Nearly Instance Optimal Sparse Matrix Approximation from Matrix-Vector Products

Christopher Musco
New York University, NY, USA

Indu Ramesh
New York University, NY, USA

Abstract

A large body of work studies the problem of learning an approximation to an implicit matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ that is only accessible implicitly via matrix-vector product queries (matvec queries) of the form $\mathbf{x} \to \mathbf{A}\mathbf{x}$ or $\mathbf{x} \to \mathbf{A}^T\mathbf{x}$. Of particular interest are methods that learn a near-optimal approximation with a *fixed sparsity pattern*. For example, we might want to learn a near-optimal diagonal, banded, or arrow-head approximation to an implicit matrix $\mathbf{A}$.

Naturally, the number of matvec queries required to solve this problem depends on the sparsity pattern, which can be encoded as a binary matrix $\mathbf{S} \in \{0,1\}^{m \times n}$. The query complexity of previous algorithms scales with quantities like the total number of ones in $\mathbf{S}$, its maximum column/row sparsity, or the chromatic number of its "conflict graph". These quantities are incomparable: for a given $\mathbf{S}$, parameterizing by one might yield lower query complexity than another.

In this work, we unify and tighten these prior results by providing a nearly sharp characterization of the matvec query complexity of sparse matrix approximation. Generalizing a definition from graph algorithms, let the *degeneracy*, $\text{degen}(\mathbf{S})$, denote the smallest number k so that, if we iteratively delete all rows and columns of $\mathbf{S}$ with $\leq k$ ones, we are left with an empty matrix. We show that a near-optimal approximation to $\mathbf{A}$ with sparsity pattern $\mathbf{S}$ can be learned with $\tilde{O}(\text{degen}(\mathbf{S}))$ matrix-vector product queries, and $\Omega(\text{degen}(\mathbf{S}))$ queries are necessary, for *any sparsity pattern* $\mathbf{S}$. Moreover, unlike prior work based on graph coloring, all of our methods run in polynomial time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Matrix learning, sparse approximation, implicit methods

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.85

Funding *Christopher Musco*: partially supported by NSF Award #2427363.

Acknowledgements We would like to thank Cameron Musco for helpful discussions.

1 Introduction

Matrix approximation based on matrix-vector products has emerged as a fundamental primitive in numerical linear algebra and optimization [1, 3, 4, 13, 16, 30, 31, 32, 34, 35, 37, 38, 39, 47, 50]. Recently, the problem has also found applications in scientific machine learning (SciML) as a powerful yet tractable special case of *operator learning* [8, 9, 10, 22, 51].

In the basic setup, we are given access to black-box matrix vector products (abbreviated "matvecs") with $\mathbf{A}$ and $\mathbf{A}^T$ for some unknown matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. I.e., we can evaluate $\mathbf{A}\mathbf{x}_1, \mathbf{A}^T\mathbf{y}_1, \ldots, \mathbf{A}\mathbf{x}_q, \mathbf{A}^T\mathbf{y}_q$ for adaptively chosen vectors $\mathbf{x}_1, \ldots, \mathbf{x}_q \in \mathbb{R}^n$ and $\mathbf{y}_1, \ldots, \mathbf{y}_q \in \mathbb{R}^m$. "Adaptively" means that the choice of $\mathbf{x}_i, \mathbf{y}_i$ can depend on all previous queries and their results. Given the results of all queries, the goal is to find a near-optimal approximation to $\mathbf{A}$ from some pre-specified family $\mathcal{F} \subset \mathbb{R}^{m \times n}$. I.e., find $\tilde{\mathbf{B}} \in \mathcal{F}$ satisfying[1]:

$$\|\mathbf{A} - \tilde{\mathbf{B}}\|_F \leq (1+\epsilon) \cdot \min_{\mathbf{B} \in \mathcal{F}} \|\mathbf{A} - \mathbf{B}\|_F. \tag{1}$$

[1] Matrix approximation is most commonly studied with error measured by the Frobenius norm, $\|\mathbf{A}-\mathbf{B}\|_F$, and this is the norm we consider. However, it is natural to study other norms, like the spectral norm [45, 52]. Additionally, the goal of $(1+\epsilon)$-near optimality can often be relaxed: the problem is still interesting when the approximation factor is a constant or grows mildly with $\mathbf{A}$'s dimensions [3].

Natural examples for the approximation family, $\mathcal{F}$, include low-rank matrices, hierarchically structured matrices, butterfly matrices, and more [37, 39, 30]. As one example result, when $\mathcal{F}$ is the class of rank-k matrices, (1) can be achieved with $O(k/\epsilon^{1/3})$ matvecs [4].

In applications, we are interested in algorithms based on matvec queries because it is often far more efficient to compute matvecs than to produce $\mathbf{A}$ explicitly. Common settings include when $\mathbf{A}$ is the Hessian matrix of a high-dimensional optimization problem, with which matvecs can be computed efficiently via autodiff [48], or when $\mathbf{A}$ is an integral transform that can be applied efficiently using the Fourier transform or Fast Multipole Method [34]. Another example arises when $\mathbf{A}$ is the inverse of a matrix $\mathbf{B}$. While producing $\mathbf{A}$ requires matrix inversion, and thus $O(n^3)$ time, matvecs can often be computed by solving a linear system involving $\mathbf{B}$, which takes time $O(n^2)$ or less when $\mathbf{A}$ is well-conditioned [6],

1.1 Sparse Matrix Approximation

An important special case of the matrix approximation problem is when $\mathcal{F}$ contains all matrices with some *fixed sparsity pattern*.[2] One example is the class of diagonal matrices [6, 5, 24]. Diagonal matrices are widely used to approximate Hessian matrices in optimization [7, 21, 42, 57] or inverse matrices in statistical applications [28, 54]. Other fixed-sparsity families of interest include block-diagonal matrices, banded matrices, and matrices whose non-zeroes correspond to a known graph (e.g., a low-dimensional mesh) [18, 30, 54, 55].

Any such family can be encoded by a binary matrix $\mathbf{S} \in \{0,1\}^{m\times n}$ representing the sparsity pattern, where a one at position (i,j) in $\mathbf{S}$ indicates that entry (i,j) is allowed to be non-zero for $\mathbf{B} \in \mathcal{F}$. The optimal Frobenius norm approximation to $\mathbf{A}$ with sparsity pattern $\mathbf{S}$ is easy to write down: it is $\mathbf{S} \circ \mathbf{A}$, where $\circ$ denotes the entrywise product. However, it is not typically possible to compute $\mathbf{S} \circ \mathbf{A}$ using a small number of matrix-vector products. In particular, while we can read any single column or row of $\mathbf{A}$ using one matrix-vector product (with a standard basis vector), $\mathbf{S}$'s ones could be distributed among many columns/rows.

As such, prior work has focused on *approximation algorithms*. For the special case of diagonal approximation, it is known that the popular Hutchinson's estimator finds a near-optimal diagonal $\tilde{\mathbf{B}}$ satisfying (1) with $O(1/\epsilon)$ matvec queries, and no algorithm achieving the same approximation guarantee can use asymptotically fewer queries [6, 24]. Optimal query complexities are also known, e.g., for banded and block diagonal matrices [2].

For more complex sparse families, however, the optimal query complexity is often unknown. A number of prior results provide generic results that give query complexities depending on certain properties of $\mathbf{S}$. For example, if $\mathbf{S}$ has at most s ones in any row (or column) then [2] shows that a $(1+\epsilon)$ error approximation can be learned using $O(s/\epsilon)$ products with random Gaussian vectors. It can also be shown that, if $\mathbf{S}$ has at most $\text{nnz}(\mathbf{S})$ total ones, then $O(\sqrt{\text{nnz}(\mathbf{S})/\epsilon})$ matvecs suffice, no matter how the ones are distributed [1, 20].

Another line of work studies algorithms based on graph coloring [17, 18, 19, 29, 41, 49, 54]. Such methods are typically only analyzed in the easier *recovery* setting, where $\mathbf{A}$ itself is assumed to be sparse. I.e., $\mathbf{S} \circ \mathbf{A} = \mathbf{A}$, and thus the optimal solution to (1) has 0 error. In this setting, the complexity of these methods depends on the chromatic number (or related properties) of various graphs related to $\mathbf{S}$. For example, [19] observes that, if we can partition the columns of $\mathbf{S}$ into k subsets, each consisting of columns with disjoint support, then

[2] This problem should not be confused with the closely related but distinct *sparse recovery problem*, where we wish to find a good sparse approximation to $\mathbf{A}$, and are free to use any sparsity pattern we wish [20]. See [2] for further discussion on differences between these problems.

any matrix with sparsity pattern $\mathbf{S}$ can be recovered with k matvecs. The smallest such k corresponds to the chromatic number of a "conflict" graph, G, associated with $\mathbf{S}$: G has a vertex for each column and an edge between any two columns with overlapping support.

Finding an optimal coloring of G (to come up with a suitable set of query vectors) is NP-hard, so heuristic approximation methods are typically used instead. NP-hard optimization problems also arise in the implementation of more complex coloring-based algorithms with refined query complexities, which have been studied in work by Mccormick [41], Power and Toint [49], Coleman and Cai [17], and others.

1.2 Our Contributions

Our paper simplifies and generalizes this prior work by identifying a single combinatorial parameter of the sparsity pattern, $\mathbf{S}$, that, up to logarithmic factors, completely characterizes the matvec complexity of learning a near-optimal approximation with sparsity pattern $\mathbf{S}$. Specifically, generalizing a notion from graph theory, we define:

▶ **Definition 1** (Degeneracy of a Sparsity Pattern). *Consider a matrix $\mathbf{S} \in \{0,1\}^{n\times m}$. For a positive integer k, delete all columns and rows of $\mathbf{S}$ with $\leq k$ ones, and then repeat this process on the smaller matrix that remains until no further columns or rows can be deleted. If we are left with an empty matrix, then we say $\mathbf{S}$ is k-degenerate. The* degeneracy *of $\mathbf{S}$, denoted by* $\text{degen}(\mathbf{S})$*, is the smallest value of k such that $\mathbf{S}$ is k-degenerate.*

If $\mathbf{S}$ was the symmetric adjacency matrix of an undirected graph, G, then $\text{degen}(\mathbf{S})$ would exactly correspond to the degeneracy of G, which is also known as the k-core number, width, or graph linkage [33, 36].[3] Our main algorithmic result is that $\text{degen}(\mathbf{S})$ bounds the complexity of approximating an unknown matrix $\mathbf{A}$ with sparsity pattern $\mathbf{S}$:

▶ **Theorem 2** (Upper Bound). *Let $\mathbf{S} \in \{0,1\}^{n\times m}$ be a known sparsity pattern and $\mathbf{A}$ be an unknown matrix. There is a polynomial time algorithm that issues $O\left(\frac{\text{degen}(\mathbf{S})}{\epsilon\delta}\cdot\log(n+m)\right)$ non-adaptive matrix-vector product queries with $\mathbf{A}$ and $\mathbf{A}^T$ and, with probability $1-\delta$, returns a matrix $\tilde{\mathbf{B}}$ with sparsity pattern $\mathbf{S}$ satisfying:*

$$\|\mathbf{A}-\tilde{\mathbf{B}}\|_F \leq (1+\epsilon)\cdot\min_{\mathbf{B}:\mathbf{B}=\mathbf{S}\circ\mathbf{B}}\|\mathbf{A}-\mathbf{B}\|_F = (1+\epsilon)\cdot\|\mathbf{A}-\mathbf{S}\circ\mathbf{A}\|_F.$$

Notably, unlike coloring methods, the set of queries issued by our algorithm can be computed in time polynomial in n and m, as can the approximation $\tilde{\mathbf{B}}$. We also note that, in the easier "recovery" setting, where $\mathbf{S}\circ\mathbf{A}=\mathbf{A}$, our algorithm succeeds with exactly $2\cdot\text{degen}(\mathbf{S})$ queries, and those queries can be chosen to simply be random Gaussian vectors.

We highlight some sparsity patterns in Table 1 to compare the upper bound of Theorem 2 to prior results. As one example, consider the "arrowhead" sparsity pattern for an $n\times n$ matrix, which has ones on the diagonal and on the first row and column of $\mathbf{S}$. The degeneracy of this pattern is 1, so our algorithm can solve the recovery problem using just 2 matvecs and the approximation problem with $O(\log n/\epsilon)$. On the other hand, prior measures of the arrowhead pattern's complexity are much higher: it has maximum row sparsity n, $3n-3$ total non-zeros, and its conflict graph chromatic number n. Thus, even the best prior results (in this case based on total non-zeros) imply query complexity no better than $O(\sqrt{n})$.

[3] The degeneracy is also equivalent to G's "coloring number" minus 1 (not to be confused with chromatic number) [26, 27] and is within a 2 factor of G's arboricity.

Table 1 We give examples of natural sparse matrix families, along with various results on the query complexity of recovering matrices from these families. "s-modular" is a sparse family introduced in Section 4.3 of [2]. As we can see, our results improve on all prior work, and our Theorem 3 confirms that no further improvement is possible, for any family. Even for the harder approximation problem, any improvements are necessarily limited to the logarithmic factor and dependence on ϵ.

	Sparse Family		
Query Complexity Upper Bound	s-banded	s-modular	arrowhead
Maximum Row Sparsity [2]	$O(s)$	$O(s)$	n
$\sqrt{\text{Total Sparsity}}$ [1]	$O(\sqrt{ns})$	$O(\sqrt{ns})$	$O(\sqrt{n})$
Conflict Graph Chromatic Number [19]	$O(s)$	$O(s^2)$	n
Degeneracy (this work)	$O(s)$	$O(s)$	$O(1)$

We complement Theorem 2 with a nearly matching lower bound for *any sparsity pattern* that holds even for adaptive methods (Theorem 2 uses non-adaptive queries[4]):

▶ **Theorem 3** (Lower Bound). *For any sparsity pattern* $\mathbf{S} \in \{0,1\}^{m\times n}$*,* $\delta < 1$ *and any finite* C*, no algorithm which queries less than* $\text{degen}(\mathbf{S})$ *matvecs with* $\mathbf{A}$ *or* $\mathbf{A}^T$ *can return, with probability* $1-\delta$*, a matrix* $\tilde{\mathbf{B}}$ *satisfying* $\|\mathbf{A}-\tilde{\mathbf{B}}\|_F \le C \cdot \min_{\mathbf{B}:\mathbf{B}=\mathbf{S}\circ\mathbf{B}} \|\mathbf{A}-\mathbf{B}\|_F$*.*

The proof of Theorem 3 is simple, following from the fact that any sparsity pattern $\mathbf{S}$ with degeneracy $\text{degen}(\mathbf{S})$ must contain a submatrix with at least $\text{degen}(\mathbf{S})$ ones in every row and column. It can be shown that, in the recovery setting where $\mathbf{A} = \mathbf{S}\circ\mathbf{A}$, recovering even this submatrix in $\mathbf{A}$ requires $\ge \text{degen}(\mathbf{S})$ matrix-vector product queries. In the recovery setting, $\min_{\mathbf{B}:\mathbf{B}=\mathbf{S}\circ\mathbf{B}} \|\mathbf{A}-\mathbf{B}\|_F = 0$, so a lower bound for recovery implies a lower bound for approximation for any finite C. This proof is detailed further in Section 3.

▶ Remark 4. As mentioned, in the recovery setting, we can learn any $\mathbf{A}$ with sparsity pattern $\mathbf{S}$ with *exactly* $2\cdot\text{degen}(\mathbf{S})$ matvecs (see Lemma 10). Theorem 3 thus implies that our recovery algorithm is essentially "instance optimal": no method can improve on its complexity by more than a factor of 2 for *any sparsity pattern*. At the same time, it avoids the computational intractability of, e.g., solving a graph coloring problem.

▶ Remark 5. For the approximation setting, there is a gap of $O\left(\log(m+n)/\epsilon\right)$ between the upper and lower bounds of Theorem 2 and Theorem 3. We believe the $\log(m+n)$ term can possibly be eliminated from the upper bound. However, the $O(1/\epsilon)$ term is inherent. There are sparsity patterns for which $\Omega(\text{degen}(\mathbf{S})/\epsilon)$ matvecs are necessary to learn a $(1+\epsilon)$-approximation. E.g., see [2] for a lower bound of $\Omega(s/\epsilon)$ for s-banded matrices. On the other hand, there are sparsity patterns where $O(\text{degen}(\mathbf{S}))$ matvecs suffice. A simple example is when $\mathbf{S}$ has only a single non-zero column, i. Such a pattern had degeneracy 1 and an exactly optimal approximation with the sparsity pattern (i.e., with $\epsilon = 0$) can be learned with one matvec: simply read $\mathbf{A}$'s i^{th} column by multiplying with a standard basis vector.

1.3 Paper Structure

We briefly review definitions and notation used throughout the paper in Section 2. In Section 3, we prove an upper bound of $\text{degen}(\mathbf{S})$ matvecs and matching lower bound for recovering a matrix $\mathbf{A}$ with sparsity pattern $\mathbf{S}$. As discussed, this lower bound immediately implies Theorem 3. In Section 4, we conclude by proving our main upper bound, Theorem 2.

[4] The specific queries chosen depend on $\mathbf{S}$, but can all be selected in advance, before issuing any queries.

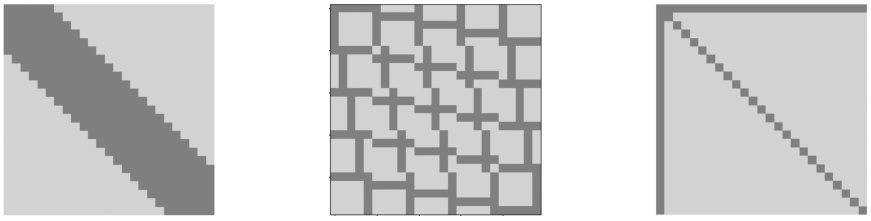

Figure 1 Illustrations of the s-banded, s-modular, and arrowhead sparsity patterns from Table 1 for 25x25 matrices with $s = 5$. Dark gray represents non-zeros, and light gray represents zeros.

2 Preliminaries

2.1 Notation

Vectors, Matrices, and Norms. We use bold upper case letters to denote matrices and bold lower case letters to denote vectors. For a vector $\mathbf{x}$, x_i denotes the i^{th} entry. For a matrix $\mathbf{M}$, $M_{i,j}$ denotes the i,j entry. $\text{nnz}(\mathbf{M})$ denote the number of non-zero entries in $\mathbf{M}$. For matrices $\mathbf{A}$ and $\mathbf{B}$ of like dimensions, $\mathbf{A} \circ \mathbf{B}$ is the entrywise product. I.e., if $\mathbf{C} = \mathbf{A} \circ \mathbf{B}$, then $C_{i,j} = A_{i,j} \cdot B_{i,j}$. For a length n vector, $\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2}$ denotes the Euclidean norm and, for an $n \times m$ matrix, $\|\mathbf{M}\|_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{m} M_{i,j}^2\right)^{1/2}$ denotes the Frobenius norm. We let $\mathbf{M}^+ \in \mathbb{R}^{m\times n}$ denote the Moore–Penrose pseudoinverse of a matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$. When $\mathbf{M}$ has full column rank (as will always be the case in this paper), $\mathbf{M}^+ = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$ and we have that $\mathbf{M}^+\mathbf{y} = \text{argmin}_{\mathbf{x}\in\mathbb{R}^m} \|\mathbf{M}\mathbf{x} - \mathbf{y}\|_2$ for any $\mathbf{y} \in \mathbb{R}^n$. We let $\mathbf{0}$ denote the all zeros matrix. The dimension will be clear from context.

Slice Notation. We will also use matrix "slice notation" following standard conventions. For example, $\mathbf{M}_{i,:}$ denotes the i^{th} row of $\mathbf{M}$ and $\mathbf{M}_{:,j}$ denotes the j^{th} column. We always treat these vectors as column vectors (i.e., with width 1). Suppose $\mathbf{M}$ has n rows and m columns and $\mathcal{S}$ and $\mathcal{R}$ are, respectively, ordered subsets of $\{1,\ldots,n\}$ and $\{1,\ldots,m\}$. Then $\mathbf{M}_{\mathcal{S},\mathcal{R}}$ denotes the $|\mathcal{S}| \times |\mathcal{R}|$ matrix consisting of all entries $M_{i,j}$ for which $i \in \mathcal{S}, j \in \mathcal{R}$. Equivalently, let $\mathbf{s} \in \{0,1\}^n$ be a set indicator vector with $s_i = 1$ for $i \in \mathcal{S}$, $s_i = 0$ for $i \notin \mathcal{S}$, and let $\mathbf{r}$ be defined analogously for $\mathcal{R}$. Then we will also allow for $\mathbf{M}_{\mathcal{S},\mathcal{R}}$ to be written as $\mathbf{M}_{\mathbf{s},\mathbf{r}}$. $\mathbf{M}_{\mathcal{S},:} = \mathbf{M}_{\mathbf{s},:}$ denotes the matrix obtained by setting $\mathcal{R} = \{1,\ldots,m\}$ to be the complete set. For a binary vector, we overload notation and let $|\mathbf{s}|$ denote $|\mathcal{S}| = \text{nnz}(\mathbf{s})$.

Index Restriction and Extension. Let $\mathbf{I}$ denote an $n \times n$ identity matrix and consider a set $\mathcal{S} \subseteq \{1,\ldots,n\}$, or equivalently, an indicator vector $\mathbf{s} \in \{0,1\}^n$ with $\mathbf{s}_i = 1$ for $i \in \mathcal{S}$. We will let $\mathbf{I}_{\mathcal{S}}$, or equivalently $\mathbf{I}_{\mathbf{s}}$, denote $\mathbf{I}_{\mathcal{S}} = \mathbf{I}_{\mathcal{S},:}$. Observe that, for any vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{I}_{\mathcal{S}}\mathbf{x} = \mathbf{I}_{\mathbf{s}}\mathbf{x}$ performs a *restriction operation*, returning a length $|\mathcal{S}|$ vector whose entries correspond to x_i for $i \in \mathcal{S}$. Similarly, for any length $|\mathcal{S}|$ vector $\mathbf{y}$, $\mathbf{I}_{\mathcal{S}}^T\mathbf{y} = \mathbf{I}_{\mathbf{s}}^T\mathbf{y}$ performs an *extension operation*, returning a length n vector $\mathbf{x}$ with $\mathbf{x}_{s_i} = y_i$ for $\{s_1,\ldots,s_{|\mathcal{S}|}\} = \mathcal{S}$.

2.2 Problem Formulation

We study algorithms that can only access an unknown target matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ through left and right matrix-vector products (i.e., "matvec queries"). We formalize this model as follows:

▶ **Definition 6** (Matrix-Vector Product Query Model)**.** *An algorithm in this model can issue queries of the form* $(\mathbf{x}, s)$*, where* $\mathbf{x}$ *is a vector and* $s \in \{L, R\}$*. If* $s = R$*, the algorithm receives query response* $\mathbf{Ax}$ *for a fixed, unknown matrix* $\mathbf{A} \in \mathbb{R}^{n\times m}$*. Otherwise, if* $s = L$*, the algorithm receives* $\mathbf{A}^T\mathbf{x}$*. A* q*-query algorithm is one that issues* q *queries,* $(\mathbf{x}_1, s_1), \ldots, (\mathbf{x}_q, s_q)$*, where the choice of* $(\mathbf{x}_i, s_i)$ *can depend on the outcome of all previous* $i - 1$ *queries.*

In the numerical analysis literature, methods implemented in the matvec query model are often referred to as "matrix-free" or "implicit" methods. In this work we consider solving two closely related problems in the matrix-vector product query model:

▶ **Problem 7** (Sparse Matrix Recovery)**.** *Given a sparsity pattern* $\mathbf{S} \in \{0, 1\}^{n\times m}$ *and query access to a matrix* $\mathbf{A}$ *where* $\mathbf{A} \circ \mathbf{S} = \mathbf{A}$ *(i.e.,* $\mathbf{A}$ *has sparsity pattern* $\mathbf{S}$*), recover* $\mathbf{A}$*.*

▶ **Problem 8** (Sparse Matrix Approximation)**.** *Given a sparsity pattern* $\mathbf{S} \in \{0, 1\}^{n\times m}$ *and query access to an arbitrary matrix* $\mathbf{A}$*, find* $\tilde{\mathbf{B}}$ *such that* $\tilde{\mathbf{B}} \circ \mathbf{S} = \tilde{\mathbf{B}}$ *that satisfies:*

$$\|\mathbf{A} - \tilde{\mathbf{B}}\|_F \leq (1 + \epsilon) \min_{\mathbf{B}:\mathbf{B}\circ\mathbf{S}=\mathbf{B}} \|\mathbf{A} - \mathbf{B}\|_F.$$

Note that we always have $\min_{\mathbf{B}:\mathbf{B}\circ\mathbf{S}=\mathbf{B}} \|\mathbf{A} - \mathbf{B}\|_F = \|\mathbf{A} - \mathbf{A} \circ \mathbf{S}\|_F$*.*

Observe that Problem 8 is strictly harder than Problem 7: if we have an algorithm that solves Problem 8 for any finite ϵ, then when run on any matrix $\mathbf{A}$ with sparsity pattern $\mathbf{S}$, it must return $\mathbf{A}$ exactly to achieve multiplicative $(1 + \epsilon)$ error (since $\min_{\mathbf{B}:\mathbf{B}\circ\mathbf{S}=\mathbf{B}} \|\mathbf{A} - \mathbf{B}\|_F = 0$).

When evaluating algorithms for these problems in the matvec query model, we are primarily interested in minimizing matvec query complexity. This is motivated by the fact that, in many applications, computing $\mathbf{Ax}$ or $\mathbf{A}^T\mathbf{x}$ is a computational bottleneck, so query complexity closely correlates with computational cost. Indeed, there has been significant recent interest in proving tight upper and lower bounds on the matvec query complexity of natural linear algebraic problems like trace estimation, linear system solving, eigenvalue computation, and more [11, 12, 14, 15, 23, 43, 44, 46, 52, 53, 56].

That said, other facets of an algorithm implemented in the matvec query model are also of interest. For example, ideally we should be able to efficiently compute the query vectors, $\mathbf{x}_1, \ldots, \mathbf{x}_q$, and we should also be able to efficiently construct $\tilde{\mathbf{B}}$ from the obtained responses. All of the algorithms in this paper run in time polynomial in the matrix dimensions n and m.

There is also interest in algorithm that require limited "adaptivity", meaning that the queries can be chosen all at once ("non-adaptively") or in a small number of rounds. Low adaptivity algorithms allow for increase parallelization query computation, which can lead to speedups in practice. All algorithms considered in this paper are fully non-adaptive.

3 Lower and Upper Bound for Sparse Matrix Recovery

We begin by observing that, for any sparsity pattern $\mathbf{S}$, $\text{degen}(\mathbf{S})$ lower bounds the matvec complexity of recovering a matrix $\mathbf{A}$ with sparsity pattern $\mathbf{S}$. As discussed in Section 1.2, our main lower bound in Theorem 3 follows as an immediate corollary.

▶ **Lemma 9.** *For any sparsity pattern* $\mathbf{S} \in \{0, 1\}^{n\times m}$*, no algorithm can solve Problem 7 with any positive success probability using* $< \text{degen}(\mathbf{S})$ *matvec queries.*

Proof. We first observe that, $\mathbf{S}$ must contain a submatrix with at least $\text{degen}(\mathbf{S})$ ones in every row and column. This is because, if all submatrices had a row or column with $\leq \text{degen}(\mathbf{S})$ ones, then by definition, the matrix must have degeneracy $< \text{degen}(\mathbf{S})$.

Let $\bar{\mathbf{S}} \in \{0,1\}^{m' \times n'}$ denote this submatrix, where $m' \leq m$ and $n' \leq n$, and let $\ell' = \max(m', n')$. $\bar{\mathbf{S}}$ contains at least $\ell' \cdot \text{degen}(\mathbf{S})$ ones. Now, consider the easier problem of recovering a matrix $\mathbf{A}$ that is only non-zero on the non-zero entries of $\bar{\mathbf{S}}$. Every multiplication by $\mathbf{A}$ on the right by a vector $\mathbf{x}$ yields a vector with at most m' non-zero entries, each of with is a linear combination of a subset of entries in $\mathbf{A}$ (those contained in a single row). Likewise, every multiplication by $\mathbf{A}$ on the left by a vector $\mathbf{x}$ yields a vector with at most n' non-zero entries, each of which is a linear combination of a subset of entries in $\mathbf{A}$.

Accordingly, no matter how our query vectors are chosen, if we issue q matvec queries, we obtain *at most* $q \cdot \ell'$ linear equations in the unknown entries of $\mathbf{A}$. It is impossible to determine these entries unless $q \cdot \ell' \geq \text{nnz}(\mathbf{A}) \geq \ell' \cdot \text{degen}(\mathbf{S})$ (since we have an underdetermined linear system [30]). In other words, we must have $q \geq \text{degen}(\mathbf{S})$ to determine $\mathbf{A}$. ◀

With Lemma 9 in place, it is natural to ask how close we can get to the limit of $\text{degen}(\mathbf{S})$ matvec queries for solving Problem 7. Surprisingly, we show that a simple, non-adaptive, polynomial time algorithm solves the problem with $2 \cdot \text{degen}(\mathbf{S})$ matvec queries. This is despite substantial work on more complex coloring-based algorithms for Problem 7, which require solving NP-hard optimization problems to select query vectors, as discussed in Section 1.1.

▶ **Lemma 10** (Upper Bound). *For any sparsity pattern $\mathbf{S} \in \{0,1\}^{n \times m}$, Problem 7 can be solved with probability 1 using $2 \cdot \text{degen}(\mathbf{S})$ matvec queries (specifically, the product of each of $\mathbf{A}$ and $\mathbf{A}^T$ with $\text{degen}(\mathbf{S})$ i.i.d. random Gaussian vectors).*

Proof. The algorithm that achieves Lemma 10 is detailed in Algorithm 1. The main idea is simple: we will iteratively recover rows and columns of $\mathbf{A}$ with $\leq \text{degen}(\mathbf{S})$ unknowns.

Algorithm 1 Sparse Matrix Recovery.

Input: Sparsity pattern $\mathbf{S} \in \{0,1\}^{m \times n}$, matvec query access to a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$.
Output: $\hat{\mathbf{A}}$ that is equal to $\mathbf{A}$ as long as $\mathbf{A} \circ \mathbf{S} = \mathbf{A}$ (i.e., $\mathbf{A}$ has sparsity pattern $\mathbf{S}$).

Compute $d = \text{degen}(\mathbf{S})$. ▷ This can be done in $O(\text{nnz}(\mathbf{S}))$ time [40].
Construct $\mathbf{W} \in \mathbb{R}^{m \times d}$ and $\mathbf{Z} \in \mathbb{R}^{n \times d}$ with i.i.d. random Gaussian entries.
Compute $\mathbf{L} \leftarrow \mathbf{A}^T\mathbf{W}$ and $\mathbf{R} \leftarrow \mathbf{A}\mathbf{Z}$. ▷ Requires $2 \cdot \text{degen}(\mathbf{S})$ matvec queries.
Set $\mathbf{L}_{init} \leftarrow \mathbf{L}$ and $\mathbf{R}_{init} \leftarrow \mathbf{R}$. Initialize $\hat{\mathbf{A}}$ as an $m \times n$ matrix of zeros.
while $\text{nnz}(\mathbf{S}) > 0$ **do**
    Let $\mathcal{R}$ contain the indices of all rows in $\mathbf{S}$ with $\leq d$ ones.
    Let $\mathcal{C}$ contain the indices of all columns in $\mathbf{S}$ with $\leq d$ ones.
    for $r \in \mathcal{R}$ **do**
        Set $\mathbf{s} \leftarrow \mathbf{S}_{r,:}$ and set $\mathbf{G} = (\mathbf{Z}_{\mathbf{s},:})^T$.
        Set $\hat{\mathbf{A}}_{r,:} \leftarrow \mathbf{I}_{\mathbf{s}}^T\mathbf{G}^+\mathbf{R}_{r,:}$. ▷ $\mathbf{I}_{\mathbf{s}}^T$ is an $n \times |\mathbf{s}|$ extension operator (see Section 2).
        Set all entries in row r in $\mathbf{S}$ equal to 0.
    for $c \in \mathcal{C}$ **do**
        Set $\mathbf{s} \leftarrow \mathbf{S}_{:,c}$ and set $\mathbf{G} = (\mathbf{W}_{\mathbf{s},:})^T$.
        Set $\hat{\mathbf{A}}_{:,c} \leftarrow \mathbf{I}_{\mathbf{s}}^T\mathbf{G}^+\mathbf{L}_{c,:}$.
        Set all entries in column c in $\mathbf{S}$ equal to 0.
    Update $\mathbf{L} \leftarrow \mathbf{L}_{init} - \hat{\mathbf{A}}^T\mathbf{W}$ and $\mathbf{R} \leftarrow \mathbf{R}_{init} - \hat{\mathbf{A}}\mathbf{Z}$. ▷ Does not require matvec queries.
Return $\hat{\mathbf{A}}$.

Consider rows to start. Suppose we multiply $\mathbf{A}$ on the right by a random Gaussian matrix $\mathbf{Z}$ with $d = \text{degen}(\mathbf{S})$ columns (using $\text{degen}(\mathbf{S})$ matvec queries). Call the result $\mathbf{R} = \mathbf{A}\mathbf{Z}$. Now, let r be the index of any row in $\mathbf{S}$ with $\leq d$ ones. Let $\mathbf{s}$ denote this row. The r^{th} row of

$\mathbf{R}$ contains d linear combinations of the entries in row r of $\mathbf{A}$. Concretely, $(\mathbf{R}_{r,:})^T = \mathbf{A}_{r,\mathbf{s}}\mathbf{Z}_{\mathbf{s},:}$. $\mathbf{Z}_{\mathbf{s},:}$ is a Gaussian matrix with d columns and $|\mathbf{s}| \leq d$ rows, so it has full row-rank with probability 1. We can thus solve for $\mathbf{A}_{r,\mathbf{s}}$ by computing $(\mathbf{Z}^T_{\mathbf{s},:})^+\mathbf{R}_{r,:}$.[5]

Symmetrically, we can learn the entries in all columns of $\mathbf{A}$ with $\leq d$ unknowns by multiplying $\mathbf{A}^T$ by a random Gaussian matrix ($\mathbf{W}$ in Algorithm 1) with d columns.

Let R and C denote the indices of all rows and columns learned in this first round. We update $\mathbf{S}$ by zeroing out all entries in these rows and columns. Moreover, we update our current "guess" for $\mathbf{A}$, which is denoted by $\hat{\mathbf{A}}$, by setting the values in these rows and columns to those that we just learned. At this point, we can iterate on the matrix $\mathbf{A} - \hat{\mathbf{A}}$ without computing additional matrix-vector products. We simply update $\mathbf{R}$ by subtracting off $\hat{\mathbf{A}}\mathbf{Z}$, which yields $\mathbf{R} = (\mathbf{A} - \hat{\mathbf{A}})\mathbf{Z}$. Similarly, we update $\mathbf{L} = \mathbf{A}^T\mathbf{W}$ by subtracting off $\hat{\mathbf{A}}^T\mathbf{W}$ to yield $\mathbf{L} = (\mathbf{A} - \hat{\mathbf{A}})^T\mathbf{W}$. From these matvecs, we proceed to learn all rows and columns in $\mathbf{A} - \hat{\mathbf{A}}$ with $\leq d$ unknowns. We repeat the process until $\mathbf{S}$ is all zeros and $\hat{\mathbf{A}} = \mathbf{A}$.

The correctness of the algorithm rests on the fact that all subsets of $\leq d$ rows from $\mathbf{Z}$ and $\mathbf{W}$ that we consider have full row rank. This holds simply because there are only a finite number of such subsets, and each has full row-rank almost surely [25]. The runtime of the algorithm is also clearly polynomial in n and m. ◀

4 Near-Optimal Sparse Matrix Approximation

We conclude by providing an $\tilde{O}(\text{degen}(\mathbf{S})/\epsilon)$ query algorithm for the harder problem of finding a near-optimal sparse approximation to an arbitrary matrix $\mathbf{A}$, which proves Theorem 2. The basic approach is similar to the recovery algorithm from Section 3. Using standard tools from randomized numerical linear algebra (specifically, Lemma 12), it is not hard to show that multiplying $\mathbf{A}$ on the left and right by $O(\text{degen}(\mathbf{S})/\epsilon)$ random Gaussian vectors suffices to near-optimally approximate any row and column with $\leq \text{degen}(\mathbf{S})$ non-zeros.

Once we have done so, a natural approach would be to subtract off our approximation to $\mathbf{A}$ and iterate on the residual. However, this leads to dependency issues: unlike the recovery setting, where we almost surely learn the exact values of rows and columns, in the approximation setting, our residual is a random matrix that depends on our random Gaussian queries. This prevents a direct argument along the lines of Lemma 10, since the same Gaussian queries cannot easily be "reused" on the residual matrix.

We deal with this dependency by simply drawing a *fresh* set of random Gaussian vectors at every iteration of the algorithm. Naively, this could be a disastrous idea: it is possible to design sparsity patterns where the number of iterations scales with the matrix dimension. For example, let $\mathbf{S}$ be the adjacency matrix of a path graph on n nodes. This pattern has degeneracy 1, and if we iteratively peel off rows/columns with 1 non-zero, it will take $n/2$ iterations. Our sample complexity will thus scale with n, instead of the desired $O(1/\epsilon)$.

The fix is simple: it is not hard to prove that, if we peel off rows and columns with sparsity $\leq 4 \cdot \text{degen}(\mathbf{S})$ instead of $\leq \text{degen}(\mathbf{S})$, then we can bound the number of iterations by $O(\log n + m)$. The only cost is a small increase in the number of Gaussian random vectors required per iteration (which scales with the sparsity of the rows and columns learned).

▶ **Lemma 11.** *Consider a sparsity pattern $\mathbf{S} \in \{0,1\}^{m\times n}$ with degeneracy $d = \text{degen}(\mathbf{S})$. Suppose we delete all rows and columns in $\mathbf{S}$ with $\leq 4 \cdot d$ ones, and then repeat this process on the remaining matrix. After $\log_2(m+n)$ rounds we are left with an empty matrix.*

[5] Recall that we take the convention that row and column slices are column vectors. So $\mathbf{R}_{r,:} \in \mathbb{R}^{d\times 1}$.

Proof. We first observe that any sparsity pattern with degeneracy d has at most $d \cdot (m+n)$ ones. By an averaging argument, it follows that $< \frac{(m+n)}{4}$ columns have sparsity $> 4 \cdot d$. Likewise, $< \frac{(m+n)}{4}$ rows have sparsity $> 4 \cdot d$. Accordingly, if we remove all rows and columns with sparsity $\leq 4 \cdot d$, we are left with a matrix with dimensions m', n' such that:

$$m' + n' < \frac{(m+n)}{4} + \frac{(m+n)}{4} = \frac{m+n}{2}.$$

In other words, the sum of $\mathbf{S}$' dimensions is reduced by a factor of at least two at every iteration. Hence, we obtain an empty matrix after $\log_2(m+n)$ iterations. ◀

With Lemma 11 in place, we are ready to present and analyze our main algorithm. We require one additional fact on recovering a sparse approximation to a vector based on inner products with a set of Gaussian random vectors. Reminiscent of results in compressed sensing, this result follows from now standard techniques in randomized numerical linear algebra. See the analysis of Theorem 1 in [2] for a complete proof.

▶ **Lemma 12.** *Let $\mathbf{y} \in \mathbb{R}^n$ be a vector and $\mathbf{s} \in \{0,1\}^n$ be a sparsity pattern of the same length. Let $\mathbf{H} \in \mathbb{R}^{q \times n}$ be a matrix with i.i.d. standard Gaussian entries. Let:*

$$\hat{\mathbf{y}} = \mathbf{I}_{\mathbf{s}}^T \mathbf{H}_{:,\mathbf{s}}^{+} \mathbf{H}\mathbf{y}, \text{ where } \mathbf{I}_{\mathbf{s}}^T \text{ is as defined in Section 2.}$$

Then we have that:

$$\mathbb{E}\left[\|\hat{\mathbf{y}} - \mathbf{s} \circ \mathbf{y}\|_2^2\right] \leq \frac{|\mathbf{s}|}{q - |\mathbf{s}| - 1} \|\mathbf{y} - \mathbf{s} \circ \mathbf{y}\|_2^2. \tag{2}$$

▶ Remark 13. Lemma 12 bounds the difference between our sparse approximation, $\hat{\mathbf{y}}$, and the optimal approximation to $\mathbf{y}$ with sparsity pattern $\mathbf{s}$, which is $\mathbf{s} \circ \mathbf{y}$. Observe that, since $\mathbf{y} - \mathbf{s} \circ \mathbf{y}$ and $\hat{\mathbf{y}} - \mathbf{s} \circ \mathbf{y}$ have disjoint support, $\|\hat{\mathbf{y}} - \mathbf{s} \circ \mathbf{y}\|_2^2 + \|\mathbf{y} - \mathbf{s} \circ \mathbf{y}\|_2^2 = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$. Accordingly, adding $\|\mathbf{y} - \mathbf{s} \circ \mathbf{y}\|_2^2$ to both sides of (2), we can see that the bound is equivalent to

$$\mathbb{E}[\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2] \leq \left(1 + \frac{|\mathbf{s}|}{q - |\mathbf{s}| - 1}\right) \|\mathbf{y} - \mathbf{s} \circ \mathbf{y}\|_2^2.$$

I.e., we can extract a $(1+\epsilon)$ near-optimal approximation to $\mathbf{y}$ with sparsity pattern $\mathbf{s}$ from the result of $\mathbf{y}$'s multiplication with $q = \frac{|\mathbf{s}|}{\epsilon} + |\mathbf{s}| + 1 = O\left(\frac{|\mathbf{s}|}{\epsilon}\right)$ random Gaussian vectors.

With Lemma 12 in place, we proceed to our main result.

Proof of Theorem 2. The result of Theorem 2 is obtain using Algorithm 2 below.

We first analyze the complexity of the algorithm. Each iteration of the while loop generates two query matrices with q query vectors each. Moreover, by by Lemma 11, the loop terminates after at most $t = \log_2(n+m)$ iterations. Hence, the total query complexity is $2 \cdot q \cdot \log_2(n+m) = O\left(\frac{\text{degen}(\mathbf{S})}{\epsilon\delta} \cdot \log(m+n)\right)$ matvec queries, as desired. Moreover, all queries are made non-adaptively: the query vectors are chosen by the while loop based on the structure of $\mathbf{S}$. All queries are then issued together on Line 11. Finally, it is clear that the algorithm runs in time polynomial in n and m.

We next bound the approximation error. Note that $\mathcal{R}^i$ and $\mathcal{C}^i$ contain the indices of rows and columns that are processed at iteration i of the main for loop. $\mathbf{S}^{i,1}$ is a sparsity pattern containing all entries of $\mathbf{S}$ that have *not yet been learned* before iteration i. $\mathbf{S}^{i,2}$ is equal to $\mathbf{S}^{i,1}$ after further removing entries learned during iteration i in the row for loop at Line 14.

Algorithm 2 Sparse Matrix Approximation.

Input: Sparsity pattern $\mathbf{S} \in \{0,1\}^{m\times n}$, matvec query access to $\mathbf{A} \in \mathbb{R}^{m\times n}$, accuracy parameter $\epsilon \in (0,1)$, failure probability $\delta \in (0,1)$.

Output: Matrix $\tilde{\mathbf{B}} = \tilde{\mathbf{B}} \circ \mathbf{S}$ with sparsity pattern $\mathbf{S}$ that approximates $\mathbf{A}$.

1: Compute $d \leftarrow \text{degen}(\mathbf{S})$. Set $q \leftarrow \frac{4d}{\epsilon\delta/2} + 4d + 1$. ▷ $q = O(d/\epsilon\delta)$.
2: Initialize $t \leftarrow 1$ and row/column index sets $\mathcal{R}^{done} = \emptyset, \mathcal{C}^{done} = \emptyset$.
3: **while** $\text{nnz}(\mathbf{S}) > 0$ **do** ▷ Preconstruct query vectors.
4: Set $\mathbf{S}^{t,1} \leftarrow \mathbf{S}$. Let $\mathcal{R}^t$ contain the indices of all rows in $\mathbf{S}$ with $\leq 4d$ ones.
5: Construct $\mathbf{Z}^t \in \mathbb{R}^{n\times q}$ with i.i.d. Gaussian entries. Set $\mathbf{Z}^t_{\mathcal{C}^{done},:} \leftarrow \mathbf{0}$.
6: Update $\mathcal{R}^{done} \leftarrow \mathcal{R}^{done} \cup \mathcal{R}^t$, $\mathbf{S}_{\mathcal{R}^t,:} \leftarrow \mathbf{0}$
7: Set $\mathbf{S}^{t,2} \leftarrow \mathbf{S}$. Let $\mathcal{C}^t$ contain the indices of all columns in $\mathbf{S}$ with $\leq 4d$ ones.
8: Construct $\mathbf{W}^t \in \mathbb{R}^{m\times q}$ with i.i.d. Gaussian entries. Set $\mathbf{W}^t_{\mathcal{R}^{done},:} \leftarrow \mathbf{0}$.
9: Update $\mathcal{C}^{done} \leftarrow \mathcal{C}^{done} \cup \mathcal{C}^t$, $\mathbf{S}_{:,\mathcal{C}^t} \leftarrow \mathbf{0}$.
10: $t \leftarrow t + 1$.
11: Compute $\mathbf{L}^i \leftarrow \mathbf{A}^T\mathbf{W}^i$ and $\mathbf{R}^i \leftarrow \mathbf{A}\mathbf{Z}^i$ for all $i = 1,\ldots,t$. ▷ Requires $2 \cdot t \cdot q$ matvecs.
12: Initialize $\tilde{\mathbf{B}}$ as an $m \times n$ all zeros matrix.
13: **for** $i = 1,\ldots,t$ **do**
14: **for** $r \in \mathcal{R}^i$ **do**
15: Set $\mathbf{s} \leftarrow \mathbf{S}^{i,1}_{r,:}$ and set $\mathbf{H} = (\mathbf{Z}^i)^T$.
16: Set $\tilde{\mathbf{B}}_{r,:} \leftarrow \mathbf{I}_{\mathbf{s}}^T\mathbf{H}^+_{:,\mathbf{s}}\mathbf{R}^i_{r,:}$.
17: **for** $c \in \mathcal{C}^i$ **do**
18: Set $\mathbf{s} \leftarrow \mathbf{S}^{i,2}_{:,c}$ and set $\mathbf{H} = (\mathbf{W}^i)^T$.
19: Set $\tilde{\mathbf{B}}_{:,c} \leftarrow \mathbf{I}_{\mathbf{s}}^T\mathbf{H}^+_{:,\mathbf{s}}\mathbf{L}^i_{c,:}$.
20: **return** $\tilde{\mathbf{B}}$

We further define a mask matrix $\mathbf{M}^{i,1} \in \{0,1\}^{m\times n}$ so that $\mathbf{M}^{i,1}_{r,c} = 0$ if $c \in \mathcal{C}^1 \cup \ldots \cup \mathcal{C}^{i-1}$ and $\mathbf{M}^{i,1}_{r,c} = 1$ otherwise. Define $\mathbf{M}^{i,2} \in \{0,1\}^{m\times n}$ so that $\mathbf{M}^{i,1}_{r,c} = 0$ if $r \in \mathcal{R}^1 \cup \ldots \cup \mathcal{R}^{i-1}$ and $\mathbf{M}^{i,2}_{r,c} = 1$. Now, suppose we are at iteration i of the main for loop. By constructing the query matrices $\mathbf{Z}^i$ and $\mathbf{W}^i$ with certain rows zero'd out, we can observe that:

$$\mathbf{R}^i = \mathbf{A}\mathbf{Z}^i = (\mathbf{M}^{i,1} \circ \mathbf{A})\bar{\mathbf{Z}}^i \qquad \text{and} \qquad \mathbf{L}^i = \mathbf{A}^T\mathbf{W}^i = (\mathbf{M}^{i,2} \circ \mathbf{A})\bar{\mathbf{W}}^i,$$

where $\bar{\mathbf{Z}}^i \in \mathbb{R}^{n\times q}$ and $\bar{\mathbf{W}}^i \in \mathbb{R}^{m\times q}$ are Gaussian matrices with i.i.d. entries (equal to $\mathbf{Z}^i$ and $\mathbf{W}^i$ *before* the rows in $\mathcal{C}^{done}$ and $\mathcal{R}^{done}$ where zero'd out). Applying Lemma 12 to the expression computed on Lines 16 and 19, we thus have that, for all $r \in \mathcal{R}^i$ and $c \in \mathcal{C}^i$,

$$\mathbb{E}\left[\|(\mathbf{S}^{i,1} \circ \tilde{\mathbf{B}})_{r,:} - (\mathbf{S}^{i,1} \circ \mathbf{M}^{i,1} \circ \mathbf{A})_{r,:}\|_2^2\right] \leq \frac{\epsilon\delta}{2}\|(\mathbf{M}^{i,1} \circ \mathbf{A})_{r,:} - (\mathbf{S}^{i,1} \circ \mathbf{M}^{i,1} \circ \mathbf{A})_{r,:}\|_2^2, \quad (3)$$

$$\mathbb{E}\left[\|(\mathbf{S}^{i,2} \circ \tilde{\mathbf{B}})_{:,c} - (\mathbf{S}^{i,2} \circ \mathbf{M}^{i,2} \circ \mathbf{A})_{:,c}\|_2^2\right] \leq \frac{\epsilon\delta}{2}\|(\mathbf{M}^{i,2} \circ \mathbf{A})_{:,c} - (\mathbf{S}^{i,2} \circ \mathbf{M}^{i,2} \circ \mathbf{A})_{:,c}\|_2^2. \quad (4)$$

Observe that the indices with value one in $\mathbf{S}^{i,1}$ are a subset of those with value one in $\mathbf{M}^{i,1}$ since, like $\mathbf{M}^{i,1}$, $\mathbf{S}^{i,1}$ has every column in $\mathcal{C}^1 \cup \ldots \cup \mathcal{C}^{i-1}$ set to zero. Likewise, the indices with value one in $\mathbf{S}^{i,2}$ are a subset of those $\mathbf{M}^{i,2}$. This allows us to simplify the right hand sides of (3) and (4) by removing the $\mathbf{M}^{i,1}$ and $\mathbf{M}^{i,2}$ terms. Additionally, referring to the right hand sides, we observe that $\mathbf{S}^{i,1}_{r,:} = \mathbf{M}^{i,1}_{r,:} \cdot \mathbf{S}_{r,:}$ and $\mathbf{S}^{i,2}_{:,2} = \mathbf{M}^{i,2}_{:,c} \cdot \mathbf{S}_{:,c}$. We thus have:

$$(\mathbf{M}^{i,1} \circ \mathbf{A})_{r,:} - (\mathbf{S}^{i,1} \circ \mathbf{M}^{i,1} \circ \mathbf{A})_{r,:} = (\mathbf{M}^{i,1} \circ \mathbf{A})_{r,:} - (\mathbf{M}^{i,1} \circ \mathbf{S} \circ \mathbf{A})_{r,:} = \mathbf{A}_{r,:} - (\mathbf{S} \circ \mathbf{A})_{r,:}$$
$$(\mathbf{M}^{i,2} \circ \mathbf{A})_{:,c} - (\mathbf{S}^{i,2} \circ \mathbf{M}^{i,2} \circ \mathbf{A})_{:,c} = (\mathbf{M}^{i,2} \circ \mathbf{A})_{:,c} - (\mathbf{M}^{i,2} \circ \mathbf{S} \circ \mathbf{A})_{:,c} = \mathbf{A}_{:,c} - (\mathbf{S} \circ \mathbf{A})_{:,c}.$$

Plugging into (3) and (4), we obtain

$$\mathbb{E}\left[\|(\mathbf{S}^{i,1}\circ\tilde{\mathbf{B}})_{r,:}-(\mathbf{S}^{i,1}\circ\mathbf{A})_{r,:}\|_2^2\right]\leq\frac{\epsilon\delta}{2}\|\mathbf{A}_{r,:}-(\mathbf{S}\circ\mathbf{A})_{r,:}\|_2^2, \tag{5}$$

$$\mathbb{E}\left[\|(\mathbf{S}^{i,2}\circ\tilde{\mathbf{B}})_{:,c}-(\mathbf{S}^{i,2}\circ\mathbf{A})_{:,c}\|_2^2\right]\leq\frac{\epsilon\delta}{2}\|\mathbf{A}_{:,c}-(\mathbf{S}\circ\mathbf{A})_{:,c}\|_2^2. \tag{6}$$

Finally, we bound the total approximation error, $\|\tilde{\mathbf{B}}-\mathbf{S}\circ\mathbf{A}\|_F^2$, by noticing that every non-zero entry in $\tilde{\mathbf{B}}-\mathbf{S}\circ\mathbf{A}$ appears either as a non-zero in $(\mathbf{S}^{i,1}\circ\tilde{\mathbf{B}})_{r,:}-(\mathbf{S}^{i,1}\circ\mathbf{A})_{r,:}$ or in $(\mathbf{S}^{i,2}\circ\tilde{\mathbf{B}})_{:,c}-(\mathbf{S}^{i,2}\circ\mathbf{A})_{:,c}$ for some value of i. We thus have:

$$\begin{aligned}\mathbb{E}\left[\|\tilde{\mathbf{B}}-\mathbf{S}\circ\mathbf{A}\|_F^2\right]&\leq\frac{\epsilon\delta}{2}\sum_{r=1}^{m}\|\mathbf{A}_{r,:}-\mathbf{S}_{r,:}\circ\mathbf{A}_{r,:}\|_2^2+\frac{\epsilon\delta}{2}\sum_{c=1}^{n}\|\mathbf{A}_{:,c}-\mathbf{S}_{:,c}\circ\mathbf{A}_{:,c}\|_2^2\\&\leq\frac{\epsilon\delta}{2}\|\mathbf{A}-\mathbf{S}\circ\mathbf{A}\|_F^2+\frac{\epsilon\delta}{2}\|\mathbf{A}-\mathbf{S}\circ\mathbf{A}\|_F^2=\epsilon\|\mathbf{A}-\mathbf{S}\circ\mathbf{A}\|_F^2.\end{aligned}$$

By Markov's inequality, it follows that $\|\tilde{\mathbf{B}}-\mathbf{S}\circ\mathbf{A}\|_F^2\leq\epsilon\|\mathbf{A}-\mathbf{S}\circ\mathbf{A}\|_F^2$ with probability at least $1-\delta$. Finally, since $\tilde{\mathbf{B}}-\mathbf{S}\circ\mathbf{A}$ and $\mathbf{A}-\mathbf{S}\circ\mathbf{A}$ have disjoint support, we can add $\|\mathbf{A}-\mathbf{S}\circ\mathbf{A}\|_F^2$ to both sides of the equation to conclude that, with probability at least $1-\delta$,

$$\|\tilde{\mathbf{B}}-\mathbf{S}\circ\mathbf{A}\|_F^2+\|\mathbf{A}-\mathbf{S}\circ\mathbf{A}\|_F^2=\|\mathbf{A}-\tilde{\mathbf{B}}\|_F^2\leq(1+\epsilon)\cdot\|\mathbf{A}-\mathbf{S}\circ\mathbf{A}\|_F^2.$$

We conclude by noting that the dependence on $1/\delta$ in Theorem 2 could be improved to logarithmic using standard techniques, like the high-dimensional median trick (see, e.g., [2] for details). We omit this improvement for simplicity of presentation. ◀

5 Conclusion and Future Directions

Our work nearly resolves the complexity of fixed-sparsity pattern matrix approximation (and recovery) in the matvec query model, showing that the degeneracy of the sparsity pattern characterizes query complexity. Some questions remain open, however. For example, can we remove the $\log(m+n)$ factor in Theorem 2 so that it matches the lower bound in Theorem 3? Additionally, is it possible to implement Algorithm 2 using only "unstructured" query vectors (e.g., random Gaussians) instead of the "masked" random vectors we currently use?

It is also interesting to ask if our generic results for sparse families can be extended to other classes of matrix families. For example, sparse families are a special case of *linearly parameterized* matrix families. A linearly parameterized family consists of all matrices that can be written as $\sum_{i=1}^{q}c_iB_i$ for a chosen set of coefficients $c_1,\ldots,c_q\in\mathbb{R}$ and fixed basis matrices $B_1,\ldots,B_q\in\mathbb{R}^{m\times n}$ [30]. It is know that, in the recovery setting, such matrices can always be learned with $\sqrt{q}$ matvecs [1]. However, many linear families (including sparse families) require fewer, often on the order of q/n. Is there a simple parameter of $\{B_1,\ldots,B_q\}$ that governs the query complexity? What about general, non-linear matrix families?

References

1 Noah Amsel, Pratyush Avi, Tyler Chen, Feyza Duman Keles, Chinmay Hegde, Cameron Musco, Christopher Musco, and David Persson. Query efficient structured matrix learning. In *Proceedings of the 39th Annual Conference on Computational Learning Theory (COLT)*, 2026.

2 Noah Amsel, Tyler Chen, Feyza Duman Keles, Diana Halikias, Cameron Musco, and Christopher Musco. Fixed-sparsity matrix approximation from matrix-vector products. *SIAM Journal on Matrix Analysis and Applications*, 2026.

3 Noah Amsel, Tyler Chen, Feyza Duman Keles, Diana Halikias, Cameron Musco, Christopher Musco, and David Persson. Quasi-optimal hierarchically semi-separable matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, 2026.

4 Ainesh Bakshi, Kenneth L. Clarkson, and David P. Woodruff. Low-rank approximation with $1/\epsilon^{1/3}$ matrix-vector products. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC)*, 2022.

5 Robert A. Baston and Yuji Nakatsukasa. Stochastic diagonal estimation: probabilistic bounds and an improved algorithm. *arXiv:2201.10684*, 2022.

6 C. Bekas, E. Kokiopoulou, and Y. Saad. An estimator for the diagonal of a matrix. *Applied Numerical Mathematics*, 57(11–12):1214–1229, 2007.

7 Mathieu Blondel and Vincent Roulet. The elements of differentiable programming. *arXiv:2403.14606*, 2024.

8 Nicolas Boullé, Diana Halikias, Samuel E. Otto, and Alex Townsend. Operator learning without the adjoint. *Journal of Machine Learning Research*, 25(364):1–54, 2024. URL: `https://jmlr.org/papers/v25/24-0162.html`.

9 Nicolas Boullé, Diana Halikias, and Alex Townsend. Elliptic PDE learning is provably data-efficient. *Proceedings of the National Academy of Sciences*, 120(39), 2023.

10 Nicolas Boullé and Alex Townsend. Learning elliptic partial differential equations with randomized linear algebra. *Foundations of Computational Mathematics*, 23(2):709–739, 2023. `doi:10.1007/S10208-022-09556-W`.

11 Mark Braverman, Elad Hazan, Max Simchowitz, and Blake Woodworth. The gradient complexity of linear regression. In *Proceedings of the 33rd Annual Conference on Computational Learning Theory (COLT)*, 2020.

12 Vladimir Braverman, Aditya Krishnan, and Christopher Musco. Sublinear time spectral density estimation. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC)*, 2022.

13 Tyler Chen, Feyza Duman Keles, Diana Halikias, Cameron Musco, Christopher Musco, and David Persson. Near-optimal hierarchical matrix approximation from matrix-vector products. In *Proceedings of the 36th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2025.

14 Tyler Chen, Thomas Trogdon, and Shashanka Ubaru. Analysis of stochastic Lanczos quadrature for spectrum approximation. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

15 Sinho Chewi, Jaume de Dios Pont, Jerry Li, Chen Lu, and Shyam Narayanan. Query lower bounds for log-concave sampling. *Journal of the ACM*, 71(4), 2024. `doi:10.1145/3673651`.

16 Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, 2009.

17 Thomas F. Coleman and Jin-Yi Cai. The cyclic coloring problem and estimation of sparse Hessian matrices. *SIAM Journal on Algebraic Discrete Methods*, 7(2):221–235, 1986.

18 Thomas F. Coleman and Jorge J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20(1):187–209, 1983.

19 A. R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *IMA Journal of Applied Mathematics*, 13(1):117–119, 1974.

20 Gautam Dasarathy, Parikshit Shah, Badri Narayan Bhaskar, and Robert D. Nowak. Sketching sparse matrices, covariances, and graphs via tensor products. *IEEE Transactions on Information Theory*, 61(3):1373–1388, 2015. `doi:10.1109/TIT.2015.2391251`.

21 Yann N. Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in Neural Information Processing Systems 28 (NeurIPS)*, 2015.

22 Maarten V. de Hoop, Nikola B. Kovachki, Nicholas H. Nelsen, and Andrew M. Stuart. Convergence rates for learning linear operators from noisy data. *SIAM/ASA Journal on Uncertainty Quantification*, 11(2):480–513, 2023. `doi:10.1137/21M1442942`.

23 Michał Dereziński, Ethan N. Epperly, and Raphael A. Meyer. The matrix-vector complexity of $Ax = b$. *arXiv:2602.04842*, 2026.

24 Prathamesh Dharangutte and Christopher Musco. A tight analysis of hutchinson's diagonal estimator. In *Proceedings of the 6th Symposium on Simplicity in Algorithms (SOSA)*, 2023.

25 Alan Edelman. Eigenvalues and condition numbers of random matrices. *SIAM Journal on Matrix Analysis and Applications*, 9(4):543–560, 1988.

26 David Eppstein. Coloring, register allocation, Strahler number, and interval graphs. Lecture notes, CS 163/CS 265, University of California, Irvine, 2025. URL: `https://ics.uci.edu/~eppstein/163/lecture6c.pdf`.

27 P. Erdős and A. Hajnal. On chromatic number of graphs and set-systems. *Acta Mathematica Academiae Scientiarum Hungarica*, 17(1):61–99, 1966.

28 David Eriksson, Kun Dong, Eric Lee, David Bindel, and Andrew G Wilson. Scaling Gaussian process regression with derivatives. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*, volume 31, 2018.

29 D. Goldfarb and Ph. L. Toint. Optimal estimation of Jacobian and Hessian matrices that arise in finite difference calculations. *Mathematics of Computation*, 43(167):69–88, 1984.

30 Diana Halikias and Alex Townsend. Structured matrix recovery from matrix-vector products. *Numerical Linear Algebra with Applications*, 31(1), 2023. `doi:10.1002/NLA.2531`.

31 N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011. `doi:10.1137/090771806`.

32 Arun Jambulapati, Jerry Li, Christopher Musco, Aaron Sidford, and Kevin Tian. Structured semidefinite programming for recovering structured preconditioners. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

33 Lefteris M. Kirousis and Dimitris M. Thilikos. The linkage of a graph. *SIAM J. Comput.*, 25(3):626–647, 1996. `doi:10.1137/S0097539793255709`.

34 James Levitt and Per-Gunnar Martinsson. Linear-complexity black-box randomized compression of rank-structured matrices. *SIAM Journal on Scientific Computing*, 46(3):A1747–A1763, 2024.

35 James Levitt and Per-Gunnar Martinsson. Randomized compression of rank-structured matrices accelerated with graph coloring. *Journal of Computational and Applied Mathematics*, 451, 2024. `doi:10.1016/J.CAM.2024.116044`.

36 D. R. Lick and A. T. White. k-degenerate graphs. *Canadian Journal of Mathematics*, 22:1082–1096, 1970.

37 Lin Lin, Jianfeng Lu, and Lexing Ying. Fast construction of hierarchical matrix representation from matrix–vector multiplication. *Journal of Computational Physics*, 230(10):4071–4087, 2011. `doi:10.1016/J.JCP.2011.02.033`.

38 Lin Lin, Jianfeng Lu, Lexing Ying, Roberto Car, and Weinan E. Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems. *Communications in Mathematical Sciences*, 7(3):755–777, 2009.

39 Yang Liu, Xin Xing, Han Guo, Eric Michielssen, Pieter Ghysels, and Xiaoye Sherry Li. Butterfly factorization via randomized matrix-vector multiplications. *SIAM Journal on Scientific Computing*, 43(2), 2021. `doi:10.1137/20M1315853`.

40 David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, 1983. `doi:10.1145/2402.322385`.

41 S. Thomas Mccormick. Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem. *Math. Program.*, 26(2):153–171, 1983. `doi:10.1007/BF02592052`.

42 L. Métivier, F. Bretaudeau, R. Brossier, S. Operto, and J. Virieux. Full waveform inversion and the truncated newton method: quantitative imaging of complex subsurface structures. *Geophysical Prospecting*, 62(6):1353–1375, 2014.

43 Raphael A. Meyer, Cameron Musco, Christopher Musco, and David Woodruff. Hutch++: optimal stochastic trace estimation. In *Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA)*, 2021.

44 Raphael A. Meyer, William Swartworth, and David Woodruff. Understanding the Kronecker matrix-vector complexity of linear algebra. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

45 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 (NeurIPS)*, 2015.

46 Cameron Musco, Christopher Musco, Lucas Rosenblatt, and Apoorv Vikram Singh. Sharper bounds for chebyshev moment matching, with applications. In *Proceedings of the 38th Annual Conference on Computational Learning Theory (COLT)*, 2025.

47 Katherine J. Pearce, Anna Yesypenko, James Levitt, and Per-Gunnar Martinsson. Randomized rank-structured matrix compression by tagging. *arXiv:2501.05528*, 2025.

48 Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural computation*, 6(1):147–160, 1994. doi:10.1162/NECO.1994.6.1.147.

49 M. J. D. Powell and Ph. L. Toint. On the estimation of sparse hessian matrices. *SIAM Journal on Numerical Analysis*, 16(6):1060–1074, 1979.

50 Florian Schäfer, Matthias Katzfuss, and Houman Owhadi. Sparse cholesky factorization by kullback–leibler minimization. *SIAM Journal on Scientific Computing*, 43(3):A2019–A2046, 2021. doi:10.1137/20M1336254.

51 Florian Schäfer and Houman Owhadi. Sparse recovery of elliptic solvers from matrix-vector products. *SIAM Journal on Scientific Computing*, 46(2), 2024. doi:10.1137/22M154226X.

52 Max Simchowitz, Ahmed El Alaoui, and Benjamin Recht. Tight query complexity lower bounds for PCA via finite sample deformed Wigner law. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, 2018.

53 Xiaoming Sun, David P. Woodruff, Guang Yang, and Jialin Zhang. Querying a matrix through matrix-vector products. In *Proceedings of the 46th International Colloquium on Automata, Languages and Programming (ICALP)*, 2019.

54 Jok M. Tang and Yousef Saad. A probing method for computing the diagonal of a matrix inverse. *Numerical Linear Algebra with Applications*, 19(3):485–501, 2011. doi:10.1002/NLA.779.

55 Umberto Villa, Noemi Petra, and Omar Ghattas. hIPPYlib: an extensible software framework for large-scale inverse problems governed by PDEs: Part i: Deterministic inversion and linearized Bayesian inference. *ACM Trans. Math. Softw.*, 47(2), 2021. doi:10.1145/3428447.

56 David Woodruff, Fred Zhang, and Richard Zhang. Optimal query complexities for dynamic trace estimation. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 2022.

57 Zhewei Yao, Amir Gholami, Sheng Shen, Kurt Keutzer, and Michael W Mahoney. ADAHESSIAN: An adaptive second order optimizer for machine learning. In *Proceedings of the AAAI Conference on Artificial (AAAI)*, 2021.