

Radial Isotropic Position via an Implicit Newton's Method

Arun Jambulapati*

Jonathan Li[†]

Kevin Tian[‡]

Abstract

Placing a dataset $A = \{\mathbf{a}_i\}_{i \in [n]} \subset \mathbb{R}^d$ in *radial isotropic position*, i.e., finding an invertible $\mathbf{R} \in \mathbb{R}^{d \times d}$ such that the unit vectors $\{(\mathbf{R}\mathbf{a}_i)\|\mathbf{R}\mathbf{a}_i\|_2^{-1}\}_{i \in [n]}$ are in isotropic position, is a powerful tool with applications in functional analysis, communication complexity, coding theory, and the design of learning algorithms. When the transformed dataset has a second moment matrix within a $\exp(\pm\epsilon)$ factor of a multiple of \mathbf{I}_d , we call \mathbf{R} an ϵ -approximate *Forster transform*.

We give a faster algorithm for computing approximate Forster transforms, based on optimizing an objective defined by Barthe [Bar98]. When the transform has a polynomially-bounded aspect ratio, our algorithm uses $O(nd^{\omega-1}(\frac{n}{\epsilon})^{o(1)})$ time to output an ϵ -approximate Forster transform with high probability, when one exists. This is almost the natural limit of this approach, as even evaluating Barthe's objective takes $O(nd^{\omega-1})$ time. Previously, the state-of-the-art runtime in this regime was based on cutting-plane methods, and scaled at least as $\approx n^3 + n^2d^{\omega-1}$. We also provide explicit estimates on the aspect ratio in the *smoothed analysis* setting, and show that our algorithm similarly improves upon those in the literature.

To obtain our results, we develop a subroutine of potential broader interest: a reduction from almost-linear time sparsification of graph Laplacians to the ability to support almost-linear time matrix-vector products. We combine this tool with new stability bounds on Barthe's objective to implicitly implement a box-constrained Newton's method [CMTV17, ALdOW17].

*Independent, jmblpati@gmail.com. Work completed while visiting the University of Texas at Austin.

[†]University of Texas at Austin, jli@cs.utexas.edu

[‡]University of Texas at Austin, kjtian@cs.utexas.edu

Contents

1	Introduction	1
1.1	Our results	3
1.2	Our techniques	7
1.3	Related work	9
2	Preliminaries	10
2.1	Notation	10
2.2	Radial isotropic position	12
3	Optimizing Barthe's Objective via Newton's Method	15
3.1	Hessian stability of Barthe's objective	15
3.2	Termination condition	18
3.3	Box-constrained Newton's method	19
3.4	Proof of Theorem 1	22
4	Sparsifying Laplacians with Matrix-Vector Queries	23
4.1	Sum-of-cliques representation	23
4.2	Discretizing low-dimensional distances	27
4.3	Long step packing SDP	31
4.4	Matrix dictionary recovery	37
4.5	Homotopy method	44
5	Conditioning of Smoothed Matrices	46
5.1	Diameter bound for deep vectors	46
5.2	Conditioning of wide and near-square smoothed matrices	49
5.3	Conditioning of tall smoothed matrices	52
5.4	Assumption 1 for smoothed matrices	54
5.5	Extension to non-uniform \mathbf{c}	56
A	Discussion of Proposition 5	62
B	Numerical Precision Considerations	62

1 Introduction

Transforming a dataset $A = \{\mathbf{a}_i\}_{i \in [n]} \subset \mathbb{R}^d$ into a canonical representation enjoying a greater deal of regularity is a powerful idea that has had myriad applications throughout computer science, statistics, and related fields. Examples of common such representations include the following.

- **Normalization:** Replacing each \mathbf{a}_i with the unit vector $\tilde{\mathbf{a}}_i := \mathbf{a}_i \|\mathbf{a}_i\|_2^{-1}$ in the same direction. Such a transformation exists whenever all of the $\{\mathbf{a}_i\}_{i \in [n]}$ are nonzero vectors.
- **Isotropic position:** Replacing each \mathbf{a}_i with $\tilde{\mathbf{a}}_i := \mathbf{R}\mathbf{a}_i$ for an invertible $\mathbf{R} \in \mathbb{R}^{d \times d}$, such that $\sum_{i \in [n]} \tilde{\mathbf{a}}_i \tilde{\mathbf{a}}_i^\top = \mathbf{I}_d$. Such a transformation exists whenever the $\{\mathbf{a}_i\}_{i \in [n]}$ span \mathbb{R}^d .

Recently, a common generalization of both of these representations known as *radial isotropic position* has emerged as a desirable data processing step in many settings.

Definition 1 (Radial isotropic position). *Let $\mathbf{c} \in (0, 1]^n$ satisfy $\|\mathbf{c}\|_1 = d$, and let $\epsilon \in (0, 1)$. We say that $\mathbf{A} \in \mathbb{R}^{n \times d}$ with rows $\{\mathbf{a}_i^\top\}_{i \in [n]}$ is in (\mathbf{c}, ϵ) -radial isotropic position (or, (\mathbf{c}, ϵ) -RIP) if*

$$\exp(-\epsilon)\mathbf{I}_d \preceq \sum_{i \in [n]} \mathbf{c}_i \cdot \frac{\mathbf{a}_i \mathbf{a}_i^\top}{\|\mathbf{a}_i\|_2^2} \preceq \exp(\epsilon)\mathbf{I}_d. \quad (1)$$

If ϵ is omitted then $\epsilon = 0$ by default, and if \mathbf{c} is omitted then $\mathbf{c} = \frac{d}{n}\mathbf{1}_n$ by default. For an invertible matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$, we say that \mathbf{R} is a (\mathbf{c}, ϵ) -Forster transform of \mathbf{A} if $\mathbf{A}\mathbf{R}^\top$ is in (\mathbf{c}, ϵ) -RIP:

$$\exp(-\epsilon)\mathbf{I}_d \preceq \sum_{i \in [n]} \mathbf{c}_i \cdot \frac{(\mathbf{R}\mathbf{a}_i)(\mathbf{R}\mathbf{a}_i)^\top}{\|\mathbf{R}\mathbf{a}_i\|_2^2} \preceq \exp(\epsilon)\mathbf{I}_d. \quad (2)$$

In other words, \mathbf{R} is a \mathbf{c} -Forster transform of $\mathbf{A} \in \mathbb{R}^{n \times d}$ representing the dataset $A = \{\mathbf{a}_i\}_{i \in [n]} \subset \mathbb{R}^d$, if the transformed-and-normalized vectors $\{(\mathbf{R}\mathbf{a}_i)\|\mathbf{R}\mathbf{a}_i\|_2^{-1}\}_{i \in [n]}$ are in isotropic position. Note that $\|\mathbf{c}\|_1 = d$ in Definition 1 is necessary as $\epsilon \rightarrow 0$, by taking traces of (1). For example, $\mathbf{c} = \frac{d}{n}\mathbf{1}_n$ induces an empirical second moment matrix with uniform weights. After applying a Forster transform, the new dataset then exhibits desirable properties that are useful in downstream applications.

Notably, although the concepts of radial isotropic position and Forster transforms first arose in early work on algebraic geometry [GGMS87] and functional analysis [Bar98], they have since enabled many surprising results in algorithms and complexity. For example, Forster transforms played a pivotal role in breakthroughs spanning disparate areas such as communication complexity [For02], subspace recovery [HM13], coding theory [DSW14], frame theory [HM19], active and noisy learning of halfspaces [HKLM20, DKT21, DTK23], and robust statistics [Che24].

Alternate characterizations of RIP. In fact, many of these results [For02, HM13, DSW14, HKLM20, DKT21, DTK23] leverage alternative characterizations of RIP, which themselves have powerful implications. We mention two here as they are relevant to our development.

First, [GGMS87, Bar98] (see also [CLL04, HKLM20]) give a tight characterization of when a \mathbf{c} -Forster transform of \mathbf{A} exists. In brief, one exists if and only if \mathbf{c} belongs to the *basis polytope* of the

independence matroid induced by A (see Propositions 1 and 2). A more intuitive and equivalent way of phrasing this result is that every k -dimensional linear subspace $V \subseteq \mathbb{R}^d$ must have

$$\sum_{\substack{i \in [n] \\ \mathbf{a}_i \in V}} \mathbf{c}_i \leq k. \quad (3)$$

The necessity of (3) is straightforward: if there exists a “heavy subspace” containing too many points (according to their weight by \mathbf{c}), then any transformation of the form $\mathbf{a}_i \rightarrow (\mathbf{R}\mathbf{a}_i) \|\mathbf{R}\mathbf{a}_i\|_2^{-1}$ retains the existence of such a heavy subspace. This in turn rules out the possibility of \mathbf{c} -RIP, because (1) cannot hold on the heavy subspace, simply by a trace argument. Further developments by e.g., [CLL04, HKLM20] showed that this is in fact the only barrier to Forster transforms.

Another dual viewpoint on Forster transforms is from the perspective of scaling the dataset to induce certain *leverage scores*. More precisely, [DR24] observed that finding $\mathbf{s} \in \mathbb{R}_{>0}^n$ such that

$$\boldsymbol{\tau}(\mathbf{S}\mathbf{A}) = \mathbf{c}, \text{ where } \mathbf{S} := \mathbf{diag}(\mathbf{s}), \quad (4)$$

implies that $\mathbf{R} = (\mathbf{A}^\top \mathbf{S}^2 \mathbf{A})^{-\frac{1}{2}}$ is a \mathbf{c} -Forster transform of \mathbf{A} . We recall this result in Lemma 1. Here, $\boldsymbol{\tau}(\mathbf{A}) \in \mathbb{R}^n$ denotes the *leverage scores* of a full-rank matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ (defined in (12)), a standard notion of the relative importance of points in a dataset. Thus, while Forster transforms are *right scalings* $\mathbf{R} \in \mathbb{R}^{d \times d}$ putting \mathbf{A} in isotropic position, we can equivalently find a *left scaling* $\mathbf{s} \in \mathbb{R}_{>0}^n$ that balances \mathbf{A} ’s rows to have target leverage scores \mathbf{c} of our choice.

Computing an approximate Forster transform. The goal of our work is designing efficient algorithms for computing a (\mathbf{c}, ϵ) -Forster transform of $\mathbf{A} \in \mathbb{R}^{n \times d}$, whenever one exists. This goal is inspired by advancements in the complexity of simpler, but related, dataset transformation problems called *matrix balancing and scaling*, for which [CMTV17, ALdOW17] achieved nearly-linear runtimes in well-conditioned regimes. Indeed, as the list of applications of radial isotropic position grows, so too does the importance of designing efficient algorithms for finding them.

Given the algorithmic significance of Forster transforms, it is perhaps surprising that investigations of their computational complexity are relatively nascent. Previous strategies for obtaining polynomial-time algorithms can largely be grouped under two categories.

Optimizing Barthe’s objective. In a seminal work on radial isotropic position [Bar98], Barthe observed that the minimizer of the convex objective $f : \mathbb{R}^n \rightarrow \mathbb{R}$ defined as

$$f(\mathbf{t}) := -\langle \mathbf{c}, \mathbf{t} \rangle + \log \det \left(\sum_{i \in [n]} \exp(\mathbf{t}_i) \mathbf{a}_i \mathbf{a}_i^\top \right), \quad (5)$$

induces a \mathbf{c} -Forster transform whenever it exists. Concretely, letting \mathbf{t}^* minimize f , we have that $\mathbf{s} := \exp(\mathbf{t}^*)$, where \exp is applied entrywise, satisfies (4) (cf. Proposition 3). Thus, to obtain a Forster transform it is enough to efficiently optimize (5).

This approach was followed by [HM13] (see also discussion in [HM19, Che24]), who proceeded via cutting-plane methods (CPMs), and [AKS20], who used first-order methods (e.g., gradient descent). However, it is somewhat challenging to quantify the accuracy needed in solving (5) to induce a

(\mathbf{c}, ϵ) -Forster transform for $\epsilon > 0$, because Barthe's objective is not strongly convex. For example, combining Lemmas B.6, B.9 of [HM13] with Corollary 4 of [HM19] gives an estimate of $\approx \epsilon \exp(-nd)$ additive error sufficing. Our work drastically improves this estimate (cf. Lemma 3), showing it is enough to obtain an additive error that is polynomial in ϵ , and $\min_{i \in [n]} \mathbf{c}_i$.

An optimistic bound on [HM13]'s runtime scales as¹ $\approx n^2 d^{\omega-1} + n^3$ (using the state-of-the-art CPM [JLSW20]), where additional $\text{poly}(n, d)$ factors are saved using our improved error bounds. Incomparably, [AKS20] gave runtimes for first-order methods depending polynomially on either the inverse target accuracy $\frac{1}{\epsilon}$ (and hence precluding high-accuracy solutions), or the inverse strong convexity of Barthe's objective, which is data-dependent but can lose $\exp(d)$ factors or worse.

Iterative scaling methods. Instead of optimizing Barthe's objective, [DTK23, DR24] recently gave alternative approaches that either iteratively refine a right-scaling $\mathbf{R} \in \mathbb{R}^{d \times d}$ to satisfy (2), or refine a left-scaling $\mathbf{s} \in \mathbb{R}_{>0}^n$ to satisfy (4). These algorithms have the advantage of running in *strongly polynomial time*, i.e., the number of arithmetic operations needed only depends on n and $\frac{1}{\epsilon}$, rather than problem conditioning notions such as bit complexity. Designing strongly polynomial time algorithms is an interesting and important goal in its own right. For instance, [DTK23] was motivated by the connection of Forster transforms to learning halfspaces with noise [DKT21], a robust generalization of linear programming, which is a basic problem for which strongly polynomial time algorithms are unknown. In a different direction, [DR24] showed that a strongly polynomial algorithm for matrix scaling by [LSW00] could be adapted to Forster transforms.

Unfortunately, the resulting runtimes from these direct iterative methods that sidestep Barthe's objective are quite large. For example, [DTK23] claim a runtime of at least $\approx n^5 d^{11} \epsilon^{-5}$, and a recent improvement in [DR24] still requires at least $\approx n^4 d^{\omega-1} \log(\frac{1}{\epsilon})$ time.

Outlook. There are a few other approaches to polynomial-time computation of approximate Forster transforms based on more general formulations of the problem, see e.g., [AGL⁺18, SV19]. We discuss these algorithms in more detail in Section 1.3, but note that they appear to lack explicit runtime bounds, and we believe they are subsumed by those described thus far.

In summary, existing methods for computing (\mathbf{c}, ϵ) -Forster transforms have runtimes at least $\approx n^2 d^{\omega-1} + n^3$ (weakly polynomial) or $\approx n^4 d^{\omega-1}$ (strongly polynomial). On the other hand, for related problems such as matrix scaling, near-optimal runtimes are known in well-conditioned regimes, via structured optimization methods that more faithfully capture the geometry of relevant objectives [CMTV17, ALdOW17]. This state of affairs prompts the natural question: can we obtain substantially faster algorithms for computing a (\mathbf{c}, ϵ) -Forster transform?

1.1 Our results

Our main contribution is to design such algorithms, primarily specialized to two settings which we call the *well-conditioned* and *smoothed analysis* regimes. We note that the distinction between well-conditioned and poorly-conditioned instances is a common artifact of fast algorithms for scaling problems, see e.g., discussions in [CMTV17, ALdOW17, BLNW20]. In particular, analogous works to ours for matrix scaling and balancing [CMTV17, ALdOW17] obtain nearly-linear runtimes in well-conditioned regimes, and polynomial runtime improvements in others.

¹We use $\omega < 2.372$ to denote the exponent of the square matrix multiplication runtime [ADV⁺25]. For all notation used throughout the paper, see Section 2.1.

For a fixed pair $\mathbf{A} \in \mathbb{R}^{n \times d}$ and $\mathbf{c} \in (0, 1]^n$ satisfying $\|\mathbf{c}\|_1 = d$, we use the following notion of conditioning for the associated problem of computing a \mathbf{c} -Forster transform of \mathbf{A} .

Assumption 1. *For f defined in (5), there is $\mathbf{t}^* \in \arg \min_{\mathbf{t} \in \mathbb{R}^n} f(\mathbf{t})$ satisfying $\|\mathbf{t}^*\|_\infty \leq \log(\kappa)$.*

To justify this, recall that $\mathbf{t}^* \in \arg \min_{\mathbf{t} \in \mathbb{R}^n} f(\mathbf{t})$ induces the optimal left scaling, in the sense of (4), via $\mathbf{s}(\mathbf{t}^*) = \exp(\frac{1}{2}\mathbf{t})$, entrywise. Further, Barthe's objective is invariant to translations by $\mathbf{1}_n$:

$$\begin{aligned} f(\mathbf{t} + \alpha \mathbf{1}_n) &= -\langle \mathbf{c}, \mathbf{t} + \alpha \mathbf{1}_n \rangle + \log \det(\mathbf{Z}(\mathbf{t} + \alpha \mathbf{1}_n)) \\ &= -\langle \mathbf{c}, \mathbf{t} \rangle - \alpha d + \log \det(\mathbf{Z}(\mathbf{t})) + \log \det(\exp(\alpha) \mathbf{I}_d) = f(\mathbf{t}). \end{aligned} \quad (6)$$

Thus, we can always shift any minimizing \mathbf{t}^* so that its extreme coordinates average to 0, which achieves the tightest ℓ_∞ bound on \mathbf{t}^* via shifts by $\mathbf{1}_n$. This shows that κ in Assumption 1 is the ratio of the largest and smallest entries of the optimal scaling $\mathbf{s} \in \mathbb{R}_{>0}^n$ achieving (4).

Radial isotropic position. We now state our main result on computing Forster transforms.

Theorem 1. *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{c} \in (0, 1]^n$ satisfy Assumption 1, and let $\delta, \epsilon \in (0, 1)$. There is an algorithm that computes \mathbf{R} , a (\mathbf{c}, ϵ) -Forster transform of \mathbf{A} , with probability $\geq 1 - \delta$, in time*

$$O\left(nd^{\omega-1} \log(\kappa) \left(\frac{n \log(\kappa)}{\delta \epsilon \mathbf{c}_{\min}}\right)^{o(1)}\right), \text{ where } \mathbf{c}_{\min} := \min_{i \in [n]} \mathbf{c}_i.$$

In the well-conditioned regime where $\kappa = \text{poly}(n)$, Theorem 1 improves upon the state-of-the-art runtimes for radial isotropic position by a factor of $\approx \max(n, n^2 d^{1-\omega})$, up to a subpolynomial overhead in problem parameters.² Moreover, Theorem 1 approaches natural limits for computing Forster transforms. For example, using current techniques, it takes $\approx nd^{\omega-1}$ time to perform basic relevant operations such as evaluating Barthe's objective (5), or verifying that a given right scaling $\mathbf{R} \in \mathbb{R}^{d \times d}$ or left scaling $\mathbf{s} \in \mathbb{R}_{>0}^n$ places a dataset in radial isotropic position.

Interestingly, we prove Theorem 1 by adapting the box-constrained Newton's method of [CMTV17, ALdOW17] to Barthe's objective, discussed further in Section 1.2. In fact, such an approach had been considered previously by [CKV20] for a more general problem of computing *maximum-entropy distributions*. We discuss this related result in more detail in Section 1.3, but briefly mention that based on the runtime analysis in [CKV20], the resulting complexity is significantly slower than CPMs, and we require several new structural observations and algorithmic insights to improve the efficiency of this approach. Indeed, as one example, it is perhaps surprising that Theorem 1's runtime depends linearly on n , given that it is a second-order method: merely writing down the Hessian of Barthe's objective takes n^2 time, which dominates the runtime of Theorem 1 for $n \gg d$.

We discuss the key algorithmic innovation that enables our result subsequently, but mention here that its use leads to the subpolynomial overheads in Theorem 1. By using explicit Hessian evaluations, we obtain an alternate runtime of

$$O\left(n^2 d^{\omega-2} \log(\kappa) \text{polylog}\left(\frac{n \log(\kappa)}{\delta \epsilon \mathbf{c}_{\min}}\right)\right),$$

²As discussed earlier, to our knowledge, even that CPMs [HM13] obtain runtimes of $\approx n^2 d^{\omega-1} + n^3$ for well-conditioned instances was unknown previously. This is enabled by our improved error tolerance analysis in Lemma 3.

as described more formally in Lemma 5 and Remark 1, which yields (improved) polylogarithmic dependences on $\frac{1}{\delta}$, $\frac{1}{\epsilon}$, and $\frac{1}{c_{\min}}$, at the cost of a multiplicative overhead of $\approx \frac{n}{d}$.

Sparsification via matrix-vector products. Our fastest runtimes are obtained by using a technical tool of potential independent interest that we develop. To explain its relevance to our setting, while the Hessian of Barthe's objective $\nabla^2 f$ is $n \times n$ and fully dense (a formula is given in Fact 2), its structure is appealing in several regards. While we do not know how to compute $\nabla^2 f$ faster than in $\approx n^2 d^{\omega-2}$ time, we can access it via matrix-vector products in $O(nd^{\omega-1})$ time (cf. Lemma 5). In addition, $\nabla^2 f$ is actually a *graph Laplacian*, i.e., it belongs to a family of matrices that have enabled many powerful algorithmic primitives, such as *spectral sparsification*. For example, breakthroughs by [SS11, ST14] show that any $n \times n$ graph Laplacian \mathbf{L} admits constant-factor spectral approximations with only $\approx n$ nonzero entries.

In this work, we add a new primitive to the graph Laplacian toolkit. We consider the following problem, which to our knowledge has not been explicitly studied before: given an (implicit) Laplacian \mathbf{L} accessible only via a matrix-vector product oracle, how many queries are needed to produce an (explicit) spectral sparsifier of \mathbf{L} ? The sparsifier can then be used as a preconditioner, enabling faster second-order methods. Our main result to this end is the following.

Theorem 2. *Let \mathbf{L} be an $n \times n$ graph Laplacian, and let $\mathcal{O} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be an oracle that returns $\mathbf{L}\mathbf{v}$ on input $\mathbf{v} \in \mathbb{R}^n$. Let $\delta \in (0, 1)$, $\Delta \in (0, \text{Tr}(\mathbf{L}))$, and let $\mathbf{\Pi} := \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$ be the projection matrix to the subspace of \mathbb{R}^n orthogonal to $\mathbf{1}_n$. There is an algorithm that takes as inputs $(\mathcal{O}, \delta, \Delta)$ and with probability $\geq 1 - \delta$, it returns $\tilde{\mathbf{L}}$, an $n \times n$ graph Laplacian satisfying*

$$\mathbf{L} + \Delta\mathbf{\Pi} \preceq \tilde{\mathbf{L}} \preceq \left(\frac{n\text{Tr}(\mathbf{L})}{\Delta\delta}\right)^{o(1)} (\mathbf{L} + \Delta\mathbf{\Pi}), \quad \text{nnz}(\tilde{\mathbf{L}}) = n \cdot \left(\frac{n\text{Tr}(\mathbf{L})}{\Delta\delta}\right)^{o(1)}, \quad (7)$$

using $(\frac{n\text{Tr}(\mathbf{L})}{\Delta\delta})^{o(1)}$ queries to \mathcal{O} , and $n \cdot (\frac{n\text{Tr}(\mathbf{L})}{\Delta\delta})^{o(1)}$ additional time.

For $\delta = \text{poly}(\frac{1}{n})$ and $\text{poly}(n)$ -well conditioned graph Laplacians, Theorem 2 produces a spectral sparsifier of a Laplacian \mathbf{L} using $n^{o(1)}$ matrix-vector products and $n^{1+o(1)}$ additional time. The approximation quality of the sparsifier is somewhat poor, i.e., $n^{o(1)}$, but in algorithmic contexts (such as that of Theorem 1), this is sufficient for use as a low-overhead preconditioner.

We believe Theorem 2 may be of independent interest to the graph algorithms and numerical linear algebra communities, as it enhances the flexibility of existing Laplacian-based tools; we discuss its connection to known results in more depth in Section 1.3. We are optimistic that its use can extend the reach of fast second-order methods for combinatorially-structured optimization problems.

Assumption 1 in the smoothed regime. Our third main contribution is to provide explicit bounds on the problem conditioning κ in Assumption 2, for “beyond worst-case” inputs \mathbf{A} . We specialize our result to the *smoothed analysis* setting, a well-established paradigm for beyond worst-case analysis in the theoretical computer science community [ST04, Rou20]. In our smoothed setting, we perturb entries of our input by Gaussian noise at noise level $\sigma > 0$. This is a standard smoothed matrix model used in the study of linear programming algorithms [ST04, SST06].

Here, we state the basic variant of our conditioning bound in the smoothed analysis regime.

Theorem 3. Let $\mathbf{A} \in \mathbb{R}^{n \times d}$ have rows $\{\mathbf{a}_i\}_{i \in [n]}$ such that $\|\mathbf{a}_i\|_2 = 1$ for all $i \in [n]$, let $\mathbf{c} := \frac{d}{n}\mathbf{1}_n$, let $\delta \in (0, 1)$, and let $\sigma \in (0, \frac{\delta}{10nd})$. Let $\tilde{\mathbf{A}} := \mathbf{A} + \mathbf{G}$, where $\mathbf{G} \in \mathbb{R}^{n \times d}$ has entries $\sim_{\text{i.i.d.}} \mathcal{N}(0, \sigma^2)$. Then with probability $\geq 1 - \delta$, if $n > Cd$ where C is any constant larger than 1, Assumption 1 holds for Barthe's objective f defined with respect to $(\tilde{\mathbf{A}}, \mathbf{c})$, where

$$\log(\kappa) = O\left(d \log\left(\frac{1}{\sigma}\right)\right).$$

That is, \mathbf{A} in Theorem 3 is a “base worst-case instance” that is smoothed into a more typical instance $\tilde{\mathbf{A}}$, which our conditioning bound of $\kappa \approx (\frac{1}{\sigma})^{O(d)}$ applies to.

The assumption in Theorem 3 that \mathbf{A} has unit norm rows is relatively mild; rescaling rows does not affect the (base) Forster transform problem, and our result still applies if row norms are in a $\text{poly}(n)$ multiplicative range. Further, while Theorem 3 is stated for uniform marginals $\mathbf{c} = \frac{d}{n}\mathbf{1}_n$, we show in Corollary 1 that as long as the marginals \mathbf{c} are bounded away from 1 entrywise by a constant, the conditioning estimate in Theorem 3 still holds for sufficiently large n . The requirement that $n > Cd$ for $C > 1$ is a minor bottleneck of our approach, discussed in Remark 3.

We are aware of few explicit conditioning bounds for Forster transforms such as Theorem 3, so we hope it (and techniques used in establishing it) become useful in future studies. Among conditioning bounds that exist presently, Lemma B.6 of [HM13] (cf. discussion in Corollary 4, [HM19]) shows that for \mathbf{A} with rows in *general position*, we have $\log(\kappa) = O(n \log(\frac{1}{D}))$, where D is the smallest determinant of a nonsingular $d \times d$ submatrix of \mathbf{A} . In particular, D can be inverse-exponential in d (or worse) for poorly-behaved instances. A crude lower bound of $D \gtrsim \exp(-d^3)$ was provided in [Che24] for essentially the smoothed model we consider in Theorem 3.

On the other hand, [DTK23, DR24], who respectively design strongly polynomial methods for iteratively updating an approximate Forster transform $\mathbf{R} \in \mathbb{R}^{d \times d}$ or dual scaling $\mathbf{s} \in \mathbb{R}_{>0}^n$, bound related conditioning quantities. Both papers contain results (cf. Section 5, [DTK23] and Section 4, [DR24]) showing that any iterate has a “nearby” iterate in bounded precision, that does not significantly affect some potential function of interest. These results do not appear to directly have implications for Assumption 1, and provide rather large bounds on the bit complexity (focusing on worst-case instances). Nonetheless, exploring connections in future work could be fruitful.

Most relatedly, Theorem 1.5 of [AKS20] proves that for target marginals \mathbf{c} that are “deep” inside the basis polytope for independent sets of \mathbf{A} ’s rows, $\log(\kappa) \lesssim d$. However, [AKS20] does not give estimates on the deepness of marginals in concrete models, and indeed our approach to proving Theorem 3 is to provide such explicit bounds in the smoothed analysis regime.

Directly combining Theorems 1 and 3 shows that for smoothed instances, the complexity of computing an approximate Forster transform is at most $\approx nd^\omega$, up to a subpolynomial factor. While it is worse than our well-conditioned runtime, our method in Theorem 1 still improves upon state-of-the-art algorithms based on CPMs by a factor of $\approx \max(\frac{n}{d}, n^3 d^{-\omega})$ in this regime.

Computational model. This paper works in the real RAM model, where we bound the number of basic arithmetic operations. Prior work on optimizing Barthe's objective [HM13, AKS20] also worked in this model, and we give a comparison on how these results are affected under finite-precision arithmetic in Appendix B. A more detailed investigation of the numerical stability of Forster transforms is an important direction for future work, but is outside our scope.

1.2 Our techniques

In this section, we overview our approaches to proving Theorems 1, 2, and 3.

Optimizing Barthe’s objective. Our algorithm for optimizing Barthe’s objective (5) is a variant of the *box-constrained Newton’s method* of [CMTV17, ALdOW17], originally developed for approximate matrix scaling and balancing. We were inspired to use this tool by noticing similarities between the derivative structure of Barthe’s objective (Fact 2) and the *softmax* function, which can be viewed as the one-dimensional case of Barthe’s objective. Previously, the softmax function was known to be *Hessian stable* in the ℓ_∞ norm (Definition 2), enabling local optimization oracles that can be implemented via Newton’s method [CJJ⁺20] over ℓ_2 or ℓ_∞ norm balls.

It is much more challenging to prove that Barthe’s objective is Hessian stable, as the proof in [CJJ⁺20] does not naturally extend to non-commuting variables. Nonetheless, we give a different proof inspired by Kadison’s inequality in operator algebra [Kad52] to establish Hessian stability of Barthe’s objective in Section 3.1. Our Hessian stability bound directly improves the best previously known in the literature, due to Lemma D.2 in [CKV20], by a factor of n , making it dimension-independent. This reflects in a multiplicative $O(n)$ savings in our final runtime.

We complement this result in Section 3.2 with bounds on the additive error on Barthe’s objective required to obtain a (\mathbf{c}, ϵ) -Forster transform, for a tolerance $\epsilon > 0$. By using the leverage score characterization (4) of exact Forster transforms, and performing a local perturbation analysis at the optimizer, we show $\text{poly}(\epsilon, \min_{i \in [n]} \mathbf{c}_i)$ error suffices. This significantly sharpens prior error bounds from [HM13, Che24], which scaled exponentially in a polynomial of the problem parameters.

With these stability bounds in place, the rest of Section 3 makes small modifications to the [CMTV17] analysis. We show that by using fast matrix multiplication, each Hessian can be computed in $\approx n^2 d^{\omega-2}$ time (Lemma 5), and that box-constrained Newton steps can be efficiently implemented using the constrained optimization methods from [CPW21]. This gives our basic runtime in Remark 1, which is sped up via sparsifiers provided by Theorem 2.

Implicit sparsification. We next describe our approach to proving Theorem 2, a reduction from $n^{o(1)}$ -approximate sparsification to $n^{o(1)}$ accesses of an implicit Laplacian via matrix-vector products. Our algorithm is an adaptation and extension of previous work [JLM⁺23], which gave an algorithm for recovering a $(1 + \epsilon)$ -spectral sparsifier of a graph G on n vertices, using $O(\text{polylog}(n))$ matrix-vector products with G ’s (pseudo)inverse Laplacian matrix, and $\approx n^2$ extra time.

The approach taken by [JLM⁺23] was to reduce the problem to a more general setting known as *matrix dictionary recovery*. Here, we are given matrix-vector access to \mathbf{B} (e.g., an implicit Laplacian), and a dictionary of $\{\mathbf{A}_i\}_{i \in [m]} \in \mathbb{S}_{\succeq 0}^{n \times n}$ with the promise that there exist weights $\mathbf{w}^* \in \mathbb{R}_{\geq 0}^n$ such that $\sum_{i \in [m]} \mathbf{w}_i^* \mathbf{A}_i = \mathbf{B}$. Our goal is to compute weights $\mathbf{w} \in \mathbb{R}_{\geq 0}^m$ such that $\mathbf{B} \preceq \sum_{i \in [m]} \mathbf{w}_i \mathbf{A}_i \preceq C\mathbf{B}$ for some approximation factor $C > 1$. This was done in [JLM⁺23] by reducing the two-sided recovery problem to a small number of one-sided *packing* semidefinite programs (SDPs):

$$\min_{\substack{\mathbf{x} \geq 0 \\ \sum_{i \in [m]} \mathbf{x}_i \mathbf{A}_i \preceq \mathbf{B}}} \mathbf{c}^\top \mathbf{x}. \quad (8)$$

Then, [JLM⁺23] applies packing SDP solvers from the literature [ZLO16, PTZ16, JLT20] to solve these problems. Combining this with a homotopy scheme yields their full algorithm.

A key limitation of this approach in the graph setting is simply the size of the matrix dictionary: in particular, we have one entry per candidate edge. For graphs on n vertices, maintaining an internal representation requires manipulating potentially-dense graphs on $m \approx n^2$ edges. This appears hard to bypass, without a priori information on which edges exist in our implicit Laplacian.

We circumvent this issue by enforcing that our intermediate Laplacians, while remaining dense combinations of our matrix dictionary, have a greater deal of structure that enables key primitives such as *sparsification* and *matrix-vector multiplication* in $n^{1+o(1)}$ time. We introduce a family of such compatible graphs, with a representation property we call *sum-of-cliques* (Definition 4). The key challenge is designing solvers for packing SDPs (8) that take few iterations, while maintaining that the solver's iterates are compatible with our sum-of-cliques machinery.

We make a crucial observation that standard matrix multiplicative weights (MMW)-based solvers for packing SDPs require gradient computations of the form

$$\langle \mathbf{M}, \mathbf{L}_e \rangle = \left\| \mathbf{M}^{\frac{1}{2}} (\mathbf{e}_u - \mathbf{e}_v) \right\|_2^2 \text{ for all } e = (u, v) \in [n] \times [n], \quad (9)$$

where $\mathbf{L}_e := (\mathbf{e}_u - \mathbf{e}_v)(\mathbf{e}_u - \mathbf{e}_v)^\top$ is an edge Laplacian, and \mathbf{M} is a response matrix given by the MMW updates. These gradients are then used to update a current iterate.

By noticing that gradient entries (9) can be viewed as distance computations between columns of $\mathbf{M}^{\frac{1}{2}}$, we apply metric embedding and hashing tools to coarsely discretize our gradients in a way that induces appropriate clique structures. In Definition 7, we further isolate sufficient conditions for this coarse discretization scheme to implement a long-step packing SDP solver, while maintaining that iterates are unions of $n^{o(1)}$ sums-of-cliques. We combine these pieces in Section 4.3 to give our main algorithmic innovation: an $n^{o(1)}$ -approximate solver for (8) that returns a Laplacian sparsifiable in $n^{1+o(1)}$ time, when the constraint \mathbf{B} is an unknown graph Laplacian.

We finally show in the rest of Section 4 that the remaining pieces of [JLM⁺23], i.e., the two-sided to one-sided reduction and the homotopy method, are compatible with our new packing solver.

Smoothed analysis of conditioning. To prove Theorem 3, our main result in the smoothed analysis setting, in Definition 9 we first extend an approach of [AKS20] that defines a notion of *deepness* of marginal vectors \mathbf{c} inside the basis polytope induced by \mathbf{A} 's independent row subsets. As we recall in Section 5.1, [AKS20] argues that if \mathbf{c} has deepness of $\eta = \Omega(1)$, then we can obtain a conditioning bound of $\log(\kappa) \approx d$ in Assumption 1. In the case of $\mathbf{c} = \frac{d}{n} \mathbf{1}_n$, this roughly translates to a robust variant of (3) that says: for all subspaces $E \subseteq \mathbb{R}^d$ of dimension k , at most a $\approx \frac{k}{d}$ fraction of \mathbf{A} 's rows (after smoothing by Gaussian noise) should lie at distance $\text{poly}(\frac{1}{n})$ from E . The rest of Section 5 proves this deepness result for smoothed matrices $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{G}$.

The key challenge is to avoid union bounding over a net of all possible subspaces E ; for $\dim(E) = \Theta(d)$, this naïve approach would require taking $n \gtrsim d^2$ samples, as nets of $\Theta(d)$ -dimensional subspaces have cardinality $\approx \exp(d^2)$. We instead show in Lemma 33 that deepness is implied by submatrices of $\tilde{\mathbf{A}}$ of appropriate size (dictated by the subspace dimension k) having at least $k+1$ large singular values, allowing us to apply union bounds to a smaller number of *data-dependent* subspaces. We combine this observation with singular value estimates from the random matrix theory literature to prove Theorem 3. Our argument requires some casework on the subspace dimension; we handle wide and near-square submatrices in Section 5.2 and tall submatrices in Section 5.3.

1.3 Related work

Forster transforms via maximum entropy. An alternative characterization of Forster transforms was followed by [SV19], who studied certain *maximum-entropy distribution* representations of specified marginals \mathbf{c} with respect to an index set \mathcal{S} , which we briefly explain for context. In our setting of finding a \mathbf{c} -Forster transform of $\mathbf{A} \in \mathbb{R}^{n \times d}$, the index set \mathcal{S} consists of all $S \subseteq [n]$ with $|S| = d$, and the underlying $\pi(S)$ is the *determinantal measure* with $\pi(S) \propto \det([\mathbf{A}^\top \mathbf{A}]_{S:S})$. Then, Section 8.3 of [SV19] applies the Cauchy-Binet formula to show that

$$\begin{aligned} \min_{\mathbf{t} \in \mathbb{R}^n} -\langle \mathbf{c}, \mathbf{t} \rangle + \log \det \left(\sum_{i \in [n]} \exp(\mathbf{t}_i) \mathbf{a}_i \mathbf{a}_i^\top \right) &= \min_{\mathbf{t} \in \mathbb{R}^n} \log \left(\sum_{S \in \mathcal{S}} \exp(\langle \mathbf{1}_S - \mathbf{c}, \mathbf{t} \rangle) \det([\mathbf{A}^\top \mathbf{A}]_{S:S}) \right) \\ &= \min_{\mathbf{t} \in \mathbb{R}^n} \log \left(\sum_{S \in \mathcal{S}} \pi(S) \exp(\langle \mathbf{1}_S - \mathbf{c}, \mathbf{t} \rangle) \right) + Z, \\ \text{where } Z &:= \log \left(\sum_{S \in \mathcal{S}} \det([\mathbf{A}^\top \mathbf{A}]_{S:S}) \right), \end{aligned} \tag{10}$$

and the starting expression is Barthe's objective (5). This shows that computing Forster transforms falls within the framework of Section 7 in [SV19], which exactly gives polynomial-time algorithms for optimizing functions in the form of the ending expression above. The runtime of [SV19] is not explicit, and we believe it is significantly slower than more direct approaches, e.g., CPMs [HM13]. Similarly, [BLNW20] develop interior-point methods for solving maximum entropy optimization problems of the form (10), but with runtimes scaling as $\text{poly}(|\mathcal{S}|)$, which in our setting is $\approx n^d$.

More closely related to our work, [CKV20] also consider a box-constrained Newton's method for solving maximum-entropy distributions over the hypercube $\{0, 1\}^n$ (which generalizes Barthe's objective, due to (10)). They prove that in this setting, maximum-entropy optimization problems (10) are $(r, O(rn))$ -Hessian stable for all $r > 0$ with respect to $\|\cdot\|_\infty$, which is a factor n worse than Proposition 4. Moreover, they do not provide an explicit runtime for solving box-constrained subproblems (relying on black-box quadratic programming solvers), and do not analyze the required accuracy for termination (e.g., Lemma 3). We believe their approach, as implemented for Barthe's objective, results in a much slower algorithm than CPMs; indeed, their stated runtime in a preprint [CKYV19] scales at least as $\approx n^{4.5}$. To obtain our runtime improvements, we require several novel structural insights on Barthe's objective and its interaction with optimization methods, as well as new algorithmic primitives (e.g., Theorem 2) capable of harnessing said structure.

Forster transforms via operator scaling. In another direction, [GGdOW17] discovered a non-trivial connection between computing Forster transforms (phrased in an equivalent way of computing Brascamp-Lieb constants, see Proposition 1.8, [GGdOW17]) and a related problem called *operator scaling*. This implies that polynomial-time algorithms for operator scaling [GGdOW16, GGdOW17, IQS17, AGL⁺18] apply to our problem as well. However, none of the aforementioned algorithms for operator scaling have an explicitly specified polynomial, and a crude analysis results in fairly substantial blowups. Also, some of these algorithms have more explicit and stronger variants analyzed in [DTK23, DR24], so we believe they are subsumed by our existing discussion.

More generally, there is an active body of research on generalizations of operator scaling and Forster

transforms [GGdOW17, Fra18, BFG⁺18, BFG⁺19], for which several state-of-the-art results are via variants of Newton’s method, broadly defined. It would be interesting to explore if the ideas developed in this paper could extend to those settings as well.

Reductions between graph primitives. Our work on implicit sparsification (Theorem 2) fits into a line of work that aims to characterize which fast (i.e., $n^{1+o(1)}$ -time) graph primitives imply others by reduction. This theme was explicitly considered by [ACSS20] (see also related work by [Qua21]), who studied these primitives for graphs implicitly defined by low-dimensional kernels. Among the three primitives of (1) fast matrix-vector multiplication, (2) fast spectral sparsification, and (3) fast Laplacian system solving, it was known previously that (3) reduces to (1) and (2) [ST04], and that (1) reduces to (2) and (3) [ACSS20]. Our work makes progress on this reduction landscape, as it shows (2) reduces to (1) (and hence, (3) also reduces to (1)). We mention that Theorem 5 in [JLM⁺23] gives a related, but slower, $\approx n^2$ -time reduction from (2) to (3).

Our work is also thematically connected to prior work on spectral sparsification under weak graph access, e.g., in streaming and dynamic settings [KLM⁺17, ADK⁺16]. Specifically, several of the rounding-via-sketching tools used to prove Theorem 2 are inspired by [KMM⁺20]. Their result is incomparable to ours, as we are unable to directly access a sketch of the graph, so we instead use these tools to speed up an optimization method rather than identify the sparsifier in one shot.

2 Preliminaries

In Section 2.1, we give notation used throughout the paper, as well as several key linear algebraic definitions. In Section 2.2, we provide preliminaries on the radial isotropic position scaling problem that we study, as well as Barthe’s objective [Bar98] used in computing Forster transforms.

2.1 Notation

General notation. We denote matrices in capital boldface and vectors in lowercase boldface throughout. By default, vectors are $d \times 1$ matrices. We let $\mathbf{0}_d$ and $\mathbf{1}_d$ denote the all-zeroes and all-ones vectors in \mathbb{R}^d , and $\mathbf{0}_{m \times n}$ is the all-zeroes $m \times n$ matrix. If $S \subseteq [d]$ and a dimension d is clear from context, we let $\mathbf{1}_S$ be the 0-1 indicator vector of S . We let $\mathbf{v} \circ \mathbf{w}$ denote the entrywise product of vectors \mathbf{v}, \mathbf{w} of the same dimension. For $n \in \mathbb{N}$ we define $[n] := \{i \in \mathbb{N} \mid i \leq n\}$. For $p \geq 1$ and $p = \infty$ we let $\|\cdot\|_p$ denote the ℓ_p norm of a vector argument, as well as the Schatten- p norm of a matrix argument. For $\bar{\mathbf{x}} \in \mathbb{R}^d$ and $r > 0$, we let $\mathbb{B}_p(\bar{\mathbf{x}}, r) := \{\mathbf{x} \in \mathbb{R}^d \mid \|\bar{\mathbf{x}} - \mathbf{x}\|_p \leq r\}$ denote the ℓ_p norm ball of radius r centered at $\bar{\mathbf{x}}$; if $\bar{\mathbf{x}}$ is unspecified then $\bar{\mathbf{x}} = \mathbf{0}_d$ by default. We let $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denote the multivariate Gaussian with specified mean and covariance. When d is clear from context, $\mathbf{e}_i \in \mathbb{R}^d$ for $i \in [d]$ denotes the i^{th} standard basis vector. For $S \subset \mathbb{R}^d$, we use $\text{conv}(S)$ to denote the convex hull of S . For an event \mathcal{E} , we let $\mathbb{1}_{\mathcal{E}}$ denote the 0-1 indicator variable.

For a multilinear form $\mathbf{T} \in \mathbb{R}^{d_1 \times \dots \times d_\ell}$ and vectors $\mathbf{v}_i \in \mathbb{R}^{d_i}$ for all $i \in [\ell]$, we denote

$$\mathbf{T}[\mathbf{v}_1, \dots, \mathbf{v}_\ell] := \sum_{\substack{i_1 \in [d_1] \\ \vdots \\ i_\ell \in [d_\ell]}} \mathbf{T}_{i_1 \dots i_\ell} [\mathbf{v}_1]_{i_1} \dots [\mathbf{v}_\ell]_{i_\ell}.$$

For example, when \mathbf{T} is an $n \times d$ matrix, $\mathbf{T}[\mathbf{u}, \mathbf{v}] = \sum_{(i,j) \in [n] \times [d]} \mathbf{T}_{ij} \mathbf{u}_i \mathbf{v}_j = \mathbf{u}^\top \mathbf{T} \mathbf{v}$.

Matrices. The $d \times d$ identity matrix is denoted \mathbf{I}_d . The span of a set of vectors $\{\mathbf{a}_i\}_{i \in [n]}$ is denoted $\text{Span}(\{\mathbf{a}_i\}_{i \in [n]})$; when \mathbf{A} is a matrix, we overload $\text{Span}(\mathbf{A})$ to mean the span of its columns. We similarly use $\text{rank}(\{\mathbf{a}_i\}_{i \in [n]})$, $\text{rank}(\mathbf{A})$ to denote the dimension of the aforementioned subspaces.

We denote the j^{th} column of $\mathbf{A} \in \mathbb{R}^{n \times d}$ by $\mathbf{A}_{\cdot j} \in \mathbb{R}^n$ for all $j \in [d]$, and we similarly denote the i^{th} row of \mathbf{A} (viewed as a column vector) by $\mathbf{A}_{i \cdot} \in \mathbb{R}^d$ for all $i \in [n]$. For row and column subsets $S \subseteq [n]$, $T \subseteq [d]$, the corresponding submatrix is denoted $\mathbf{A}_{S:T}$; $S = [n]$ and $T = [d]$ by default if excluded. We denote the Frobenius and $(2 \rightarrow 2)$ operator norms of a matrix argument by $\|\cdot\|_{\text{F}}$ and $\|\cdot\|_{\text{op}}$. For a vector $\mathbf{v} \in \mathbb{R}^d$ we let $\text{diag}(\mathbf{v})$ be the $d \times d$ diagonal matrix with \mathbf{v} along the diagonal. We use $\text{nnz}(\mathbf{M})$ to denote the number of nonzero entries of a matrix \mathbf{M} , and we use $\mathcal{T}_{\text{mv}}(\mathbf{M})$ to mean the time required to compute $\mathbf{M}\mathbf{v}$ for an arbitrary vector \mathbf{v} of appropriate dimension. We denote the projection matrix onto a subspace $E \subseteq \mathbb{R}^d$ by $\mathbf{\Pi}_E \in \mathbb{S}_{\geq 0}^{d \times d}$.

We denote the set of symmetric $d \times d$ matrices by $\mathbb{S}^{d \times d}$, and the positive semidefinite (PSD) cone by $\mathbb{S}_{\geq 0}^{d \times d} \subset \mathbb{S}^{d \times d}$. We equip $\mathbb{S}_{\geq 0}^{d \times d}$ with the Loewner partial ordering \preceq . We define the induced seminorm of $\mathbf{M} \in \mathbb{S}_{\geq 0}^{d \times d}$ by $\|\mathbf{v}\|_{\mathbf{M}}^2 := \mathbf{v}^\top \mathbf{M} \mathbf{v}$. For $\mathbf{M} \in \mathbb{S}^{d \times d}$ we let \mathbf{M}^\dagger denote its pseudoinverse, which satisfies $\mathbf{M}\mathbf{M}^\dagger = \mathbf{M}^\dagger\mathbf{M}$ is the projection matrix onto $\text{Span}(\mathbf{M})$. The eigenvalues of $\mathbf{M} \in \mathbb{S}^{d \times d}$ are denoted $\boldsymbol{\lambda}(\mathbf{M}) \in \mathbb{R}^d$, where our convention is to order $\boldsymbol{\lambda}_1(\mathbf{M}) \geq \boldsymbol{\lambda}_2(\mathbf{M}) \geq \dots \geq \boldsymbol{\lambda}_d(\mathbf{M})$. We similarly denote the (monotone nonincreasing) singular values of $\mathbf{A} \in \mathbb{R}^{n \times d}$ by $\boldsymbol{\sigma}(\mathbf{A}) \in \mathbb{R}^{\min(n,d)}$.

The trace of $\mathbf{M} \in \mathbb{S}^{d \times d}$ is denoted $\text{Tr}(\mathbf{M})$. For $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times d}$, we define the matrix inner product by $\langle \mathbf{A}, \mathbf{B} \rangle := \text{Tr}(\mathbf{A}^\top \mathbf{B}) = \sum_{(i,j) \in [n] \times [d]} \mathbf{A}_{ij} \mathbf{B}_{ij}$. We use the following notion of multiplicative approximation between PSD matrices: for $\mathbf{A}, \mathbf{B} \in \mathbb{S}_{\geq 0}^{d \times d}$ and $\epsilon > 0$, we write

$$\mathbf{A} \approx_\epsilon \mathbf{B} \iff \exp(-\epsilon) \mathbf{B} \preceq \mathbf{A} \preceq \exp(\epsilon) \mathbf{B}. \quad (11)$$

For nonnegative scalars $a, b \in \mathbb{R}_{\geq 0}$ we write $a \approx_\epsilon b$ to be the 1-d specialization of (11), and $\mathbf{u} \approx_\epsilon \mathbf{v}$ is an entrywise definition for nonnegative vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}_{\geq 0}^d$.

We define the *leverage scores* of a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ with rows $\{\mathbf{a}_i^\top\}_{i \in [n]}$ by

$$\tau_i(\mathbf{A}) := \mathbf{a}_i^\top (\mathbf{A}^\top \mathbf{A})^\dagger \mathbf{a}_i. \quad (12)$$

Leverage scores are a common measure of the relative importance of rows for preserving spectral information in numerical linear algebra, and are a crucial concept used throughout this paper. We summarize several basic facts about leverage scores here, see e.g., [CLM⁺15] for proofs.

Fact 1. For all $\mathbf{A} \in \mathbb{R}^{n \times d}$, we have $\boldsymbol{\tau}(\mathbf{A}) \in [0, 1]^n$, and $\sum_{i \in [n]} \tau_i(\mathbf{A}) = \text{rank}(\mathbf{A})$.

Finally, we let $\omega < 2.372$ denote the exponent of the square matrix multiplication runtime [ADV⁺25].

Optimization. For k -times differentiable $f : \mathbb{R}^n \rightarrow \mathbb{R}$, we let $\nabla^k f$ denote the k^{th} derivative tensor of f , e.g., ∇f and $\nabla^2 f$ are the gradient and Hessian of f . We use the following notion of multiplicative stability for analyzing Newton's method, patterned off [CMTV17, KSJ18, CJJ⁺20].

Definition 2 (Hessian stability). We say that twice-differentiable $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is (r, ϵ) -Hessian stable with respect to norm $\|\cdot\|$ if for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ with $\|\mathbf{x} - \mathbf{y}\| \leq r$, we have following (11) that

$$\nabla^2 f(\mathbf{x}) \approx_\epsilon \nabla^2 f(\mathbf{y}).$$

We specifically will use the $\|\cdot\|_\infty$ case of Definition 2 to design our algorithms, via toolkits provided by [CMTV17], who called this property “second-order robustness,” and [CPW21].

2.2 Radial isotropic position

Our main contribution is an algorithm to scale a feasible matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ to approximately satisfy a strong linear algebraic condition known as *radial isotropic position* (Definition 1). This condition was introduced to the theoretical computer science community by [Bar98] to study inequalities in functional analysis, but a similar definition arose earlier in algebraic geometry [GGMS87].

To briefly demystify Definition 1, let $\mathbf{A} \in \mathbb{R}^{n \times d}$ with rows $\{\mathbf{a}_i^\top\}_{i \in [n]}$ have $\text{rank}(\mathbf{A}) = d$ (so $n \geq d$). Then, it is well-known that \mathbf{A} can be scaled by invertible $\mathbf{R} \in \mathbb{R}^{d \times d}$ so that

$$\mathbf{R}\mathbf{A}^\top \mathbf{diag}(\mathbf{c}) \mathbf{A}\mathbf{R}^\top = \sum_{i \in [n]} \mathbf{c}_i (\mathbf{R}\mathbf{a}_i) (\mathbf{R}\mathbf{a}_i)^\top = \mathbf{I}_d. \quad (13)$$

Indeed, choosing $\mathbf{R} = (\mathbf{A}^\top \mathbf{diag}(\mathbf{c}) \mathbf{A})^{-1/2}$ suffices. The condition (13) is sometimes referred to as being scaled to be in \mathbf{c} -isotropic position, and there are natural ϵ -approximate generalizations.

Similarly, as long as all $\mathbf{a}_i \neq \mathbf{0}_d$, there is a diagonal scaling \mathbf{S} so that $\mathbf{S}\mathbf{A}$ has unit-norm rows: let

$$\mathbf{S} = \mathbf{diag}(\mathbf{s}) \text{ where } \mathbf{s}_i = \frac{1}{\|\mathbf{a}_i\|_2} \text{ for all } i \in [n] \implies \|[\mathbf{S}\mathbf{A}]_{i:}\|_2 = 1 \text{ for all } i \in [n]. \quad (14)$$

Each of the transformations (13) and (14) is used in many applications to improve the regularity of a point set given by viewing the rows of \mathbf{A} as points in \mathbb{R}^d . The purpose of \mathbf{c} -radial isotropic position (Definition 1) is to give a Forster transform matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$ inducing a scaling $\mathbf{S} = \mathbf{diag}(\mathbf{s})$ via $\mathbf{s}_i^{-1} = \|\mathbf{R}\mathbf{a}_i\|_2$ for all $i \in [n]$, such that the left-and-right scaled matrix $\mathbf{S}\mathbf{A}\mathbf{R}^\top$ simultaneously has unit-norm rows, and is in \mathbf{c} -isotropic position. Our goal is to efficiently approximate \mathbf{R} .

Existence of Forster transform. Not all point sets admit a \mathbf{c} -Forster transform. A sequence of works [GGMS87, Bar98, CLL04, DSW14, HKLM20] gave two useful characterizations of feasibility.

Proposition 1 (Lemma 4.19, [HKLM20]). *Given a point set $A := \{\mathbf{a}_i\}_{i \in [n]} \subset \mathbb{R}^d$ and $\mathbf{c} \in (0, 1]^n$ satisfying $\|\mathbf{c}\|_1 = d$, the following conditions are equivalent where $\mathbf{A} \in \mathbb{R}^{n \times d}$ has rows A .*

1. For any $\epsilon > 0$ there exists $\mathbf{R} \in \mathbb{R}^{d \times d}$, a (\mathbf{c}, ϵ) -Forster transform of \mathbf{A} .
2. For every $k \in [d]$, every k -dimensional linear subspace $V \subseteq \mathbb{R}^d$ satisfies

$$\sum_{\substack{i \in [n] \\ \mathbf{a}_i \in V}} \mathbf{c}_i \leq k. \quad (15)$$

One direction of Proposition 1 is straightforward: if a k -dimensional subspace V is too “heavy” (i.e., (15) is violated) then there still exists a heavy subspace under any transform \mathbf{R} . Taking the trace of both sides of the definition (2) restricted to this heavy subspace yields a contradiction for sufficiently small ϵ . In the case of $\mathbf{c} = \frac{d}{n}\mathbf{1}_n$, (15) simply translates to no k -dimensional subspace containing more than $\frac{k}{d} \cdot n$ of the points. One simple way for this condition to hold is if A is in *general position*. Note that by taking $V = \text{Span}(A)$, (15) implies that $n \geq d$ and \mathbf{A} has full rank.

The other direction is significantly more challenging, and [HKLM20] gives an iterative construction based on decompositions with respect to the *basis polytope* of $A = \{\mathbf{a}_i\}_{i \in [n]}$. This is the central object in the next characterization we will use, so we define it here.

Definition 3 (Basis polytope). *Consider a point set $A = \{\mathbf{a}_i\}_{i \in [n]}$. Let $\mathcal{B} \subseteq 2^{[n]}$ be the set of subsets $B \subseteq [n]$ such that $\{\mathbf{a}_i\}_{i \in B}$ is a basis of \mathbb{R}^d , i.e., it is a linearly-independent set that spans \mathbb{R}^d . Letting $\mathbf{1}_B \in \{0, 1\}^n$ denote the 0-1 indicator vector of each $B \in \mathcal{B}$, we let*

$$\mathcal{P}(A) := \text{conv}(\{\mathbf{1}_B\}_{B \in \mathcal{B}})$$

denote the basis polytope corresponding to the independent set matroid induced by A .

Another convenient reformulation of the necessary and sufficient condition in Proposition 1 was given by [CLL04]; see also [Bar98, DSW14, HKLM20] for interpretations of this condition.

Proposition 2 (Theorem 4.4, [CLL04]). *The conditions in Proposition 1 hold iff $\mathbf{c} \in \mathcal{P}(A)$.*

Scaling via leverage scores. There is a primal-dual viewpoint of Definition 1, as described in our derivation (13), (14). In particular, \mathbf{c} -RIP can equivalently be viewed as being induced by a pair (\mathbf{R}, \mathbf{s}) , where the diagonal scaling \mathbf{s} is an implicit function of the Forster transform \mathbf{R} .

We may ask if this correspondence goes the other direction; are there conditions on $\mathbf{s} \in \mathbb{R}_{>0}^n$ such that one can deduce $\mathbf{R} \in \mathbb{R}^{d \times d}$ that scales \mathbf{A} to be in \mathbf{c} -RIP? The following observation, patterned from [DR24], shows the answer is yes: it is enough for $\tau(\mathbf{S}\mathbf{A}) = \mathbf{c}$, where $\mathbf{S} := \text{diag}(\mathbf{s})$.

Lemma 1. *Given a point set $A := \{\mathbf{a}_i\}_{i \in [n]} \subset \mathbb{R}^d$ and $\mathbf{c} \in (0, 1]^n$ satisfying $\|\mathbf{c}\|_1 = d$, suppose (15) holds. Then letting $\mathbf{A} \in \mathbb{R}^{n \times d}$ have rows A , if some $\mathbf{s} \in \mathbb{R}_{>0}^n$ satisfies $\tau(\mathbf{S}\mathbf{A}) = \mathbf{c}$, where $\mathbf{S} := \text{diag}(\mathbf{s})$, then*

$$\sum_{i \in [n]} \mathbf{c}_i \cdot \frac{(\mathbf{R}\mathbf{a}_i)(\mathbf{R}\mathbf{a}_i)^\top}{\|\mathbf{R}\mathbf{a}_i\|_2^2} = \mathbf{I}_d \text{ for } \mathbf{R} := (\mathbf{A}^\top \mathbf{S}^2 \mathbf{A})^{-\frac{1}{2}}. \quad (16)$$

More generally, for any $\epsilon > 0$, if $\tau(\mathbf{S}\mathbf{A}) \approx_\epsilon \mathbf{c}$, then

$$\sum_{i \in [n]} \mathbf{c}_i \cdot \frac{(\mathbf{R}\mathbf{a}_i)(\mathbf{R}\mathbf{a}_i)^\top}{\|\mathbf{R}\mathbf{a}_i\|_2^2} \approx_\epsilon \mathbf{I}_d \text{ for } \mathbf{R} := (\mathbf{A}^\top \mathbf{S}^2 \mathbf{A})^{-\frac{1}{2}}. \quad (17)$$

Proof. We prove (17), which implies (16) by taking $\epsilon \rightarrow 0$. Indeed, by using $\tau(\mathbf{S}\mathbf{A}) \approx_\epsilon \mathbf{c}$,

$$\tau_i(\mathbf{S}\mathbf{A}) = \mathbf{s}_i^2 \|\mathbf{R}\mathbf{a}_i\|_2^2 \approx_\epsilon \mathbf{c}_i \implies \frac{\mathbf{c}_i}{\|\mathbf{R}\mathbf{a}_i\|_2^2} \approx_\epsilon \mathbf{s}_i^2 \text{ for all } i \in [n].$$

This directly implies that (17) holds:

$$\sum_{i \in [n]} \mathbf{c}_i \cdot \frac{(\mathbf{R}\mathbf{a}_i)(\mathbf{R}\mathbf{a}_i)^\top}{\|\mathbf{R}\mathbf{a}_i\|_2^2} \approx_\epsilon \sum_{i \in [n]} \mathbf{s}_i^2 (\mathbf{R}\mathbf{a}_i)(\mathbf{R}\mathbf{a}_i)^\top = \mathbf{R} \left(\sum_{i \in [n]} \mathbf{s}_i^2 \mathbf{a}_i \mathbf{a}_i^\top \right) \mathbf{R} = \mathbf{I}_d.$$

□

Barthe's objective. In the rest of the paper, we fix a point set $A := \{\mathbf{a}_i\}_{i \in [n]} \subset \mathbb{R}^d$, that forms the rows of $\mathbf{A} \in \mathbb{R}^{n \times d}$. We also let $\mathbf{c} \in (0, 1]^n$ satisfy $\|\mathbf{c}\|_1 = d$, such that the condition (15) holds. For some $\epsilon \in (0, 1)$, we will give an algorithm for computing a (\mathbf{c}, ϵ) -Forster transform of \mathbf{A} .

To ease our exposition we fix the following notation throughout, for $\mathbf{t} \in \mathbb{R}^n$:

$$\begin{aligned}\mathbf{Z}(\mathbf{t}) &:= \mathbf{A}^\top \mathbf{diag}(\exp(\mathbf{t})) \mathbf{A} = \sum_{i \in [n]} \exp(\mathbf{t}_i) \mathbf{a}_i \mathbf{a}_i^\top, \\ \mathbf{R}(\mathbf{t}) &:= \mathbf{Z}(\mathbf{t})^{-\frac{1}{2}} = \left(\mathbf{A}^\top \mathbf{diag}(\exp(\mathbf{t})) \mathbf{A} \right)^{-\frac{1}{2}}, \\ \mathbf{S}(\mathbf{t}) &:= \mathbf{diag}(\mathbf{s}(\mathbf{t})), \text{ where } \mathbf{s}(\mathbf{t}) := \exp\left(\frac{\mathbf{t}}{2}\right), \\ \tilde{\mathbf{a}}_i(\mathbf{t}) &:= \mathbf{R}(\mathbf{t}) \mathbf{a}_i, \text{ for all } i \in [n].\end{aligned}\tag{18}$$

In (18), we let \exp be applied to a vector argument entrywise. Note that all of the matrices and vectors in (18) correspond to those arising in our earlier discussion, after reparameterizing the problem by $\mathbf{t} = 2 \log(\mathbf{s})$ entrywise. This reparameterization becomes convenient shortly.

Our approach follows the seminal work [Bar98], which gave an algorithmic proof of Propositions 1, 2, by explicitly characterizing the scaling $\mathbf{s} \in \mathbb{R}_{>0}^n$ in Lemma 1 such that $\tau(\mathbf{S}\mathbf{A}) = \mathbf{c}$ for $\mathbf{S} := \mathbf{diag}(\mathbf{s})$, by way of a $\mathbf{t} \in \mathbb{R}^n$ that achieves $\mathbf{s} = \mathbf{s}(\mathbf{t})$. To explain, we first define Barthe's objective:

$$f(\mathbf{t}) := -\langle \mathbf{c}, \mathbf{t} \rangle + \log \det(\mathbf{Z}(\mathbf{t})).\tag{19}$$

Then Barthe's result can be stated as follows.

Proposition 3 (Proposition 6, [Bar98]). *Following notation (18), (19), $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a convex function, and its minimizer is attained iff \mathbf{A}, \mathbf{c} satisfy $\|\mathbf{c}\|_1 = d$ and the condition (15). Moreover, letting $\mathbf{t}^* := \arg \min_{\mathbf{t} \in \mathbb{R}^n} f(\mathbf{t})$, $\mathbf{R}(\mathbf{t}^*)$ is a \mathbf{c} -Forster transform of \mathbf{A} .*

Proposition 3 can be somewhat demystified by computing the derivatives of Barthe's objective. We introduce one additional piece of notation here:

$$\mathbf{M}_i(\mathbf{t}) := \mathbf{s}_i(\mathbf{t})^2 \tilde{\mathbf{a}}_i(\mathbf{t}) \tilde{\mathbf{a}}_i(\mathbf{t})^\top = \mathbf{R}(\mathbf{t}) \left(\exp(\mathbf{t}_i) \mathbf{a}_i \mathbf{a}_i^\top \right) \mathbf{R}(\mathbf{t}), \text{ for all } i \in [n].\tag{20}$$

Fact 2. *Following notation (18), (19), we have for all $(i, j) \in [n] \times [n]$ that*

$$\begin{aligned}\nabla_i f(\mathbf{t}) &= -\mathbf{c}_i + \text{Tr}(\mathbf{M}_i(\mathbf{t})), \\ \nabla_{ij}^2 f(\mathbf{t}) &= \text{Tr}(\mathbf{M}_i(\mathbf{t})) \mathbb{I}_{i=j} - \text{Tr}(\mathbf{M}_i(\mathbf{t}) \mathbf{M}_j(\mathbf{t})).\end{aligned}\tag{21}$$

From Fact 2 we can glean several different parts of Proposition 3. For example, the fact that $\sum_{i \in [n]} \mathbf{M}_i(\mathbf{t}) = \mathbf{R}(\mathbf{t}) \mathbf{Z}(\mathbf{t}) \mathbf{R}(\mathbf{t}) = \mathbf{I}_d$, combined with Kadison's inequality [Kad52] (see also Theorem 2.3.2, [Bha07]), shows that for all vectors $\mathbf{v} \in \mathbb{R}^d$,

$$\left(\sum_{i \in [n]} \mathbf{v}_i \mathbf{M}_i(\mathbf{t}) \right)^2 \preceq \sum_{i \in [n]} \mathbf{v}_i^2 \mathbf{M}_i(\mathbf{t}).$$

In particular, taking a trace of both sides above shows

$$\nabla^2 f(\mathbf{t})[\mathbf{v}, \mathbf{v}] = \text{Tr} \left(\sum_{i \in [n]} \mathbf{v}_i^2 \mathbf{M}_i(\mathbf{t}) \right) - \text{Tr} \left(\left(\sum_{i \in [n]} \mathbf{v}_i \mathbf{M}_i(\mathbf{t}) \right)^2 \right) \geq 0,$$

which implies that f is convex. Similarly, letting \mathbf{t}^* minimize f , we have from (21) that

$$\mathbf{c}_i = \text{Tr}(\mathbf{M}_i(\mathbf{t}^*)) = \boldsymbol{\tau}_i(\mathbf{S}(\mathbf{t}^*) \mathbf{A}) \text{ for all } i \in [n]. \quad (22)$$

Using the characterization of $\mathbf{S}(\mathbf{t}^*)$ in (22) as obtaining the leverage scores \mathbf{c} , and applying Lemma 1, we have shown that $\mathbf{R}(\mathbf{t}^*)$ is indeed a \mathbf{c} -Forster transform of \mathbf{A} , as stated in Proposition 3.

3 Optimizing Barthe's Objective via Newton's Method

In this section, we give our algorithm for computing approximate Forster transforms. Our algorithm is a variant of the box-constrained Newton's method of [CMTV17], which solves box-constrained quadratics to optimize a Hessian-stable function in $\|\cdot\|_\infty$ to high precision.

We first make our key technical observation in Section 3.1: that Barthe's objective is Hessian-stable with respect to $\|\cdot\|_\infty$. We then give a termination condition in Section 3.2 that suffices for $\mathbf{t} \in \mathbb{R}^n$ to induce an ϵ -Forster transform $\mathbf{R}(\mathbf{t})$. In Section 3.3, we leverage our implicit Laplacian sparsification algorithm from Section 4 to implement the iteration of the [CMTV17] Newton's method. We put all the pieces together in Section 3.4 to give our main result.

Throughout this section, we fix a pair $\mathbf{A} \in \mathbb{R}^{n \times d}$ and $\mathbf{c} \in (0, 1]^n$ satisfying $\|\mathbf{c}\|_1 = d$ and (15). We follow the notation outlined in Section 2.2, in particular, (18), (19), and (20). We also will state our results under the diameter bound in Assumption 1.

3.1 Hessian stability of Barthe's objective

In this section, we prove the following key structural result enabling our approach.

Proposition 4. *For all $r > 0$, f is $(r, 2r)$ -Hessian stable with respect to $\|\cdot\|_\infty$.*

A similar result to Proposition 4 was previously established for the softmax objective

$$\mathbf{t} \rightarrow \log \left(\sum_{i \in [n]} \exp(\mathbf{t}_i) \right),$$

in Lemma 14, [CJJ⁺20], using more elementary techniques, i.e., directly establishing that the softmax satisfies a third-order regularity property called *quasi-self-concordance*.

Due to complications arising from Barthe's objective being defined with respect to potentially non-commuting matrices, we follow a substantially different approach in this section. We need the following helper lemma, where we define the *Schur complement*

$$\text{SC}(\mathbf{M}, S) := \mathbf{M}_{S:S} - \mathbf{M}_{S:S^c} \mathbf{M}_{S^c:S}^\dagger \mathbf{M}_{S^c:S}, \quad (23)$$

for any square matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$, subset $S \subseteq [d]$, and $S^c := [d] \setminus S$.

Lemma 2. *If $\mathbf{M}, \mathbf{N} \in \mathbb{S}_{\geq 0}^{d \times d}$ and $\mathbf{M} \approx_{\epsilon} \mathbf{N}$, then*

$$\text{SC}(\mathbf{M}, S) \approx_{\epsilon} \text{SC}(\mathbf{N}, S) \text{ for all } S \subseteq [d].$$

Proof. It suffices to show that if $\mathbf{A}, \mathbf{B} \in \mathbb{S}_{\geq 0}^{d \times d}$,

$$\mathbf{A} \succeq \mathbf{B} \implies \text{SC}(\mathbf{A}, S) \succeq \text{SC}(\mathbf{B}, S). \quad (24)$$

The claim then follows by applying (24) with $(\mathbf{A}, \mathbf{B}) \leftarrow (\exp(\epsilon)\mathbf{M}, \mathbf{N})$ and $\leftarrow (\exp(\epsilon)\mathbf{N}, \mathbf{M})$, since $\text{SC}(\alpha\mathbf{M}, S) = \alpha\text{SC}(\mathbf{M}, S)$ for any scaling coefficient $\alpha \in \mathbb{R}$.

We now establish (24). It is well-known (see, e.g., Appendix A.5.5 of [BV04]) that

$$\mathbf{x}^T \text{SC}(\mathbf{A}, S) \mathbf{x} = \min_{\mathbf{y} \in \mathbb{R}^{S^c}} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}^T \mathbf{A} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$$

for all $S \subseteq [d]$, $\mathbf{x} \in \mathbb{R}^S$. Now (24) follows from

$$\mathbf{x}^T \text{SC}(\mathbf{A}, S) \mathbf{x} = \min_{\mathbf{y} \in \mathbb{R}^{S^c}} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}^T \mathbf{A} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \geq \min_{\mathbf{y} \in \mathbb{R}^{S^c}} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}^T \mathbf{B} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \mathbf{x}^T \text{SC}(\mathbf{B}, S) \mathbf{x}.$$

□

We are now ready to prove Proposition 4.

Proof of Proposition 4. Throughout, fix $\mathbf{t}, \mathbf{t}' \in \mathbb{R}^n$ with $\|\mathbf{t} - \mathbf{t}'\|_{\infty} \leq r$. Our goal is to show

$$\nabla^2 f(\mathbf{t}) \approx_{2r} \nabla^2 f(\mathbf{t}').$$

We follow the notation (18), (20), and whenever the argument is dropped, it is implied to be at \mathbf{t} ; we will use a superscript $'$ whenever the argument is at \mathbf{t}' . So, for example, $\mathbf{M}_i \equiv \mathbf{M}_i(\mathbf{t})$ and $\mathbf{M}'_i \equiv \mathbf{M}_i(\mathbf{t}')$ for all $i \in [n]$. Also, to ease notation in this proof we define

$$\mathbf{C}_i := \exp(\mathbf{t}_i) \mathbf{a}_i \mathbf{a}_i^T, \quad \mathbf{C}'_i := \exp(\mathbf{t}'_i) \mathbf{a}_i \mathbf{a}_i^T,$$

so that $\mathbf{M}_i = \mathbf{R} \mathbf{C}_i \mathbf{R}$ for all $i \in [n]$. We first claim that for all $i \in [n]$,

$$\begin{pmatrix} \mathbf{C}_i & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{C}_i & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_i & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{C}_i & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix} \approx_r \begin{pmatrix} \mathbf{C}'_i & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{C}'_i & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}'_i & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{C}'_i & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{pmatrix}, \quad (25)$$

where both matrices in (25) have dimensions $(n+1)d \times (n+1)d$, and only the $(1, 1)$, $(i+1, 1)$, $(1, i+1)$, and $(i+1, i+1)$ -indexed $d \times d$ blocks are nonzero. We can verify (25) by direct expansion with respect to a $2d$ -dimensional test vector with blocks \mathbf{x}, \mathbf{y} , which reduces the claim to

$$(\mathbf{x} + \mathbf{y})^T \mathbf{C}_i (\mathbf{x} + \mathbf{y}) \approx_r (\mathbf{x} + \mathbf{y})^T \mathbf{C}'_i (\mathbf{x} + \mathbf{y}) \iff \mathbf{C}_i \approx_r \mathbf{C}'_i,$$

where the latter fact above follows from $\|\mathbf{t} - \mathbf{t}'\|_\infty \leq r$. Summing (25) for all $i \in [n]$ shows

$$\mathbf{L} \approx_r \mathbf{L}', \text{ where } \mathbf{L} := \begin{pmatrix} \sum_{i \in [n]} \mathbf{C}_i & \mathbf{C}_1 & \mathbf{C}_2 & \cdots & \mathbf{C}_n \\ \mathbf{C}_1 & \mathbf{C}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}_2 & \mathbf{0} & \mathbf{C}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_n & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}_n \end{pmatrix},$$

$$\mathbf{L}' := \begin{pmatrix} \sum_{i \in [n]} \mathbf{C}'_i & \mathbf{C}'_1 & \mathbf{C}'_2 & \cdots & \mathbf{C}'_n \\ \mathbf{C}'_1 & \mathbf{C}'_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}'_2 & \mathbf{0} & \mathbf{C}'_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}'_n & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}'_n \end{pmatrix}.$$

Next, using Lemma 2 to take Schur complements of \mathbf{L}, \mathbf{L}' onto the index set $[(n+1)d] \setminus [d]$ shows

$$\mathbf{K} \approx_r \mathbf{K}', \text{ where } \mathbf{K} := \begin{pmatrix} \mathbf{C}_1 - \mathbf{C}_1 \mathbf{Z}^{-1} \mathbf{C}_1 & \cdots & -\mathbf{C}_1 \mathbf{Z}^{-1} \mathbf{C}_n \\ -\mathbf{C}_2 \mathbf{Z}^{-1} \mathbf{C}_1 & \cdots & -\mathbf{C}_2 \mathbf{Z}^{-1} \mathbf{C}_n \\ \vdots & \ddots & \vdots \\ -\mathbf{C}_n \mathbf{Z}^{-1} \mathbf{C}_1 & \cdots & \mathbf{C}_n - \mathbf{C}_n \mathbf{Z}^{-1} \mathbf{C}_n \end{pmatrix},$$

$$\mathbf{K}' := \begin{pmatrix} \mathbf{C}'_1 - \mathbf{C}'_1 (\mathbf{Z}')^{-1} \mathbf{C}'_1 & \cdots & -\mathbf{C}'_1 (\mathbf{Z}')^{-1} \mathbf{C}'_n \\ -\mathbf{C}'_2 (\mathbf{Z}')^{-1} \mathbf{C}'_1 & \cdots & -\mathbf{C}'_2 (\mathbf{Z}')^{-1} \mathbf{C}'_n \\ \vdots & \ddots & \vdots \\ -\mathbf{C}'_n (\mathbf{Z}')^{-1} \mathbf{C}'_1 & \cdots & \mathbf{C}'_n - \mathbf{C}'_n (\mathbf{Z}')^{-1} \mathbf{C}'_n \end{pmatrix},$$

where we used that $\mathbf{Z} = \sum_{i \in [n]} \mathbf{C}_i$ and $\mathbf{Z}' = \sum_{i \in [n]} \mathbf{C}'_i$. Finally, fix some vector $\mathbf{v} \in \mathbb{R}^n$. Let

$$\mathbf{J} := \mathbf{v} \mathbf{v}^\top \otimes \mathbf{Z}^{-1} = \begin{pmatrix} \mathbf{v}_1^2 \mathbf{Z}^{-1} & \mathbf{v}_1 \mathbf{v}_2 \mathbf{Z}^{-1} & \mathbf{v}_1 \mathbf{v}_3 \mathbf{Z}^{-1} & \cdots & \mathbf{v}_1 \mathbf{v}_n \mathbf{Z}^{-1} \\ \mathbf{v}_1 \mathbf{v}_2 \mathbf{Z}^{-1} & \mathbf{v}_2^2 \mathbf{Z}^{-1} & \mathbf{v}_2 \mathbf{v}_3 \mathbf{Z}^{-1} & \cdots & \mathbf{v}_2 \mathbf{v}_n \mathbf{Z}^{-1} \\ \mathbf{v}_1 \mathbf{v}_3 \mathbf{Z}^{-1} & \mathbf{v}_2 \mathbf{v}_3 \mathbf{Z}^{-1} & \mathbf{v}_3^2 \mathbf{Z}^{-1} & \cdots & \mathbf{v}_3 \mathbf{v}_n \mathbf{Z}^{-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}_1 \mathbf{v}_n \mathbf{Z}^{-1} & \mathbf{v}_2 \mathbf{v}_n \mathbf{Z}^{-1} & \mathbf{v}_3 \mathbf{v}_n \mathbf{Z}^{-1} & \cdots & \mathbf{v}_n^2 \mathbf{Z}^{-1} \end{pmatrix},$$

where \otimes denotes the Kronecker product. Similarly define $\mathbf{J}' := \mathbf{v} \mathbf{v}^\top \otimes (\mathbf{Z}')^{-1}$. Because we established each $\mathbf{C}_i \approx_r \mathbf{C}'_i$, we also have $\mathbf{Z} \approx_r \mathbf{Z}'$ and thus $\mathbf{Z}^{-1} \approx_r (\mathbf{Z}')^{-1}$. By well-known properties of the Kronecker product (cf. Theorem 2.3, [Sch13]), we conclude that $\mathbf{J} \approx_r \mathbf{J}'$. We have thus shown:

$$\mathbf{K} \approx_r \mathbf{K}', \mathbf{J} \approx_r \mathbf{J}'.$$

Finally, it is standard that if $\mathbf{A} \preceq \mathbf{B}$ and $\mathbf{C} \preceq \mathbf{D}$ then $\langle \mathbf{A}, \mathbf{C} \rangle \leq \langle \mathbf{B}, \mathbf{D} \rangle$, if all of $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are PSD matrices of the same dimension. Thus, $\langle \mathbf{K}, \mathbf{J} \rangle \approx_{2r} \langle \mathbf{K}', \mathbf{J}' \rangle$. The conclusion follows upon realizing

$$\begin{aligned} \langle \mathbf{K}, \mathbf{J} \rangle &= \sum_{i \in [n]} \mathbf{v}_i^2 \text{Tr}(\mathbf{Z}^{-1} \mathbf{C}_i) - \sum_{(i,j) \in [n] \times [n]} \mathbf{v}_i \mathbf{v}_j \text{Tr}(\mathbf{Z}^{-1} \mathbf{C}_i \mathbf{Z}^{-1} \mathbf{C}_j) \\ &= \sum_{i \in [n]} \mathbf{v}_i^2 \text{Tr}(\mathbf{M}_i) - \sum_{(i,j) \in [n] \times [n]} \mathbf{v}_i \mathbf{v}_j \text{Tr}(\mathbf{M}_i \mathbf{M}_j) = \mathbf{v}^\top \nabla^2 f(\mathbf{t}) \mathbf{v}, \end{aligned}$$

and similarly, $\langle \mathbf{K}', \mathbf{J}' \rangle = \mathbf{v}^\top \nabla^2 f(\mathbf{t}') \mathbf{v}$, by comparing to Fact 2 and using the cyclic property of trace. This establishes $\mathbf{v}^\top \nabla^2 f(\mathbf{t}) \mathbf{v} \approx_{2r} \mathbf{v}^\top \nabla^2 f(\mathbf{t}') \mathbf{v}$ for all $\mathbf{v} \in \mathbb{R}^n$, as desired. \square

3.2 Termination condition

In this section, we quantify the suboptimality gap (with respect to Barthe's objective) needed for $\mathbf{t} \in \mathbb{R}^n$ to induce a (\mathbf{c}, ϵ) -Forster transform $\mathbf{R}(\mathbf{t})$, as a function of ϵ and problem parameters. Our proof makes use of local adjustments and is inspired by a similar technique in [CMTV17].

Lemma 3. *Let $\epsilon \in (0, 1)$, and suppose $\mathbf{t} \in \mathbb{R}^n$ satisfies*

$$f(\mathbf{t}) - f(\mathbf{t}^*) \leq \frac{\epsilon^2 \min_{i \in [n]} \mathbf{c}_i^2}{2} \quad (26)$$

where $\mathbf{t}^* \in \arg \min_{\mathbf{t} \in \mathbb{R}^n} f(\mathbf{t})$. Then $\mathbf{R}(\mathbf{t})$ is a (\mathbf{c}, ϵ) -Forster transform.

Proof. We prove the contrapositive. Suppose $\mathbf{R}(\mathbf{t})$ is not a (\mathbf{c}, ϵ) -Forster transform. By Lemma 1, there are two cases of leverage score violations to consider. We show that both cases contradict (26), by designing local improvements to \mathbf{t} in any coordinate with a violating leverage score.

Case 1. Suppose that for some $i \in [n]$, we have $\tau_i(\mathbf{S}(\mathbf{t})\mathbf{A}) > \exp(\epsilon)\mathbf{c}_i$. Let $\mathbf{t}' := \mathbf{t} - \delta \mathbf{e}_i$ for some choice of $\delta > 0$ that we will optimize later. Then,

$$\begin{aligned} f(\mathbf{t}) - f(\mathbf{t}') &= \log \left(\frac{\det(\mathbf{Z}(\mathbf{t}))}{\det(\mathbf{Z}(\mathbf{t}'))} \right) - \delta \mathbf{c}_i \\ &= \log \left(\frac{\det(\mathbf{Z}(\mathbf{t}))}{\det(\mathbf{Z}(\mathbf{t}) + (\exp(-\delta) - 1) \exp(\mathbf{t}_i) \mathbf{a}_i \mathbf{a}_i^\top)} \right) - \delta \mathbf{c}_i \\ &= \log \left(\frac{\det(\mathbf{Z}(\mathbf{t}))}{\det(\mathbf{Z}(\mathbf{t})) (1 + (\exp(-\delta) - 1) \exp(\mathbf{t}_i) \mathbf{a}_i^\top \mathbf{Z}(\mathbf{t})^{-1} \mathbf{a}_i)} \right) - \delta \mathbf{c}_i \\ &= \log \left(\frac{\det(\mathbf{Z}(\mathbf{t}))}{\det(\mathbf{Z}(\mathbf{t})) (1 + (\exp(-\delta) - 1) \tau_i(\mathbf{S}(\mathbf{t})\mathbf{A}))} \right) - \delta \mathbf{c}_i \\ &= -\log(1 + (\exp(-\delta) - 1) \tau_i(\mathbf{S}(\mathbf{t})\mathbf{A})) - \delta \mathbf{c}_i \\ &\geq (1 - \exp(-\delta)) \tau_i(\mathbf{S}(\mathbf{t})\mathbf{A}) - \delta \mathbf{c}_i \\ &\geq \left(\delta - \frac{\delta^2}{2} \right) \tau_i(\mathbf{S}(\mathbf{t})\mathbf{A}) - \delta \mathbf{c}_i > \delta \epsilon \mathbf{c}_i - \frac{\delta^2}{2} \geq \frac{1}{2} (\epsilon \mathbf{c}_i)^2, \end{aligned}$$

where the first two lines expanded definitions, the third line uses the matrix determinant lemma, the fourth line uses the definition of leverage scores (12), the sixth line uses $-\log(1 + c) \geq -c$ for all $c \in \mathbb{R}$, and the seventh line uses $1 - \exp(-c) \geq c - \frac{c^2}{2}$, $\exp(c) - 1 \geq c$ for $c > 0$ and $\tau_i(\mathbf{S}(\mathbf{t})\mathbf{A}) \leq 1$ (Fact 1). By choosing the optimal $\delta = \mathbf{c}_i \epsilon$, we have a contradiction to (26).

Case 2. Suppose that for some $i \in [n]$, we have $\tau_i(\mathbf{S}(\mathbf{t})\mathbf{A}) < \exp(-\epsilon)\mathbf{c}_i \leq \frac{\mathbf{c}_i}{1+\epsilon}$. Let $\mathbf{t}' := \mathbf{t} + \delta \mathbf{e}_i$ for some choice of $\delta > 0$ that we will optimize later. Then, following analogous derivations as before,

$$\begin{aligned} f(\mathbf{t}) - f(\mathbf{t}') &= -\log(1 + (\exp(\delta) - 1) \tau_i(\mathbf{S}(\mathbf{t})\mathbf{A})) + \delta \mathbf{c}_i \\ &> -\log \left(1 + (\exp(\delta) - 1) \frac{\mathbf{c}_i}{1 + \epsilon} \right) + \delta \mathbf{c}_i \\ &= \log(1 + \epsilon) - (1 - \mathbf{c}_i) \log \left(1 + \frac{\epsilon}{1 - \mathbf{c}_i} \right) \geq \frac{1}{2} (\epsilon \mathbf{c}_i)^2, \end{aligned}$$

where the second line uses $\tau_i(\mathbf{S}(\mathbf{t})\mathbf{A}) < \frac{\mathbf{c}_i}{1+\epsilon}$, the third line chose $\delta = \log(1 + \frac{\epsilon}{1-\mathbf{c}_i})$, and used $\log(1+c) - b \log(1 + \frac{c}{b}) \geq \frac{1}{2}((1-b)c)^2$ for all $b, c \in (0, 1)$. Again this gives a contradiction to (26). We remark that the case of $\mathbf{c}_i = 1$ can be handled using a limiting argument. \square

3.3 Box-constrained Newton's method

Thus far we have established that f is Hessian stable in $\|\cdot\|_\infty$ (Proposition 4) and needs to be minimized to error $\frac{1}{2}\epsilon^2 \min_{i \in [n]} \mathbf{c}_i^2$ for our desired application (Lemma 3). We also are given under Assumption 1 that the global minimizer lies inside $\mathbb{B}_\infty(\log(\kappa))$.

It remains to give an algorithm for efficiently optimizing Hessian stable functions. Fortunately, such a toolkit was provided by [CMTV17, CPW21]. The former work designed an approximation-tolerant box-constrained Newton's method, tailored towards objectives whose Hessians display a certain combinatorial structure, and the latter work showed how to optimize box-constrained quadratics in these structured Hessians. We can leverage this toolkit due to the next observation.

Lemma 4. *For all $\mathbf{t} \in \mathbb{R}^n$, $\nabla^2 f(\mathbf{t})$ is a graph Laplacian, i.e., $\nabla_{ij}^2 f(\mathbf{t}) \geq 0$ iff $i = j$, and*

$$\sum_{j \in [n]} \nabla_{ij}^2 f(\mathbf{t}) = 0 \text{ for all } i \in [n].$$

Proof. The first property is immediate by inspection of (21), and using that all the $\mathbf{M}_i(\mathbf{t}) \in \mathbb{S}_{\geq 0}^{d \times d}$. The second property follows because $\sum_{j \in [n]} \mathbf{M}_j(\mathbf{t}) = \mathbf{I}_d$, so for all $i \in [n]$ we have

$$\sum_{j \in [n]} \nabla_{ij}^2 f(\mathbf{t}) = \text{Tr}(\mathbf{M}_i(\mathbf{t})) - \left\langle \mathbf{M}_i(\mathbf{t}), \sum_{j \in [n]} \mathbf{M}_j(\mathbf{t}) \right\rangle = \text{Tr}(\mathbf{M}_i(\mathbf{t})) - \langle \mathbf{M}_i(\mathbf{t}), \mathbf{I}_d \rangle = 0.$$

\square

We remark that Lemma 4 gives another short proof of f 's convexity: it is well-known that graph Laplacians are PSD matrices, which follows e.g., by the Gershgorin circle theorem.

One complication that arises in our algorithm is that computing the Hessian $\nabla^2 f$ is more expensive than providing matrix-vector query access to it, due to a convenient factorization.

Lemma 5. *Given $\mathbf{t} \in \mathbb{R}^n$, we can compute $\nabla f(\mathbf{t})$ in $O(nd^{\omega-1})$ time and $\nabla^2 f(\mathbf{t})$ in $O(n^2 d^{\omega-2})$ time. Additionally, given $\mathbf{t}, \mathbf{v} \in \mathbb{R}^n$, we can compute $\nabla^2 f(\mathbf{t})\mathbf{v}$ in $O(nd^{\omega-1})$ time.*

Proof. Recall the formulas for $\nabla f(\mathbf{t})$, $\nabla^2 f(\mathbf{t})$ in Fact 2. For the former claim, following (18), (20), we first compute $\mathbf{S}(\mathbf{t})\mathbf{A}$ in time $O(nd)$, which lets us compute $\mathbf{Z}(\mathbf{t})$ in time $O(nd^{\omega-1})$ by multiplying $d \times n$ and $n \times d$ matrices. We can then compute $\mathbf{A}\mathbf{R}(\mathbf{t})$ to obtain all of the vectors $\tilde{\mathbf{a}}_i(\mathbf{t})$ in $O(nd^{\omega-1})$ time. This lets us obtain all $\text{Tr}(\mathbf{M}_i(\mathbf{t})) = \mathbf{s}_i(\mathbf{t})^2 \|\tilde{\mathbf{a}}_i(\mathbf{t})\|_2^2$ in $O(nd)$ additional time.

It remains to compute the $n \times n$ matrix with $(i, j)^{\text{th}}$ entry $\text{Tr}(\mathbf{M}_i(\mathbf{t})\mathbf{M}_j(\mathbf{t}))$. Observe that

$$\text{Tr}(\mathbf{M}_i(\mathbf{t})\mathbf{M}_j(\mathbf{t})) = \mathbf{s}_i(\mathbf{t})^2 \mathbf{s}_j(\mathbf{t})^2 \langle \tilde{\mathbf{a}}_i(\mathbf{t}), \tilde{\mathbf{a}}_j(\mathbf{t}) \rangle^2.$$

Thus it is enough to form the matrix with $(i, j)^{\text{th}}$ entry $\langle \tilde{\mathbf{a}}_i(\mathbf{t}), \tilde{\mathbf{a}}_j(\mathbf{t}) \rangle$, multiply it entrywise by $\mathbf{s}_i(\mathbf{t})\mathbf{s}_j(\mathbf{t})$, and entrywise square it, in $O(n^2)$ time. The former matrix is $\mathbf{A}\mathbf{Z}(\mathbf{t})^{-1}\mathbf{A}^\top$, which takes time $O(n^2 d^{\omega-2})$ time to compute by multiplying $n \times d$, $d \times d$, and $d \times n$ matrices.

For the latter claim, we can again first compute

$$\text{diag} \left(\{\text{Tr}(\mathbf{M}_i(\mathbf{t}))\}_{i \in [n]} \right) \mathbf{v}$$

in time $O(nd^{\omega-1})$ using the steps described above. To implement $\nabla^2 f(\mathbf{t})\mathbf{v}$, it remains to compute

$$\left\langle \mathbf{M}_i(\mathbf{t}), \sum_{j \in [n]} \mathbf{v}_j \mathbf{M}_j(\mathbf{t}) \right\rangle = \exp(\mathbf{t}_i) \left(\mathbf{R}(\mathbf{t}) \left(\sum_{j \in [n]} \mathbf{v}_j \mathbf{M}_j(\mathbf{t}) \right) \mathbf{R}(\mathbf{t}) \right) [\mathbf{a}_i, \mathbf{a}_i] \text{ for all } i \in [n].$$

Observe that

$$\mathbf{C} := \mathbf{R}(\mathbf{t}) \left(\sum_{j \in [n]} \mathbf{v}_j \mathbf{M}_j(\mathbf{t}) \right) \mathbf{R}(\mathbf{t}) = \mathbf{Z}(\mathbf{t})^{-1} \left(\sum_{j \in [n]} \mathbf{v}_j \exp(\mathbf{t}_j) \mathbf{a}_j \mathbf{a}_j^\top \right) \mathbf{Z}(\mathbf{t})^{-1},$$

which can be computed in $O(nd^{\omega-1})$ time by first forming the middle matrix on the right-hand side via multiplying $d \times n$ and $n \times d$ matrices. Finally to compute all $\mathbf{C}[\mathbf{a}_i, \mathbf{a}_i]$, we can take the rows of \mathbf{AC} and obtain their dot products with rows of \mathbf{A} which requires $O(nd^{\omega-1})$ time to compute. \square

To capitalize on the faster matrix-vector access given by Lemma 4, in Section 4 we give a proof of Theorem 2, our main result on the implicit sparsification of graph Laplacians. We will apply Theorem 2 to sparsify the Hessian of a regularized variant of Barthe's objective. Next, we require a tool from [CPW21] for optimizing box-constrained quadratics in a graph Laplacian.

Proposition 5 (Theorem 1.1, [CPW21]). *Let $\delta \in (0, 1)$, let $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ have $\mathbf{l} \leq \mathbf{u}$ entrywise, let $\mathbf{b}, \mathbf{t} \in \mathbb{R}^n$, and let $\mathbf{L} \in \mathbb{S}_{\geq \mathbf{0}}^{n \times n}$ be a graph Laplacian with $\text{nnz}(\mathbf{L}) \leq m$. Let*

$$\mathcal{W} := \{\mathbf{w} \in \mathbb{R}^n \mid \mathbf{l}_i \leq \mathbf{t}_i + \mathbf{w}_i \leq \mathbf{r}_i \text{ for all } i \in [n]\}.$$

There is an algorithm $\mathcal{O}(\mathbf{L}, \mathbf{b}, \mathcal{W})$ that runs in time $O((n + m)^{1+o(1)} \log(\frac{1}{\delta}))$, and with probability $\geq 1 - \delta$, it returns $\mathbf{v} \in \mathcal{W}$, satisfying

$$\langle \mathbf{b}, \mathbf{v} \rangle + \frac{1}{2} \mathbf{L} [\mathbf{v}, \mathbf{v}] \leq \frac{1}{2} \min_{\mathbf{w} \in \mathcal{W}} \left\{ \langle \mathbf{b}, \mathbf{w} \rangle + \frac{1}{2} \mathbf{L} [\mathbf{w}, \mathbf{w}] \right\}.$$

We give additional discussion on Proposition 5 in Appendix A, as in [CPW21] it was only stated for the case $\mathbf{l} = \mathbf{0}_n$ and \mathbf{u} is ∞ in each coordinate, i.e., the box constraint is simply the positive orthant $\mathbb{R}_{\geq 0}^n$. However, the techniques extend straightforwardly to general box constraints [CPW25].

We now show how to use Proposition 5 to efficiently optimize an ℓ_∞ -Hessian stable function. The following proof is based on Theorem 3.4, [CMTV17], but adapts it to tolerate multiplicative error in the Hessian computation. We note that a similar multiplicatively-robust generalization appeared earlier as Lemma 20, [AJJ⁺22], but was too restrictive for our purposes.

Lemma 6. *Let convex $F : \mathbb{R}^n \rightarrow \mathbb{R}$ be $(1, 2)$ -Hessian stable with respect to $\|\cdot\|_\infty$, and let $\mathbf{t}^* \in \arg \min_{\mathbf{t} \in \mathbb{R}^n} F(\mathbf{t})$ have $\|\mathbf{t}^*\|_\infty \leq \log(\kappa)$. For $\mathbf{t} \in \mathbb{R}^n$, $\alpha \geq 1$, let $\tilde{\mathbf{L}} \in \mathbb{S}_{\geq \mathbf{0}}^{n \times n}$ be a graph Laplacian with*

$$\nabla^2 F(\mathbf{t}) \preceq \tilde{\mathbf{L}} \preceq \alpha \nabla^2 F(\mathbf{t}).$$

Then for any $\mathbf{t} \in \mathbb{B}_\infty(\log(\kappa))$, if $\mathbf{t}' \leftarrow \mathbf{t} + \mathcal{O}(8\tilde{\mathbf{L}}, \nabla F(\mathbf{t}), \mathbb{B}_\infty(\mathbf{t}, 1) \cap \mathbb{B}_\infty(\log(\kappa)))$, where \mathcal{O} is as in Proposition 5, we have

$$F(\mathbf{t}') - F(\mathbf{t}^*) \leq \left(1 - \frac{1}{240\alpha \log(\kappa)}\right) (F(\mathbf{t}) - F(\mathbf{t}^*)).$$

Proof. For any \mathbf{u} with $\|\mathbf{t} - \mathbf{u}\|_\infty \leq 1$, Hessian stability of F yields the bounds

$$\begin{aligned} F(\mathbf{u}) - F(\mathbf{t}) - \langle \nabla F(\mathbf{t}), \mathbf{u} - \mathbf{t} \rangle &= \int_0^1 (1 - \lambda) \nabla^2 F((1 - \lambda)\mathbf{t} + \lambda\mathbf{u}) [\mathbf{u} - \mathbf{t}, \mathbf{u} - \mathbf{t}] d\lambda \\ &\leq \int_0^1 (1 - \lambda) e^2 \tilde{\mathbf{L}} [\mathbf{u} - \mathbf{t}, \mathbf{u} - \mathbf{t}] d\lambda \\ &\leq \frac{e^2}{2} \tilde{\mathbf{L}} [\mathbf{u} - \mathbf{t}, \mathbf{u} - \mathbf{t}] \leq 4\tilde{\mathbf{L}} [\mathbf{u} - \mathbf{t}, \mathbf{u} - \mathbf{t}], \\ F(\mathbf{u}) - F(\mathbf{t}) - \langle \nabla F(\mathbf{t}), \mathbf{u} - \mathbf{t} \rangle &\geq \frac{1}{2\alpha e^2} \tilde{\mathbf{L}} [\mathbf{u} - \mathbf{t}, \mathbf{u} - \mathbf{t}] \geq \frac{1}{15\alpha} \tilde{\mathbf{L}} [\mathbf{u} - \mathbf{t}, \mathbf{u} - \mathbf{t}]. \end{aligned} \tag{27}$$

Next define

$$\hat{\boldsymbol{\delta}} := \arg \min_{\substack{\|\boldsymbol{\delta}\|_\infty \leq 1 \\ \mathbf{t} + \boldsymbol{\delta} \in \mathbb{B}_\infty(\log(\kappa))}} \langle \nabla F(\mathbf{t}), \boldsymbol{\delta} \rangle + 4\tilde{\mathbf{L}}[\boldsymbol{\delta}, \boldsymbol{\delta}],$$

and observe that for $\boldsymbol{\delta} := \mathbf{t}' - \mathbf{t} = \mathcal{O}(8\tilde{\mathbf{L}}, \nabla F(\mathbf{t}), \mathbb{B}_\infty(\mathbf{t}, 1) \cap \mathbb{B}_\infty(\log(\kappa)))$, we have

$$\begin{aligned} \langle \nabla F(\mathbf{t}), \boldsymbol{\delta} \rangle + 4\tilde{\mathbf{L}}[\boldsymbol{\delta}, \boldsymbol{\delta}] &\leq \frac{1}{2} \left(\langle \nabla F(\mathbf{t}), \hat{\boldsymbol{\delta}} \rangle + 4\tilde{\mathbf{L}}[\hat{\boldsymbol{\delta}}, \hat{\boldsymbol{\delta}}] \right) \\ &\leq \frac{1}{2} \left(\langle \nabla F(\mathbf{t}), \boldsymbol{\delta}^* \rangle + 4\tilde{\mathbf{L}}[\boldsymbol{\delta}^*, \boldsymbol{\delta}^*] \right), \\ \text{for any } \|\boldsymbol{\delta}^*\|_\infty \leq 1 \text{ with } \mathbf{t} + \boldsymbol{\delta}^* \in \mathbb{B}_\infty(\log(\kappa)), \end{aligned} \tag{28}$$

from the oracle guarantee and definition of $\hat{\boldsymbol{\delta}}$. Hence, applying the upper bounds in (27) and (28),

$$\begin{aligned} F(\mathbf{t}') &\leq F(\mathbf{t}) + \langle \nabla F(\mathbf{t}), \boldsymbol{\delta} \rangle + 4\tilde{\mathbf{L}}[\boldsymbol{\delta}, \boldsymbol{\delta}] \\ &\leq F(\mathbf{t}) + \frac{1}{2} \left(\langle \nabla F(\mathbf{t}), \boldsymbol{\delta}^* \rangle + 4\tilde{\mathbf{L}}[\boldsymbol{\delta}^*, \boldsymbol{\delta}^*] \right), \end{aligned} \tag{29}$$

for our choice of $\boldsymbol{\delta}^*$ satisfying the bounds in (28). We choose $\boldsymbol{\delta}^* = \frac{c}{2\log(\kappa)}(\mathbf{t}^* - \mathbf{t})$ where $c = \frac{1}{60\alpha}$. First observe that this is a valid choice of movement, because

$$\begin{aligned} \mathbf{t} + \boldsymbol{\delta}^* &= \left(1 - \frac{c}{2\log(\kappa)}\right) \mathbf{t} + \frac{c}{2\log(\kappa)} \mathbf{t}^* \in \mathbb{B}_\infty(\log(\kappa)), \\ \|\boldsymbol{\delta}^*\|_\infty &\leq \frac{1}{2\log(\kappa)} (\|\mathbf{t}\|_\infty + \|\mathbf{t}^*\|_\infty) \leq \frac{2\log(\kappa)}{2\log(\kappa)} = 1, \end{aligned}$$

where both inequalities used that $\mathbf{t}, \mathbf{t}^* \in \mathbb{B}_\infty(\log(\kappa))$, which is a convex set. Thus,

$$\begin{aligned} \frac{1}{2} \left(\langle \nabla F(\mathbf{t}), \boldsymbol{\delta}^* \rangle + 4\tilde{\mathbf{L}}[\boldsymbol{\delta}^*, \boldsymbol{\delta}^*] \right) &= \frac{1}{120\alpha} \left(\left\langle \nabla F(\mathbf{t}), \frac{1}{c} \boldsymbol{\delta}^* \right\rangle + \frac{1}{15\alpha} \tilde{\mathbf{L}} \left[\frac{1}{c} \boldsymbol{\delta}^*, \frac{1}{c} \boldsymbol{\delta}^* \right] \right) \\ &\leq \frac{1}{120\alpha} \left(F \left(\mathbf{t} + \frac{1}{c} \boldsymbol{\delta}^* \right) - F(\mathbf{t}) \right) \end{aligned}$$

$$\leq -\frac{1}{240\alpha \log(\kappa)} (F(\mathbf{t}) - F(\mathbf{t}^*)).$$

The first inequality above used the lower bound in (27), and the second inequality used convexity of F . At this point, combining with (29) yields the conclusion. \square

3.4 Proof of Theorem 1

In this section we put together the pieces we have built to obtain our final algorithm. To begin, we note that under Assumption 1, the following simple initial function error bound holds.

Lemma 7. *Under Assumption 1, letting $\mathbf{t}^* \in \arg \min_{\mathbf{t} \in \mathbb{R}^n} f(\mathbf{t}) \cap \mathbb{B}_\infty(\log(\kappa))$, we have that*

$$f(\mathbf{0}_n) - f(\mathbf{t}^*) \leq \frac{d \log^2(\kappa)}{2}.$$

Proof. We first note that for all $\mathbf{t}, \mathbf{v} \in \mathbb{R}^n$, we have

$$\nabla^2 f(\mathbf{t}) [\mathbf{v}, \mathbf{v}] \leq \mathbf{diag} \left(\{\text{Tr}(\mathbf{M}_i(\mathbf{t}))\}_{i \in [n]} \right) [\mathbf{v}, \mathbf{v}] = \sum_{i \in [n]} \boldsymbol{\tau}_i(\mathbf{S}(\mathbf{t})\mathbf{A}) \mathbf{v}_i^2 \leq d \|\mathbf{v}\|_\infty^2,$$

i.e., f is d -smooth with respect to $\|\cdot\|_\infty$ (we used Fact 1 in the last inequality above). Thus by the second-order Taylor expansion from \mathbf{t}^* to $\mathbf{0}_n$, and using that $\nabla f(\mathbf{t}^*) = \mathbf{0}_n$,

$$\begin{aligned} f(\mathbf{0}_n) &= f(\mathbf{t}^*) + \int_0^1 (1-\lambda) \nabla^2 f((1-\lambda)\mathbf{t}^*) [\mathbf{t}^*, \mathbf{t}^*] d\lambda \\ &\leq f(\mathbf{t}^*) + \frac{d}{2} \|\mathbf{t}^*\|_\infty^2 \leq f(\mathbf{t}^*) + \frac{d \log^2(\kappa)}{2}. \end{aligned}$$

\square

We can now prove Theorem 1, the main result of this section.

Theorem 1. *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{c} \in (0, 1]^n$ satisfy Assumption 1, and let $\delta, \epsilon \in (0, 1)$. There is an algorithm that computes \mathbf{R} , a (\mathbf{c}, ϵ) -Forster transform of \mathbf{A} , with probability $\geq 1 - \delta$, in time*

$$O \left(n d^{\omega-1} \log(\kappa) \left(\frac{n \log(\kappa)}{\delta \epsilon \mathbf{c}_{\min}} \right)^{o(1)} \right), \text{ where } \mathbf{c}_{\min} := \min_{i \in [n]} \mathbf{c}_i.$$

Proof. Throughout this proof, we let f be Barthe's objective, and

$$F(\mathbf{t}) := f(\mathbf{t}) + \frac{\epsilon^2 \mathbf{c}_{\min}^2}{4 \log^2(\kappa)} \mathbf{t}^\top \mathbf{\Pi} \mathbf{t},$$

where $\mathbf{\Pi} := \mathbf{I}_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^\top$. Our goal is to optimize F to error $\frac{\epsilon^2 \mathbf{c}_{\min}^2}{4}$ over $\mathbb{B}_\infty(\log(\kappa))$. To see why this suffices, note that for all $\mathbf{t} \in \mathbb{B}_\infty(\log(\kappa))$, we have $\mathbf{t}^\top \mathbf{\Pi} \mathbf{t} \leq \|\mathbf{t}\|_2^2 \leq \log^2(\kappa)$, and hence any $\frac{\epsilon^2 \mathbf{c}_{\min}^2}{4}$ -minimizer to F over $\mathbb{B}_\infty(\log(\kappa))$ satisfies

$$f(\mathbf{t}) - f(\mathbf{t}^*) \leq F(\mathbf{t}) - F(\mathbf{t}^*) + \frac{\epsilon^2 \mathbf{c}_{\min}^2}{4} \leq \frac{\epsilon^2 \mathbf{c}_{\min}^2}{2}.$$

Now, applying Lemma 3 gives the claim, because computing $\mathbf{R}(\mathbf{t})$ does not dominate the stated runtime (as described in Lemma 5). Note that each time we apply Lemma 6 with $\alpha \leftarrow (\frac{n \log(\kappa)}{\epsilon \mathbf{c}_{\min}})^{o(1)}$, we improve the function error by a multiplicative $1 - \Omega((\alpha \log(\kappa))^{-1})$. Moreover, the initial function error is bounded as in Lemma 7. Thus to obtain the stated runtime, it is enough to show how to implement each call to Lemma 6 with $\alpha \leftarrow (\frac{n \log(\kappa)}{\epsilon \mathbf{c}_{\min}})^{o(1)}$, in time

$$O\left(nd^{\omega-1} \left(\frac{n \log(\kappa)}{\delta \epsilon \mathbf{c}_{\min}}\right)^{o(1)}\right). \quad (30)$$

Adjusting δ by the number of calls and taking a union bound then gives the claim.

To achieve this runtime we first produce a sparse graph Laplacian matrix $\tilde{\mathbf{L}}$ satisfying (7) for $\Delta := \frac{\epsilon^2 \mathbf{c}_{\min}^2}{4 \log^2(\kappa)}$, and $\mathbf{L} \leftarrow \nabla^2 f(\mathbf{t})$ for some iterate \mathbf{t} . Recalling that $\text{Tr}(\mathbf{L}) = \text{Tr}(\mathbf{I}_d) = d$, Theorem 2 and Lemma 5 guarantee that we can compute such a $\tilde{\mathbf{L}}$ with probability $\geq 1 - \delta$ within time (30). Given $\tilde{\mathbf{L}}$, the per-iteration runtime follows from Proposition 5 which does not dominate. \square

Remark 1. *For highly-accurate solutions or extremely small failure probabilities (i.e., δ, ϵ smaller than an inverse polynomial in n), the subpolynomial dependences on $\frac{1}{\delta}, \frac{1}{\epsilon}$ in Theorem 1 could be dominant factors. However, these subpolynomial factors only arise due to the use of Theorem 2 to sparsely approximate Hessians of Barthe's objective. If we instead directly compute the Hessians via Lemma 5, then slightly modifying the proof of Theorem 1 yields an alternate runtime of*

$$O\left(n^2 d^{\omega-2} \log(\kappa) \text{polylog}\left(\frac{n \log(\kappa)}{\delta \epsilon \mathbf{c}_{\min}}\right)\right).$$

This runtime gives a worse dependence on n , but improves the dependences on other parameters (i.e., $\frac{1}{\delta}, \frac{1}{\epsilon}, \frac{1}{\mathbf{c}_{\min}}$) from subpolynomial to polylogarithmic.

4 Sparsifying Laplacians with Matrix-Vector Queries

In this section, we prove Theorem 2. Our approach is inspired by [JLM⁺23], who showed how to tightly approximate a graph Laplacian using a dictionary of edge Laplacians, given appropriate access (i.e., to its inverse). Our result is incomparable, as it obtains a substantially faster runtime under weaker access, but gives a much looser approximation factor.

In Section 4.1, we first define a combinatorial structure that we maintain for the iterates of our methods, supporting efficient matrix-vector products. In Section 4.2 we next show how to discretize a distance-structured vector in a way that is compatible with our combinatorial structure.

We use our vector approximations in Section 4.3 to implement an approximate packing SDP oracle for combinatorially structured inputs. Finally, in Sections 4.4 and 4.5, we give an efficient reduction from the two-sided approximation in (7) to our packing SDP oracle.

4.1 Sum-of-cliques representation

Throughout this section, we let E index the (unordered) edges of a graph on $[n] \times [n]$, i.e., $|E| = \binom{n}{2}$ and E has one index (i, j) for each unordered tuple $(i, j) \in [n] \times [n]$. For an index $e = (i, j) \in E$,

we also define the corresponding edge Laplacian $\mathbf{L}_e \in \mathbb{S}_{\geq 0}^{n \times n}$:

$$\mathbf{L}_e := (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top.$$

We also define the graph Laplacian induced by a weight vector $\mathbf{v} \in \mathbb{R}_{\geq 0}^E$ by

$$\mathcal{L}(\mathbf{v}) := \sum_{e \in E} \mathbf{v}_e \mathbf{L}_e.$$

Additionally, we define the “dual” operator \mathcal{L}^* which takes input $\mathbf{M} \in \mathbb{R}^{n \times n}$ as

$$\mathcal{L}^*(\mathbf{M}) := \{\langle \mathbf{L}_e, \mathbf{M} \rangle\}_{e \in E} \in \mathbb{R}^E.$$

Next, we define a helpful combinatorial structure for maintaining our iterates efficiently.

Definition 4 (Sum-of-cliques). *For $K \in \mathbb{N}$, we say $\mathbf{v} \in \mathbb{R}_{\geq 0}^E$ is a K -sum-of-cliques (K -SOC) if there exists $\mathbf{w} \in \mathbb{R}_{\geq 0}^K$ and K partitions of subsets $\{S_j\}_{j \in [K]}$ of $[n]$, $\{\mathcal{P}_j\}_{j \in [K]}$, such that*

$$\mathcal{L}(\mathbf{v}) = \sum_{j \in [K]} \mathbf{w}_j \sum_{S \in \mathcal{P}_j} \mathbf{L}_S, \quad (31)$$

where \mathbf{L}_S refers to an unweighted clique Laplacian placed over the set $S \subseteq [n]$, i.e.,

$$\mathbf{L}_S = |S| \mathbf{I}_S - \mathbf{1}_S \mathbf{1}_S^\top. \quad (32)$$

We say $\mathbf{a} \in \{0, 1\}^E$ is an anti-sum-of-cliques (ASOC) if there exists a partition \mathcal{A} of a subset $S \subseteq [n]$ such that

$$\mathcal{L}(\mathbf{a}) = \mathbf{L}_S - \sum_{A \in \mathcal{A}} \mathbf{L}_A. \quad (33)$$

The rest of this section shows that sum-of-cliques and anti-sum-of-cliques vectors induce Laplacians that support fast matrix-vector products and sparsification.

Remark 2. *We briefly discuss our computational model for representing partitions; it will be clear that this representation can be maintained under all operations in this section. We represent a k -piece partition $\mathcal{P} = \{P_i\}_{i \in [k]}$ of a subset $S \subseteq [n]$ using two arrays. The first array has k elements, each of which is an array containing the members of P_i . The second array has n elements, each of which marks the partition piece vertex $j \in [n]$ belongs to, or 0 if $j \notin \cup_{i \in [k]} P_i$.*

Matrix-vector products. We now prove that SOCs and ASOCs induce Laplacians with fast matrix-vector products. We require a helper result on computing mutual refinements of partitions.

Lemma 8. *Let \mathcal{P}, \mathcal{A} be partitions of $[n]$. Let \mathcal{B} be the mutual refinement of \mathcal{P}, \mathcal{A} , i.e., \mathcal{B} is a partition of $[n]$ such that if $S \in \mathcal{P}$ and $T \in \mathcal{A}$, then $S \cap T \in \mathcal{B}$. If \mathcal{P} and \mathcal{A} are explicitly given, we can compute \mathcal{B} in $O(n \log(n))$ time.³*

Proof. In $O(n)$ time, we compute for each element $i \in [n]$ the ordered tuple (a, p) , where a is the partition piece in \mathcal{A} containing i , and similarly p is the partition piece in \mathcal{P} . Next we can sort the $\{(a, p)\}$ in lexicographical order to find the indices of the partition pieces in \mathcal{B} . Finally we can assign each element of $[n]$ to the appropriate partition piece in \mathcal{B} . \square

³There is a simple randomized $O(n)$ time solution, but it uses hashing, so we include this solution for completeness.

Lemma 9. Let \mathbf{v} be a K -SOC satisfying (31) with respect to $\mathbf{w} \in \mathbb{R}_{\geq 0}^K$ and $\{\mathcal{P}_j\}_{j \in [K]}$, partitions of $\{S_j\}_{j \in [K]}$, and let \mathbf{a} be an ASOC satisfying (33) with respect to \mathcal{A} , a partition of $S \subseteq [n]$. Suppose \mathbf{w} , $\{\mathcal{P}_j\}_{j \in [K]}$, and \mathcal{A} are explicitly given. We can compute

$$\mathcal{L}(\mathbf{v} \circ \mathbf{a})\mathbf{u} = \sum_{e \in E} \mathbf{v}_e \mathbf{a}_e \mathbf{L}_e \mathbf{u}$$

for any $\mathbf{u} \in \mathbb{R}^n$ in $O(nK \log(n))$ time.

Proof. Let $\mathbf{v}_j \in \{0, 1\}^E$ be the 1-SOC corresponding to a unit weight and a single partition \mathcal{P}_j , so that $\mathbf{v} = \sum_{j \in [K]} \mathbf{w}_j \mathbf{v}_j$. We will show that we can implement the matrix-vector product

$$\mathcal{L}(\mathbf{v}_j \circ \mathbf{a})\mathbf{u} = \sum_{e \in E} [\mathbf{v}_j]_e \mathbf{a}_e \mathbf{L}_e \mathbf{u}$$

in $O(n \log(n))$ time. Summing over all $j \in [K]$ then gives the result.

In the rest of the proof let $\{T_1, \dots, T_m\}$ be the partition \mathcal{P}_j inducing \mathbf{v}_j . Also let $\mathbf{b} = \mathbf{1}_F - \mathbf{a}$ where $F \subseteq E$ indicates the edges corresponding to the clique on S , which is a 1-SOC. Observe that $\mathcal{L}(\mathbf{v}_j \circ \mathbf{a})\mathbf{u} = \mathcal{L}(\mathbf{v}_j) - \mathcal{L}(\mathbf{v}_j \circ \mathbf{b})$, and $\mathcal{L}(\mathbf{v}_j)\mathbf{u}$ is computable in $O(n)$ time:

$$\mathcal{L}(\mathbf{v}_j)\mathbf{u} = \sum_{T_i \in \mathcal{P}_j} \mathbf{L}_{T_i} \mathbf{u}.$$

In particular, each $\mathbf{L}_{T_i} \mathbf{u}$ can be evaluated in $O(T_i)$ time using the formula (32), and $\sum_{i \in [m]} |T_i| \leq n$.

Moreover, note that $\mathbf{b} \circ \mathbf{v}_j$ is itself a vector in $\{0, 1\}^E$ corresponding to a 1-SOC on the set of vertices $S \cap S_j \subseteq [n]$ (all other coordinates are 0 in either \mathbf{b} or \mathbf{v}_j). This is because each pair of vertices corresponds to a weight-1 edge in $\mathbf{b} \circ \mathbf{v}_j$ iff they lie in the same partition piece in both \mathcal{P}_j and \mathcal{A} , which is an equivalence relation forming a partition of $S \cap S_j$. In fact, this partition is the mutual refinement of \mathcal{P}_j and \mathcal{A} , following Lemma 8 (which we call with $[n] \leftarrow S \cap S_j$). Thus, we can compute the mutual refinement inducing $\mathbf{b} \circ \mathbf{v}_j$ in $O(n \log(n))$ time, and the same logic as above shows we can implement matrix-vector products through $\mathcal{L}(\mathbf{v}_j \circ \mathbf{b})$ in $O(n)$ time. \square

Sparsification. As a second key result, we show that it is possible to efficiently sparsify the product of a clique and an ASOC. Specifically, let $\mathbf{c} \in \{0, 1\}_{\geq 0}^E$ be the indicator vector of a clique over vertices $S \subseteq [n]$ (so $\mathcal{L}(\mathbf{c}) = \mathbf{L}_S$ defined in (32)), and let \mathbf{a} be an ASOC associated with partition \mathcal{A} of $A \subseteq [n]$. We give an algorithm for sparsifying $\mathcal{L}(\mathbf{c} \circ \mathbf{a})$ down to $\approx |S|$ edges.

We next provide some notation for capturing the structure of $\mathbf{c} \circ \mathbf{a}$. Let the pieces of \mathcal{A} be denoted $\{A_i\}_{i \in [k]}$. Without loss of generality, if there are ℓ pieces with nonempty intersection with S , let them be $\{A_i\}_{i \in [\ell]}$, i.e., the first ℓ pieces. Finally, denote $S_i := A_i \cap S$ for all $i \in [\ell]$. Then,

$$\mathbf{c} \circ \mathbf{a} = \sum_{i \in [\ell]} \sum_{j \in [i-1]} \mathbf{1}_{K_{ij}}, \tag{34}$$

where K_{ij} is the *complete unweighted bipartite graph* between S_i and S_j , and $\mathbf{1}_{K_{ij}} \subseteq \{0, 1\}^E$ indicates its edges. Correspondingly we denote each complete unweighted bipartite graph Laplacian by

$$\mathbf{L}_{ij} := \mathcal{L}(\mathbf{1}_{K_{ij}}), \text{ for all } 1 \leq j < i \leq \ell.$$

We next require a basic combinatorial fact on balanced partitions of integers.

Lemma 10. Let $\{n_i\}_{i \in [\ell]} \subset \mathbb{N}$ and let $Z := \sum_{i \in [\ell]} n_i$. Then if there exists no $i \in [\ell]$ with $n_i > \frac{Z}{3}$, there exists a partition of $[\ell]$ into $S, T = [\ell] \setminus S$ such that $\sum_{i \in S} n_i \in [\frac{1}{3}Z, \frac{2}{3}Z]$.

Proof. Iterate over the elements, adding them to S , until the next element would cause the sum of S to exceed $\frac{2Z}{3}$. Because each element is bounded by $\frac{Z}{3}$, $\sum_{i \in S} n_i \in [\frac{1}{3}Z, \frac{2}{3}Z]$ holds at termination. \square

We also will use a fast sparsification algorithm for complete bipartite graphs.

Lemma 11. Let G be a complete unweighted bipartite graph with bipartition $V = L \cup R$, and let $\delta \in (0, 1)$, and let $\mathbf{L}_G \in \mathbb{R}^{V \times V}$ denote its graph Laplacian. There is an algorithm that runs in time $O(|V| \log(\frac{|V|}{\delta}))$ and returns a graph Laplacian \mathbf{L}_H such that with probability $\geq 1 - \delta$,

$$\mathbf{L}_H \approx_1 \mathbf{L}_G, \text{nnz}(\mathbf{L}_H) = O\left(|V| \log\left(\frac{|V|}{\delta}\right)\right).$$

Proof. Let $m = O(|V| \log(\frac{|V|}{\delta}))$ denote the allowed sparsity parameter. For a sufficiently large constant in m , Theorem 1 of [SS11] shows that sampling m random edges of G with replacement, and adding them to H with weight $\frac{|L||R|}{m}$, gives a Laplacian \mathbf{L}_H meeting the desired criteria. Here, we used that all the effective resistances of the edges in G are the same, by symmetry. \square

We now give our overall sparsification algorithm for the graph with Laplacian $\mathbf{L}(\mathbf{c} \circ \mathbf{a})$. The main idea is to recursively apply Lemma 11 across balanced partitions whenever possible.

Lemma 12. Let $\{S_i\}_{i \in [\ell]}$ be a partition of $S \subseteq [n]$, let $\delta \in (0, 1)$, and let

$$\mathbf{L} := \sum_{i \in [\ell]} \sum_{j \in [i-1]} \mathbf{L}_{ij},$$

where \mathbf{L}_{ij} is the graph Laplacian of the complete unweighted bipartite graph between S_i and S_j . There is an algorithm that runs in time $O(|S| \log(n) \log(\frac{n}{\delta}))$, and returns a graph Laplacian $\tilde{\mathbf{L}}$ such that with probability $\geq 1 - \delta$,

$$\tilde{\mathbf{L}} \approx_1 \mathbf{L}, \text{nnz}(\tilde{\mathbf{L}}) = O\left(|S| \log(n) \log\left(\frac{n}{\delta}\right)\right).$$

Proof. Throughout the proof, let $C = \Theta(\log(\frac{n}{\delta}))$ with a large enough constant, such that when we apply Lemma 11 at most n times, we can assume that all calls succeed, run in time $\leq C|V|$ if called with input vertex set V , and return graphs with sparsity at most $C|V|$. We condition on this event for the remainder of the proof which gives the failure probability.

Our sparsification algorithm proceeds recursively in one of two ways as follows. Consider the sizes of partition pieces $\{n_i\}_{i \in [\ell]}$, where $n_i := |S_i|$ for all $i \in [\ell]$, and let $Z := |S| \leq n$.

If there is some partition piece $i \in [\ell]$ with $n_i > \frac{Z}{3}$, then we form a bipartition where one side consists of S_i and the other side consists of $\{S_j\}_{j \in [\ell] \setminus \{i\}}$. Note that all edges cross this bipartition are present in \mathbf{L} , due to each term \mathbf{L}_{ij} . Thus, we sparsify this complete unweighted bipartite graph using Lemma 11, and recurse on the remaining partition pieces $\{S_j\}_{j \in [\ell] \setminus \{i\}}$.

Otherwise, if there is no such partition piece, then by Lemma 10 there exists a balanced partition into two subsets of S containing between $\frac{Z}{3}$ and $\frac{2Z}{3}$ vertices. We again sparsify the complete unweighted bipartite graph across this bipartition using Lemma 11, and recurse on the two sides separately.

Overall, if we let $\mathcal{T}(Z)$ denote the time it takes to implement this recursive strategy given an input partitioned set $S \subseteq [n]$ with size $Z = |S|$, we have shown the recursion

$$\mathcal{T}(Z) \leq \max_{Z' \in [\frac{Z}{3}, \frac{2Z}{3}]} \mathcal{T}(Z') + \mathcal{T}(Z - Z') + CZ, \quad (35)$$

because both cases of the recursion are handled by (35) (in the first case discussed, there is only one recursion piece, which is dominated by the second case discussed). It is clear that (35) solves to $\mathcal{T}(Z) = O(CZ \log(Z))$, and the conclusion regarding the runtime follows from our choice of C and using that $Z \leq n$. A similar recursion holds for the sparsity bound. \square

4.2 Discretizing low-dimensional distances

In this section, we show how to discretize a target implicit vector $\mathbf{g} \in \mathbb{R}^E$ into a small number of SOCs or ASOCs via grid rounding, where E indexes the edge set of the clique on $[n]$. Our algorithm will specifically apply to the case where \mathbf{g} is implicitly specified by distances between the columns $\{\mathbf{q}_i\}_{i \in [n]} \subset \mathbb{R}^k$ of a matrix $\mathbf{Q} \in \mathbb{R}^{k \times n}$ in the following way:

$$\mathbf{g}_e = \|\mathbf{q}_u - \mathbf{q}_v\|_2^2, \text{ for all } e = (u, v) \in E.$$

Here, we think of k as a small parameter, obtained via a low-dimensional embedding.

We give efficient algorithms for finding structured approximations of \mathbf{g} in the following senses.

Definition 5 (SOC approximation). *Let $\alpha, \beta, \gamma > 0$, $m \in \mathbb{N}$, and $\mathbf{g} \in \mathbb{R}_{\geq 0}^E$. We say $\{\tilde{\mathbf{g}}^{(i)}\}_{i \in [m]} \subset \mathbb{R}_{\geq 0}^E$ is an $(\alpha, \beta, \gamma, m)$ -SOC approximation to \mathbf{g} if the following hold.*

- For all $i \in [m]$, $\tilde{\mathbf{g}}^{(i)}$ is a scalar multiple of a SOC.
- For $\mathbf{x} := \sum_{i \in [m]} \tilde{\mathbf{g}}^{(i)}$, we have $\mathbf{0}_E \leq \mathbf{x} \leq \alpha \mathbf{1}_E$ entrywise.
- For any $e \in E$ where $\mathbf{g}_e \leq 1$, $\mathbf{x}_e \geq \beta$.
- For any $e \in E$ where $\mathbf{g}_e > \gamma$, $\mathbf{x}_e = 0$.

Definition 6 (ASOC approximation). *Let $\alpha, \beta, \gamma > 0$ and for $\mathbf{Q} \in \mathbb{R}^{k \times n}$, let $\mathbf{g} \in \mathbb{R}_{\geq 0}^E$ be defined as*

$$\mathbf{g}_e = \|\mathbf{Q}_{:u} - \mathbf{Q}_{:v}\|_2^2 \text{ for all } e = (u, v) \in E.$$

We say $\{\tilde{\mathbf{g}}^{(i)}\}_{i \in [m]} \subset \mathbb{R}_{\geq 0}^E$ is a $(\alpha, \beta, \gamma, m)$ -ASOC approximation to \mathbf{g} if the following hold.

- For all $i \in [m]$, $\tilde{\mathbf{g}}^{(i)}$ is a scalar multiple of an ASOC.
- $\gamma \geq \max_{(u, v) \in E} \max_{j \in [k]} (\mathbf{Q}_{ju} - \mathbf{Q}_{jv})^2$.
- For all $i \in [m]$, $\tilde{\mathbf{g}}^{(i)} \leq \beta \mathbf{g}$ entrywise, and $\sum_{i \in [m]} \tilde{\mathbf{g}}^{(i)} \geq \mathbf{g}^{(\geq \frac{\gamma}{\alpha})}$ entrywise, where for $c > 0$,

$$\left[\mathbf{g}^{(\geq c)} \right]_e := \sum_{j \in [k]} (\mathbf{Q}_{ju} - \mathbf{Q}_{jv})^2 \mathbb{1}_{(\mathbf{Q}_{ju} - \mathbf{Q}_{jv})^2 \geq c}, \text{ for all } e = (u, v) \in E.$$

SOC approximation. We next describe our SOC approximation algorithm (Definition 5), which uses a standard grid rounding scheme provided below to form partitions of our point set.

Lemma 13. *Let $\{\mathbf{q}_i\}_{i \in [n]} \subset \mathbb{R}^k$ be given, and let $\rho > 0$. There is a randomized algorithm that in $O(nk \log(n))$ time returns a partition $\mathcal{P} = \{S_j\}_{j \in [\ell]}$ of $[n]$, satisfying the following properties.*

- *If $u, v \in S_j$ for some $j \in [\ell]$, then $\|\mathbf{q}_u - \mathbf{q}_v\|_2 \leq \rho\sqrt{k}$.*
- *Over the randomness of the algorithm, the probability that each pair $(u, v) \in [n] \times [n]$ do not belong to the same partition piece is at most $\frac{2\sqrt{k}}{\rho} \|\mathbf{q}_u - \mathbf{q}_v\|_2$.*

Proof. The algorithm works as follows: we discretize each dimension of \mathbb{R}^k into intervals of length ρ , where the interval endpoints are shifted by a random amount chosen uniformly in $[0, \rho)$. We then place two points in the same partition piece if and only if they lie in the same grid box in \mathbb{R}^k . It is clear that the distance bound of $\rho\sqrt{k}$ holds for points in the same grid box.

We now prove the second claim. First suppose that $\|\mathbf{q}_u - \mathbf{q}_v\|_2 \leq \frac{\rho}{2}$. Then by independence, the probability that \mathbf{q}_u and \mathbf{q}_v are in the same partition piece is the product of probabilities that they are not separated in any dimension, which we can lower bound by

$$\begin{aligned} \prod_{i \in [k]} \left(1 - \frac{|\mathbf{q}_u - \mathbf{q}_v|_i}{\rho}\right) &\geq \prod_{i \in [k]} \exp\left(-\frac{2|\mathbf{q}_u - \mathbf{q}_v|_i}{\rho}\right) \\ &= \exp\left(-\frac{2}{\rho} \|\mathbf{q}_u - \mathbf{q}_v\|_1\right) \\ &\geq 1 - \frac{2}{\rho} \|\mathbf{q}_u - \mathbf{q}_v\|_1 \geq 1 - \frac{2\sqrt{k}}{\rho} \|\mathbf{q}_u - \mathbf{q}_v\|_2. \end{aligned}$$

Thus they are separated with probability at most $\frac{2\sqrt{k}}{\rho} \|\mathbf{q}_u - \mathbf{q}_v\|_2$. In the other case of $\|\mathbf{q}_u - \mathbf{q}_v\|_2 > \frac{\rho}{2}$, the bound of $\frac{2\sqrt{k}}{\rho} \|\mathbf{q}_u - \mathbf{q}_v\|_2$ is trivial as it exceeds 1.

For the runtime, it is enough to first compute the index of each point, i.e., a k -tuple of integers specifying grid boxes in each dimension, in $O(nk)$ time. We can then find all unique grid boxes by sorting the indices in lexicographical order, which takes $O(nk \log(n))$ time. \square

Given this result, our SOC approximation follows straightforwardly.

Lemma 14. *Let $\{\mathbf{q}_i\}_{i \in [n]} \subset \mathbb{R}^k$, $\beta > 0$, $\delta \in (0, 1)$, and for all $e = (u, v) \in E$ suppose*

$$\mathbf{g}_e = \|\mathbf{q}_u - \mathbf{q}_v\|_2^2.$$

Then we can compute $\{\tilde{\mathbf{g}}^{(i)}\}_{i \in [m]}$, an $(\alpha, \beta, \gamma, m)$ -SOC approximation to \mathbf{g} , for

$$\alpha = \beta m, \quad \gamma = 16k^2, \quad m = \left\lceil 2 \log_2 \left(\frac{n}{\delta} \right) \right\rceil,$$

with probability $\geq 1 - \delta$ in time $O(nk \log(n) \log(\frac{n}{\delta}))$.

Proof. Let $\rho = \sqrt{\gamma/k}$. We repeatedly call Lemma 13 with the given value of ρ , for m times in total. For the i^{th} call, we let $\tilde{\mathbf{g}}^{(i)}$ be defined as follows: for all $(u, v) \in E$,

$$\tilde{\mathbf{g}}_{(u,v)}^{(i)} := \begin{cases} \beta & (u, v) \text{ belong to the same partition piece,} \\ 0 & \text{otherwise} \end{cases}. \quad (36)$$

Each $\tilde{\mathbf{g}}^{(i)}$ defined via (36) is a scalar multiple of a SOC. Also, because \mathbf{x} is a sum of m vectors from $\{0, \beta\}^E$, the bound $\alpha = \beta m$ clearly holds. This gives the first two conditions in Definition 5.

To obtain the third condition in Definition 5, consider some $e = (u, v) \in E$ such that $\mathbf{g}_e \leq 1$, so $\|\mathbf{q}_u - \mathbf{q}_v\|_2 \leq 1$. Then u and v belong to the same partition piece with probability at least

$$1 - \frac{2\sqrt{k}}{\rho} = 1 - \frac{2k}{\sqrt{\gamma}} \geq \frac{1}{2}.$$

Thus, repeating m times guarantees that some $\tilde{\mathbf{g}}_e^{(i)} = \beta$ except with probability $\frac{\delta}{n^2}$, and then union bounding over all edges in E gives the overall failure probability.

To obtain the last condition in Definition 5, by the first guarantee in Lemma 13, if some coordinate \mathbf{g}_e exceeds γ , i.e., $\|\mathbf{q}_u - \mathbf{q}_v\|_2 > \sqrt{\gamma} = \rho\sqrt{k}$ where $e = (u, v)$, then u and v will never be partitioned together and thus $\mathbf{x}_e = 0$ always. This does not contribute to the failure probability. \square

ASOC approximation. We now similarly give our ASOC approximation of distance vectors. We begin with a one-dimensional partitioning result.

Lemma 15. *Let $\{q_i\}_{i \in [n]} \subset \mathbb{R}$ be given, and let $\rho > 0$. There is an algorithm that in $O(n \log(n))$ time returns a set $S \subseteq [n]$, as well as a partition $\{S_j\}_{j \in [\ell]}$ of S , satisfying the following properties.*

1. *If $(u, v) \in [n] \times [n]$ have $|q_u - q_v| \in [\frac{3}{2}\rho, 2\rho]$, then with probability at least $\frac{1}{4}$, we have that $\{u, v\} \in S$, and u, v belong to different pieces of the partition.*
2. *If $(u, v) \in [n] \times [n]$ have $u \neq v$ and $|q_u - q_v| \leq \rho$, then either $\{u, v\} \subseteq S_j$ for some $j \in [\ell]$, or S contains at most one of u or v .*

Proof. The algorithm is similar to Lemma 13. We first tile \mathbb{R} with intervals of length ρ (with a random offset in $[0, \rho)$), and we color the intervals either ‘‘black’’ or ‘‘white’’ in alternating fashion. Then with probability $\frac{1}{2}$ we flip black and white intervals. We let S consist of the points in black intervals, and the partition pieces correspond to points falling in the same black interval. The runtime is immediate by sorting the list of interval indices. The second condition is also clear because if $|q_u - q_v| \leq \rho$, then the two points lie in either the same interval or adjacent intervals.

For the first condition, it is clearly enough to lower bound the probability that $\{u, v\} \in S$, because they cannot fall in the same piece of the partition if this is the case (as $|q_u - q_v| > \rho$). Without loss of generality, suppose that $q_u = 0$ and $q_v \in [\frac{3}{2}\rho, 2\rho]$. Let $s \in [0, \rho)$ be the uniformly random shift, so that $0 \in (s - \rho, s]$. With probability $\frac{1}{2}$, this is a black interval. Moreover,

$$\Pr_{s \sim \text{unif.}[0, \rho)} [q_v \in (s + \rho, s + 2\rho)] = \Pr_{s \sim \text{unif.}[0, \rho)} [s \in [q_v - 2\rho, q_v - \rho]] \geq \frac{1}{2}.$$

Thus, the probability that $(s - \rho, s]$ is black and q_v falls in the next black interval is at least $\frac{1}{4}$. \square

By repeatedly applying Lemma 15, we obtain an ASOC approximation when $k = 1$.

Lemma 16. *Let $\{q_i\}_{i \in [n]} \subset \mathbb{R}$ be given, let $\beta \geq 4$, $\gamma \geq \max_{(u,v) \in E} |q_u - q_v|^2$, $\alpha \geq 1$, $\delta \in (0, 1)$, and for all $e = (u, v) \in E$ suppose*

$$\mathbf{g}_e = (q_u - q_v)^2.$$

Then we can compute $\{\tilde{\mathbf{g}}^{(i)}\}_{i \in [m]}$, an $(\alpha, \beta, \gamma, m)$ -ASOC approximation to \mathbf{g} , for

$$m = O\left(\log\left(\frac{1}{\alpha}\right) \log\left(\frac{n}{\delta}\right)\right)$$

with probability $\geq 1 - \delta$ in time $O(n \log(n) \log(\frac{1}{\alpha}) \log(\frac{n}{\delta}))$.

Proof. First, let $\mathcal{P} = \{\rho_a\}_{a \in [p]} \subset \mathbb{R}_{>0}$ be a set of candidate distances, such that $p = O(\log(\frac{1}{\alpha}))$ and for every $\mathbf{g}_e \in [\frac{\gamma}{\alpha}, \gamma]$, there is some $a \in [p]$ such that $\sqrt{\mathbf{g}_e} \in [\frac{3}{2}\rho_a, 2\rho_a]$. This is doable by e.g., setting ρ_1^2 to be $\frac{4\gamma}{9\alpha}$, and then increasing ρ_a by multiples of 1.1 until $\rho_p^2 \geq \frac{\gamma}{4}$.

Next, for each $\rho_a \in \mathcal{P}$, we repeatedly call Lemma 15 with $\rho \leftarrow \rho_a$, for $\frac{m}{p} := \lceil 8 \log(\frac{n}{\delta}) \rceil$ times in total. Let $\{S_j\}_{j \in [\ell]}$ denote the partition pieces from the i^{th} call. Then we define for all $(u, v) \in E$,

$$\tilde{\mathbf{g}}_{(u,v)}^{(i,a)} := \begin{cases} \beta \rho_a^2 & \{u, v\} \subset S \text{ and } u \in S_j, v \in S_{j'} \text{ where } j \neq j' \\ 0 & \text{else} \end{cases}.$$

In other words, we assign a weight of $\beta \rho_a^2$ to all vertex pairs in different partition pieces, and zero out all other weights. We return the collection of $\{\tilde{\mathbf{g}}^{(i,a)}\}_{a \in [p], i \in [\frac{m}{p}]}$ as our ASOC approximation.

Then every $\tilde{\mathbf{g}}^{(i,a)}$ is a scalar multiple of an ASOC, and for all $i \in [m]$, $a \in [p]$, $\tilde{\mathbf{g}}^{(i,a)} \leq \beta \mathbf{g}$ edgewise, because the only way $\tilde{\mathbf{g}}_e^{(i,a)} > 0$ is if $\mathbf{g}_e > \rho_a^2$ by the second claim in Lemma 15.

Finally we need to show that with probability $\geq 1 - \delta$, we have that

$$\sum_{(i,a) \in [\frac{m}{p}] \times [p]} \tilde{\mathbf{g}}^{(i,a)} \geq \mathbf{g}^{(\geq \frac{\gamma}{\alpha})}. \quad (37)$$

We argue this edgewise. Let $e = (u, v) \in E$ have $\mathbf{g}_e \geq \frac{\gamma}{\alpha}$, and let a be the corresponding index satisfying $\sqrt{\mathbf{g}_e} \in [\frac{3}{2}\rho_a, 2\rho_a]$. By the first claim in Lemma 15, after running $\frac{m}{p}$ independent trials, there is at most a $\frac{\delta}{n^2}$ chance that u, v never fall in different black intervals. Moreover, if u, v fall in different black intervals in the i^{th} trial at distance ρ_a , then

$$\tilde{\mathbf{g}}_e^{(i,a)} = \beta \rho_a^2 \geq 4\rho_a^2 \geq \mathbf{g}_e.$$

Thus, by union bounding over all $|E| \leq n^2$ edges, (37) holds except with probability δ . \square

We can now apply Lemma 16 multiple times to generalize our result to distance vectors in \mathbb{R}^k .

Lemma 17. *Let $\{\mathbf{q}_i\}_{i \in [n]} \subset \mathbb{R}^k$, $\beta \geq 4$, $\delta \in (0, 1)$, and for all $e = (u, v) \in E$ suppose*

$$\mathbf{g}_e = \|\mathbf{q}_u - \mathbf{q}_v\|_2^2.$$

Also, let

$$\gamma \geq \max_{(u,v) \in E} \max_{j \in [k]} \left([\mathbf{q}_u]_j - [\mathbf{q}_v]_j \right)^2, \quad \alpha \geq 1.$$

Then we can compute $\{\tilde{\mathbf{g}}^{(i)}\}_{i \in [m]}$, an $(\alpha, \beta, \gamma, m)$ -ASOC approximation to \mathbf{g} , for

$$m = O \left(k \log \left(\frac{1}{\alpha} \right) \log \left(\frac{nk}{\delta} \right) \right),$$

with probability $\geq 1 - \delta$ in time $O(nk \log(n) \log(\frac{1}{\alpha}) \log(\frac{nk}{\delta}))$.

Proof. We apply Lemma 16 separately to each coordinate, adjusting the failure probability by a k factor, and return the union of the k resulting sets as our ASOC approximation. The condition that $\tilde{\mathbf{g}}^{(i)} \leq \beta \mathbf{g}$ coordinatewise always holds because $\mathbf{g}_{(u,v)} \geq |[\mathbf{q}_u]_j - [\mathbf{q}_v]_j|^2$ for any coordinate $j \in [k]$. Similarly, $\sum_{i \in [m]} \tilde{\mathbf{g}}^{(i)} \geq \mathbf{g}^{(\geq \frac{\gamma}{\alpha})}$ holds by summing over the contributions of each coordinate. \square

4.3 Long step packing SDP

In this section, we give an algorithm for approximately solving packing semidefinite programs. Our algorithm is inspired by existing counterparts in the literature [ZLO16, PTZ16, JLT20], but differs in a few ways. On the one hand, it only gives very coarse multiplicative approximation guarantees instead of $1 + \epsilon$ factors for arbitrarily small ϵ . On the other hand, it only uses highly-structured steps to update iterates, which support the efficient access tools from Sections 4.1 and 4.2.

Throughout this section, we consider the following optimization problem for semidefinite matrices $\{\mathbf{A}_e\}_{e \in E} \subset \mathbb{S}_{\geq 0}^{n \times n}$ for an index set E , and $\mathbf{c} \in \{0, 1\}^E$:

$$\max_{\substack{\mathbf{x} \in \mathbb{R}_{\geq 0}^E \\ \sum_{e \in E} \mathbf{x}_e \mathbf{A}_e \preceq \mathbf{I}_n}} \mathbf{c}^\top \mathbf{x}. \quad (38)$$

We are specifically interested in the specialization of (38) to

$$\mathbf{A}_e = \mathbf{P} \mathbf{L}_e \mathbf{P}, \text{ where } \mathbf{L}_e := (\mathbf{e}_u - \mathbf{e}_v) (\mathbf{e}_u - \mathbf{e}_v)^\top \text{ for all } e = (u, v) \in E, \quad (39)$$

where \mathbf{P} supports matrix-vector query access, and E indexes unordered $(u, v) \in [n] \times [n]$. Analogously to Section 4.1, we define $\mathcal{A} : \mathbb{R}^E \rightarrow \mathbb{S}^{n \times n}$ and $\mathcal{A}^* : \mathbb{S}^{n \times n} \rightarrow \mathbb{R}^E$ as

$$\mathcal{A}(\mathbf{v}) = \sum_{e \in E} \mathbf{v}_e \mathbf{A}_e \quad \text{and} \quad \mathcal{A}^*(\mathbf{Y}) = \{\langle \mathbf{A}_e, \mathbf{Y} \rangle\}_{e \in E}.$$

We split our discussion into three parts. We first propose an algorithm for solving the *decision variant* of (38), that guesses an optimal value and certifies whether it is approximately achievable. Our algorithm uses a certain step oracle (Definition 7), which we then discuss how to implement using Lemma 14. Finally, we reduce the optimization variant (38) to the decision variant.

Decision variant: correctness. In Algorithm 1, we state our general solver framework for the decision variant of (38). It uses the following type of oracle to implement its steps.

Definition 7 (Step oracle). *We say that $\mathcal{O} : \mathbb{S}_{\geq 0}^{n \times n} \rightarrow \mathbb{R}^E$ is an $(\alpha, \beta, \gamma, m)$ -step oracle for $\{\mathbf{A}_e\}_{e \in E}$ if on input $\mathbf{Z} \in \mathbb{S}_{\geq 0}^{n \times n}$, $\mathbf{x} = \mathcal{O}(\mathbf{Z})$ is a m -SOC and the following hold.*

- $\mathbf{0}_E \leq \mathbf{x} \leq \alpha \mathbf{1}_E$ entrywise.
- For any $e \in E$ where $[\mathcal{A}^*(\mathbf{Z})]_e \leq 1$, $\mathbf{x}_e \geq \beta$.
- For any $e \in E$ where $[\mathcal{A}^*(\mathbf{Z})]_e > \gamma$, $\mathbf{x}_e = 0$.

Algorithm 1: SOCpacking($\mathcal{O}, \mathbf{c}, p, T$)

```

1 Input:  $\mathcal{O}$ , an  $(\alpha, \beta, \gamma, m)$ -step oracle for  $\{\mathbf{A}_e\}_{e \in E}$ ,  $\mathbf{c} \in \{0, 1\}^E$ ,  $p \geq 2$ ,  $T \in \mathbb{N}$ 
2  $\mathbf{w}_0 \leftarrow \mathbf{c}$ 
3 for  $0 \leq t < T$  do
4    $\mathbf{Y}_t \leftarrow \frac{\mathcal{A}(\mathbf{w}_t)}{\|\mathcal{A}(\mathbf{w}_t)\|_p}$ 
5    $\boldsymbol{\delta}_t \leftarrow \mathcal{O}(\mathbf{Y}_t^{p-1})$ 
6    $\mathbf{w}_{t+1} = \mathbf{w}_t \circ (\mathbf{1}_E + \boldsymbol{\delta}_t)$ 
7   if  $\mathbf{c}^\top \mathbf{w}_{t+1} \geq \beta^{\frac{T}{2}}$  then
8     Return:  $\frac{\mathbf{w}_{t+1}}{\mathbf{c}^\top \mathbf{w}_{t+1}}$ 
9   end
10 end

```

We proceed to analyze the correctness of Algorithm 1, discussing how to implement \mathcal{O} later in this section. We begin with a helper lemma.

Lemma 18. *For any $p \geq 2$ and $\mathbf{M} \in \mathbb{S}_{\geq 0}^{n \times n}$ with $\mathbf{M} \preceq \alpha \mathbf{I}_n$, $(\mathbf{I}_n + \mathbf{M})^p \preceq \mathbf{I}_n + p(1 + \alpha)^{p-1} \mathbf{M}$.*

Proof. Because \mathbf{I}_n commutes with \mathbf{M} , we may diagonalize both sides; the claim reduces to showing $(1 + x)^p \leq 1 + p(1 + \alpha)^{p-1}x$ for $x \in [0, \alpha]$. This follows from the intermediate value theorem. \square

We now make a simple observation on the sparsity pattern of the iterates \mathbf{w}_t in Algorithm 1.

Lemma 19. *For any $0 \leq t < T$ and $e \in E$, $[\mathbf{w}_t]_e$ is nonzero if and only if \mathbf{c}_e is.*

Proof. We proceed by induction on t : the base case is trivial since $\mathbf{w}_0 = \mathbf{c}$. For any other iteration t , observe $[\mathbf{w}_{t+1}]_e = [\mathbf{w}_t]_e(1 + [\boldsymbol{\delta}_t]_e)$ for all $e \in E$, and since $\boldsymbol{\delta}_t \geq 0$ the inductive claim is obvious. \square

With this, we show that a certain potential function is nonincreasing throughout the algorithm.

Lemma 20. *In the setting of Algorithm 1, define the function*

$$\Phi_t = \|\mathcal{A}(\mathbf{w}_t)\|_p - (1 + \alpha)^{p-1} \gamma \mathbf{c}^\top \mathbf{w}_t, \text{ for all } 0 \leq t < T.$$

Then for any $0 \leq t < T$, we have $\Phi_{t+1} \leq \Phi_t$.

Proof. For simplicity, we drop the index t and let $\mathbf{w} := \mathbf{w}_t$, $\mathbf{Y} := \mathbf{Y}_t$. Define the matrices

$$\mathbf{M}_0 := \mathcal{A}(\mathbf{w}), \quad \mathbf{M}_1 := \mathcal{A}(\boldsymbol{\delta} \circ \mathbf{w}).$$

By the Lieb-Thirring inequality, i.e., $\text{Tr}((\mathbf{ABA})^p) \leq \text{Tr}(\mathbf{A}^{2p}\mathbf{B}^p)$, we have

$$\|\mathbf{M}_0 + \mathbf{M}_1\|_p^p = \text{Tr}((\mathbf{M}_0 + \mathbf{M}_1)^p) \leq \text{Tr}\left(\mathbf{M}_0^p \left(\mathbf{I}_n + \mathbf{M}_0^{-\frac{1}{2}}\mathbf{M}_1\mathbf{M}_0^{-\frac{1}{2}}\right)^p\right).$$

Next, noting that $\boldsymbol{\delta} \leq \alpha \mathbf{1}_E$ by definition of the step oracle, applying Lemma 18 gives

$$\|\mathbf{M}_0 + \mathbf{M}_1\|_p^p \leq \text{Tr}\left(\mathbf{M}_0^p + p(1 + \alpha)^{p-1}\mathbf{M}_0^{p-1}\mathbf{M}_1\right).$$

If we define $\mathbf{Y} = \frac{\mathbf{M}_0}{\|\mathbf{M}_0\|_p}$, we observe

$$\begin{aligned} \|\mathbf{M}_0 + \mathbf{M}_1\|_p^p &\leq \text{Tr}\left(\mathbf{M}_0^p + p(1 + \alpha)^{p-1}\mathbf{M}_0^{p-1}\mathbf{M}_1\right) \\ &= \|\mathbf{M}_0\|_p^p \left(1 + p(1 + \alpha)^{p-1} \left\langle \mathbf{Y}^{p-1}, \frac{\mathbf{M}_1}{\|\mathbf{M}_0\|_p} \right\rangle\right). \end{aligned}$$

By using $1 + px \leq (1 + x)^p$ for all $p \geq 2$ and $x \geq 0$, taking p^{th} roots we thus have

$$\|\mathbf{M}_0 + \mathbf{M}_1\|_p \leq \|\mathbf{M}_0\|_p + (1 + \alpha)^{p-1} \langle \mathbf{Y}^{p-1}, \mathbf{M}_1 \rangle.$$

Now via linearity of trace,

$$\langle \mathbf{Y}^{p-1}, \mathbf{M}_1 \rangle = \sum_{e \in E} \langle \mathbf{Y}^{p-1}, \mathbf{A}_e \rangle \boldsymbol{\delta}_e \mathbf{w}_e \leq \sum_{e \in E} \gamma \boldsymbol{\delta}_e \mathbf{w}_e \quad (40)$$

as $\langle \mathbf{Y}^{p-1}, \mathbf{A}_j \rangle > \gamma$ implies $\boldsymbol{\delta}_e = 0$. We now claim

$$\sum_{e \in E} \boldsymbol{\delta}_e \mathbf{w}_e = \sum_{e \in E} \mathbf{c}_e \boldsymbol{\delta}_e \mathbf{w}_e.$$

This follows immediately from the observation that $\mathbf{w}_e \neq 0$ implies $\mathbf{c}_e = 1$ by Lemma 19 and the fact that all $\mathbf{c}_e \in \{0, 1\}$. Combining this with (40) gives the desired result. \square

We augment this potential bound by showing that after T steps of the algorithm we must be able to certify either a primal or dual bound for the quality of a packing solution.

Lemma 21. *Let $p \geq 2$ and $q := \frac{p}{p-1}$. After T steps of Algorithm 1, define $\bar{\mathbf{Y}} = \frac{1}{T} \sum_{0 \leq t < T} \mathbf{Y}_t^{p-1}$ and $\mathbf{x} = \frac{\mathbf{w}_T}{\mathbf{c}^\top \mathbf{w}_T}$. Then at least one of the following is true.*

- $\mathcal{A}^*(\bar{\mathbf{Y}}) \geq \frac{1}{2} \mathbf{c}$ and $\|\bar{\mathbf{Y}}\|_q \leq 1$.
- The algorithm terminates on Line 8 in some iteration.

Proof. First we observe that all $\|\mathbf{Y}_t\|_q = 1$ by inspection, so by convexity of norms, we indeed have $\|\bar{\mathbf{Y}}\|_q \leq 1$. Similarly, we have $\mathbf{c}^\top \mathbf{x} = 1$ by definition.

Assume that the first claim is false, and there exists some $e \in E$ such that $\langle \mathbf{A}_e, \bar{\mathbf{Y}} \rangle < \frac{1}{2}\mathbf{c}_e$. We show that the second claim must hold. Because $\langle \mathbf{A}_e, \mathbf{Y}_j^{p-1} \rangle \geq 0$ for any $e \in E, 0 \leq t < T$,

$$0 \leq \langle \mathbf{A}_e, \bar{\mathbf{Y}} \rangle < \frac{1}{2}\mathbf{c}_e,$$

and thus $\mathbf{c}_e = 1$ since $\mathbf{c} \in \{0, 1\}^E$. Applying Markov's inequality, we have that at least $\frac{T}{2}$ iterations $0 \leq t < T$ have $\langle \mathbf{A}_e, \mathbf{Y}_t^{p-1} \rangle \leq \mathbf{c}_e$ by Markov's inequality. By the guarantee of \mathcal{O} , every such step t must grow $[\mathbf{w}_t]_e$ by a factor of β since $\langle \mathbf{A}_e, \mathbf{Y}_t^{p-1} \rangle = \mathcal{A}^*(\mathbf{Y}_t^{p-1})_e = \mathbf{g}_e$. With this, we have

$$\mathbf{c}^\top \mathbf{w}_T \geq \mathbf{c}_e [\mathbf{w}_T]_e \geq \beta^{\frac{T}{2}} \mathbf{c}_e [\mathbf{w}_0]_e = \beta^{\frac{T}{2}},$$

so if the algorithm was allowed to run for T iterations, it must terminate as claimed. \square

We provide an alternative characterization of the first case in Lemma 21 to help apply the result.

Lemma 22. *Let $p \geq 2$ and $q := \frac{p}{p-1}$. If there exists $\mathbf{Z} \in \mathbb{S}_{\geq 0}^{n \times n}$ with $\|\mathbf{Z}\|_q \leq 1$ and $\mathcal{A}^*(\mathbf{Z}) \geq \frac{1}{2}\mathbf{c}$,*

$$\max_{\substack{\mathbf{x} \in \mathbb{R}_{\geq 0}^E \\ \mathcal{A}(\mathbf{x}) \preceq \mathbf{I}_n}} \mathbf{c}^\top \mathbf{x} \leq 2n^{\frac{1}{p}}.$$

Proof. Assume $\|\mathbf{Z}\|_q \leq 1$ and $\mathcal{A}^*(\mathbf{Z}) \geq \frac{1}{2}\mathbf{c}$, and suppose that there exists some $\mathbf{x} \in \mathbb{R}_{\geq 0}^E$ satisfying $\mathbf{c}^\top \mathbf{x} > 2$ and $\|\mathcal{A}(\mathbf{x})\|_p \leq 1$. Then by Hölder's inequality, we have a contradiction:

$$1 < \frac{1}{2} \langle \mathbf{x}, \mathbf{c} \rangle \leq \langle \mathbf{x}, \mathcal{A}^*(\mathbf{Z}) \rangle = \langle \mathbf{Z}, \mathcal{A}(\mathbf{x}) \rangle \leq \|\mathcal{A}(\mathbf{x})\|_p \leq 1.$$

Thus, all $\mathbf{x} \in \mathbb{R}_{\geq 0}^E$ with $\|\mathcal{A}(\mathbf{x})\|_p \leq 1$ must have $\mathbf{c}^\top \mathbf{x} \leq 2$. This implies the conclusion upon performing a norm conversion from $\|\cdot\|_p$ to $\|\cdot\|_\infty$ in the constraint, losing a $n^{\frac{1}{p}}$ factor. \square

Decision variant: implementation. We now turn to implementation. We begin by providing an efficient step oracle \mathcal{O} for the matrices \mathbf{Y}_t^{p-1} in Line 5. It will first be helpful to collapse our true oracle input to a smaller dimension. To motivate this, Algorithm 1 requires approximating

$$\langle \mathbf{Y}^{p-1}, \mathbf{P} \mathbf{L}_e \mathbf{P} \rangle = \left\| \mathbf{Y}^{\frac{p-1}{2}} \mathbf{P} (\mathbf{e}_u - \mathbf{e}_v) \right\|_2^2, \quad (41)$$

for all $e = (u, v) \in E$. This makes it clear that this is a squared distance between columns of $\mathbf{Y}^{\frac{p-1}{2}} \mathbf{P}$, which are natively n -dimensional vectors that we can only implicitly access. Using the Johnson-Lindenstrauss lemma, we can embed our vectors into $k \approx \log(n)$ dimensions at a constant factor approximation loss, which drastically improves parameters when applying tools from Section 4.2.

Lemma 23. *Following notation in Algorithm 1, let $\delta \in (0, 1)$, suppose that $p \geq 2$ is an odd integer, and let $\mathbf{Y} = \frac{\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P}}{\|\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P}\|_p}$ for $\mathbf{w} \in \mathbb{R}^E$. In time*

$$O\left((\mathcal{T}_{\text{mv}}(\mathbf{P}) + \mathcal{T}_{\text{mv}}(\mathcal{L}(\mathbf{w}))) \cdot p \log\left(\frac{n}{\delta}\right)\right)$$

we can output $\mathbf{Q} \in \mathbb{R}^{k \times n}$ where $k = O(\log(\frac{n}{\delta}))$, such that if we define $\mathbf{g}'_e := \|\mathbf{Q}_{:u} - \mathbf{Q}_{:v}\|_2^2$ for all $e = (u, v) \in E$, we have with probability $\geq 1 - \delta$ that

$$\frac{1}{2}\mathcal{A}^*(\mathbf{Y}^{p-1}) \leq \mathbf{g}' \leq \mathcal{A}^*(\mathbf{Y}^{p-1}) \text{ entrywise.}$$

Proof. First, we claim that we can output a number Z such that

$$Z \leq \|\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P}\|_p^p = \text{Tr}((\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P})^p) \leq 1.1Z,$$

within the allotted time, with failure probability $\leq \frac{\delta}{2}$. This is immediate from the Johnson-Lindenstrauss lemma, see e.g., its application in Lemma 2 of [JLM⁺23]. This also implies that

$$Z^{\frac{1}{2} - \frac{1}{2p}} \leq \|\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P}\|_p^{\frac{p-1}{2}} \leq 1.1Z^{\frac{1}{2} - \frac{1}{2p}}.$$

Next, by the Johnson-Lindenstrauss lemma (see e.g., Lemma 3 of [JLM⁺23]), letting $k = O(\log(\frac{n}{\delta}))$ with a sufficiently large constant, and $\mathbf{G} \in \mathbb{R}^{k \times n}$ have i.i.d. entries $\sim \mathcal{N}(0, \frac{1}{k})$, we have that

$$\begin{aligned} 0.9 \left\| (\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P})^{\frac{p-1}{2}} \mathbf{P}(\mathbf{e}_u - \mathbf{e}_v) \right\|_2^2 &\leq \left\| \mathbf{G} (\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P})^{\frac{p-1}{2}} \mathbf{P}(\mathbf{e}_u - \mathbf{e}_v) \right\|_2^2 \\ &\leq 1.1 \left\| (\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P})^{\frac{p-1}{2}} \mathbf{P}(\mathbf{e}_u - \mathbf{e}_v) \right\|_2^2, \end{aligned}$$

with probability $\geq 1 - \frac{\delta}{2}$, for all $e = (u, v) \in E$. Thus combining our claims and recalling (41), we have that with probability $\geq 1 - \delta$, it is enough to output $\mathbf{Q} = (1.1)^{-\frac{1}{2}} Z^{\frac{1}{2p} - \frac{1}{2}} \mathbf{G} (\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P})^{\frac{p-1}{2}}$, which takes the stated amount of time, as it uses $O(pk)$ matrix-vector products through $\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P}$. \square

We can now directly apply Lemma 14 to implement our step oracle.

Lemma 24. *Following notation in Algorithm 1, let $\beta > 0$, $\delta \in (0, 1)$, suppose that $p \geq 2$ is an odd integer, and let $\mathbf{Y} = \frac{\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P}}{\|\mathbf{P}\mathcal{L}(\mathbf{w})\mathbf{P}\|_p}$ for $\mathbf{w} \in \mathbb{R}^E$. We can implement \mathcal{O} , an $(\alpha, \beta, \gamma, m)$ -step oracle for \mathbf{Y}^{p-1} , for*

$$\alpha = 2\beta m, \gamma = O\left(\log^2\left(\frac{n}{\delta}\right)\right), m = O\left(\log\left(\frac{n}{\delta}\right)\right),$$

with probability $\geq 1 - \delta$ in time

$$O\left((\mathcal{T}_{\text{mv}}(\mathbf{P}) + \mathcal{T}_{\text{mv}}(\mathcal{L}(\mathbf{w}))) \cdot p \log\left(\frac{n}{\delta}\right) + n \log(n) \log^2\left(\frac{n}{\delta}\right)\right).$$

Proof. We apply Lemma 14 (with $\delta \leftarrow \frac{\delta}{2}$) to \mathbf{Q} where \mathbf{Q} is the output of Lemma 23 (with $\delta \leftarrow \frac{\delta}{2}$). The resulting oracle parameters at most grow by a factor of 2 due to the lossiness of Lemma 23. \square

We next show inductively that our iterates $\{\mathbf{w}_t\}_{0 \leq t < T}$ induce Laplacians $\mathcal{L}(\mathbf{w}_t)$ that support efficient matrix-vector queries by using tools for K -SOCs from Section 4.1.

Lemma 25. *Following notation in Algorithm 1, for each $0 \leq t < T$, $\mathbf{w}_t = \mathbf{y}_t \circ \mathbf{c}$ where \mathbf{y}_t is a $(m+1)^t$ -SOC where $m = O(\log(\frac{n}{\delta}))$. Moreover, we can compute \mathbf{y}_t in $O(n \log(n)(m+1)^t)$ time.*

Proof. We proceed by induction on t . For the base case, $\mathbf{w}_0 = \mathbf{c} = \mathbf{1}_E \circ \mathbf{c}$ and $\mathbf{1}_E$ is a 1-SOC. Next, for some $t \geq 0$, assume that \mathbf{w}_t can be written as

$$\sum_{i \in [(m+1)^t]} \mathbf{v}_i \circ \mathbf{c},$$

where each \mathbf{v}_i is a 1-SOC. Further, note that by the step oracle guarantee, each $\delta_t = \sum_{j \in [m]} \mathbf{w}_j$ is always a m -SOC, where each \mathbf{w}_j is a 1-SOC. This proves the inductive guarantee, as

$$\left(\sum_{i \in [(m+1)^t]} \mathbf{v}_i \circ \mathbf{c} \right) \circ \left(\mathbf{1}_E + \sum_{j \in [m]} \mathbf{w}_j \right) = \sum_{i \in [(m+1)^t]} \mathbf{v}_i \circ \mathbf{c} + \sum_{\substack{i \in [(m+1)^t] \\ j \in [m]}} \mathbf{v}_i \circ \mathbf{w}_j \circ \mathbf{c},$$

and the latter expression clearly has $(m+1)^{t+1}$ terms, each of which is either $\mathbf{v}_i \circ \mathbf{c}$ (where \mathbf{v}_i is a 1-SOC), or $\mathbf{v}_i \circ \mathbf{w}_j \circ \mathbf{c}$ (where $\mathbf{v}_i \circ \mathbf{w}_j$ is a 1-SOC because of Lemma 8). Finally, note that we can compute each $\mathbf{v}_i \circ \mathbf{w}_j$ as a 1-SOC using Lemma 8 in time $O(n \log(n))$. \square

We can now put together all the pieces developed thus far to give our analysis for Algorithm 1.

Lemma 26. *In the setting of Algorithm 1, suppose $\|\mathcal{A}(\mathbf{c})\|_{\text{op}} \leq \rho$ for $\rho \geq 1$, let $\delta \in (0, 1)$, and let*

$$p = \Theta\left(\log^{\frac{1}{3}}(n\rho)\right), \quad T = \Theta\left(\log^{\frac{2}{3}}(n\rho)\right),$$

where p is an odd integer. We have with probability $\geq 1 - \delta$ that if Algorithm 1 returns on Line 8,

$$\max_{\substack{\mathbf{x} \in \mathbb{R}_{\geq 0}^E \\ \mathcal{A}(\mathbf{x}) \preceq \mathbf{I}_n}} \mathbf{c}^\top \mathbf{x} \geq \exp\left(-O\left(\log^{\frac{2}{3}}(n\rho) \log \log\left(\frac{n\rho}{\delta}\right)\right)\right), \quad (42)$$

and otherwise,

$$\max_{\substack{\mathbf{x} \in \mathbb{R}_{\geq 0}^E \\ \mathcal{A}(\mathbf{x}) \preceq \mathbf{I}_n}} \mathbf{c}^\top \mathbf{x} \leq \exp\left(O\left(\log^{\frac{2}{3}}(n\rho)\right)\right), \quad (43)$$

for appropriate constants. The algorithm can be implemented to run in time

$$O\left(\mathcal{T}_{\text{mv}}(\mathbf{P}) \cdot \log^{\frac{5}{3}}\left(\frac{n\rho}{\delta}\right) + n \log(n) \log\left(\frac{n\rho}{\delta}\right)^{O(\log^{2/3}(n\rho))}\right),$$

and if it returns $\mathbf{x} = \frac{\mathbf{w}_{t+1}}{\mathbf{c}^\top \mathbf{w}_{t+1}}$ on Line 8, then $\mathbf{x} = \mathbf{c} \circ \mathbf{y}$ where \mathbf{y} is a $O(\log(\frac{nT}{\delta}))^T$ -SOC.

Proof. By induction, Lemma 25 shows that we can maintain each \mathbf{w}_t within the allotted time as a $(m+1)^T$ -SOC, for some $m = O(\log(\frac{nT}{\delta}))$ (where we adjusted the failure probability by a $O(T)$ factor). Thus, Lemma 24 shows we can obtain an $(\alpha, \beta, \gamma, m)$ -step oracle to implement Line 5 in all T iterations in the stated time, with failure probability $\frac{\delta}{2}$, and parameters

$$\beta = \exp\left(\Theta\left(\log^{\frac{1}{3}}(n\rho)\right)\right), \quad \alpha = 2\beta m, \quad \gamma = O\left(\log^2\left(\frac{nT}{\delta}\right)\right), \quad m = O\left(\log\left(\frac{nT}{\delta}\right)\right).$$

In the remainder of the proof, assume that all calls on Lemma 24 succeeded. It only remains to establish correctness of (42) and (43). First, if the algorithm ever successfully returns on Line 8 for some iteration $t + 1$, we have by inducting on Lemma 20 that for $\mathbf{x} := \frac{\mathbf{w}_{t+1}}{\mathbf{c}^\top \mathbf{w}_{t+1}}$,

$$\begin{aligned}\|\mathcal{A}(\mathbf{x})\|_{\text{op}} &\leq \frac{1}{\mathbf{c}^\top \mathbf{w}_{t+1}} \|\mathcal{A}(\mathbf{w}_{t+1})\|_p = \frac{1}{\mathbf{c}^\top \mathbf{w}_{t+1}} \Phi_{t+1} + (1 + \alpha)^{p-1} \gamma \\ &\leq \frac{1}{\beta^{\frac{p}{2}}} \Phi_0 + \exp\left(O\left(\log^{\frac{2}{3}}(n\rho) \log \log\left(\frac{n\rho}{\delta}\right)\right)\right) = \exp\left(O\left(\log^{\frac{2}{3}}(n\rho) \log \log\left(\frac{n\rho}{\delta}\right)\right)\right),\end{aligned}$$

once we plug in our choice of parameters. Using that $\mathbf{c}^\top \mathbf{x} = 1$ and normalizing by the final expression above proves (42). Conversely, if the algorithm never returns on Line 8, then (43) follows from our choice of parameters, the first case in Lemma 21, and the characterization in Lemma 22. \square

Optimization variant. Lemma 26 lets us certify whether the value of a problem (38) is very large or very small, at a given scale. We complete this section by wrapping this decision problem result in a binary search to solve the optimization variant of (38).

Proposition 6. *In an instance of (38) where (39) holds, let $\delta \in (0, 1)$, let OPT denote the value of (38), and suppose we know $\text{OPT} \in [\ell, u] \subset \mathbb{R}_{>0}$. There is an algorithm running in time*

$$O\left(\mathcal{T}_{\text{mv}}(\mathbf{P}) \cdot \log^2\left(\frac{nu}{\delta\ell}\right) + n \exp\left(O\left(\log^{\frac{2}{3}}\left(\frac{nu}{\ell}\right) \log \log\left(\frac{nu}{\delta\ell}\right)\right)\right)\right).$$

that returns \mathbf{x} satisfying

$$\langle \mathbf{c}, \mathbf{x} \rangle \geq \exp\left(-O\left(\log^{\frac{2}{3}}\left(\frac{nu}{\ell}\right) \log \log\left(\frac{nu}{\delta\ell}\right)\right)\right) \text{OPT}, \quad \mathcal{A}(\mathbf{x}) \preceq \mathbf{I}_n.$$

Moreover, $\mathbf{x} = \mathbf{v} \circ \mathbf{c}$ where \mathbf{v} is a K -SOC, for

$$K = \exp\left(O\left(\log^{\frac{2}{3}}\left(\frac{nu}{\ell}\right) \log \log\left(\frac{nu}{\delta\ell}\right)\right)\right).$$

Proof. The proof follows by combining Lemma 26 with the binary search in Proposition 5 of [JLM⁺23]. In particular, there are $\log \log(\frac{u}{\ell})$ phases of binary search, each calling Lemma 26 once with a multiple of \mathbf{c} . We remark that we never need to pass any scaling that does not satisfy the premise $\|\mathcal{A}(\mathbf{c})\|_{\text{op}} \geq 1$, which we can certify up to a constant factor via the power method. This is because if $\|\mathcal{A}(\mathbf{c})\|_{\text{op}} \leq 1$, then (42) immediately holds by using the choice $\mathbf{x} = \mathbf{c}$. By the same logic, no scaling considered will ever have $\rho \geq \frac{u}{\ell}$, because we only ever truncate the range of interest. \square

4.4 Matrix dictionary recovery

In this section, we build upon Section 4.3 to give a two-sided matrix dictionary recovery result. Our result approximates an unknown graph Laplacian, given very weak access in the forms of matrix-vector products and a preconditioned packing oracle (with one-sided guarantees).

JL embedding. Our algorithm is based on the matrix multiplicative weights updates, which produces iterates of the form

$$\mathbf{Y} = \frac{\exp(-\mathbf{S})}{\text{Tr} \exp(-\mathbf{S})}. \tag{44}$$

In our setting, the matrix \mathbf{S} has a special structure. Assume that we have matrix-vector product access to $\mathbf{P} \in \mathbb{S}_{\geq 0}^{n \times n}$, and let $\{\mathbf{a}_s, \mathbf{v}_s\}_{s \in [S]} \in \mathbb{R}^E$ be such that all of the $\{\mathbf{a}_s\}_{s \in [S]}$ are ASOCs, and all of the $\{\mathbf{v}_s\}_{s \in [S]}$ are K -SOCs. We are interested in supporting access to \mathbf{Y} in (44), for

$$\mathbf{S} = \mathbf{P}\mathcal{L}(\mathbf{x})\mathbf{P}, \text{ where } \mathbf{x} = \sum_{s \in [S]} \mathbf{a}_s \circ \mathbf{v}_s. \quad (45)$$

To simplify notation, let $\tilde{\mathbf{Y}} := \mathbf{P}\mathbf{Y}\mathbf{P}$. Our algorithm requires applying the techniques of Section 4.2 to the vector $\mathbf{g} \in \mathbb{R}^E$ defined by

$$\mathbf{g}_e = \langle \mathbf{Y}, \mathbf{PL}_e\mathbf{P} \rangle = \langle \tilde{\mathbf{Y}}, \mathbf{L}_e \rangle \text{ for all } e = (u, v) \in E. \quad (46)$$

As in Section 4.3, we will treat each \mathbf{g}_e as a squared distance between columns of $\tilde{\mathbf{Y}}^{\frac{1}{2}}$, which we can embed into low dimensions. We first require the following guarantee on approximating $\text{Tr} \exp(-\mathbf{S})$.

Lemma 27. *Let $\mathbf{x} \in \mathbb{R}^E$ have the form (45) where all $\{\mathbf{a}_s\}_{s \in [S]}$ are ASOCs and all $\{\mathbf{v}_s\}_{s \in [S]}$ are K -SOCs, and define \mathbf{S}, \mathbf{Y} as in (45), (44). Assume $\|\mathbf{S}\|_{\text{op}} \leq R$, and let $\delta \in (0, 1)$. In time*

$$O\left((\mathcal{T}_{\text{mv}}(\mathbf{P}) + nKS \log(n)) R \log\left(\frac{n}{\delta}\right)\right),$$

we can output $Z \in \mathbb{R}$ satisfying $\frac{9}{10} \text{Tr} \exp(-\mathbf{S}) \leq Z \leq \text{Tr} \exp(-\mathbf{S})$ with probability $\geq 1 - \delta$.

Proof. This follows from Lemma 2 of [JLM⁺23] with $\epsilon \leftarrow \frac{1}{30}$, $\kappa \leftarrow 1$, and $R \leftarrow R$, because $\mathcal{T}_{\text{mv}}(\mathbf{S}) = 2\mathcal{T}_{\text{mv}}(\mathbf{P}) + \mathcal{T}_{\text{mv}}(\mathcal{L}(\mathbf{x}))$, and $\mathcal{T}_{\text{mv}}(\mathcal{L}(\mathbf{x})) = O(nKT \log(n))$ using (45) and Lemma 9. \square

We now give our main embedding result, which approximates \mathbf{g} entrywise in low dimensions.

Lemma 28. *Let $\mathbf{x} \in \mathbb{R}^E$ have the form (45) where all $\{\mathbf{a}_s\}_{s \in [S]}$ are ASOCs and all $\{\mathbf{v}_s\}_{s \in [S]}$ are K -SOCs, and define $\mathbf{S}, \mathbf{Y}, \mathbf{g}$ as in (45), (44), (46). Assume $\|\mathbf{S}\|_{\text{op}} \leq R$, and let $\delta \in (0, 1)$. In time*

$$O\left((\mathcal{T}_{\text{mv}}(\mathbf{P}) + nKS \log(n)) R \log\left(\frac{n}{\delta}\right)\right),$$

we can output $\mathbf{Q} \in \mathbb{R}^{k \times n}$ where $k = O(\log(\frac{n}{\delta}))$, such that if we define $\mathbf{f}_e := \|\mathbf{Q}_{:u} - \mathbf{Q}_{:v}\|_2^2$ for all $e = (u, v) \in E$, we have with probability $\geq 1 - \delta$ that $\frac{1}{2}\mathbf{g} \leq \mathbf{f} \leq \mathbf{g}$ entrywise.

Proof. As our first step, we obtain an estimate satisfying $\frac{9}{10} \text{Tr} \exp(-\mathbf{S}) \leq Z \leq \text{Tr} \exp(-\mathbf{S})$ with probability $\geq 1 - \frac{\delta}{2}$ from Lemma 27, within the allotted runtime.

Our next step is to approximate $\exp(-\mathbf{S})$ with a polynomial in \mathbf{S} . Specifically, Theorem 4.1 of [SV14] shows that because $\mathbf{0}_{n \times n} \preceq \mathbf{S} \preceq R\mathbf{I}_n$, there is a matrix \mathbf{M} such that

$$\frac{9}{10}\mathbf{M} \preceq \exp\left(-\frac{1}{2}\mathbf{S}\right) \preceq \mathbf{M}, \quad (47)$$

and \mathbf{M} is a degree- $O(R)$ polynomial in \mathbf{S} . Because \mathbf{M}, \mathbf{S} commute, this also shows that

$$\frac{4}{5}\mathbf{M}^2 \preceq \exp(-\mathbf{S}) \preceq \mathbf{M}^2.$$

Combining these pieces shows that

$$\frac{3}{4} \langle \tilde{\mathbf{Y}}, \mathbf{L}_e \rangle \leq \frac{1}{Z} \|\mathbf{MP}(\mathbf{e}_u - \mathbf{e}_v)\|_2^2 \leq \frac{5}{4} \langle \tilde{\mathbf{Y}}, \mathbf{L}_e \rangle, \text{ for all } e = (u, v) \in E. \quad (48)$$

Next, by the Johnson-Lindenstrauss lemma, letting $k = O(\log(\frac{n}{\delta}))$ for an appropriately large constant, and letting $\mathbf{G} \in \mathbb{R}^{k \times n}$ have i.i.d. entries $\sim \mathcal{N}(0, \frac{1}{k})$, we have

$$\frac{2}{3} \langle \tilde{\mathbf{Y}}, \mathbf{L}_e \rangle \leq \frac{1}{Z} \|\mathbf{GMP}(\mathbf{e}_u - \mathbf{e}_v)\|_2^2 \leq \frac{4}{3} \langle \tilde{\mathbf{Y}}, \mathbf{L}_e \rangle, \text{ for all } e = (u, v) \in E, \quad (49)$$

with probability $\geq 1 - \frac{\delta}{2}$. This implies that to get $\frac{1}{2}\mathbf{g} \leq \mathbf{f} \leq \mathbf{g}$, it is enough to set

$$\mathbf{Q} = \sqrt{\frac{3}{4Z}} \mathbf{GMP},$$

which takes the stated time to compute. We can see this because outputting \mathbf{Q} requires applying \mathbf{M} and \mathbf{P} to k different vectors, and $\mathcal{T}_{\text{mv}}(\mathbf{M}) = O(R \cdot \mathcal{T}_{\text{mv}}(\mathbf{S}))$. \square

Preconditioned matrix dictionary recovery. We next give a coarse approximation to a Laplacian using a method inspired by [JLM⁺23]. Let $\mathbf{L} \in \mathbb{S}_{\geq 0}^{n \times n}$ be an unknown graph Laplacian, and assume that we have matrix-vector query access to $\mathbf{P} \in \mathbb{S}_{\geq 0}^{n \times n}$ satisfying

$$\mathbf{L}^\dagger \preceq \mathbf{P}^2 \preceq 2\mathbf{L}^\dagger. \quad (50)$$

We let $\mathbf{\Pi} := \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$ be the graph Laplacian corresponding to the clique on $[n]$, as in Theorem 2. We also require the following notion of a *SOC packing oracle*.

Definition 8. Let $\mathbf{L} \in \mathbb{S}_{\geq 0}^{n \times n}$ be an unknown graph Laplacian, and let $Q \geq 1$, $K \in \mathbb{N}$, $\delta \in (0, 1)$. We say $\mathcal{O} : \mathbb{R}_{\geq 0}^E \times \mathbb{S}_{\geq 0}^{n \times n} \rightarrow \mathbb{R}_{\geq 0}^E$ is a (Q, K, δ) -SOC packing oracle for \mathbf{L} if on inputs $\mathbf{g} \in \mathbb{R}_{\geq 0}^E$ and $\mathbf{P} \in \mathbb{S}_{\geq 0}^{n \times n}$ satisfying (50), with probability $\geq 1 - \delta$, \mathcal{O} returns $\mathbf{x} \in \mathbb{R}_{\geq 0}^E$ satisfying

$$\mathbf{x} \in \mathcal{F} \text{ and } \langle \mathbf{g}, \mathbf{x} \rangle \geq \frac{1}{Q} \left(\max_{\mathbf{y} \in \mathcal{F}} \langle \mathbf{g}, \mathbf{y} \rangle \right), \text{ where } \mathcal{F} := \{ \mathbf{y} \in \mathbb{R}_{\geq 0}^E \mid \mathbf{P}\mathcal{L}(\mathbf{y})\mathbf{P} \preceq \mathbf{\Pi} \}. \quad (51)$$

Moreover, the only access used by \mathcal{O} to its input \mathbf{P} is matrix-vector query access, and its output \mathbf{x} has the form $\mathbf{x} = \mathbf{v} \circ \mathbf{a}$ where \mathbf{v} is a K -SOC and \mathbf{a} is an ASOC.

Indeed, note that Proposition 6 exactly provides an oracle of this form, when \mathbf{g} is an ASOC. We next show that packing problems with a well-conditioned solution are not substantially affected by small entries, and that we can approximate certain distances encountered in our algorithm. These helper observations are for using our ASOC approximation scheme (Lemma 17).

Lemma 29. Let $\mathbf{w} \in \mathbb{R}_{>0}^E$ have $\max_{e \in E} \mathbf{w}_e \leq \rho \min_{e \in E} \mathbf{w}_e$, let $\mathbf{Q} \in \mathbb{R}^{k \times n}$, $\mathbf{f} \in \mathbb{R}^E$ be such that

$$\mathbf{f}_e = \|\mathbf{Q}_{:u} - \mathbf{Q}_{:v}\|_2^2 \text{ for all } e = (u, v) \in E.$$

Also, suppose $\gamma, \alpha > 0$ satisfy

$$\gamma \geq \max_{(u, v) \in E} \max_{j \in [k]} (\mathbf{Q}_{ju} - \mathbf{Q}_{jv})^2, \quad \frac{\gamma}{\alpha} \leq \frac{1}{2\rho n^2 k} \max_{(u, v) \in E} \max_{j \in [k]} (\mathbf{Q}_{ju} - \mathbf{Q}_{jv})^2.$$

Then, following notation from Definition 6, if $\frac{1}{2}\mathbf{g} \leq \mathbf{f} \leq \mathbf{g}$ entrywise,

$$\langle \mathbf{f}^{(\geq \frac{\gamma}{\alpha})}, \mathbf{w} \rangle \geq \frac{1}{4} \langle \mathbf{g}, \mathbf{w} \rangle.$$

Proof. It is enough to show that $\langle \mathbf{f}^{(\geq \frac{\gamma}{\alpha})}, \mathbf{w} \rangle \geq \frac{1}{2} \langle \mathbf{f}, \mathbf{w} \rangle$, because we know that $\mathbf{f} \geq \frac{1}{2} \mathbf{g}$ entrywise. Throughout the proof let $(a, b) \in E$ and $\ell \in [k]$ be the maximizing arguments in

$$\max_{(u,v) \in E} \max_{j \in [k]} (\mathbf{Q}_{ju} - \mathbf{Q}_{jv})^2.$$

Because

$$\langle \mathbf{f}, \mathbf{w} \rangle = \sum_{(u,v) \in E} \sum_{j \in [k]} \mathbf{w}_{(u,v)} (\mathbf{Q}_{ju} - \mathbf{Q}_{jv})^2 \geq \mathbf{w}_{(a,b)} (\mathbf{Q}_{\ell a} - \mathbf{Q}_{\ell b})^2,$$

for any $(u, v) \in E$ and $j \in [k]$ that is a pair such that the corresponding squared distance in \mathbf{Q} does not participate in $\mathbf{f}^{(\frac{\gamma}{\alpha})}$, we can bound the contribution of the pair to the objective value, i.e.,

$$\mathbf{w}_{(u,v)} (\mathbf{Q}_{ju} - \mathbf{Q}_{jv})^2 \leq \rho \mathbf{w}_{(a,b)} \cdot \frac{\gamma}{\alpha} \leq \frac{1}{2n^2 k} \cdot \mathbf{w}_{(a,b)} (\mathbf{Q}_{\ell a} - \mathbf{Q}_{\ell b})^2.$$

By combining the above two displays, for any such non-contributing $(u, v) \in E$, $j \in [k]$,

$$\mathbf{w}_{(u,v)} (\mathbf{Q}_{ju} - \mathbf{Q}_{jv})^2 \leq \frac{1}{2n^2 k} \langle \mathbf{f}, \mathbf{w} \rangle.$$

Summing over the contributions of all $\leq kn^2$ possible pairs of $(u, v) \in E$ and $j \in [k]$, we have shown that the dropped coordinates contribute at most half the value of $\langle \mathbf{f}, \mathbf{w} \rangle$ as claimed. \square

Lemma 30. *Let $\mathbf{x} \in \mathbb{R}^E$ have the form (45) where all $\{\mathbf{a}_s\}_{s \in [S]}$ are ASOCs and all $\{\mathbf{v}_s\}_{s \in [S]}$ are K-SOCs, and define \mathbf{S}, \mathbf{Y} as in (45), (44). Assume $\|\mathbf{S}\|_{\text{op}} \leq R$, and let $\delta \in (0, 1)$. In time*

$$O\left((\mathcal{T}_{\text{mv}}(\mathbf{P}) + nKS \log(n)) R \log\left(\frac{n}{\delta}\right)\right),$$

we can output $Z \in \mathbb{R}$ satisfying $\frac{9}{10} \text{Tr}(\mathbf{P} \mathbf{Y} \mathbf{P}) \leq Z \leq \text{Tr}(\mathbf{P} \mathbf{Y} \mathbf{P})$ with probability $\geq 1 - \delta$.

Proof. This is standard in the literature, e.g., it is implicit in Lemma 2 of [JLM⁺23], which shows that to obtain a constant factor multiplicative estimate of the trace of $\mathbf{P} \mathbf{Y} \mathbf{P} \in \mathbb{S}_{\geq 0}^{n \times n}$ with probability $\geq 1 - \delta$, it is enough to compute $O(\log(\frac{n}{\delta}))$ many matrix-vector multiplications through $\mathbf{P} \mathbf{Y} \mathbf{P}$. Here we use the polynomial approximation from Lemma 28 and the trace approximation from Lemma 27 to simulate constant factor multiplicative approximate access to \mathbf{Y} . \square

Finally, we recall a standard regret bound on the matrix multiplicative weights algorithm.

Proposition 7 (Theorem 3.1, [ZLO15]). *Consider a sequence of gain matrices $\{\mathbf{G}_t\}_{0 \leq t < T} \subset \mathbb{S}_{\geq 0}^{n \times n}$, which all satisfy for step size $\eta > 0$, that $\|\eta \mathbf{G}_t\|_{\text{op}} \leq 1$. Letting $\mathbf{S}_0 := \mathbf{0}_{n \times n}$ and iteratively defining*

$$\mathbf{Y}_t := \frac{\exp(-\mathbf{S}_t)}{\text{Tr} \exp(-\mathbf{S}_t)}, \quad \mathbf{S}_{t+1} := \mathbf{S}_t + \eta \mathbf{G}_t \text{ for all } 0 \leq t < T,$$

we have the bound for any $\mathbf{U} \in \mathbb{S}_{\geq 0}^{n \times n}$ with $\text{Tr}(\mathbf{U}) = 1$,

$$\frac{1}{T} \sum_{0 \leq t < T} \langle \mathbf{G}_t, \mathbf{Y}_t - \mathbf{U} \rangle \leq \frac{\log(n)}{\eta T} + \frac{1}{T} \sum_{t \in [T]} \eta \|\mathbf{G}_t\|_{\text{op}} \langle \mathbf{G}_t, \mathbf{Y}_t \rangle.$$

Moreover, if all \mathbf{G}_t have spans contained in a subspace $S \subseteq \mathbb{R}^n$, then the above bound holds for all $\mathbf{U} \in \mathbb{S}_{\geq 0}^{n \times n}$ with $\text{Tr}(\mathbf{U})$ whose spans are contained in S .

Algorithm 2: OracleMDR($\mathbf{P}, \rho, Q, K, \delta, m, \mathcal{O}$)

1 **Input:** Matrix-vector query access to $\mathbf{P} \in \mathbb{S}_{\geq 0}^{n \times n}$ satisfying (50) for unknown $\mathbf{L} = \mathcal{L}(\mathbf{w}) \in \mathbb{S}_{\geq 0}^{n \times n}$, where $\max_{e \in E} \mathbf{w}_e \leq \rho \min_{e \in E} \mathbf{w}_e$, $(Q, K, m, \delta) \in \mathbb{R}_{\geq 1} \times \mathbb{N} \times \mathbb{N} \times (0, 1)$, \mathcal{O} a $(Q, K, \frac{\delta}{900m^2Q\log(n)})$ -SOC packing oracle for \mathbf{L}

2 **Output:** $\bar{\mathbf{x}} \in \mathbb{R}_{\geq 0}^E$ such that with probability $\geq 1 - \delta$,

$$\frac{1}{512mQ} \mathbf{L} \preceq \mathcal{L}(\bar{\mathbf{x}}) \preceq \mathbf{L}$$

3 $(\eta, \mathbf{S}_0, T, \alpha) \leftarrow (\frac{1}{2}, \mathbf{0}_{n \times n}, \lceil 256mQ\log(n) \rceil, O(\rho n^4 \log^2(\frac{nT}{\delta})))$

4 **for** $0 \leq t < T$ **do**

5 $\mathbf{Y}_t \leftarrow \frac{\exp(-\mathbf{S}_t)}{\text{Tr} \exp(-\mathbf{S}_t)}$

6 $\gamma \leftarrow \text{value satisfying } \text{Tr}(\mathbf{P} \mathbf{Y}_t \mathbf{P}) \leq \gamma \leq \frac{10}{9} \text{Tr}(\mathbf{P} \mathbf{Y}_t \mathbf{P}) \text{ with probability } \geq 1 - \frac{\delta}{900mQ\log(n)}$

7 $\{\tilde{\mathbf{g}}_t^{(i)}\}_{i \in [m]} \leftarrow (8, m)$ -ASOC approximation to \mathbf{g}_t with probability $\geq 1 - \frac{\delta}{900mQ\log(n)}$, where

$$[\mathbf{g}_t]_e := \langle \mathbf{P} \mathbf{Y}_t \mathbf{P}, \mathbf{L}_e \rangle \text{ for all } e \in E \quad (52)$$

8 **for** $i \in [m]$ **do**

9 $\mathbf{x}_t^{(i)} \leftarrow \mathcal{O}(\tilde{\mathbf{g}}_t^{(i)}, \mathbf{P})$

10 **end**

11 $\mathbf{x}_t \leftarrow \frac{1}{m} \sum_{i \in [m]} \mathbf{x}_t^{(i)}$

12 $\mathbf{G}_t \leftarrow \mathbf{P} \mathcal{L}(\mathbf{x}_t) \mathbf{P}$

13 $\mathbf{S}_{t+1} \leftarrow \mathbf{S}_t + \eta \mathbf{G}_t$

14 **end**

15 **Return:** $\bar{\mathbf{x}} := \frac{1}{T} \sum_{0 \leq t < T} \mathbf{x}_t$

We can now state our matrix dictionary recovery method and its guarantees.

Proposition 8. *Following notation in Algorithm 2, suppose that we use Proposition 6 as our SOC packing oracle, that we use Lemma 30 to implement Line 6, and that we use Lemmas 17 and 28 to implement Line 7. Then, Algorithm 2 returns $\bar{\mathbf{x}} \in \mathbb{R}_{\geq 0}^E$ such that with probability $\geq 1 - \delta$,*

$$\exp\left(-O\left(\log^{\frac{2}{3}}(n\rho) \log \log\left(\frac{n\rho}{\delta}\right)\right)\right) \mathbf{L} \preceq \mathcal{L}(\bar{\mathbf{x}}) \preceq \mathbf{L}. \quad (53)$$

Moreover, $\bar{\mathbf{x}} = \sum_{s \in [S]} \mathbf{v}_s \circ \mathbf{a}_s$ where all $\{\mathbf{a}_s\}_{s \in [S]}$ are ASOCs and all $\{\mathbf{v}_s\}_{s \in [S]}$ are K -SOCs, where

$$K = S = \exp\left(O\left(\log^{\frac{2}{3}}(n\rho) \log \log\left(\frac{n\rho}{\delta}\right)\right)\right),$$

and the algorithm can be implemented to run in time

$$(\mathcal{T}_{\text{mv}}(\mathbf{P}) + n) \cdot \exp\left(O\left(\log^{\frac{2}{3}}(n\rho) \log \log\left(\frac{n\rho}{\delta}\right)\right)\right).$$

Proof. Throughout this proof let $\beta := 8$, and let

$$k = O\left(\log\left(\frac{nT}{\delta}\right)\right), \quad m = O\left(k^2 \log\left(\frac{1}{\alpha}\right)\right), \quad Q = \exp\left(O\left(\log^{\frac{2}{3}}\left(\frac{nu}{\ell}\right) \log \log\left(\frac{nTu}{\delta\ell}\right)\right)\right),$$

be the parameters from Lemmas 17 and 28 and Proposition 6, where the last expression is for bounds ℓ, u on the value of our packing objectives that we will specify. As all above expressions depend logarithmically on T (which depends linearly on mQ), there are no conflicts for an appropriate choice of constants. Further, since $T \leq 300mQ \log(n)$, by a union bound, we may assume that all computations on Lines 6, 7 are correct, and that all calls to \mathcal{O} succeed, except with probability δ .

Condition on these events for the rest of the proof. We next specify valid bounds ℓ, u that hold with probability 1 for all packing problems encountered in the algorithm. Specifically we let

$$\ell := \frac{1}{2} \min_{e \in E} \mathbf{w}_e, \quad u := \frac{n^4}{4} \max_{e \in E} \mathbf{w}_e \implies \frac{u}{\ell} \leq n^4 \rho. \quad (54)$$

To see why the bounds in (54) are valid, first, observe that

$$\mathcal{L}(\mathbf{w}) \preceq 2 \sum_{e \in E} \mathbf{w}_e \mathbf{\Pi} \preceq \left(n^2 \max_{e \in E} \mathbf{w}_e\right) \mathbf{\Pi}, \quad (55)$$

where we used that $2\mathbf{\Pi}$ dominates any \mathbf{L}_e . Thus for any packing problem of the form (38) encountered by the algorithm, by using the assumption (50), the optimal $\mathbf{x} \in \mathbb{R}_{\geq 0}^E$ cannot have

$$\mathbf{x}_e > \frac{n^2 \max_{e \in E} \mathbf{w}_e}{2} \geq \frac{1}{2\lambda_n(\mathbf{P})^2} \text{ for any } e \in E,$$

because then this coordinate alone would violate the feasibility constraint:

$$\mathbf{x}_e \mathbf{A}_e = \mathbf{x}_e \mathbf{P} \mathbf{L}_e \mathbf{P} \not\preceq \mathbf{\Pi}.$$

This implies u in (54) is a valid upper bound on $\langle \mathbf{c}, \mathbf{x} \rangle$ for feasible \mathbf{x} and $\mathbf{c} \in \{0, 1\}^E$, because $\|\mathbf{c}\|_1 \leq \frac{n^2}{2}$. Next, to obtain the lower bound, by pre-multiplying and post-multiplying (50) by $\mathbf{L}^{\frac{1}{2}}$, all eigenvalues of \mathbf{PLP} are in $[1, 2]$ (other than a zero eigenvalue in the $\mathbf{1}_n$ direction), i.e.,

$$\mathbf{\Pi} \preceq \mathbf{PLP} \preceq 2\mathbf{\Pi}.$$

Thus, $\frac{1}{2}\mathbf{w} \in \mathcal{F}$ for the feasible region \mathcal{F} defined in (51), because

$$\mathbf{PL} \left(\frac{1}{2}\mathbf{w} \right) \mathbf{P} = \frac{1}{2}\mathbf{PLP} \preceq \mathbf{\Pi}.$$

Thus the optimal $\langle \mathbf{c}, \mathbf{x} \rangle$ is always at least $\frac{1}{2} \min_{e \in E} \mathbf{w}_e$, by plugging in the feasible choice of $\mathbf{x} = \frac{1}{2}\mathbf{w}$. This concludes the proof that ℓ in (38) is also valid. We simplify our parameters accordingly:

$$\begin{aligned} Q &= \exp \left(O \left(\log^{\frac{2}{3}} (n\rho) \log \log \left(\frac{n\rho}{\delta} \right) \right) \right), \quad T = \exp \left(O \left(\log^{\frac{2}{3}} (n\rho) \log \log \left(\frac{n\rho}{\delta} \right) \right) \right), \\ \alpha &= O \left(n^4 \rho \log^2 \left(\frac{n\rho}{\delta} \right) \right), \quad k = O \left(\log \left(\frac{n\rho}{\delta} \right) \right), \quad m = O \left(\log^3 \left(\frac{n\rho}{\delta} \right) \right). \end{aligned} \quad (56)$$

Continuing, we have for any \mathbf{Y} such that $\text{Tr}(\mathbf{Y}) = 1$ and $\text{Span}(\mathbf{Y}) = \text{Span}(\mathbf{\Pi})$:

$$\langle \mathbf{P} \mathbf{Y} \mathbf{P}, \mathcal{L}(\mathbf{w}) \rangle = \langle \mathbf{Y}, \mathbf{PLP} \rangle \succeq \langle \mathbf{Y}, \mathbf{\Pi} \rangle = 1. \quad (57)$$

By the guarantee on \mathcal{O} (Definition 8), we have that for all $i \in [m]$ and $0 \leq t < T$,

$$\langle \tilde{\mathbf{g}}_t^{(i)}, \mathbf{x}^{(i)} \rangle \geq \frac{1}{2Q} \langle \tilde{\mathbf{g}}_t^{(i)}, \mathbf{w} \rangle \implies \langle \mathbf{g}_t, \mathbf{x}_t^{(i)} \rangle \geq \frac{1}{2\beta Q} \langle \tilde{\mathbf{g}}_t^{(i)}, \mathbf{w} \rangle.$$

The above implication used that $\mathbf{g}_t \geq \frac{1}{\beta} \tilde{\mathbf{g}}_t^{(i)}$ for all $i \in [m]$ by the ASOC approximation guarantee. Averaging this bound for all $i \in [m]$,

$$\langle \mathbf{g}_t, \mathbf{x}_t \rangle = \frac{1}{m} \sum_{i \in [m]} \langle \mathbf{g}_t, \mathbf{x}_t^{(i)} \rangle \geq \frac{1}{2\beta m Q} \sum_{i \in [m]} \langle \tilde{\mathbf{g}}_t^{(i)}, \mathbf{w} \rangle \geq \frac{1}{8\beta m Q} \langle \mathbf{g}_t, \mathbf{w} \rangle. \quad (58)$$

The last inequality in (58) is derived as follows. Recall that $\sum_{i \in [m]} \tilde{\mathbf{g}}_t^{(i)} \geq \mathbf{f}_t^{(\geq \frac{\gamma}{\alpha})}$ by combining our ASOC approximation guarantee with $\mathbf{f}_t \geq \frac{1}{2}\mathbf{g}_t$ for our estimate \mathbf{f}_t constructed in Lemma 28. Note that γ is at most a $\frac{20}{9}n^2k$ factor overestimate of the largest entry in Lemma 29, because

$$\sum_{e \in E} \mathbf{g}_e = \left\langle \mathbf{P} \mathbf{Y} \mathbf{P}, \sum_{e \in E} \mathbf{L}_e \right\rangle = \langle \mathbf{P} \mathbf{Y} \mathbf{P}, n\mathbf{\Pi} \rangle = n \text{Tr}(\mathbf{P} \mathbf{Y} \mathbf{P}),$$

so our choice of $\alpha = \frac{40}{9}\rho n^4 k^2$ shows that Lemma 29 applies. Thus,

$$\sum_{i \in [m]} \langle \tilde{\mathbf{g}}_t^{(i)}, \mathbf{w} \rangle \geq \langle \mathbf{f}_t^{(\frac{\gamma}{\alpha})}, \mathbf{w} \rangle \geq \frac{1}{4} \langle \mathbf{g}_t, \mathbf{w} \rangle.$$

However, we also have by the definition of \mathbf{g}_t in (52) that

$$\langle \mathbf{g}_t, \mathbf{w} \rangle = \langle \mathbf{P} \mathbf{Y}_t \mathbf{P}, \mathcal{L}(\mathbf{w}) \rangle \geq 1,$$

where we used (57) in the last inequality. Combining with (58), we have shown

$$\langle \mathbf{G}_t, \mathbf{Y}_t \rangle = \langle \mathbf{P}\mathcal{L}(\mathbf{x}_t)\mathbf{P}, \mathbf{Y}_t \rangle = \langle \mathbf{g}_t, \mathbf{x}_t \rangle \geq \frac{1}{8\beta m Q}, \quad (59)$$

for all $0 \leq t < T$. Also, by the packing oracle guarantee $\mathbf{x}_t \in \mathcal{F}$,

$$\mathbf{G}_t = \mathbf{P}\mathcal{L}(\mathbf{x}_t)\mathbf{P} \preceq \mathbf{\Pi}. \quad (60)$$

Now, rearranging Proposition 7 gives for all $\mathbf{U} \in \mathbb{S}_{\geq 0}^{n \times n}$ with $\text{Span}(\mathbf{U}) = \text{Span}(\mathbf{\Pi})$ and $\text{Tr}(\mathbf{U}) = 1$,

$$\begin{aligned} \langle \mathbf{P}\mathcal{L}(\bar{\mathbf{x}})\mathbf{P}, \mathbf{U} \rangle &= \frac{1}{T} \sum_{0 \leq t < T} \langle \mathbf{G}_t, \mathbf{U} \rangle \\ &\geq \frac{1}{2T} \sum_{0 \leq t < T} \langle \mathbf{G}_t, \mathbf{Y}_t \rangle - \frac{\log(n)}{\eta T} \\ &= \frac{1}{16\beta m Q} - \frac{1}{32\beta m Q} = \frac{1}{32\beta m Q}. \end{aligned} \quad (61)$$

The first line above used the definition of $\bar{\mathbf{x}}$, the second used Proposition 7 with $\eta = \frac{1}{2}$ and $\|\mathbf{G}_t\|_{\text{op}} \leq 1$ by (60), and the third used (59) and plugged in our choices of η and T . Minimizing (61) over all valid choices of \mathbf{U} shows that

$$\frac{1}{32\beta m Q} \mathbf{\Pi} \preceq \mathbf{P}\mathcal{L}(\bar{\mathbf{x}})\mathbf{P} \preceq \mathbf{\Pi},$$

where the upper bound holds by averaging (60) across all iterations. Thus, we indeed have

$$\frac{1}{64\beta m Q} \mathbf{L} \preceq \mathcal{L}(\bar{\mathbf{x}}) \preceq \mathbf{L},$$

which proves (53) upon plugging in $\beta = 8$ and our parameters from (56).

Next we discuss the maintenance of our iterates. Because of the SOC packing oracle guarantee, every \mathbf{x}_t for $0 \leq t < T$ is the average of m products of a K -SOC and an ASOC, where m is as in (56) and K is as stated in Proposition 6. Thus, the average of all T iterates has $S = mT$ components, each a product of a K -SOC and an ASOC, in its representation as claimed.

There are three runtime bottlenecks in our algorithm: for each of T iterations, we call Lemma 30 to implement Line 6, we call Lemmas 28 and 17 to implement Line 7, and we call Proposition 6 m times to implement our SOC packing oracle. Moreover, $\|\mathbf{S}_t\|_{\text{op}} \leq \frac{T}{2}$ for all iterations $0 \leq t < T$ by inspection. The result follows by combining all of these runtimes with our parameter choices. \square

4.5 Homotopy method

In this section, we show how recursively calling Proposition 8 in phases grants us access to \mathbf{P} in (50) needed by the next phase. We use an implicit approximation provided in [JLM⁺23].

Lemma 31. *Suppose that for $\bar{\mathbf{x}} \in \mathbb{R}^E$, $\Delta, Q > 0$, and unknown graph Laplacian $\mathbf{L} \in \mathbb{S}_{\geq 0}^{n \times n}$, we have*

$$\frac{1}{Q} (\mathbf{L} + \Delta \mathbf{\Pi}) \preceq \mathcal{L}(\bar{\mathbf{x}}) \preceq \mathbf{L} + \Delta \mathbf{\Pi},$$

where $\Pi := \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$. Further, suppose that $\bar{\mathbf{x}} = \sum_{s \in [S]} \mathbf{a}_s \circ \mathbf{v}_s$ where all $\{\mathbf{a}_s\}_{s \in [S]}$ are ASOCs and all $\{\mathbf{v}_s\}_{s \in [S]}$ are K-SOCs. Then we can provide matrix-vector product access to a matrix \mathbf{P} that satisfies the following with probability $\geq 1 - \delta$:

$$(\mathbf{L} + \Delta\Pi)^\dagger \preceq \mathbf{P}^2 \preceq 2(\mathbf{L} + \Delta\Pi)^\dagger,$$

$$\mathcal{T}_{\text{mv}}(\mathbf{P}) = O\left(\left(\mathcal{T}_{\text{mv}}(\mathbf{L}) + nKS \log^2(n) \log^2\left(\frac{nK\text{Tr}(\mathbf{L})}{\Delta\delta}\right)\right) \sqrt{Q} \log^6\left(\frac{Q\text{Tr}(\mathbf{L})}{\Delta\delta}\right)\right).$$

Proof. We first sparsify $\mathcal{L}(\bar{\mathbf{x}})$ using Lemma 12 to obtain a graph Laplacian $\tilde{\mathbf{L}}$ that satisfies

$$\frac{1}{Q \exp(2)} (\mathbf{L} + \Delta\Pi) \preceq \tilde{\mathbf{L}} \preceq \mathbf{L} + \Delta\Pi, \quad \text{nnz}(\tilde{\mathbf{L}}) = O\left(nKS \log(n) \log\left(\frac{nKS}{\delta}\right)\right),$$

except with probability $\frac{\delta}{2}$. Next, we can provide access to constant-factor approximations of $(\tilde{\mathbf{L}} + \lambda\Pi)^\dagger$ for any L values of $\lambda \geq 0$ with failure probability $\frac{\delta}{2}$, in time $O(\text{nnz}(\tilde{\mathbf{L}}) \log(n) \log(\frac{nL}{\delta}))$ by using the Laplacian system solver of [KMP14]. The conclusion follows from Lemma 13, [JLM⁺23], which requires this primitive with $L = O(\log(\frac{\text{Tr}(\mathbf{L})}{\Delta}))$ (see Proposition 3 in [JLM⁺23]). We remark that the approximation factor worsens by a $\frac{\text{Tr}(\mathbf{L})}{\Delta}$ factor due to Lemma 14 in [JLM⁺23], but this is accounted for by the polylogarithmic terms above, where the accuracy dependence lies. \square

We are now ready to prove Theorem 2.

Theorem 2. *Let \mathbf{L} be an $n \times n$ graph Laplacian, and let $\mathcal{O} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be an oracle that returns $\mathbf{L}\mathbf{v}$ on input $\mathbf{v} \in \mathbb{R}^n$. Let $\delta \in (0, 1)$, $\Delta \in (0, \text{Tr}(\mathbf{L}))$, and let $\Pi := \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$ be the projection matrix to the subspace of \mathbb{R}^n orthogonal to $\mathbf{1}_n$. There is an algorithm that takes as inputs $(\mathcal{O}, \delta, \Delta)$ and with probability $\geq 1 - \delta$, it returns $\tilde{\mathbf{L}}$, an $n \times n$ graph Laplacian satisfying*

$$\mathbf{L} + \Delta\Pi \preceq \tilde{\mathbf{L}} \preceq \left(\frac{n\text{Tr}(\mathbf{L})}{\Delta\delta}\right)^{o(1)} (\mathbf{L} + \Delta\Pi), \quad \text{nnz}(\tilde{\mathbf{L}}) = n \cdot \left(\frac{n\text{Tr}(\mathbf{L})}{\Delta\delta}\right)^{o(1)}, \quad (7)$$

using $(\frac{n\text{Tr}(\mathbf{L})}{\Delta\delta})^{o(1)}$ queries to \mathcal{O} , and $n \cdot (\frac{n\text{Tr}(\mathbf{L})}{\Delta\delta})^{o(1)}$ additional time.

Proof. We closely follow the outline of the homotopy method outlined in Section 3.2 of [JLM⁺23]. The method proceeds in $p = O(\log \frac{\text{Tr}(\mathbf{L})}{\Delta})$ phases, and in each phase $q \in [p]$, we apply Proposition 8 to the unknown regularized Laplacian

$$\mathbf{L}_q := \mathbf{L} + \Delta 2^{p-q}\Pi.$$

Note that in each phase, $\mathbf{L}_q = \mathcal{L}(\mathbf{w}_q)$ for a vector $\mathbf{w}_q \in \mathbb{R}_{\geq 0}^E$ that is entrywise at least $2^{p-q} \cdot \frac{\Delta}{n}$, and whose sum of entries is at most $2^{p-q} \cdot \Delta n + \text{Tr}(\mathbf{L})$. Thus, in all calls to Proposition 8, we have

$$\rho = O\left(n^2 + \frac{n\text{Tr}(\mathbf{L})}{\Delta}\right).$$

To initialize each phase, we require a matrix \mathbf{P}_q satisfying (50). In the first phase $q = 1$, for a large enough constant in the definition of p , it is enough to choose \mathbf{P}_1 to be a known multiple of Π . In every phase q after this, \mathbf{P}_q results from applying Lemma 31. Because

$$\mathbf{L}_q \preceq \mathbf{L}_{q-1} \preceq 2\mathbf{L}_q$$

for all $q \in [p]$, the multiplicative approximation factor given by Proposition 8 only worsens by 2. The runtime follows from Proposition 8 and our matrix-vector query access in Lemma 31. Finally, the sparsity of the final output follows by applying the sparsification described in the proof of Lemma 31 to the output of the final phase p , and scaling it up to obtain the approximation guarantee. \square

5 Conditioning of Smoothed Matrices

In this section, we provide an ℓ_∞ diameter bound for a minimizing vector of Barthe's objective, when computing a Forster transform of a smoothed matrices of the form

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{G}, \text{ where } \mathbf{A} \in \mathbb{R}^{n \times d}, \text{ and } \mathbf{G} \in \mathbb{R}^{n \times d} \text{ has entries } \sim_{\text{i.i.d.}} \mathcal{N}(0, \sigma^2).$$

We first define a notion of deepness in Section 5.1 and derive a diameter bound for deep vectors, patterned off [AKS20]. In Sections 5.2 and 5.3, we show tail bounds for the singular values of a smoothed matrix. We combine these results to prove Theorem 3 in Section 5.4, our main result on the conditioning of Forster transforms of smoothed matrices.

Throughout this section, we follow notation (18), (19), (20) from Section 2.2 and make the following simplifying, and somewhat mild, assumption on the relationship between n and d .

Assumption 2. $n \geq Cd$ for some constant $C > 1$.

For example, our results apply if $n \geq 1.1d$. Lifting the restriction in Assumption 2 significantly complicates our approach, as explained by Remark 3, and we leave it as an interesting open question to establish similar bounds in the parameter regime $n \in [d, d + o(d)]$.

Finally, we mention that our result in Theorem 3 is stated for the case of $\mathbf{c} = \frac{d}{n}\mathbf{1}_n$, which is the most interesting setting we are aware of in typical applications. However, we discuss the case of general \mathbf{c} in Section 5.5, where our techniques readily apply under a strengthening of Assumption 2.

5.1 Diameter bound for deep vectors

Our strategy for our diameter bound follows an analysis in [AKS20], based on the assumption that \mathbf{c} lies nontrivially inside the interior of the basis polytope (cf. Proposition 2).

We start by extending Definition 1.4 and proving a stronger version of Lemma 4.4 from [AKS20].

Definition 9 (Deepness). *Let $\mathbf{c} \in \mathbb{R}_{>0}^n$ satisfy $\|\mathbf{c}\|_1 = d$, $\eta \in [0, 1]$, and $\Delta \geq 0$. We say that \mathbf{c} lies (η, Δ) -deep inside the basis polytope of $\{\mathbf{a}_i\}_{i \in [n]} \subset \mathbb{R}^d$ if for all $k \in [d-1]$ and subspaces $E \subseteq \mathbb{R}^d$ with dimension k ,*

$$\sum_{i \in [n]} \mathbf{c}_i \mathbb{1}_{\|\mathbf{a}_i - \mathbf{\Pi}_E \mathbf{a}_i\|_2 \leq \Delta} \leq (1 - \eta)k.$$

When $\mathbf{c} = \frac{d}{n}\mathbf{1}_n$, this is equivalent to the following: for all $k \in [d-1]$ and subspaces E with dimension k , at most $\frac{(1-\eta)kn}{d}$ of the \mathbf{a}_i satisfy $\|\mathbf{a}_i - \mathbf{\Pi}_E \mathbf{a}_i\|_2 \leq \Delta$.

We remark that every vector in the basis polytope is $(0, 0)$ -deep by Proposition 2. Furthermore, increasing Δ potentially increases the number of \mathbf{c}_i considered in the sum, and increasing η tightens the inequality, so (η, Δ) -deepness becomes a stronger condition for larger values of η and Δ .

We now show, following [AKS20], that deepness in the basis polytope implies a diameter bound.

Lemma 32. Let $\mu, M, \eta, \Delta > 0$ and $\mathbf{A} \in \mathbb{R}^{n \times d}$ with rows $\{\mathbf{a}_i\}_{i \in [n]}$ satisfying $\mu \leq \|\mathbf{a}_i\|_2^2 \leq M$ for all $i \in [n]$. If $\mathbf{c} \in \mathbb{R}_{>0}^n$ has minimum entry \mathbf{c}_{\min} and lies (η, Δ) -deep inside the basis polytope of $\{\mathbf{a}_i\}_{i \in [n]}$, then there exists $\mathbf{t}^* \in \arg \min_{\mathbf{t} \in \mathbb{R}^n} f(\mathbf{t})$ satisfying

$$\|\mathbf{t}^*\|_\infty \leq \frac{1}{2} \log \left(\frac{M}{\mu \mathbf{c}_{\min}} \left(\frac{4M}{\eta \Delta^2} \right)^{d-1} \right).$$

Proof. Let $\mathbf{t}^* \in \arg \min_{\mathbf{t} \in \mathbb{R}^n} f(\mathbf{t})$ such that $\min_{i \in [n]} \mathbf{t}_i^* = 0$, which exists by Proposition 2, Proposition 3, and (6). Let $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$ be the eigenvalues of $\mathbf{R} := \mathbf{R}(\mathbf{t}^*)$.

Fix $k \in [d-1]$. Let E be a k -dimensional subspace spanned by eigenvectors of \mathbf{R} with eigenvalues $\lambda_d, \lambda_{d-1}, \dots, \lambda_{d-k+1}$. By Proposition 3, \mathbf{R} is a \mathbf{c} -Forster transform of \mathbf{A} , so

$$\sum_{i \in [n]} \mathbf{c}_i \cdot \frac{(\mathbf{R}\mathbf{a}_i)(\mathbf{R}\mathbf{a}_i)^\top}{\|\mathbf{R}\mathbf{a}_i\|_2^2} = \mathbf{I}_d.$$

Projecting onto E^\perp and taking a trace of both sides,

$$\sum_{i \in [n]} \mathbf{c}_i \cdot \frac{\|\Pi_{E^\perp} \mathbf{R}\mathbf{a}_i\|_2^2}{\|\mathbf{R}\mathbf{a}_i\|_2^2} = \text{Tr} \left(\sum_{i \in [n]} \mathbf{c}_i \cdot \frac{(\Pi_{E^\perp} \mathbf{R}\mathbf{a}_i)(\Pi_{E^\perp} \mathbf{R}\mathbf{a}_i)^\top}{\|\mathbf{R}\mathbf{a}_i\|_2^2} \right) = \text{Tr}(\Pi_{E^\perp}) = d - k. \quad (62)$$

Now, consider the \mathbf{a}_i such that $\|\mathbf{a}_i - \Pi_E \mathbf{a}_i\|_2 > \Delta$, and decompose these \mathbf{a}_i as $\mathbf{a}_i = \mathbf{y}_i + \mathbf{z}_i$, where $\mathbf{y}_i \in E$ and $\mathbf{z}_i \in E^\perp$. Then $\|\mathbf{z}_i\|_2 > \Delta$ and $\|\mathbf{y}_i\|_2 < \sqrt{\|\mathbf{a}_i\|_2^2 - \Delta^2} \leq \sqrt{M - \Delta^2}$. Since E and E^\perp are both spanned by eigenvectors of \mathbf{R} , $\mathbf{R}\mathbf{y}_i$ and $\mathbf{R}\mathbf{z}_i$ are orthogonal, and $\|\mathbf{R}\mathbf{a}_i\|_2^2 = \|\mathbf{R}\mathbf{y}_i\|_2^2 + \|\mathbf{R}\mathbf{z}_i\|_2^2$. Furthermore, since E is spanned by eigenvectors with eigenvalues at most λ_{d-k+1} and E^\perp is spanned by eigenvectors with eigenvalues at least λ_{d-k} ,

$$\|\mathbf{R}\mathbf{y}_i\|_2 \leq \lambda_{d-k+1} \|\mathbf{y}_i\|_2 \leq \lambda_{d-k+1} \sqrt{M - \Delta^2} \text{ and } \|\mathbf{R}\mathbf{a}_i\|_2 \geq \|\mathbf{R}\mathbf{z}_i\|_2 \geq \lambda_{d-k} \|\mathbf{z}_i\|_2 \geq \lambda_{d-k} \Delta.$$

It follows that

$$\frac{\|\Pi_{E^\perp} \mathbf{R}\mathbf{a}_i\|_2^2}{\|\mathbf{R}\mathbf{a}_i\|_2^2} = \left\| \Pi_{E^\perp} \frac{\mathbf{R}\mathbf{a}_i}{\|\mathbf{R}\mathbf{a}_i\|_2} \right\|_2^2 = 1 - \left\| \Pi_E \frac{\mathbf{R}\mathbf{a}_i}{\|\mathbf{R}\mathbf{a}_i\|_2} \right\|_2^2 = 1 - \frac{\|\mathbf{R}\mathbf{y}_i\|_2^2}{\|\mathbf{R}\mathbf{a}_i\|_2^2} \geq 1 - \frac{\lambda_{d-k+1}^2 (M - \Delta^2)}{\lambda_{d-k}^2 \Delta^2}.$$

Combining this with (62) and the (η, Δ) -deepness of \mathbf{c} ,

$$\begin{aligned} d - k &= \sum_{i \in [n]} \mathbf{c}_i \cdot \frac{\|\Pi_{E^\perp} \mathbf{R}\mathbf{a}_i\|_2^2}{\|\mathbf{R}\mathbf{a}_i\|_2^2} \geq \sum_{i \in [n]} \mathbf{c}_i \mathbb{1}_{\|\mathbf{a}_i - \Pi_E \mathbf{a}_i\|_2 > \Delta} \cdot \frac{\|\Pi_{E^\perp} \mathbf{R}\mathbf{a}_i\|_2^2}{\|\mathbf{R}\mathbf{a}_i\|_2^2} \\ &\geq \left(1 - \frac{\lambda_{d-k+1}^2 (M - \Delta^2)}{\lambda_{d-k}^2 \Delta^2} \right) \sum_{i \in [n]} \mathbf{c}_i \mathbb{1}_{\|\mathbf{a}_i - \Pi_E \mathbf{a}_i\|_2 > \Delta} \\ &\geq \left(1 - \frac{\lambda_{d-k+1}^2 (M - \Delta^2)}{\lambda_{d-k}^2 \Delta^2} \right) (d - (1 - \eta)k), \end{aligned}$$

which rearranges to

$$\frac{\lambda_{d-k}}{\lambda_{d-k+1}} \leq \left(\frac{d-k+\eta k}{\eta k} \cdot \frac{M-\Delta^2}{\Delta^2} \right)^{\frac{1}{2}}. \quad (63)$$

Since (63) holds for all $k \in [d-1]$,

$$\begin{aligned} \frac{\lambda_1}{\lambda_d} &= \prod_{k \in [d-1]} \frac{\lambda_{d-k}}{\lambda_{d-k+1}} \leq \prod_{k \in [d-1]} \left(\frac{d-k+\eta k}{\eta k} \cdot \frac{M-\Delta^2}{\Delta^2} \right)^{\frac{1}{2}} \\ &= \left(\frac{M-\Delta^2}{\Delta^2} \right)^{\frac{d-1}{2}} \prod_{k \in [d-1]} \left(\frac{d-k+\eta k}{\eta k} \right)^{\frac{1}{2}} \\ &= \left(\frac{M-\Delta^2}{\Delta^2} \right)^{\frac{d-1}{2}} \prod_{k=1}^{\lfloor \frac{d}{1+\eta} \rfloor} \left(\frac{d-k+\eta k}{\eta k} \right)^{\frac{1}{2}} \prod_{k=\lfloor \frac{d}{1+\eta} \rfloor + 1}^{d-1} \left(\frac{d-k+\eta k}{\eta k} \right)^{\frac{1}{2}} \\ &\leq \left(\frac{M-\Delta^2}{\Delta^2} \right)^{\frac{d-1}{2}} \prod_{k=1}^{\lfloor \frac{d}{1+\eta} \rfloor} \left(\frac{2(d-k)}{\eta k} \right)^{\frac{1}{2}} \prod_{k=\lfloor \frac{d}{1+\eta} \rfloor + 1}^{d-1} \left(\frac{2}{\eta} \right)^{\frac{1}{2}} \\ &= \left(\frac{2(M-\Delta^2)}{\eta \Delta^2} \right)^{\frac{d-1}{2}} \prod_{k=1}^{\lfloor \frac{d}{1+\eta} \rfloor} \left(\frac{d-k}{k} \right)^{\frac{1}{2}} = \left(\frac{2(M-\Delta^2)}{\eta \Delta^2} \right)^{\frac{d-1}{2}} \left(\frac{d-1}{\lfloor \frac{d}{1+\eta} \rfloor} \right)^{\frac{1}{2}} \\ &\leq \left(\frac{4(M-\Delta^2)}{\eta \Delta^2} \right)^{\frac{d-1}{2}} \leq \left(\frac{4M}{\eta \Delta^2} \right)^{\frac{d-1}{2}}, \end{aligned} \quad (64)$$

where the fourth line uses $\frac{d-k+\eta k}{\eta k} \leq \frac{2(d-k)}{\eta k}$ for $k \leq \frac{d}{1+\eta}$ and $\frac{d-k+\eta k}{\eta k} \leq 2 \leq \frac{2}{\eta}$ for $k \geq \frac{d}{1+\eta}$ and the sixth line uses $\binom{a}{b} \leq 2^a$.

Let $j \in [n]$ such that $\mathbf{t}_j^* = \|\mathbf{t}^*\|_\infty$, and note that the largest eigenvalue of $\mathbf{Z}(\mathbf{t}^*)$ is $\frac{1}{\lambda_d^2}$. Thus

$$\frac{1}{\lambda_d^2} = \max_{\|\mathbf{x}\|_2=1} \sum_{i \in [n]} \exp(\mathbf{t}_i^*) \langle \mathbf{a}_i, \mathbf{x} \rangle^2 \geq \max_{\|\mathbf{x}\|_2=1} \exp(\mathbf{t}_j^*) \langle \mathbf{a}_j, \mathbf{x} \rangle^2 = \|\mathbf{a}_j\|_2^2 \exp(\mathbf{t}_j^*) \geq \mu \exp(\|\mathbf{t}^*\|_\infty). \quad (65)$$

Let $\ell \in [n]$ such that $\mathbf{t}_\ell^* = 0$. Since \mathbf{t}^* minimizes f , we have

$$\mathbf{c}_\ell = \text{Tr}(\mathbf{M}_\ell(\mathbf{t}^*)) = \|\mathbf{R}\mathbf{a}_\ell\|_2^2 \leq \lambda_1^2 \|\mathbf{a}_\ell\|_2^2 \leq M \lambda_1^2 \quad (66)$$

by Fact 2. Combining (65) and (66) gives

$$\frac{\lambda_1^2}{\lambda_d^2} \geq \frac{\mu \mathbf{c}_{\min}}{M} \exp(\|\mathbf{t}^*\|_\infty),$$

and combining with (64) and rearranging gives

$$\|\mathbf{t}^*\|_\infty \leq \log \left(\frac{M}{\mu \mathbf{c}_{\min}} \left(\frac{4M}{\eta \Delta^2} \right)^{d-1} \right).$$

Since f is invariant to translations by $\mathbf{1}_n$ and $\min_{i \in [n]} \mathbf{t}_i^* = 0$ by assumption, we can shift \mathbf{t}^* to obtain a minimizer that has extreme coordinates averaging to 0, which gives the result. \square

With Lemma 32 in hand, we must show $\frac{M}{\mu}$ and $\frac{1}{\Delta}$ are polynomially bounded for an appropriate choice of η , in the smoothed setting. The bulk of our remaining analysis establishes this result.

Remark 3. *Our strategy in this section is to restrict η to be a constant, e.g., $\eta = 0.1$, in which case our goal is to show that at most $\frac{0.9k}{d}n$ of the vectors in $\{\mathbf{a}_i\}_{i \in [n]}$ lie close to any k -dimensional subspace. If, for instance, $\frac{n}{d} < 1.1$, this is clearly impossible, because choosing $k = 1$ implies that not even a single vector lies close to any 1-dimensional subspace, which is false just by taking $\text{Span}(\mathbf{a}_i)$ for any $i \in [n]$. Thus, the restriction $\frac{n}{d} \geq 1.1$ (or more generally, a constant bounded away from 1) is somewhat inherent in the regime $\eta = \Omega(1)$. It is possible that our strategy can be modified to extend to even smaller n , but for simplicity, we focus on the setting of Assumption 2.*

To frame the rest of the section, we provide a helper lemma relating the condition in Definition 9 to appropriate submatrices having small singular values.

Lemma 33. *Let $\mathbf{A} = \{\mathbf{a}_i\}_{i \in [m]} \in \mathbb{R}^{d \times m}$, $k < m$, and $\Delta > 0$. Suppose there exists a k -dimensional subspace E of \mathbb{R}^d such that $\|\mathbf{a}_i - \mathbf{\Pi}_E \mathbf{a}_i\|_2 \leq \Delta$ for all $i \in [m]$. Then $\sigma_{k+1}(\mathbf{A}) \leq \sqrt{m}\Delta$.*

Proof. Let $\mathbf{V} \in \mathbb{R}^{d \times (d-k)}$ have columns that form an orthonormal basis for E^\perp . By assumption,

$$\|\mathbf{V}\mathbf{V}^\top \mathbf{a}_i\|_2^2 = \|\mathbf{V}^\top \mathbf{a}_i\|_2^2 \leq \Delta^2 \text{ for all } i \in [m].$$

By the min-max theorem,

$$\begin{aligned} \sigma_{k+1}(\mathbf{A}) &= \min_{\substack{E \subseteq \mathbb{R}^d \\ \dim(E) = d-k}} \max_{\substack{\mathbf{x} \in E \\ \|\mathbf{x}\|_2=1}} \|\mathbf{A}^\top \mathbf{x}\|_2 \\ &\leq \max_{\substack{\mathbf{x} \in \text{Span}(\mathbf{V}) \\ \|\mathbf{x}\|_2=1}} \|\mathbf{A}^\top \mathbf{x}\|_2 = \max_{\substack{\mathbf{v} \in \mathbb{R}^{d-k} \\ \|\mathbf{v}\|_2=1}} \|\mathbf{A}^\top \mathbf{V}\mathbf{v}\|_2. \end{aligned}$$

Observe that $\mathbf{A}^\top \mathbf{V}$ has rows $\{\mathbf{V}^\top \mathbf{a}_i\}_{i \in [m]}$, so by the Cauchy–Schwarz inequality,

$$\|\mathbf{A}^\top \mathbf{V}\mathbf{v}\|_2^2 = \sum_{i \in [m]} \langle \mathbf{V}^\top \mathbf{a}_i, \mathbf{v} \rangle^2 \leq \sum_{i \in [m]} \Delta^2 = m\Delta^2$$

for all $\mathbf{v} \in \mathbb{R}^{d-k}$ with $\|\mathbf{v}\|_2 = 1$. It follows that $\sigma_{k+1}(\mathbf{A}) \leq \sqrt{m}\Delta$. \square

In Sections 5.2 and 5.3, we show that with high probability, the conclusion of Lemma 33 is violated for all appropriately-sized smoothed submatrices. This shows that the premise of Lemma 33 is also violated, which we establish in Section 5.4, so that $\mathbf{c} = \frac{d}{n}\mathbf{1}_n$ is indeed (η, Δ) -deep.

5.2 Conditioning of wide and near-square smoothed matrices

Throughout this section, we let

$$\eta := 1 - \frac{1}{\sqrt{C}}, \quad K := \sqrt[3]{C}, \quad k \in [d-1], \quad \text{and } m := \left\lceil \frac{(1-\eta)kn}{d} \right\rceil.$$

Remark 4. We note that these definitions imply $0 < \eta < 1 - \frac{1}{C}$ and $1 < K < (1 - \eta)C$. Indeed, nothing in our analysis relies on our choices of η and K other than the fact that they are constants that satisfy these inequalities, which limit our analysis in the following places.

- In Lemma 34, we require $(1 - \eta)C > 1$ (equivalently, $\eta < 1 - \frac{1}{C}$) so that $m - k = \Omega(m)$.
- In Lemma 35, we require $K < (1 - \eta)C$ so that $d - k = \Omega(d)$.
- In Lemma 35 and Lemma 40, we require $K > 1$, and in Theorem 3, we require $\eta > 0$.

We first use the following results from the literature to establish tail bounds for the singular values of smoothed matrices in the cases $m \leq d$ and $d < m \leq Kd$, i.e., m that are at most a constant factor larger than d . In Section 5.3, we use a different argument to handle $m > Kd$.

Fact 3 (Theorem 1.2, [Sza91]). *Let $\mathbf{G} \in \mathbb{R}^{d \times d}$ have entries $\sim_{\text{i.i.d.}} \mathcal{N}(0, 1)$. Then for all $j \in [d]$,*

$$\Pr \left[\sigma_{d-j+1}(\mathbf{G}) < \frac{\alpha j}{\sqrt{d}} \right] \leq \left(\sqrt{2e}\alpha \right)^{j^2}.$$

Fact 4 (Theorem 2.4, [BKMS21]). *Let $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{d \times d}$ such that $\sigma_i(\mathbf{M}) \geq \sigma_i(\mathbf{N})$ for all $i \in [d]$. Then for every $t \geq 0$, there exists a joint distribution on pairs of matrices $(\mathbf{G}, \mathbf{H}) \in \mathbb{R}^{d \times d} \times \mathbb{R}^{d \times d}$ such that the marginals \mathbf{G} and \mathbf{H} have entries $\sim_{\text{i.i.d.}} \mathcal{N}(0, 1)$ and*

$$\Pr[\sigma_i(\mathbf{M} + t\mathbf{G}) \geq \sigma_i(\mathbf{N} + t\mathbf{H})] = 1 \text{ for all } i \in [d].$$

We note that these results imply tail bounds for the singular values of square smoothed matrices: given $\sigma > 0$, $\mathbf{A} \in \mathbb{R}^{d \times d}$, and $\mathbf{G} \in \mathbb{R}^{d \times d}$ with entries $\sim_{\text{i.i.d.}} \mathcal{N}(0, 1)$, we let $\mathbf{M} \leftarrow \mathbf{A}$, $\mathbf{N} \leftarrow \mathbf{0}$, $t \leftarrow \sigma$, and (\mathbf{G}, \mathbf{H}) have the distribution in Fact 4. Then by Fact 4 and Fact 3,

$$\begin{aligned} \Pr \left[\sigma_{d-j+1}(\mathbf{A} + \sigma\mathbf{G}) < \frac{\alpha\sigma j}{\sqrt{d}} \right] &\leq \Pr \left[\sigma_{d-j+1}(\sigma\mathbf{H}) < \frac{\alpha\sigma j}{\sqrt{d}} \right] \\ &= \Pr \left[\sigma_{d-j+1}(\mathbf{H}) < \frac{\alpha j}{\sqrt{d}} \right] \leq \left(\sqrt{2e}\alpha \right)^{j^2}. \end{aligned} \tag{67}$$

Lemma 34. *Under Assumption 2, let $\delta \in (0, 1)$, $\sigma \in (0, \frac{1}{2})$, $\eta := 1 - \frac{1}{\sqrt{C}}$, $k \in [d-1]$, $m := \lceil \frac{(1-\eta)kn}{d} \rceil$, $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{G} \in \mathbb{R}^{n \times d}$ have entries $\sim_{\text{i.i.d.}} \mathcal{N}(0, \sigma^2)$, $\tilde{\mathbf{A}} := \mathbf{A} + \mathbf{G}$, and suppose $m \leq d$. Then for any $S \subseteq [n]$ with $|S| = m$,*

$$\Pr \left[\sigma_{k+1}(\tilde{\mathbf{A}}_{S:}) \leq \sqrt{m}\Delta \right] \leq \frac{\delta}{(d-1)\binom{n}{m}}, \text{ where } \Delta = \left(\frac{\delta\sigma}{n} \right)^{O(1)}.$$

Proof. Remove any $d - m$ columns of $\tilde{\mathbf{A}}_{S:}$ to obtain $\mathbf{B} \in \mathbb{R}^{m \times m}$. We have $\sigma_{k+1}(\tilde{\mathbf{A}}_{S:}) \geq \sigma_{k+1}(\mathbf{B})$ by the min-max theorem:

$$\sigma_{k+1}(\tilde{\mathbf{A}}_{S:}) = \max_{\substack{E \subseteq \mathbb{R}^d \\ \dim(E) = k+1}} \min_{\substack{\mathbf{x} \in E \\ \|\mathbf{x}\|_2 = 1}} \left\| \tilde{\mathbf{A}}_{S:} \mathbf{x} \right\|_2 \geq \max_{\substack{E \subseteq \mathbb{R}^m \\ \dim(E) = k+1}} \min_{\substack{\mathbf{x} \in E \\ \|\mathbf{x}\|_2 = 1}} \|\mathbf{B} \mathbf{x}\|_2 = \sigma_{k+1}(\mathbf{B}).$$

Then, by (67) with $d \leftarrow m$ and $j \leftarrow m - k$,

$$\Pr \left[\boldsymbol{\sigma}_{k+1}(\tilde{\mathbf{A}}_{S:}) \leq \frac{\alpha\sigma(m-k)}{\sqrt{m}} \right] \leq \Pr \left[\boldsymbol{\sigma}_{k+1}(\mathbf{B}) \leq \frac{\alpha\sigma(m-k)}{\sqrt{m}} \right] \leq \left(\sqrt{2e}\alpha \right)^{(m-k)^2}. \quad (68)$$

Let $C' := \frac{1}{2} \left(1 - \frac{1}{(1-\eta)C} \right) > 0$, and note that

$$m - k \geq \left(\frac{(1-\eta)n}{d} - 1 \right) k = \left(1 - \frac{d}{(1-\eta)n} \right) \left(\frac{(1-\eta)kn}{d} \right) \geq C'm.$$

Setting $\alpha = \frac{m\Delta}{\sigma(m-k)} \leq \frac{\Delta}{C'\sigma}$ in (68),

$$\Pr \left[\boldsymbol{\sigma}_{k+1}(\tilde{\mathbf{A}}_{S:}) \leq \sqrt{m}\Delta \right] \leq \left(\sqrt{2e}\alpha \right)^{(m-k)^2} \leq \left(\frac{\sqrt{2e}\Delta}{C'\sigma} \right)^{(m-k)^2}.$$

Now, we can set $\Delta = \frac{C'}{\sqrt{2e}} \left(\frac{\delta\sigma}{n} \right)^{\frac{2}{C'}} \leq \frac{C'\sigma}{\sqrt{2e}} \left(\frac{\delta}{n} \right)^{\frac{2}{C'}}$ to give

$$\left(\frac{\sqrt{2e}\Delta}{C'\sigma} \right)^{(m-k)^2} \leq \left(\frac{\delta}{n} \right)^{\frac{2(m-k)^2}{C'}} \leq \left(\frac{\delta}{n} \right)^{2m(m-k)} \leq \left(\frac{\delta}{n} \right)^{m+1} \leq \frac{\delta^{m+1}}{n \binom{n}{m}} \leq \frac{\delta}{(d-1) \binom{n}{m}},$$

which establishes the claim. \square

Lemma 35. *In the setting of Lemma 34, the result holds if we suppose instead that $d < m \leq Kd$, where $K := \sqrt[3]{C}$.*

Proof. Similarly to the proof of Lemma 34, remove any $m - d$ rows of $\tilde{\mathbf{A}}_{S:}$ to obtain $\mathbf{B} \in \mathbb{R}^{d \times d}$. Then

$$\Pr \left[\boldsymbol{\sigma}_{k+1}(\tilde{\mathbf{A}}_{S:}) \leq \frac{\alpha\sigma(d-k)}{\sqrt{d}} \right] \leq \Pr \left[\boldsymbol{\sigma}_{k+1}(\mathbf{B}) \leq \frac{\alpha\sigma(d-k)}{\sqrt{d}} \right] \leq \left(\sqrt{2e}\alpha \right)^{(d-k)^2}. \quad (69)$$

Since $m \leq Kd$, we have $\frac{(1-\eta)kn}{d} \leq Kd$, which implies $k \leq \frac{Kd^2}{(1-\eta)n} \leq \frac{Kd}{(1-\eta)C} < d$. Thus $d - k \geq K'd$, where $K' := 1 - \frac{K}{(1-\eta)C} > 0$. Setting $\alpha = \frac{\sqrt{md}\Delta}{\sigma(d-k)} \leq \frac{\sqrt{K}\Delta}{K'\sigma}$ in (69),

$$\Pr \left[\boldsymbol{\sigma}_{k+1}(\tilde{\mathbf{A}}_{S:}) \leq \sqrt{m}\Delta \right] \leq \left(\sqrt{2e}\alpha \right)^{(d-k)^2} \leq \left(\frac{\sqrt{2eK}\Delta}{K'\sigma} \right)^{(d-k)^2}.$$

Now, we can set $\Delta = \frac{K'}{\sqrt{2eK}} \left(\frac{\delta\sigma}{n} \right)^{\frac{2K}{K'}} \leq \frac{K'\sigma}{\sqrt{2eK}} \left(\frac{\delta}{n} \right)^{\frac{2K}{K'}}$ to give

$$\left(\frac{\sqrt{2eK}\Delta}{K'\sigma} \right)^{(d-k)^2} \leq \left(\frac{\delta}{n} \right)^{\frac{2K(d-k)^2}{K'}} \leq \left(\frac{\delta}{n} \right)^{2Kd} \leq \left(\frac{\delta}{n} \right)^{m+1} \leq \frac{\delta^{m+1}}{n \binom{n}{m}} \leq \frac{\delta}{(d-1) \binom{n}{m}},$$

which establishes the claim. \square

5.3 Conditioning of tall smoothed matrices

In this section we provide tools for lower bounding the smallest singular value of a random $\Omega(d) \times d$ smoothed matrix $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{G}$, where \mathbf{G} has entries $\sim_{\text{i.i.d.}} \mathcal{N}(0, \sigma^2)$. Specifically we provide estimates, for sufficiently small α , on the quantity

$$\Pr \left[\sigma_d(\tilde{\mathbf{A}}) < \alpha \right]. \quad (70)$$

We first use the following standard result bounding $\|\tilde{\mathbf{A}}\|_{\text{op}}$.

Lemma 36. *Let $\sigma \in (0, 1)$, $m \geq d$, $\mathbf{A} \in \mathbb{R}^{m \times d}$ have rows $\{\mathbf{a}_i\}_{i \in [m]}$ such that $\|\mathbf{a}_i\|_2 = 1$ for all $i \in [m]$, $\mathbf{G} \in \mathbb{R}^{m \times d}$ have entries $\sim_{\text{i.i.d.}} \mathcal{N}(0, \sigma^2)$, and $\tilde{\mathbf{A}} := \mathbf{A} + \mathbf{G}$. Then there exists a constant $C_{\text{op}} > 0$ such that for all $\delta \in (0, 1)$,*

$$\Pr \left[\|\tilde{\mathbf{A}}\|_{\text{op}} > C_{\text{op}} \sqrt{m + \log \left(\frac{1}{\delta} \right)} \right] \leq \delta.$$

Proof. By Theorem 4.4.5, [Ver24], we have that for some constant $C_{\text{op}} > 2$,

$$\Pr \left[\|\mathbf{G}\|_{\text{op}} > \frac{C_{\text{op}}}{2} \sqrt{m + \log \left(\frac{1}{\delta} \right)} \right] \leq \delta,$$

where we used $\sigma \leq 1$. The conclusion follows as $\|\mathbf{A}\|_{\text{op}} \leq \|\mathbf{A}\|_{\text{F}} \leq \sqrt{m}$. \square

We also require the definition of an ϵ -net and a standard bound on its size.

Definition 10. *Let $S \subseteq \mathbb{R}^d$, $\mathcal{N} \subset S$ be finite, and $\epsilon \in (0, 1)$. We say that \mathcal{N} is an ϵ -net of S if*

$$\sup_{\mathbf{u} \in S} \min_{\mathbf{v} \in \mathcal{N}} \|\mathbf{v} - \mathbf{u}\|_2 \leq \epsilon.$$

Fact 5 (Corollary 4.2.13, [Ver24]). *Let $\partial \mathbb{B}_2(1)$ denote the boundary of the unit norm ball in \mathbb{R}^d . For all $\epsilon \in (0, 1)$, there exists an ϵ -net of $\partial \mathbb{B}_2(1)$ with $|\mathcal{N}| \leq (\frac{3}{\epsilon})^d$.*

We next observe that it suffices to provide estimates on a net, given an operator norm bound.

Lemma 37. *Let $m \geq d$, $\epsilon \in (0, 1)$, and \mathcal{N} be an ϵ -net of $\partial \mathbb{B}_2(1) \subset \mathbb{R}^d$, and suppose that $\tilde{\mathbf{A}} \in \mathbb{R}^{m \times d}$ satisfies $\|\tilde{\mathbf{A}}\|_{\text{op}} \leq \rho$. Then $\sigma_d(\tilde{\mathbf{A}}) \geq \min_{\mathbf{v} \in \mathcal{N}} \|\tilde{\mathbf{A}}\mathbf{v}\|_2 - \epsilon\rho$.*

Proof. Let \mathbf{u} realize $\sigma_d(\tilde{\mathbf{A}})$ in the definition $\sigma_d(\tilde{\mathbf{A}}) = \min_{\mathbf{u} \in \partial \mathbb{B}_2(1)} \|\tilde{\mathbf{A}}\mathbf{u}\|_2$. Then if we define $\mathbf{v} := \arg \min_{\mathbf{v} \in \mathcal{N}} \|\mathbf{u} - \mathbf{v}\|_2$, the result follows from the triangle inequality:

$$\|\tilde{\mathbf{A}}\mathbf{u}\|_2 \geq \|\tilde{\mathbf{A}}\mathbf{v}\|_2 - \|\tilde{\mathbf{A}}\|_{\text{op}} \|\mathbf{u} - \mathbf{v}\|_2 \geq \min_{\mathbf{v} \in \mathcal{N}} \|\tilde{\mathbf{A}}\mathbf{v}\|_2 - \epsilon\rho.$$

\square

Finally, we provide tail bounds on the contraction given by $\tilde{\mathbf{A}}$ on a single fixed vector.

Lemma 38. *In the setting of Lemma 36, let $\mathbf{v} \in \mathbb{R}^d$ have $\|\mathbf{v}\|_2 = 1$. Then,*

$$\Pr \left[\|\tilde{\mathbf{A}}\mathbf{v}\|_2 < \alpha \right] \leq \left(\frac{\alpha}{\sigma} \right)^m \text{ for all } \alpha \in (0, 1).$$

Proof. Observe that if $\|\tilde{\mathbf{A}}\mathbf{v}\|_2 \leq \alpha$, then every coordinate of $\tilde{\mathbf{A}}\mathbf{v}$ is bounded by α . Each coordinate of $\tilde{\mathbf{A}}\mathbf{v}$ is distributed independently as $\mathcal{N}(\langle \mathbf{a}_i, \mathbf{v} \rangle, \sigma^2)$. Now the claim follows because for all $i \in [m]$,

$$\begin{aligned} \Pr_{\xi \sim \mathcal{N}(\langle \mathbf{a}_i, \mathbf{v} \rangle, \sigma^2)} [|\xi| < \alpha] &= \Pr_{\xi \sim \mathcal{N}(\frac{1}{\sigma} \langle \mathbf{a}_i, \mathbf{v} \rangle, 1)} \left[\left| \xi \right| < \frac{\alpha}{\sigma} \right] \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\frac{\alpha}{\sigma}}^{\frac{\alpha}{\sigma}} \exp \left(-\frac{(\xi - \frac{1}{\sigma} \langle \mathbf{a}_i, \mathbf{v} \rangle)^2}{2} \right) d\xi \leq \frac{\alpha}{\sigma}. \end{aligned}$$

□

We can now prove our main tail bound on $\sigma_d(\tilde{\mathbf{A}})$.

Lemma 39. *In the setting of Lemma 36, suppose that $m \geq Kd$ for a constant $K > 1$. Then there exists a constant $\beta > 0$ such that*

$$\Pr \left[\sigma_d(\tilde{\mathbf{A}}) < \alpha \right] \leq 2 \left(\frac{2\alpha}{\sigma} \right)^{\frac{(K-1)m}{2K}} \text{ for all } \alpha \in \left(0, \frac{\sigma^{\frac{K+3}{K-1}}}{\beta m^{\frac{2}{K-1}}} \right).$$

Proof. Let $\delta := 2 \left(\frac{2\alpha}{\sigma} \right)^{\frac{(K-1)m}{2K}}$. By Lemma 36, there exists $C_{\text{op}} > 0$ such that

$$\Pr \left[\|\tilde{\mathbf{A}}\|_{\text{op}} > \rho \right] \leq \frac{\delta}{2} \text{ for } \rho := C_{\text{op}} \sqrt{m + \log \left(\frac{2}{\delta} \right)}.$$

Let $L > 0$ be a constant such that $\sqrt{\log(\frac{c}{2})} \leq Lc^{\frac{K-1}{4}}$ for all $c \geq 2$, and let

$$\beta := \max \left(2e, \left(2^{\frac{K+1}{2}} \cdot 6C_{\text{op}}L \right)^{\frac{4}{K-1}} \right) \geq 2e.$$

Then for the stated range of α ,

$$\alpha \leq \frac{\sigma}{2e} \implies \rho = C_{\text{op}} \sqrt{m + \frac{(K-1)m}{2K} \log \left(\frac{\sigma}{2\alpha} \right)} \leq 2C_{\text{op}} \sqrt{m \log \left(\frac{\sigma}{2\alpha} \right)}.$$

Let $\epsilon := \frac{\alpha}{\rho}$, and let \mathcal{N} be an ϵ -net of $\partial\mathbb{B}_2(1) \in \mathbb{R}^d$ with size $|\mathcal{N}| \leq (\frac{3}{\epsilon})^d$, as guaranteed by Fact 5. Then by taking a union bound over Lemma 38 applied to each $\mathbf{v} \in \mathcal{N}$,

$$\begin{aligned} \Pr \left[\min_{\mathbf{v} \in \mathcal{N}} \|\tilde{\mathbf{A}}\mathbf{v}\|_2 < 2\alpha \right] &\leq |\mathcal{N}| \left(\frac{2\alpha}{\sigma} \right)^m \leq \left(\frac{6C_{\text{op}} \sqrt{m \log(\frac{\sigma}{2\alpha})}}{\alpha} \right)^{\frac{m}{K}} \left(\frac{2\alpha}{\sigma} \right)^m \\ &= \left(\frac{6C_{\text{op}} \sqrt{m \log(\frac{\sigma}{2\alpha})}}{\alpha} \right)^{\frac{m}{K}} \left(\frac{2\alpha}{\sigma} \right)^{\frac{(K+1)m}{2K}} \left(\frac{2\alpha}{\sigma} \right)^{\frac{(K-1)m}{2K}} \end{aligned}$$

$$\begin{aligned}
&\leq \left(\frac{2^{\frac{K+1}{2}} \cdot 6C_{\text{op}} \sqrt{m \log(\frac{\sigma}{2\alpha})}}{\sigma^{\frac{K+1}{2}}} \cdot \frac{\sigma^{\frac{K+3}{4}} \alpha^{\frac{K-1}{4}}}{2^{\frac{K+1}{2}} \cdot 6C_{\text{op}} L \sqrt{m}} \right)^{\frac{m}{K}} \left(\frac{2\alpha}{\sigma} \right)^{\frac{(K-1)m}{2K}} \\
&\leq \left(\frac{2^{\frac{K+1}{2}} \cdot 6C_{\text{op}} \sqrt{m \log(\frac{\sigma}{2\alpha})}}{\sigma^{\frac{K+1}{2}}} \cdot \frac{\sigma^{\frac{K+1}{2}}}{2^{\frac{K+1}{2}} \cdot 6C_{\text{op}} \sqrt{m \log(\frac{\sigma}{2\alpha})}} \right)^{\frac{m}{K}} \left(\frac{2\alpha}{\sigma} \right)^{\frac{(K-1)m}{2K}} \\
&= \left(\frac{2\alpha}{\sigma} \right)^{\frac{(K-1)m}{2K}} = \frac{\delta}{2},
\end{aligned}$$

where the third line uses

$$\alpha^{\frac{K-1}{4}} \leq \frac{\sigma^{\frac{K+3}{4}}}{2^{\frac{K+1}{2}} \cdot 6C_{\text{op}} L \sqrt{m}} \implies \alpha^{\frac{K-1}{2}} \leq \frac{\sigma^{\frac{K+3}{4}} \alpha^{\frac{K-1}{4}}}{2^{\frac{K+1}{2}} \cdot 6C_{\text{op}} L \sqrt{m}}$$

for the stated range of α and the fourth line uses $\sqrt{\log(\frac{c}{2})} \leq Lc^{\frac{K-1}{4}}$ for $c \geq 2$. The claim follows from a union bound on the above two events and Lemma 37. \square

Applying Lemma 39 then gives our extension to tall matrices.

Lemma 40. *In the setting of Lemma 34, the result holds if we suppose instead that $m > Kd$, where $K := \sqrt[3]{C}$, and in addition that \mathbf{A} has rows $\{\mathbf{a}_i\}_{i \in [n]}$ satisfying $\|\mathbf{a}_i\|_2 = 1$ for all $i \in [n]$.*

Proof. Let β be the constant in Lemma 39, and let $\alpha = \sqrt{m}\Delta$, where

$$\Delta = \frac{1}{2} \left(\frac{\delta\sigma}{\beta n} \right)^{\frac{4K}{K-1}+1} \leq \frac{\sigma}{2\sqrt{m}} \left(\frac{\delta\sigma}{\beta n} \right)^{\frac{4K}{K-1}} \in \left(0, \frac{\sigma^{\frac{K+3}{K-1}}}{\beta m^{\frac{2}{K-1}+\frac{1}{2}}} \right).$$

By Lemma 39,

$$\begin{aligned}
\Pr \left[\boldsymbol{\sigma}_{k+1}(\tilde{\mathbf{A}}_{S:}) \leq \sqrt{m}\Delta \right] &\leq \Pr \left[\boldsymbol{\sigma}_d(\tilde{\mathbf{A}}_{S:}) \leq \sqrt{m}\Delta \right] \leq 2 \left(\left(\frac{\delta\sigma}{\beta n} \right)^{\frac{4K}{K-1}} \right)^{\frac{(K-1)m}{2K}} \\
&\leq \left(\frac{\delta}{n} \right)^{2m} \leq \left(\frac{\delta}{n} \right)^{m+1} \leq \frac{\delta^{m+1}}{n \binom{n}{m}} \leq \frac{\delta}{(d-1) \binom{n}{m}},
\end{aligned}$$

which establishes the claim. \square

5.4 Assumption 1 for smoothed matrices

We can now put together the previous results to give a diameter bound for smoothed matrices. To begin, we show simple norm bounds on the rows of a smoothed matrix.

Lemma 41. *Under Assumption 2, let $\delta \in (0, 1)$, $\frac{1}{\sigma} \geq 10(d + \log(\frac{n}{\delta}))$, $\mathbf{A} \in \mathbb{R}^{n \times d}$ have rows $\{\mathbf{a}_i\}_{i \in [n]}$ such that $\|\mathbf{a}_i\|_2 = 1$ for all $i \in [n]$, $\mathbf{G} \in \mathbb{R}^{n \times d}$ have entries $\sim_{\text{i.i.d.}} \mathcal{N}(0, \sigma^2)$, and $\tilde{\mathbf{A}} := \mathbf{A} + \mathbf{G}$ have rows $\{\tilde{\mathbf{a}}_i\}_{i \in [n]}$. Then with probability $\geq 1 - \delta$, $\frac{1}{6} \leq \|\tilde{\mathbf{a}}_i\|_2^2 \leq 2$ for all $i \in [n]$.*

Proof. By using the inequalities

$$\frac{1}{2} \|\mathbf{a}_i\|_2^2 - \|\mathbf{g}_i\|_2^2 \leq \|\mathbf{a}_i + \mathbf{g}_i\|_2^2 \leq \frac{3}{2} \|\mathbf{a}_i\|_2^2 + 3 \|\mathbf{g}_i\|_2^2,$$

it is enough to show that for all $i \in [n]$, the probability that $\|\mathbf{g}_i\|_2^2 \geq \frac{1}{6}$ is bounded by $\frac{\delta}{n}$. This follows from a standard χ^2 tail bound, e.g., Lemma 1, [LM00], for our choice of σ . \square

It remains to show that $\frac{d}{n}\mathbf{1}_n$ is deep inside the basis polytope of the rows of $\tilde{\mathbf{A}}$, so that we can use Lemma 32 to obtain a diameter bound for minimizing Barthe's objective.

Lemma 42. *In the setting of Lemma 41, $\frac{d}{n}\mathbf{1}_n$ is (η, Δ) -deep inside the basis polytope of the rows of $\tilde{\mathbf{A}}$ with probability $\geq 1 - \delta$, where $\eta := 1 - \frac{1}{\sqrt{C}}$ and $\Delta = \left(\frac{\delta\sigma}{n}\right)^{O(1)}$.*

Proof. Let $k \in [d-1]$. By Lemma 33, if some $m := \lceil \frac{(1-\eta)kn}{d} \rceil$ rows of $\tilde{\mathbf{A}}$ indexed by S violate the condition for (η, Δ) -deepness, then $\sigma_{k+1}(\tilde{\mathbf{A}}_{S:}) \leq \sqrt{m}\Delta$. By Lemma 34, Lemma 35, and Lemma 40,

$$\Pr \left[\sigma_{k+1}(\tilde{\mathbf{A}}_{S:}) \leq \sqrt{m}\Delta \right] \leq \frac{\delta}{(d-1)\binom{n}{m}}$$

for any $S \subseteq [n]$ with $|S| = m$, in every range of $k \in [d-1]$. By a union bound over all S , the failure probability is at most $\frac{\delta}{d-1}$. The result follows by a union bound over all $k \in [d-1]$. \square

At this point, we have all the tools necessary to prove Theorem 3.

Theorem 3. *Let $\mathbf{A} \in \mathbb{R}^{n \times d}$ have rows $\{\mathbf{a}_i\}_{i \in [n]}$ such that $\|\mathbf{a}_i\|_2 = 1$ for all $i \in [n]$, let $\mathbf{c} := \frac{d}{n}\mathbf{1}_n$, let $\delta \in (0, 1)$, and let $\sigma \in (0, \frac{\delta}{10nd})$. Let $\tilde{\mathbf{A}} := \mathbf{A} + \mathbf{G}$, where $\mathbf{G} \in \mathbb{R}^{n \times d}$ has entries $\sim_{\text{i.i.d.}} \mathcal{N}(0, \sigma^2)$. Then with probability $\geq 1 - \delta$, if $n > Cd$ where C is any constant larger than 1, Assumption 1 holds for Barthe's objective f defined with respect to $(\tilde{\mathbf{A}}, \mathbf{c})$, where*

$$\log(\kappa) = O \left(d \log \left(\frac{1}{\sigma} \right) \right).$$

Proof. By Lemma 41, in the relevant range of σ , the conclusion $\frac{1}{6} \leq \|\tilde{\mathbf{a}}_i\|_2^2 \leq 2$ holds for all $i \in [n]$ with probability $\geq 1 - \frac{\delta}{2}$. Moreover, let $\Delta = \left(\frac{\delta\sigma}{n}\right)^{O(1)}$ so that $\frac{d}{n}\mathbf{1}_n$ is (η, Δ) -deep inside the basis polytope of the rows of $\tilde{\mathbf{A}}$ with probability $\geq 1 - \frac{\delta}{2}$ by Lemma 42. By a union bound on these events, with probability $\geq 1 - \delta$, we can apply Lemma 32 and conclude that there exists $\mathbf{t}^* \in \arg \min_{\mathbf{t} \in \mathbb{R}^n} f(\mathbf{t})$ satisfying

$$\|\mathbf{t}^*\|_\infty \leq \frac{1}{2} \log \left(\frac{12n}{d} \left(\frac{8}{\eta\Delta^2} \right)^{d-1} \right) = O \left(d \log \left(\frac{1}{\sigma} \right) \right).$$

\square

5.5 Extension to non-uniform \mathbf{c}

Our smoothed diameter bound in Theorem 3 is stated with respect to uniform marginals $\mathbf{c} = \frac{d}{n}\mathbf{1}_n$. However, the analysis in this section can be straightforwardly extended to hold for \mathbf{c} with nonuniform entries by a reduction, as long as \mathbf{c} is sufficiently bounded away from $\mathbf{1}_n$ entrywise.

Corollary 1. *In the setting of Theorem 3, the result holds if we suppose instead that $\mathbf{c} \in (0, 1]^n$ satisfies $\|\mathbf{c}\|_1 = d$ and $\mathbf{c} \leq c \cdot \frac{d}{n}\mathbf{1}_n$ entrywise, where $1 < c < C \leq \frac{n}{d}$ for constants c, C .*

Proof. Our proof of Lemma 42 shows that with probability $\geq 1 - \delta$ in the setting of Theorem 3, $\frac{d}{n}\mathbf{1}_n$ is (η, Δ) -deep for a constant η arbitrarily close to $1 - \frac{1}{C}$, where $C \leq \frac{n}{d}$ (see Remark 4). However, this also implies that for all $k \in [d - 1]$ and k -dimensional subspaces E , recalling Definition 9,

$$\sum_{i \in [n]} \mathbf{c}_i \mathbb{I}_{\|\mathbf{a}_i - \mathbf{\Pi}_E \mathbf{a}_i\|_2 \leq \Delta} \leq c \sum_{i \in [n]} \frac{d}{n} \mathbb{I}_{\|\mathbf{a}_i - \mathbf{\Pi}_E \mathbf{a}_i\|_2 \leq \Delta} \leq c(1 - \eta)k = (1 - (1 - c(1 - \eta)))k.$$

Thus, we have shown that \mathbf{c} is also $(1 - c(1 - \eta), \Delta)$ -deep. Since η can be arbitrarily close to $1 - \frac{1}{C}$ and $c < C$, we can verify that the new parameter $1 - c(1 - \eta)$ satisfies $0 < 1 - c(1 - \eta) < 1 - \frac{1}{C}$, so the rest of our proof applies (propagating constant changes appropriately) by Remark 4. \square

Acknowledgments

We would like to thank Mehtaab Sawhney for providing several pointers to the random matrix theory literature, in particular suggesting the use of Fact 4. We would also like to thank Li Chen for providing clarifications on [CPW21], that were used in Proposition 5 and Appendix A, as well as Richard Peng and Di Wang for providing other helpful comments on [CPW21].

References

- [ACSS20] Josh Alman, Timothy Chu, Aaron Schild, and Zhao Song. Algorithms and hardness for linear algebra on geometric graphs. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 541–552. IEEE, 2020.
- [ADK⁺16] Ittai Abraham, David Durfee, Ioannis Koutis, Sebastian Krinninger, and Richard Peng. On fully dynamic graph sparsifiers. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 335–344. IEEE Computer Society, 2016.
- [ADV⁺25] Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025*, pages 2005–2039. SIAM, 2025.
- [AGL⁺18] Zeyuan Allen-Zhu, Ankit Garg, Yuanzhi Li, Rafael Mendes de Oliveira, and Avi Wigderson. Operator scaling via geodesically convex optimization, invariant theory and polynomial identity testing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 172–181. ACM, 2018.

[AJJ⁺22] Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 627–669. SIAM, 2022.

[AKS20] Shiri Artstein-Avidan, Haim Kaplan, and Micha Sharir. On radial isotropic position: Theory and algorithms. *CoRR*, abs/2005.04918, 2020.

[ALdOW17] Zeyuan Allen-Zhu, Yuanzhi Li, Rafael Mendes de Oliveira, and Avi Wigderson. Much faster algorithms for matrix scaling. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 890–901. IEEE Computer Society, 2017.

[Bar98] Franck Barthe. On a reverse form of the brascamp-lieb inequality. *Inventiones Mathematicae*, 134(2):335–361, 1998.

[BFG⁺18] Peter Bürgisser, Cole Franks, Ankit Garg, Rafael Mendes de Oliveira, Michael Walter, and Avi Wigderson. Efficient algorithms for tensor scaling, quantum marginals, and moment polytopes. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 883–897. IEEE Computer Society, 2018.

[BFG⁺19] Peter Bürgisser, Cole Franks, Ankit Garg, Rafael Mendes de Oliveira, Michael Walter, and Avi Wigderson. Towards a theory of non-commutative optimization: Geodesic 1st and 2nd order methods for moment maps and polytopes. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 845–861. IEEE Computer Society, 2019.

[Bha07] Rajendra Bhatia. *Positive Definite Matrices*. Princeton University Press, 2007.

[BKMS21] Jess Banks, Archit Kulkarni, Satyaki Mukherjee, and Nikhil Srivastava. Gaussian regularization of the pseudospectrum and davies’ conjecture. *Communications on Pure and Applied Mathematics*, 74(10):2114–2131, 2021.

[BLNW20] Peter Bürgisser, Yinan Li, Harold Nieuwboer, and Michael Walter. Interior-point methods for unconstrained geometric programming and scaling problems. *CoRR*, abs/2008.12110, 2020.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[Che24] Yeshwanth Cherapanamjeri. Computing approximate centerpoints in polynomial time. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024*, pages 1654–1668. IEEE, 2024.

[CJJ⁺20] Yair Carmon, Arun Jambulapati, Qijia Jiang, Yujia Jin, Yin Tat Lee, Aaron Sidford, and Kevin Tian. Acceleration with a ball optimization oracle. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[CKV20] L. Elisa Celis, Vijay Keswani, and Nisheeth K. Vishnoi. Data preprocessing to mitigate bias: A maximum entropy based approach. In *Proceedings of the 37th International*

Conference on Machine Learning, ICML 2020, volume 119 of *Proceedings of Machine Learning Research*, pages 1349–1359. PMLR, 2020.

[CKYV19] L. Elisa Celis, Vijay Keswani, Ozan Yildiz, and Nisheeth K. Vishnoi. Fair distributions from biased samples: A maximum entropy optimization framework. *CoRR*, abs/1906.02164, 2019.

[CLL04] Eric Carlen, Elliott Lieb, and Michael Loss. A sharp analog of young’s inequality on s^n and related entropy inequalities. *The Journal of Geometric Analysis*, 14:487–520, 2004.

[CLM⁺15] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015*, pages 181–190. ACM, 2015.

[CMTV17] Michael B. Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained newton’s method and interior point methods. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 902–913. IEEE Computer Society, 2017.

[CPW21] Li Chen, Richard Peng, and Di Wang. 2-norm flow diffusion in near-linear time. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 540–549. IEEE, 2021.

[CPW25] Li Chen, Richard Peng, and Di Wang. Personal communication, 2025.

[DKT21] Ilias Diakonikolas, Daniel Kane, and Christos Tzamos. Forster decomposition and learning halfspaces with noise. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pages 7732–7744, 2021.

[DR24] Daniel Dadush and Akshay Ramachandran. Strongly polynomial frame scaling to high precision. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*, pages 962–981. SIAM, 2024.

[DSW14] Zeev Dvir, Shubhangi Saraf, and Avi Wigderson. Breaking the quadratic barrier for 3-lcc’s over the reals. In *Symposium on Theory of Computing, STOC 2014*, pages 784–793. ACM, 2014.

[DTK23] Ilias Diakonikolas, Christos Tzamos, and Daniel M. Kane. A strongly polynomial algorithm for approximate forster transforms and its application to halfspace learning. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023*, pages 1741–1754. ACM, 2023.

[For02] Jurgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *Journal of Computer and System Sciences*, 65(4):612–625, 2002.

[Fra18] Cole Franks. Operator scaling with specified marginals. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 190–203. ACM, 2018.

[GGdOW16] Ankit Garg, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson. A deterministic polynomial time algorithm for non-commutative rational identity testing. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 109–117. IEEE Computer Society, 2016.

[GGdOW17] Ankit Garg, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson. Algorithmic and optimization aspects of brascamp-lieb inequalities, via operator scaling. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 397–409. ACM, 2017.

[GGMS87] I. M. Gelfand, R. M. Goresky, R. D. MacPherson, and V. V. Serganova. Combinatorial geometries, convex polyhedra, and schubert cells. *Advances in Mathematics*, 63(3):301–316, 1987.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988.

[Gü92] Osman Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

[HJTY24] Lunjia Hu, Arun Jambulapati, Kevin Tian, and Chutong Yang. Testing calibration in nearly-linear time. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

[HKLM20] Max Hopkins, Daniel Kane, Shachar Lovett, and Gaurav Mahajan. Point location and active learning: Learning halfspaces almost optimally. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 1034–1044. IEEE, 2020.

[HM13] Moritz Hardt and Ankur Moitra. Algorithms and hardness for robust subspace recovery. In *COLT 2013 - The 26th Annual Conference on Learning Theory*, volume 30 of *JMLR Workshop and Conference Proceedings*, pages 354–375. JMLR.org, 2013.

[HM19] Linus Hamilton and Ankur Moitra. The paulsen problem made simple. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019*, volume 124 of *LIPICS*, pages 41:1–41:6. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[IQS17] Gábor Ivanyos, Youming Qiao, and K. V. Subrahmanyam. Constructive non-commutative rank computation is in deterministic polynomial time. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017*, volume 67 of *LIPICS*, pages 55:1–55:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[JLM⁺23] Arun Jambulapati, Jerry Li, Christopher Musco, Kirankumar Shiragur, Aaron Sidford, and Kevin Tian. Structured semidefinite programming for recovering structured preconditioners. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023*, 2023.

[JLSW20] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 944–953. ACM, 2020.

[JLT20] Arun Jambulapati, Jerry Li, and Kevin Tian. Robust sub-gaussian principal component analysis and width-independent schatten packing. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, 2020.

[Kad52] Richard V. Kadison. A generalized schwarz inequality and algebraic invariants for operator algebras. *Annals of Mathematics*, 56(3):494–503, 1952.

[KLM⁺17] Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. *SIAM J. Comput.*, 46(1):456–477, 2017.

[KMM⁺20] Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. Fast and space efficient spectral sparsification in dynamic streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 1814–1833. SIAM, 2020.

[KMP14] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. *SIAM J. Comput.*, 43(1):337–354, 2014.

[KSJ18] Sai Praneeth Karimireddy, Sebastian U. Stich, and Martin Jaggi. Global linear convergence of newton’s method without strong-convexity or lipschitz gradients. *CoRR*, abs/1806.00413, 2018.

[LM00] Béatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *The Annals of Statistics*, 28(5):1302–1338, 2000.

[LSW00] Nathan Linial, Alex Samorodnitsky, and Avi Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Comb.*, 20(4):545–568, 2000.

[PTZ16] Richard Peng, Kanat Tangwongsan, and Peng Zhang. Faster and simpler width-independent parallel algorithms for positive semidefinite programming. *CoRR*, abs/1201.5135v3, 2016.

[Qua21] Kent Quanrud. Spectral sparsification of metrics and kernels. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1445–1464. SIAM, 2021.

[Rou20] Tim Roughgarden, editor. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.

[Sch13] Kathrin Schacke. On the kronecker product, 2013.

[SS11] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

[SST06] Arvind Sankar, Daniel A. Spielman, and Shang-Hua Teng. Smoothed analysis of the condition numbers and growth factors of matrices. *SIAM J. Matrix Anal. Appl.*, 28(2):446–476, 2006.

- [ST04] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.
- [ST14] Daniel A. Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM J. Matrix Anal. Appl.*, 35(3):835–885, 2014.
- [SV14] Sushant Sachdeva and Nisheeth K. Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends in Theoretical Computer Science*, 9(2):125–210, 2014.
- [SV19] Damian Straszak and Nisheeth K. Vishnoi. Maximum entropy distributions: Bit complexity and stability. In *Conference on Learning Theory, COLT 2019*, volume 99 of *Proceedings of Machine Learning Research*, pages 2861–2891. PMLR, 2019.
- [Sza91] Stanislaw J. Szarek. Condition numbers of random matrices. *Journal of Complexity*, 7(2):131–149, 1991.
- [Ver24] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2024.
- [ZLO15] Zeyuan Allen Zhu, Zhenyu Liao, and Lorenzo Orecchia. Spectral sparsification and regret minimization beyond matrix multiplicative updates. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 237–245, 2015.
- [ZLO16] Zeyuan Allen Zhu, Yin Tat Lee, and Lorenzo Orecchia. Using optimization to obtain a width-independent, parallel, simpler, and faster positive SDP solver. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 1824–1831. SIAM, 2016.

A Discussion of Proposition 5

In this section, we describe the modifications to [CPW21] that are needed to prove Proposition 5. It is clear that by reparameterizing $\mathbf{l} \leftarrow \mathbf{l} - \mathbf{t}$ and $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{t}$ for the vectors $(\mathbf{t}, \mathbf{l}, \mathbf{r})$ defining the input set \mathcal{W} in Proposition 5, it is enough to describe how to obtain the stated guarantee with $\mathbf{t} = \mathbf{0}_n$. In other words, we need to obtain \mathbf{v} satisfying $\mathbf{l} \leq \mathbf{v} \leq \mathbf{r}$ entrywise, and

$$\langle \mathbf{b}, \mathbf{v} \rangle + \frac{1}{2} \mathbf{L}[\mathbf{v}, \mathbf{v}] \leq \frac{1}{2} \min_{\mathbf{l} \leq \mathbf{w} \leq \mathbf{r}} \left\{ \langle \mathbf{b}, \mathbf{w} \rangle + \frac{1}{2} \mathbf{L}[\mathbf{w}, \mathbf{w}] \right\}.$$

Structure of [CPW21]. We first provide an overview of the components of [CPW21]. Their core algorithmic result (which proves Proposition 5 when the box constraint is $[0, \infty)^n$) consists of three sub-pieces, Lemmas 4.3, 4.4, and 4.5. These pieces respectively correspond to an accelerated proximal point method, the construction of j -tree sparsifiers, and solving a diffusion instance on a j -tree. Of the three, Lemma 4.4 is independent of the constraint on the optimization problem, and thus can be left untouched, so we focus on the other two components.

Modifying Lemma 4.5. Lemma 4.5 is proven in Sections 6 and 7 of [CPW21], but the only piece of the proof that interacts with the constraint set explicitly is Lemma 6.6, shown in Section 7.2. This result provides a data structure that supports efficient modifications to a VWF (vertex weighting function, i.e., a convex piecewise-quadratic function with concave continuous derivative). The data structure maintains the coefficients and cutoff points for each piece of the VWF.

The data structure is implemented using a segment tree, where each leaf corresponds to a cutoff point s_i in a piecewise quadratic with k pieces. Implicitly, the data structure in [CPW21] sets the rightmost cutoff point $s_k = \infty$ and never modifies it. Instead, we can augment the data structure to maintain an explicit cutoff s_k . It is straightforward to check that all updates to s_k (i.e., in the operations given by Claims 7.9, 7.10, 7.11) in [CPW21] can be handled in $O(1)$ time, as there are closed-form formulas in each case. We remark that similar segment tree-based data structures supporting changes to a dynamic piecewise-polynomial function, but with an explicit right endpoint, have appeared in the recent literature, see e.g., Section 3 of [HJTY24].

Modifying Lemma 4.3. Lemma 4.3 is proven in Section 8 of [CPW21], and is an approximation-tolerant variant of the classical accelerated proximal point algorithm [Gü92]. The only explicit property about the constraint set used in this section is convexity, as efficient projection is handled by Lemma 4.5. This property holds for axis-aligned boxes, so Lemma 4.3 extends to our setting.

B Numerical Precision Considerations

For brevity, we specialize our discussion in this section to the setting of Theorem 1, where $\mathbf{c}_{\min}, \epsilon, \delta = \text{poly}(\frac{1}{n})$, and $\log(\kappa)$ is either $O(\log(n))$ (the *well-conditioned regime*), or $O(d \log(n))$ (the *smoothed analysis regime*). The latter name is justified by applying Theorem 3 with $\sigma = \text{poly}(\frac{1}{n})$. We also assume that entries of \mathbf{A} are represented with b -bit numbers, where $b = O(\log(n))$.

From the perspective of numerical stability, the bottleneck in both regimes is computing derivatives of Barthe's objective up to $\text{poly}(\frac{1}{n})$ additive error. All other operations performed by the box-constrained Newton's method in Theorem 1, either using explicit Hessian computations (Remark 1)

or implicit Hessian sparsification (Theorem 2), is tolerant to $\text{poly}(\frac{1}{n})$ additive error in entries of vectors and matrices. Thus, we can simply truncate our bit representations to $O(\log(n))$ -sized words, as long as we can compute gradients, Hessians, or Hessian-vector products stably.

Bottleneck operations. Recall Fact 2, and define for $\mathbf{t} \in \mathbb{R}^n$,

$$\mathbf{Z}(\mathbf{t}) := \sum_{i \in [n]} \exp(\mathbf{t}_i) \mathbf{a}_i \mathbf{a}_i^\top.$$

For an instance of Definition 1 where Assumption 1 holds, the bottleneck operations are evaluating, for $\mathbf{t} \in \mathbb{R}^n$ with $\|\mathbf{t}\|_\infty \leq \log(\kappa)$, gradients $\nabla f(\mathbf{t})$:

$$\left\{ \text{Tr} \left(\exp(\mathbf{t}_i) \mathbf{a}_i \mathbf{a}_i^\top (\mathbf{Z}(\mathbf{t}))^{-1} \right) \right\}_{i \in [n]},$$

Hessians $\nabla^2 f(\mathbf{t})$:

$$\left\{ \text{Tr} \left(\exp(\mathbf{t}_i) \mathbf{a}_i \mathbf{a}_i^\top (\mathbf{Z}(\mathbf{t}))^{-1} \right) \mathbb{1}_{i=j} - \text{Tr} \left(\exp(\mathbf{t}_i + \mathbf{t}_j) \mathbf{a}_i \mathbf{a}_i^\top (\mathbf{Z}(\mathbf{t}))^{-1} \mathbf{a}_j \mathbf{a}_j^\top (\mathbf{Z}(\mathbf{t}))^{-1} \right) \right\}_{(i,j) \in [n] \times [n]},$$

and Hessian-vector products with the above form. Lemma 5 gives methods for performing these operations without consideration of numerical stability.

In the well-conditioned regime $\|\mathbf{t}\|_\infty = O(\log(n))$, all of the numbers in the above expressions are representable up to $\text{poly}(\frac{1}{n})$ error using $O(\log(n))$ -bit words. Thus, we believe that our algorithms are numerically stable at constant overhead to the bit complexity, as these bottleneck operations reduce to standard matrix manipulations on $O(\log(n))$ -bit entries.

In the smoothed analysis regime $\|\mathbf{t}\|_\infty \gtrsim d$, it is possible that the computation of gradients and Hessians described here is not numerically stable using $O(\log(n))$ -bit representations of numbers. As a result, our algorithms as stated may require larger bit complexity, and suffer in runtime.

We note that all other work on optimizing Barthe's objective, based on cutting-plane methods [HM13] or gradient descent [AKS20], also suffers from the same numerical stability challenges, because they also require computing gradients. Thus, under any reasonable cost model our algorithm is the state-of-the-art by at least a $\approx \frac{n}{d}$ factor in the smoothed analysis regime.

We think the strategy of optimizing Barthe's objective for computing Forster transforms is a natural one, and hence evaluating the numerical stability of gradient and Hessian computation under finite bit precision is an important goal (for any derivative-based algorithm, not just ours). Alternatively, can we prove tighter conditioning bounds than Theorem 3, in natural statistical models?

Strongly polynomial methods. Computing Forster transforms in finite-precision arithmetic was explicitly studied by [DTK23, DR24], who considered a definition of strongly polynomial introduced by [GLS88] that requires polynomial space complexity. Each proved the types of rounding result described at the end of Section 1.1; however, the resulting bit complexities are rather large. For example, Theorem 5.1 of [DTK23] proves magnitude bounds of $\approx \exp(d^3 b)$ where b is an initial word size, with an unspecified intermediate bit complexity. Corollary 4.10 of [DR24] yields the improved estimate of $\approx n d b$ bits required, for stable implementation on worst-case instances. However, it is unclear how these rounding procedures affect progress on Barthe's objective.