$.

H.2 Online low-rank inverse and the oMV conjecture.

The following *static* data structure problem has received significant attention recently (see [HKNS15, BNS19] and references therein).

Definition H.1 (Online matrix-vector multiplication (oMV)). *Preprocess a (fixed) matrix $M \in \mathbb{R}^{n \times n}$ so that, given an online sequence of T vectors $h_t \in \mathbb{R}^n$ (arriving one by one), the data structure can efficiently output Mh_t (exactly) before the next iteration $t+1$.*

The oMV Conjecture [HKNS15] states that with poly(n) preprocessing time, the (amortized) query time of any word-RAM data structure for oMV is at least $t_q > n^{2-o(1)}$. Note that this is in sharp contrast to the *offline* setting where the vectors $\{h_t\}_{t \in T}$ are given as a batch, in which case fast-matrix multiplication can achieve $n^{\omega-1} < n^{1.37}$ query time on average (assuming $T = n$, say). Now consider the following static problem:

Definition H.2 (Online low-rank inverse multiplication). *Preprocess a fixed matrix $M \in \mathbb{R}^{n \times n}$ and a fixed vector h, so that given an online sequence of T pairs of vectors $u_t, v_t \in \mathbb{R}^n$, the data structure can efficiently output $(M + u_t v_t^\top)^{-1} h$ (exactly) before the next iteration $t+1$.*

Perhaps surprisingly, we prove that these problems are essentially equivalent in the word-RAM model with polynomial preprocessing time. We first show that oMV is at least as hard as the problem in Definition H.2:

Lemma H.3. *If there is a data structure A with polynomial preprocessing time for oMV (Definition H.1) with worst case query time $\mathcal{T}_A(n,T)$, then there is a data structure A' for the online low-rank inverse problem in Definition H.2 with polynomial preprocessing time and $O(\mathcal{T}_A(n,T))$ query time.*

Proof. We design A' as follows. In the preprocessing time, we use $O(n^\omega)$ time to pre-compute the vector $x := M^{-1} \cdot h \in \mathbb{R}^n$ and run the oMV data structure A on the input matrix M^{-1}. Recall in the query stage, we are given $u_t, v_t \in \mathbb{R}^n$. By Woodbury's identity, the solution $(M + u_t v_t^\top)^{-1} \cdot h$ can be written as

$$(M + u_t v_t^\top)^{-1} h = M^{-1} h - M^{-1} u_t (1 + v_t^\top M^{-1} u_t)^{-1} v_t^\top M^{-1} \cdot h.$$

Thus, using a single invocation of the query algorithm of A, we can compute the product $y := M^{-1} \cdot u_t$, and the remaining calculation is

$$M^{-1} h - M^{-1} u_t (1 + v_t^\top M^{-1} u_t)^{-1} v_t^\top M^{-1} \cdot h = x - y(1 + v_t^\top \cdot y)^{-1} v_t^\top \cdot x,$$

which only involves vector inner product calculation and can be done in $O(n)$ time.

Therefore, A' takes $O(\mathcal{T}_A(n,T)) + O(Tn)$ worst case query time. The lemma is proved by observing that $\mathcal{T}_A(n,T)$ is at least $\Omega(Tn)$. □

We proceed to the other direction of the proof.

Lemma H.4. *Given a word-RAM data structure B with polynomial preprocessing time for the online low-rank inverse problem, with worst-case query time $\mathcal{T}_B(n,T)$, there is a data structure B' for the oMV problem with polynomial preprocessing time and $O(\mathcal{T}_B(n,T))$ worst case query time.*

Proof. We construct B' as follows: Given the input M in Definition H.1, the data structure first compute M^{-1}, and finds an arbitrary vector h for which $h^\top M h \neq 0$, then pre-computes $x := Mh$ and $y := h^\top M$. This takes $O(n^\omega)$ preprocessing time. We can now invoke the preprocessing function of the data structure B (for online low-rank inverse) with inputs $M \leftarrow M^{-1}$, $h \leftarrow h$. In each iteration $t \in [T]$, given the vector h_t of Definition H.1, invoke B with query vector $u_t \leftarrow h_t$, $v_t \leftarrow h$ to get an answer g. Note that by Woodbury's identity

$$g = (M^{-1} + u_t v_t^\top)^{-1} h = Mh - Mu_t (1 + v_t^\top M u_t)^{-1} v_t^\top M \cdot h = Mh - Mh_t (1 + h^\top M u_t)^{-1} h^\top M \cdot h.$$

Then B' outputs the query answer

$$(x - g) \cdot \frac{1 + y \cdot h_t}{y \cdot h} = (Mh - g) \cdot \frac{1 + (h^\top M) h_t}{h^\top M \cdot h} = Mh_t,$$

which only involves vector inner product calculation and can be done in $O(n)$ time.

Therefore, B' takes polynomial preprocessing time and $O(\mathcal{T}_B(n,T)) + O(Tn)$ worst case query time. The lemma is proved by observing that $\mathcal{T}_B(n,T)$ is at least $\Omega(Tn)$. □

As a corollary, we get that the following problem is at least as hard as the oMV problem:

Definition H.5 (Online cumulative low-rank inverse)**.** *Preprocess a fixed matrix $M \in \mathbb{R}^{n \times n}$ and a fixed vector h, so that given an online sequence of T pairs of vectors $u_t, v_t \in \mathbb{R}^n$, the data structure can efficiently output $(M + \sum_{i=1}^t u_i v_i^\top)^{-1} h$ (exactly) before the next iteration $t + 1$.*

H.3 Optimizing the parameters of Eq. (3)

Claim H.6. *For any positive integer n, K, the following equation holds.*

$$\min_{n=n_1\geq n_2\geq\cdots n_{K-1}\geq n_K\geq 1}\left(\sum_{k=1}^{K-1} n\cdot n_k/\sqrt{n_{k+1}}+n_{K-1}\cdot n_K\right)=O(K)\cdot n^{1.5+\frac{1}{6\cdot(2^{K-1}-1)}}.$$

Proof. Denote $p=\frac{1}{6\cdot(2^{K-1}-1)}$. Define a_k as the exponent of n_k such that $n_k=n^{a_k}$. Then it is equivalent to finding $a_1,\cdots,a_K\in\mathbb{R}$ such that the following three conditions hold:

1. $1=a_1\geq a_2\geq\cdots\geq a_{K-1}\geq a_K\geq 0$.
2. $1+a_i-a_{i+1}/2\leq 1.5+p$, $\forall i\in[K-1]$.
3. $a_{K-1}+a_K\leq 1.5+p$.

The optimal solution is given by $a_i=1-1/(3\cdot 2^{K-i})+(2-2^{-K+i+1})\cdot p$, $\forall i\in[K]$. Now we prove that the three conditions are satisfied with this solution.

Condition 1.

$$a_1=1-\frac{1}{3\cdot 2^{K-1}}+\frac{2-2^{-K+2}}{6\cdot(2^{K-1}-1)}=1-\frac{1}{3\cdot 2^{K-1}}+\frac{(2^{K-1}-1)/2^{K-1}}{3\cdot(2^{K-1}-1)}=1.$$

Since a_i is decreasing with i, $a_i\geq a_{i+1}$ is also satisfied. Finally, $a_K=\frac{2}{3}\geq 0$.

Condition 2. For all $i\in[K-1]$,

$$\begin{aligned}1+a_i-a_{i+1}/2&=1+(1-\frac{1}{3\cdot 2^{K-i}}+(2-2^{-K+i+1})\cdot p)-(1-\frac{1}{3\cdot 2^{K-i-1}}+(2-2^{-K+i+2})\cdot p)/2\\&=1.5-\frac{1}{3\cdot 2^{K-i}}+(2-2^{-K+i+1})\cdot p+\frac{1}{3\cdot 2^{K-i}}-(1-2^{-K+i+1})\cdot p\\&=1.5+p.\end{aligned}$$

Condition 3. $a_{K-1}+a_K=(1-\frac{1}{6}+p)+\frac{2}{3}=1.5+p$. □

I A feasible algorithm

In previous sections, we show a fast algorithm calculating an LP solution $\overline{x}$. However, $\overline{x}$ is not always a feasible solution since we used sketching to calculate $\widehat{\delta}_x$ and hence $A\widehat{\delta}_x$ is not 0. In this section we present how to turn the output x of Theorem 4.1 to a feasible LP solution, i.e. $\|Ax-b\|_1$ is bounded. Our technique is based on the robust central path of [LSZ19], and we extend it to two level update setting. Our algorithm use G, M, u_1, u_2, u_3, u_4 to implicitly maintain the solutions $x=Gu_1+u_2$ and $s=Mu_3+u_4$. Each iteration, the algorithm updates G, M, u_1, u_2, u_3, u_4 so that x and s change by $\widetilde{\delta}_x$ and $\widetilde{\delta}_s$ in each iteration (see Definition B.4). In this way, we can postpone the expensive matrix vector multiplication to every $\sqrt{n}$ iterations. The running time of maintaining x and s is dominated by the pre-existing computations, so our algorithm still achieves the same overall running time. Also since we do not multiply the sketching matrix on the left when computing $\widetilde{\delta}_x$ and $\widetilde{\delta}_s$, $A\widetilde{\delta}_x=0$ is always satisfied and in each iteration we always have $Ax=Ax^{(0)}=b$.

We introduce a smaller error constant ϵ_{tiny} in this section.

Definition I.1. *We define* $\epsilon_{\text{tiny}}=\frac{\epsilon_{\text{mp}}\cdot\epsilon}{3200\log^3 n}$.

Algorithm 18 Feasible version of MAIN (Algorithm 17)

procedure MAIN($A, b, c, \delta, a, \widetilde{a}$) ▷ Theorem G.3+I.2, A, b, c are inputs of LP, δ is the accuracy parameter
  $\epsilon_{\mathrm{mp}} \leftarrow 10^{-5}/\log n$
  $\epsilon_{\mathrm{far}} \leftarrow \epsilon_{\mathrm{mp}}/100\log n$
  $\epsilon \leftarrow 10^{-7}/\log n$
  $\lambda \leftarrow 40\log n$
  $\delta \leftarrow \min\{\frac{\delta}{2}, \frac{1}{\lambda}\}$
  $b_{\mathrm{sketch}} \leftarrow 10^{12}\sqrt{n}\log^8 n/\epsilon_{\mathrm{mp}}^2$
  $L_{\mathrm{sketch}} \leftarrow n^{1/2+o(1)}$
  Create L_{sketch} sketching matrices $R_1, R_2, \cdots, R_{L_{\mathrm{sketch}}} \in \mathbb{R}^{b_{\mathrm{sketch}} \times n}$ ▷ Lemma B.14
  Let $R = [R_1^\top, R_2^\top, \cdots, R_{L_{\mathrm{sketch}}}^\top]^\top$
  Modify the linear program and obtain an initial $x^{(0)}$ and $s^{(0)}$.
  Let mp_t and mp_Φ be projection maintenance data structures.
  Let $f_t : x \mapsto \sqrt{x}$, $f_\Phi : x \mapsto \lambda \sinh(\lambda(x-1))/\sqrt{x}$
  mp_t.INITIALIZE($f_t, \epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b_{\mathrm{sketch}}, L_{\mathrm{sketch}}, A, \frac{x^{(0)}}{s^{(0)}}, x^{(0)}s^{(0)}, R$) ▷ Algorithm 21
  mp_Φ.INITIALIZE($f_\Phi, \epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b_{\mathrm{sketch}}, L_{\mathrm{sketch}}, A, \frac{x^{(0)}}{s^{(0)}}, \frac{x^{(0)}s^{(0)}}{t}, R$) ▷ Algorithm 21
  Global $\overline{x}, \overline{s}, x, s, t, t^{\mathrm{new}}, w^{\mathrm{old}}$
  $\overline{x} \leftarrow x \leftarrow x^{(0)}$, $\overline{s} \leftarrow s \leftarrow s^{(0)}$, $w^{\mathrm{old}} \leftarrow x^{(0)}s^{(0)}$
  $t^{\mathrm{old}} \leftarrow t \leftarrow 1$, $j \leftarrow 0$
  while $t > \delta^2/(32n^3)$ **do**
    $t^{\mathrm{new}} \leftarrow (1 - \frac{\epsilon}{3\sqrt{n}})t$, $j \leftarrow j+1$
    repeat
      $\widehat{\delta}_x, \widehat{\delta}_s, w^{\mathrm{appr}} \leftarrow$ ONESTEPCENTRALPATH($\mathrm{mp}_t, \mathrm{mp}_\Phi, t, t^{\mathrm{new}}$) ▷ Algorithm 19
      if the L_{sketch} sketching matrices are used up **then**
        re-initialize mp_t and mp_Φ with new ones.
      end if
    until $\|\overline{x}^{-1}\widehat{\delta}_x\|_\infty \le 5\epsilon$, $\|\overline{s}^{-1}\widehat{\delta}_s\|_\infty \le 5\epsilon$, if this condition is false, revoke the updates of u_1, u_2, u_3, u_4 in mp_t and mp_Φ
    $\overline{x} \leftarrow \overline{x} + \widehat{\delta}_x$, $\overline{s} \leftarrow \overline{s} + \widehat{\delta}_s$
    MAKEFEASIBLE(w^{appr})
    if $j > \sqrt{n}$ or $t < t^{\mathrm{old}}/2$ **then**
      $x \leftarrow x - (\mathrm{mp}_t.u_1 + \mathrm{mp}_t.G \cdot \mathrm{mp}_t.u_2) - (\mathrm{mp}_\Phi.u_1 + \mathrm{mp}_\Phi.G \cdot \mathrm{mp}_\Phi.u_2)$
      $s \leftarrow s + (\mathrm{mp}_t.u_3 + \mathrm{mp}_t.M \cdot \mathrm{mp}_t.u_4) + (\mathrm{mp}_\Phi.u_3 + \mathrm{mp}_\Phi.M \cdot \mathrm{mp}_\Phi.u_4)$
      mp_t.INITIALIZE($\epsilon_{mp}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b, L, A, w^{\mathrm{appr}}, h^{\mathrm{appr}}, R$)
      mp_Φ.INITIALIZE($\epsilon_{mp}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b, L, A, w^{\mathrm{appr}}, h^{\mathrm{appr}}/t, R$)
      $j \leftarrow 1$, $t^{\mathrm{old}} \leftarrow t$
    end if
    if $\Phi_\lambda(\overline{x}\overline{s}/t - 1) > n^3$ **then**
      $(\overline{x}, \overline{s}) \leftarrow$ CLASSICALSTEP($\overline{x}, \overline{s}, t^{\mathrm{new}}$) ▷ Use the central path step of [Vai89].
      $x \leftarrow \overline{x}$, $s \leftarrow \overline{s}$
      Construct sketching matrices R similar as before.
      mp_t.INITIALIZE($f_t, \epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b_{\mathrm{sketch}}, L_{\mathrm{sketch}}, A, \frac{\overline{x}}{\overline{s}}, \overline{x}\overline{s}, R$) ▷ Algorithm 21
      mp_Φ.INITIALIZE($f_\Phi, \epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b_{\mathrm{sketch}}, L_{\mathrm{sketch}}, A, \frac{\overline{x}}{\overline{s}}, \frac{\overline{x}\overline{s}}{t}, R$) ▷ Algorithm 21
    end if
    $t \leftarrow t^{\mathrm{new}}$
  end while
  return x
end procedure

Algorithm 19 Feasible version of OneStepCentralPath (Algorithm 3)

procedure OneStepCentralPath($\mathrm{mp}_t, \mathrm{mp}_\Phi, t, t^{\mathrm{new}}$) ▷ Part 1 of Theorem I.2
  $\overline{w} \leftarrow \overline{x}/\overline{s}$ ▷ $\overline{x}, \overline{s}$ is global variable
  $\overline{\mu} \leftarrow \overline{x}\overline{s}$
  $(q_{t,x}, p_{t,x}, p_{t,s}, w^{\mathrm{appr}}) \leftarrow \mathrm{mp}_t.\textsc{UpdateQuery}(\overline{w}, \overline{\mu})$ ▷ Algorithm 23
  ▷ this data-structure works with function $f_t(x) = \sqrt{x}$
  $(q_{\Phi,x}, p_{\Phi,x}, p_{\Phi,s}, w^{\mathrm{appr}}) \leftarrow \mathrm{mp}_\Phi.\textsc{UpdateQuery}(\overline{w}, \overline{\mu}/t)$ ▷ Algorithm 23
  ▷ this data-structure works with function $f_\Phi(x) = \nabla\Phi(x-1)/\sqrt{x}$
  $\widehat{\delta}_x \leftarrow q_{t,x} + q_{\Phi,x} - (p_{t,x} + p_{\Phi,x})$
  $\widehat{\delta}_s \leftarrow p_{t,s} + p_{\Phi,s}$
  $x \leftarrow x + q_{t,x} + q_{\Phi,x}$
  return $(\widehat{\delta}_x, \widehat{\delta}_s, w^{\mathrm{appr}})$
end procedure

Algorithm 20 Data structure : MakeFeasible.

data structure
procedure MakeFeasible(w^{appr}) ▷ Part 2 of Theorem I.2
  $\widehat{S} \leftarrow \{i : |w_i^{\mathrm{old}} - w_i^{\mathrm{appr}}| > w_i^{\mathrm{old}}/2\}$
  $\overline{x}_{\widehat{S}} \leftarrow x_{\widehat{S}} - ((\mathrm{mp}_t.u_1)_{\widehat{S}} + (\mathrm{mp}_t.G \cdot \mathrm{mp}_t.u_2)_{\widehat{S}}) - ((\mathrm{mp}_\Phi.u_1)_{\widehat{S}} + (\mathrm{mp}_\Phi.G \cdot \mathrm{mp}_\Phi.u_2)_{\widehat{S}})$
  $\overline{s}_{\widehat{S}} \leftarrow s_{\widehat{S}} + ((\mathrm{mp}_t.u_3)_{\widehat{S}} + (\mathrm{mp}_t.M \cdot \mathrm{mp}_t.u_4)_{\widehat{S}}) + ((\mathrm{mp}_\Phi.u_3)_{\widehat{S}} + (\mathrm{mp}_\Phi.M \cdot \mathrm{mp}_\Phi.u_4)_{\widehat{S}})$
  $w_{\widehat{S}}^{\mathrm{old}} \leftarrow w_{\widehat{S}}^{\mathrm{appr}}$
end procedure
end data structure

Algorithm 21 Data structure : feasible version of members (Algorithm 4), invariants (Assumption D.1), and Initialize (Algorithm 5)

data structure

members
  $\cdots$ ▷ We continue to have all previous members of Algorithm 4.
  $u_1, u_2, u_3, u_4 \in \mathbb{R}^n$
  $G \in \mathbb{R}^{n \times n}$
end members

invariant
  $\cdots$ ▷ We continue to maintain all previous invariants of Assumption D.1.
  $G = \widetilde{V} A^\top (AVA^\top)^{-1} A$ ▷ $G \in \mathbb{R}^{n \times n}$
  $x = u_1 + Gu_2$ ▷ $x \in \mathbb{R}^n$
  $s = u_3 + Mu_4$ ▷ $x \in \mathbb{R}^n$
end invariant

procedure initialize($f, \epsilon_{\mathrm{mp}}, \epsilon_{\mathrm{far}}, a, \widetilde{a}, b, L, A, w_0, h_0, R$)
  $\cdots$ ▷ All previous members are initialized as of Algorithm 5.
  $G \leftarrow W_0 A^\top (AVA^\top)^{-1} A$ ▷ $G \in \mathbb{R}^{n \times n}$
  $u_1, u_2, u_3, u_4 \leftarrow 0$ ▷ $u_1, u_2, u_3, u_4 \in \mathbb{R}^n$
end procedure

end data structure

To make the output feasible, we present in this section the modified algorithms. In Section I.1

Algorithm 22 Data structure : ScalarC.

procedure ScalarC(h^{appr}) ▷ Output a number $c \in \mathbb{R}$
  if self.name $= \mathrm{mp}_t$ **then**
    $c \leftarrow \frac{t^{\mathrm{new}}}{t} - 1$
  end if
  if self.name $= \mathrm{mp}_\Phi$ **then**
    $\widetilde{\mu} \leftarrow h^{\mathrm{appr}} \cdot t$
    $c \leftarrow -\frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{1}{\sqrt{t}\|\nabla \Phi_\lambda(\widetilde{\mu}/t-1)\|_2}$
  end if
  return c
end procedure

Algorithm 23 Data structure : feasible version of UpdateQuery (Algorithm 8)

data structure ▷ Theorem C.9

procedure UpdateQuery(w^{new}, h^{new}) ▷ Theorem I.2
  $h^{\mathrm{appr}}, p, \widetilde{p} \leftarrow$ UpdateG(h^{new}) ▷ Algorithm 10, p and $\widetilde{p}$ are only used for analysis.
  $w^{\mathrm{appr}}, k, \widetilde{k} \leftarrow$ UpdateV($w^{\mathrm{new}}, h^{\mathrm{appr}}$) ▷ Algorithm 9, k and $\widetilde{k}$ are only used for analysis.
  $r \leftarrow$ Query($w^{\mathrm{appr}}, h^{\mathrm{appr}}$) ▷ Algorithm 24, Lemma D.7, E.3, I.10
  ▷ Compute $r = R[l]^\top R[l] \sqrt{W^{\mathrm{appr}}} A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$
  $c \leftarrow$ ScalarC(h^{appr})
  if self.name $= \mathrm{mp}_t$ **then**
    $\widetilde{\mu} \leftarrow h^{\mathrm{appr}}$
  end if
  if self.name $= \mathrm{mp}_\Phi$ **then**
    $\widetilde{\mu} \leftarrow h^{\mathrm{appr}} \cdot t$
  end if
  $\widetilde{w} \leftarrow w^{\mathrm{appr}}$
  $\widetilde{x} \leftarrow \sqrt{\widetilde{\mu} \cdot \widetilde{w}}$, $\widetilde{s} \leftarrow \sqrt{\widetilde{\mu}/\widetilde{w}}$ ▷ $\widetilde{\mu} = \widetilde{x}\widetilde{s}$ and $\widetilde{w} = \widetilde{x}/\widetilde{s}$
  $q_x \leftarrow \sqrt{\frac{\widetilde{x}}{\widetilde{s}}} \cdot c \cdot f(h^{\mathrm{appr}})$
  $p_x \leftarrow \sqrt{\frac{\widetilde{x}}{\widetilde{s}}} \cdot c \cdot r$
  $p_s \leftarrow \sqrt{\frac{\widetilde{s}}{\widetilde{x}}} \cdot c \cdot r$
  return $q_x, p_x, p_s, w^{\mathrm{appr}}$
end procedure

end data structure

we show that the error guarantees of central path method still has the same bound with the modifications, this section should be seen as a complement of Section B. In Section I.2 we prove the correctness of this feasible algorithm when implicitly maintaining x and s, this section should be seen as a complement of Section D. In Section I.4 we bound that the running time of maintaining x and s, this section should be seen as a complement of Section E.

I.1 Analysis

Consider the j-th iteration. Assume at the beginning of the j-th iteration, we have $x^{(j)}$, $s^{(j)}$, $\overline{x}^{(j)}$, $\overline{s}^{(j)}$. Define $w := w^{(j)} = x^{(j)} s^{(j)}$, $\mu := \mu^{(j)} = \frac{x^{(j)}}{s^{(j)}}$, $\overline{w} := \overline{w}^{(j)} = \overline{x}^{(j)} \overline{s}^{(j)}$, $\overline{\mu} := \overline{\mu}^{(j)} = \frac{\overline{x}^{(j)}}{\overline{s}^{(j)}}$, $w^{\mathrm{new}} := w^{(j+1)} = x^{(j+1)} s^{(j+1)}$, $\mu^{\mathrm{new}} := \mu^{(j+1)} = \frac{x^{(j+1)}}{s^{(j+1)}}$. We will prove *inductively* that the guarantees of

Algorithm 24 Data structure : feasible version of QUERY (Algorithm 12)

data structure ▷ Theorem C.9

procedure QUERY($w^{\mathrm{appr}}, h^{\mathrm{appr}}$) ▷ Lemma D.7, E.3, I.10

$\partial\Delta, \partial\Gamma, \partial\xi, \partial S, \Delta^{\mathrm{new}}, \_, \_, S^{\mathrm{new}}, S' \leftarrow$ COMPUTELOCALVARIABLES($w^{\mathrm{appr}}, \widetilde{g}^{\mathrm{new}}$)

▷ Algorithm 11

$r_1 \leftarrow \beta_1[l]$ ▷ $r_1 \in \mathbb{R}^{n^b}$

$r_2 \leftarrow Q[l]\xi + R[l]\gamma_2 + R[l]\partial\Gamma M(\xi + \partial\xi) + \big(Q[l] + R[l]\Gamma M\big)\partial\xi$ ▷ $r_2 \in \mathbb{R}^{n^b}$

$r_3 \leftarrow R[l](\Gamma + \partial\Gamma)\beta_2$ ▷ $r_3 \in \mathbb{R}^{n^b}$

$\partial\gamma \leftarrow B \cdot (\mathcal{L}_r[(\beta_2)_{\partial S \setminus S}] - \mathcal{L}_r[(\beta_2)_{S'}]) + B \cdot (\mathcal{L}_r[(M_{\partial S \setminus S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) \cdot (\xi + \partial\xi) + E \cdot \partial\xi$

▷ local variable $\partial\gamma \in \mathbb{R}^{6n^a}$

$(U', C, U) \leftarrow$ DECOMPOSE$\Big(\mathcal{L}_*[(\Delta^{\mathrm{new}}_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}}, S^{\mathrm{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}]\Big)$

▷ DECOMPOSE is defined in Lemma C.4. $U', U \in \mathbb{R}^{6n^a \times 3|\partial S|}$, $C \in \mathbb{R}^{3|\partial S| \times 3|\partial S|}$

$\partial E \leftarrow E_{\partial S} - B_{(\partial S \cap S)} \cdot M_{(\partial S \cap S), \partial S}$

$(\partial E)_{S'} \leftarrow -(\partial E)_{S'}, \quad (\partial E)_{(S \cap \partial S) \setminus S'} \leftarrow 0$ ▷ local variable $\partial E \in \mathbb{R}^{6n^a \times |\partial S|}$

$U^{\mathrm{tmp}} \leftarrow [B_{\partial S}, B_{\partial S}, \partial E]$

▷ local variable $U^{\mathrm{tmp}} \in \mathbb{R}^{6n^a \times 3|\partial S|}$, $U^{\mathrm{tmp}} = BU'$ (Corollary C.7)

$\gamma^{\mathrm{tmp}} \leftarrow U^{\mathrm{tmp}}(C^{-1} + U^\top U^{\mathrm{tmp}})^{-1} U^\top \cdot (\gamma_1 + \partial\gamma)$ ▷ local variable, $\gamma^{\mathrm{tmp}} \in \mathbb{R}^{6n^a}$

$r_4 \leftarrow \Big(\mathcal{L}_c[(Q[l])_{S^{\mathrm{new}}}] + F[l] + R[l]\Gamma(\mathcal{L}_c[M_{\partial S \setminus S}] - \mathcal{L}_c[M_{S'}]) + R[l]\partial\Gamma\mathcal{L}_c[M_{S^{\mathrm{new}}}]\Big)(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)$

$r \leftarrow R[l]^\top (r_1 + r_2 + r_3 + r_4)$ ▷ $r \in \mathbb{R}^n$

$l \leftarrow l + 1$

$c \leftarrow$ SCALARC(h^{appr})

$u_1 \leftarrow u_1 + c \cdot (W^{\mathrm{appr}} - \widetilde{V})\Big(\beta_2 + M \cdot \big(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \sqrt{V} f(g) + \mathbf{1}_{S^{\mathrm{new}}}(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)\big)\Big)$

▷ $\mathbf{1}_{S^{\mathrm{new}}} \in \mathbb{R}^{n \times 6n^a}$ only has ones in positions (i,i) for $i \in S^{\mathrm{new}}$

$u_2 \leftarrow u_2 + c \cdot \Big(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) + \mathbf{1}_{S^{\mathrm{new}}}(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)\Big)$

$u_4 \leftarrow u_4 + c \cdot \Big(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) + \mathbf{1}_{S^{\mathrm{new}}}(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)\Big)$

return r

end procedure

end data structure

Section B are still satisfied for the modified algorithm.

Robustness of central path. We first prove the following two statements about the robustness of central path method:

1. The w^{appr} and h^{appr} used in the data structures satisfy $w^{\mathrm{appr}} \approx_{2\epsilon_{\mathrm{mp}}} w$, and $h^{\mathrm{appr}} \approx_{2\epsilon_{\mathrm{mp}}} \mu$. (This corresponds to Part 1 and 2 of Assumption B.5. We only lose a constant factor here.)
2. $\mu^{\mathrm{new}} \approx_{0.2} t$. (This corresponds to Part 3 of Assumption B.5. We only loose a constant factor here.)

In Part 3 of Theorem I.2, we prove that in any iteration, $x, \overline{x}$, $s, \overline{s}$ are entry-wise close with high probability, i.e. $\overline{x} \approx_{\epsilon_{\mathrm{tiny}}} x$ and $\overline{s} \approx_{\epsilon_{\mathrm{tiny}}} s$ holds with probability $1 - 1/\operatorname{poly}(n)$. We will use this to prove that the above two statements are still satisfied in our modified algorithm.

1. The data structure directly ensures $w^{\mathrm{appr}} \approx_{\epsilon_{\mathrm{mp}}} \overline{w}$, and $h^{\mathrm{appr}} \approx_{\epsilon_{\mathrm{mp}}} \overline{\mu}$. And since $\overline{x}^{(j)} \approx_{\epsilon_{\mathrm{tiny}}} x^{(j)}$ and $\overline{s}^{(j)} \approx_{\epsilon_{\mathrm{tiny}}} s^{(j)}$ (Part 3 of Theorem I.2) we directly get the desired result that $w^{\mathrm{appr}} \approx_{2\epsilon_{\mathrm{mp}}} w$ and $h^{\mathrm{appr}} \approx_{2\epsilon_{\mathrm{mp}}} \mu$.

Algorithm 25 Data structure : feasible version of MatrixUpdate (Algorithm 13)

data structure ▷ Theorem C.9

procedure MatrixUpdate($w^{\mathrm{appr}}, h^{\mathrm{appr}}$) ▷ Lemma D.17, E.12, I.7

  $\_, \_, \_, \_, \Delta^{\mathrm{new}}, \Gamma^{\mathrm{new}}, \_, S^{\mathrm{new}}, \_ \leftarrow$ ComputeLocalVariables($w^{\mathrm{appr}}, \_$) ▷ Algorithm 11

  $M^{\mathrm{tmp}} \leftarrow M - M_{S^{\mathrm{new}}} \cdot ((\Delta^{\mathrm{new}}_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}}, S^{\mathrm{new}}})^{-1} \cdot (M_{S^{\mathrm{new}}})^\top$

  $Q^{\mathrm{tmp}} \leftarrow Q + R(\Gamma^{\mathrm{new}} M^{\mathrm{tmp}}) + R\sqrt{V}(M^{\mathrm{tmp}} - M)$

  $\beta_1^{\mathrm{tmp}} \leftarrow Q^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}} f(g)$

  $\beta_2^{\mathrm{tmp}} \leftarrow M^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}} f(g)$

  $\xi^{\mathrm{tmp}} \leftarrow \sqrt{W^{\mathrm{appr}}}(f(\widetilde{g}) - f(g))$

  ▷ We start to refresh variables in the memory of data structure

  $c \leftarrow$ ScalarC(h^{appr})

  $G \leftarrow W^{\mathrm{appr}} M^{\mathrm{tmp}}$

  $u_1 \leftarrow u_1 + G u_2 + c \cdot W^{\mathrm{appr}} M^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$

  $u_3 \leftarrow u_3 + M u_4 + c \cdot M^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$

  $u_2 \leftarrow 0$, $u_4 \leftarrow 0$

  $Q \leftarrow Q^{\mathrm{tmp}}$, $M \leftarrow M^{\mathrm{tmp}}$

  $\beta_1 \leftarrow \beta_1^{\mathrm{tmp}}$, $\beta_2 \leftarrow \beta_2^{\mathrm{tmp}}$, $\xi \leftarrow \xi^{\mathrm{tmp}}$

  $v \leftarrow \widetilde{v} \leftarrow w^{\mathrm{appr}}$

  $B \leftarrow I$, $F \leftarrow 0$, $E \leftarrow 0$

  $S \leftarrow \emptyset$, $\Delta \leftarrow \Gamma \leftarrow 0$, $\gamma_1 \leftarrow \gamma_2 \leftarrow 0$

end procedure

end data structure

2. Note that previously $\mu \approx_{0.1} t$ was proved in Lemma B.27 by bounding the potential function. And the proof of Lemma B.27 uses the bounds given by Lemma B.21. We define the potential function to be $\Phi(\frac{xs}{t} - 1)$ here instead of $\Phi(\frac{\overline{x}\overline{s}}{t} - 1)$. Note that conditioned on $\overline{x}^{(j)}$ and $\overline{s}^{(j)}$, $x^{(j+1)}$ and $s^{(j+1)}$ are deterministic. Thus Part 2 and Part 4 of Lemma B.21 are trivial. We still have an analog of Part 1 and Part 3 of Lemma B.21: $\|\overline{\mu}^{-1}(\mu^{\mathrm{new}} - \overline{\mu} - \overline{\delta}_t - \widetilde{\delta}_\Phi)\|_2 \leq O(\epsilon_{\mathrm{mp}}$ and $\|\overline{\mu}^{-1}(\mu^{\mathrm{new}} - \overline{\mu})\|_\infty \leq O(\epsilon)$ using the fact that $\overline{x}^{(j)}$ and $\overline{s}^{(j)}$ are close to $x^{(j)}$ and $s^{(j)}$ (Part 3 of Theorem I.2). Thus we still have an analog of Lemma B.27 that

$$\Phi_\lambda(\mu^{\mathrm{new}}/t^{\mathrm{new}} - 1) \leq \Phi_\lambda(\overline{\mu}/t - 1) - \frac{\lambda\epsilon}{15\sqrt{n}} \cdot (\Phi_\lambda(\overline{\mu}/t - 1) - 10n).$$

   Using Part 3 of Theorem I.2 again and the fact that the derivative of $\cosh(x)$ function is constant when $x < 2$, we have

$$\Phi_\lambda(\mu^{\mathrm{new}}/t^{\mathrm{new}} - 1) \leq \Phi_\lambda(\mu/t - 1) - \Omega(\frac{\lambda\epsilon}{\sqrt{n}}) \cdot (\Phi_\lambda(\mu/t - 1) - O(n^2)).$$

   Thus we can still inductively prove a polynomial upper bound on the potential function $\Phi_\lambda(\frac{xs}{t} - 1)$. Note that we only loose a constant factor in the approximation ratio of μ with t when the last term is $O(n^2)$ instead of $O(n)$.

w and h move slowly. The last part of the inductive analysis is to show that w and μ are both moving slowly as that of Section B.6 and B.7. Now we only have the guarantee for w and μ, instead of $\overline{w}$ and $\overline{\mu}$, so we use $\psi(w_i/v_i - 1)$ and $\psi(w_i/\widetilde{v}_i - 1)$ in the analysis and use $\psi(\overline{w}_i/v_i - 1)$ and $\psi(\overline{w}_i/\widetilde{v}_i - 1)$ for the algorithm (because the algorithm doesn't know w and μ). The amortized analysis of Section F need to be modified accordingly. We have the following two statements:

Algorithm 26 Data structure : feasible version of PartialMatrixUpdate (Algorithm 14).

data structure ▷ Theorem C.9

procedure PartialMatrixUpdate($w^{\mathrm{appr}}, h^{\mathrm{appr}}$) ▷ Lemma D.21, E.18, I.8

  $\_, \partial\Gamma, \_, \partial S, \Delta^{\mathrm{new}}, \Gamma^{\mathrm{new}}, \_, S^{\mathrm{new}}, \_ \leftarrow$ ComputeLocalVariables($w^{\mathrm{appr}}, \_$) ▷ Algorithm 11

  $(U', C, U) \leftarrow \textsc{Decompose}\Big(\mathcal{L}_*[(\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}}] - \mathcal{L}_*[\Delta^{-1}_{S,S} + M_{S,S}]\Big)$

  ▷ Decompose is defined in Lemma C.4

  $B^{\mathrm{tmp}} \leftarrow B - BU'(C^{-1} + U^\top BU')^{-1}U^\top B$

  $F^{\mathrm{tmp}} \leftarrow F + R\Gamma \cdot (\mathcal{L}_c[M_{\partial S \setminus S}] - \mathcal{L}_c[M_{S'}]) + R\partial\Gamma \cdot \mathcal{L}_c[M_{S^{\mathrm{new}}}]$

  $E^{\mathrm{tmp}} \leftarrow E + B^{\mathrm{tmp}}(\mathcal{L}_r[(M_{\partial S \setminus S})^\top] - \mathcal{L}_r[(M_{S'})^\top]) - BU'(C^{-1} + U^\top BU')^{-1}U^\top E$

  $\xi^{\mathrm{tmp}} \leftarrow \sqrt{W^{\mathrm{appr}}} f(\widetilde{g}) - \sqrt{V} f(g)$

  $\gamma_1^{\mathrm{tmp}} \leftarrow B^{\mathrm{tmp}} \cdot \mathcal{L}_r[\beta_{2,S^{\mathrm{new}}}] + B^{\mathrm{tmp}} \cdot \mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top]\xi^{\mathrm{tmp}}$

  $\gamma_2^{\mathrm{tmp}} \leftarrow \gamma_2 + (\Gamma + \partial\Gamma)M(\sqrt{W^{\mathrm{appr}}} - \sqrt{\widetilde{V}})f(\widetilde{g}) + \partial\Gamma M(\sqrt{\widetilde{V}} f(\widetilde{g}) - \sqrt{V} f(g))$

  $c \leftarrow \textsc{ScalarC}(h^{\mathrm{appr}})$

  $G \leftarrow G + (W^{\mathrm{appr}} - \widetilde{V}) \cdot M$

  $u_1 \leftarrow u_1 + (W^{\mathrm{appr}} - \widetilde{V}) \cdot M u_2$

  $u_2 \leftarrow u_2 + c \cdot \Big(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \mathbf{1}_{S^{\mathrm{new}}} B^{\mathrm{tmp}} \mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top] \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})\Big)$

  $u_4 \leftarrow u_4 + c \cdot \Big(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \mathbf{1}_{S^{\mathrm{new}}} B^{\mathrm{tmp}} \mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top] \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})\Big)$

  ▷ We start to refresh variables in the memory of data structure

  $B \leftarrow B^{\mathrm{tmp}}$, $F \leftarrow F^{\mathrm{tmp}}$, $E \leftarrow E^{\mathrm{tmp}}$

  $\xi \leftarrow \xi^{\mathrm{tmp}}$ $\gamma_1 \leftarrow \gamma_1^{\mathrm{tmp}}$, $\gamma_2 \leftarrow \gamma_2^{\mathrm{tmp}}$

  $\widetilde{v} \leftarrow w^{\mathrm{appr}}, S \leftarrow S^{\mathrm{new}}$, $\Delta \leftarrow \Delta^{\mathrm{new}}$, $\Gamma \leftarrow \Gamma^{\mathrm{new}}$

end procedure

end data structure

1. The three bounds of Lemma B.28 and Lemma B.29 hold for $w^{\mathrm{new}}/w - 1$ and $\mu^{\mathrm{new}}/\mu - 1$. (Originally used in subsection F.4.3, F.3.3.)

2. Lemma F.24 and Lemma F.33 still hold for new potential function with use w instead of $\overline{w}$.

Proof Sketch.

1. Note that Lemma B.28 and Lemma B.29 only use the relative error bounds stated in Lemma B.16. We can prove that all $\overline{x}^{-1}$, $\overline{s}^{-1}$ and $\overline{\mu}^{-1}$ terms of Lemma B.16 can be replaced by x, s, and μ since we proved that in iteration j, $x^{(j)} \approx_{\epsilon_{\mathrm{tiny}}} \overline{x}^{(j)}$ and $s^{(j)} \approx_{\epsilon_{\mathrm{tiny}}} \overline{s}^{(j)}$. Following the same proof of Lemma B.28 and Lemma B.29, and using the error bound of the x and s version of Lemma B.16, we can prove the desired statement.

2. Since $x \approx_{\epsilon_{\mathrm{tiny}}} \overline{x}$ in any iteration (Part 3 of Theorem I.2), we have $w^{(j+1)} \approx_{\epsilon_{\mathrm{tiny}}} \overline{w}^{(j+1)}$, hence $\psi(\overline{w}_i^{(j+1)}/v_i^{(j+1)} - 1)$ is within the range of $[\psi(w_i^{(j+1)}/v_i^{(j+1)} - 1) - \epsilon_{\mathrm{tiny}}, \psi(w_i^{(j+1)}/v_i^{(j+1)} - 1) + \epsilon_{\mathrm{tiny}}]$. So it is fine to use $\psi(w_i^{(j+1)}/v_i^{(j+1)} - 1)$ in the analysis while using $\psi(\overline{w}_i^{(j+1)}/v_i^{(j+1)} - 1)$ in the algorithm. The only problem we need to resolve is that when v and $\widetilde{v}$ are updated to $w^{\mathrm{appr}} = \overline{w}$, the potential function is not cleared to be 0, but instead remain a small number $\epsilon_{\mathrm{tiny}} \ll \epsilon_{\mathrm{mp}}$. But this problem is already solved in Lemma F.33.

□

I.2 Correctness of feasible algorithm

We first present the following main theorem, and prove it using lemmas proved subsequently.

Theorem I.2. *In the j-th iteration of the while-loop from Line 19 to Line 44 of* Main *(Algorithm 18), let $\overline{x}^{(j)}$ and $\overline{s}^{(j)}$ be the values of global variables $\overline{x}$ and $\overline{s}$ at the beginning of j-th iteration, and let $\overline{x}^{(j+1)}$, $\overline{s}^{(j+1)}$, $x^{(j+1)}$, $s^{(j+1)}$ be the values of global variables $\overline{x}$, $\overline{s}$, x, s at the end of j-th iteration. Let $\widetilde{x}$, $\widetilde{s}$, $\widetilde{P}$, $\widetilde{\delta}_t$, $\widetilde{\delta}_\Phi$, $\widetilde{\delta}_x$, $\widetilde{\delta}_s$ be defined as in Definition B.4. Let $R \in \mathbb{R}^{b\times n}$ be the subsampled randomized Hadamard matrix we used in our algorithm, defined in Definition B.10.*

Then our algorithm guarantees that

1. The output of OneStepCentralPath *(Algorithm 19) satisfies*

$$\widehat{\delta}_x = \sqrt{\frac{\widetilde{X}}{\widetilde{S}}}(I - (R^\top R)\widetilde{P})\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}(\widetilde{\delta}_t + \widetilde{\delta}_\Phi), \quad \widehat{\delta}_s = \sqrt{\frac{\widetilde{S}}{\widetilde{X}}}(R^\top R)\widetilde{P}\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}(\widetilde{\delta}_t + \widetilde{\delta}_\Phi),$$

$\widehat{\delta}_x$ and $\widehat{\delta}_s$ match the definition in Definition B.6.

2. In each iteration, when the algorithm reaches MakeFeasible *(Algorithm 20) on Line 28 of* Main *(Algorithm 18), the following holds:*

$$x - \big((\mathrm{mp}_t.u_1) + (\mathrm{mp}_t.G)\cdot(\mathrm{mp}_t.u_2)\big) - \big((\mathrm{mp}_\Phi.u_1) + (\mathrm{mp}_\Phi.G)\cdot(\mathrm{mp}_\Phi.u_2)\big) = x^{(0)} + \sum_{i=1}^{j}\widetilde{\delta}_x^{(i)}$$

$$s + \big((\mathrm{mp}_t.u_3) + (\mathrm{mp}_t.M)\cdot(\mathrm{mp}_t.u_4)\big) + \big((\mathrm{mp}_\Phi.u_3) + (\mathrm{mp}_\Phi.M)\cdot(\mathrm{mp}_\Phi.u_4)\big) = s^{(0)} + \sum_{i=1}^{j}\widetilde{\delta}_s^{(i)}.$$

3. $\overline{x} \approx_{\epsilon_{\mathrm{tiny}}} x$, $\overline{s} \approx_{\epsilon_{\mathrm{tiny}}} s$ with high probability.

Proof. For simplicity, we only prove these three statements for x. The case for s follows from similar reasons. Since we have two data structures sharing the same code, we denote $q_{x,t}$, $q_{x,\Phi}$, $p_{x,t}$, $p_{x,\Phi}$ as the q_x and p_x defined on Line 17 and 18 of UpdateQuery (Algorithm 23) in data structure mp_t and mp_Φ respectively. Also, We denote c_t as the output of ScalarC (Algorithm 22) of data structure mp_t, and c_Φ corresponds to mp_Φ. From the description of ScalarC, we have

$$c_t := (\frac{t^{\mathrm{new}}}{t} - 1) \qquad c_\Phi := -\frac{\epsilon}{2}\cdot t^{\mathrm{new}}\cdot\frac{1}{\sqrt{t}\|\nabla\Phi_\lambda(\widetilde{\mu}/t - 1)\|_2}. \tag{71}$$

And recall f_t, f_Φ defined in Line 13, Algorithm 18) are

$$f_t(x) := \sqrt{x} \qquad f_\Phi(x) := \nabla\Phi_\lambda(x)/\sqrt{x}. \tag{72}$$

In UpdateQuery (Algorithm 23), the two data structure approximate w^{appr} and h^{appr} in the same way, so their $\widetilde{w}$ (Line 10) and $\widetilde{\mu}$ (Line 10 and 13) are the same. So they also have the same $\widetilde{x}$ and $\widetilde{s}$ (Line 16). Note that $\widetilde{w}$, $\widetilde{\mu}$, $\widetilde{x}$, $\widetilde{s}$ calculated in the data structure all matches Definition B.4.

We first show the following that will be used in both Part 1 and 2:

$$\begin{aligned} c_t\cdot f_t(\widetilde{\mu}) + c_\Phi\cdot f_\Phi(\widetilde{\mu}/t) &= (\frac{t^{\mathrm{new}}}{t} - 1)\cdot\sqrt{\widetilde{\mu}} - \frac{\epsilon}{2}\cdot t^{\mathrm{new}}\cdot\frac{1}{\sqrt{t}\|\nabla\Phi_\lambda(\widetilde{\mu}/t - 1)\|_2}\cdot\nabla\Phi_\lambda(\widetilde{\mu}/t)/\sqrt{\widetilde{\mu}/t} \\ &= \frac{1}{\sqrt{\widetilde{\mu}}}\left((\frac{t^{\mathrm{new}}}{t} - 1)\cdot\widetilde{\mu} - \frac{\epsilon}{2}\cdot t^{\mathrm{new}}\cdot\frac{\nabla\Phi_\lambda(\widetilde{\mu}/t)}{\|\nabla\Phi_\lambda(\widetilde{\mu}/t - 1)\|_2}\right) \\ &= \frac{1}{\sqrt{\widetilde{\mu}}}(\widetilde{\delta}_t + \widetilde{\delta}_\Phi) = \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}(\widetilde{\delta}_t + \widetilde{\delta}_\Phi), \end{aligned} \tag{73}$$

where the first step is by the definition of c and f (Eq.(71),(72)), the third step is by the definition of $\widetilde{\delta}_t$ and $\widetilde{\delta}_\Phi$ (Definition B.4), the last step is by $\widetilde{\mu} = \widetilde{x}\widetilde{s}$.

Part 1. First we calculate $q_{x,t} + q_{x,\Phi}$:

$$q_{x,t} + q_{x,\Phi} = \sqrt{\frac{\widetilde{X}}{\widetilde{S}}} (c_t \cdot f_t(\widetilde{\mu}) + c_\Phi \cdot f_\Phi(\widetilde{\mu}/t)) = \sqrt{\frac{\widetilde{X}}{\widetilde{S}}} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} (\widetilde{\delta}_t + \widetilde{\delta}_\Phi) \tag{74}$$

where the first step is by assignment of q_x (Line 17 of UpdateQuery, Algorithm 23), the second step is by Eq.(73).

Next, we calculate $p_{x,t} + p_{x,\Phi}$. Note that the output r of Query is calculated in the same way as before, so by Lemma D.7 we have

$$r = R[l]^\top R[l] \sqrt{W^{\mathrm{appr}}} A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) = R[l]^\top R[l] \widetilde{P} f(h^{\mathrm{appr}}). \tag{75}$$

Therefore,

$$\begin{aligned} p_{x,t} + p_{x,\Phi} &= \sqrt{\frac{\widetilde{X}}{\widetilde{S}}} (c_t \cdot r_t + c_\Phi \cdot r_\Phi) = \sqrt{\frac{\widetilde{X}}{\widetilde{S}}} \Big(c_t \cdot R[l]^\top R[l] \widetilde{P} f_t(\widetilde{\mu}) + c_\Phi \cdot R[l]^\top R[l] \widetilde{P} f_\Phi(\widetilde{\mu}/t) \Big) \\ &= \sqrt{\frac{\widetilde{X}}{\widetilde{S}}} R[l]^\top R[l] \widetilde{P} (c_t \cdot f_t(\widetilde{\mu}) + c_\Phi \cdot f_\Phi(\widetilde{\mu}/t)) = \sqrt{\frac{\widetilde{X}}{\widetilde{S}}} R[l]^\top R[l] \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \cdot (\widetilde{\delta}_t + \widetilde{\delta}_\Phi), \end{aligned}$$

where the first step is by assignment of p_x (Line 18 of UpdateQuery, Algorithm 23), the second step is by Eq.(75), the last step is by Eq.(73).

Then as we calculate $\widehat{\delta}_x$ in line 8 of OneStepCentralPath (Algorithm 19),

$$\begin{aligned} \widehat{\delta}_x &= q_{t,x} + q_{\Phi,x} - (p_{t,x} + p_{\Phi,x}) \\ &= \sqrt{\frac{\widetilde{X}}{\widetilde{S}}} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} (\widetilde{\delta}_t + \widetilde{\delta}_\Phi) - \sqrt{\frac{\widetilde{X}}{\widetilde{S}}} R[l]^\top R[l] \widetilde{P} \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \cdot (\widetilde{\delta}_t + \widetilde{\delta}_\Phi) \\ &= \sqrt{\frac{\widetilde{X}}{\widetilde{S}}} \Big(I - R[l]^\top R[l] \widetilde{P} \Big) \frac{1}{\sqrt{\widetilde{X}\widetilde{S}}} \cdot (\widetilde{\delta}_t + \widetilde{\delta}_\Phi). \end{aligned}$$

Part 2. We prove Part 2 by induction. In the basic case when $j = 0$, we initialize $x \leftarrow x^{(0)}$, so it is true.

When $j > 0$, let $j_1 < j$ be the last iteration that we call Initialize on Line 32 and 33 of Main (Algorithm 18). In the j_1-th iteration, the Main algorithm executes Line 30: $x \leftarrow x - (\mathrm{mp}_t.u_1 + \mathrm{mp}_t.G \cdot \mathrm{mp}_t.u_2) - (\mathrm{mp}_\Phi.u_1 + \mathrm{mp}_\Phi.G \cdot \mathrm{mp}_\Phi.u_2)$, and then re-initialize $\mathrm{mp}_t.u_1 \leftarrow \mathrm{mp}_t.u_2 \leftarrow \mathrm{mp}_\Phi.u_1 \leftarrow \mathrm{mp}_\Phi.u_2 \leftarrow 0$. Therefore by induction on j, we have $x^{(j_1)} = x^{(0)} + \sum_{i=1}^{j_1} \widetilde{\delta}_x^{(i)}$.

Now apply Part 3 of Lemma I.3, we have that

$$\begin{aligned} & \big((\mathrm{mp}_t.u_1) + (\mathrm{mp}_t.G) \cdot (\mathrm{mp}_t.u_2) \big) + \big((\mathrm{mp}_\Phi.u_1) + (\mathrm{mp}_\Phi.G) \cdot (\mathrm{mp}_\Phi.u_2) \big) \\ = & \sum_{i=j_1+1}^{j} c_t^{(i)} \cdot \sqrt{W^{\mathrm{appr},(i)}} \widetilde{P}^{(i)} f_t(\widetilde{\mu}^{(i)}) + \sum_{i=j_1+1}^{j} c_\Phi^{(i)} \cdot \sqrt{W^{\mathrm{appr},(i)}} \widetilde{P}^{(i)} f_\Phi(\widetilde{\mu}^{(i)}/t^{(i)}) \\ = & \sum_{i=j_1+1}^{j} \sqrt{\frac{\widetilde{X}^{(i)}}{\widetilde{S}^{(i)}}} \widetilde{P}^{(i)} \frac{1}{\sqrt{\widetilde{X}^{(i)}\widetilde{S}^{(i)}}} (\widetilde{\delta}_t^{(i)} + \widetilde{\delta}_\Phi^{(i)}). \end{aligned}$$

where the first step is by Part 3 of Lemma I.3, the second step is by Eq.(73).

Also notice that in every iteration, we add $q_{t,x} + q_{\Phi,x}$ to x in OneStepCentralPath (Line 10 of Algorithm 19), so

$$x^{(j)} - x^{(j_1)} = \sum_{i=j_1+1}^{j} (q_{t,x}^{(i)} + q_{\Phi,x}^{(i)}) = \sum_{i=j_1+1}^{j} \sqrt{\frac{\widetilde{X}^{(i)}}{\widetilde{S}^{(i)}}} \frac{1}{\sqrt{\widetilde{X}^{(i)}\widetilde{S}^{(i)}}} (\widetilde{\delta}_t^{(i)} + \widetilde{\delta}_\Phi^{(i)}).$$

where the last step is by Eq.(74).

Therefore,

$$\begin{aligned}
& x^{(j)} - \big((\mathrm{mp}_t.u_1) + (\mathrm{mp}_t.G) \cdot (\mathrm{mp}_t.u_2)\big) - \big((\mathrm{mp}_\Phi.u_1) + (\mathrm{mp}_\Phi.G) \cdot (\mathrm{mp}_\Phi.u_2)\big) \\
= & x^{(j_1)} + \sum_{i=j_1+1}^{j} \left(\sqrt{\frac{\widetilde{X}^{(i)}}{\widetilde{S}^{(i)}}} \frac{1}{\sqrt{\widetilde{X}^{(i)}\widetilde{S}^{(i)}}} (\widetilde{\delta}_t^{(i)} + \widetilde{\delta}_\Phi^{(i)}) - \sqrt{\frac{\widetilde{X}^{(i)}}{\widetilde{S}^{(i)}}} \cdot \widetilde{P}^{(i)} \cdot \frac{1}{\sqrt{\widetilde{X}^{(i)}\widetilde{S}^{(i)}}} (\widetilde{\delta}_t^{(i)} + \widetilde{\delta}_\Phi^{(i)}) \right) \\
= & x^{(j_1)} + \sum_{i=j_1+1}^{j} \widetilde{\delta}_x^{(i)} \\
= & x^{(0)} + \sum_{i=1}^{j} \widetilde{\delta}_x^{(i)}.
\end{aligned}$$

Part 3. Consider a fixed coordinate i. We use j_i to denote the last iteration that the algorithm includes i into $\widehat{S}$ when enter the MakeFeasible procedure on Line 28 of Main (Algorithm 18) or when re-initialize. We prove the following properties in order to apply Lemma I.5.

1. $\overline{x}_i^{(j_i)} = x_i^{(j_i)}$. If the algorithm enter the Initialize procedure, we have $\overline{x}_i^{(j_i)} = x_i^{(j_i)}$. Otherwise we enter the MakeFeasible procedure, and according to the updating rule $\overline{x}_{\widehat{S}} \leftarrow x_{\widehat{S}}$ (Line 4 of MakeFeasible, Algorithm 20), we also have $\overline{x}_i^{(j_i)} = x_i^{(j_i)}$.

2. $t^{(j)} > t^{(j_i)}/2$ and $j - j_i \leq \sqrt{n}$, since the algorithm re-Initialize whenever it passes $\sqrt{n}$ iterations or t changes too much (Line 29 in Main, Algorithm 18).

3. For all $l \in \{j_i + 1, \cdots, j\}$, $w^{\mathrm{appr},(l)} \in [w^{\mathrm{old}}/2, 2w^{\mathrm{old}}]$, since the algorithm doesn't include coordinate i in $\widetilde{S}$ during iteration l.

Then by Lemma I.5, $x_i^{(j)} \approx_{\epsilon_x} \overline{x}_i^{(j)}$ holds with probability at least $1 - \delta$, where $\epsilon_x = \frac{200n^{1/4}}{\sqrt{b}} \cdot \log(n/\delta)\epsilon \leq \frac{\epsilon_{\mathrm{mp}} \cdot \epsilon}{3200 \log^3 n}$ by our assignment of $b = 10^{12}\sqrt{n}\log^8 n/\epsilon_{\mathrm{mp}}^2$ (Line 7 of Main, Algorithm 18) and $\delta = 1/\operatorname{poly}(n)$. This satisfies the requirement of Def. I.1.

By choosing δ to be $1/n^{10}$ and union bound on all coordinates i, $x^{(j)} \approx_{\epsilon_{\mathrm{tiny}}} \overline{x}^{(j)}$ holds w.h.p. □

The following lemma proves the invariants that are true throughout the algorithm. It is used to prove Theorem I.2. This lemma should be seen as a complement of Section D, and we directly use results proved in that section.

Lemma I.3 (Invariants)**.** *In the end of the j-th iteration, the following invariants hold:*

1. $G = \widetilde{V} A^\top (AVA^\top)^{-1} A$.

2. $u_3 + Mu_4 = \sum_{i=j_1+1}^{j} c^{(i)} \cdot A^\top (AW^{\mathrm{appr},(i)} A^\top)^{-1} A \sqrt{W^{\mathrm{appr},(i)}} f(h^{\mathrm{appr},(i)})$,

3. $u_1 + Gu_2 = \sum_{i=j_1+1}^{j} c^{(i)} \cdot W^{\mathrm{appr},(i)} A^\top (A W^{\mathrm{appr},(i)} A^\top)^{-1} A \sqrt{W^{\mathrm{appr},(i)}} f(h^{\mathrm{appr},(i)})$,

where j_1 is the last iteration we call INITIALIZE*, and $c^{(j)}$ is defined as in* SCALARC*: in data structure* mp_t*,*

$$c := (\frac{t^{\mathrm{new}}}{t} - 1),$$

and in data structure mp_Φ*,*

$$c := -\frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{1}{\sqrt{t} \|\nabla \Phi_\lambda(\widetilde{\mu}/t - 1)\|_2}.$$

Proof. We consider the j-th iteration in this proof. Throughout the proof, we call the updated version of all date structure members at the end of the j-th iteration as the "new" version, with a "new" superscript in the notation, e.g. u_1^{new}.

Part 1. Note that V, $\widetilde{V}$ and G are only updated in MATRIXUPDATE and PARTIALMATRIXUPDATE.

Case 1. MATRIXUPDATE (Algorithm 25). On Line 12, G is updated to be $W^{\mathrm{appr}} M^{\mathrm{tmp}}$, where W^{appr} is the new value of $\widetilde{V}$ (Line 18), and M^{tmp} is the new value of M (Line 16). We have

$$G^{\mathrm{new}} = \widetilde{V} M = \widetilde{V} A^\top (A V A^\top)^{-1} A,$$

since in Lemma D.17 we proved that the invariant of M still holds.

Case 2. PARTIALMATRIXUPDATE (Algorithm 26). First note that V and M are not updated in PARTIALMATRIXUPDATE. On Line 14, G is updated to be $W^{\mathrm{appr}} M$, where W^{appr} is the new value of $\widetilde{V}$, so the invariant of G still holds.

Part 2. It suffices to prove that the additive term in the j-th iteration is

$$u_3^{\mathrm{new}} + M^{\mathrm{new}} u_4^{\mathrm{new}} - (u_3 + M u_4) = c \cdot A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}).$$

Then starting from the j_1-th iteration where we re-INITIALIZE and set $u_3^{(j_1)} = u_4^{(j_1)} = 0$, we can inductively prove the statement for iterations $i \in \{j_1 + 1, \cdots, j\}$.

There are three cases that we update u_3 and u_4 in the j-th iteration: in MATRIXUPDATE, PARTIALMATRIXUPDATE, or QUERY. Note that even though we might enter QUERY after executing MATRIXUPDATE or PARTIALMATRIXUPDATE, u_3 and u_4 won't change inside QUERY since they were already updated in MATRIXUPDATE or PARTIALMATRIXUPDATE. So we can consider the updates to u_3 and u_4 in these three procedures separately.

Case 1, MATRIXUPDATE (Algorithm 25).

$$\begin{aligned} u_3^{\mathrm{new}} + M^{\mathrm{new}} u_4^{\mathrm{new}} - (u_3 + M u_4) &= u_3 + M u_4 + c \cdot M^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) + M^{\mathrm{new}} \cdot 0 - (u_3 + M u_4) \\ &= c \cdot M^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) \\ &= c \cdot A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}), \end{aligned}$$

where the first step is by the assignment of u_3^{new} (Line 14), u_4^{new} (Line 15), the last step is by $M^{\mathrm{tmp}} = A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A$ that we already proved in Lemma D.17.

Case 2, PARTIALMATRIXUPDATE (Algorithm 26).

$$\begin{aligned} & u_3^{\mathrm{new}} + M^{\mathrm{new}} u_4^{\mathrm{new}} - (u_3 + M u_4) \\ = \; & M (u_4^{\mathrm{new}} - u_4) \\ = \; & M \cdot c \cdot \Big(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \mathbf{1}_{S^{\mathrm{new}}} B^{\mathrm{tmp}} \mathcal{L}_r [(M_{S^{\mathrm{new}}})^\top] \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) \Big) \end{aligned}$$

$$
\begin{aligned}
&= c \cdot \Big(M - \mathcal{L}_c[M_{S^{\mathrm{new}}}] B^{\mathrm{tmp}} \mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top] \Big) \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) \\
&= c \cdot \Big(M - \mathcal{L}_c[M_{S^{\mathrm{new}}}] \cdot \mathcal{L}_*[((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1} + M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}] \cdot \mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top] \Big) \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) \\
&= c \cdot A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}),
\end{aligned}
$$

where the first step is by $u_3^{\mathrm{new}} = u_3$ and $M^{\mathrm{new}} = M$ since they are not modified in PartialMatrixUpdate, the second step is by the assignment of u_4^{new} (Line 17), the fourth step is by the invariant on B^{tmp} (Part 10 of Assumption D.1), the fifth step is by Woodbury identity (Lemma C.8) and $M = A^\top (AVA^\top)^{-1} A$.

Case 3, Query (Algorithm 24).

$$
\begin{aligned}
& u_3^{\mathrm{new}} + M^{\mathrm{new}} u_4^{\mathrm{new}} - (u_3 + M u_4) \\
&= M(u_4^{\mathrm{new}} - u_4) \\
&= M \cdot c \cdot \Big(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) + \mathbf{1}_{S^{\mathrm{new}}} (\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma) \Big) \\
&= c \cdot \Big(M - \mathcal{L}_c[M_{S^{\mathrm{new}}}] \cdot \mathcal{L}_*[((\Delta^{\mathrm{new}}_{S^{\mathrm{new}},S^{\mathrm{new}}}) + M_{S^{\mathrm{new}},S^{\mathrm{new}}})^{-1}] \cdot L_r[(M_{S^{\mathrm{new}}})^\top] \Big) \cdot \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) \\
&= c \cdot A^\top (A W^{\mathrm{appr}} A^\top)^{-1} A \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}),
\end{aligned}
$$

where the first step is by $u_3^{\mathrm{new}} = u_3$ and $M^{\mathrm{new}} = M$ since they are not modified in MatrixUpdate, the second step is by the assignment of u_4^{new}(Line 25), the third step is by the close-form formula of $\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma$ (Eq. (42)), the fourth step is by Woodbury identity (Lemma C.8) and $M = A^\top (AVA^\top)^{-1} A$.

Part 3. The proof for $u_1 + Gu_2 = \sum_{i=j_1+1}^{j} c^{(i)} \cdot W^{\mathrm{appr},(i)} A^\top (A W^{\mathrm{appr},(i)} A^\top)^{-1} A \sqrt{W^{\mathrm{appr},(i)}} f(h^{\mathrm{appr},(i)})$ follows from similar reasons as that of Part 2. We omit the details here.

□

I.3 Bounding x and $\overline{x}$

In this section we prove that the explicit $\overline{x}$ is always within an error of ϵ_{tiny} with the implicitly maintained x. This fact is used to prove Part 3 of Theorem I.2. Similar as in previous sections, we use a superscript (j) to denote the variable at the beginning of the j-th iteration.

Remark I.4. *Our entire analysis is an induction-based argument. In the j-th iteration, the induction hypothesis allows us to assume that Assumption B.5 is true for the $(j-1)$-th iteration, it also allows us to assume the following Lemma I.5 is true for $x^{(j)}$ and $\overline{x}^{(j)}$.*

Using these two induction hypothesis we proved that Assumption B.5 is still true for the j-th iteration in Section I.1.

Then we further use these two induction hypothesis together with Assumption B.5 for the j-th iteration *to prove the following Lemma I.5.*

Lemma I.5 (x and $\overline{x}$ are close)**.** *Consider a fixed coordinate $i \in [n]$ and a fixed iteration number $k \leq \sqrt{n}$. Let b denote the size of sketching matrix. If the following are true: (1) $x_i^{(0)} = \overline{x}_i^{(0)}$. (2) $t^{(k)} > t^{(0)}/2$. (3) There is a constant $w > 0$ such that for all $j \in [k]$, $w_i^{\mathrm{appr},(j)} \in [w/2, 2w]$. (4) Inductively Assumption B.5 is true for iterations $1, 2, \ldots, k$. Then we have:*

$$
|(x_i^{(k)})^{-1} (\overline{x}_i^{(k)} - x_i^{(k)})| \leq \epsilon_x
$$

holds with probability $1 - \delta$ over the randomness of sketching matrices $R^{(1)}, \cdots, R^{(k)} \in \mathbb{R}^{b \times n}$, where $\epsilon_x = \frac{200 n^{1/4}}{\sqrt{b}} \cdot \log(n/\delta) \epsilon$.

Proof. We denote $t = t^{(k)}$. Since $t^{(0)} \leq 2t^{(k)}$ and the central path iteration will only decrease $t^{(j)}$, we have $t \leq t^{(j)} \leq 2t$ for all $j \in [k]$.

From the definition of $\widetilde{\delta}_x$ (Definition B.4) and $\widehat{\delta}_x$ (Definition B.6), we have

$$\begin{aligned}\widetilde{\delta}_x - \widehat{\delta}_x &= \frac{\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}}(I - \widetilde{P})\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu - \frac{\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}}(I - R^\top R\widetilde{P})\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu \\ &= \frac{\widetilde{X}}{\sqrt{\widetilde{X}\widetilde{S}}}(\widetilde{P} - R^\top R\widetilde{P})\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu = \sqrt{W^{\mathrm{appr}}}(\widetilde{P} - R^\top R\widetilde{P})\frac{1}{\sqrt{\widetilde{X}\widetilde{S}}}\widetilde{\delta}_\mu. \end{aligned} \tag{76}$$

Then the difference between $x_i^{(k)}$ and $\overline{x}_i^{(k)}$ can be written as

$$\begin{aligned}|x_i^{(k)} - \overline{x}_i^{(k)}| &= \Big|\big(x_i^{(0)} + \sum_{j=1}^k \widetilde{\delta}_{x,i}^{(j)}\big) - \big(\overline{x}_i^{(0)} + \sum_{j=1}^k \widehat{\delta}_{x,i}^{(j)}\big)\Big| = \Big|\sum_{j=1}^k (\widetilde{\delta}_{x,i}^{(j)} - \widehat{\delta}_{x,i}^{(j)})\Big| \\ &= \Big|\sum_{j=1}^k \sqrt{w^{\mathrm{appr},(j)}}_i \big((\widetilde{P}^{(j)} - R^{\top(j)}R^{(j)}\widetilde{P}^{(j)})\frac{1}{\sqrt{\widetilde{X}^{(j)}\widetilde{S}^{(j)}}}\widetilde{\delta}_\mu^{(j)}\big)_i\Big|, \end{aligned} \tag{77}$$

where the second step is by $x_i^{(0)} = \overline{x}_i^{(0)}$, the third step is by Eq.(76).

We define a random vector Y_j for the j-th iteration: $Y_j = \sqrt{w^{\mathrm{appr},(j)}}_i \Big((I - R^{(j)\top}R^{(j)})\widetilde{P}^{(j)}\frac{\widetilde{\delta}_\mu^{(j)}}{\sqrt{\widetilde{X}^{(j)}\widetilde{S}^{(j)}}}\Big)_i$.

We first bound the ℓ_2 norm of the right part $\frac{\widetilde{\delta}_\mu^{(j)}}{\sqrt{\widetilde{X}^{(j)}\widetilde{S}^{(j)}}}$:

$$\Big\|\frac{\widetilde{\delta}_\mu^{(j)}}{\sqrt{\widetilde{X}^{(j)}\widetilde{S}^{(j)}}}\Big\|_2 \leq \mathbf{Sup}\Big[\frac{1}{\sqrt{\widetilde{X}^{(j)}\widetilde{S}^{(j)}}}\Big] \cdot \|\widetilde{\delta}_\mu^{(j)}\|_2 \leq \frac{1.1}{\sqrt{t^{(j)}}} \cdot \|\widetilde{\delta}_\mu^{(j)}\|_2 \leq 2.2\epsilon\sqrt{t^{(j)}} \leq 5\epsilon\sqrt{t},$$

where the first step is by $\|a \cdot b\|_2 \leq \mathbf{Sup}[a] \cdot \|b\|_2$, the second step is by the inductive Assumption B.5 that $\widetilde{X}^{(j)}\widetilde{S}^{(j)} \approx_{0.1} t^{(j)}$, the third step is by $\|\widetilde{\delta}_\mu^{(j)}\|_2 \leq 2\epsilon t^{(j)}$ (Fact B.9), the last step is by $t^{(j)} \leq 2t$.

By Lemma B.14, for each j and with randomness over $R^{(j)}$, and use the fact that $w_i^{\mathrm{appr},(j)} \in [w/2, 2w]$ for all $j \in [k]$, we have

$$\mathbb{E}[Y_j] = 0 \quad \text{and} \quad \mathbb{E}[(Y_j)^2] \leq \frac{w_i^{\mathrm{appr},(j)}}{b}\Big\|\frac{\widetilde{\delta}_\mu^{(j)}}{\sqrt{\widetilde{X}^{(j)}\widetilde{S}^{(j)}}}\Big\|_2^2 \leq (2w) \cdot 25\epsilon^2 t/b,$$

and with probability $1 - \delta/n$,

$$|Y_j| \leq \sqrt{w_i^{\mathrm{appr},(j)}} \cdot \Big\|\frac{\widetilde{\delta}_\mu^{(j)}}{\sqrt{\widetilde{X}^{(j)}\widetilde{S}^{(j)}}}\Big\|_2 \cdot \frac{\log(n/\delta)}{\sqrt{b}} \leq (\sqrt{2w}) \cdot 5\epsilon\sqrt{t}\frac{\log(n/\delta)}{\sqrt{b}} := M.$$

Now, we apply Bernstein inequality on these zero-mean independent random variable $\{Y_j\}_{j=1}^k$ (Lemma A.3), $\forall \tau > 0$,

$$\Pr\Big[|\sum_{j=1}^k (Y_j)| > \tau\Big] \leq 2\exp\Big(-\frac{\tau^2/2}{\sum_{j=1}^k \mathbb{E}[(Y_j)^2] + M\tau/3}\Big)$$

Choosing $\tau = 64\frac{\sqrt{wkt}}{\sqrt{b}}\log(n/\delta)^2\epsilon$, we have

$$\Pr\Big[|\sum_{j=1}^k (Y_j)| > \tau\Big] \leq 2\exp\Big(-\frac{\tau^2/2}{50wk\epsilon^2 t/b + \tau \cdot 5\epsilon\sqrt{2wt}\frac{\log(n/\delta)}{3\sqrt{b}}}\Big) \leq 2\exp(-10\log(n/\delta)).$$

Then take a union bound on all events that $|Y_j| \leq M$, we have $|\sum_{j=1}^k (Y_j)| \leq \tau$ with probability at least $1 - \delta$. Therefore,

$$\begin{aligned} |x_i^{(k)} - \overline{x}_i^{(k)}| \leq |\sum_{j=1}^k (Y_j)| \leq \tau \leq \frac{64\sqrt{wkt}}{\sqrt{b}} \cdot \log(n/\delta)^2 \epsilon \\ \leq x_i^{(k)} \cdot \frac{200\sqrt{k}}{\sqrt{b}} \cdot \log(n/\delta)^2 \epsilon \leq x_i^{(k)} \cdot \frac{200 n^{1/4}}{\sqrt{b}} \cdot \log(n/\delta)^2 \epsilon \leq x_i^{(k)} \epsilon_x, \end{aligned}$$

where the first step is by Eq.(77), the fourth step is by $wt \leq 2w_i^{(\mathrm{appr}),(k)} \cdot 2t^{(k)} \leq (2x_i^{(k)}/s_i^{(k)}) \cdot (3x_i^{(k)} s_i^{(k)}) \leq 6(x_i^{(k)})^2$ by Assumption B.5, the fifth step is by $k \leq \sqrt{n}$. □

I.4 Running time of feasible data structure

Lemma I.6. *It takes $O(n)$ time to execute* ScalarC.

Proof. The proof is straightforward. □

Lemma I.7. *In the procedure* MatrixUpdate*, it takes*

1. *$O(n^2)$ time to compute $G \leftarrow W^{\mathrm{appr}} M^{\mathrm{tmp}}$*
2. *$O(n^2)$ time to compute $u_1 \leftarrow u_1 + G u_2 + c \cdot W^{\mathrm{appr}} M^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$*
3. *$O(n^2)$ time to compute $u_3 \leftarrow u_3 + M u_4 + c \cdot M^{\mathrm{tmp}} \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$*

Overall, refreshing G, u_1, u_2 takes $O(n^2)$ time.

Proof. This lemma directly follows from the algorithm of MatrixUpdate (ALgorithm 25). □

Lemma I.8. *In the procedure* PartialMatrixUpdate*, it takes*

1. *$O(n^{1+a})$ time to compute $G \leftarrow G + (W^{\mathrm{appr}} - \widetilde{V}) \cdot M$.*
2. *$O(n^{1+a})$ time to compute $u_1 \leftarrow u_1 + (W^{\mathrm{appr}} - \widetilde{V}) \cdot M u_2$.*
3. *$O(n^{1+a})$ time to compute $u_2 \leftarrow u_2 + c \Big(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \mathbf{1}_{S^{\mathrm{new}}} B^{\mathrm{tmp}} \mathcal{L}_r [(M_{S^{\mathrm{new}}})^\top] \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) \Big)$. And computing u_4 takes the same time.*

Overall, refreshing G, u_1, u_2, u_4 takes $O(n^{1+a})$ time.

Proof. From Lemma E.1, we have that when entering PartialUpdate, $\|w^{\mathrm{appr}} - \widetilde{v}\|_0 \leq O(n^a)$, and $|S^{\mathrm{new}}| \leq O(n^a)$.
Part 1. Since $(W^{\mathrm{appr}} - \widetilde{V})$ is a $O(n^a)$-sparse diagonal matrix, multiplying it with a $n \times n$ matrix M takes $O(n^{1+a})$ time. By memory operation, adding $(W^{\mathrm{appr}} - \widetilde{V})M$ on G also takes $O(n^{1+a})$ time.
Part 2. Multiplying a $O(n^a)$-sparse diagonal matrix $(W^{\mathrm{appr}} - \widetilde{V})$ with a $n \times n$ matrix M and then with a $n \times 1$ vector u_2 takes $O(n^{1+a})$ time.
Part 3. Computing $\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$ takes $O(n)$ time. Multiplying a $O(n^a) \times n$ matrix $\mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top]$ with a $n \times 1$ vector $\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$ takes $O(n^{1+a})$ time. Multiplying a $O(n^a) \times O(n^a)$ matrix B^{tmp} with a $O(n^a) \times 1$ vector $\mathcal{L}_r[(M_{S^{\mathrm{new}}})^\top] \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$ takes $O(n^{2a})$ time. Finally multiplying a $n \times O(n^a)$ matrix $\mathbf{1}_{S^{\mathrm{new}}}$ that only has $O(n^a)$ non-zero entries with a $O(n^a) \times 1$ vector $B^{\mathrm{tmp}} (M_{S^{\mathrm{new}}})^\top \sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}})$ takes $O(n)$ time. Thus in total this step takes $O(n^{1+a})$ time. □

Remark I.9. *Note that this running time of $O(n^{1+a})$ is always dominated by the $O(\mathcal{T}_{\mathrm{mat}}(n, n^a, \widetilde{k}))$ time of other computations of* PartialMatrixUpdate *(see Lemma E.18).*

Lemma I.10. *In the procedure* Query*, it takes*

1. *$O(n^{a+\widetilde{a}})$ time to compute $u_1 \leftarrow u_1 + c \cdot (W^{\mathrm{appr}} - \widetilde{V}) \Big(\beta_2 + M \cdot \big(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \sqrt{V} f(g) + \mathbf{1}_{S^{\mathrm{new}}} (\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma) \big) \Big)$.*
2. *$O(n)$ time to compute $u_2 \leftarrow u_2 + c \cdot \Big(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) + \mathbf{1}_{S^{\mathrm{new}}} (\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma) \Big)$. And computing u_4 takes the same time.*

Overall, calculating u_1, u_2, u_4 takes $O(n^{a+\widetilde{a}})$ time.

Proof. When entering Query, from Lemma E.1, we have that $\|w^{\mathrm{appr}} - v\|_0 \leq n^a$ and $\|h^{\mathrm{appr}} - g\|_0 \leq n^a$, thus $\|\sqrt{W}^{\mathrm{appr}} f(h^{\mathrm{appr}}) - \sqrt{V} f(g)\|_0 \leq O(n^a)$. From Lemma E.1, we also have that $\|w^{\mathrm{appr}} - \widetilde{v}\|_0 \leq n^{\widetilde{a}}$, and $|S^{\mathrm{new}}| \leq O(n^a)$.
Part 1. We need to compute the following four parts:

1. Multiplying a $n \times n$ diagonal matrix $W^{\mathrm{appr}} - \widetilde{V}$ with a $n \times 1$ vector β_2 takes $O(n)$ time.
2. Multiplying a $n^{\widetilde{a}}$-sparse $n \times n$ diagonal matrix $W^{\mathrm{appr}} - \widetilde{V}$ with a $n \times n$ matrix M and then with a $O(n^a)$-sparse $n \times 1$ vector $(\sqrt{W^{\mathrm{appr}}} f(h^{\mathrm{appr}}) - \sqrt{V} f(g))$ takes $O(n^{a+\widetilde{a}})$ time.
3. Computing $\mathbf{1}_{S^{\mathrm{new}}} (\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)$ takes $O(n)$ time. Multiplying a $n^{\widetilde{a}}$-sparse $n \times n$ diagonal matrix $W^{\mathrm{appr}} - \widetilde{V}$ with a $n \times n$ matrix M and then with a $O(n^a)$-sparse $n \times 1$ vector $\mathbf{1}_{S^{\mathrm{new}}} (\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)$ takes $O(n^{a+\widetilde{a}})$ time.

So overall this step takes $O(n^{a+\widetilde{a}})$ time.
Part 2. Computing $\sqrt{W^{\mathrm{appr}}} \cdot f(h^{\mathrm{appr}})$ takes $O(n)$ time. And multiplying a $n \times O(n^a)$ matrix $\mathbf{1}_{S^{\mathrm{new}}}$ that only has $O(n^a)$ non-zero entries with a $O(n^a) \times 1$ vector $(\gamma^{\mathrm{tmp}} - \gamma_1 - \partial\gamma)$ takes $O(n)$ time. Thus in total this step takes $O(n)$ time. □

Remark I.11. *Note that this running time of $O(n^{a+\widetilde{a}})$ is the same as other computations of* Query *(see Lemma E.3).*

Lemma I.12. *The amortized running time of procedure* MakeFeasible *is*

$$(C_1/\epsilon_{\mathrm{mp}} + C_2/\epsilon_{\mathrm{mp}}^2) \cdot n^{1.5}.$$

Proof. In this proof, we will use superscript notations that are consistent with section F. Let $\overline{w}^{(j+1)}$ denote the input w^{new} of UpdateQuery in the j-th iteration. Let $w^{\mathrm{appr},(j+1)}$ be the output of UpdateQuery in the j-th iteration. Let $w^{\mathrm{old},(j)}$ be the values of w^{old} at the beginning of the j-th iteration. Let $\widehat{S}^{(j)} := \{i : |w_i^{\mathrm{old},(j)} - w_i^{\mathrm{appr},(j+1)}| > w_i^{\mathrm{old},(j)}/2\}$, and we define $\widehat{k}_j := |\widehat{S}^{(j)}|$. Note that in the j-th iteration, procedure MakeFeasible takes worst-case $O(n \cdot \widehat{k}_j)$ time. We use a similar amortized argument as that of Lemma F.19.

We define the following potential function

$$\Phi_j = \sum_{i=1}^{n} \psi(w_i^{(j)} / w_i^{\mathrm{old},(j)} - 1),$$

where the function ψ is defined as Definition F.4. We let $g \in \mathbb{R}^n$ be the all one vector. Applying Lemma F.23 (set $v^{(j)}$ of the lemma statement to be $w^{\mathrm{old},(j)}$), we have

$$\begin{aligned}(w \text{ move})^{(j)} &:= \sum_{i=1}^n g_i \cdot \mathbb{E}\left[\psi(w_i^{(j+1)}/w_i^{\mathrm{old},(j)} - 1) - \psi(w_i^{(j)}/w_i^{\mathrm{old},(j)} - 1) \;\middle|\; w^{(j)}, w^{\mathrm{old},(j)}\right] \\ &= O(C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \|g\|_2 = O((C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \sqrt{n}). \end{aligned} \tag{78}$$

From Section I.1, we have $w^{\mathrm{appr},(j+1)} \approx_{\epsilon_{\mathrm{mp}}} \overline{w}^{(j+1)}$. And from Lemma I.5 we have $\overline{w}^{(j+1)} \approx_{\epsilon_{\mathrm{tiny}}} w^{(j+1)}$. So $w^{\mathrm{appr},(j+1)} \approx_{\epsilon_{\mathrm{mp}}+\epsilon_{\mathrm{tiny}}} w^{(j+1)}$. For every $i \in \widehat{S}^{(j)}$, we have $|w_i^{\mathrm{appr},(j+1)}/w_i^{\mathrm{old},(j)} - 1| > 1/2$ by definition of $\widehat{S}^{(j)}$, thus

$$\begin{aligned}|w_i^{(j+1)}/w_i^{\mathrm{old},(j)} - 1| &= |(w_i^{\mathrm{appr},(j+1)}/w_i^{\mathrm{old},(j)}) \cdot (w_i^{(j+1)}/w_i^{\mathrm{appr},(j+1)}) - 1| \\ &> 1/2 - 2\epsilon_{\mathrm{mp}} - 2\epsilon_{\mathrm{tiny}} > 2\epsilon_{\mathrm{mp}},\end{aligned}$$

where the last step follows from $\epsilon_{\mathrm{mp}} < \frac{1}{10}$ (Assumption F.5) and $\epsilon_{\mathrm{tiny}} < \frac{\epsilon_{\mathrm{mp}}}{1000}$ (Def. I.1). Thus $\psi(w_i^{(j+1)}/w_i^{\mathrm{old},(j)} - 1) > \epsilon_{\mathrm{mp}}$ from the definition of ψ. Since for $i \in \widehat{S}^{(j)}$, $w_i^{\mathrm{old},(j+1)}$ is updated to be $w_i^{\mathrm{appr},(j+1)}$, we have

$$|w_i^{(j+1)}/w_i^{\mathrm{old},(j+1)} - 1| = |w_i^{(j+1)}/w_i^{\mathrm{appr},(j+1)} - 1| \leq \epsilon_{\mathrm{mp}} + \epsilon_{\mathrm{tiny}} < 1.1\epsilon_{\mathrm{mp}}.$$

Thus $\psi(w_i^{(j+1)}/w_i^{\mathrm{old},(j+1)} - 1) < 0.6\epsilon_{\mathrm{mp}}$ from the definition of ψ (Definition F.4).

So we have

$$\begin{aligned}(v \text{ move})^{(j)} &:= \sum_{i=1}^n \mathbb{E}\left[\psi(w_i^{(j+1)}/w_i^{\mathrm{old},(j)} - 1) - \psi(w_i^{(j+1)}/w_i^{\mathrm{old},(j+1)} - 1) \;\middle|\; w^{(j)}, w^{\mathrm{old},(j)}\right] \\ &\geq \sum_{i \in \widehat{S}^{(j)}} \mathbb{E}\left[\psi(w_i^{(j+1)}/w_i^{\mathrm{old},(j)} - 1) - \psi(w_i^{(j+1)}/w_i^{\mathrm{old},(j+1)} - 1) \;\middle|\; w^{(j)}, w^{\mathrm{old},(j)}\right] \\ &\geq \sum_{i \in \widehat{S}^{(j)}} (\epsilon_{\mathrm{mp}} - 0.6\epsilon_{\mathrm{mp}}) = \Omega(\epsilon_{\mathrm{mp}} \widehat{k}_{j+1}). \end{aligned} \tag{79}$$

Combining Eq. (78) and Eq. (79), we have

$$\mathbb{E}[\Phi_T] - \Phi_0 = \sum_{j=0}^{T-1}((w \text{ move})^{(j)} - (v \text{ move})^{(j)}) \leq T \cdot (C_1 + C_2/\epsilon_{\mathrm{mp}}) \cdot \sqrt{n} - \sum_{j=1}^T \Omega(\epsilon_{\mathrm{mp}} \widehat{k}_j).$$

Since when initialize we set w^{old} to be w_0, we have $\Phi_0 = 0$. And since $\mathbb{E}[\Phi_T] \geq 0$, we have

$$\sum_{j=1}^T \widehat{k}_j \leq T \cdot (C_1/\epsilon_{\mathrm{mp}} + C_2/\epsilon_{\mathrm{mp}}^2) \cdot \sqrt{n}.$$

Thus the amortized running time of MakeFeasible is $(C_1/\epsilon_{\mathrm{mp}} + C_2/\epsilon_{\mathrm{mp}}^2) \cdot n^{1.5}$. □

Remark I.13. *Note that the amortized time of* MakeFeasible *is dominated by the amortized time of* MatrixUpdate *(Lemma F.19).*

J History of Matrix Multiplication and LP

Year	Reference	ω	Reference	α
1969	[Str69]	2.808		
1978	[Pan78]	2.796		
1979	[BCRL79]	2.78		
1981	[Sch81]	2.548		
1982	[Rom82]	2.517	[Cop82]	0.172
1982	[CW82]	2.496		
1986	[Str86]	2.479		
1987	[CW87]	2.376		
1997			[Cop97]	0.29462
2012	[Wil12]	2.3729		
2014	[LG14]	2.37286	[LG14]	0.30298
2018			[GU18]	0.31389

Table 16: The history of exponent of matrix multiplication ω and the dual exponent of matrix multiplication α.

Year	Author	Reference	Complexity
1947	Dantzig	[Dan47]	$2^{O(n)}$
1979	Khachiyan	[Kha80]	n^6
1984	Karmarkar	[Kar84]	$n^{3.5}$
1986	Renegar	[Ren88]	n^3
1987	Vaidya	[Vai87]	n^3
1989	Vaidya	[Vai89]	$n^{2.5}$
1994	Nesterov, Nemirovskii	[NN94]	$n^{2.5}$
2014	Lee, Sidford	[LS14]	$n^{2.5}$
2015	Lee, Sidford	[LS15]	$n^{2.5}$
2019	Cohen, Lee, Song	[CLS19]	$n^\omega + n^{2.5-\alpha/2} + n^{2+1/6}$
2019	Lee, Song, Zhang	[LSZ19]	$n^\omega + n^{2.5-\alpha/2} + n^{2+1/6}$
2020	Brand	[Bra20]	$n^\omega + n^{2.5-\alpha/2} + n^{2+1/6}$
2020	Brand, Lee, Sidford, Song	[BLSS20]	n^3
2020		This paper	$n^\omega + n^{2.5-\alpha/2} + n^{2+1/18}$

Table 17: Let ω denote the exponent of the current matrix multiplication. LP has n variables, $d = \Theta(n)$ constraints, and all number can be encoded in L bits. The running time of all these algorithms has a nearly linear dependence on L. We consider the case where A is a dense full rank matrix. ω denotes the exponent of matrix multiplication, and α denotes the dual exponent of matrix multiplication. We remark that in some previous papers the running time is presented with explicit d and $\mathrm{nnz}(A)$. Here we present the running time assuming $d = \Theta(n)$, $\mathrm{rank}(A) = n$ and $\mathrm{nnz}(A) = n^2$.