Query Efficient Structured Matrix Approximation

Noah Amsel* Pratyush Avi* Tyler Chen* Feyza Duman Keles*
Chinmay Hegde* Christopher Musco* Cameron Musco† David Persson‡*

Abstract

We study the problem of learning a structured approximation (low-rank, sparse, banded, etc.) to an unknown matrix $\mathbf{A}$ given access to matrix-vector product queries of the form $\mathbf{x} \to \mathbf{A}\mathbf{x}$ and $\mathbf{x} \to \mathbf{A}^\mathsf{T}\mathbf{x}$. Among many other applications at the intersection of machine learning and scientific computing, this problem arises as a natural abstraction of the operator learning problem in Scientific Machine Learning [Boullé, Townsend, FoCM 2023]. We take a step towards understanding the sample complexity of structured matrix approximation by proving matching upper and lower bounds for the number of queries needed to find a near-optimal approximation to $\mathbf{A}$ from any structured family, $\mathcal{F}$, of finite size. In particular, we show that $\Theta(\sqrt{\log|\mathcal{F}|})$ queries always suffice, and are necessary in the worst case. The upper bound improves on the natural baseline of $O(\log|\mathcal{F}|)$ queries. In this workshop submission, we provide a self-contained proof of this result in the simplified realizable setting, demonstrating the key ideas and techniques used in forthcoming work to prove the general version.

1 Introduction

Operator learning has emerged as a central task in Scientific Machine Learning (SciML) [Lu+21; Li+21; BT24]. In its most abstract form, the goal is to find an approximation to an unknown operator $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ that maps between function spaces given training observations of the form $(x, \mathcal{A}x)$, where $x \in \mathcal{X}$. In SciML, this problem arises when developing data-driven methods for solving partial differential equations (PDEs). Here, $\mathcal{A}$ is the solution operator of a differential equation, which maps an input forcing function, boundary condition, and/or equation parameters to a solution of the corresponding PDE. Given example input-output pairs (typically computed using a traditional numerical solver), the goal is to learn an approximation to $\mathcal{A}$ (e.g., a neural network) in order to efficiently solve the differential equation for future inputs.

The benefit of a learning-based approach is that it can lead to higher efficiency in applications where the same equation needs to be solved for a multitude of different inputs. This is often the case in applications like uncertainty quantification (UQ), where PDE parameters are random quantities [GFWG10]; in model-driven design, where parameters are sequentially modified to optimize some objective [LR10]; and in data assimilation [Bin+17], where parameters are tuned to fit the solution to measured data.

The operator learning approach has been remarkably effective for several use cases, yielding approximations to solution operators that can be applied orders-of-magnitude faster than directly solving the PDE using a traditional numerical solver [Bar+23; Lu+21]. However, faster inference comes at the price of slow training, which is often dominated by the cost of computing a dataset of input-output pairs using a traditional numerical solver. Given the inherent computational cost of this task, understanding the *sample complexity* of operator learning is a critical challenge in SciML. Recent research has sought to put the sample complexity problem on firmer theoretical footing [ADM24; AGM25; BT24; BHOT24; BHT23].

1.1 From Operator Learning to Structured-Matrix Approximation

In this paper, we study a recent formalization of the operator learning problem put forward by Boullé, Townsend, and others, with the aim of capturing the important subclass of *linear PDEs* [BT23; BHT23;

*New York Univeristy (noah.amsel@nyu.edu, pratyushavi@nyu.edu, tyler.chen@nyu.edu, fd2135@nyu.edu, chinmay.h@nyu.edu, cmusco@nyu.edu)
†University of Massachusetts Amherst (cmusco@cs.umass.edu)
‡Flatiron Institute (dpersson@flatironinstitute.org)

SO24]. For such problems, the solution operator we wish to approximate is linear, and moreover, if the input and output domains are discretized, it can actually be seen as a finite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$.[1] The goal is to learn an approximation to this unknown $\mathbf{A}$ given input-output pairs of the form:

$$(\mathbf{x}, \mathbf{A}\mathbf{x}) \qquad \text{or} \qquad (\mathbf{x}, \mathbf{A}^\mathsf{T}\mathbf{x}),$$

where $\mathbf{x} \in \mathbb{R}^n$ can be chosen to be any vector. We consider the setting where we can issue multiple "matrix-vector product" queries $\mathbf{x}_1, \ldots, \mathbf{x}_q$ and where each $\mathbf{x}_{i+1}$ can be chosen adaptively depending on the results of the previous queries, $\mathbf{x}_1, \ldots, \mathbf{x}_i$. For shorthand, we refer to these vectors as *matvec queries*. The number of matvec queries needed perform a particular operation on $\mathbf{A}$, or compute a quantity about $\mathbf{A}$, is known as the "query complexity". In general, understanding the matvec query complexity of various matrix problems has become a central topic of theoretical work in numerical linear algebra, with applications across computational science and machine learning [SER18; BHSW20; BCW22; BN23; MMMW21; Che+24].

Without making any assumptions on $\mathbf{A}$, it is easy to see that $\Omega(n)$ queries are needed to learn a meaningful approximation (see [HT23] for a formal argument). Moreover, $\mathbf{A}$ can be exactly recovered by choosing the n standard basis vectors as queries. As such, the matrix approximation problem is only interesting if we impose additional constraints. Most commonly, we restrict ourselves to outputting an approximation from some hypothesis class that is a structured family of matrices. This restriction is natural if our knowledge of the problem at hand indicates that $\mathbf{A}$ lies in or nearly lies in this structured class. For example, $\mathcal{F}$ might be the class of low-rank matrices, sparse matrices, or banded matrices. For applications in SciML (e.g., learning solution operators for linear PDEs) hierarchical low-rank matrices [BT23; BHT23] and related block-structured matrices [WT23] have been proven to yield especially effective approximations. In this work, we study a general structured approximation problem of the following form:

Problem 1. *For an unknown target matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and a matrix family $\mathcal{F} \subset \mathbb{R}^{n\times n}$ (i.e., a hypothesis class), find $\tilde{\mathbf{B}} \in \mathcal{F}$ satisfying*

$$\|\mathbf{A} - \tilde{\mathbf{B}}\|_F \leq C \cdot \min_{\mathbf{B}\in\mathcal{F}} \|\mathbf{A} - \mathbf{B}\|_F$$

for some approximation factor $C \geq 1$ using as few matvec queries to $\mathbf{A}$ and $\mathbf{A}^\mathsf{T}$ as possible.

In other words, our goal is to find an approximation from $\mathcal{F}$ that is competitive with the *best possible* approximation to $\mathbf{A}$ in the class. We measure error with respect to the Frobenius norm, but other norms, such as the spectral norm, could be considered as well. The problem is interesting in the setting where C is close to 1 (e.g., $C = 1 + \epsilon$ for some small ϵ), in the case when C is a fixed constant, or even when C is allowed to grow in n or other problem parameters [Ams+25a].

Beyond operator learning, sample-efficient methods for structured matrix approximation have many other applications at the intersection of machine learning and computational science. For example, among other applications [Amb+20], methods for approximation via hierarchical low-rank matrices are used to construct fast direct solvers for integral equations given access to an implicit solver like a fast multipole method [LLY11; Mar16]. Methods for learning diagonal and sparse approximations to Hessian matrices—for which $\mathbf{x} \to \mathbf{A}\mathbf{x}$ can be computed efficiently using automatic differentiation [Pea94]—are used to construct preconditioners for first-order optimization methods [DVB15; YGKM20]. And as a special case, Problem 1 captures the ubiquitous problem of near-optimal low-rank approximation [Woo14; HMT11; MM15].

1.2 Prior Work

Given the wide applicability of structured matrix approximation, significant research effort has focused on obtaining query efficient algorithms for solving Problem 1 for various choices of the matrix family $\mathcal{F}$. There has also been interest in proving lower bounds on query complexity. Perhaps the most well-studied class is that of rank-k matrices [HMT11; RST09; Woo14; MM15], for which it is known that $O(k/\epsilon^{1/3})$ queries suffice [BCW22] and $\Omega(k + 1/\epsilon^{1/3})$ queries are necessary [BN23] to achieve an approximation factor $C = 1 + \epsilon$. There has also been interest in families with a fixed pattern of s non-zeros per row, including diagonal, banded, and block diagonal matrices [BKS07; TS11; BN22; DM23], as well as sparse matrices with unknown

[1] We can assume that the matrix is square without loss of generality by padding with zeros.

sparsity patterns [CPR74; CC86; CM83; PN24; WEV13; DSBN15; WSB11; SO24]. Matching upper and lower bounds of $\Theta(s/\epsilon)$ queries are known for $C = 1+\epsilon$ when the sparsity pattern is fixed [Ams+24]. Recent work has also analyzed matvec query algorithms for hierarchically low-rank matrix families (HOLDR, HSS, etc.) [LLY11; LM24; Mar16; Che+25; Ams+25b], butterfly matrices [LY17; Liu+21], and more [HT23].

1.3 Beyond Specific Matrix Families

Given this large and growing body of work studying *specific* matrix-families, it is natural to ask if a more general theory exists for determining the query complexity of approximation via a given class, $\mathcal{F}$. In supervised learning, a rich theory based on VC dimension, Rademacher complexity, and covering numbers categorizes the complexity of learning arbitrary hypothesis classes. Does an analogous theory hold for the problem of structured matrix approximation with arbitrary hypothesis classes?

We believe this problem is interesting because Problem 1 differs in important ways from standard supervised learning problems. Most obviously, data collection is active and adaptive, although there has been significant work on active learning [Set09]. Perhaps more interesting is the fact that the target/label corresponding to any data example $\mathbf{x} \in \mathbb{R}^n$ is itself an n dimensional vector, $\mathbf{Ax}$. We might therefore hope to learn *more* from each example than in a problem with scalar valued targets. An example of a such a problem is the well-studied rank-1 matrix sensing problem from compressed sensing, in which the goal is to find an approximation to $\mathbf{A}$, but the matrix is only accessible via queries of the form $\mathbf{x}^\mathsf{T}\mathbf{Ay}$ [ZJD15]. Indeed, for many of the families discussed above, there is a lot to be gained from the multidimensional output. Rank-k matrices, for example, requires $O(nk)$ parameters to represent, and $\Omega(nk)$ queries to learn in the matrix-sensing setting, but can be learned using just $O(k)$ matvec queries. Similarly, matrices with s non-zeros per row have $O(ns)$ parameters, but can be learned using $O(s)$ matrix-vector product queries [Ams+24].

In this work, we take an initial step towards understanding structured matrix-approximation in greater generality by considering any *finite* family of matrices $\mathcal{F}$ (i.e, with finite size $|\mathcal{F}|$). Such families require $\log(|\mathcal{F}|)$ bits (i.e., parameters) to represent, and it is not hard to show using standard techniques from sketching/compressed sensing that it is always possible to solve Problem 1 with fixed constant C using $O(\log|\mathcal{F}|)$ queries (see Section 2.2 for more details). Indeed, this bound can even be achieved in the more restrictive setting where queries are of the form $\mathbf{x}^\mathsf{T}\mathbf{Ay}$. Optimistically, we might hope to improve the bound to something like $O(\log|\mathcal{F}|/n)$ given that, as discussed above, any query of the form $\mathbf{Ax}$ returns up to n times more information than a single quadratic form query.

Unfortunately however, it is easy to come up with families for which achieving such a bound is impossible: simply consider the family of matrices that are zero everywhere except their top left $k \times k$ block, on which they can equal any possible $k \times k$ binary matrix. This family has size 2^{k^2}, so $\log(|\mathcal{F}|) = k^2$. However, it clearly requires $\Omega(k) > k^2/n$ queries to learn an approximation from this family to any constant accuracy (see Section 4 for more details). Perhaps more realistically, constant rank butterfly matrices are a common class of structured matrices used to approximate non-uniform Fourier transforms and other operators. Such matrices require $O(n\log n)$ parameters to represent but our best known matvec-query algorithms for approximation via constant-rank butterfly matrix require $\tilde{O}(\sqrt{n})$ matvecs [Liu+21], and $\tilde{\Omega}(\sqrt{n})$ can be shown to be tight. Interestingly, even for these "hard" matrix families, matvec queries still allow for a significant *quadratic* improvement over, e.g., scalar valued $\mathbf{x}^\mathsf{T}\mathbf{Ay}$ queries.

1.4 Our Contributions

Our main result is to establish that the examples of the previous paragraph are essentially worst-case. In particular, in forthcoming work, we prove the following general result for any finite matrix family.

Theorem 1. *There exists an algorithm that, for any matrix family $\mathcal{F} \subset \mathbb{R}^{n\times n}$ with size $|\mathcal{F}|$, finds, with high probability, an approximation $\tilde{\mathbf{B}} \in \mathcal{F}$ to a target matrix $\mathbf{A}$ using $\tilde{O}(\sqrt{\log|\mathcal{F}|})$ matrix-vector product queries with $\mathbf{A}$ and $\mathbf{A}^\mathsf{T}$, such that, for a fixed constant C,*

$$\|\tilde{\mathbf{B}} - \mathbf{A}\|_F < C \cdot \min_{\mathbf{B}\in\mathcal{F}} \|\mathbf{B} - \mathbf{A}\|_F.$$

This theorem establishes that, surprisingly, it is possible to *always* achieve a quadratic improvement over the naive $O(\log|\mathcal{F}|)$ bound. Our algorithm that achieves this bound is adaptive and strongly relies on the

ability to query both $\mathbf{A}$ and $\mathbf{A}^\mathsf{T}$: it is not hard to see that a lower bound of $\Omega(\log|\mathcal{F}|)$ holds if we can only query $\mathbf{A}$ (see Section 4). The same lower bound holds for $\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{y}$ queries since these can be evaluated while only querying $\mathbf{A}\mathbf{y}$.

In this workshop paper, we prove a special case of Theorem 1 when $\mathbf{A}$ is assumed to lie in the hypothesis class $\mathcal{F}$: i.e., $\min_{\mathbf{B}\in\mathcal{F}}\|\mathbf{A}-\mathbf{B}\|_F = 0$. In machine learning language, we might call this the *realizable setting*. In particular, we prove the following in Section 3:

Theorem 2. *There exists an algorithm (Algorithm 2) that, for any matrix family $\mathcal{F}\subset\mathbb{R}^{n\times n}$ with size $|\mathcal{F}|$ and any $\delta\in(0,1)$, can recover an unknown matrix $\mathbf{A}\in\mathcal{F}$ with probability $1-\delta$ using $O\left(\sqrt{\log|\mathcal{F}|}+\log(1/\delta)\right)$ matrix-vector product queries with $\mathbf{A}$ and $\mathbf{A}^\mathsf{T}$.*

The proof of Theorem 2 is based on a simulation argument: we show how to simulate a basic one-sided $O(\log|\mathcal{F}|)$ query algorithm (i.e., of the form $\mathbf{A}\mathbf{x}$) with $\tilde{O}(\sqrt{\log(|\mathcal{F}|)})$ two-sided queries (i.e., of the form $\mathbf{A}\mathbf{x}$ and $\mathbf{A}^\mathsf{T}\mathbf{x}$). The proof captures the main ideas of that of Theorem 1.

Remark 1. A reader might notice that, if we are not more careful with how we define our computational model, in the special case when $\mathbf{A}\in\mathcal{F}$, the problem above can be trivially solved with a *single query*: choose $\mathbf{x}$ to be a random real-valued Gaussian vector. With probability 1, $\mathbf{A}\mathbf{x}\neq\mathbf{C}\mathbf{x}$ for any $\mathbf{C}\in\mathcal{F}$ with $\mathbf{C}\neq\mathbf{A}$, so we can uniquely determine $\mathbf{A}$ from the result of the query. This approach, however, requires $\mathbf{x}$ to have arbitrary precision entries (which is unrealistic in applications) and does not extend to the approximation setting of Theorem 1. Our algorithm, on the other hand, uses bounded bit complexity vectors (in fact, all queries have ±1 entries) and naturally extends to the more challenging setting where $\mathbf{A}\notin\mathcal{F}$.

2 Preliminaries

Notation. For $i\in\{1,\dots,n\}$, a_i is the ith entry of the vector $\mathbf{a}\in\mathbb{R}^n$. For $i\in\{1,\dots,m\}$ and $j\in\{1,\dots,n\}$, B_{ij} is the entry at the ith row and jth column of the matrix $\mathbf{B}\in\mathbb{R}^{m\times n}$. For $\mathbf{B}\in\mathbb{R}^{m\times n}$, $\|\mathbf{B}\|_F = \left(\sum_{i=1}^m\sum_{j=1}^n B_{ij}^2\right)^{1/2}$ is the Frobenius norm. For $\mathbf{a},\mathbf{b}\in\mathbb{R}^n$, $\langle\mathbf{a},\mathbf{b}\rangle = \sum_{i=1}^n a_ib_i$ is the inner product. $\mathrm{unif}(\{-1,1\}^{m\times n})$ is the distribution over $m\times n$ matrices with independently and identically drawn (i.i.d.) Rademacher entries, i.e., uniform over $\{-1,1\}$. Throughout, $\log(x)$ refers to the base-2 logarithm of x.

For $\mathbf{x}\in\mathbb{R}^n$, a *query* is defined as an oracle call to a matrix-vector mutiplication routine for our matrix $\mathbf{A}$, or its transpose. I.e., a query is either of the form $\mathbf{x}\to\mathbf{A}\mathbf{x}$ or $\mathbf{x}\to\mathbf{A}^\mathsf{T}\mathbf{x}$; we call these *right* and *left* queries, respectively. The *query complexity* of an algorithm is the total number of oracle calls that it makes to the target matrix $\mathbf{A}$. We do not consider computational costs in this paper, only query complexity. We note that all of our methods run in time linear in $|\mathcal{F}|$, but the goal in practice is typically to obtain much faster runtimes, i.e., on the order of $\mathrm{poly}(n,\log|\mathcal{F}|)$, which requires $\mathcal{F}$ to have some compact representation. Obtaining simultaneously low query complexity and computational complexity is an interesting topic in its own right.

2.1 Zero Testing

Our analysis requires the following standard fact on detecting if a matrix is zero via multiplication by a random vector. Readers may have seen this fact before in the context of Freivalds' algorithm for testing the correctness of a matrix-matrix product.

Claim 1. *Consider any non-zero $\mathbf{X}\in\mathbb{R}^{m\times n}$ and let $\mathbf{V}\sim\mathrm{unif}(\{-1,1\}^{n\times q})$ with $q=\log_2(1/\delta)$ columns.*

$$\Pr\left[\mathbf{X}\mathbf{V}=\mathbf{0}\right]\leq\delta,$$

where $\mathbf{0}$ is the all 0s matrix.

Proof. Every non-zero matrix $\mathbf{X}$ has at least one non-zero row, $\mathbf{x}$. It suffices to show that $\Pr\left[\mathbf{x}^\mathsf{T}\mathbf{V}=0\right]\leq\delta$. To prove this, it suffices to show that $\mathbf{x}^\mathsf{T}\mathbf{v}^{(i)}\neq0$ for *any* column $\mathbf{v_i}$ in $\mathbf{V}$. Moreover, since these columns are chosen independently, it suffices to show that $\Pr\left[\mathbf{x}^\mathsf{T}\mathbf{v}^{(i)}=0\right]\leq1/2$.

To see why this is the case, observe that $\mathbf{x}$ must have at least one non-zero entry, x_k. Then, for any $\mathbf{w} \in \{-1, 1\}^n$, consider the vector $\mathbf{w}'$ formed by negating the kth entry and keeping all other entries the same. It is not hard to see that, for any $\mathbf{x}$, either $\langle \mathbf{w}, \mathbf{x} \rangle$ *or* $\langle \mathbf{w}', \mathbf{x} \rangle$ is non-zero. In particular, $\langle \mathbf{w}, \mathbf{x} \rangle - \langle \mathbf{w}', \mathbf{x} \rangle = \pm 2 w_k x_k$, which is nonzero because $x_k \neq 0$ and $w_k \in \{-1, 1\}$. Since the kth entry of $\mathbf{v}$ is equally like to be 1 or negative -1, we immediately conclude that $\Pr\left[\mathbf{x}^\mathsf{T}\mathbf{v}^{(i)} \neq 0\right] \geq 1/2$. The probability that $\Pr\left[\mathbf{x}^\mathsf{T}\mathbf{v}^{(i)} = 0\right]$ for all $i \in \{1, \ldots, q\}$ is thus at most $(1/2)^q \leq \delta$, which establishes the claim. □

2.2 Warm-Up: $O(\log |\mathcal{F}|)$ queries

It is not hard to use Claim 1 to give an $O(\log |\mathcal{F}|)$ matvec query algorithm for recovering a matrix $\mathbf{A}$ from a finite family $\mathcal{F}$. While there are many ways to see why this is the case, we consider an iterative application of the claim, using it to refine a candidate set of matrices, $\mathcal{C}$. In particular, begin with $\mathcal{C} = \mathcal{F}$. Draw a single random vector, $\mathbf{v} \in \mathbb{R}^n$, and remove any candidates $\mathbf{B} \in \mathcal{C}$ such that $\mathbf{Bv} \neq \mathbf{Av}$. After $O(\log(|\mathcal{F}|/\delta))$ repetitions, simply return any matrix remaining $\mathcal{C}$. We will argue that with high probability, the only such matrix is $\mathbf{A}$ itself. We formalize this "iterative refinement" approach in Algorithm 1. Our main result (Theorem 2) will achieve $O(\sqrt{\log |\mathcal{F}|})$ query complexity via an efficient simulation of this algorithm.

Algorithm 1 One-Sided Recovery

Input: Finite family $\mathcal{F}$, failure probability δ, oracle for computing $\mathbf{Ax}$ for target matrix $\mathbf{A} \in \mathcal{F}$.
Output: Matrix $\mathbf{A}$ with probability $\geq 1 - \delta$.

1: Initialize candidate set $\mathcal{C}^{(0)} = \mathcal{F}$.
2: **for** $i = 1, \ldots, \log(|\mathcal{F}|/\delta)$ **do**
3: Draw $\mathbf{v}^{(i)} \sim \text{unif}(\{-1, 1\}^n$
4: For all $\mathbf{B} \in \mathcal{C}^{(i-1)}$, add $\mathbf{B}$ to $\mathcal{C}^{(i)}$ if $\mathbf{Bv}^{(i)} = \mathbf{Av}^{(i)}$.
5: **return** Any remaining $\mathbf{B} \in \mathcal{C}^{(\log(|\mathcal{F}|/\delta))}$.

Theorem 3. *Algorithm 1 returns the unknown matrix $\mathbf{A} \in \mathcal{F}$ with probability $1 - \delta$, and uses $O\left(\log(|\mathcal{F}|/\delta)\right)$ (one-sided) matrix-vector product queries with $\mathbf{A}$.*

Proof. First, it is clear that $\mathbf{A}$ always gets added to $\mathcal{C}^{(i)}$ in Line 4, so the matrix remains in $\mathcal{C}^{(\log(|\mathcal{F}|/\delta))}$. It thus suffices to show that no $\mathbf{B} \neq \mathbf{A}$ also remain in $\mathcal{C}^{(\log(|\mathcal{F}|/\delta))}$.

To see why this is the case, fix a single matrix $\mathbf{B}$. By Claim 1, $\Pr[\mathbf{Bv}^{(i)} = \mathbf{Av}^{(i)}] \leq 1/2$ for all i. The probability that $\mathbf{B}$ is added to $\mathcal{C}^{(i)}$ conditioned on being in $\mathcal{C}^{(i-1)}$ is thus $\leq 1/2$. We conclude that:

$$\Pr\left[\mathbf{B} \in \mathcal{C}^{(\log(|\mathcal{F}|/\delta))}\right] \leq \left(\frac{1}{2}\right)^{\log(|\mathcal{F}|/\delta)} = \frac{\delta}{|\mathcal{F}|}.$$

By a union bound, we conclude that the probability that *any* $\mathbf{B} \in \mathcal{F}$ with $\mathbf{B} \neq \mathbf{A}$ is in $\mathcal{C}^{(\log(|\mathcal{F}|/\delta))}$ is less than $\frac{\delta}{|\mathcal{F}|} \cdot (|\mathcal{F}| - 1) \leq \delta$. So with probability $\geq 1 - \delta$, the algorithm returns $\mathbf{A}$, as desired. It uses one matvec query per iteration to compute $\mathbf{Av}^{(i)}$, so a total of $\log(|\mathcal{F}|/\delta)$ queries. □

3 Main Result

In this section, we will present and analyze Algorithm 2, which achieves the $\tilde{O}(\sqrt{\log |\mathcal{F}|})$ query complexity of Theorem 2. The aim of the method is to use a combination of left and right queries to efficiently simulate the one-sided Algorithm 1 given in Section 2.2. Like that method, the goal is to repeatedly reduce the size of a candidate set $\mathcal{C}$, initially containing all of $\mathcal{F}$, until it contains only $\mathbf{A}$ itself.

The high-level idea of our algorithm begins with the observation that any potential right query $\mathbf{x}$ partitions the current candidate set $\mathcal{C}$ into subsets of matrices that would return the same response given that query. We call such subsets "buckets", and give a formal definition below:

Definition 1. Let $\mathcal{C} \subset \mathbb{R}^{n \times n}$ be a family of matrices . For query vector $\mathbf{x} \in \mathbb{R}^n$, a *bucket* $T_{\mathbf{x} \to \mathbf{z}}$ is the set of all matrices $\mathbf{B} \in \mathcal{C}$ such that $\mathbf{Bx} = \mathbf{z}$, i.e.,

$$T_{\mathbf{x} \to \mathbf{z}} = \{\mathbf{B} \in \mathcal{C} : \mathbf{Bx} = \mathbf{z}\}.$$

To recovery $\mathbf{A}$ using a small number of queries, the hope is that we can find a query $\mathbf{x}$ for which $T_{\mathbf{x}\to\mathbf{Ax}}$ is much smaller than the current candidate set $\mathcal{C}$ – i.e., we want $\mathbf{Ax}$ to lie in a small bucket, since we can update our candidate set to only contain other matrices in that bucket. If we choose a random Rademacher vector, as in the analysis of Theorem 3, we are guaranteed that $T_{\mathbf{x}\to\mathbf{Ax}}$ is no more than roughly half the size of $\mathcal{C}$. Accordingly, we obtain a query complexity of $O(\log|\mathcal{F}|)$ by repeatedly halving the candidate set.

Our key idea to improve this bound is to take advantage of a natural dichotomy that arises. While $T_{\mathbf{x}\to\mathbf{Ax}}$ is guaranteed to be roughly $1/2$ the size of $\mathcal{C}$ in the *worst-case* (via Claim 1), in many cases it can be much much smaller. If it is much smaller, than we can make more progress by issuing the query $\mathbf{Ax}$. In particular, imagine an ideal situation where we always have $T_{\mathbf{x}\to\mathbf{Ax}} \lesssim |\mathcal{C}|/2^{\sqrt{\log|\mathcal{F}|}}$. Then we could hope to converge to a candidate set containing just $\mathbf{A}$ after just $\log_{2^{\sqrt{\log|\mathcal{F}|}}}(|\mathcal{F}|) = O(\sqrt{\log|\mathcal{F}|})$ queries.

Of course, this ideal situation may not always arise. However, suppose on the other hand that $T_{\mathbf{x}\to\mathbf{Ax}} > |\mathcal{C}|/2^{\sqrt{\log|\mathcal{F}|}}$ – i.e., $\mathbf{Ax}$ lies in a "large bucket". There can only be at most $2^{\sqrt{\log|\mathcal{F}|}}$ such buckets, each corresponding to some different length n vector $\mathbf{z}$. In this case, our idea is to use a fixed set of random left queries, $\mathbf{\Pi A}$, to *learn* $\mathbf{Ax}$ *without ever issuing the query* $\mathbf{x}$. In particular, since there are only have $2^{\sqrt{\log|\mathcal{F}|}}$ possible choices for $\mathbf{Ax}$, we only need $\mathbf{\Pi}$ to have roughly $O(\sqrt{\log|\mathcal{F}|})$ rows in order to uniquely identify $\mathbf{Ax}$ from $\mathbf{\Pi Ax}$ with high probability. Moreover, the same left sketch can be reused across iterations of the our algorithm to refine the candidate set, recovering $\mathbf{Ax}$ for any query $\mathbf{x}$ that does not make sufficient.

This high-level strategy is formalized in Algorithm 2, which is presented and analyzed below.

Algorithm 2 Finite Family Recovery

Input: Finite family $\mathcal{F}$, failure probability δ, oracles for computing $\mathbf{Ax}$ and $\mathbf{A}^\mathsf{T}\mathbf{x}$ for target matrix $\mathbf{A}\in\mathcal{F}$.
Output: Matrix $\mathbf{A}$ with probability $\geq 1-\delta$.

Let $q \leftarrow \lceil 3\sqrt{\log|\mathcal{F}|} + 2\log(1/\delta)\rceil$.
Draw $\mathbf{\Pi}\sim \text{unif}(\{-1,1\}^{q\times n})$ and compute $\mathbf{\Psi}\leftarrow\mathbf{\Pi A}$. ▷ Costs $O(\sqrt{\log|\mathcal{F}|})$ left queries.
Initialize candidate set $\mathcal{C}^{(0)}\leftarrow\mathcal{F}$.
for $i = 1,\ldots,\log(|\mathcal{F}|/\delta)$ **do**
  Draw $\mathbf{v}^{(i)}\sim\text{unif}(\{-1,1\}^n)$.
  Compute set $\mathcal{Z}^{(i)}\leftarrow\left\{\mathbf{z}:\mathbf{z}=\mathbf{B}\mathbf{v}^{(i)} \text{ for some } \mathbf{B}\in\mathcal{C}^{(i-1)}\right\}$ ▷ Set of all possible outputs for $\mathbf{Av}^{(i)}$.
  For all $\mathbf{z}\in\mathcal{Z}^{(i)}$, compute $T_{\mathbf{v}^{(i)}\to\mathbf{z}} = \left\{\mathbf{B}\in\mathcal{C}^{(i-1)}:\mathbf{B}\mathbf{v}^{(i)}=\mathbf{z}\right\}$. ▷ Bucket for each output $\mathbf{z}$.
  Collect large bucket representatives $\mathcal{L}^{(i)} = \left\{\mathbf{z}\in\mathcal{Z}^{(i)} : |T_{\mathbf{v}^{(i)}\to\mathbf{z}}| > \frac{|\mathcal{C}^{(i-1)}|}{2^{\sqrt{\log|\mathcal{F}|}}}\right\}$.
  if $\mathbf{\Psi}\mathbf{v}^{(i)} = \mathbf{\Pi z}$ for some $\mathbf{z}\in\mathcal{L}^{(i)}$ **then**
    $\mathbf{z}_{out}\leftarrow\mathbf{z}$ ▷ Goal is that $\mathbf{z}_{out}=\mathbf{Av}^{(i)}$.
  else
    $\mathbf{z}_{out}\leftarrow\mathbf{Av}^{(i)}$. ▷ Costs 1 right query.
  $\mathcal{C}^{(i)}\leftarrow T_{\mathbf{v}^{(i)}\to\mathbf{z}_{out}}$
return Any remaining $\mathbf{B}\in\mathcal{C}^{(\log(|\mathcal{F}|/\delta))}$

Proof of Theorem 2. In addition to the variables defined in the algorithm, let $\mathcal{S}^{(i)}$ be the set of "small buckets representitives" included by query $\mathbf{v}^{(i)}$. Concretely,

$$\mathcal{S}^{(i)} = \left\{\mathbf{z}\in\mathcal{Z}^{(i)} : |T_{\mathbf{v}(i)\to\mathbf{z}}| \leq \frac{|\mathcal{C}^{(i-1)}|}{2^{\sqrt{\log|\mathcal{F}|}}}\right\}$$

Note that $\mathcal{S}^{(i)}\cup\mathcal{L}^{(i)} = \mathcal{Z}^{(i)}$. We consider two cases in our analysis: when $\mathbf{Av}^{(i)}\in\mathcal{S}^{(i)}$ (i.e., $\mathbf{Av}^{(i)}$ is in a small bucket), we should issue the query, and make significantly progress towards reducing the size of $\mathbf{C}^{(i)}$. On the other hand, when $\mathbf{Av}^{(i)}\in\mathcal{L}^{(i)}$ (i.e., $\mathbf{Av}^{(i)}$ is in a large bucket), we should detect this, and avoid issuing the query explicitly by using our left queries $\mathbf{\Psi}$ to recover $\mathbf{Av}^{(i)}$.

Case 1: $\mathbf{Av}^{(i)}\in\mathcal{S}^{(i)}$. The key claim we will prove holds in this case is as follows:

$$\mathbf{\Psi}\mathbf{v}^{(i)}\neq\mathbf{\Pi z} \text{ for all } \mathbf{z}\in\mathcal{L}^{(i)} \text{ with probability } \geq 1-\frac{\delta^2}{2^{2\sqrt{\log|\mathcal{F}|}}}. \tag{1}$$

A consequence of this fact is that, with high probability, we issue 1 right matvec query on Line 11 of Algorithm 2 whenever $\mathbf{A}\mathbf{v}^{(i)} \in \mathcal{S}^{(i)}$. Moreover, we will have $|\mathcal{C}^{(i)}| \leq \frac{|\mathcal{C}^{(i-1)}|}{2^{\sqrt{\log|\mathcal{F}|}}}$, and $\mathcal{C}^{(i)}$ contains $\mathbf{A}$, as required. So, while a right query is issued, we make significant progress in cutting down the candidate set.

Assuming $\mathcal{C}^{(i-1)}$ is independent of $\mathbf{\Pi}$ (we will argue why this is the case later), we can prove (1) as a direct consequence of Claim 1. In particular, this would imply that $\mathcal{Z}^{(i)}$, $\mathcal{L}^{(i)}$, and $\mathcal{S}^{(i)}$) are independent of $\mathbf{\Pi}$, so for any $\mathbf{z} \in \mathcal{L}^{(i)}$, we have that:

$$\Pr\left[\mathbf{\Pi}\mathbf{z} - \mathbf{\Psi}\mathbf{v}^{(i)} = \mathbf{0}\right] = \Pr\left[\mathbf{\Pi}\mathbf{z} - \mathbf{\Pi}\mathbf{A}\mathbf{v}^{(i)} = \mathbf{0}\right] \leq \frac{1}{2^{3\sqrt{\log|\mathcal{F}|}+2\log(1/\delta)}} = \frac{\delta^2}{2^{3\sqrt{\log|\mathcal{F}|}}}$$

There are at most $2^{\sqrt{\log|\mathcal{F}|}}$ vectors in $\mathcal{L}^{(i)}$, so we conclude via a union bound that:

$$\Pr\left[\mathbf{\Psi}\mathbf{v}^{(i)} \neq \mathbf{\Pi}\mathbf{z} \text{ for all } \mathbf{z} \in \mathcal{L}^{(i)}\right] \geq 1 - \frac{\delta^2 \cdot |\mathcal{L}^{(i)}|}{2^{3\sqrt{\log|\mathcal{F}|}}} \geq 1 - \frac{\delta^2}{2^{2\sqrt{\log|\mathcal{F}|}}}. \tag{2}$$

Case 2: $\mathbf{A}\mathbf{v}^{(i)} \in \mathcal{L}^{(i)}$. In this case, we would like identify $\mathbf{A}\mathbf{v}^{(i)} \in \mathcal{L}^{(i)}$ without issuing a right query, and store its value as $\mathbf{z}_{out}$. We will do so using $\mathbf{\Psi}$. Be definition, $\mathbf{\Psi}\mathbf{v}^{(i)} = \mathbf{\Pi}\mathbf{A}\mathbf{v}^{(i)}$. So, we will only fail to set $\mathbf{z}_{out}$ equal to $\mathbf{A}\mathbf{v}^{(i)}$ on Line 9 if there is some other $\mathbf{z} \in \mathcal{L}^{(i)}$ for which $\mathbf{\Psi}\mathbf{v}^{(i)} = \mathbf{\Pi}\mathbf{z}$. By the same logic as in Case 1, the probability that this happens is low:

$$\Pr\left[\mathbf{\Psi}\mathbf{v}^{(i)} \neq \mathbf{\Pi}\mathbf{z} \text{ for all } \mathbf{z} \in \mathcal{L}^{(i)}, \mathbf{z} \neq \mathbf{A}\mathbf{v}^{(i)}\right] \geq 1 - \frac{\delta^2 \cdot (|\mathcal{L}^{(i)}| - 1)}{2^{3\sqrt{\log|\mathcal{F}|}}} \geq 1 - \frac{\delta^2}{2^{2\sqrt{\log|\mathcal{F}|}}} \tag{3}$$

Correctness. Combining (1) and (3) and observing that the algorithm runs for $\log(|\mathcal{F}|/\delta)$ iterations, we can conclude via a union bound that $\mathbf{z}_{out} = \mathbf{A}\mathbf{v}^{(i)}$ for all i. In particular, with probability at least:

$$1 - \frac{\delta^2 \cdot \log(|\mathcal{F}|/\delta)}{2^{2\sqrt{\log|\mathcal{F}|}}} \geq 1 - \delta^2 \cdot \log(1/\delta) - \frac{\delta^2 \log(|\mathcal{F}|)}{2^{2\sqrt{\log|\mathcal{F}|}}} \geq 1 - \delta.$$

Above we used that $\frac{\log(|\mathcal{F}|)}{2^{2\sqrt{\log|\mathcal{F}|}}} < .5$ for any sized family. Importantly, in applying (1) and (3), we also require that $\mathbf{\Pi}$ is independent of $\mathcal{C}^{(i)}$ for all i. This holds inductively: conditioned on $\mathbf{z}_{out} = \mathbf{A}\mathbf{v}^{(j)}$ for all $j < i$, $\mathcal{C}^{(i)}$ is only a function of $\mathcal{F}$ and $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(j)}$.

Finally we note that, iff $\mathbf{z}_{out} = \mathbf{A}\mathbf{v}^{(i)}$ for all i, then the output of Algorithm 2 is actually identical to that of Algorithm 1, so correctness follows immediately. We are left to confirm query complexity.

Query complexity. We issue $q \leftarrow \lceil 3\sqrt{\log|\mathcal{F}|} + 2\log(1/\delta)\rceil$ left queries to form $\mathbf{\Psi}$. Via the same union bound argument as above, with overall probability $1 - \delta$, we issue a single right query every time we are in Case 1 and we issue not queries when in Case 2. Moreover, it is not hard to see that we can only enter Case 1 at most $\sqrt{\log|\mathcal{F}|}$ times. Indeed, every time we enter Case 1, the size of $\mathcal{C}^{(i)}$ reduces by a multiplicative factor of $1/2^{\sqrt{\log|\mathcal{F}|}}$. If this happened more than $\sqrt{\log|\mathcal{F}|}$ times, we would have an empty candidate set, which is not possible since $\mathbf{A}$ always remains in the candidate set with probability $1 - \delta$.

Our final query complexity is thus:

$$3\sqrt{\log|\mathcal{F}|} + 2\log(1/\delta) + \sqrt{\log|\mathcal{F}|}.$$ □

4 Lower Bounds

It is interesting to ask whether the recovery results from Theorem 3 (for one-sided query access) and Theorem 2 (for two-sided query access) are optimal. In this section, we provide short information theoretical arguments proving that they are up to logarithmic factors. As discussed in Remark 1, to do so, we must assume that all matvec queries to $\mathbf{A}$ or $\mathbf{A}^\mathsf{T}$ only involve inputs, $\mathbf{x}$, whose entries have bounded precision; otherwise, $\mathbf{A}$ can be trivially recovered from any finite family $\mathcal{F}$ using a single query with arbitrary precision.

We note that the assumption of finite precision is both reflective of the sorts of queries and responses that can be computed in applications. Moreover, the assumption can be eliminated when proving lower bounds for the more challenging "approximation" problem (Problem 1), which we will address in forthcoming work.

Formally, a natural restriction is to assume that $\mathbf{x}$ contains integer entries bounded between $[-m, m]$ for some constant m. I.e., its entries can be represented using $\log(2m)$ bits. Since any such $\mathbf{x}$ can be written as a weighted sum of $O(\log m)$ binary $\{0,1\}$ vectors, up to a logarithmic factor in the query complexity, we can equivalently assume binary queries. This is how our theorems are stated below.

Theorem 4. *For any integer n and any $k \leq n$, there exists a finite family $\mathcal{F} \subset \mathbb{R}^{n\times n}$ with $\log|\mathcal{F}| = k$ such that the following is true: Suppose an algorithm issues q adaptive, binary, one-sided queries $\mathbf{x}_1, \ldots, \mathbf{x}_q \in \{0,1\}^n$ to an unknown $\mathbf{A} \in \mathcal{F}$, obtaining responses $\mathbf{A}\mathbf{x}_1, \ldots, \mathbf{A}\mathbf{x}_q \in \{0,1\}^n$. If from these responses, the algorithm returns $\mathbf{A}$ with probability $> 1/2$, then it must be that $q \geq \frac{\log|\mathcal{F}|}{\log\log|\mathcal{F}|}$.*

This theorem establishes that Theorem 3 is optimal, up to a $\log\log|\mathcal{F}|$ factor.

Proof. Let $\mathcal{F}$ contain any $\mathbf{B}$ where the first k elements of the first row are either 0 or 1. All other entries in the first row, and all entries in all other rows are 0. Clearly $|\mathcal{F}| = 2^k$, so $\log|\mathcal{F}| = k$.

Now, consider multiplying any $\mathbf{A} \in \mathcal{F}$ by a binary query vector, $\mathbf{x}$. For any such query, only the first entry of $\mathbf{A}\mathbf{x}$ is non-zero, and this entry will contain an integer between 0 and k. Accordingly, the result of the query only contains $\log k = \log\log|\mathcal{F}|$ bits of information. By pigeonhole principal, there will be more than one matrix in $\mathcal{F}$ that results in the same query responses unless our number of queries q satisfies:

$$q \cdot \log\log|\mathcal{F}| \geq \log|\mathcal{F}| \implies q \geq \frac{\log|\mathcal{F}|}{\log\log|\mathcal{F}|}.$$

Of course, if ≥ 2 matrices result in the same query responses, then any algorithm can at best guess $\mathbf{A}$'s identity with probability $1/2$. □

A similar result holds for the case when the algorithm can make two-sided queries. In particular, the following result shows that our Theorem 2 is tight up to a logarithmic factor.

Theorem 5. *For any integer n and any perfect square $k \leq n$, there exists a finite family $\mathcal{F} \subset \mathbb{R}^{n\times n}$ with $\log|\mathcal{F}| = k$ such that the following is true: Suppose an algorithm issues q adaptive, binary, two-sided queries $\mathbf{x}_1, \ldots, \mathbf{x}_q \in \{0,1\}^n$ to an unknown $\mathbf{A} \in \mathcal{F}$, obtaining responses $\mathbf{A}\mathbf{x}_1, \ldots, \mathbf{A}\mathbf{x}_q \in \{0,1\}^n$ and $\mathbf{A}^\mathsf{T}\mathbf{x}_1, \ldots, \mathbf{A}^\mathsf{T}\mathbf{x}_q \in \{0,1\}^n$. If from these responses, the algorithm returns $\mathbf{A}$ with probability $> 1/2$, then it must be that $q \geq \frac{\sqrt{\log|\mathcal{F}|}}{\log\log|\mathcal{F}|}$.*

Proof. Let $\mathcal{F}$ contain any $\mathbf{B}$ where the upper left $\sqrt{k} \times \sqrt{k}$ entries are either 0 or 1 and all other entries are 0. Here, $\mathcal{F} = 2^k$ and $\log\mathcal{F} = k$. Consider querying an unknown target matrix $\mathbf{A} \in \mathcal{F}$ with a binary vector $\mathbf{x}$ and receiving *both* the left and right response $\mathbf{A}\mathbf{x}$ and $\mathbf{A}^\mathsf{T}\mathbf{x}$. Only the first $\sqrt{k}$ entries in each of these vectors is non-zero, and each non-zero entry contains an integer in the range $[0, \sqrt{k}]$. I.e., any query only returns $2\sqrt{k}\log(\sqrt{k}) = \sqrt{\log|\mathcal{F}|}\log\log|\mathcal{F}|$ bits of information. By the same argument as in Theorem 4, there must exist matrices in $\mathcal{F}$ that return the same set of query responses unless q satisfies the following:

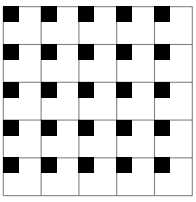
$$q \cdot \sqrt{\log|\mathcal{F}|}\log\log|\mathcal{F}| \geq \log|\mathcal{F}| \implies q \geq \frac{\sqrt{\log|\mathcal{F}|}}{\log\log|\mathcal{F}|}.$$ □

Butterfly matrices. While the family considered in Theorem 5 might seem artificial, the result actually immediately implies a lower bound for the well-studied family of rank-1 butterfly matrices [LY17; Liu+21]. In particular, consider the family $\mathcal{F} \subset \mathbb{R}^{n\times n}$ where only the top-left entry of every block in a partition of the matrix into n, $\sqrt{n} \times \sqrt{n}$ blocks is non-zero. I.e., for any $\mathbf{B} \in \mathcal{F}$:

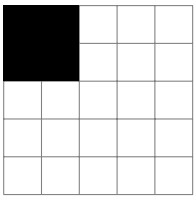
$$B_{ij} = \begin{cases} 0 & \text{if } (i \mod \sqrt{n}) \neq 0 \text{ and } (j \mod \sqrt{n}) \neq 0 \\ 0 \text{ or } 1 & \text{otherwise} \end{cases}.$$

It is not hard to see that, up to a permutation of row and column indices, this family is equivalent to the one considered in Theorem 5, and we thus requires $\tilde{\Omega}(\sqrt{n})$ matvecs queries to recover any $\mathbf{A} \in \mathcal{F}$.

However, we can also observe that any matrix in this family has *complimentary rank*-1 *structure* (see [LY17] for details), and thus admits a rank-1 butterfly factorization. Accordingly, we conclude that recovering a rank-1 butterfly matrix $\mathbf{A}$ from matvec queries also requires $\tilde{\Omega}(\sqrt{n})$ such queries, which matches existing upper bounds up to a $\log n$ factor [Liu+21].



(a) *Only the top-left entries of every block in a* $\sqrt{n} \times \sqrt{n}$ *partition are non-zero.*



(b) *Only the top-left* $\sqrt{n} \times \sqrt{n}$ *entries are non-zero.*

Figure 1: *Matrix with complimentary low rank structure (left) and its permutation (right).*

References

[ADM24] B. Adcock, N. Dexter, and S. Moraga. "Optimal deep learning of holomorphic operators between Banach spaces". In: *Advances in Neural Information Processing Systems 37 (NeurIPS)*. 2024 (cited on page 1).

[AGM25] B. Adcock, M. Griebel, and G. Maier. "Learning Lipschitz Operators with respect to Gaussian Measures with Near-Optimal Sample Complexity". In: (2025) (cited on page 1).

[Amb+20] I. Ambartsumyan, W. Boukaram, T. Bui-Thanh, O. Ghattas, D. Keyes, G. Stadler, G. Turkiyyah, and S. Zampini. "Hierarchical Matrix Approximations of Hessians Arising in Inverse Problems Governed by PDEs". In: *SIAM Journal on Scientific Computing* 42.5 (2020), A3397–A3426 (cited on page 2).

[Ams+24] N. Amsel, T. Chen, D. Halikias, F. D. Keles, C. Musco, and C. Musco. "Fixed-Sparsity Matrix Approximation from Matrix-Vector Products". In: *2402.09379* (2024) (cited on page 3).

[Ams+25a] N. Amsel, T. Chen, D. Halikias, F. D. Keles, C. Musco, and D. Persson. "Quasi-optimal approximation by Hierarchically Semi-Separable matrices". In: *In submission.* (2025) (cited on page 2).

[Ams+25b] N. Amsel, T. Chen, F. D. Keles, D. Halikias, C. Musco, C. Musco, and D. Persson. "Quasi-optimal hierarchically semi-separable matrix approximation". In: *2505.16937* (2025) (cited on page 3).

[Bar+23] I. A. Baratta, J. P. Dean, J. S. Dokken, M. Habera, J. S. Hale, C. N. Richardson, M. E. Rognes, M. W. Scroggs, N. Sime, and G. N. Wells. *DOLFINx: The next generation FEniCS problem solving environment*. 2023 (cited on page 1).

[BCW22] A. Bakshi, K. L. Clarkson, and D. 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, pp. 1130–1143 (cited on page 2).

[BHOT24] N. Boullé, D. Halikias, S. E. Otto, and A. Townsend. "Operator learning without the adjoint". In: *Journal of Machine Learning Research* 25.364 (2024), pp. 1–54 (cited on page 1).

[BHSW20] M. Braverman, E. Hazan, M. Simchowitz, and B. Woodworth. "The Gradient Complexity of Linear Regression". In: *Proceedings of the 33rd Annual Conference on Computational Learning Theory (COLT)*. Vol. 125. 2020, pp. 627–647 (cited on page 2).

[BHT23] N. Boullé, D. Halikias, and A. Townsend. "Elliptic PDE learning is provably data-efficient". In: *Proceedings of the National Academy of Sciences* 120.39 (2023) (cited on pages 1, 2).

[Bin+17] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk. "Data Assimilation in Reduced Modeling". In: *SIAM/ASA Journal on Uncertainty Quantification* 5.1 (2017), pp. 1–29 (cited on page 1).

[BKS07] C. Bekas, E. Kokiopoulou, and Y. Saad. "An estimator for the diagonal of a matrix". In: *Applied Numerical Mathematics* 57.11 (2007), pp. 1214–1229 (cited on page 2).

[BN22] R. A. Baston and Y. Nakatsukasa. *Stochastic diagonal estimation: probabilistic bounds and an improved algorithm*. 2022 (cited on page 2).

[BN23] A. Bakshi and S. Narayanan. "Krylov Methods are (nearly) Optimal for Low-Rank Approximation". In: *Proceedings of the 64th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2023 (cited on page 2).

[BT23] N. Boullé and A. Townsend. "Learning Elliptic Partial Differential Equations with Randomized Linear Algebra". In: *Foundations of Computational Mathematics* 23.2 (2023) (cited on pages 1, 2).

[BT24] N. Boullé and A. Townsend. "A mathematical guide to operator learning". In: *Numerical Analysis Meets Machine Learning*. Handbook of Numerical Analysis. 2024 (cited on page 1).

[CC86] T. F. Coleman and J.-Y. Cai. "The Cyclic Coloring Problem and Estimation of Sparse Hessian Matrices". In: *SIAM Journal on Algebraic Discrete Methods* 7.2 (1986), pp. 221–235 (cited on page 3).

[Che+24] S. Chewi, J. de Dios Pont, J. Li, C. Lu, and S. Narayanan. "Query lower bounds for log-concave sampling". In: *J. ACM* (2024) (cited on page 2).

[Che+25] T. Chen, D. Halikias, F. D. Keles, C. Musco, and D. Persson. "Near-optimal hierarchical matrix approximation from matrix-vector products". In: *SIAM Symp. on Discrete Algo.* 2025 (cited on page 3).

[CM83] T. F. Coleman and J. J. Moré. "Estimation of Sparse Jacobian Matrices and Graph Coloring Blems". In: *SIAM Journal on Numerical Analysis* 20.1 (1983), pp. 187–209 (cited on page 3).

[CPR74] A. R. Curtis, M. J. D. Powell, and J. K. Reid. "On the Estimation of Sparse Jacobian Matrices". In: *IMA Journal of Applied Mathematics* 13.1 (1974), pp. 117–119 (cited on page 3).

[DM23] P. Dharangutte and C. Musco. "A Tight Analysis of Hutchinson's Diagonal Estimator". In: *Symposium on Simplicity in Algorithms (SOSA)*. Society for Industrial and Applied Mathematics, 2023, pp. 353–364 (cited on page 2).

[DSBN15] G. Dasarathy, P. Shah, B. N. Bhaskar, and R. D. Nowak. "Sketching Sparse Matrices, Covariances, and Graphs via Tensor Products". In: *IEEE Transactions on Information Theory* 61.3 (2015), pp. 1373–1388 (cited on page 3).

[DVB15] Y. N. Dauphin, H. d. Vries, and Y. Bengio. "Equilibrated adaptive learning rates for non-convex optimization". In: *Advances in Neural Information Processing Systems 28 (NeurIPS)* (2015) (cited on page 2).

[GFWG10] D. Galbally, K. Fidkowski, K. Willcox, and O. Ghattas. "Non-linear model reduction for uncertainty quantification in large-scale inverse problems". In: *International journal for numerical methods in engineering* 81.12 (2010), pp. 1581–1608 (cited on page 1).

[HMT11] N. Halko, P.-G. Martinsson, and J. A. Tropp. "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions". In: *SIAM Review* 53.2 (2011), pp. 217–288 (cited on page 2).

[HT23] D. Halikias and A. Townsend. "Structured matrix recovery from matrix-vector products". In: *Numerical Linear Algebra with Applications* 31.1 (2023) (cited on pages 2, 3).

[Li+21] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. "Fourier Neural Operator for Parametric Partial Differential Equations". In: *Proceedings of the 9th International Conference on Learning Representations (ICLR)*. 2021 (cited on page 1).

[Liu+21] Y. Liu, X. Xing, H. Guo, E. Michielssen, P. Ghysels, and X. S. Li. "Butterfly Factorization Via Randomized Matrix-Vector Multiplications". In: *SIAM Journal on Scientific Computing* 43.2 (2021) (cited on pages 3, 8, 9).

[LLY11] L. Lin, J. Lu, and L. Ying. "Fast construction of hierarchical matrix representation from matrix–vector multiplication". In: *Journal of Computational Physics* 230.10 (2011), pp. 4071–4087 (cited on pages 2, 3).

[LM24] J. Levitt and P.-G. Martinsson. "Linear-Complexity Black-Box Randomized Compression of Rank-Structured Matrices". In: *SIAM Journal on Scientific Computing* 46.3 (2024), A1747–A1763 (cited on page 3).

[LR10] T. Lassila and G. Rozza. "Parametric free-form shape design with PDE models and reduced basis method". In: *Computer Methods in Applied Mechanics and Engineering* 199.23 (2010), pp. 1583–1592 (cited on page 1).

[Lu+21] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature Machine Intelligence* 3 (2021), pp. 218–229 (cited on page 1).

[LY17] Y. Li and H. Yang. "Interpolative Butterfly Factorization". In: *SIAM Journal on Scientific Computing* 39.2 (2017), A503–A531 (cited on pages 3, 8, 9).

[Mar16] P.-G. Martinsson. "Compressing Rank-Structured Matrices via Randomized Sampling". In: *SIAM Journal on Scientific Computing* 38.4 (2016), A1959–A1986 (cited on pages 2, 3).

[MM15] C. Musco and C. Musco. "Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition". In: *Advances in Neural Information Processing Systems 28 (NeurIPS)*. 2015, pp. 1396–1404 (cited on page 2).

[MMMW21] R. A. Meyer, C. Musco, C. Musco, and D. Woodruff. "Hutch++: Optimal Stochastic Trace Estimation". In: *Proceedings of the 4th Symposium on Simplicity in Algorithms (SOSA)* (2021) (cited on page 2).

[Pea94] B. A. Pearlmutter. "Fast exact multiplication by the Hessian". In: *Neural computation* 6.1 (1994), pp. 147–160 (cited on page 2).

[PN24] T. Park and Y. Nakatsukasa. "Approximating Sparse Matrices and their Functions using Matrix-vector products". In: *2310.05625* (2024) (cited on page 3).

[RST09] V. Rokhlin, A. Szlam, and M. Tygert. "A Randomized Algorithm for Principal Component Analysis". In: *SIAM Journal on Matrix Analysis and Applications* 31.3 (2009), pp. 1100–1124 (cited on page 2).

[SER18] M. Simchowitz, A. El Alaoui, and B. 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, pp. 1249–1259 (cited on page 2).

[Set09] B. Settles. *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison, 2009 (cited on page 3).

[SO24] F. Schäfer and H. Owhadi. "Sparse Recovery of Elliptic Solvers from Matrix-Vector Products". In: *SIAM Journal on Scientific Computing* 46.2 (2024), A998–A1025 (cited on pages 2, 3).

[TS11] J. M. Tang and Y. Saad. "Domain-Decomposition-Type Methods for Computing the Diagonal of a Matrix Inverse". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2823–2847 (cited on page 2).

[WEV13] T. Wimalajeewa, Y. C. Eldar, and P. K. Varshney. "Recovery of sparse matrices via matrix sketching". In: *1311.2448* (2013) (cited on page 3).

[Woo14] D. P. Woodruff. "Sketching as a Tool for Numerical Linear Algebra". In: *CoRR* abs/1411.4357 (2014) (cited on page 2).

[WSB11] A. Waters, A. Sankaranarayanan, and R. Baraniuk. "SpaRCS: Recovering low-rank and sparse matrices from compressive measurements". In: *Advances in Neural Information Processing Systems 24 (NeurIPS)*. Vol. 24. 2011 (cited on page 3).

[WT23] C. Wang and A. Townsend. "Operator learning for hyperbolic partial differential equations". In: *2312.17489* (2023) (cited on page 2).

[YGKM20] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney. "PyHessian: Neural Networks Through the Lens of the Hessian". In: *2020 IEEE International Conference on Big Data (Big Data)*. 2020, pp. 581–590 (cited on page 2).

[ZJD15] K. Zhong, P. Jain, and I. S. Dhillon. "Efficient Matrix Sensing Using Rank-1 Gaussian Measurements". In: *Proceedings of the 26th International Conference on Algorithmic Learning Theory*. 2015, pp. 3–18 (cited on page 3).