Spectrum Approximation Beyond Fast Matrix Multiplication: Algorithms and Hardness*

Cameron Musco[1], Praneeth Netrapalli[2], Aaron Sidford[3], Shashanka Ubaru[4], and David P. Woodruff[5]

1 **Massachusetts Institute of Technology, Cambridge, MA, USA**
cnmusco@mit.edu
2 **Microsoft Research, Bangalore, India**
praneeth@microsoft.com
3 **Stanford University, Stanford, CA, USA**
sidford@stanford.edu
4 **University of Minnesota, Minneapolis, MN, USA**
ubaru001@umn.edu
5 **Carnegie Mellon University, Pittsburgh, PA, USA**
dwoodruf@cs.cmu.edu

Abstract

Understanding the singular value spectrum of a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a fundamental task in countless numerical computation and data analysis applications. In matrix multiplication time, it is possible to perform a full SVD of $\mathbf{A}$ and directly compute the singular values $\sigma_1, \ldots, \sigma_n$. However, little is known about algorithms that break this runtime barrier.

Using tools from stochastic trace estimation, polynomial approximation, and fast linear system solvers, we show how to efficiently isolate different ranges of $\mathbf{A}$'s spectrum and approximate the number of singular values in these ranges. We thus effectively compute an *approximate histogram* of the spectrum, which can stand in for the true singular values in many applications.

We use our histogram primitive to give the first algorithms for approximating a wide class of symmetric matrix norms and spectral sums *faster than the best known runtime for matrix multiplication*. For example, we show how to obtain a $(1+\epsilon)$ approximation to the Schatten 1-norm (i.e. the nuclear or trace norm) in just $\tilde{O}((\text{nnz}(\mathbf{A})n^{1/3} + n^2)\epsilon^{-3})$ time for $\mathbf{A}$ with uniform row sparsity or $\tilde{O}(n^{2.18}\epsilon^{-3})$ time for dense matrices. The runtime scales smoothly for general Schatten-p norms, notably becoming $\tilde{O}(p\,\text{nnz}(\mathbf{A})\epsilon^{-3})$ for any real $p \geq 2$.

At the same time, we show that the complexity of spectrum approximation is inherently tied to fast matrix multiplication in the small ϵ regime. We use fine-grained complexity to give conditional lower bounds for spectrum approximation, showing that achieving milder ϵ dependencies in our algorithms would imply triangle detection algorithms for general graphs running in faster than state of the art matrix multiplication time. This further implies, through a reduction of [72], that highly accurate spectrum approximation algorithms running in subcubic time can be used to give subcubic time matrix multiplication. As an application of our bounds, we show that precisely computing all effective resistances in a graph in less than matrix multiplication time is likely difficult, barring a major algorithmic breakthrough.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems

Keywords and phrases spectrum approximation, matrix norm computation, fine-grained complexity, linear algebra

Digital Object Identifier 10.4230/LIPIcs.ITCS.2018.8

* A full version of the paper is available at https://arxiv.org/abs/1704.04163

1 Introduction

Given $\mathbf{A} \in \mathbb{R}^{n\times d}$, a central primitive in numerical computation and data analysis is to compute $\mathbf{A}$'s spectrum: the singular values $\sigma_1(\mathbf{A}) \geq \ldots \geq \sigma_d(\mathbf{A}) \geq 0$. These values can reveal matrix structure and low effective dimensionality, which can be exploited in a wide range of spectral data analysis methods [33, 67]. The singular values are also used as tuning parameters in many numerical algorithms performed on $\mathbf{A}$ [22], and in general, to determine some of the most well-studied matrix functions [29]. For example, for any $f : \mathbb{R}^+ \to \mathbb{R}^+$, we can define the *spectral sum*:

$$\mathcal{S}_f(\mathbf{A}) \stackrel{\text{def}}{=} \sum_{i=1}^{d} f(\sigma_i(\mathbf{A})).$$

Spectral sums often serve as snapshots of $\mathbf{A}$'s spectrum and are important in many applications. They encompass, for example, the log-determinant, the trace inverse, the Schatten-p norms, including the nuclear norm, and general Orlicz norms (see Section 1.2 for details).

While the problem of computing a few of the largest or smallest singular values of $\mathbf{A}$ has been exhaustively studied [56, 60], much less is known about algorithms that approximate the full spectrum, and in particular, allow for the computation of summary statistics such as spectral sums. In n^ω time, it is possible to perform a full SVD and compute the singular values exactly.[1] Here, and throughout, $\omega \approx 2.3729$ denotes the *current* best exponent of fast matrix multiplication [70]. However, even if one simply desires, for example, a constant factor approximation to the nuclear norm $\|\mathbf{A}\|_1$, no $o(n^\omega)$ time algorithm is known. We study the question of spectrum approximation, asking whether obtaining an accurate picture of $\mathbf{A}$'s spectrum is truly as hard as matrix multiplication, or if it is possible to break this barrier. We focus on spectral sums as a motivating application.

1.1 Our Contributions

1.1.1 Upper Bounds

On the upper bound side, we show that significant information about $\mathbf{A}$'s spectrum can be determined in $o(n^\omega)$ time, for the current value of ω. We show how to compute a histogram of the spectrum, which gives approximate counts of the number of squared singular values in the ranges $[(1-\alpha)^t\sigma_1^2(\mathbf{A}), (1-\alpha)^{t-1}\sigma_1^2(\mathbf{A})]$ for some width parameter α and for t ranging from 0 to some maximum T. Specifically our algorithm satisfies the following:

▶ **Theorem 1** (Histogram Approximation – Informal). *Given $\mathbf{A} \in \mathbb{R}^{n\times d}$, let b_t be the number of squared singular values of $\mathbf{A}$ on the range $[(1-\alpha)^t\sigma_1^2(\mathbf{A}), (1-\alpha)^{t-1}\sigma_1^2(\mathbf{A})]$. Then given error parameter $\epsilon > 0$, with probability 99/100, Algorithm 1 outputs for all $t \in \{0, ..., T\}$, $\tilde{b}_t$ satisfying:*

$$(1-\epsilon)b_t \leq \tilde{b}_t \leq (1+\epsilon)b_t + \epsilon(b_{t-1} + b_{t+1}).$$

For input parameter $k \in \{1, ..., d\}$, let $\bar{\kappa} \stackrel{\text{def}}{=} \frac{k\sigma_k^2(\mathbf{A}) + \sum_{i=k+1}^{d}\sigma_i^2(\mathbf{A})}{d\cdot(1-\alpha)^T}$ and $\hat{\kappa} \stackrel{\text{def}}{=} \frac{\sigma_{k+1}^2(\mathbf{A})}{(1-\alpha)^T}$. Let $d_s(\mathbf{A})$

[1] Note that an exact SVD is incomputable even with exact arithmetic [65]. Nevertheless, direct methods for the SVD obtain superlinear convergence rates and hence are often considered to be 'exact'.

be the maximum number of nonzeros in a row of $\mathbf{A}$*. The algorithm's runtime is bounded by:*

$$\tilde{O}\left(\frac{\text{nnz}(\mathbf{A})k + dk^{\omega-1} + \sqrt{\text{nnz}(\mathbf{A})[d \cdot d_s(\mathbf{A}) + dk]\bar{\kappa}}}{\text{poly}(\epsilon, \alpha)}\right) \text{ or } \tilde{O}\left(\frac{\text{nnz}(\mathbf{A})k + dk^{\omega-1} + (\text{nnz}(\mathbf{A}) + dk)\lceil\sqrt{\bar{\kappa}}\rceil}{\text{poly}(\epsilon, \alpha)}\right)$$

for sparse $\mathbf{A}$ *or* $\tilde{O}\left(\frac{nd^{\gamma-1}+n^{1/2}d^{3/2}\sqrt{\bar{\kappa}})}{\text{poly}(\epsilon,\alpha)}\right)$ *for dense* $\mathbf{A}$*, where* d^γ *is the time it takes to multiply a* $d \times d$ *matrix by a* $d \times k$ *matrix using fast matrix multiplication.*

This primitive is useful on its own – the summary of $\mathbf{A}$'s spectrum which can be used in many downstream applications. Setting the parameter k appropriately to balance costs (see overview in Section 1.3), we use it to give the first $o(n^\omega)$ algorithms for computing $(1 \pm \epsilon)$ relative error approximations to a broad class of spectral sums for functions f, which are a) smooth and b) quickly growing, so that very small singular values cannot make a significant contribution to $\mathcal{S}_f(\mathbf{A})$. This class includes for example the Schatten p-norms for all $p > 0$, the SVD entropy, the Ky Fan norms, and many general Orlicz norms.

For a summary of our p-norm results see Table 1. Focusing for simplicity on square matrices, with uniformly sparse rows, and assuming ϵ, p are constants, our algorithms approximate $\|\mathbf{A}\|_p^p$ in $\tilde{O}(\text{nnz}(\mathbf{A}))$ time for any real $p \geq 2$.[2] For $p \leq 2$, we achieve $\tilde{O}\left(\text{nnz}(\mathbf{A})n^{\frac{1/p-1/2}{1/p+1/2}} + n^{\frac{4/p-1}{2/p+1}}\sqrt{\text{nnz}(\mathbf{A})}\right)$ runtime. In the important case of $p = 1$, this becomes $\tilde{O}\left(\text{nnz}(\mathbf{A})n^{1/3} + n\sqrt{\text{nnz}(\mathbf{A})}\right)$. Note that $n\sqrt{\text{nnz}(\mathbf{A})} \leq n^2$, and for sparse enough $\mathbf{A}$, this bound is subquadratic. For dense $\mathbf{A}$, we use fast matrix multiplication, achieving time $\tilde{O}\left(n^{\frac{2.3729-.0994p}{1+.0435p}}\right)$ for all $p < 2$. For $p = 1$, this gives $\tilde{O}(n^{2.18})$. Even without fast matrix multiplication, the runtime is $\tilde{O}(n^{2.33})$, and so $o(n^\omega)$ for $\omega \approx 2.3729$.

1.1.2 Lower Bounds

On the lower bound side, we show that obtaining $o(n^\omega)$ time spectrum approximation algorithms with very high accuracy may be difficult. Our runtimes all depend polynomially on the error ϵ, and we show that improving this, e.g., to $\log(1/\epsilon)$, or even to a better polynomial, would give faster algorithms for the well studied Triangle Detection problem.

Specifically, for a broad class of spectral sums, including all Schatten p-norms with $p \neq 2$, SVD entropy, $\log\det(\mathbf{A})$, $\text{tr}(\mathbf{A}^{-1})$, and $\text{tr}(\exp(\mathbf{A}))$, we show that any $(1 \pm \epsilon)$ approximation algorithm running in $O(n^\gamma \epsilon^{-c})$ time yields an algorithm for triangle detection running in $O(n^{\gamma+O(c)})$ time. For $\gamma < \omega$ and sufficiently small c, such an algorithm would improve the state of the art in triangle detection, which currently requires $\Theta(n^\omega)$ time on dense graphs. Furthermore, through a reduction of [72], any subcubic time triangle detection algorithm yields a subcubic time algorithm for Boolean Matrix Multiplication (BMM). Thus, any spectral sum algorithm achieving subcubic runtime and $\frac{1}{\epsilon^c}$ accuracy for small enough constant c, must (implicitly) implement fast matrix multiplication. This is in stark contrast to the fact that, for $c = 3$, for many spectral sums, including all Schatten-p with $p \geq 1/2$, we are able to obtain subcubic, and in fact $o(n^\omega)$ for $\omega = 2.3729$, runtimes without using fast matrix multiplication (see Table 1 for precise ϵ dependencies).

Our lower bounds hold even for well-conditioned matrices and structured matrices like symmetric diagonally dominant (SDD) systems, both of which admit nearly linear time algorithms for system solving [63]. This illustrates a dichotomy between linear algebraic

[2] For any $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\text{nnz}(\mathbf{A})$ denotes the number of nonzero entries in $\mathbf{A}$.

Table 1 Summary of our results for approximating the Schatten-p norms. We define $f(p,\epsilon) = \min\{1, p^3\} \cdot \epsilon^{\max\{3,1+1/p\}}$, which appears as a factor in many of the bounds. The uniform sparsity assumption is that the maximum row sparsity $d_s(\mathbf{A}) \leq \frac{\xi}{n}\operatorname{nnz}(\mathbf{A})$ for some constant ξ. In our theorems, we give general runtimes, parameterized by ξ. When we do not have the uniform sparsity assumption, we are still able to give a $(1+\epsilon)$ approximation in $\tilde{O}(\epsilon^{-3}\operatorname{nnz}(\mathbf{A})\sqrt{n} + n^2)$ time for example for $\|\mathbf{A}\|_1$. We can also give $1/\gamma$ approximation for any constant $\gamma < 1$ by paying an $n^{\gamma/2}$ factor in our runtime. Note that for dense matrices, for all p we obtain $o(n^\omega)$ runtime, or $o(n^3)$ runtime if we do not use fast matrix multiplication. Theorems numbers reference the full paper [53].

p	Sparsity	Appx.	Runtime	Theorem
$p > 2$	uniform	$(1+\epsilon)$	$\tilde{O}\left(\operatorname{nnz}(\mathbf{A}) \cdot p/\epsilon^3\right)$	Thm 32
$p \leq 2$	uniform	$(1+\epsilon)$	$\tilde{O}\left(\frac{1}{f(p,\epsilon)}\left[\operatorname{nnz}(\mathbf{A})n^{\frac{1/p-1/2}{1/p+1/2}} + n^{\frac{4/p-1}{2/p+1}}\sqrt{\operatorname{nnz}(\mathbf{A})}\right]\right)$	Thm 33
$p \leq 2$	dense	$(1+\epsilon)$	$\tilde{O}\left(\frac{1}{f(p,\epsilon)}n^{\frac{2.3729-.0994p}{1+.0435p}}\right)$, $\tilde{O}\left(\frac{1}{f(p,\epsilon)}n^{\frac{3+p/2}{1+p/2}}\right)$ w/o FMM	Thm 31
$p > 0$	general	$(1+\epsilon)$	$\tilde{O}\left(\frac{1}{f(p,\epsilon)}\left[\operatorname{nnz}(\mathbf{A})n^{\frac{1}{1+p}} + n^{1+\frac{2}{1+p}}\right]\right)$	Thms 32, 33
$p > 2$	general	$1/\gamma$	$\tilde{O}(p\operatorname{nnz}(\mathbf{A}) \cdot n^\gamma)$	Thm 34
$p < 2$	general	$1/\gamma$	$\tilde{O}\left(\frac{1}{p^3}\left[\operatorname{nnz}(\mathbf{A})n^{\frac{1/p-1/2}{1/p+1/2}+\gamma/2} + \sqrt{\operatorname{nnz}(\mathbf{A})} \cdot n^{\frac{4/p-1}{2/p-1}}\right]\right)$	Thm 34

primitives like applying $\mathbf{A}^{-1}$ to a vector and spectral summarization tasks like precisely computing $\operatorname{tr}(\mathbf{A}^{-1})$. Our analysis has ramifications regarding natural open problems in graph theory and numerical computation. For example, for graph Laplacians, we show that accurately computing all *effective resistances* yields an accurate algorithm for computing $\operatorname{tr}(\mathbf{A}^{-1})$ of certain matrices, which is enough to give triangle detection.

1.2 Related Work on Spectral Sums

The applications of approximate spectral sum computation are broad. When $\mathbf{A}$ is positive semidefinite (PSD) and $f(x) = \log(x)$, $\mathcal{S}_f(\mathbf{A})$ is the log-determinant, which is important in machine learning and inference applications [57, 13, 19]. For $f(x) = 1/x$, $\mathcal{S}_f(\mathbf{A})$ is the trace of the inverse, used in uncertainty quantification [7] and quantum chromodynamics [64].

When $f(x) = x^p$, $S_f(\mathbf{A}) = \|\mathbf{A}\|_p^p$ where $\|\mathbf{A}\|_p$ is the Schatten p-norm of $\mathbf{A}$. Computation of the Schatten 1-norm, also known as the nuclear or trace norm, is required in a wide variety of applications. It is often used in place of the matrix rank in matrix completion algorithms and other convex relaxations of rank-constrained optimization problems [10, 15, 31, 54]. It appears as the 'graph energy' in theoretical chemistry [25, 26], the 'singular value bound' in differential privacy [28, 42], and in rank aggregation and collaborative ranking [48].

Similar to the nuclear norm, general Schatten p-norms are used in convex relaxations for rank-constrained optimization [55]. They also appear in image processing applications [76], classification [49], restoration [75], and feature extraction [17].

When $f(x) = -x\log x$ (after $\mathbf{A}$ is normalized by $\|\mathbf{A}\|_1$), $\mathcal{S}_f(\mathbf{A})$ is the SVD entropy [2], used in feature selection [69, 5], financial data analysis [11, 24], and genomic applications [2].

Despite their importance, prior to our work, few algorithms for fast computation of spectral sums existed. Only a few special cases of the Schatten p-norms were known to be computable in $o(n^\omega)$ time. The Frobenius norm ($p = 2$) is trivially computed in $O(\operatorname{nnz}(\mathbf{A}))$ time. The spectral norm ($p = \infty$) which can be estimated via the Lanczos method in $\tilde{O}(\operatorname{nnz}(\mathbf{A})\epsilon^{-\frac{1}{2}})$ time [38]. Finally, the Schatten-p norms for *even integers* $p > 2$, or general integers with PSD $\mathbf{A}$. These norms can be approximated in $O(\operatorname{nnz}(\mathbf{A})\epsilon^{-2})$ time via trace estimation [74, 9], since when p is even or $\mathbf{A}$ is PSD, $\mathbf{A}^p$ is PSD and so its trace equals $\|\mathbf{A}\|_p^p$.

There are a number of works which consider estimating matrix norms in sublinear space and with a small number of passes over $\mathbf{A}$ [44, 3, 45, 9, 46]. However, in these works, the main focus is on space complexity, and no non-trivial runtime bounds are given. We seem to be the first to tackle the arguably more fundamental problem of obtaining the best time complexity for simple norms like the Schatten-p norms.

Another interesting line of works tries to estimate the Schatten-p norms of an underlying covariance matrix from a small number of samples from the distribution [37], or from entrywise sampling under various incoherence assumptions [34]. This model is different from ours, as we do not assume an underlying distribution or any incoherence properties. Moreover, even with such assumptions, these algorithms also only give non-trivial sample complexity when either $\mathbf{A}$ is PSD and p is an integer, or $\mathbf{A}$ is a general matrix but p is an even integer, which as mentioned above are easy to handle from the perspective of time complexity alone.

A number of works have focused on computing spectral sums when $\mathbf{A}$ has bounded condition number, and relative error results exist for the log-determinant, $\mathrm{tr}(\exp(\mathbf{A}))$, $\mathrm{tr}(\mathbf{A}^{-1})$, and the Schatten p-norms [8, 27, 66]. We are the first to give relative error results in $o(\omega)$ time for general matrices, without the condition number dependence. Our histogram approach resembles spectral filtering and spectral density estimation techniques that have been considered in the numerical computation literature [77, 16, 47, 67, 68]. However, this literature typically requires assuming gaps between the singular values and existing work is not enough to give relative error spectral sum approximation for general matrices

1.3 Algorithmic Approach

We now give a high level overview of the techniques used in our algorithms.

1.3.1 Spectral Sums via Trace Estimation

A common approach to spectral sum approximation is to reduce to a trace estimation problem involving the PSD matrix $\mathbf{A}^T\mathbf{A}$, using the fact that the trace of this matrix equals the sum of its singular values. In fact, this has largely been the only known technique, other than the full SVD, for obtaining aggregate information about $\mathbf{A}$'s singular values [30, 64, 59, 18, 8, 27]. The idea is, letting $g(x) = f(x^{1/2})$, we have $\mathcal{S}_f(\mathbf{A}) = \mathcal{S}_g(\mathbf{A}^T\mathbf{A})$. Writing the SVD $\mathbf{A}^T\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, and defining the matrix function $g(\mathbf{A}^T\mathbf{A}) \stackrel{\text{def}}{=} \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^T$ where $[g(\mathbf{\Lambda})]_{i,i} = g([\mathbf{\Lambda}]_{i,i})$, we have $\mathcal{S}_g(\mathbf{A}^T\mathbf{A}) = \mathrm{tr}(g(\mathbf{A}^T\mathbf{A}))$ since, if $g(\cdot)$ is nonnegative, $g(\mathbf{A}^T\mathbf{A})$ is PSD and its trace equals the sum of its singular values.

It is well known that this trace can be approximated up to $(1 \pm \epsilon)$ accuracy by averaging $\tilde{O}(\epsilon^{-2})$ samples of the form $\mathbf{x}^T g(\mathbf{A}^T\mathbf{A})\mathbf{x}$ where $\mathbf{x}$ is a random Gaussian or sign vector [30, 4]. While $g(\mathbf{A}^T\mathbf{A})$ cannot be explicitly computed without a full SVD, a common approach is to approximate g with a low-degree polynomial ϕ [8, 27]. If ϕ has degree q, one can apply $\phi(\mathbf{A}^T\mathbf{A})$ to any vector $\mathbf{x}$ in $O(\mathrm{nnz}(\mathbf{A}) \cdot q)$ time, and so estimate its trace in just $O(\mathrm{nnz}(\mathbf{A}) \cdot \frac{q}{\epsilon^2})$ time. Unfortunately, for many of the functions most important in applications, e.g., $f(x) = x^p$ for odd p, $f(x) = x \log x$, $f(x) = x^{-1}$, $g(x)$ has a discontinuity at $x = 0$ and *cannot* be approximated well by a low-degree polynomial near zero. While the approximation only needs to be good in the range $[\sigma_n^2(\mathbf{A}), \sigma_1^2(\mathbf{A})]$, the required degree q will still typically depend on $\sqrt{\kappa}$ where $\kappa \stackrel{\text{def}}{=} \frac{\sigma_1^2(\mathbf{A})}{\sigma_n^2(\mathbf{A})}$ is the condition number, which can be unbounded in general.

1.3.2 Singular Value Deflation for Improved Conditioning

Our first observation is that, for many functions, it is not necessary to approximate $g(x)$ on the full spectral range. For example, for $g(x) = x^{p/2}$ (i.e., when $\mathcal{S}_g(\mathbf{A}^T\mathbf{A}) = \|\mathbf{A}\|_p^p$), setting $\lambda = (\frac{\epsilon}{n}\|\mathbf{A}\|_p^p)^{1/p}$:

$$\sum_{\{i|\sigma_i(\mathbf{A})\leq\lambda\}} \sigma_i(\mathbf{A})^p \leq n \cdot \frac{\epsilon}{n}\|\mathbf{A}\|_p^p \leq \epsilon\|\mathbf{A}\|_p^p.$$

Hence we can safely 'ignore' any $\sigma_i(\mathbf{A}) \leq \lambda$ and still obtain a relative error approximation to $\mathcal{S}_g(\mathbf{A}^T\mathbf{A}) = \|\mathbf{A}\|_p^p$. The larger p is, the larger we can set λ (corresponding to $(1-\alpha)^T$ in Theorem 1) to be, since, after powering, the singular values below this threshold do not contribute significantly to $\|\mathbf{A}\|_p^p$. For $\|\mathbf{A}\|_p^p$, our 'effective condition number' for approximating $g(x)$ becomes $\hat{\kappa} = \frac{\sigma_1^2(\mathbf{A})}{\lambda^2} = (\frac{n}{\epsilon})^{2/p} \cdot \frac{\sigma_1^2(\mathbf{A})}{\|\mathbf{A}\|_p^2}$. Unfortunately, in the worst case, we may have $\sigma_1(\mathbf{A}) \approx \|\mathbf{A}\|_p$ and hence $\sqrt{\hat{\kappa}} = (\frac{n}{\epsilon})^{1/p}$. Hiding ϵ dependences, this gives runtime $\tilde{O}(\text{nnz}(\mathbf{A}) \cdot n)$ when $p = 1$.

To improve the effective condition number, we can apply *singular vector deflation*. Our above bound on $\hat{\kappa}$ is only tight when $\sigma_1(\mathbf{A})$ is very large and so dominates $\|\mathbf{A}\|_p$. We can remedy this by flattening $\mathbf{A}$'s spectrum by deflating off the top k singular vectors (corresponding to k in Theorem 1), and including their values in the spectral sum directly.

Specifically, let $\mathbf{P}_k$ be the projection onto the top k singular vectors of $\mathbf{A}$ and consider the deflated matrix $\bar{\mathbf{A}} \stackrel{\text{def}}{=} \mathbf{A}(\mathbf{I} - \mathbf{P}_k)$, which has $\sigma_1(\bar{\mathbf{A}}) = \sigma_{k+1}(\mathbf{A})$. Importantly, $\sigma_{k+1}^p(\mathbf{A}) \leq \frac{1}{k}\|\mathbf{A}\|_p^p$, and so this singular value cannot dominate the p-norm. For example, considering $p = 1$ and ignoring ϵ dependencies, our effective condition number after deflation is

$$\hat{\kappa} = \frac{n^2 \cdot \sigma_{k+1}^2(\mathbf{A})}{\|\mathbf{A}\|_1^2} \leq \frac{n^2}{k^2} \tag{1}$$

The runtime required to approximate $\mathbf{P}_k$ via an iterative method (ignoring possible gains from fast matrix multiplication) is roughly $O(\text{nnz}(\mathbf{A})k + nk^2)$. We then require $\tilde{O}(\text{nnz}(\mathbf{A})\sqrt{\hat{\kappa}} + nk\sqrt{\hat{\kappa}})$ time to approximate the polynomial trace of $\bar{\mathbf{A}}^T\bar{\mathbf{A}}$. The $nk\sqrt{\hat{\kappa}}$ term comes from projecting off the top singular directions with each application of $\bar{\mathbf{A}}^T\bar{\mathbf{A}}$. Setting $k = \sqrt{n}$ to balance the costs, we obtain runtime $\tilde{O}(\text{nnz}(\mathbf{A})\sqrt{n} + n^2)$.

For $p \neq 1$ a similar argument gives runtime $\tilde{O}(\text{nnz}(\mathbf{A})n^{\frac{1}{p+1}} + n^{2+\frac{1}{p+1}})$. This is already a significant improvement over a full SVD. As p grows larger, the runtime approaches $\tilde{O}(\text{nnz}(\mathbf{A}))$ reflecting the fact that for larger p we can ignore a larger and larger portion of the small singular values in $\mathbf{A}$ and correspondingly deflate off fewer and fewer top values.

Unfortunately, we get stuck here. Considering the important Schatten-1 norm, for a matrix with $\sqrt{n}$ singular values each equal to $\sqrt{n}$ and $\Theta(n)$ singular values each equal to 1, the tail of small singular values contributes a constant fraction of $\|\mathbf{A}\|_1 = \Theta(n)$. However, there is no good polynomial approximation to $g(x) = x^{1/2}$ on the range $[1, n]$ with degree $o(\sqrt{n})$ (recall that we pick this function since $\mathcal{S}_g(\mathbf{A}^T\mathbf{A}) = \|\mathbf{A}\|_1$). So to accurately approximate $g(\mathbf{A}^T\mathbf{A})$, we either must deflate off all $\sqrt{n}$ top singular values, requiring $\Theta(\text{nnz}(\mathbf{A})\sqrt{n})$ time, or apply a $\Theta(\sqrt{n})$ degree polynomial approximation, requiring the same amount of time.

1.3.3 Further Improvements with Stochastic Gradient Descent

To push beyond this barrier, we look to *stochastic gradient* methods for linear systems. When using polynomial approximation, our bounds depend on the condition number of the interval over which we must approximate $g(\mathbf{A}^T\mathbf{A})$, after ignoring the smallest singular

values and deflating off the largest. This is analogous to the condition number dependence of iterative linear system solvers like conjugate gradient or accelerated gradient descent, which approximate $f(\mathbf{A}^T\mathbf{A})$ for $f = 1/x$ using a polynomial of $\mathbf{A}^T\mathbf{A}$.

However, recent advances in convex optimization offer an alternative. Stochastic gradient methods [32, 61] sample one row, $\mathbf{a}_i$, of $\mathbf{A}$ at a time, updating the current iterate by adding a multiple of $\mathbf{a}_i$. They trade a larger number of iterations for updates that take $O(\text{nnz}(\mathbf{a}_i))$ time, rather than $O(\text{nnz}(\mathbf{A}))$ time to multiply $\mathbf{A}$ by a vector. These methods give much finer dependencies on the singular value spectrum. Specifically, it is possible to approximately apply $(\mathbf{A}^T\mathbf{A})^{-1}$ to a vector with the number of iterations dependent on the *average condition number:* $\bar{\kappa} = \frac{\frac{1}{n}\sum_{i=1}^{n}\sigma_i^2(\mathbf{A})}{\sigma_n^2(\mathbf{A})}$. $\bar{\kappa}$ is always at most the standard condition number, $\kappa = \frac{\sigma_1^2(\mathbf{A})}{\sigma_n^2(\mathbf{A})}$. It can be significantly smaller when $\mathbf{A}$ has a quickly decaying spectrum, and hence $\frac{1}{n}\sum_{i=1}^{n}\sigma_i^2(\mathbf{A}) \ll \sigma_1^2(\mathbf{A})$. Further, the case of a quickly decaying spectrum with a few large and many small singular values is *exactly the hard case for our earlier approach.* If we can understand how to translate improvements on linear system solvers to spectral sum approximation, we can handle this hard case.

1.3.4 From Linear System Solvers to Histogram Approximation

The key idea to translating the improved average condition number bounds for linear systems to our problem of approximating $\mathcal{S}_f(\mathbf{A})$ is to note that linear system solvers can be used to apply threshold functions to $\mathbf{A}^T\mathbf{A}$.

Specifically, given any vector $\mathbf{y}$, we can first compute $\mathbf{A}^T\mathbf{A}\mathbf{y}$. We can then apply a fast system solver to approximate $(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{A}\mathbf{y}$. The matrix function $r_\lambda(\mathbf{A}^T\mathbf{A}) \stackrel{\text{def}}{=} (\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{A}$ has a number of important properties. All its singular values are between 0 and 1. Further, any singular value in $\mathbf{A}^T\mathbf{A}$ with value $\geq \lambda$ is mapped to a singular value in $r_\lambda(\mathbf{A}^T\mathbf{A})$ which is $\geq 1/2$. Correspondingly, any singular value $< \lambda$ is mapped to $< 1/2$.

Thus, we can apply a low degree polynomial approximation to a step function at $1/2$ to $r_\lambda(\mathbf{A}^T\mathbf{A})$ to obtain $s_\lambda(\mathbf{A}^T\mathbf{A})$, which approximates a step function at λ [20]. For some steepness parameter γ which affects the degree of the polynomial approximation, for $x \geq (1+\gamma)\lambda$ we have $s_\lambda(x) \approx 1$ and for $x < (1-\gamma)\lambda$, $s_\lambda(x) \approx 0$. On the intermediate range $x \in [(1-\gamma)\lambda, (1+\gamma)\lambda]$, $s_\lambda(x)$ falls somewhere between 0 and 1.

Composing approximate threshold functions lets us 'split' our spectrum into a number of small spectral windows. For example, $s_a(\mathbf{A}^T\mathbf{A}) \cdot (\mathbf{I} - s_b(\mathbf{A}^T\mathbf{A}))$ is ≈ 1 on the range $[a, b]$ and ≈ 0 outside this range, with some ambiguity near a and b.

Splitting our spectrum into windows of the form $[(1-\alpha)^{t-1}, (1-\alpha)^t]$ for a width parameter α, and applying trace estimation on each window lets us produce an approximate spectral histogram. Of course, this histogram is not exact and in particular, the 'blurring' of our windows at their boundaries can introduce significant error. However, by applying a random shifting technique and setting the steepness parameter γ small enough (i.e., $1/\text{poly}(\alpha, \epsilon)$), we can ensure that most of the spectral weight falls outside these boundary regions with good probability, giving Theorem 1.

1.3.5 From Histogram Approximation to Spectral Sums

If α is small enough, and $f(\cdot)$ and correspondingly $g(\cdot)$ (where $g(x) = f(x^{1/2})$) are smooth enough, we can approximate $\mathcal{S}_f(\mathbf{A}) = \mathcal{S}_g(\mathbf{A}^T\mathbf{A})$ by simply summing over each window in the histogram, approximating $g(x)$ by its value at one end of the window.

This technique can be applied for any spectral sum. The number of windows required (controlled by α) and the histogram accuracy ϵ scale with the smoothness of $f(\cdot)$ and the desired accuracy in computing the sum, giving runtime dependencies on these parameters.

However, the most important factor determining the final runtime is the smallest value λ (corresponding to $(1-\alpha)^T$ in Theorem 1) which we must include in our histogram in order to approximate $S_f(\mathbf{A})$. The cost of computing the last window of the histogram is proportional to the cost of applying $s_\lambda(\mathbf{A}^T\mathbf{A})$, and hence of approximately computing $(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{A}\mathbf{y}$. Using stochastic gradient descent this depends on the average condition number of $(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})$.

Again considering the Schatten 1-norm for illustration, we can ignore any singular values with $\sigma_i(\mathbf{A}) \leq \frac{\epsilon}{n}\|\mathbf{A}\|_1$. Hiding ϵ dependence, this means that in our histogram, we must include any singular values of $\mathbf{A}^T\mathbf{A}$ with value $\sigma_i(\mathbf{A}^T\mathbf{A}) = \sigma_i^2(\mathbf{A}) \geq \frac{1}{n^2}\|\mathbf{A}\|_1^2$. This gives us effective average condition number after deflating off the top k singular values:

$$\bar{\kappa} = \frac{n^2\sum_{i=k+1}^n \sigma_i^2(\mathbf{A})}{n\|\mathbf{A}\|_1^2} \leq \frac{n^2\sigma_{k+1}(\mathbf{A})\cdot\sum_{i=k+1}^n \sigma_i(\mathbf{A})}{n\|\mathbf{A}\|_1^2} \leq \frac{n}{k} \tag{2}$$

where the last inequality follows since $\sigma_{k+1}(\mathbf{A}) \leq \frac{1}{k}\|\mathbf{A}\|_1$ and $\sum_{i=k+1}^n \sigma_i(\mathbf{A}) \leq \|\mathbf{A}\|_1$. Comparing to (1), this bound is better by an n/k factor.

Ignoring details and using a simplification of the runtimes in Theorem 1, we obtain an algorithm running in $\tilde{O}(\text{nnz}(\mathbf{A})k + nk^2)$ time to deflate k singular vectors, along with $\tilde{O}\left(\text{nnz}(\mathbf{A})\sqrt{\bar{\kappa}} + \sqrt{\text{nnz}(\mathbf{A})nk\bar{\kappa}}\right)$ time to approximate the spectral sum over the deflated matrix. Choosing k to balance these costs, gives our final runtimes. For the nuclear norm, using the bound on $\bar{\kappa}$ from (2), we set $k = n^{1/3}$ which gives $\bar{\kappa} = n^{2/3}$ and runtime $\tilde{O}(\text{nnz}(\mathbf{A})n^{1/3} + n^{3/2}\sqrt{d_s})$ where $d_s \leq n$ is the maximum row sparsity. For dense $\mathbf{A}$ this is $\tilde{O}(n^{2.33})$, which is faster than state of the art matrix multiplication time. It can be further accelerated using fast matrix multiplication. See details in Section 7 of the full paper [53].

Returning to our original hard example for intuition, we have $\mathbf{A}$ with $\sqrt{n}$ singular values at $\sqrt{n}$ and $\Theta(n)$ singular values at 1. Even without deflation, we have (again ignoring ϵ dependencies) $\bar{\kappa} = \frac{\sum_{i=1}^n \sigma_i^2(\mathbf{A})}{n\lambda} = \frac{n\|\mathbf{A}\|_F^2}{\|\mathbf{A}\|_1^2}$. Since $\|\mathbf{A}\|_F^2 = \Theta(n^{3/2})$ and $\|\mathbf{A}\|_1^2 = \Theta(n^2)$, this gives $\bar{\kappa} = \Theta(\sqrt{n})$. Thus, we can actually approximate $\|\mathbf{A}\|_1$ in just $\tilde{O}(\text{nnz}(\mathbf{A})n^{1/4})$ time.

With average condition number dependence, our performance is limited by a new hard case. Consider $\mathbf{A}$ with $n^{1/3}$ singular values at $n^{2/3}$ and $\Theta(n)$ at 1. $\mathbf{A}$'s average condition number is $\frac{n\|\mathbf{A}\|_F^2}{\|\mathbf{A}\|_1^2} = \Theta\left(\frac{n^{5/3}}{n}\right) = \Theta(n^{2/3})$ giving $\sqrt{\bar{\kappa}} = \Theta(n^{1/3})$. Further, unless we deflate off nearly all $n^{1/3}$ top singular vectors, we do not improve this bound significantly.

1.4 Lower Bound Approach

We now shift focus to our lower bounds, which explore the fundamental limits of spectrum approximation using fine-grained complexity. Fine-grained complexity has had much success for graph problems, string problems, and problems in other areas (see, e.g., [71] for a survey), and is closely tied to understanding the complexity of matrix multiplication. However, to the best of our knowledge it has not been applied broadly to problems in linear algebra.

Existing hardness results for linear algebraic problems tend to apply to restricted computational models such as arithmetic circuits [6], bilinear circuits or circuits with bounded coefficients and number of divisions [51, 58], algorithms for dense linear systems that can only add multiples of rows to each other [35, 36], and algorithms with restrictions on the dimension of certain manifolds defined in terms of the input [73, 14]. In contrast, we obtain conditional lower bounds for arbitrary polynomial time algorithms by showing that faster algorithms for them imply faster algorithms for canonical hard problems.

1.4.1 From Schatten 3-norm to Triangle Detection

We start with the fact that the number of triangles in any unweighted graph G is equal to $\text{tr}(\mathbf{A}^3)/6$, where $\mathbf{A}$ is the adjacency matrix. Any algorithm for approximating $\text{tr}(\mathbf{A}^3)$ to high enough accuracy therefore gives an algorithm for detecting if a graph has a triangle.

$\mathbf{A}$ is not PSD, so $\text{tr}(\mathbf{A}^3)$ is actually not a function of $\mathbf{A}$'s singular values – it depends on the signs of $\mathbf{A}$'s eigenvalues. However, the graph Laplacian given by $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the diagonal degree matrix, is PSD and we have:

$$\|\mathbf{L}\|_3^3 = \text{tr}(\mathbf{L}^3) = \text{tr}(\mathbf{D}^3) - 3\,\text{tr}(\mathbf{D}^2\mathbf{A}) + 3\,\text{tr}(\mathbf{D}\mathbf{A}^2) - \text{tr}(\mathbf{A}^3).$$

$\text{tr}(\mathbf{D}^2\mathbf{A}) = 0$ since $\mathbf{A}$ has an all 0 diagonal. Further, it is not hard to see that $\text{tr}(\mathbf{D}\mathbf{A}^2) = \text{tr}(\mathbf{D}^2)$. So this term and $\text{tr}(\mathbf{D}^3)$ are easy to compute exactly. Thus, if we approximate $\|\mathbf{L}\|_3^3$ up to additive error 6, we can determine if $\text{tr}(\mathbf{A}^3) = 0$ or $\text{tr}(\mathbf{A}^3) \geq 6$ and so detect if G contains a triangle. $\|\mathbf{L}\|_3^3 \leq 8n^4$ for any unweighted graph on n nodes, and hence computing this norm up to $(1 \pm \epsilon)$ relative error for $\epsilon = 3/(6n^4)$ suffices to detect a triangle. If we have an $O(n^\gamma \epsilon^{-c})$ time $(1 \pm \epsilon)$ approximation algorithm for the Schatten 3-norm, we can thus perform triangle detection in $O(n^{\gamma+4c})$ time.

Our strongest algorithmic result for the Schatten 3-norm requires just $\tilde{O}(n^2/\epsilon^3)$ time for dense matrices. Improving the ϵ dependence to $o(1/\epsilon^{(\omega-2)/4}) = O(1/\epsilon^{.09})$ for the current value of ω, would yield an algorithm for triangle detection running in $o(n^\omega)$ time for general graphs, breaking a longstanding runtime barrier for this problem. Even a $1/\epsilon^{1/3}$ dependence would give a sub-cubic time triangle detection algorithm, and hence could be used to give a subcubic time matrix multiplication algorithm via the reduction of [72].

1.4.2 Generalizing to Other Spectral Sums

We can generalize the above approach to the Schatten 4-norm by adding λ self-loops to each node of G, which corresponds to replacing $\mathbf{A}$ with $\lambda\mathbf{I} + \mathbf{A}$. We then consider $\text{tr}((\lambda\mathbf{I} + \mathbf{A})^4) = \|\lambda\mathbf{I} + \mathbf{A}\|_4^4$. This is the sum over all vertices of the number of paths that start at v_i and return to v_i in four steps. All of these paths are either (1) legitimate four cycles, (2) triangles combined with self loops, or (3) combinations of self-loops and two-step paths from a vertex v_i to one of its neighbors and back. The number of type (3) paths is exactly computable using the node degrees and number of self loops. Additionally, if the number of self loops λ is large enough, the number of type (2) paths will dominate the number of type (1) paths, even if there is just a single triangle in the graph. Hence, an accurate approximation to $\|\lambda\mathbf{I} + \mathbf{A}\|_4^4$ will give us the number of type (2) paths, from which we can easily compute the number of triangles.

This argument extends to a very broad class of spectral sums by considering a power series expansion of $f(x)$ and showing that for large enough λ, $\text{tr}\left(f(\lambda\mathbf{I} + \mathbf{A})\right)$ is dominated by $\text{tr}(\mathbf{A}^3)$ along with some exactly computable terms. Thus, an accurate approximation to this spectral sum allows us to determine the number of triangles in G. This approach works for any $f(x)$ that can be represented as a power series, with reasonably well-behaved coefficients on some interval of $\mathbb{R}^+$, giving bounds for all $\|\mathbf{A}\|_p$ with $p \neq 2$, the SVD entropy, $\log\det(\mathbf{A})$, $\text{tr}(\mathbf{A}^{-1})$, and $\text{tr}(\exp(\mathbf{A}))$.

We further show that approximating $\text{tr}(\mathbf{A}^{-1})$ for the $\mathbf{A}$ used in our lower bound can be reduced to computing all effective resistances of a certain graph Laplacian up to $(1 \pm \epsilon)$ error. Thus, we rule out highly accurate (with $1/\epsilon^c$ dependence for small c) approximation algorithms for all effective resistances, despite the existence of linear time system solvers (with $\log(1/\epsilon)$ error dependence) for Laplacians [63]. Effective resistances and leverage scores

are quantities that have recently been crucial to achieving algorithmic improvements to fundamental problems like graph sparsification [62] and regression [43, 12]. While crude multiplicative approximations to the quantities suffice for these problems, more recently computing these quantities has been used to achieve breakthroughs in solving maximum flow and linear programming [40], cutting plane methods [41], and sampling random spanning trees [50]. In each of these settings having more accurate estimates would be a natural route to either simplify or possibly improve existing results; we show that this is unlikely to be successful if the precision requirements are two high.

1.5 Paper Outline

Section 2: Preliminaries. We review notations that will be used throughout.

Section 3: Spectral Windows. We show how to approximately restrict the spectrum of a matrix to a small window. This is our main primitive for accessing the spectrum.

Section 4: Spectral Histogram. We show how our spectral window algorithms can be used to compute an approximate spectral histogram. We give applications to approximating general spectral sums, including the Schatten-p norms, Orlicz norms, and Ky Fan norms.

The last three sections are including in our full paper [53].

Section 5: Lower Bounds. We prove lower bounds showing that highly accurate spectral sum algorithms can be used to give algorithms for triangle detection and matrix multiplication.

Section 6: Improved Algorithms via Polynomial Approximation. We demonstrate how to tighten ϵ dependencies in our runtimes using a polynomial approximation approach.

Section 7: Optimized Runtime Bounds. We instantiate the techniques of Section 6 give our best runtimes for the Schatten p-norms and SVD entropy.

2 Preliminaries

Here we outline notation and conventions used throughout the paper.

Matrix Properties: For $\mathbf{A} \in \mathbb{R}^{n \times d}$ we assume w.l.o.g. that $d \leq n$. We let $\sigma_1(\mathbf{A}) \geq \ldots \geq \sigma_d(\mathbf{A}) \geq 0$ denote the matrix's singular values, $\text{nnz}(\mathbf{A})$ denote the number of non-zero entries, and $d_s(\mathbf{A}) \in [\text{nnz}(\mathbf{A})/n, d]$ denote the maximum number of non-zero entries in a row.

Fast Matrix Multiplication: Let $\omega \approx 2.3729$ denote the current best exponent of fast matrix multiplication [70, 21]. Additionally, let $\omega(\gamma)$ denote the exponent such that it takes $O\left(d^{\omega(\gamma)}\right)$ time to multiply a $d \times d$ matrix by a $d \times d^\gamma$ matrix for any $\gamma \leq 1$. $\omega(\gamma) = 2$ for $\gamma < \alpha$ where $\alpha > 0.31389$ and $\omega(\gamma) = 2 + (\omega - 2)\frac{\gamma - \alpha}{1 - \alpha}$ for $\gamma \geq \alpha$ [39, 21]. For $\gamma = 1$, $\omega(\gamma) = \omega$.

Asymptotic Notation: We use $\tilde{O}(\cdot)$ notation to hide poly-logarithmic factors in the input parameters, including dimension, failure probability, and error ϵ. We use 'with high probability' or 'w.h.p.' to refer to events happening with probability at least $1 - 1/d^c$ for some constant c, where d is our smaller input dimension.

Other: We denote $[d] \stackrel{\text{def}}{=} \{0, \ldots, d\}$. For any $\mathbf{y} \in \mathbb{R}^d$ and PSD $\mathbf{N} \in \mathbb{R}^{d \times d}$, we denote $\|\mathbf{y}\|_{\mathbf{N}} \stackrel{\text{def}}{=} \sqrt{\mathbf{y}^T \mathbf{N} \mathbf{y}}$.

3 Approximate Spectral Windows via Ridge Regression

In this section, we give state-of-the-art results for approximating spectral windows over $\mathbf{A}$. As discussed, our algorithms will split $\mathbf{A}$'s spectrum into small slices using these window functions, performing trace estimation to estimate the number of singular values on each window and producing an approximate spectral histogram.

In Section 3.1 we show how to efficiently apply smooth approximations to threshold functions of the spectrum given access to an algorithm for solving regularized regression problems with the matrix. In Section 3.2 we then provide the fastest known algorithms for the regression problems in the given parameter regimes using both stochastic gradient methods and traditional solvers. Departing from our high level description in Section 1.3, we incorporate singular vector deflation directly into our system solvers to reduce condition number. This simplifies our final algorithms but has the same effect as the techniques discussed in Section 1.3. In Section 3.3 we give runtimes for applying smooth approximations to window functions of the spectrum, which is the main export of this section.

3.1 Step Function Approximation

To compute a window over $\mathbf{A}$'s spectrum, we will combine two threshold functions at the boundaries of the window. We begin by discussing how to compute these threshold functions.

Let $s_\lambda : [0,1] \to [0,1]$ be the threshold function at λ. $s_\lambda(x) = 1$ for $x \in [\lambda, 1]$ and 0 for $x \in [0, \lambda)$. For some gap γ we define a soft step function by:

▶ **Definition 2** (Soft Step Function). *$s_\lambda^\gamma : [0,1] \to [0,1]$ is a γ-soft step at $\lambda > 0$ if:*

$$s_\lambda^\gamma(x) = \begin{cases} 0 \text{ for } x \in [0, (1-\gamma)\lambda] \\ 1 \text{ for } x \in [\lambda, 1] \end{cases} \quad \text{and } s_\lambda^\gamma(x) \in [0,1] \text{ for } x \in [(1-\gamma)\lambda, \lambda]. \tag{3}$$

We use the strategy from [20], which, for $\mathbf{A}$ with $\|\mathbf{A}\|_2 \leq 1$ shows how to efficiently multiply a γ-soft step $s_\lambda^\gamma(\mathbf{A}^T\mathbf{A})$ by any $\mathbf{y} \in \mathbb{R}^d$ using ridge regression. The trick is to first approximately compute $\mathbf{A}^T\mathbf{A}(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{y} = r_\lambda(\mathbf{A}^T\mathbf{A})\mathbf{y}$ where $r_\lambda(x) \stackrel{\text{def}}{=} \frac{x}{x+\lambda}$. Then, note that $s_{1/2}(r_\lambda(x)) = s_\lambda(x)$. Additionally, the symmetric step function $s_{1/2}$ can be well approximated with a low degree polynomial. Specifically, there exists a polynomial of degree $O(\gamma^{-1}\log(1/(\gamma\epsilon)))$ that is within additive ϵ of a true γ-soft step at $1/2$ and can be applied stably such that any error in computing $r_\lambda(\mathbf{A}^T\mathbf{A})$ remains bounded. The upshot, following from Theorem 7.4 of [1] is:

▶ **Lemma 3** (Step Function via Ridge Regression). *Let $\mathcal{A}(\mathbf{A}, \mathbf{y}, \lambda, \epsilon)$ be an algorithm that on input $\mathbf{A} \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \mathbb{R}^d$, $\lambda, \epsilon > 0$ returns $\mathbf{x} \in \mathbb{R}^d$ such that $\left\|\mathbf{x} - (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{y}\right\|_2 \leq \epsilon \|\mathbf{y}\|_2$ with high probability. Then there is an algorithm $\mathcal{B}(\mathbf{A}, \mathbf{y}, \lambda, \gamma, \epsilon)$ which on input $\mathbf{A} \in \mathbb{R}^{n\times d}$ with $\|\mathbf{A}\|_2 \leq 1$, $\mathbf{y} \in \mathbb{R}^d$, $\lambda \in (0,1)$, and $\gamma, \epsilon > 0$, returns $\mathbf{x} \in \mathbb{R}^d$ with*

$$\left\|\mathbf{x} - s_\lambda^\gamma(\mathbf{A}^T\mathbf{A})\mathbf{y}\right\|_2 \leq \epsilon \|\mathbf{y}\|_2$$

where s_λ^γ is a γ-soft step at λ (i.e. satisfies Defn. 2). $\mathcal{B}(\mathbf{A}, \mathbf{y}, \lambda, \gamma, \epsilon)$ requires $O(\gamma^{-1}\log(1/\epsilon\gamma))$ calls to $\mathcal{A}(\mathbf{A}, \mathbf{y}, \lambda, \epsilon')$ along with $O(\text{nnz}(\mathbf{A}))$ additional runtime, where $\epsilon' = \text{poly}(1/(\gamma\epsilon))$.

3.2 Ridge Regression

Given Lemma 3, to efficiently compute $s_\lambda^\gamma(\mathbf{A}^T\mathbf{A})\mathbf{y}$ for $s_\lambda^\gamma(\cdot)$ satisfying Definition 2, it suffices to quickly approximate $(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{y}$ (i.e. to provide the algorithm $\mathcal{A}(\mathbf{A}, \mathbf{y}, \lambda, \epsilon)$ used in

the lemma). In this section we provide two theorems which give the state-of-the-art ridge regression running times achievable in our parameter regime, using sampling, acceleration, and singular value deflation.

Naively, computing $(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})^{-1}\mathbf{y}$ using an iterative system solver involves a dependence on the condition number $\sigma_1^2(\mathbf{A})/\lambda$. In our theorems, this condition number is replaced by a deflated condition number depending on $\sigma_k^2(\mathbf{A})$ for some input parameter $k \in [d]$. We achieve this improved dependence following the techniques presented in [23]. We first approximate the top k singular vectors of $\mathbf{A}$ and then construct a preconditioner based on this approximation, which significantly flattens the spectrum of the matrix. By using this preconditioner in conjunction with a stochastic gradient based linear system solver, we further enjoy an average condition number dependence. The following theorem summarizes the results.

▶ **Theorem 4** (Ridge Regression – Accelerated Preconditioned SVRG). *For any $\mathbf{A} \in \mathbb{R}^{n\times d}$ and $\lambda > 0$, let $\mathbf{M}_\lambda \stackrel{\text{def}}{=} \mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}$. Let $\bar{\kappa} \stackrel{\text{def}}{=} \frac{k\sigma_k^2(\mathbf{A})+\sum_{i=k+1}^{d}\sigma_i^2(\mathbf{A})}{d\lambda}$ where $k \in [d]$ is an input parameter. There is an algorithm that builds a preconditioner for $\mathbf{M}_\lambda$ using precomputation time $\tilde{O}(\text{nnz}(\mathbf{A})k + dk^{\omega-1})$ for sparse $\mathbf{A}$ or $\tilde{O}(nd^{\omega(\log_d k)-1})$ time for dense $\mathbf{A}$, and for any input $\mathbf{y} \in \mathbb{R}^d$, returns $\mathbf{x}$ such that with high probability $\left\|\mathbf{x} - \mathbf{M}_\lambda^{-1}\mathbf{y}\right\|_{\mathbf{M}_\lambda} \leq \epsilon \left\|\mathbf{y}\right\|_{\mathbf{M}_\lambda^{-1}}$ in*

$$\tilde{O}\left(\text{nnz}(\mathbf{A}) + \sqrt{\text{nnz}(\mathbf{A})[d \cdot d_s(\mathbf{A}) + dk]\bar{\kappa}}\right)$$

time for sparse $\mathbf{A}$ or $\tilde{O}\left(nd + n^{1/2}d^{3/2}\sqrt{\bar{\kappa}})\right)$ time for dense $\mathbf{A}$.

We give a proof in Appendix A of the full paper [53]. Note that the ϵ dependence in the runtime is $\log(1/\epsilon)$ and so is hidden by the $\tilde{O}(\cdot)$ notation.

When $\mathbf{A}$ is dense, the runtime of Theorem 4 is essentially the best known. Due to its average condition number dependence, the method always outperforms traditional iterative methods, like conjugate gradient, up to log factors. However, in the sparse case, traditional approaches can give faster runtimes if the rows of $\mathbf{A}$ are not uniformly sparse and $d_s(\mathbf{A})$ is large. We have the following, also proved in Appendix A of the full paper using the same deflation-based preconditioner as in Theorem 4:

▶ **Theorem 5** (Ridge Regression – Preconditioned Iterative Method). *For any $\mathbf{A} \in \mathbb{R}^{n\times d}$ and $\lambda > 0$, let $\mathbf{M}_\lambda \stackrel{\text{def}}{=} \mathbf{A}^T\mathbf{A}+\lambda\mathbf{I}$ and $\hat{\kappa} \stackrel{\text{def}}{=} \frac{\sigma_{k+1}^2(\mathbf{A})}{\lambda}$ where $k \in [d]$ is an input parameter. There is an algorithm that builds a preconditioner for $\mathbf{M}_\lambda$ using precomputation time $\tilde{O}(\text{nnz}(\mathbf{A})k+dk^{\omega-1})$, and for any input $\mathbf{y} \in \mathbb{R}^d$, returns $\mathbf{x}$ such that with high probability $\left\|\mathbf{x} - \mathbf{M}_\lambda^{-1}\mathbf{y}\right\|_{\mathbf{M}_\lambda} \leq \epsilon \left\|\mathbf{y}\right\|_{\mathbf{M}_\lambda^{-1}}$ in $\tilde{O}\left((\text{nnz}(\mathbf{A}) + dk)\lceil\sqrt{\hat{\kappa}}\rceil\right)$ time.*

3.3 Overall Runtimes For Spectral Windows

Combined with Lemma 3, the ridge regression routines above let us efficiently compute soft step functions of $\mathbf{A}$'s spectrum. Composing step functions then gives our key computational primitive: the ability to approximate soft window functions that restrict $\mathbf{A}$'s spectrum to a specified range. We first define our notion of soft window functions and then discuss runtimes. The corresponding Theorem 7 is our main tool for spectrum approximation.

▶ **Definition 6** (Soft Window Function). Given $0 < a < b$, and $\gamma \in [0, 1]$, $h_{[a,b]}^\gamma : [0, 1] \to [0, 1]$ is a γ-soft window for $[a, b]$ if $h_{[a,b]}^\gamma(x) \in [0, 1]$ for $x \in [(1 - \gamma)a, a] \cup [b, (1 + \gamma)b]$ and:

$$h_{[a,b]}^\gamma(x) = \begin{cases} 1 \text{ for } x \in [a, b] \\ 0 \text{ for } x \in [0, (1 - \gamma)a] \cup [(1 + \gamma)b, 1] \end{cases}$$

▶ **Theorem 7** (Spectral Windowing). *For $\mathbf{A} \in \mathbb{R}^{n\times d}$ with $\|\mathbf{A}\|_2 \leq 1$, $\mathbf{y} \in \mathbb{R}^d$, and $a, b, \gamma, \epsilon \in (0, 1]$, with $a < b$, there is an algorithm $\mathcal{W}(\mathbf{A}, \mathbf{y}, a, b, \gamma, \epsilon)$ that returns $\mathbf{x}$ satisfying w.h.p.:*

$$\left\|\mathbf{x} - h^{\gamma}_{[a,b]}(\mathbf{A}^T\mathbf{A})\mathbf{y}\right\|_2 \leq \epsilon \|\mathbf{y}\|_2$$

where $h^{\gamma}_{[a,b]}$ is a soft window function satisfying Def. 6. Let $\bar{\kappa} \stackrel{\text{def}}{=} \frac{k\sigma_k^2(\mathbf{A}) + \sum_{i=k+1}^{d}\sigma_i^2(\mathbf{A})}{d\cdot a}$ and $\hat{\kappa} \stackrel{\text{def}}{=} \frac{\sigma_{k+1}^2(\mathbf{A})}{a}$ where $k \in [d]$ is an input parameter. The algorithm uses precomputation time $\tilde{O}(\text{nnz}(\mathbf{A})k + dk^{\omega-1})$ for sparse $\mathbf{A}$ or $\tilde{O}(nd^{\omega(\log_d k)-1})$ for dense $\mathbf{A}$ after which given any $\mathbf{y}$ it returns $\mathbf{x}$ in time:

$$\tilde{O}\left(\frac{\text{nnz}(\mathbf{A}) + \sqrt{\text{nnz}(\mathbf{A})[d\cdot d_s(\mathbf{A}) + dk]\bar{\kappa}}}{\gamma}\right) \quad \text{or} \quad \tilde{O}\left(\frac{(\text{nnz}(\mathbf{A}) + dk)\lceil\sqrt{\hat{\kappa}}\rceil}{\gamma}\right)$$

for sparse $\mathbf{A}$ or $\tilde{O}\left(\frac{nd + n^{1/2}d^{3/2}\sqrt{\bar{\kappa}})}{\gamma}\right)$ for dense $\mathbf{A}$.

Proof. If $b \geq 1/(1+\gamma)$ then we can simply define $h^{\gamma}_{[a,b]}(x) = s^{\gamma}_a(x)$ for any s^{γ}_a satisfying Definition 2. Otherwise, given soft steps s^{γ}_a and $s^{\gamma/2}_{(1+\gamma)b}$ satisfying Definition 2, we can define $h^{\gamma}_{[a,b]}(x) = s^{\gamma}_a(x)\cdot(1 - s^{\gamma/2}_{(1+\gamma)b}(x))$. Since $\frac{\gamma}{2} \leq \frac{\gamma}{1+\gamma}$ we can verify that this will be a valid soft window function for $[a, b]$ (i.e. satisfy Definition 6). Further, we have for any $\mathbf{y} \in \mathbb{R}^d$:

$$h^{\gamma}_{[a,b]}(\mathbf{A}^T\mathbf{A})\mathbf{y} = s^{\gamma}_a(\mathbf{A}^T\mathbf{A})(\mathbf{I} - s^{\gamma/2}_{(1+\gamma)b}(\mathbf{A}^T\mathbf{A}))\mathbf{y}. \tag{4}$$

We can compute $s^{\gamma}_a(\mathbf{A}^T\mathbf{A})\mathbf{y}$ and $s^{\gamma/2}_{(1+\gamma)b}(\mathbf{A}^T\mathbf{A})\mathbf{y}$ each up to error $\epsilon\|\mathbf{y}\|_2$ via Lemma 3. This gives the error bound in the theorem, since we have both $\left\|s^{\gamma}_a(\mathbf{A}^T\mathbf{A})\right\|_2 \leq 1$ and $\left\|\mathbf{I} - s^{\gamma/2}_{(1+\gamma)b}(\mathbf{A}^T\mathbf{A})\right\|_2 \leq 1$ so the computation in (4) does not amplify error. Our runtime follows from combining Theorems 4 and 5 with $\lambda = a, b$ with Lemma 3. The errors in these theorems are measured with respect to $\|\cdot\|_{\mathbf{M}_\lambda}$. To obtain the error in $\|\cdot\|_2$ as used by Lemma 3, we simply apply the theorems with $\epsilon' = \epsilon\kappa(\mathbf{M}_\lambda)$ which incurs an additional $\log(\kappa(\mathbf{M}_\lambda))$ cost. Since $a < b$ the runtime is dominated by the computation of $s^{\gamma}_a(\mathbf{A}^T\mathbf{A})\mathbf{y}$, which depends on the condition number $\bar{\kappa} \stackrel{\text{def}}{=} \frac{k\sigma_k^2(\mathbf{A}) + \sum_{i=k+1}^{d}\sigma_i^2(\mathbf{A})}{d\cdot a}$ when using SVRG (Theorem 4) or $\hat{\kappa} \stackrel{\text{def}}{=} \frac{\sigma_{k+1}^2(\mathbf{A})}{a}$ for a traditional iterative solver (Theorem 5). ◀

4 Approximating Spectral Sums via Spectral Windows

We now use the window functions discussed in Section 3 to compute an approximate spectral histogram of $\mathbf{A}$. We give our main histogram algorithm and approximation guarantee in Section 4.1. In Section 4.2 we show how this guarantee translates to accurate spectral sum approximation for any smooth and sufficiently quickly growing function $f(x)$. In Section 4.3 we apply this general result to approximating the Schatten p-norms for all real $p > 0$. We give applications to bounded Orlicz norms and the Ky Fan norms in our full paper [53].

4.1 Approximate Spectral Histogram

Our main histogram approximation method is given as Algorithm 1. The algorithm is reasonably simple. Assuming $\|\mathbf{A}\|_2 \leq 1$ (this is w.l.o.g. as we can just scale the matrix), and given cutoff λ, below which we will not evaluate $\mathbf{A}'s$ spectrum, we split the range $[\lambda, 1]$ into successive windows $R_0, ..., R_T$ where $R_t = [a_1(1-\alpha)^t, a_1(1-\alpha)^{t-1}]$. Here α determines the

Algorithm 1 Approximate Spectral Histogram

Input: $\mathbf{A} \in \mathbb{R}^{n \times d}$ with $\|\mathbf{A}\|_2 \leq 1$, accuracy parameters $\epsilon_1, \epsilon_2 \in (0,1)$, width parameter $\alpha \in (0,1)$, and minimum singular value parameter $\lambda \in (0,1)$.

Output: Set of range boundaries $a_{T+1} < a_T < ... < a_1 < a_0$ and counts $\{\tilde{b}_0, ..., \tilde{b}_T\}$ where $\tilde{b}_t$ approximates the number of squared singular values of $\mathbf{A}$ on $[a_{t+1}, a_t]$.

Set $\gamma = c_1\epsilon_2\alpha$, $T = \lceil \log_{(1-\alpha)} \lambda \rceil$, and $S = \frac{\log n}{c_2\epsilon_1^2}$.
Set $a_0 = 1$ and choose a_1 uniformly at random in $[1 - \alpha/4, 1]$.
Set $a_t = a_1(1-\alpha)^{t-1}$ for $2 \leq t \leq T+1$.
for $t = 0 : T$ **do** ▷ Iterate over histogram buckets.
  Set $\tilde{b}_t = 0$. ▷ Initialize bucket size estimate.
  for $s = 1 : S$ **do** ▷ Estimate bucket size via trace estimation.
    Choose $\mathbf{y} \in \{-1, 1\}^d$ uniformly at random.
    Set $\tilde{b}_t = \tilde{b}_t + \frac{1}{S} \cdot \mathbf{y}^T \mathcal{W}(\mathbf{A}^T\mathbf{A}, \mathbf{y}, a_{t+1}, a_t, \gamma, c_3\epsilon_1^2/n)$. ▷ Apply soft window via Thm 7.
    If $\tilde{b}_t \leq 1/2$ set $\tilde{b}_t = 0$. ▷ Round small estimates to ensure relative error.
  end for
end for
return a_1 and $\tilde{b}_t$ for $t = 0 : T$. ▷ Output histogram representation.

width of our windows. In our final spectral approximation algorithms, we will set $\alpha = \Theta(\epsilon)$. a_1 is a random shift, which insures that, in expectation, the boundaries of our soft windows do not overlap too many singular values. This argument requires that most of the range $[\lambda, 1]$ is not covered by boundary regions. Thus, we set the steepness parameter $\gamma = \Theta(\epsilon_2\alpha)$ where ϵ_2 will control the error introduced by the boundaries. Finally, we iterate over each window, applying trace estimation to approximate the singular value count in each window.

In our final algorithms, the number of windows and samples required for trace estimation will be $\tilde{O}(\text{poly}(1/\epsilon))$. The dominant runtime cost comes from the lowest range R_T, which incurs a dependence on the condition number of $\mathbf{A}^T\mathbf{A} + a_T\mathbf{I}$ with $a_T = \Theta(\lambda)$.

▶ **Theorem 8** (Histogram Approximation). *Let a_1, $\tilde{b}_0, ..., \tilde{b}_T$ be output by Algorithm 1. Let $R_0 = [a_1, 1]$, $R_t = [a_1(1-\alpha)^t, a_1(1-\alpha)^{t-1}]$ for $t \geq 1$, and $b_t = \left|\{i : \sigma_i^2(\mathbf{A}) \in R_t\}\right|$ be the number of squared singular values of $\mathbf{A}$ on the range R_t. Then, for sufficiently small c_1, c_2, c_3, with probability 99/100, for all $t \in \{0, ..., \lceil \log_{(1-\alpha)} \lambda \rceil\}$, $\tilde{b}_t$ output by Algorithm 1 satisfies:*

$$(1-\epsilon_1)b_t \leq \tilde{b}_t \leq (1+\epsilon_1)b_t + \lceil \log_{(1-\alpha)} \lambda \rceil \cdot \epsilon_2(b_{t-1} + b_{t+1}).$$

That is, $\tilde{b}_t$ approximates the number of singular values of the range R_t up to multiplicative $(1 \pm \epsilon_1)$ error and additive error $\lceil \log_{(1-\alpha)} \lambda \rceil \cdot \epsilon_2(b_{t-1} + b_{t+1})$. Note that by setting $\epsilon_2 \leq \frac{\epsilon_1}{\lceil \log_{(1-\alpha)} \lambda \rceil}$, the error on each bucket is just multiplicative on its size plus the size of the two adjacent buckets, which contain singular values in nearby ranges. For simplicity we assume $\mathbf{A}$ passed to the algorithm has $\|\mathbf{A}\|_2 \leq 1$. This is w.l.o.g.: we can estimate $\|\mathbf{A}\|_2$ in $\tilde{O}(\text{nnz}(\mathbf{A}))$ time via the power or Lanczos methods [38, 52], and scale down the matrix appropriately.

The runtime of Algorithm 1 is dominated by the calls to $\mathcal{W}$ for the bucket corresponding to the smallest singular values, with $a_T = \Theta(\lambda)$. This runtime is given by Theorem 7. Since balancing the deflation parameter k with the minimum squared singular value λ considered can be complex, we wait to instantiate full runtimes until employing Algorithm 1 for specific spectral sum computations. See full paper [53] for a proof of Theorem 8.

4.2 Application to General Spectral Sums

While Theorem 8 is useful on its own, we now apply it to approximate a broad class of spectral sums. We need two assumptions. First, for the histogram discretization to be relatively accurate, we need the sum function to be smooth. Second, it is expensive to compute the histogram over very small singular values of $\mathbf{A}$ (i.e. with λ very small in Algorithm 1) as this makes the condition number in Theorem 7 large. So it is important that small singular values cannot contribute significantly to our sum. We start with the following definition:

▶ **Definition 9** (Multiplicative Smoothness). *$f : \mathbb{R}^+ \to \mathbb{R}^+$ is δ_f-multiplicatively smooth if for some $\delta_f \geq 1$, for all x, $|f'(x)| \leq \delta_f \frac{f(x)}{x}$.*

For the Schatten-p norm, $f(x) = x^p$, $f'(x) = px^{p-1}$ and so f is p-multiplicatively smooth. We have the following claim, proven in Appendix D of the full paper [53]:

▶ **Claim 10.** *Let f be a δ_f-multiplicatively smooth function. For all $x, y \in \mathbb{R}^+$ and $c \in (0, \frac{1}{3\delta_f})$*

$$y \in [(1-c)x, (1+c)x] \Rightarrow f(y) \in [(1-3\delta_f c)f(x), (1+3\delta_f c)f(x)].$$

We now give our general approximation theorem, showing that any spectral sum depending on sufficiently smooth and rapidly growing f can be computed using Algorithm 1:

▶ **Theorem 11** (Spectral Sums Via Approximate Histogram). *Consider any $\mathbf{A} \in \mathbb{R}^{n \times d}$ and any function $f : \mathbb{R}^+ \to \mathbb{R}^+$ satisfying:*

- *Multiplicative Smoothness: For some $\delta_f \geq 1$, f is δ_f-multiplicatively smooth (Defn. 9).*
- *Small Tail: For any $\epsilon > 0$ there exists $\lambda_f(\epsilon)$ such that for $x \in [0, \lambda_f(\epsilon)]$, $f(x) \leq \frac{\epsilon}{n}\mathcal{S}_f(\mathbf{A})$*

Given error parameter $\epsilon \in (0,1)$ and spectral norm estimate $M \in [\|\mathbf{A}\|_2, 2\|\mathbf{A}\|_2]$, for sufficiently small constant c, if we run Algorithm 1 on $\frac{1}{M}\mathbf{A}$ with input parameters $\epsilon_1, \epsilon_2 = c\epsilon$, $\alpha = c\epsilon/\delta_f$ and $\lambda = \lambda_f(c\epsilon)^2/M^2$ then with probability $99/100$, letting a_1, $\tilde{b}_0, ..., \tilde{b}_T$ be the outputs of the algorithm and $g(x) = f(x^{1/2})$:

$$(1-\epsilon)\mathcal{S}_f(\mathbf{A}) \leq \sum_{t=0}^{T} g(M^2 \cdot a_1(1-\alpha)^t) \cdot \tilde{b}_t \leq (1+\epsilon)\mathcal{S}_f(\mathbf{A}).$$

For parameter $k \in [d]$, letting $\bar{\kappa} \stackrel{\text{def}}{=} \frac{k\sigma_k^2(\mathbf{A}) + \sum_{i=k+1}^{d}\sigma_i^2(\mathbf{A})}{d \cdot \lambda}$ and $\hat{\kappa} \stackrel{\text{def}}{=} \frac{\sigma_{k+1}^2(\mathbf{A})}{\lambda}$, the algorithm runs in

$$\tilde{O}\left(\text{nnz}(\mathbf{A})k + dk^{\omega-1} + \frac{\text{nnz}(\mathbf{A}) + \sqrt{\text{nnz}(\mathbf{A})[d \cdot d_s(\mathbf{A}) + dk]\bar{\kappa}}}{\epsilon^5/(\delta_f^2 \log(1/\lambda))}\right)$$

$$\text{or } \tilde{O}\left(\text{nnz}(\mathbf{A})k + dk^{\omega-1} + \frac{(\text{nnz}(\mathbf{A}) + dk)\lceil\sqrt{\hat{\kappa}}\rceil}{\epsilon^5/(\delta_f^2 \log(1/\lambda))}\right)$$

time for sparse $\mathbf{A}$ or $\tilde{O}\left(nd^{\omega(\log_d k)-1} + \frac{nd + n^{1/2}d^{3/2}\sqrt{\bar{\kappa}})}{\epsilon^5/(\delta_f^2 \log(1/\lambda))}\right)$ for dense $\mathbf{A}$.

That is, we accurately approximate $\mathcal{S}_f(\mathbf{A})$ by discretizing over the histogram of Algorithm 1. We can boost our probability of success to $1-\delta$ by repeating the algorithm $\Theta(\log(1/\delta))$ times and taking the median of the outputs. Theorem 11 is proven in our full paper [53].

4.3 Application to Schatten-p Norms

Theorem 11 is very general, allowing us to approximate any function satisfying a simple smoothness condition as long as the smaller singular values of $\mathbf{A}$ cannot contribute significantly to $\mathcal{S}_f(\mathbf{A})$. As an example, we show how it gives the fastest known algorithms for Schatten-p norm estimation. We will not go into all runtime tradeoffs now as our best runtimes will be worked out in detail in Sections 6 and 7 of the full paper [53].

▶ **Corollary 12** (Schatten-p norms via Histogram Approximation). *For any $\mathbf{A} \in \mathbb{R}^{n\times n}$ with uniformly sparse rows (i.e. $d_s(\mathbf{A}) = O(\text{nnz}(\mathbf{A})/n)$), given error parameter $\epsilon \in (0,1)$ and $M \in [\|\mathbf{A}\|_2, 2\|\mathbf{A}\|_2]$, if we run Algorithm 1 on $\frac{1}{M}\mathbf{A}$ with $\epsilon_1, \epsilon_2 = c\epsilon$, $\alpha = c\epsilon/\max\{1,p\}$ and $\lambda = \frac{1}{M^2}\left(\frac{c\epsilon}{n}\|\mathbf{A}\|_p^p\right)^{2/p}$ for sufficiently small constant c then with probability $99/100$, $(1-\epsilon)\|\mathbf{A}\|_p^p \leq \sum_{t=0}^{T}\left[M^2 a_1(1-\alpha)^t\right]^{p/2}\cdot\tilde{b}_t \leq (1+\epsilon)\|\mathbf{A}\|_p^p$. The algorithm's runtime is:*

$$\tilde{O}\left(\frac{\text{nnz}(\mathbf{A})p^2}{\epsilon^{5+1/p}}\right) \text{ for } p \geq 2 \quad \text{and} \quad \tilde{O}\left(\frac{\text{nnz}(\mathbf{A})n^{\frac{1/p-1/2}{1/p+1/2}} + n^{\frac{5/p-1/2}{2/p+1}}\sqrt{d_s(\mathbf{A})}}{p\cdot\epsilon^{5+1/p}}\right) \text{ for } p \leq 2.$$

For dense inputs this can be sped up to $\tilde{O}\left(\frac{n^{\frac{2.3729-.1171p}{1+.0346p}}}{p\cdot\epsilon^{5+1/p}}\right)$ using fast matrix multiplication.

For constant $\epsilon, p > 2$ the first runtime is $\tilde{O}(\text{nnz}(\mathbf{A}))$, and for the nuclear norm ($p = 1$), for constant ϵ the second runtime gives $\tilde{O}(\text{nnz}(\mathbf{A})n^{1/3} + n^{3/2}\sqrt{d_s(\mathbf{A})})$ which is at worst $\tilde{O}(\text{nnz}(\mathbf{A})n^{1/3} + n^2)$. For dense matrices, the nuclear norm estimation time is $\tilde{O}(n^{2.18})$ using fast matrix multiplication. It is already $\tilde{O}(n^{2.33})$, without using fast matrix multiplication.

Note that we can compute the spectral norm approximation used to scale $\mathbf{A}$ via the Lanczos or power method in $\tilde{O}(\text{nnz}(\mathbf{A}))$ time. λ depends on $\|\mathbf{A}\|_p$ which we are estimating. However, as we will discuss in the proof, we can use a rough estimate for $\|\mathbf{A}\|_p$ which suffices. λ could also be identified via binary search. We can start with $\lambda = \sigma_1(\mathbf{A})^2/M^2$ and successively decrease λ running Algorithm 1 up to the stated runtime bounds. If it does not finish in the allotted time, we know that we have set λ too small. Thus, we can output the result with the smallest λ such that the algorithm completes within in the stated bounds.

Proof. We invoke Theorem 11 with $f(x) = x^p$. We have $f'(x) = p\frac{f(x)}{x}$ so $\delta_f = \max\{1,p\}$ and our setting of $\alpha = c\epsilon/\max\{1,p\}$ suffices. Additionally, for any c, we can set $\lambda_f(c\epsilon) = \left(\frac{c\epsilon}{n}\|\mathbf{A}\|_p^p\right)^{1/p} = \frac{c^{1/p}\epsilon^{1/p}}{n^{1/p}}\|\mathbf{A}\|_p$ and so our setting of λ suffices. Thus the accuracy bound follows from Theorem 11. We now consider runtime. For $p \geq 2$:

$$\bar{\kappa} = \frac{k\sigma_k^2(\mathbf{A}) + \sum_{i=k+1}^{n}\sigma_i^2(\mathbf{A})}{n\lambda} \leq \frac{n^{2/p-1}}{\epsilon^{2/p}}\cdot\frac{\|\mathbf{A}\|_F^2}{\|\mathbf{A}\|_p^2}.$$

We can bound $\|\mathbf{A}\|_F \leq n^{1/2-1/p}\|\mathbf{A}\|_p$ and so have $\bar{\kappa} \leq \frac{1}{\epsilon^{2/p}}$. For $p < 2$ we have:

$$\bar{\kappa} = \frac{n^{2/p-1}}{\epsilon^{2/p}}\cdot\frac{k\sigma_k^2(\mathbf{A}) + \sum_{i=k+1}^{n}\sigma_i^2(\mathbf{A})}{\|\mathbf{A}\|_p^2} \leq \frac{n^{2/p-1}}{\epsilon^{2/p}}\cdot\frac{\sigma_k^{2-p}(\mathbf{A})\sum_{i=1}^{n}\sigma_i^p(\mathbf{A})}{\|\mathbf{A}\|_p^2} = \frac{n^{2/p-1}}{\epsilon^{2/p}}\cdot\frac{\sigma_k^{2-p}(\mathbf{A})}{\|\mathbf{A}\|_p^{2-p}}.$$

Using the fact that $\sigma_k^p(\mathbf{A}) \leq \frac{1}{k}\|\mathbf{A}\|_p^p$ we have the tradeoff between k and $\bar{\kappa}$:

$$\bar{\kappa} \leq \frac{1}{\epsilon^{2/p}}\left(\frac{n}{k}\right)^{2/p-1}. \tag{5}$$

As mentioned, λ depends on the value of $\|\mathbf{A}\|_p$. We can simply lower bound $\|\mathbf{A}\|_p^p$ by $k\sigma_k^p(\mathbf{A})$, which we estimate up to multiplicative error when performing deflation. We can use

this lower bound to set λ. Our estimated λ will only be smaller than the true value, giving a better approximation guarantee and the above condition number bound will still hold.

Recall that for $f(x) = x^p$, $\delta_f = \max\{1, p\}$. Correspondingly, $\log(1/\lambda) = \tilde{O}(\max\{1, 1/p\})$ and so $\delta_f^2 \log(1/\lambda) = \max\{p^2, 1/p\}$. Plugging into the first runtime of Theorem 11, using the uniform sparsity assumption and the fact that $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$ we have:

$$\tilde{O}\left(\text{nnz}(\mathbf{A})k + nk^{\omega-1} + \frac{\text{nnz}(\mathbf{A})\sqrt{\bar{\kappa}} + \sqrt{\text{nnz}(\mathbf{A})nk\bar{\kappa}}}{\epsilon^5/(\max\{p^2, 1/p\}}\right).$$

For $p \geq 2$ we just set $k = 0$ and have $\tilde{O}(\text{nnz}(\mathbf{A})p^2/\epsilon^{5+1/p})$ runtime by our bound $\bar{\kappa} \leq \frac{1}{\epsilon^{2/p}}$. For $p \leq 2$, not trying to optimize $\text{poly}(1/\epsilon)$ terms, we write the runtime as

$$\tilde{O}\left(nd_s(\mathbf{A})k + nk^{\omega-1} + \frac{nd_s(\mathbf{A})\sqrt{\bar{\kappa}} + n\sqrt{d_s(\mathbf{A})k\bar{\kappa}}}{\epsilon^5 p}\right).$$

Balancing the first two coefficients on n, set $k = n^{\frac{1/p-1/2}{1/p+1/2}}$ which gives $\sqrt{\bar{\kappa}} = n^{\frac{1/p-1/2}{1/p+1/2}}$ by (5) and so $nd_s(\mathbf{A})k = nd_s(\mathbf{A})\sqrt{\bar{\kappa}}$. We then have $n\sqrt{d_s(\mathbf{A})k\bar{\kappa}} = n\sqrt{d_s(\mathbf{A})}k^{3/2} = n^{\frac{5/p-1/2}{2/p+1}}\sqrt{d_s(\mathbf{A})}$. Finally, the $nk^{\omega-1}$ is dominated by the $n\sqrt{d_s(\mathbf{A})}k^{3/2}$ term so we drop it.

Finally, for dense $\mathbf{A}$ we apply the third runtime which gives

$$\tilde{O}\left(n^{\omega(\log_n k)} + \frac{n^2\sqrt{\bar{\kappa}}}{\epsilon^5 p}\right) = \tilde{O}\left(n^{\omega(\log_n k)} + \frac{n^{3/2+1/p-(\log_n k)(1/p-1/2)}}{\epsilon^{5+1/p} \cdot p}\right).$$

We now balance the terms, again ignoring ϵ dependence. Writing $\gamma = \log_n k$, $\omega(\gamma) = 2$ for $\gamma < \alpha$ where $\alpha > 0.31389$ and $2 + (\omega - 2)\frac{\gamma-\alpha}{1-\alpha}$ for $\gamma \geq \alpha$ [21]. Assuming $\gamma > \alpha$ we set: $2+(\omega-2)\frac{\gamma-\alpha}{1-\alpha} = \frac{3}{2} + \frac{1}{p} - \frac{\gamma}{p} + \frac{\gamma}{2}$ which gives $\gamma \approx \frac{1/p-.3294}{1/p+.0435} > \alpha$ for all $p < 2$ (so our assumption that $\gamma \geq \alpha$ was valid.) This yields total runtime $\tilde{O}\left(n^{\frac{2.3729-.0994p}{1+.0435p}}\right)$. Without using fast matrix multiplication, the first term in the runtime becomes n^2k and so we balance costs by setting: $n^{2+\gamma} = n^{3/2+1/p-\gamma/p+\gamma/2}$ which gives $\gamma = \frac{1/p-1/2}{1/p+1/2}$ and total runtime $\tilde{O}\left(n^{\frac{3+p/2}{1+p/2}}\right)$. ◀

References

1. Zeyuan Allen-Zhu and Yuanzhi Li. Faster principal component regression and stable matrix Chebyshev approximation. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.
2. Orly Alter, Patrick O Brown, and David Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97(18):10101–10106, 2000.
3. Alexandr Andoni, Robert Krauthgamer, and Ilya P. Razenshteyn. Sketching and embedding are equivalent for norms. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 479–488, 2015.
4. Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM*, 58(2):8, 2011.
5. Monami Banerjee and Nikhil R Pal. Feature selection with SVD entropy: some modification and extension. *Information Sciences*, 264:118–134, 2014.
6. Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317–330, 1983.
7. Constantine Bekas, Alessandro Curioni, and I Fedulova. Low cost high performance uncertainty quantification. In *Proceedings of the 2nd Workshop on High Performance Computational Finance*, page 8. ACM, 2009.

8 Christos Boutsidis, Petros Drineas, Prabhanjan Kambadur, and Anastasios Zouzias. A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *Linear Algebra and its Applications*, 533:95–119, 2017.

9 Vladimir Braverman, Stephen R Chestnut, Robert Krauthgamer, and Lin F Yang. Sketches for matrix norms: Faster, smaller and more general. *arXiv:1609.05885*, 2016.

10 Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

11 Petre Caraiani. The predictive power of singular value decomposition entropy for stock market dynamics. *Physica A: Statistical Mechanics and its Applications*, 393:571–578, 2014.

12 Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 6th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 181–190, 2015.

13 Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 209–216, 2007.

14 James Demmel. An arithmetic complexity lower bound for computing rational functions, with applications to linear algebra. *submitted to SIMAX*, 2013.

15 Amit Deshpande, Madhur Tulsiani, and Nisheeth K. Vishnoi. Algorithms and hardness for subspace approximation. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 482–496, 2011.

16 Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 2016.

17 Haishun Du, Qingpu Hu, Manman Jiang, and Fan Zhang. Two-dimensional principal component analysis based on Schatten p-norm for image feature extraction. *Journal of Visual Communication and Image Representation*, 32:55–62, 2015.

18 JK Fitzsimons, MA Osborne, SJ Roberts, and JF Fitzsimons. Improved stochastic trace estimation using mutually unbiased bases. *arXiv:1608.00117*, 2016.

19 Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

20 Roy Frostig, Cameron Musco, Christopher Musco, and Aaron Sidford. Principal component projection without principal component analysis. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2349–2357. JMLR.org, 2016. URL: `http://jmlr.org/proceedings/papers/v48/frostig16.html`.

21 François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. *arXiv:1708.05622*, 2017.

22 Gene H. Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

23 Alon Gonen, Francesco Orabona, and Shai Shalev-Shwartz. Solving ridge regression using sketched preconditioned SVRG. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

24 Rongbao Gu, Wei Xiong, and Xinjie Li. Does the singular value decomposition entropy have predictive power for stock market? evidence from the Shenzhen stock market. *Physica A: Statistical Mechanics and its Applications*, 439:103–113, 2015.

25 Ivan Gutman. Total π-electron energy of benzenoid hydrocarbons. In *Advances in the Theory of Benzenoid Hydrocarbons II*, pages 29–63. Springer, 1992.

26 Ivan Gutman. The energy of a graph: old and new results. In *Algebraic Combinatorics and Applications*, pages 196–211. Springer, 2001.

27 Insu Han, Dmitry Malioutov, Haim Avron, and Jinwoo Shin. Approximating the spectral sums of large-scale matrices using Chebyshev approximations. *SIAM Journal on Scientific Computing*, 39(4), 2017.

28 Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 2348–2356, 2012.

29 Nicholas J Higham. *Functions of matrices: theory and computation*. SIAM, 2008.

30 Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

31 Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 665–674, 2013.

32 Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 315–323, 2013.

33 Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

34 Ashish Khetan and Sewoong Oh. Matrix norm estimation from a few entries. In *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017.

35 V. V. Klyuev and N. I. Kokovkin-Shcherbak. Minimization of the number of arithmetic operations in the solution of linear algebra systems of equations. *USSR Computational Mathematics and Mathematical Physics*, 5(1):25–43, 1965.

36 N. I. Kokovkin-Shcherbak. Minimization of numerical algorithms for solving arbitrary systems of linear equations. *Ukrainskii Matematicheskii Zhurnal*, 22(4):494–502, 1970.

37 Weihao Kong and Gregory Valiant. Spectrum estimation from samples. *Annals of Statistics*, 2016.

38 J Kuczyński and H Woźniakowski. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1094–1122, 1992.

39 François Le Gall. Faster algorithms for rectangular matrix multiplication. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2012.

40 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2014.

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

42 Chao Li and Gerome Miklau. Measuring the achievable error of query sets under differential privacy. *arXiv:1202.3399*, 2012.

43 Mu Li, Gary L. Miller, and Richard Peng. Iterative row sampling. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

44 Yi Li, Huy L. Nguyen, and David P. Woodruff. On sketching matrix norms and the top singular vector. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1562–1581, 2014.

45 Yi Li and David P. Woodruff. On approximating functions of the singular values in a stream. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, 2016.

46 Yi Li and David P. Woodruff. Embeddings of schatten norms with applications to data streams. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 60:1–60:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.60.

47 Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM Review*, 58(1):34–65, 2016.

48 Yu Lu and Sahand N Negahban. Individualized rank aggregation using nuclear norm regularization. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing*, pages 1473–1479. IEEE, 2015.

49 Lei Luo, Jian Yang, Jinhui Chen, and Yicheng Gao. Schatten p-norm based matrix regression model for image classification. In *Pattern Recognition*. Springer, 2014.

50 Aleksander Madry, Damian Straszak, and Jakub Tarnawski. Fast generation of random spanning trees and the effective resistance metric. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2019–2036, 2015.

51 Jacques Morgenstern. Note on a lower bound on the linear complexity of the fast Fourier transform. *Journal of the ACM (JACM)*, 20(2):305–306, 1973.

52 Cameron Musco and Christopher Musco. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 1396–1404, 2015.

53 Cameron Musco, Praneeth Netrapalli, Aaron Sidford, Shashanka Ubaru, and David P Woodruff. Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness. *arXiv:1704.04163*, 2017.

54 Praneeth Netrapalli, UN Niranjan, Sujay Sanghavi, Animashree Anandkumar, and Prateek Jain. Non-convex robust PCA. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 1107–1115, 2014.

55 Feiping Nie, Heng Huang, and Chris Ding. Low-rank matrix recovery via efficient Schatten p-norm minimization. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

56 Beresford N Parlett. *The symmetric eigenvalue problem*. SIAM, 1998.

57 Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

58 Ran Raz and Amir Shpilka. Lower bounds for matrix product in bounded depth circuits with arbitrary gates. *SIAM Journal on Computing*, 32(2):488–513, 2003.

59 Farbod Roosta-Khorasani and Uri Ascher. Improved bounds on sample size for implicit matrix trace estimators. *Foundations of Computational Mathematics*, 2015.

60 Yousef Saad. *Numerical Methods for Large Eigenvalue Problems: Revised Edition*, volume 66. SIAM, 2011.

61 Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 64–72, 2014.

62 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, 2008.

63 Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2004.

64 Andreas Stathopoulos, Jesse Laeuchli, and Kostas Orginos. Hierarchical probing for estimating the trace of the matrix inverse on toroidal lattices. *SIAM Journal on Scientific Computing*, 35(5):S299–S322, 2013.

65 Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.

66 Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of $tr(f(a))$ via stochastic Lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, 2017.

67 Shashanka Ubaru and Yousef Saad. Fast methods for estimating the numerical rank of large matrices. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 468–477, 2016.

68 Shashanka Ubaru, Yousef Saad, and Abd-Krim Seghouane. Fast estimation of approximate matrix ranks using spectral densities. *Neural Computation*, 2017.

69 Roy Varshavsky, Assaf Gottlieb, Michal Linial, and David Horn. Novel unsupervised feature filtering of biological data. *Bioinformatics*, 22(14):e507–e513, 2006.

70 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, 2012.

71 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, pages 17–29, 2015.

72 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 645–654, 2010.

73 Shmuel Winograd. *Arithmetic Complexity of Computations*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, 1987. doi:10.1137/1.9781611970364.

74 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014.

75 Y. Xie, Y. Qu, D. Tao, W. Wu, Q. Yuan, and W. Zhang. Hyperspectral image restoration via iteratively regularized weighted Schatten p-norm minimization. *IEEE Transactions on Geoscience and Remote Sensing*, PP(99):1–18, 2016.

76 Yuan Xie, Shuhang Gu, Yan Liu, Wangmeng Zuo, Wensheng Zhang, and Lei Zhang. Weighted Schatten p-norm minimization for image denoising and background subtraction. *IEEE transactions on Image Processing*, 25(10):4842–4857, 2016.

77 Yuchen Zhang, Martin J Wainwright, and Michael I Jordan. Distributed estimation of generalized matrix rank: Efficient algorithms and lower bounds. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.