# STRUCTURED MATRIX RECOVERY AND APPROXIMATION FROM MATRIX-VECTOR PRODUCTS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Diana Halikias

May 2025

# STRUCTURED MATRIX RECOVERY AND APPROXIMATION FROM MATRIX-VECTOR PRODUCTS

Diana Halikias, Ph.D.

Cornell University 2025

Algorithms and theory are developed for the task of recovering a structured, unknown matrix $A$ from matrix-vector products $x \mapsto Ax$ and $y \mapsto A^\top y$. Upper and lower bounds on the number of matrix-vector products necessary to recover many common structured matrix families, such as tridiagonal, Toeplitz, Hankel, low-rank, hierarchical, and sparse matrices, are provided. For sparse and hierarchical matrices, the more general problem of near-optimal approximation from matrix-vector products is analyzed. This research is motivated by the field of operator learning, where one wishes to recover an infinite-dimensional operator from only operator-function products. The problem of matrix recovery and approximation also encompasses important current research directions in numerical linear algebra, such as low-rank approximation and diagonal estimation.

Algorithms and theory for hierarchical matrix recovery are extended to infinite dimensions for recovery of the Green's function of a uniformly elliptic PDE over a 3D domain with Lipschitz-smooth boundary subject to Dirichlet boundary conditions. We prove that one can accurately approximate the Green's function with relatively few input-output pairs of forcing terms and solutions by exploiting hierarchical low-rank structure and PDE theory.

Throughout the work, it is a central question whether one can prove an accuracy guarantee for peeling algorithms, which recover hierarchical matrices from matrix-vector products using a recursive strategy that may result in a large accumulation

of error. Upper bounds on the error of peeling-type algorithms are provided in both the finite and infinite-dimensional settings. In infinite dimensions, a combination of Green's function theory for elliptic PDEs and an adaptive adjustment of the target rank at each hierarchical level are needed. In finite dimensions, the choice of low-rank approximation algorithm is shown to be crucial, and a perforated sketch structure based on CountSketch is also employed. In each setting, perturbation analysis for low-rank approximation techniques such as the randomized SVD and generalized Nyström method is provided. The first theoretical guarantees for the accuracy of peeling algorithms are presented, as well as the existence of hard cases for which these algorithms accumulate uncontrollable error.

Lastly, motivated by realistic settings in PDE learning and numerical linear algebra, the assumption of access to the action of the transpose of a matrix, $y \mapsto A^\top y$, and adjoint of an operator is removed. Transpose-free techniques for matrix recovery and least squares problems are investigated, and transpose data is shown to be essential to each of these problems.

## BIOGRAPHICAL SKETCH

Diana Halikias earned her undergraduate degree in mathematics from Yale University in 2020. In August of 2020, she went on to pursue a PhD in mathematics at Cornell University under the supervision of Alex Townsend. In the fall of 2025, she will be a Berkeley-Simons visiting postdoctoral fellow at the Simons Institute for the Theory of Computing in Berkeley, California. She will then start as a Courant Instructor at New York University in January of 2026.

Dedicated to my mom, Michelle Chua.

## ACKNOWLEDGEMENTS

Many important individuals have not only made my academic career possible, but also left an indelible mark on me. In the first math class I ever took as an undergraduate at Yale, Stefan Steinerberger was a dynamic force of a professor who transformed the way I perceive mathematics. He told us that Math 230 would make us suffer (and suffer we did!), but it took me years to realize that he meant that in a good way, and that the some of the best ideas are the ones worth suffering over. Asher Auel taught me, among many other things, that a math lecture is a true art form and a deep proof can be read like a novel. I want to thank him for his kindness during my undergraduate years and for the old window AC unit he gave us during a scalding summer of math research in New Haven. I am also thankful for the wonderful teaching of Dan Spielman and Ronald Coifman, as well as Nick Marshall's guidance through my first math research project.

Above all else, I am grateful to my advisor, Alex Townsend, for his unwavering encouragement and wisdom over the last five years. Alex's excitement for mathematics is infectious, and his brilliance is inspiring. I am honored to have worked with him so closely during my PhD, to gradually learn to speak his language and inherit his research style. Alex sees so much meaning in a mathematical statement and has a boundless ability to draw analogies and make connections. He bridges many academic worlds, promotes open collaborations, and discovers fascinating problems everywhere. He is also a fantastic mentor, setting a high standard for excellence that is a vote of confidence in his students. From the very start, Alex has believed in me, even and especially in moments when I did not believe in myself. I eagerly anticipate our weekly meetings and keep a running list of all the ideas I want to share with him. I couldn't have asked for a better advisor, and I desperately hope that one day I can offer the same quality of mentorship to others.

I have benefited immensely from the support of my collaborators. Nicolas Boullé and Sam Otto are insightful, caring colleagues whom I am lucky to interact with every week. I cherish the rare occasions where we all get to be in the same place at once, and somehow it feels like we can make many months' progress in just a few days whenever this happens. I also want to thank Tyler Chen for being an excellent and kind collaborator and for bringing me into the orbit of many other wonderful researchers at NYU. It has been especially eye-opening to learn from theoretical computer scientists there. I have really appreciated the opportunity to work with Noah Amsel, Feyza Duman-Keles, Cameron Musco, and David Persson. I especially want to thank Chris Musco for welcoming me into his research circle; I am excited to keep working together as I join Courant next year!

Ever since the first conference I attended, I have known that the numerical linear algebra community is unique. I want to express thanks for the many fruitful (academic and non-academic) conversations that I've had with Ben Erichson, Andrew Horning, Lorenzo Lazzarino, Michael Mahoney, Maike Meier, Dmitriy Morozov, Yuji Nakatsukasa, Taejun Park, Lawrence Saul, Tianyi Shi, Chris Wang, Heather Wilber, Annan Yu, and Jennifer Zvonek. I am also grateful for the feedback I received from my committee members, Anil Damle and Laurent Saloff-Coste.

I am also indebted to my wonderful community in Ithaca for making graduate school so enjoyable. Thank you to Colby for being my first and forever friend here, I wouldn't have gotten through algebraic topology or anything else without you. Thank you to Nicki for all the Pomodoros, walks to Aldi, and silly, happy moments. Chase, thank you for being always down. Ben, I can't believe we only found each other in your last semester in Ithaca, but I am comforted by the fact that we will be friends for a long time. I'm so grateful for the joyful memories I've shared with Alexia, Annika, Claire, Kelley, Sara, Rodrigo, Juno, and Zola, too. I

also want to recognize the Big Red Barn and Lindseth climbing center, which kept me remarkably sane all these years.

Finally, I am so grateful for my family and friends, who have given me endless support in my math career. Sonali, thank you for introducing yourself to me in Math 230 office hours, for toiling over problem sets with me all night for so many nights at Yale, and for looking up from our homework to watch the sunrise together in the Morse common room. I am certain I would not have gone to graduate school if not for you. Alex, Julia, and Margo, you have always rooted for me and listened eagerly to my crazy math stories. I am proud to have grown up with you, and I am confident that there is no other lawyer-art historian-chemist team that collectively knows so much about the linear algebra world!

To Noah: thank you for being there for me through all of it. We are lucky to share so much, and yet at the same time, I am always learning from you. Lastly, I want to express deep gratitude to my grandparents, parents, and siblings. My grandfather is the quintessential mad scientist who has always inspired me ask questions and probe the unknown. When I was growing up, my mother constantly encouraged me to do math and always pushed me to do more, on weekends and in the summers. Those were some of the most impactful educational experiences I have had, and I can't thank her enough.

During my PhD, I was extremely fortunate to be supported by the NSF-GRFP (DGE – 2139899), the ONR (N00014-23-1-2729), and the Cornell Mathematics First Year Fellowship.

## TABLE OF CONTENTS

CHAPTER 1

# INTRODUCTION

This thesis addresses the following problem: given a structured, unknown matrix $A \in \mathbb{R}^{N \times N}$ that one can only query with matrix-vector products $x \mapsto Ax$ and $y \mapsto A^\top y$, how can one approximate $A$ using as few queries as possible? This question is fundamental to the matrix-free model of computation, where one solves linear algebra problems using only matrix-vector products with a matrix, as opposed to accessing all of its entries. Matrix-free methods are vital for several reasons. First, in problems involving vast amounts of data (e.g., representing pixels of a high-resolution image or atoms in electronic structure calculations), matrices of interest may be too large to store explicitly. Matrix-vector products enable us to still work with this data in an efficient manner. Secondly, there is an arsenal of fast matrix-vector multiplication algorithms which one can exploit in matrix-free methods. Finally, one may have implicit access to matrix-vector products, even when the underlying matrix is unknown. For example, one can simulate queries with a discretized Green's function using experimental data, or implicitly query the Hessian matrix of a neural network using backpropagation.

The matrix-free model encompasses the highly successful Krylov methods, which query $Ab, A^2b, A^3b, \ldots$ to solve linear systems and compute eigenvalues and eigenvectors. Recently, randomized matrix-vector products, or "sketches," constitute an increasingly popular tool for reducing the dimensionality of a matrix, enabling fast algorithms for least-squares problems and low-rank approximation. We examine the "complexity" of common structured matrices in the matrix-free model by investigating the number of queries needed for recovery. We also explore the impact of adaptivity and randomness in the inputs on these problems. Our

work synthesizes two aspects of ongoing research in numerical linear algebra: the rise of the matrix-free model of computation and the prevalence of structured matrices (e.g., diagonal, Toeplitz, and low-rank) in applications. We also open the door to theory for operator learning, a growing area of scientific machine learning.

We describe practical recovery algorithms, as well as establish theoretical upper and lower bounds on the number of sketches needed to recover common structured matrices. We pay special attention to low-rank and so-called hierarchical low-rank structure, which frequently arise in the solution of partial differential equations (PDEs) and computational science. The chief goal of this work is to establish a cohesive framework for matrix recovery, which encompasses popular matrix-free tools for low-rank approximation, trace and diagonal estimation, testing for properties like symmetry and rank, and graph problems. Each chapter contains novel contributions to numerical linear algebra and applications to operator learning.

In Chapter 2, we consider the problems of *exact matrix recovery* and *matrix approximation* from matrix-vector products. Structured approximations are ubiquitous in numerical linear algebra, and we consider various common structures such as symmetric, low-rank, Toeplitz, Hankel, circulant, and sparse. Exact matrix recovery refers to recovering the entries of a matrix in exact arithmetic when one knows its structure in advance. Matrix approximation refers to the more general problem where one finds a structured approximation to a general matrix. In the exact setting, our lower bounds indicate that low-rank approximation algorithms like the randomized SVD and generalized Nyström method are near-optimal in the sense of matrix-vector products. We also highlight some surprising matrix classes, such as symmetric and orthogonal, which, despite their structure, still require the maximum possible number of matrix-vector products for recovery.

Using ideas from compressed sensing, we propose an algorithm and derive an accompanying accuracy guarantee for near-optimal sparse approximation of a general matrix, when the sparsity pattern is known in advance. We also derive a general lower bound that applies even to the well-studied case of best approximation by a diagonal matrix, and more broadly, to approximation by a banded matrix. To the best of our knowledge, our lower bounds are the first of their kind (adaptive or non-adaptive), even for these special cases.

In Chapter 3, we address the recovery and approximation problems for hierarchical matrices, structured low-rank matrices which serve as useful compressed formats that capture long-range, smooth kernel interactions arising in PDEs. Many fast algorithms leverage hierarchical structure for factorizations, inversion, and the solution of least squares and eigenvalue problems. In this realm, a key technique is peeling-type algorithms, which compress a general matrix into a hierarchical format using matrix-vector products. While peeling algorithms are observed to work well in practice, they recover the entries of a matrix recursively and in a predetermined order, which suggests that the approximation error may accumulate exponentially. Thus, we propose a variant of peeling which uses a recursive projection procedure, as opposed to recursive elimination, analogous to the difference between the QR and LU factorizations. We also prove the first near-optimal error guarantee and derive the first hard cases for peeling-type algorithms. We demonstrate that the generalized Nyström method damps the error in peeling, whereas the randomized SVD can propagate uncontrolled exponential error in the approximation.

The second half of this thesis focuses on recovery problems in the operator learning setting. In Chapter 4, we develop an infinite-dimensional version of the peeling algorithm to approximate the Green's function associated with a uniformly elliptic

PDE in 3 dimensions defined on a domain with Lipschitz smooth boundary subject to Dirichlet conditions. We prove that our algorithm achieves $\varepsilon$-accuracy with high probability using only $\mathcal{O}(\log^5(1/\varepsilon))$ operator-function products, i.e., pairs of forcing terms and solutions. Our proof exploits a generalization of the randomized SVD to Hilbert–Schmidt operators, the regularity of Green's functions, and a careful analysis of the peeling algorithm at each level in the hierarchy. We thus upper bound the sample complexity of the problem, providing a theoretical explanation for the so-called data-efficient phenomenon, where operator neural networks require little training data to "learn" the solution operator of an elliptic PDE.

In the operator learning setting, it is highly impractical to assume access to adjoint-operator-function products (the analogue of transpose-matrix-vector products). This motivates the subject of Chapter 5, where we provide an in-depth analysis of the problem of transpose-free low-rank recovery. We show that without prior information about the right singular vectors, it is practically impossible to efficiently recover a low-rank matrix from only forward matrix-vector products. We prove rigorous guarantees on the quality of the low-rank reconstruction in terms of how close the matrix is to symmetric. This idea mirrors results in operator learning, which show that one can exploit regularity results from PDE theory to estimate the range of the adjoint of the solution operator. Thus, prior information is needed to guarantee the accuracy of an adjoint-free approximation. We also demonstrate that the transpose is essential for sketched least-squares problems.

## 1.1 Query complexity

Suppose that there is an unknown structured matrix $A \in \mathbb{R}^{N \times N}$ that one can only access via the matrix-vector product operations $x \mapsto Ax$ and $x \mapsto A^\top x$, where $A^\top$ is the transpose of $A$. We denote the number of matrix-vector product queries made to $A$ and $A^\top$ by $s_R$ and $s_L$, respectively. How can one recover $A$ while minimizing the total number of matrix-vector product queries $s = s_L + s_R$?

Any matrix can be recovered in at most $N$ queries, as it can be recovered column-by-column with $Ae_j$ for $1 \leq j \leq N$, where $e_j$ is the $j$th canonical unit vector. However, if $A$ is known to have a structure such as tridiagonal, symmetric, orthogonal, Toeplitz-like, or hierarchical low-rank, one hopes to recover $A$ in fewer queries. To phrase our question more formally, we introduce the definition of query complexity, borrowing terminology from a survey of a more general problem [158].

**Definition 1.1.1.** *Suppose $\mathcal{A} \subset \mathbb{R}^{N \times N}$ is a family of structured matrices. Given matrices $X \in \mathbb{R}^{N \times k_1}$ and $Y \in \mathbb{R}^{N \times k_2}$, we define the maps $f_X : \mathcal{A} \to \mathbb{R}^{N \times k_1}$ and $g_Y : \mathcal{A} \to \mathbb{R}^{N \times k_2}$ as $f_X(A) = AX$ and $g_Y(A) = A^\top Y$. Then, the query complexity $QC(\mathcal{A})$ is equal to $s$ if $s$ is the smallest natural number for which there exists an $0 \leq s_R \leq s$ and $s_L = s - s_R$ and there exist input matrices $X \in \mathbb{R}^{N \times s_R}, Y \in \mathbb{R}^{N \times s_L}$ such that the maps $f_X$ and $g_Y$ are injective.*

Informally, the query complexity is the maximum over all matrices $A \in \mathcal{A}$ of the smallest total number of queries $s = s_L + s_R$ to $A$ and $A^\top$ needed to uniquely recover $A$ within $\mathcal{A}$. In particular, there is no other matrix $A' \in \mathcal{A}$ satisfying the same $s_R$ matrix-vector products with $A$ and $s_L$ matrix-vector products with $A^\top$.

Query complexity is a measure of the information or complexity of a family of

structured matrices from the matrix-vector product perspective. This is a natural notion due to both the highly efficient matrix-free model of computation, as well as the prevalence of structure in many applications in data science and scientific computing. Investigating the query complexity of different matrices yields efficient, matrix-free algorithms that exploit structure, as well as a theoretical understanding of the limits of these techniques.

In the present work, we address the following questions: given a particular structured family of matrices $\mathcal{A}$, what is $\mathrm{QC}(\mathcal{A})$? If we know in advance the structure of $\mathcal{A}$, can we devise a practical algorithm that recovers any matrix $A \in \mathcal{A}$ in $\mathrm{QC}(\mathcal{A})$ queries? The column-by-column approach yields the upper bound $\mathrm{QC}(\mathcal{A}) \leq N$ for any family $\mathcal{A}$ of $N \times N$ matrices.

There are two main types of vector inputs in matrix recovery problems: (1) Predetermined input vectors, where one tries to recover $A$ from given matrix-vector product pairs $y_1 = Au_1, \ldots, y_{s_R} = Au_{s_R}$ and $z_1 = A^\top v_1, \ldots, z_{s_L} = A^\top v_{s_L}$,[1] and (2) Algorithmically-determined input vectors, where an algorithm can select the $u_j$'s and $v_j$'s used in the queries $Au_1, \ldots, Au_{s_R}, A^\top v_1, \ldots, A^\top v_{s_L}$. We also distinguish between adaptive and non-adaptive queries. Adaptive inputs $u_k$ and $v_k$ may depend on $Au_1, \ldots, Au_{k-1}$ and $Av_1, \ldots, Av_{k-1}$. Finally, we consider both deterministic and randomly generated input vectors. Importantly, Definition 1.1.1 does not distinguish between different types of input vectors, so that query complexity describes even the "best possible" inputs for a given recovery problem. We provide a summary of recovery algorithms developed in this work in Table 1.1.

There are several existing approaches for matrix recovery problems from matrix-vector products. If we know in advance that a matrix is well-approximated

[1]This is equivalent to trying to simultaneously solve matrix equations of the form $Y = AU$ and $Z = A^\top V$ for a structured matrix $A$.

| Structure | $\#x \mapsto Ax$ | $\#x \mapsto A^\top x$ |
|---|---|---|
| diagonal‡ | 1 | - |
| block-$k$ diagonal‡ | $k$ | 0 |
| tridiagonal‡ | 3 | 0 |
| symm. tridiag.‡ | 2 | - |
| rank-$k$ | $k+p$ | $k$ |
| symm. rank-$k$ | $k+p$ | - |
| circulant‡ | 1 | 0 |
| circulant-plus-diagonal‡ | 2 | 0 |
| Toeplitz or Hankel‡ | 2 | 0 |
| symmetric‡ | $N$ | - |
| orthogonal‡ | $N$ | 0 |
| Toeplitz-like | $2p+4$ | $5p+8$ |
| p-HSS rank-$k$ | $6k+2p$ | $2k$ |
| symm. p-HSS rank-$k$ | $5k+p$ | - |
| HODLR rank-$k$ | $(6k+4p)\lceil\log_2(N)\rceil$ | $4k\lceil\log_2(N)\rceil$ |
| symm. HODLR rank-$k$ | $(6k+2p)\lceil\log_2(N)\rceil$ | - |

Table 1.1: A summary of deterministic and randomized matrix recovery algorithms for structured matrices from matrix-vector products. There exists a randomized algorithm for all structures, and for some structures, randomization is essential. Each of the randomized algorithms performs exact recovery with a success probability of 1. In the table, ‡ denotes the existence of a deterministic algorithm. The oversampling parameter $p$ can be though to of as a constant, e.g., $p = 5$, and is needed for recovery algorithms using randomized linear algebra. For HSS and HODLR matrices, $\ell$ is such that the diagonal blocks are size $2^\ell \times 2^\ell$ (see Sections 3.1.1 and 3.1.4). The number of queries listed in the table is biased toward using matvecs with $A$ over $A^\top$ whenever possible.

by a rank-$k$ matrix, the randomized singular value decomposition (SVD) [85, 120] selects random Gaussian input vectors and can stably recover it with high probability $1 - 6p^{-p}$ using $m = k + p$ and $n = k + p$, where $p$ is a small fixed constant, i.e., $p = 5$. One can also use the generalized Nyström algorithm to stably recover such a matrix with $m = k + p$ and $n = 2m + p$ [130, 167]. Peeling algorithms use structured random vectors and a recursive elimination procedure to recover hierarchical low-rank matrices [111, 117, 106]. Moreover, sparse matrices with columns that have a disjoint sparsity pattern can be recovered with one query [151, Fig. 1], which leads to an algorithm to recover positive semidefinite hierarchical matrices.

Another family of hierarchical matrices, hierarchical semiseparable (HSS) matrices can be recovered with a linear-complexity algorithm that exploits a telescoping factorization [105].

If one does not know if an unknown matrix $A$ is structured or not, then there are algorithms for testing if a matrix has a particular property using matrix-vector products. Property detection is often easier than matrix recovery, requiring far fewer queries. In particular, only $\mathcal{O}(1)$ queries are required to determine with high probability if a matrix is diagonal or symmetric, and precisely 1 query is necessary to determine if a matrix is orthogonal [158].

Our motivation for matrix recovery from matrix-vector products arises from partial differential equation (PDE) learning [35]. In that setting, one selects forcing terms $f_1, \ldots, f_{s_R}$ of a PDE and then observes the corresponding solutions $u_1, \ldots, u_{s_R}$. The goal is to learn the solution operator that maps forcing terms to responses, given training data $\{(f_j, u_j)\}_{j=1}^{s_R}$ [30, 73, 99, 108, 109, 113, 172]. For the case of an elliptic or parabolic linear PDE, the solution operator can be represented as an integral operator, and we seek its Green's function kernel [35, 33]. The discrete version of Green's function recovery is hierarchical low-rank matrix recovery from matrix-vector products. For variable coefficient elliptic PDEs, the discretized Green's function resembles a so-called hierarchical off-diagonal low-rank (HODLR) matrix (see Figure 1.2), as its off-diagonal blocks have rapidly decaying singular values [18]. For constant coefficient elliptic PDEs, the discrete analogue is the recovery of a special type of HODLR matrix, the so-called hierarchical semiseparable matrices, sometimes also called hierarchical block separable (HBS) matrices (see Section 3.1.1). In this paper, we use the HSS notation.

There are other emerging applications of matrix recovery from matrix-vector

products, including the computation of matrix functions, i.e., $f(A)$, where $f(A)$ is structured, from the matrix-vector products $x \mapsto f(A)x$ [138]. Other settings include when $A$ is the Hessian of an optimization problem that can be applied efficiently to vectors (e.g., via back-propagation for neural network loss functions in machine learning) [141, 88] or a data matrix in compressed sensing [93].

## 1.2 Operator learning

Operator learning is an emerging field that combines physics and machine learning to recover information about physical systems from data. In this thesis, we consider partial differential equations over a bounded domain $\mathcal{D} \subset \mathbb{R}^d$ with Lipschitz-smooth boundary of the following form:

$$\mathcal{L}u = f$$

subject to zero Dirichlet boundary conditions, where $\mathcal{L} : \mathcal{H}^2(\mathcal{D}) \cap \mathcal{H}_0^1(\mathcal{D}) \to L^2(\mathcal{D})$, is a linear partial differential operator, $u : \mathcal{D} \to \mathbb{R}$ is a solution, and $f : \mathcal{D} \to \mathbb{R}$ is a forcing term. Here, $\mathcal{H}^s$ is the $s$th Sobolev space, $\mathcal{H}_0^s$ is the closure of the infinitely differentiable functions compactly supported in $\Omega$ in $\mathcal{H}^s(\mathcal{D})$, and $L^2(\mathcal{D})$ is the set of square-integrable functions over $\mathcal{D}$. More specifically, we consider the family of elliptic PDEs, which take the divergence form

$$\mathcal{L}u = -\nabla \cdot (A(x)\nabla u),$$

where $A(x) \in \mathbb{R}^{d \times d}$ is a symmetric positive definite matrix for every $x \in \mathcal{D}$ with $\kappa_C = \sup\{\lambda_{\max}(x)/\lambda_{\min}(x) | x \in \mathcal{D}\} < \infty$, and $A$ has bounded coefficients, i.e., $A_{ij} \in L^\infty(\mathcal{D})$ for $1 \leq i, j \leq d$. Here, $\lambda_{\max}$ and $\lambda_{\min}$ denote the smallest and largest eigenvalues of $A(x)$.

In operator learning, in contrast to the related problems of PDE discovery and solving [36], one accesses input-output pairs of forcing terms and solutions $\{(f_i, u_i)\}_{i=1}^{s_R}$ to approximate the solution operator, which maps forcing terms to solutions [30, 73, 99, 108, 109, 113, 172]. To this end, inspired by recent progress in machine learning, neural operators have been developed to learn mappings between infinite-dimensional spaces [99].

There are many motivations for operator learning. When nothing is known about a physical system of interest, one hopes to uncover unknown physics in the form of dynamical systems or PDEs [113]. These techniques have succeeded in revealing important physical insights, such as symmetry and singularities of the operator [30]. Even when the PDE is already known, operator learning can be used to build reduced order models and speed up existing numerical solvers. This has been used in the place of traditional numerical methods when very fine discretizations are needed, or when one needs to cheaply evaluate the solution operator many times. Already, operator learning has achieved some success in modeling turbulence flows [108], aiding diffusion model sampling [188], solving inverse problems [126], and surrogate modeling for uncertainty quantification [163].

While the exciting recent advances in operator learning signal great promise for the field of scientific machine learning, practice has rapidly outpaced theory. In particular, data generation of pairs $(f, u)$ can be prohibitively expensive, as it requires running a costly experiment or simulation. The training time and memory-intensive nature of deep learning models is also something to keep in mind when comparing against more classical numerical methods. For this reason, there is an urgent need to theoretically quantify the type and amount of data needed for PDE learning to achieve high accuracy. This question is closely related to that of query

complexity for matrix recovery from matrix-vector products. To see why, we state the problem of PDE learning in terms of Green's function recovery. Given an $\mathcal{L}$ under our assumptions, a Green's function $G : \Omega \times \Omega \to \mathbb{R} \cup \{\infty\}$ is a kernel operator which can be uniquely associated with a PDE. In particular, $G$ satisfies

$$u(x) = \int_{\mathcal{D}} G(x, y) f(y) dy \quad \text{for all } x \in \mathcal{D}. \tag{1.1}$$

Instead of learning the solution operator from data, one may equivalently recover the Green's function kernel $G$ in Equation (1.1) [35, 33]. For $d \leq 3$, $G$ is square-integrable, and the discrete analogue of Green's function recovery from operator-function products satisfying Equation (1.1) is matrix recovery from matrix-vector products. Moreover, any knowledge about $G$ can be leveraged to recover the corresponding matrix using as few queries as possible. For example, it is known that $G$ admits a separable expansion analogous to low-rank factorization over well-separated subdomains [18], and a similar result has been shown in the case of time-dependent parabolic PDEs [33]. Prior knowledge about the structure of $G$ can be exploited in both practical operator learning architectures for PDE learning [30], as well as for deriving sample complexity guarantees for the problem at hand [35, 32]. This thesis is largely motivated by the latter question.

## 1.3 Low-rank matrices

Low-rank matrices are pervasive throughout computational mathematics, machine learning, and statistics [169]. Often, one wishes to find the best rank-$k$ approximation to a general matrix $A \in \mathbb{R}^{M \times N}$ (denoted as $A_k$), which is equivalent to solving the following optimization:

$$A_k = \underset{B \in \mathbb{R}^{N \times N}}{\operatorname{argmin}} \|A - B\|_F \quad \text{where} \quad \operatorname{rank}(B) = k. \tag{1.2}$$

Here, and throughout the rest of this work, $\|\cdot\|_F$ denotes the Frobenius norm. One may also solve the optimization Equation (1.2) in the spectral norm $\|\cdot\|_2$, i.e., the largest singular value of a matrix. Given the singular value decomposition (SVD) of $A$, the Eckart–Young theorem provides $A_k$:

**Theorem 1.3.1** (Eckart–Young)**.** *Given $A \in \mathbb{R}^{N\times N}$, write $A$'s SVD as $A = U\Sigma V^\top$. Here, $U \in \mathbb{R}^{N\times N}$ and $V \in \mathbb{R}^{N\times N}$ are orthogonal matrices of left and right singular vectors respectively, and $\Sigma \in \mathbb{R}^{N\times N}$ is a diagonal matrix whose $i$th entry is $A$'s singular values $\sigma_i$ in non-increasing order. Then, the best rank-$k$ approximation $A_k$ which solves Equation* (1.2) *is the matrix given by the truncated SVD:*

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^\top,$$

*where $u_i$ and $v_i$ are the $i$th columns of $U$ and $V$, respectively, and the error of the approximation is given by*

$$\|A - A_k\|_F = \left( \sum_{i=k+1}^{N} \sigma_i(A)^2 \right)^{1/2}. \tag{1.3}$$

Thus, computing the SVD of a matrix is a way to obtain its low-rank approximations. Because computing the SVD can be computationally expensive for large $N$, there is a plethora of low-rank approximation techniques for finding a near-optimal approximation without having to compute the entire SVD, e.g., by instead using random test vectors [120], such as random perturbations [5], sparse sign matrices [50, 122, 133, 170], and subsampled trigonometric transforms [4, 5, 139, 181]. By near-optimal approximation, we mean an approximation which achieves an error slightly larger than that of the optimal approximation, (say, up to multiplication by a small factor). In the present work, we focus on two particular low-rank approximation techniques: the randomized SVD (RSVD) [85] and the generalized Nyström Method [49, 167, 130], summarized below for a general matrix $B \in \mathbb{R}^{N\times N}$.

We emphasize that these are descriptions of the algorithms in the exact mathematical form that facilitates our analysis, not necessarily descriptions of stable implementations. For discussions of stable implementations, see [85, 167, 130].

**Algorithm.** Randomized SVD

**procedure** RSVD($B, k, s_{\mathrm{R}}$)
  $\Omega \in \mathbb{R}^{N \times s_{\mathrm{R}}}$, $\Omega_{ij} \sim \mathcal{N}(0,1)$ i.i.d. for all $1 \le i \le N, 1 \le j \le s_{\mathrm{R}}$
  $Q = \mathrm{orth}(B\Omega)$
  $X = Q^\top B$ ▷ $\mathrm{argmin}_Z \|B - QZ\|_{\mathsf{F}}$
  **return** $QX$
**end procedure**

**Algorithm.** Generalized Nyström Method

**procedure** GNM($B, k, s_{\mathrm{R}}, s_{\mathrm{L}}$)
  $\Omega \in \mathbb{R}^{N \times s_{\mathrm{R}}}$, $\Omega_{ij} \sim \mathcal{N}(0,1)$ i.i.d. for all $1 \le i \le N, 1 \le j \le s_{\mathrm{R}}$
  $\Psi \in \mathbb{R}^{N \times s_{\mathrm{L}}}$, $\Psi_{ij} \sim \mathcal{N}(0,1)$ i.i.d. for all $1 \le i \le N, 1 \le j \le s_{\mathrm{L}}$
  $Q = \mathrm{orth}(B\Omega)$
  $X = (\Psi^\top Q)^\dagger \Psi^\top B$ ▷ $\mathrm{argmin}_Z \|\Psi^\top B - \Psi^\top QZ\|_{\mathsf{F}}$
  **return** $QX$
**end procedure**

Here, the subroutine orth() takes in a matrix and returns an orthonormal basis for its column space. We can take this operation to return the $Q$ factor of a column-pivoted QR decomposition. Both methods first compute an orthonormal basis $Q$ for the column span of the matrix $B\Omega$, formed by multiplying $B$ by a random sketching matrix $\Omega$ with $s_{\mathrm{R}}$ columns. Throughout, we take $\Omega$ to be Gaussian for simplicity, although most other popular sketching distributions, such as Rademacher, sparse maps, and subsampled randomized trigonometric transforms, can also be employed [120]. While other distributions may be even faster to work with, we use the explicit bounds developed for Gaussian matrices to rely on existing

theory for both the randomized SVD and the generalized Nyström method.

RSVD then projects $B$ onto this column span and forms the best rank-$k$ approximation of the result. Generalized Nyström follows the same idea, but instead computes an approximate projection of $B$ onto $Q$ by using a second sketch $\Psi$ on the left. In particular, the generalized Nyström returns the rank-$k$ approximation

$$A\Omega(\Psi^\top A\Omega)^\dagger \Psi^\top A, \tag{1.4}$$

where $\dagger$ denotes the pseudo-inverse. In contrast to the randomized SVD, the generalized Nyström method requires only one pass over the matrix and therefore uses non-adaptive sketches. In the case that $A$ is symmetric positive definite, the generalized Nyström method tends to be preferable to the randomized SVD because it naturally preserves symmetry and is faster. Additionally, a variant of the generalized Nyström method has been introduced to overcome instability when $A$ is symmetric, but possibly indefinite [132].

If $\operatorname{rank}(B) \leq k$, RSVD and generalized Nyström both recover $B$ exactly (with probability one) if, respectively, $s_{\mathrm{R}} \geq k$ or $s_{\mathrm{R}}, s_{\mathrm{L}} \geq k$. More generally, these methods can obtain a near-optimal low-rank approximation for arbitrary $B$. In particular, if $s_{\mathrm{R}} = O(k/\beta)$, then the output of RSVD satisfies $\|B - QX_k\|_{\mathsf{F}} \leq (1+\beta)\cdot\|B - B_k\|_{\mathsf{F}}$ with high probability [149]. The output of generalized Nyström gives the same guarantee when $s_{\mathrm{R}} = O(k/\beta)$ and $s_{\mathrm{L}} = O(k/\beta^3)$ [167]. The following is a more recent near-optimality guarantee for the randomized SVD, with more explicit bounds [29, Theorem 1].

**Theorem 1.3.2** (Halko, Martinsson, Tropp)**.** *Let $A \in \mathbb{R}^{M\times N}$ with singular values $\sigma_1 \geq \sigma_2 \geq \cdots$, and choose a target rank $k \geq 2$ and an oversampling parameter $p \geq 4$ satisfying $k + p \leq \min\{M, N\}$. Then, for all $u, t \geq 1$, the projection $QQ^\top A$*

*returned by the randomized SVD algorithm satisfies the approximation error*

$$\|A - QQ^\top A\|_F \leq \left(1 + t\sqrt{\frac{3k}{p+1}}\right)\sqrt{\sum_{j=k+1}^{N}\sigma_j^2(A)} + ut\frac{\sqrt{k+p}}{p+1}\sigma_{k+1}(A) \qquad (1.5)$$

*with failure probability at most* $2t^{-p} + e^{-u^2}$.

By comparing Equation (1.5) and Equation (1.3), we can factor Equation (1.5) to obtain a near-optimality constant and show that the error of the randomized SVD is close to the best-possible error in the Frobenius norm, while its computational cost is much lower than that of obtaining the optimal rank-$k$ approximation. There exist error bounds in the spectral norm as well [85, Theorem 10.8], however they are not near-optimal in the same sense. In fact, if one could achieve the same near-optimal bound in the spectral norm for all input sketches, one could compute $\|A_k\|_2$ and derive a near-optimal approximation of $\|A\|_2$ using fewer than $N$ sketches, which would violate the lower bound in [180, Chapter 6.2] An error bound close to Equation (1.5) has been proven for the stabilized generalized Nyström method [130].

## 1.4 Hierarchical matrices

There are several different types of hierarchical low-rank matrices, including $\mathcal{H}$-matrices and $\mathcal{H}^2$-matrices. In this work, we consider one of the most general classes: hierarchical off-diagonal low-rank (HODLR) matrices and hierarchical semiseparable (HSS) matrices.[2] The vast literature on this topic, as well as the applications to many different areas in the computational sciences, has resulted in some discrepancies between definitions for these matrix structures. In this section, we provide

[2]This class has more recently been also referred to as hierarchical block-separable (HBS) [105].

the definitions that the present work relies upon and conventions that we adopt throughout this thesis.

A hierarchical matrix has a recursive structure, where diagonal blocks are iteratively subdivided into smaller blocks. Informally, as its name would suggest, a HODLR matrix's blocks which do not intersect the diagonal are low-rank. In order to define HODLR and HSS matrices, we first introduce the definition of a hierarchical tree, which allows us to easily refer to different subblocks of a hierarchical matrix (see Figure 1.1).

**Definition 1.4.1.** *Let $A \in \mathbb{R}^{N\times N}$ and $\mathcal{I}$ be the index set of $A$, so that $\mathcal{I} = [1, 2, \ldots, N]$. For simplicity, we suppose that $N$ is a power of 2. $A$'s hierarchical tree $\mathcal{T}$ is a binary tree of depth $1 \leq L \leq \log_2(N)$ whose root vertex represents $\mathcal{I}$. At every subsequent level $1 \leq \ell \leq L$, any pair of sibling nodes $\alpha, \beta$ with parent $\gamma$ represent the index sets $\mathcal{I}_\alpha$ and $\mathcal{I}_\beta$, which consecutively partition $\mathcal{I}_\gamma$, that is, $\mathcal{I}_\alpha = [1, \ldots, n_0]$ and $\mathcal{I}_\beta = [n_0 + 1, \ldots, |\mathcal{I}_\gamma|]$ for some $1 < n_0 < |\mathcal{I}_\gamma|$.*

In practice, one selects the depth $L$ based on how fine a partition is desired. Additionally, one often uses a balanced binary tree, where the index sets at each level are roughly of the same size. To ensure that our index sets are well-defined, we use the convention that $N$ is a power of 2; however, hierarchical techniques can be easily extended to more general partitions. Admissibility describes whether a pair of nodes in the hierarchical tree corresponds to a low-rank block.

**Definition 1.4.2.** *Given a matrix $A \in \mathbb{R}^{N\times N}$ with hierarchical tree $\mathcal{T}$, a pair of nodes $(\alpha, \beta) \in \mathcal{T}$ is called admissible if the subblock $A(\mathcal{I}_\alpha, \mathcal{I}_\beta)$ is of low rank*[3]

[3]Some works alternatively require that these blocks are of low *numerical* rank, i.e., having singular values after the target rank drop below a certain threshold. However, in this work, we already distinguish between the two settings of exact recovery and approximation from matrix-vector products, and the numerical low-rank case fits into the latter setting of hierarchical matrix approximation, which we address in Section 3.2.

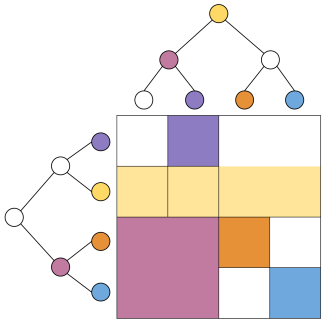


Figure 1.1: An illustration of the correspondence between pairs of nodes in a hierarchical tree and subblocks of the partitioned hierarchical matrix. Vertices of a given color indicate the corresponding subblock of the same color.

*relative to the dimensions of the block.*

Intuitively, if a hierarchical matrix represents a kernel operator, its low rank subblocks signify approximately smooth interactions between points in the domain. Hierarchical matrix techniques can also be viewed as matrix analogues for functional approximation algorithms like the Barnes–Hut and Fast Multipole methods [13, 77], which are analogous to HODLR and HSS matrices, respectively.

In this work, for ease of notation, we consider HODLR matrices with the "weakly-admissible" partition, unless otherwise noted. This means that any hierarchical tree we consider is a balanced binary tree, and if two nodes $\alpha \neq \beta$ are siblings, then they are an admissible pair corresponding to a low-rank block. This structure is illustrated in Figure 1.2 and formally defined below.

**Definition 1.4.3.** *Fix a rank parameter $k$ and leaf level $L$. An $N \times N$ matrix $A$ is* $\mathrm{HODLR}(N, k, L)$ *if $L = 0$ and $N \leq k$ or if $A$ can be partitioned into $(N/2) \times (N/2)$ blocks*

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

*such that $A_{1,2}$ and $A_{2,1}$ are each of rank at most $k$ and $A_{1,1}$ and $A_{2,2}$ are each*

HODLR$(N/2, k, L-1)$. *Here, the integer* $L \geq 0$ *denotes the number of levels of the partition, so that the dimension* $N = N_{base} \cdot 2^L$ *for some* $N_{base} \leq k$.

$$
A = \left[\begin{array}{cc|cc|cc|cc}
A_{11} & W_3 Z_3^\top & W_1 Z_1^\top & & W_0 Z_0^\top & & & \\
U_3 V_3^\top & A_{22} & & & & & & \\
\hline
U_1 V_1^\top & & A_{33} & W_4 Z_4^\top & & & & \\
& & U_4 V_4^\top & A_{44} & & & & \\
\hline
U_0 V_0^\top & & & & A_{55} & W_5 Z_5^\top & W_2 Z_2^\top & \\
& & & & U_5 V_5^\top & A_{66} & & \\
\hline
& & & & U_2 V_2^\top & & A_{77} & W_6 Z_6^\top \\
& & & & & & U_6 V_6^\top & A_{88}
\end{array}\right]
$$

Figure 1.2: A HODLR matrix after three levels of partitioning. The matrices $U_i$, $V_i$, $W_i$, and $Z_i$ have at most $k$ columns, and the $A_{ii}$'s will be partitioned further until a chosen stopping point.

HODLR structure is an ideal data-sparse format for computing, and the hm-toolbox library has been developed to work with such matrices [121]. In particular, a matrix in HODLR$(N, k, L)$ can be stored with just $\mathcal{O}(NkL)$ parameters, as opposed to $\mathcal{O}(N^2)$, as there are $L$ levels and $\mathcal{O}(Nk)$ parameters define the low-rank factors at each level. Similarly, there exist very fast routines for HODLR matrices, e.g., matrix-vector multiplication and matrix-matrix addition cost just $\mathcal{O}(NkL)$ operations, inversion costs $\mathcal{O}(k^2 NL^2)$ operations [80], and one can solve HODLR linear systems in just $\mathcal{O}(Nk^3 L + Nk^2 L^2)$ time [12]. In applications, there are numerous fast techniques involving HODLR matrices, such as computing an approximation to a boundary integral operator in the plane in $\mathcal{O}(k^2 N)$ time [118].

HSS matrices are special cases of HODLR matrices, as are other well-studied structured classes, like diagonal plus low-rank matrices [128, 168, 24]. In addition to having off-diagonal low-rank blocks, HSS matrices also satisfy a nested basis property, and the distinction between HSS and HODLR roughly corresponds to solution operators for constant coefficient elliptic PDEs and variable coefficient elliptic PDEs, respectively. This nestedness often reduces the $L \leq \log_2(N)$ factor in the complexity of HODLR storage and algorithms, as there are shared low-rank factors across different levels, and one therefore does not need to traverse every level of the hierarchy to work with an HSS matrix.

There are several definitions for HSS, and each one is useful in a particular context. For completeness, we include two definitions of HSS matrices: the nested basis definition and the telescoping factorization definition. Before stating these definitions, we first define the useful notion of an HSS block-row and block-column.

**Definition 1.4.4.** *Consider the matrix $A \in \mathrm{HODLR}(N, k, L)$ whose hierarchy is given by the binary tree $\mathcal{T}$. If $\alpha \in \mathcal{T}$ is a node at level $\ell$, the HSS row-block corresponding to $\alpha$ is $A(\mathcal{I}_\alpha, \mathcal{I} \setminus \mathcal{I}_\alpha)$. Similarly, the HSS column-block corresponding to $\alpha$ is $A(\mathcal{I} \setminus \mathcal{I}_\alpha, \mathcal{I}_\alpha)$.*

We now state the first definition of HSS matrices, which describes the ranks of the HSS row-blocks and HSS column-blocks at each level.

**Definition 1.4.5.** *The matrix $A \in \mathbb{R}^{N \times N}$ also belongs to the class $\mathrm{HSS}(N, k, L)$ if $A \in \mathrm{HODLR}(N, k, L)$ and at every level from 1 to $L$, every HSS block-row and block-column is rank-$k$.*

While the previous definition is an intuitive description of HSS structure, the following formulation is also useful in many fast algorithms using HSS matrices,

such as compression [105], computation of matrix functions [42], efficient matrix-matrix multiplications [8], and fast matrix factorizations [182].

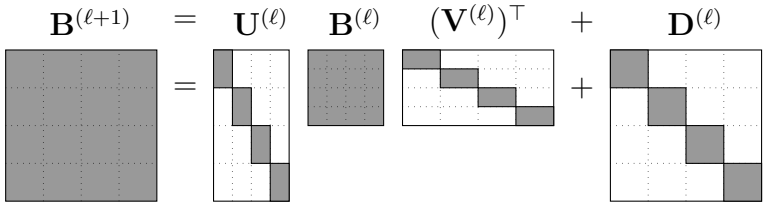
**Definition 1.4.6.** *Let* $\mathbf{B} \in \mathbb{R}^{N\times N}$. *We say that* $\mathbf{B} \in \mathrm{HSS}(N, k, L)$ *if it satisfies*

$$\begin{aligned} \mathbf{B}^{(L+1)} &= \mathbf{B}, \\ \mathbf{B}^{(\ell+1)} &= \mathbf{U}^{(\ell)}\mathbf{B}^{(\ell)}(\mathbf{V}^{(\ell)})^\top + \mathbf{D}^{(\ell)} \quad \textit{for } \ell = 1, \ldots, L, \quad \textit{and} \\ \mathbf{B}^{(1)} &= \mathbf{D}^{(0)}. \end{aligned} \tag{1.6}$$

*Here, we let* $m = N/2^L$, *the dimension of the leaf blocks, and then require that* $\mathbf{D}^{(0)} \in \mathbb{R}^{m\times m}$ *and, for* $\ell = 1, \ldots, L$,

$$\begin{aligned} \mathbf{U}^{(\ell)} &= \mathrm{blockdiag}\left(\mathbf{U}_1^{(\ell)}, \ldots, \mathbf{U}_{2^\ell}^{(\ell)}\right) \quad \textit{where } \mathbf{U}_i^{(\ell)} \in \mathbb{R}^{m\times k} \textit{ has orthonormal columns,} \\ \mathbf{V}^{(\ell)} &= \mathrm{blockdiag}\left(\mathbf{V}_1^{(\ell)}, \ldots, \mathbf{V}_{2^\ell}^{(\ell)}\right) \quad \textit{where } \mathbf{V}_i^{(\ell)} \in \mathbb{R}^{m\times k} \textit{ has orthonormal columns, and} \\ \mathbf{D}^{(\ell)} &= \mathrm{blockdiag}\left(\mathbf{D}_1^{(\ell)}, \ldots, \mathbf{D}_{2^\ell}^{(\ell)}\right) \quad \textit{where } \mathbf{D}_i^{(\ell)} \in \mathbb{R}^{m\times m}. \end{aligned}$$

One level of $\ell = 2$ decomposition is illustrated below.




As in the HODLR case, we assume that the binary tree is balanced and that the size of the diagonal blocks corresponding to leaf-nodes in the tree is $2k$. The analysis for hierarchical matrices in this work extends to more general binary trees in a straightforward way.

The two definitions of HSS matrices can be related by taking the data-sparse representation and appropriately defining the corresponding telescoping factors [42, Proposition 4.2]. In particular, the basis matrices $U$ and $V$ for each level can be stored in the block orthonormal matrices $U^{(\ell)}$ and $V^{(\ell)}$.

Finally, we describe an important subclass of HSS matrices. This structure arises from the observation that is most easily seen from Definition 1.4.5. Given all the low-rank factors corresponding to the finest leaf-node HSS block-rows and block-columns, one can reconstruct the low-rank factors for every square block in the weakly admissible partition at each coarser level, thus determining the blocks up to multiplication by a $k \times k$ so-called core matrix [121]. One may therefore consider the set of HSS matrices whose two largest off-diagonal blocks at the coarsest level determine all the low-rank factors at subsequent finer levels.

**Definition 1.4.7.** *A matrix $A \in \text{HSS}(N, k, L)$ is also in $\text{p-HSS}(N, k, L)$ if $L = 0$ and $N \leq k$ or if $A$ can be partitioned such that its two largest off-diagonal blocks are factored as $UV^\top$ and $WZ^\top$, for $U, V, W, Z \in \mathbb{R}^{N/2 \times k}$, and the rest of $A$ satisfies the following recursive structure:*

$$A = \left[\begin{array}{c|c|cc} H_{11} & W(1{:}\tfrac{N}{4},:)H_{12}V(\tfrac{N}{4}{+}1{:}\tfrac{N}{2},:)^\top & \multicolumn{2}{c}{\multirow{2}{*}{$WZ^\top$}} \\ \hline W(\tfrac{N}{4}{+}1{:}\tfrac{N}{2},:)H_{21}V(1{:}\tfrac{N}{4},:)^\top & H_{22} & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$UV^\top$}} & H_{33} & U(1{:}\tfrac{N}{4},:)H_{34}Z(\tfrac{N}{4}{+}1{:}\tfrac{N}{2},:)^\top \\ \cline{3-4} & & U(\tfrac{N}{4}{+}1{:}\tfrac{N}{2},:)H_{43}Z(1{:}\tfrac{N}{4},:)^\top & H_{44} \end{array}\right], \tag{1.7}$$

*where $H_{ij} \in \mathbb{R}^{k \times k}$ for $j \neq i$, and the diagonal blocks given by the block matrices*

$$\begin{bmatrix} H_{11} & W(1{:}\frac{N}{4},:)H_{12}V(\frac{N}{4}{+}1{:}\frac{N}{2},:)^\top \\ W(\frac{N}{4}{+}1{:}\frac{N}{2},:)H_{21}V(1{:}\frac{N}{4},:)^\top & H_{22} \end{bmatrix} \quad \textit{and}$$

$$\begin{bmatrix} H_{33} & U(1{:}\frac{N}{4},:)H_{34}Z(\frac{N}{4}{+}1{:}\frac{N}{2},:)^\top \\ U(\frac{N}{4}{+}1{:}\frac{N}{2},:)H_{43}Z(1{:}\frac{N}{4},:)^\top & H_{44} \end{bmatrix}$$

*are themselves in $\text{p-HSS}(N/2, k, L-1)$.*

This recursive definition implies that each of the diagonal blocks $H_{jj}$ for $1 \leq$

$j \leq 4$ in Equation (1.7) can be further partitioned into two rank-$k$ off-diagonal blocks, which also inherit the corresponding restricted row and column spaces of the larger blocks. We introduce this family because it forms a large subclass of HSS matrices (that is, a generic HSS matrix is also p-HSS). In addition to HODLR recovery and approximation, we describe a simple recovery algorithm for p-HSS matrices in Chapter 3, and reduce the exact HODLR recovery problem to p-HSS recovery.

### 1.4.1 The peeling algorithm

Recall from Definition 1.4.3 that any HODLR($k$) matrix $A \in \mathbb{R}^{N \times N}$ is composed of four $(N/2) \times (N/2)$ sized blocks. The off-diagonal blocks, $A_{1,2}$ and $A_{2,1}$, are rank-$k$ and the on-diagonal blocks, $A_{1,1}$ and $A_{2,2}$, are themselves HODLR($k$). The key idea of the peeling algorithm is to first recover the low-rank off-diagonal blocks $A_{1,2}$ and $A_{2,1}$, to implicitly subtract them from the matrix, and to then continue on to recursively recover the diagonal HODLR($k$) blocks. More formally, peeling relies on the following observations:

**Observation 1.** We can perform matrix-vector products with the off-diagonal blocks $A_{1,2}$ and $A_{2,1}$ (and their transposes) using matrix-vector products with $A$ and $A^\top$. For instance, we can compute products with $A_{2,1}$ and $(A_{2,1})^\top$ by

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \begin{bmatrix} \Omega \\ 0 \end{bmatrix} = \begin{bmatrix} \sim \\ A_{2,1}\Omega \end{bmatrix}, \quad \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}^\top \begin{bmatrix} 0 \\ \Psi \end{bmatrix} = \begin{bmatrix} (A_{2,1})^\top \Psi \\ \sim \end{bmatrix}. \tag{1.8}$$

and analogously with $A_{1,2}$ and $(A_{1,2})^\top$ by

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \begin{bmatrix} 0 \\ \Omega \end{bmatrix} = \begin{bmatrix} A_{1,2}\Omega \\ \sim \end{bmatrix}, \quad \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}^\top \begin{bmatrix} \Psi \\ 0 \end{bmatrix} = \begin{bmatrix} \sim \\ (A_{1,2})^\top \Psi \end{bmatrix}. \tag{1.9}$$

Here, " $\sim$ " indicates a block of the output that is typically nonzero, but we ignore in our computations. As discussed in Section 1.3, algorithms like RSVD and generalized Nyström can exactly recover a rank-$k$ matrix (with probability one) using $k$ products with each the matrix and its transpose. Thus, when $A$ is exactly HODLR($k$), we can fully recover both off-diagonal blocks using just $4k$ total queries to $A$ (implementing $k$ queries with each of $A_{1,2}, (A_{1,2})^\top, A_{2,1}$, and $(A_{2,1})^\top$).

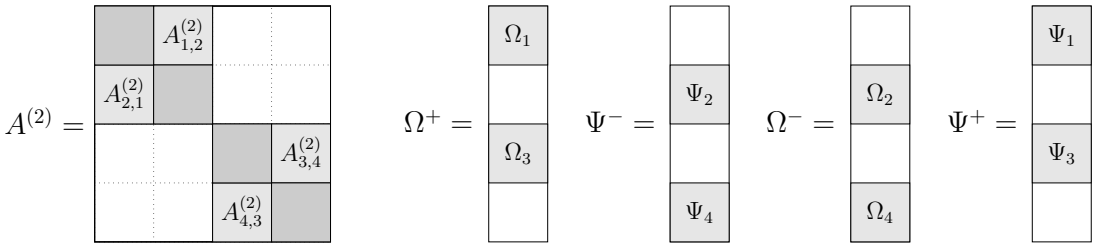



Figure 1.3: State of the hierarchical matrix at the start of the $\ell = 2$ level of peeling. The matrix $A^{(2)}$ denotes the matrix $A$ after subtracting the low-rank off-diagonal blocks recovered at the first level – these blocks are zero in $A^{(2)}$ and shown as white in the figure. For $j = 1, 3$, we can simultaneously obtain the products $A^{(2)}_{j+1,j}\Omega_j$ and the transpose-products $(A^{(2)}_{j+1,j})^\top \Psi_{j+1}$ from the sketches $A\Omega^+$ and $A^\top\Psi^-$. Letting $\Omega^+$ and $\Psi^-$ have $k$ columns and blocks chosen to be appropriate sketching matrices, these products can be used to exactly recover the rank-$k$ blocks $A^{(2)}_{j+1,j}$ for $j = 1, 3$. Obtaining products with and recovering $A^{(2)}_{j-1,j}$ for $j = 2, 4$ can be done analogously using $\Omega^-$ and $\Psi^+$. Overall, we can recover all four rank-$k$ off diagonal blocks at level $\ell = 2$ using just $4k$ queries to $A$ ($k$ for each of $\Omega^+, \Omega^-, \Psi^+$, and $\Psi^-$).

**Observation 2.** After exactly obtaining the blocks $A_{2,1}$ and $A_{1,2}$, we can *simultaneously* perform matrix-vector products with the diagonal blocks $A_{1,1}$ and $A_{2,2}$ using matrix-vector products with $A$. In particular, we can compute for any $\Omega_1$ and $\Omega_1$,

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \begin{bmatrix} \Omega_1 \\ \Omega_2 \end{bmatrix} - \begin{bmatrix} A_{1,2}\Omega_2 \\ A_{2,1}\Omega_1 \end{bmatrix} = \begin{bmatrix} A_{1,1}\Omega_1 \\ A_{2,2}\Omega_2 \end{bmatrix},$$

and analogously

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}^\top \begin{bmatrix} \Psi_1 \\ \Psi_2 \end{bmatrix} - \begin{bmatrix} (A_{1,2})^\top\Psi_2 \\ (A_{2,1})^\top\Psi_1 \end{bmatrix} = \begin{bmatrix} (A_{1,1})^\top\Psi_1 \\ (A_{2,2})^\top\Psi_2 \end{bmatrix}.$$

Since $A_{1,1}$ and $A_{2,2}$ are themselves HODLR($k$), by Observation 1, we can recover their low-rank off diagonal blocks (each of size $(N/4) \times (N/4)$) using just $4k$ queries to $A$ (see Figure 1.3).

**Observation 3.** We can repeat this process, recursing toward the diagonal, to recover smaller and smaller off-diagonal low-rank blocks. Critically, the number of matrix-vector products used to recover the off-diagonal blocks at a given level depends only on the rank parameter $k$ and is *independent* of the level itself. After $L = \lceil \log_2(N/k) \rceil$ recursive steps, the blocks will be of size at most $k$. At this point, the diagonal blocks can be recovered all at once using $k$ matrix-vector products. Overall, $4k$ queries to $A$ are used at each level, and $k$ are used at the final level, giving total cost of $O(k \cdot (L + 1))$ matvec queries.

CHAPTER 2

# STRUCTURED MATRIX RECOVERY AND APPROXIMATION FROM MATRIX-VECTOR PRODUCTS

In this chapter[1], we describe and prove results for the query complexity of recovering of common structured matrices, such as symmetric, orthogonal, circulant, Toeplitz, and low-rank matrices. We establish matching upper and lower bounds for the query complexity of these recovery problems. Upper bounds are derived from error guarantees for recovery algorithms, whereas lower bounds require other types of analysis. We also investigate the more general problem of near-optimal matrix approximation for the family of sparse matrices with a known sparsity pattern. For simplicity, we consider algorithmically-determined input vectors and even allow our inputs to be selected adaptively and without constraints. Example structures that have been studied extensively in this setting include low-rank matrices [85, 166], hierarchical low-rank matrices [111, 117, 106, 105, 151, 84], diagonal matrices [19, 160, 15, 61], sparse matrices [55, 53, 52, 178, 56], and beyond [173, 150].

## 2.1 Matrix recovery from matrix-vector products for basic matrix structures

It is always possible to recover any $N \times N$ matrix $A$ in $N$ queries by selecting the input vectors as canonical basis vectors and recovering $A$ column-by-column.

[1]This chapter includes excerpts and results from two papers [84, 6]. I worked closely with Alex Townsend to develop all of the algorithms and theory in [84]. The work in [6] was the result of a large collaboration led by Chris Musco, joint with Noah Amsel, Tyler Chen, Fezya Duman Keles, and Cameron Musco. I contributed theory for upper and lower bounds and algorithmic ideas. The numerical experiments included in this section were performed by Tyler Chen.

However, if $A$ is a structured matrix, we would hope to exploit that structure and recover $A$ using far fewer queries. Each matrix-vector product query yields $N$ equations linear in the parameters defining the entries of $A$ as

$$\begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & \ddots & \vdots \\ A_{N1} & \cdots & A_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix} \quad \Longrightarrow \quad \begin{matrix} A_{11}x_1 + \cdots + A_{1N}x_N = b_1 \\ \vdots \\ A_{N1}x_1 + \cdots + A_{NN}x_N = b_N \end{matrix} .$$

This suggests that one may perform enough matrix-vector products to construct a linear system with more equations than unknowns and solve for them. Of course, if there are $N^2$ unknowns, this requires $N$ matrix-vector products. At this point, one could have more efficiently recovered $A$ column-by-column. However, if the entries of $A$ are functions that are linear in fewer than $N^2$ parameters, solving a linear system can be a reasonable strategy. This observation motivates the following definition.

**Definition 2.1.1.** *A linearly parametrized family of $N \times N$ matrices $\mathcal{A} \subset \mathbb{R}^{N\times N}$ is given by the map $\mathcal{A} : \mathbb{R}^p \to \mathbb{R}^{N\times N}$, which takes*

$$\theta \mapsto \sum_{i=1}^{p} \theta_i A_i,$$

*where $\theta \in \mathbb{R}^p$ is a vector of $p$ parameters and $\{A_i\}_{i=1}^p$ is a set of linearly independent basis matrices.*

We note that according to this definition, any matrix $A \in \mathcal{A}$ is uniquely defined by its $p$ parameters. Examples of linearly parametrized families include tridiagonal matrices, symmetric matrices, circulant matrices, and Toeplitz matrices. For these structures, one can take the canonical basis matrices so that the parameters are entries of the structured matrix. The family of rank-$k$ matrices is not linearly parametrized. The following lower bound on $\mathrm{QC}(\mathcal{A})$ holds when $\mathcal{A}$ is linearly parameterized.

**Lemma 2.1.2.** *If $\mathcal{A}$ is a linearly parametrized family of matrices in $p$ parameters,*

$$QC(\mathcal{A}) \geq \left\lceil \frac{p}{N} \right\rceil, \tag{2.1}$$

*where $\lceil \cdot \rceil$ is the ceiling function.*

*Proof.* Recovering any $A \in \mathcal{A}$ is equivalent to recovering the $p$ parameters which define $A$. Each matrix-vector product query yields $N$ linear equations in these $p$ parameters. If this linear system has a unique solution, there must be at least as many equations as unknowns. Thus, $N \times (\#\text{ queries}) \geq p$. □

For some linearly parametrized families, one can derive an algorithm that achieves the lower bound in Equation (2.1), while for others, we prove that it is not feasible. Note that Equation (2.1) is not a valid lower bound for matrix structures that are not linearly parametrized. Thus, Equation (2.1) cannot be applied to the recovery of rank-$k$ or HODLR matrices.

The construction of the linear system guarantees the existence of solutions, as the linear system is generated from matrix-vector products, so we know a priori that there exist $p$ parameters satisfying the equations. A recent result by Otto [136] shows that if the matrix recovery problem for a given linearly parametrized family $\mathcal{A}$ has a unique solution matrix using a particular set of $s$ matrix-vector products, then for almost all other sets of $s$ input vectors with respect to the Lebesgue measure, the matrix recovery problem for $\mathcal{A}$ has a unique solution. In particular, if one has a deterministic recovery algorithm for $\mathcal{A}$ using $s$ input vectors, then the linear system generated by $s$ random Gaussian matrix-vector products has a unique solution with probability 1. We employ this useful result several times, as it allows us to relate deterministic and randomized recovery algorithms.

### 2.1.1 Recovering some common structured matrices

We begin by considering how to exactly recover several common structured matrices. Some form linearly parametrized families and others do not. We also consider both deterministic and randomized inputs, where the probability of success is 1. Table 1.1 displays our results.

#### Diagonal matrices

If $A$ is known to be a diagonal matrix, then its diagonal entries satisfy $\text{diag}(A) = A\mathbf{1}_N$, where $\mathbf{1}_N$ is the all ones vector of size $N$. This means a diagonal matrix can be recovered with one matrix-vector product, so QC(diagonal matrices) = 1.

#### Block diagonal matrices

If $A$ is known to be a block diagonal matrix with $k \times k$ blocks, then its diagonal blocks can be recovered from $k$ matrix-vector products of the form $A(\mathbf{1}_{N/k} \otimes e_j)$ for $1 \leq j \leq k$, where $A(\mathbf{1}_{N/k} \otimes e_j)$ returns the $j$th column of each block stacked into a vector. Here, $e_j$ is the $j$th unit canonical vector of size $k$ and '$\otimes$' denotes the Kronecker product. Then, QC($k$-block diagonal matrices) $\leq k$. By Lemma 2.1.2, this is an equality.

#### Tridiagonal matrices

If $A$ is known to be a tridiagonal matrix, then it can be recovered with three matrix-vector products, but not fewer by Equation (2.1). Therefore,

QC(tridiagonal matrices) = 3. Since $A$ is tridiagonal, we have

$$\begin{bmatrix} A_{11} \\ A_{21}+A_{23} \\ A_{33} \\ A_{43}+A_{45} \\ \vdots \end{bmatrix} = A \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}, \quad \begin{bmatrix} A_{12} \\ A_{22} \\ A_{32}+A_{34} \\ A_{44} \\ \vdots \end{bmatrix} = A \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}, \quad \begin{bmatrix} A_{11}+A_{21} \\ A_{12}+A_{22}+A_{32} \\ A_{23}+A_{33}+A_{43} \\ A_{34}+A_{44}+A_{54} \\ \vdots \end{bmatrix} = A^\top \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \vdots \end{bmatrix}, \tag{2.2}$$

where $A_{jk}$ denotes the $(j,k)$ entry of $A$. Thus, the entries of the tridiagonal matrix can be found recursively from Equation (2.2) using $\mathcal{O}(N)$ operations; the first two matrix-vector products immediately give the diagonal entries $A_{11}, A_{22}, \ldots$ and $A_{12}$; the third query then gives $A_{21}$ and $A_{32}$; from this, the first query gives $A_{23}$; and so on. Alternatively, one can multiply $A$ by the three inputs $[1,0,0,1,\cdots]^\top$, $[0,1,0,0,1,\cdots]^\top$, and $[0,0,1,0,0,1,\cdots]^\top$, extracting each nonzero entry of the tridiagonal matrix [131]. Both algorithms yield the upper bound of 3 on QC(tridiagonal matrices), and a parameter count and Lemma 2.1.2 imply this is an equality.

If $A$ is a symmetric tridiagonal matrix, then only two matrix-vector products are required, as the third query in Equation (2.2) is unnecessary because the first two input vectors sum to the third. Then, QC(symmetric tridiagonal matrices) = 2.

**Rank-k matrices**

Intuitively, one needs to query a rank-$k$ matrix (and its transpose) $k$ times to recover the $k$-dimensional row and column spaces. Indeed, we show that QC(rank-$k$ matrices) = $2k$ in the sense of exact recovery.

In the numerical setting, if $A$ is a rank-$k$ matrix, the randomized SVD recovers

$A$ with probability 1 from $2k+p$ matrix-vector products [120], where $p$ is a small oversampling factor, i.e., $p = 5$. Let $\Omega \in \mathbb{R}^{N\times(k+p)}$ be a random matrix with i.i.d. standard Gaussian entries. Then, with probability 1, we have $A = QQ^\top A$, where $Q \in \mathbb{R}^{N\times k}$ is a matrix with orthonormal columns that form a basis for the column space of $A\Omega$. To construct $QQ^\top A$, we only need to do matrix-vector products. We first compute $A\Omega$, which takes $k+p$ matrix-vector products, then compute $Q^\top A = (A^\top Q)^\top$, which costs $k$ further queries. The matrix $Q$ can be computed from $A\Omega$ by a column-pivoted QR factorization, and since $A\Omega$ is a rank-$k$ matrix, one can take an economized version for which $Q$ has only $k$ columns, not $k+p$. An algorithmic description of the randomized SVD can be found on page 9 of [85].

It is important to have a randomized algorithm for low-rank matrix recovery to avoid the input vectors being in the $N-k$ dimensional nullspace of $A$. In fact, it is impossible to recover any low-rank matrix using a deterministic algorithm with fixed input vectors. The oversampling parameter is also critical for a stable algorithm as there is always a nontrivial chance that a random Gaussian vector has a large component in the nullspace of $A$. The randomized SVD requires $2k+p$ matrix-vector product queries to recover a rank-$k$ matrix. It also recovers a near-optimal approximation of a numerically rank-$k$ matrix with probability at least $1-6p^{-p}$ [85], where a matrix has numerical rank $k$ if its largest $k$ singular values are all above some tolerance. This probability converges very quickly to 1 for only moderate increases in $p$, and importantly does not depend on $N$ or $k$. For a symmetric rank-$k$ matrix, the Nyström method may be preferred, as it only requires $k+p$ matrix-vector product queries because it can exploit the symmetry of $A$ [130].

For a nonsymmetric rank-$k$ matrix, the randomized SVD achieves near-optimal query complexity due to the following lemma, which shows QC(rank-$k$ matrices) = 2k. While this result is intuitive, the proof is more complex than we expect, particularly because we must consider several degenerate cases of input-output pairs. Specifically, we deal with the cases where inputs lie in the nullspace of the matrix, and the matrix-vector products therefore yield zero vectors. When inputs are chosen randomly in randomized recovery algorithms such as the randomized SVD and the Nyström method, these cases do not occur with probability 1. The following result not only provides a lower bound on the query complexity of rank-$k$ matrices, but also constructs infinite families of rank-$k$ matrices that satisfy the matrix-vector products when the number of queries is too low.

As a final note, in the following result we make the assumption that the input matrices $X$ and $W$ have orthonormal columns. We can do this for the following reasons. First, without loss of generality, we assume the columns of $X$ and $W$ are linearly independent; if not, then some matrix-vector products only provide redundant information. We also assume their columns are orthonormal. Let $X = Q_1R_1$ and $W = Q_2R_2$ be the QR factorizations of $X$ and $W$. Then, we reduce the problem to the equivalent recovery problem given by the matrix-vector products $AQ_1 = YR_1^{-1}$ and $A^\top Q_2 = ZR_2^{-1}$. In fact, there exists an orthonormal basis $[\tilde{X}\ \hat{X}]$ for col(X), where $\tilde{X} \in \mathbb{R}^{N\times p}$, $\text{col}(\tilde{X}) \subseteq \text{col}(A^\top)$, and $\text{col}(\hat{X}) \subseteq \text{null}(A)$. Similarly, $[\tilde{W}\ \hat{W}]$ is an orthonormal basis for the col(W) such that $\tilde{W} \in \mathbb{R}^{N\times q}$, $\text{col}(\tilde{W}) \subseteq \text{col}(A)$, and $\text{col}(\hat{W}) \subseteq \text{null}(A^\top)$. We define $\tilde{Y} = A\tilde{X}$, $\hat{Y} = A\hat{X}$, $\tilde{Z} = A^\top\tilde{W}$, and $\hat{Z} = A^\top\hat{W}$ and solve the equivalent recovery problem with inputs $[\tilde{X}\ \hat{X}]$ and $[\tilde{W}\ \hat{W}]$.

**Lemma 2.1.3.** *Let $N$ and $k$ be integers such that $1 \leq k < N$, and let $X \in \mathbb{R}^{N\times k_1}$ and $W \in \mathbb{R}^{N\times k_2}$ have orthonormal columns. Suppose that a matrix $A$ of*

*rank at most $k$ satisfies the matrix-vector products $AX = Y$ and $A^\top W = Z$. If $\min(k_1, k_2) < k$ and $\max(k_1, k_2) < N$, then there are infinitely many distinct $N \times N$ matrices $B$ of rank $\leq k$ satisfying the same $k_1$ matrix-vector products $BX = Y$ and $k_2$ matrix-vector products $B^\top W = Z$.*

*Proof.* Suppose $AX = Y$ and $A^\top W = Z$, where $X \in \mathbb{R}^{N \times k_1}$ and $W \in \mathbb{R}^{N \times k_2}$. We now construct a matrix $B \neq A$ in several cases.

**Case 1: $\mathbf{p = q = 0}$.** Since $p = q = 0$, the whole of $\mathrm{col}(X)$ and $\mathrm{col}(W)$ lie in the null spaces of $A$ and $A^\top$, respectively. We may trivially take $B = 2A$, so that $B \neq A$ if $A$ is nonzero. Otherwise, if $A = 0$, we can construct a rank-1 matrix $B$ as follows. Because $\mathrm{null}(X^\top)$ and $\mathrm{null}(W^\top)$ are nontrivial subspaces, we can select nonzero vectors $v \in \mathrm{null}(X^\top)$ and $u \in \mathrm{null}(W^\top)$. Then, define $B = uv^\top$. It is clear that $BX = 0$ and $B^\top W = 0$, however $B$ is nonzero so $B \neq A$.

**Case 2: $\mathbf{p = 0}$ and $\mathbf{q > 0}$.** Since $q \leq k_1 < N$ and the columns of $\tilde{W}$ are linearly independent, there exist infinitely many matrices $P \in \mathbb{R}^{q \times N}$ such that $P\tilde{W} = I_{q \times q}$. Consider any matrix of the form $B = P^\top \tilde{W}^\top A$ for any such $P$. We note that $BX = P^\top \tilde{W}^\top AX = 0$ (as $p = 0$), $B^\top W = A^\top \tilde{W} PW = Z$ (as $P\tilde{W} = I_{q \times q}$), and $\mathrm{rank}(B) \leq k$. It remains to show that there is a choice of $P$ so that $B \neq A$.

We demonstrate this by producing two distinct matrices $B_1 = P_1^\top \tilde{W}^\top A$ and $B_2 = P_2^\top \tilde{W}^\top A$ such that $B_1 \neq B_2$ and conclude that at least one of $B_1$ or $B_2$ must differ from $A$. Since $q \leq \max\{k_1, k_2\} < N$ there exists a nonzero vector $v$ such that $\tilde{W}^\top v = 0$. We select $P_1$ such that $P_1 \tilde{W} = I_{q \times q}$ and $P_1 v = 0$ but select $P_2$ such that $P_2 \tilde{W} = I_{q \times q}$ and $P_2 v = e_1$, where $e_1$ is the first canonical unit vector. We note that $B_1^\top v = A^\top \tilde{W} P_1 v = 0$ but $B_2^\top v = A^\top \tilde{W} e_1 \neq 0$ so $B_1 \neq B_2$.

**Case 3: $\mathbf{p > 0}$ and $\mathbf{q = 0}$.** This case follows by applying case 2 to $A^\top$.

**Case 4: $\mathbf{p > 0}$ and $\mathbf{q > 0}$.** Consider a family of possible $B$'s given by

$$B = \begin{bmatrix} \tilde{Y} & \tilde{W} \end{bmatrix} \begin{bmatrix} I_{p\times p} - C\tilde{Z}^\top \tilde{X} & C \\ (\tilde{W}^\top \tilde{Y} C - I_{q\times q})\tilde{Z}^\top \tilde{X} & I_{q\times q} - \tilde{W}^\top \tilde{Y} C \end{bmatrix} \begin{bmatrix} \tilde{X}^\top \\ \tilde{Z}^\top \end{bmatrix}, \tag{2.3}$$

where $C$ is any $p \times q$ matrix. Any $B$ in Equation (2.3) satisfies $BX = Y$ since

$$\begin{aligned} BX &= \begin{bmatrix} \tilde{Y} & \tilde{W} \end{bmatrix} \begin{bmatrix} I_{p\times p} - C\tilde{Z}^\top \tilde{X} & C \\ (\tilde{W}^\top \tilde{Y} C - I_{q\times q})\tilde{Z}^\top \tilde{X} & I_{q\times q} - \tilde{W}^\top \tilde{Y} C \end{bmatrix} \begin{bmatrix} \tilde{X}^\top X \\ \tilde{Z}^\top X \end{bmatrix} \\ &= \begin{bmatrix} \tilde{Y} & \tilde{W} \end{bmatrix} \begin{bmatrix} I_{p\times p} - C\tilde{Z}^\top \tilde{X} & C \\ (\tilde{W}^\top \tilde{Y} C - I_{q\times q})\tilde{Z}^\top \tilde{X} & I_{q\times q} - \tilde{W}^\top \tilde{Y} C \end{bmatrix} \begin{bmatrix} I_{p\times p} & 0 \\ \tilde{Z}^\top \tilde{X} & 0 \end{bmatrix} \\ &= \begin{bmatrix} \tilde{Y} & \tilde{W} \end{bmatrix} \begin{bmatrix} I_{p\times p} & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \tilde{Y} & 0 \end{bmatrix} = Y, \end{aligned}$$

where we used the fact that $\tilde{Z}^\top \hat{X}$ is the zero matrix as $\mathrm{col}(\tilde{Z}) \subseteq \mathrm{col}(A^\top)$ and $\mathrm{col}(\hat{X}) \subset \mathrm{null}(A)$. A similar argument shows that $B^\top W = Z$. Moreover, $\mathrm{rank}(B) \leq \mathrm{rank}([\tilde{Y}\ \tilde{W}]) \leq k$, where the last inequality follows from the fact that $\mathrm{col}([\tilde{Y}\ \tilde{W}]) \subseteq \mathrm{col}(A)$. This means for any choice of $C$, the matrix $B$ in Equation (2.3) satisfies $BX = Y$, $B^\top W = Z$, and $\mathrm{rank}(B) \leq k$. Now, we just have to show that there is a choice of $C$ so that $B \neq A$.

**Case 4 (i): $\mathbf{k_1 = \min\{k_1, k_2\}}$.** In this case, we know that $p \leq k_1 < k$, so there exists a nonzero vector $v \in \mathrm{col}(A^\top)$ such that $\tilde{X}^\top v = 0$. Since $v \notin \mathrm{null}(A)$, we have $Av \neq 0$ and we now give a choice of $C$ so that $B \neq A$. Note that we have

$$Bv = (I - \tilde{W}\tilde{W}^\top)\tilde{Y}C\tilde{Z}^\top v + \tilde{W}\tilde{Z}^\top v.$$

If $(I - \tilde{W}\tilde{W}^\top)\tilde{Y}C\tilde{Z}^\top v \neq 0$ is nonzero for any matrix $C$, then we can generate two different $B$'s by replacing $C$ by $2C$. Since these two $B$s cannot both be equal to

$A$, the matrix $A$ is not uniquely determined by its matrix-vector products. On the other hand, if $(I - \tilde{W}\tilde{W}^\top)\tilde{Y}C\tilde{Z}^\top v = 0$ for all choices of the matrix $C$, we conclude that $\tilde{Z}^\top v = 0$ so that $Bv = 0$. Since $Av \neq 0$, we must have $A \neq B$.

**Case 4 (ii): $\mathbf{k_2} = \min\{\mathbf{k_1}, \mathbf{k_2}\}$.** Now, $q \leq k_2 < k$, and there exists a nonzero vector $u \in \mathrm{col}(A)$ such that $\tilde{W}^\top u = 0$. Since $u \notin \mathrm{null}(A^\top)$, we have $A^\top u \neq 0$ and $B^\top u = (I - \tilde{X}\tilde{X}^\top)\tilde{Z}C^\top\tilde{Y}^\top u + \tilde{X}\tilde{Y}^\top u$. Analogously, to case 4(i) we have $B \neq A$. □

In Lemma 2.1.3, we find that we need $k_1 \geq k$ and $k_2 \geq k$ to hope to exactly recover a rank-$k$ matrix. Therefore, one needs at least $2k$ matrix-vector products. The randomized SVD is a stable recovery algorithm using only $2k + p$, making it near-optimal in terms of the number of matrix-vector products. As such, we obtain the equality $\mathrm{QC}(N \times N \text{ rank-}k \text{ matrices}) = 2k$.

### Circulant, Toeplitz, and Hankel matrices

We now find the query complexities of circulant, Toeplitz, and Hankel matrices. For these recovery problems, we prefer to use randomized inputs, which have a natural extension to Gaussian processes in infinite dimensions. An $N \times N$ circulant matrix $C_c$ is determined by one vector $c \in \mathbb{R}^N$, where $c = [c_0; c_1; \cdots; c_{N-1}]$:

$$C_c = \begin{bmatrix} c_0 & c_{N-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{N-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{N-2} & & \ddots & \ddots & c_{N-1} \\ c_{N-1} & c_{N-2} & \cdots & c_1 & c_0 \end{bmatrix}.$$

One can recover $C_c$ with one matrix-vector product $e_1$, which extracts the vector $c$ exactly, allowing us to recover all of $C_c$. Thus, QC(circulant matrices) = 1. This is consistent with Equation (2.1), as the family of circulant matrices is clearly linearly parametrized. We would also like to develop a randomized circulant recovery algorithm using an input vector $g$, a random Gaussian vector in $\mathbb{R}^N$. Because $C_c$ can be viewed as the integral kernel of a convolution operator and convolution is commutative, we have $C_c g = C_g c = y$. One can easily solve $C_g c = y$ for the vector $c$, as circulant matrices are diagonalized by the discrete Fourier transform (DFT) matrix. That is, $C_g = \frac{1}{N} F^{-1} \Lambda F$, where $F$ is the $N \times N$ DFT matrix and $\Lambda$ is a diagonal matrix with diagonal entries given by $Fg$. We find that $c = NF^{-1}\Lambda^{-1}Fy$, which can be computed in $\mathcal{O}(N \log N)$ operations using the fast Fourier transform (FFT).

One can use two matrix-vector products to recover an $N \times N$ Toeplitz matrix $T$ as it is uniquely defined among all Toeplitz matrices by its first column $t_1$ and first row $t_2^\top$, where $t_1, t_2 \in \mathbb{R}^N$. The deterministic matrix-vector products $Te_1$ and $T^\top e_1$ will extract the parameters which define $T$. This also realizes the bound in Equation (2.1), as $T$ is defined by $2N-1$ parameters. Thus, QC(Toeplitz matrices) = 2. However, we again prefer to recover $T$ using the two random matrix-vector products $Tg = y$ and $Th = z$, where $g$ and $h$ are random Gaussian input vectors $\mathbb{R}^N$, with i.i.d. entries. Since a Toeplitz matrix $T$ is constant along its diagonals, there exists a circulant matrix $C_a$ such that

$$Tg = \begin{bmatrix} I_N & 0_N \end{bmatrix} C_a \begin{bmatrix} g \\ 0 \end{bmatrix}, \qquad Th = \begin{bmatrix} I_N & 0_N \end{bmatrix} C_a \begin{bmatrix} h \\ 0 \end{bmatrix},$$

where $a$ is the $2N \times 1$ vector given by $a = [t_1; 0; t_2(N:-1:2)]$,[2] $I_n$ is the $N \times N$

[2]Here,$t_2(N:-1:2)$ is the vector obtained by removing the first entry of $t_2$ and then reversing the order of the entries.

identity matrix, and $0_N$ is the $N \times N$ matrix of zeros. We now note that

$$C_a \begin{bmatrix} g \\ 0 \end{bmatrix} = C_{[g;0]}a, \qquad C_a \begin{bmatrix} h \\ 0 \end{bmatrix} = C_{[h;0]}a.$$

Since left multiplication by $[I_N \ 0_N]$ restricts to the top half of the output, each product gives us $N$ equations in the $2N-1$ entries of $a$. Putting these together yields $2N$ equations in $2N-1$ unknowns, and one can then solve for the entries of $t_1$ and $t_2$, together with the constraint that the first entry of $t_1$ and $t_2$ are equal.

For example, in the linear system for the case of $N = 3$ is as follows:

$$\begin{bmatrix} g_1 & 0 & 0 & g_3 & g_2 \\ g_2 & g_1 & 0 & 0 & g_3 \\ g_3 & g_2 & g_1 & 0 & 0 \\ h_1 & 0 & 0 & h_3 & h_2 \\ h_2 & h_1 & 0 & 0 & h_3 \\ h_3 & h_2 & h_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{23} \\ t_{22} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix}.$$

The columns of this linear system can be permuted so that the last two columns are moved to the front. This yields a $2N$ by $2N-1$ Sylvester matrix. Because a Sylvester matrix satisfies a low-rank displacement structure, we strongly suspect that an $\mathcal{O}(N^2)$ solver can be used to recover $T$ [75], or possibly even an $\mathcal{O}(N \log N)$ solver [119].

While this algorithm recovers a Toeplitz matrix exactly, there has also been recent work on the recovery of a near-optimal approximation of a Toeplitz matrix in the sense of the Frobenius norm using sublinear query complexity, and this approximation is itself Toeplitz [94]. This approach considers query complexity in terms of both entry-wise sample complexity and vector sample complexity.

Finally, one can recover an $N \times N$ Hankel matrix $H$ with two matrix-vector

products, as suggested by the bound Equation (2.1). Any Hankel matrix is a Toeplitz matrix with permuted columns, i.e., $H = PT$, where $P$ is an exchange matrix and $T$ is a Toeplitz matrix. If $g$ and $h$ are random Gaussian vectors, one recovers $H$ by recovering $T$ since $Hg = y$ and $Hh = z$ are equivalent to $Tg = Py$ and $Th = Pz$, respectively.

### Symmetric matrices

Unfortunately, there are some structured matrices for which one needs many more matrix-vector products than suggested by the lower bound in Equation (2.1). If $A$ is known to be a symmetric matrix, then it has $N(N+1)/2$ parameters, suggesting that $\lceil (N+1)/2 \rceil$ queries might be enough. However, a simple argument reveals that a symmetric matrix cannot be recovered from fewer than $N$ matrix-vector products, regardless of the input vectors.

**Lemma 2.1.4.** *Suppose a symmetric matrix satisfies $N-1$ matrix-vector products. Then, there are infinitely many other $N \times N$ symmetric matrices satisfying the same $N-1$ matrix-vector products. As a consequence, $QC(\text{symmetric matrices}) = N$.*

*Proof.* Suppose a symmetric matrix $A$ satisfies $Ax_j = y_j$ for $1 \leq j \leq N-1$. Consider the symmetric matrix $B = A + vv^\top$, where $v$ is any nontrivial vector orthogonal to the span of $\{x_1, \ldots, x_{N-1}\}$. The matrix $B$ is symmetric, as it is the sum of two symmetric matrices. By construction $B \neq A$, but $Bx_j = y_j$ for $1 \leq j \leq N-1$. □

Of course, $N$ matrix-vector queries can be used to recover a symmetric matrix. We note that this proof is constructive and quantifies the uniqueness of possible symmetric matrices $B$ satisfying the same $N-1$ matrix-vector products as $A$. The

proof in Lemma 2.1.4 also includes the recovery of positive definite matrices, as if $A$ is positive definite, then so is $A + vv^\top$.

### Orthogonal matrices

If $A$ is known to be an orthogonal matrix, then one needs $N$ matrix-vector products to recover $A$.

**Lemma 2.1.5.** *Suppose an $N \times N$ orthogonal matrix satisfies $N-1$ matrix-vector products. Then, there are infinitely many other $N \times N$ orthogonal matrices satisfying the same $N-1$ matrix-vector products. Therefore, QC(orthogonal matrices) = $N$.*

*Proof.* Suppose an orthogonal matrix $A$ satisfies $Ax_j = y_j$ for $1 \leq j \leq N-1$. Consider the matrix $B = A(I - 2\frac{vv^\top}{v^\top v})$, where $v$ is any nontrivial vector orthogonal to the span of $\{x_1, \ldots, x_{N-1}\}$. The matrix $B$ is orthogonal, as it is the product of two orthogonal matrices. It is easy to check that $B \neq A$, but $Bx_j = y_j$ for $1 \leq j \leq N-1$. □

### Toeplitz-like matrices

We say that a matrix $A \in \mathbb{R}^{N \times N}$ is Toeplitz-like if it satisfies the following so-called displacement structure [89, Part II, chapt. 2], [92, chapt. 7]:

$$Z_1 A - A Z_{-1} = G H^\top, \qquad Z_t = \begin{bmatrix} 0 & t \\ I_{N-1} & 0 \end{bmatrix}, \quad G, H \in \mathbb{R}^{N \times 2}, \tag{2.4}$$

where $I_{N-1}$ is the $(N-1) \times (N-1)$ identity matrix. Since the eigenvalues of $Z_1$ and $Z_{-1}$ are disjoint, the matrix $A$ is uniquely defined by a rank-$k$ matrix $GH^T$

with $k = 2$ [159]. Therefore, we recover $GH^T$ using matrix-vector products with $A$ and $A^\top$. Let $X$ be $N \times m$ and $Y$ be $N \times n$ matrices with i.i.d. random Gaussian entries. Then, from Equation (2.4), we find that

$$GH^\top X = Z_1 AX - AZ_{-1}X \quad \text{and} \quad HG^\top Y = A^\top Z_1^\top H - Z_{-1}^\top A^\top H$$

The matrix $GH^\top$ can be recovered by the randomized Nyström method [130, 167]:

$$GH^\top = GH^\top X (Y^\top GH^\top X)^\dagger (HG^\top Y)^\top,$$

where $^\dagger$ denotes the pseudoinverse. This means that $GH^\top$ can be recovered with $m = 2p + 4$ and $n = 5p + 8$ for a total of $7p + 12$ queries. Once $G$ and $H$ are recovered, the matrix $A$ can be computed by solving Equation (2.4) using the Bartels–Stewart algorithm in $\mathcal{O}(N^3)$ operations [14]. We note that one may reduce the number of matrix-vector products by setting $X = \begin{bmatrix} g \mid Z_{-1}g \mid \cdots \mid Z_{-1}^{p+1}g \end{bmatrix}$ and $Y = \begin{bmatrix} h \mid Z_1^\top h \mid \cdots \mid (Z_1^\top)^{p+1}h \end{bmatrix}$, totaling $2p + 6$ queries instead. However, in this case, the inputs $X$ and $Y$ do not have independent columns, making the recovery algorithm's theoretical analysis challenging. Due to their displacement structure, similar recovery algorithms are possible for Hankel-like, Toeplitz-and-Hankel-like, and Bézout-like matrices [25].

### Diagonal-plus-circulant matrices

Using two matrix-vector products, we recover an $N \times N$ matrix $A$ which can be written as $A = D + C$, where $D$ is an $N \times N$ diagonal matrix and $C$ is an $N \times N$

circulant matrix. The matrix $A$ has the following form:

$$A = \begin{bmatrix} d_1 & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & d_N \end{bmatrix} + \begin{bmatrix} c_0 & c_{N-1} & \cdots & c_1 \\ \vdots & \vdots & \ddots & \vdots \\ c_{N-1} & c_{N-2} & \cdots & c_0 \end{bmatrix}$$

$$= \begin{bmatrix} d_1 + c_0 & c_{N-1} & \cdots & c_2 & c_1 \\ c_1 & d_2 + c_0 & c_{N-1} & \cdots & c_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{N-1} & c_{N-2} & \cdots & \cdots & d_N + c_0 \end{bmatrix}$$

Note that the parameter $c_0$ is not needed to define the matrix, and we can rename the diagonal entries $d_i + c_0$ to $d_i$, for each $1 \leq i \leq n$. First, we query $A$ with the first canonical basis vector $e_i$: $Ae_i = y_1$. Because $y_1$ is the first column of $A$, we thus recover the parameters $d_1, c_1, \ldots, c_{N-1}$. Now, we multiply $A$ by the all ones vector: $A\mathbf{1} = y_2$. The $i$th entry of $y_2$ is the sum of the entries in the $i$th row of $A$. Thus, the $i$th entry of $y_2$ is equal to $d_i + \sum_{j=1}^{n-1} c_j$. Having learned $c_1, \ldots, c_{n-1}$ we calculate $y_2(i) - \sum_{j=1}^{N-1} c_j = d_i$ for each $1 \leq i \leq N$. Thus, we recover the diagonal entries of $A$.

The family of matrices in the form of $A$ is linearly parametrized and defined by $2N - 1$ parameters. Then, Lemma 2.1.2 and our algorithm together imply that QC(diagonal-plus-circulant) = 2. Moreover, one could also take the more general approach using Gaussian inputs and solving the associated linear system.

## 2.2 Sparse matrix approximation

Having discussed the exact matrix recovery problem at length, we now consider the matrix approximation problem, which characterizes the best (or near-best)

structured approximation to a general matrix.

We focus on the task of approximating $A$ with a matrix of a *specified* sparsity pattern, with error competitive with the best approximation of the given sparsity pattern. This is a natural task; indeed, there are many existing linear algebra algorithms for matrices of a given sparsity pattern (e.g. diagonal, tridiagonal, banded, block diagonal, etc.), so it is a common goal to obtain an approximation compatible with such algorithms. As we discuss in Section 2.2.2, several important special cases including diagonal approximation and exact recovery of matrices with known-sparsity have been studied extensively in prior work.

Formally, using "$\circ$" to indicate the Hadamard (entrywise) product and $\|\cdot\|_{\mathsf{F}}$ to denote the Frobenius norm, we consider the following problem:

**Problem 1** (Best approximation by a matrix of fixed sparsity)**.** *Given a matrix $A \in \mathbb{R}^{n\times d}$ and a binary matrix $S \in \{0,1\}^{n\times d}$, find a matrix $\widetilde{A}$ so that $\widetilde{A} = S \circ \widetilde{A}$ and*

$$\|A - \widetilde{A}\|_{\mathsf{F}} \leq (1+\varepsilon)\,\|A - S\circ A\|_{\mathsf{F}}.$$

Observe that $S \circ A$ is the matrix of sparsity $S$ nearest to $A$ in the Frobenius norm:

$$S \circ A = \operatorname*{argmin}_{X = S\circ X} \|A - X\|_{\mathsf{F}}.$$

Hence, Problem 1 is asking for a *near-optimal* approximation to $A$ of the given sparsity $S$.[3] In particular, if $A$ already has sparsity pattern $S$, then $A = S \circ A$, so solving Problem 1 will recover $A$ exactly.

In addressing Problem 1, it will also be beneficial to consider the closely related problem of recovering the "sparse-part" of a matrix:

[3]This is reminiscent of the low-rank approximation problem, in which we aim to find a rank-$k$ approximation to $A$ competitive with the best rank-$k$ approximation [85, 166].

**Problem 2** (Best approximation to on-sparsity-pattern entries)**.** *Given a matrix* $A \in \mathbb{R}^{n\times d}$ *and a binary matrix* $S \in \{0,1\}^{n\times d}$*, find a matrix* $\widetilde{A}$ *so that* $\widetilde{A} = S \circ \widetilde{A}$ *and*

$$\|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2 \leq \varepsilon \|A - S \circ A\|_{\mathsf{F}}^2.$$

Note the presence of the squared norms in Problem 2. Since $\widetilde{A}$ has the same sparsity as $S$, i.e. $\widetilde{A} = S \circ \widetilde{A}$,

$$\|A - \widetilde{A}\|_{\mathsf{F}}^2 = \|A - S \circ A\|_{\mathsf{F}}^2 + \|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2. \tag{2.5}$$

Thus, using the fact that $\sqrt{1+2\varepsilon} < 1 + \varepsilon$ for all $\varepsilon > 0$, a solution to Problem 2 with accuracy $2\varepsilon$ immediately yields a solution to Problem 1 with accuracy $\varepsilon$. Conversely, if $\varepsilon \in (0,1)$ so that $\sqrt{1+3\varepsilon} \geq 1 + \varepsilon$, then a solution to Problem 1 with accuracy $\varepsilon$ yields a solution to Problem 2 with accuracy $3\varepsilon$. In this sense, the problems are equivalent.

### 2.2.1 Roadmap

Our first contribution in this section is to analyze a simple algorithm (Algorithm 3) that solves Problems 1 and 2. When the sparsity pattern $S$ has at most $s$ non-zero entries per row, this algorithm uses $m = O(s/\varepsilon)$ non-adaptive matrix-vector product queries. Specifically, the algorithm computes $Z = AG$, where $G$ is a $d \times m$ matrix with independent standard normal entries, and then outputs the matrix

$$\widetilde{A} = \operatorname*{argmin}_{X = S \circ X} \|Z - XG\|_{\mathsf{F}}. \tag{2.6}$$

We make no claims about the novelty of the algorithm, which follows immediately from ideas in compressed sensing. In Section 3.2.4, using standard tools from

random matrix theory and high dimensional probability, we provide an analysis of Algorithm 3 and prove the following:

**Theorem 2.2.1.** *Consider any $A \in \mathbb{R}^{n\times d}$ and any $S \in \{0,1\}^{n\times d}$ with at most $s$ nonzero entries per row. Then, for any $m \geq s+2$, using $m$ randomized matrix-vector queries, Algorithm 3 returns a matrix $\widetilde{A}$ with $\widetilde{A} = S \circ \widetilde{A}$ and $\mathbb{E}[\widetilde{A}] = S \circ A$, satisfying*

$$\mathbb{E}\Big[\|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2\Big] \leq \frac{s}{m-s-1}\|A - S \circ A\|_{\mathsf{F}}^2.$$

*The above inequality is equality if each row of $S$ has exactly $s$ non-zero entries.*

Owing to Equation (2.5) and Jensen's inequality, Theorem 2.2.1 also gives an expectation bound for Problem 1. Setting $m = O(s/\varepsilon)$ implies that Problems 1 and 2 are solved in expectation. Using Markov's inequality, we derive a probability bound:

**Corollary 2.2.2.** *In the setting of Theorem 2.2.1, for any $\varepsilon > 0$ and $\delta \in (0,1)$, if $m \geq s+2$ then*

$$m \geq s\left(\frac{1}{2\delta\varepsilon}+1\right)+1 \quad \Longrightarrow \quad \mathbb{P}\Big[\|A - \widetilde{A}\|_{\mathsf{F}} \geq (1+\varepsilon)\,\|A - S \circ A\|_{\mathsf{F}}\Big] \leq \delta.$$

Hence, using $O(s/\varepsilon)$ Algorithm 3 solves Problem 1 except with small constant probability, say $\leq 1/100$.

We proceed, in Section 3.2.9, to study lower bounds for the matvec query complexity of Problems 1 and 2. We show that up to constant factors, the upper bound in Corollary 2.2.2 is optimal, even for the stronger class of adaptive matvec query algorithms:

**Theorem 2.2.3.** *Fix $\gamma \in (0,1)$. Then there exist constants $c, C > 0$ (depending only on $\gamma$) such that the following holds:*

*For any $\varepsilon \in (0, c)$ and integer $s \geq 1$, there is a distribution on (symmetric) matrices $A \in \mathbb{R}^{d \times d}$ such that, for any sparsity pattern $S$ whose rows and columns each have between $\gamma s$ and $s$ nonzero entries, and for any (possibly randomized) algorithm that uses $m < Cs/\varepsilon$ (possibly adaptive) matrix-vector queries to $A$ to output $\widetilde{A}$ with $\widetilde{A} \circ S = \widetilde{A}$,*

$$\mathbb{P}\Big[\|A - \widetilde{A}\|_{\mathsf{F}} \leq (1 + \varepsilon) \|A - S \circ A\|_{\mathsf{F}}\Big] \leq \frac{1}{25}.$$

Thus, $\Omega(s/\varepsilon)$ queries are required to solve Problem 1 with any reasonable probability. Note that since solving Problem 1 gives a solution to Problem 2, the lower bound Theorem 2.2.3 also implies that $\Omega(s/\varepsilon)$ queries are required to solve Problem 2. Our approach uses an invariance property of Wishart matrices after a sequence of adaptive queries [37]. Note that since our hard instance is symmetric, Theorem 2.2.3 also holds against algorithms which are allowed to use matvec queries with $A^\top$.

Our lower bound applies even to the well-studied case of best approximation by a diagonal matrix, and more broadly, to approximation by a banded matrix. To the best of our knowledge, our lower bounds are the first (adaptive or non-adaptive) for Problems 1 and 2 even for these special cases. In Section 2.2.7[4] we present several numerical experiments on test problems to illustrate the sharpness of our upper bound, and in Section 2.2.8 we discuss the potential for future work, including the potential for algorithms which combine the algorithm in this paper with coloring algorithms to obtain more robustness to noise.

[4]tyler or noah did these, but we include them for the exposition

### 2.2.2 Past work

To the best of our knowledge, Problems 1 and 2 have not been previously stated explicitly in the given generality. However, there is a range of past work that studies special cases of these problems. We categorize these works into the zero error case, where $A = S \circ A$, and the nonzero error case, where $A \neq S \circ A$. In addition, we discuss how Problems 1 and 2 differ from the standard sparse-recovery problem in compressed sensing.

#### Zero error

Some of the earliest work relating to Problems 1 and 2 seeks to recover Jacobian and Hessian matrices with a known sparsity pattern from matvecs [55, 53, 52]. These methods make use of the fact that a *graph coloring* of a particular graph, induced by the sparsity pattern of $A$, can be used to obtain a set of query vectors which are sufficient to exactly recover $A$. In many (but not all) cases, exact recovery is possible with $s$ queries. Coloring methods have also influenced many algorithms for the nonzero error setting. We compare our results to such coloring-based methods in [6, Section 4], arguing that Algorithm 3 always performs better in the zero-error setting, since it always requires just $s$ queries.

More generally, [84] studies the matvec query complexity of exact recovery of a wide range of linearly parameterized matrix families, proving matching upper and lower-bounds on the number of queries required. In particular, it is shown that recovering a diagonal matrix requires one query, recovering a block-diagonal matrix with $s \times s$ blocks requires $s$ queries, and recovering a tridiagonal matrix requires 3 queries. Recovering a general $n \times d$ matrix is shown to require $d$ queries. With

probability one, Algorithm 3 matches these lower bounds (see Proposition 2.2.4).

### Nonzero error

The most theoretically well-studied instance of Problem 2 is arguably the case $S = I$; i.e. the task of approximating the diagonal of a matrix. For this task, it is common to use Hutchinson's diagonal estimator, defined as

$$d_m = \Big[\sum_{j=1}^{m} r_j \circ (Ar_j)\Big] \oslash \Big[\sum_{j=1}^{m} r_j \circ r_j\Big], \tag{2.7}$$

where "$\oslash$" indicates entrywise division and the entries of the vectors $r_j$ are all independent random variables with mean zero and variance one. A number of analyses of this estimator have been given for various distributions [19, 160, 15, 86, 61]. In particular, [15, 61] give error bounds for Problem 2, showing it suffices to set $m = O(1/\varepsilon)$, matching Theorem 2.2.1 up to constant factors. In fact, when $S = I$ our Algorithm 3 is equivalent to Equation (2.7) if the query vectors $r_i$ are Gaussian. We detail this connection in Remark 3.

Past work has also studied Problem 1 with the goal of approximating a potentially non-sparse matrix by a sparse matrix. For instance, there is a long line of work on approximating matrix functions $A = f(H)$ from matvecs. If $H$ is sparse, then the entries of $A = f(H)$ decay exponentially away from the nonzero entries of $H$ under mild assumptions on $f(x)$ [60, 21, 20]. As such, it is reasonable to approximate $A$ with a sparse matrix of a sparsity similar to $H$. This observation has been used in matrix approximation algorithms [160, 154, 72, 138]. Broadly speaking, these algorithms aim to combine the coloring methods described above with the estimator $d_m$ described in Equation (2.7). For banded matrices, one can use the existing analyses of $d_m$ to analyze the performance of these methods, showing

that they solve Problem 2 to accuracy $\varepsilon$ using $O(s/\varepsilon)$ matrix-vector queries. We include a note on this in [6, Section 4.1], as we were unable to find such an analysis in the literature. Theorem 2.2.1 shows that Algorithm 3 matches this bound for arbitrary sparsity patterns.

More recently, motivated by the field of partial differential equation (PDE) learning, there has been widespread interest in learning the solution operators of PDEs from input-output data of forcing terms and solutions, analogous to matrix-vector products [150, 151, 32, 95]. The method in [151] obtains a fixed-sparsity approximation to the sparse Cholesky factorization of the solution operator by coloring, which is provably accurate for certain problems. This makes use of the fact that in certain settings a fixed-sparsity Cholesky factorization is accurate and can be efficiently computed [150]. This is broadly related to the (factorized) sparse approximate inverse problem for obtaining preconditioners [23]. The method in [32] also derives a continuous analogue of a generalized coloring algorithm for targeting low-rank subblocks of hierarchical matrices [106], and then recovering these subblocks using the randomized SVD [85]. The final step of this algorithm reduces to the recovery of a block diagonal matrix.

**Relationship to sparse recovery and compressed-sensing**

It is important to contrast the aims and results of this paper with the rich literature on sparse recovery and compressed sensing [65, 71, etc.]. Most relevant to our work are results on the $\ell_2/\ell_2$ sparse recovery problem. Given access to a length $d$ vector $a$ through linear measurements $a$: $M \mapsto Ma$, the goal of the $\ell_2/\ell_2$ sparse recovery problem is to obtain an $s$-sparse vector $\widetilde{a}$ for which

$$\|a - \widetilde{a}\|_2 \leq (1+\varepsilon) \min_{a' \text{ } s\text{-sparse}} \|a - a'\|_2.$$

Algorithms for solving this sparse recovery problem can be immediately adapted to matrix recovery [138]. In particular, by applying $\ell_2/\ell_2$ sparse recovery to all rows in a matrix $A$ (which requires evaluating the matvec queries $AM^T$), we can recover an $s$-sparse approximation $\tilde{A}$ satisfying $\|\tilde{A} - A\|_F \leq (1+\epsilon)\|A - S \circ A\|_F$. I.e., we can solve Problem 1.

However, this approach has a number of disadvantages. For one, if sparse recovery is used as a black-box, while the approximation $\tilde{A}$ will be sparse, it cannot be guaranteed to be non-zero in the exact locations specified by $S$. This is a disadvantage when the goal is to approximate $A$ by a matrix that has useful computational properties due to a specific sparsity pattern (e.g., by a matrix that is easily invertible like a diagonal or banded matrix).

Moreover, sparse recovery algorithms necessarily have worse dependencies on $\varepsilon$ and $d$ than the bounds we prove for the Algorithm 3. In particular, it can be shown that any algorithm solving the $\ell_2/\ell_2$ sparse recovery problem necessarily requires $\Omega(s/\varepsilon^2 + s\log(d/s)/\varepsilon)$ linear measurements, which would equate to $\Omega(s/\varepsilon^2 + s\log(d/s)/\varepsilon)$ matrix-vector products to obtain a near-optimal approximation to $A$ with $s$-sparse rows [143]. In contrast, our upper-bound of $O(s/\varepsilon)$ from Theorem 2.2.1/Corollary 2.2.2 has both a better dependence on $\varepsilon$, and no dependence on the dimension $d$ at all.

A number of past works [173, 179, 56, etc.] have also studied a matrix version of the sparse recovery problem in which one aims to obtain an $sn$-sparse matrix $\widetilde{A}$ for which

$$\|A - \widetilde{A}\|_{\mathsf{F}} \leq (1+\varepsilon) \min_{X \ sn\text{-sparse}} \|A - X\|_{\mathsf{F}},$$

using bi-linear measurements of $A$: $(U, V) \mapsto UAV^\top$. Again, while related to our problem, prior results for this task inherently involve worse dependencies on $\epsilon$ and

the dimension than our results, and do not return an approximation $\tilde{A}$ with a specified sparsity pattern.

Finally, we remark that, while there has been significant work on *lower bounds* for sparse recovery [9, 143], such bounds do not imply lower bounds akin to our Theorem 2.2.3 for Problems 1 and 2. In particular, the hardness of vector sparse recovery is critically tied to the fact that the entries of $\tilde{a}$ are not specified in advance. Indeed, we can trivially find the best $s$-sparse approximation to a vector $a$ with a specified sparsity pattern using $s$ linear measurements (just read the corresponding entries in $a$). There is no dependence on $\epsilon$ at all. Our lower bound of $\Omega(s/\epsilon)$ from Theorem 2.2.3 must leverage a different source of hardness: in the matrix setting, while the sparsity pattern of $\tilde{A}$ is specified in advance, the difficulty of the problem comes from the fact that the sparsity pattern can *differ* across rows. We must use this fact to obtain the lower bound of Theorem 2.2.3, which depends on both $s$ and $\epsilon$.

### 2.2.3 A sparse approximation algorithm and upper bound

We begin by writing down an explicit algorithm (Algorithm 3) for solving Problems 1 and 2. This algorithm proceeds row-by-row, taking a advantage of the fact that different rows of the solution to Equation (2.6) do not depend on one another (except through the common use of $Z = AG$). For each row, we can solve for the entries of $\widetilde{A}$ via an appropriate least squares problem.

We introduce some notation for Algorithm 3. We use Gaussian$(n, d)$ (or Gaussian$(d)$ if $n = d$) to denote the distribution on $n \times d$ matrices, where each entry of the matrix is independent and identically distributed (iid) with distribution

$\mathcal{N}(0,1)$. We denote the complement of a set $S$ as $S^{\mathsf{c}}$ (determined from context). For $d \geq 1$, we define $[d] = \{1, 2, \ldots, d\}$, and for $R \subset [n]$ and $C \subset [d]$, $[X]_{R,C}$ indicates the $|R| \times |C|$ submatrix of a $n \times d$ matrix $X$ corresponding to the rows in $R$ and columns in $C$. If $R$ or $C$ contain only one element, we will simply write this element. Likewise, when $R = [n]$ or $C = [d]$, we will use a colon, e.g., $[X]_{1,:}$ is the first column of $X$.

**Algorithm 1** Fixed-sparse-matrix recovery

**procedure** FIXED-SPARSE-MATRIX-RECOVERY($A, S, m$)
  Form $G \in \mathbb{R}^{d \times m}$, $G_{ij} \sim \mathcal{N}(0,1)$ iid, $1 \leq i \leq d, 1 \leq j \leq m$ ▷ $d \times m$ iid Gaussian matrix
  Compute $Z = AG$ ▷ $m$ non-adaptive matvec queries
  **for** $i = 1, 2, \ldots, n$ **do**
    Let $S_i = \{j : [S]_{i,j} = 1\}$ ▷ nonzero entries of $i$th row of $S$
    Let $z_i^\top = [Z]_{i,:}$ ▷ $i$-th row of $Z$
    Let $G_i^\top = [G]_{S_i,:}$ ▷ submatrix formed by taking the rows from $S_i$
    Compute $\widetilde{a}_i = G_i^\dagger z_i$ ▷ solve a $m \times |S_i|$ least squares problem
    Set $[\widetilde{A}]_{i,S_i} = \widetilde{a}_i^\top$ and $[\widetilde{A}]_{i,S_i^{\mathsf{c}}} = 0^\top$ ▷ construct $i$th row of $\widetilde{A}$
  **end for**
  **return** $\widetilde{A}$
**end procedure**

Note that in the case that $A$ has nonzeros only in positions where $S$ is nonzero, Algorithm 3 will exactly recover $A$ as long as the least squares problem for each row is fully determined.

**Proposition 2.2.4.** *If $S \circ A = A$ and $m \geq s$, then Algorithm 3 returns a matrix $\widetilde{A} = A$ with probability one.*

*Proof.* Consider the $i$-th row, and let $x_i \in \mathbb{R}^{|S_i|}$ be the set of non-zero entries in that row. The corresponding $G_i \in \mathbb{R}^{m \times |S|}$ is full rank $|S_i|$ with probability one. Observe that $z_i = G_i x_i$. Thus, since $G_i$ is full-rank, $\widetilde{a}_i = G_i^\dagger z = G_i^\dagger G_i x_i = x_i$.

Thus, we recover the row exactly. By a union bound, with probability one, this simultaneously happens for all the rows. □

Our main focus will be the case where $A$ may have nonzeros off of the specified sparsity pattern. We first recall a standard result from high dimensional probability.

**Proposition 2.2.5.** *Let $G \in \mathbb{R}^{p\times q}$ have entries sampled as $G_{ij} \sim \mathcal{N}(0,1)$ iid for all $1 \le i \le p, 1 \le j \le q$. Then, for compatible matrices $X$ and $Y$,*

$$\mathbb{E}\big[\|XGY\|_{\mathsf{F}}^2\big] = \|X\|_{\mathsf{F}}^2\|Y\|_{\mathsf{F}}^2. \tag{2.8}$$

*Moreover, if $p - q \ge 2$, then*

$$\mathbb{E}\big[\|G^\dagger\|_{\mathsf{F}}^2\big] = \frac{q}{p-q-1}. \tag{2.9}$$

*Proof.* The expression Equation (2.8) is an elementary calculation; see for instance [85, Proposition A.1]. The expression Equation (2.9) follows from the fact that $G^\top G$ is invertible with probability one, and $(G^\top G)^{-1}$ has a inverse Wishart distribution, which, for $p - q \ge 2$ has mean $I/(p-q-1)$ [127, §3.2 (12)]. Since $\|G^\dagger\|_{\mathsf{F}}^2 = \mathrm{tr}((G^\dagger)^\top(G^\dagger)) = \mathrm{tr}((G^\dagger)^\top(G^\dagger))$, the result follows from the linearity of the expectation; see for instance [85, Proposition A.5]. □

Using Theorem 3.2.9, we establish the following theorem.

**Theorem 2.2.1.** *Consider any $A \in \mathbb{R}^{n\times d}$ and any $S \in \{0,1\}^{n\times d}$ with at most $s$ nonzero entries per row. Then, for any $m \ge s+2$, using $m$ randomized matrix-vector queries, Algorithm 3 returns a matrix $\widetilde{A}$ with $\widetilde{A} = S \circ \widetilde{A}$ and $\mathbb{E}[\widetilde{A}] = S \circ A$, satisfying*

$$\mathbb{E}\Big[\|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2\Big] \le \frac{s}{m-s-1}\|A - S \circ A\|_{\mathsf{F}}^2.$$

*The above inequality is equality if each row of $S$ has exactly $s$ non-zero entries.*

We also obtain a probability bound:

**Corollary 2.2.2.** *In the setting of Theorem 2.2.1, for any $\varepsilon > 0$ and $\delta \in (0,1)$, if $m \geq s + 2$ then*

$$m \geq s\left(\frac{1}{2\delta\varepsilon} + 1\right) + 1 \quad \Longrightarrow \quad \mathbb{P}\Big[\|A - \widetilde{A}\|_{\mathsf{F}} \geq (1+\varepsilon)\,\|A - S \circ A\|_{\mathsf{F}}\Big] \leq \delta.$$

*Proof of Theorem 2.2.1.* The algorithm processes $Z = AG \in \mathbb{R}^{n\times m}$ sequentially to approximate the rows of $A$. Fix $i$ and let $S_i$ be the indices of the nonzero entries of $[S]_{i,:}$ and $z_i^\top = [Z]_{i,:}$ be the $i$-th row of $Z$ and $S_i^{\mathsf{c}} = [d] \setminus S_i$. Let $G_i^\top = [G]_{S_i,:}$ and $\widehat{G}_i^\top = [G]_{S_i^{\mathsf{c}},:}$ be submatrices of $G$ formed by taking the rows of $G$ in $S_i$ and $S_i^{\mathsf{c}}$ respectively. Define $x_i^\top = [A]_{i,S_i}$ and $y_i^\top = [A]_{i,S_i^{\mathsf{c}}}$ and observe that

$$z_i^\top := [AG]_{i,:} = x_i^\top G_i^\top + y_i^\top \widehat{G}_i^\top.$$

To enforce the sparsity pattern, Algorithm 3 tries to recover $x_i \in \mathbb{R}^s$ from $z_i \in \mathbb{R}^m$ by solving the least squares problem:

$$\widetilde{a}_i := G_i^\dagger z_i = G_i^\dagger(G_i x_i + \widehat{G}_i y_i) = x_i + G_i^\dagger \widehat{G}_i y_i.$$

Here we have used that $G_i$ is full-rank with probability one.

Since $G_i$ and $\widehat{G}_i$ are independent, clearly $\mathbb{E}[\widetilde{a}_i] = x_i$ as $\mathbb{E}[\widehat{G}_i] = 0$. Thus, Algorithm 3 outputs an unbiased estimator for $S \circ A$.

As long as $m \geq |S_i| + 2$, it follows from standard results in random matrix

theory that

$$\begin{aligned}
\mathbb{E}\big[\|x_i - \widetilde{a}_i\|_2^2\big] &= \mathbb{E}\big[\|G_i^\dagger \widehat{G}_i y_i\|_2^2\big] \\
&= \mathbb{E}\big[\mathbb{E}\big[\|G_i^\dagger \widehat{G}_i y_i\|_2^2 \,\big|\, G_i\big]\big] \\
&= \mathbb{E}\big[\|G_i^\dagger\|_{\mathsf{F}}^2 \cdot \|y_i\|_2^2\big] && \text{Equation (2.8) in Theorem 3.2.9} \\
&= \frac{|S_i|}{m - |S_i| - 1} \cdot \|y_i\|_2^2 && \text{Equation (2.9) in Theorem 3.2.9} \\
&\le \frac{s}{m - s - 1} \cdot \|y_i\|_2^2,
\end{aligned}$$

where we have used that $|S_i| \le s$ in the final line (and hence we have equality if $|S_i| = s$).

Let $\widetilde{A}$ be the output of Algorithm 3. Then, by the linearity of expectation,

$$\begin{aligned}
\mathbb{E}\Big[\|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2\Big] &= \sum_{i=1}^n \mathbb{E}\Big[\|x_i - \widetilde{a}_i\|_2^2\Big] \\
&\le \frac{s}{m - s - 1} \sum_{i=1}^n \|y_i\|_2^2 \\
&= \frac{s}{m - s - 1} \|A - S \circ A\|_{\mathsf{F}}^2.
\end{aligned}$$

Observe that we have equality if $|S_i| = s$ for each row $i \in [n]$. □

*Proof of Corollary 2.2.2.* Applying Markov's inequality to Theorem 2.2.1, we find

$$\mathbb{P}\big[\|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2 \ge \alpha\big] \le \frac{s}{m - s - 1} \frac{\|A - S \circ A\|_{\mathsf{F}}^2}{\alpha}. \tag{2.10}$$

Set $\alpha = 2\varepsilon \|A - S \circ A\|_{\mathsf{F}}^2$. Then, using that $\sqrt{1 + 2\varepsilon} \le 1 + \varepsilon$ for all $\varepsilon > 0$ and recalling Equation (2.5) gives that

$$\begin{aligned}
\mathbb{P}\big[\|A - \widetilde{A}\|_{\mathsf{F}} \ge (1 + \varepsilon)\|A - S \circ A\|_{\mathsf{F}}\big] &\le \mathbb{P}\big[\|A - \widetilde{A}\|_{\mathsf{F}}^2 \ge (1 + 2\varepsilon)\|A - S \circ A\|_{\mathsf{F}}^2\big] \\
&= \mathbb{P}\big[\|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2 \ge 2\varepsilon \|A - S \circ A\|_{\mathsf{F}}^2\big] \\
&\le s / \big((m - s - 1)(2\varepsilon)\big).
\end{aligned}$$

By assumption $m \ge s(1/(2\delta\varepsilon) + 1) + 1$, which gives the result. □

We now make several comments about Algorithm 3 and our analysis.

**Remark 1.** *Our bound in Corollary 2.2.2 has an unfavorable $O(1/\delta)$ dependence on the failure probability $\delta$. One could apply Markov's inequality to each row and Hoeffding's inequality to the sum to obtain a dependence $O(\log(n/\delta))$. However this has a dependence on the dimension $n$ which we would like to avoid. In [6, Appendix C], we show that one can apply a high-dimensional analog of the "median trick" to obtain an algorithm with a $O(\log(1/\delta))$ failure probability (without any dependence on the dimensions $n$ and $d$).*

**Remark 2.** *If $A$ and $S$ are symmetric, then it is better to return $(\widetilde{A}+\widetilde{A}^\top)/2$ than $\widetilde{A}$ since, by the triangle inequality,*

$$\|A - (\widetilde{A} + \widetilde{A}^\top)/2\|_{\mathsf{F}} = \|(A - \widetilde{A})/2 + (A - \widetilde{A})^\top/2\|_{\mathsf{F}} \leq \|A - \widetilde{A}\|_{\mathsf{F}}.$$

**Remark 3.** *If the entries of the $r_i$ are Gaussian, then the diagonal estimator $d_m$ from Equation (2.7) is equivalent to Equation (2.6) with $S = I$. Let $r_j$ denote the $j$th column of $G$. By definition, $z_i = [[Ar_1]_i, \ldots, [Ar_m]_i]^\top$ and in this case,*

$$G_i^\top := [G]_{S_i,:} = [G]_{i,:} = [[r_1]_i, \ldots, [r_m]_i]$$

*is a vector. The $i$-th row of $d_m$ is*

$$[d_m]_i := \frac{\sum_{j=1}^m [r_j]_i \cdot [Ar_j]_i}{\sum_{j=1}^m [r_j]_i^2} = \frac{G_i^\top z_i}{G_i^\top G_i} = G_i^\dagger z.$$

*In this sense, Algorithm 3 for computing Equation (2.6) is a generalization of Equation (2.7) to non-diagonal sparsity patterns. Interestingly, however, we have not seen Equation (2.7) interpreted in terms of a least-squares problem or pseudoinverse in the literature. This is perhaps because past work focused on diagonal estimation (Problem 2) rather than approximation by a diagonal (Problem 1).*

**Remark 4.** *Algorithm 3 requires solving $n$ least squares problems with a coefficient matrix of size $m \times s$. So, in addition to the application dependent cost of*

*computing $Z = AG$, its runtime is just $O(nms^2)$. There are a number of practical improvements which can be made upon implementation. First, for many sparsity patterns, the matrices $G_i$ and $G_{i+1}$ differ only by a permutation and low-rank update. Thus, by downdating/updating appropriate quantities, the cost of solving all $n$ least-squares problems may be lower than $n$ times the cost of solving a single system. In addition, a posteriori variance estimates could also be obtained through Jack-knife type techniques [67].*

### 2.2.4 A lower-bound for adaptive algorithms

Algorithm 3 solves Problem 1 using $O(s/\varepsilon)$ matvec queries. In this section, we show that there are distributions of matrices and sparsity patterns for which no matvec query algorithm can reliably solve Problem 1 using $\Omega(s/\varepsilon)$ matvecs. In particular, we show the following:

**Theorem 2.2.3.** *Fix $\gamma \in (0, 1)$. Then there exist constants $c, C > 0$ (depending only on $\gamma$) such that the following holds:*

*For any $\varepsilon \in (0, c)$ and integer $s \geq 1$, there is a distribution on (symmetric) matrices $A \in \mathbb{R}^{d \times d}$ such that, for any sparsity pattern $S$ whose rows and columns each have between $\gamma s$ and $s$ nonzero entries, and for any (possibly randomized) algorithm that uses $m < Cs/\varepsilon$ (possibly adaptive) matrix-vector queries to $A$ to output $\widetilde{A}$ with $\widetilde{A} \circ S = \widetilde{A}$,*

$$\mathbb{P}\Big[\|A - \widetilde{A}\|_{\mathsf{F}} \leq (1 + \varepsilon) \|A - S \circ A\|_{\mathsf{F}}\Big] \leq \frac{1}{25}.$$

This implies that for certain hard instances of Problems 1 and 2, our Corollary 2.2.2 and thus Theorem 2.2.1 are optimal up to constants. In particular,

adaptivity can only improve constants; it will not lead to an improved dependence on $s$ or $\epsilon$. If $A$ is known to have a particular structure, it is possible that adaptive algorithms may perform better than non-adaptive algorithms for these problems.

We note that the condition $\varepsilon < c$ is benign. In particular, if $C < c/2$, then $m \leq Cs/\varepsilon$ implies $m \leq s/2$, in which case one cannot solve Problem 1, even in the zero error case, due to a parameter counting argument.

### 2.2.5 Key technical tools

Before we prove Theorem 2.2.3, we introduce several key results.

Our hard distribution will be $A = G^\top G$, where $G \in \mathbb{R}^{d \times d}$ has entries sampled as $G_{ij} \sim \mathcal{N}(0,1)$ iid for all $1 \leq i, j \leq d$. This is a special case of a so-called Wishart matrix. Our lower bound will make use of the fact that the conditional distribution of a Wishart matrix after a sequence of adaptive matrix-vector queries still looks like a slightly smaller transformed Wishart matrix [37, Lemma 3.4]. Similar hard input distributions have been used in a number of lower-bounds for matvec query tasks [153, 37, 91, 48]. We believe other simple distributions such as $A = G$ or $A = G + G^\top$ would also suffice to prove something like Theorem 2.2.3. We have chosen to use $A = G^\top G$ because it is symmetric, and it allows us to use a conceptually intuitive anti-concentration result based on the Berry–Esseen theorem.

The following is essentially Lemma 3.4 from [37], restated to suit our needs:

**Proposition 2.2.6.** *Suppose $G \in \mathbb{R}^{d \times r}$ with random entries $G_{ij} \sim \mathcal{N}(0,1)$ iid for all $1 \leq i \leq d$, $1 \leq j \leq r$. Let $x_1, \ldots, x_m$ and $y_1 = G^\top G x_1, \ldots, y_m = G^\top G x_m$ be*

*such that, for each* $j = 1, \ldots, m$, $x_j$ *was chosen based only on the query vectors* $x_1, \ldots, x_{j-1}$ *and the outputs* $y_1, \ldots, y_{j-1}$.

*Then, there is an* $n \times n$ *orthonormal matrix* $V_m$ *and an* $n \times n$ *matrix* $\Delta_m$, *each constructed solely as functions of* $x_1, \ldots, x_m$ *and* $y_1, \ldots, y_m$, *and a matrix* $G_m \in \mathbb{R}^{d-m \times r-m}$ *with entries* $G_{m_{ij}} \sim \mathcal{N}(0,1)$ *iid for all* $1 \leq i \leq d-m, 1 \leq j \leq r-m$, *independent of* $x_1, \ldots, x_m$ *and* $y_1, \ldots, y_m$ *such that*

$$V_m^\top G^\top G V_m = \Delta_m + \begin{bmatrix} 0_{m,m} & 0_{m,d-m} \\ 0_{d-m,m} & G_m^\top G_m \end{bmatrix}.$$

This result is essentially Lemma 3.4 from [37], and the proof can be found in the appendix of [6, Appendix B.1]. We will also use the following bound about the anti-concentration of independent random variables, which is an immediate consequence of the Berry–Esseen Theorem and a basic anti-concentration result for Gaussians. The proof can be found in [6, Appendix B.2].

**Proposition 2.2.7.** *There exists a constant* $C > 0$ *such that, if* $X_1, \ldots, X_k$ *are independent random variables with* $\mathbb{V}[X_i] \geq \sigma^2$, *and* $\mathbb{E}[|X_i - \mathbb{E}[X_i]|^3] \leq \rho$, *and if we define*

$$X = X_1 + \cdots + X_k,$$

*then for any* $t \in \mathbb{R}$ *and* $\alpha > 0$, *if* $k > C\rho^2/(\alpha^2\sigma^6)$,

$$\mathbb{P}\Big[|X - t| < \alpha\sigma\sqrt{k}\Big] < \alpha.$$

Using Proposition 2.2.7, we can derive a more specific consequence which we will use directly in the proof of Theorem 2.2.3. The proof of this result is also contained in [6, Appendix B.1].

**Lemma 2.2.8.** *There exists a constant* $C > 0$ *such that, if* $x = Gu$ *and* $y = Gv$, *where* $G \in \mathbb{R}^{k \times k}$ *with* $G_{ij} \sim \mathcal{N}(0,1)$ *iid for all* $1 \leq i, j \leq k$, *and* $u, v \in \mathbb{R}^k$ *such*

*that* $\|u\|_2 \le 1$ *and* $\|v\|_2 \le 1$*, then for any* $\alpha > 0$ *and* $t \in \mathbb{R}$*, if* $k > C/(\alpha^2 \|u\|_2^6 \|v\|_2^6)$ *then*

$$\mathbb{P}\Big[\big|x^\top y - t\big| < \alpha \|u\|_2 \|v\|_2 \sqrt{k}\Big] < \alpha.$$

Finally, we will use the following observation about sparsity patterns that overlap all sufficiently large principal submatrices.

**Lemma 2.2.9.** *Let* $\gamma < 1$ *and suppose* $S \in \{0,1\}^{d \times d}$ *is binary matrix for which each row and column has between* $\gamma s$ *and* $s$ *non-zero entries. Let* $I \subset [d]$ *with* $|I| \ge 2d/(2+\gamma)$*. Then, the principal submatrix* $[S]_{I,I}$ *of* $S$ *contains at least* $\gamma ds/(2+\gamma)$ *nonzero entries.*

*Proof.* Let $I \subset [d]$ with $|I| \ge 2d/(2+\gamma)$. Note that

$$\big\|[S]_{I,[d]}\big\|_{\mathsf{F}}^2 = \sum_{i \in I} \sum_{j \in [d]} [S]_{i,j} \ge |I| \cdot \gamma s = \frac{2\gamma}{2+\gamma} ds.$$

Next, note that, since $|I^{\mathsf{c}}| \le d - 2d/(2+\gamma) = \gamma d/(2+\gamma)$,

$$\big\|[S]_{I,I^{\mathsf{c}}}\big\|_{\mathsf{F}}^2 = \sum_{i \in I} \sum_{j \in I^{\mathsf{c}}} [S]_{i,j} \le |I^{\mathsf{c}}| \cdot s \le \frac{\gamma}{2+\gamma} ds.$$

Finally, since $[d]$ is partitioned into $I$ and $I^{\mathsf{c}}$,

$$\big\|[S]_{I,I}\big\|_{\mathsf{F}}^2 = \big\|[S]_{I,[d]}\big\|_{\mathsf{F}}^2 - \big\|[S]_{I,I^{\mathsf{c}}}\big\|_{\mathsf{F}}^2 \ge \frac{2\gamma}{2+\gamma} ds - \frac{\gamma}{2+\gamma} ds = \frac{\gamma}{2+\gamma} ds. \qquad \square$$

### 2.2.6 Proof of Theorem 2.2.3

We now have the tools necessary to prove Theorem 2.2.3. The general strategy will be to show that the conditional distribution of a Wishart matrix after a sequence of (adaptive) queries is hard to approximate. That is, that the on-sparsity entries are anti-concentrated (conditioned on the queries) relative to the off-sparsity mass,

which is $O(d^{3/2})$ with high probability. We will then use this result to prove Theorem 2.2.3.

**Lemma 2.2.10** (Main technical lemma)**.** *Fix $\gamma \in (0,1)$. Then there exist constants $c_1, c_2, c_3 > 0$ (depending only on $\gamma$) such that the following holds:*

*Suppose $A = G^\top G$, where $G \in \mathbb{R}^{r\times d}$ with $G_{ij} \sim \mathcal{N}(0,1)$ iid for all $1 \le i \le r, 1 \le j \le d$. Then, for any sparsity pattern $S$ whose rows and columns each have between $\gamma s$ and $s$ nonzero entries, and for any (possibly randomized) algorithm that uses $m$ (possibly adaptive) matrix-vector queries to $A$ to output $\widetilde{A}$ with $\widetilde{A} \circ S = \widetilde{A}$, if $m \le c_1 d$ and, for any $\alpha \in (0,1)$, $r > m + c_3/\alpha^2$, then*

$$\mathbb{P}\left[\|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2 < c_2\alpha^2 ds(r-m)\right] < \alpha.$$

*Proof.* Fix $\gamma \in (0,1)$. Let $C$ denote the absolute constant in Lemma 2.2.8, define

$$c_1 = \frac{\gamma}{4+2\gamma} \qquad c_2 = \frac{c_1^2}{4} \qquad c_3 = \frac{C}{c_1^6},$$

and suppose $m$, $r$, and $d$ are integers such that

$$m \le c_1 d, \qquad r - m > \frac{c_3}{\alpha^2}.$$

Suppose we do $m$ (possibly adaptive) queries to $A$. Proposition 2.2.6 implies that there exists an $d \times d$ matrix $\Delta$, and $d \times (d-m)$ matrix $V$ with orthonormal columns, both constructed solely as functions of the queries and measurements, and a matrix $\widehat{G} \in \mathbb{R}^{r-m\times d-m}$ with entries distributed as $\widehat{G}_{ij} \sim \mathcal{N}(0,1)$ for all $1 \le i \le r-m$, $1 \le j \le d-m$ independent of the queries and measurements such that

$$A = \Delta + VWV^\top, \qquad W = \widehat{G}^\top \widehat{G}.$$

Let $S$ be any sparsity pattern for which each row and column has between $\gamma s$ and $s$ non-zeros; i.e. for which we can apply Lemma 2.2.9.

Let $T \in \mathbb{R}^{d\times d}$ be any matrix with sparsity $S$ determined solely as a function of the queries and measurements, and hence independent of $G$. Without loss of generality, we will absorb $S \circ \Delta$ into $T$. Note also that it suffices to assume $T$ is deterministic, as the following argument holds for all possible draws of a random $T$ (and by extension, for the expectation over random draws of $T$).

Define the set of indices

$$P = \Big\{(i,j) \in [d] \times [d] : \big|[S \circ VWV^\top]_{i,j} - [T]_{i,j}\big|^2 > (\alpha/2)^2 c_1 (r-m)\Big\},$$

and the event

$$E = \Big\{|P| \geq c_1 ds\Big\}.$$

Note that if $E$ holds, we have that

$$\begin{aligned}\|S \circ A - T\|_{\mathsf{F}}^2 &\geq \sum_{(i,j)\in P} \big|[S \circ VWV^\top]_{i,j} - [T]_{i,j}\big|^2 \\ &\geq |P| \cdot (\alpha/2)^2 c_1 (r-m) \geq c_1 ds \cdot (\alpha/2)^2 c_1 (r-m) = \alpha^2 c_2 ds(r-m),\end{aligned}$$

so it remains to show $\mathbb{P}[E] \geq 1 - \alpha$.

Towards this end, let $v_i$ be the $i$th row of $V$, and define

$$I = \big\{i \in [d] : \|v_i\|_2^2 \geq c_1\big\}, \qquad M = \big\{(i,j) \in I \times I : [S]_{i,j} = 1\big\}.$$

We must have $|I| \geq 2d/(2+\gamma)$. Otherwise, since $V$ has orthonormal columns so that $\|v_i\|_2^2 \leq 1$, we would have:

$$d - m = \|V\|_{\mathsf{F}}^2 = \sum_{i=1}^{d} \|v_i\|_2^2 < \sum_{i\in I^{\mathsf{c}}} c_1 + \sum_{i\in I} 1 < c_1 d + \frac{2d}{2+\gamma} = (1-c_1)d,$$

which contradicts our assumption $m \leq c_1 d \Leftrightarrow d - m \geq (1-c_1)d$. Then, since $|I| \geq 2d/(2+\gamma)$, Lemma 2.2.9 and our assumption on $S$ give that

$$|M| \geq \frac{\gamma ds}{2+\gamma} = 2c_1 ds.$$

We now show that if $(i,j) \in M$ then $(i,j) \in P$ with high probability. Fix arbitrary $(i,j) \in M$ (and note that this means $i,j \in I$). Since $[S]_{i,j} = 1$, note that $[S \circ VWV^\top]_{i,j} = x_i^\top x_j$, where $x_i = \widehat{G} v_i$ and $x_j = \widehat{G} v_j$ and $v_i$ and $v_j$ are the $i$-th and $j$-th rows of $V$ respectively. We also have that $\|v_i\|_2^2, \|v_j\|_2^2 \geq c_1$. With this in mind, our choice of constants implies

$$r - m \geq \frac{c_3}{\alpha^2} = \frac{C}{\alpha^2 c_1^6} \geq \frac{C}{\alpha^2 \|v_i\|_2^6 \|v_j\|_2^6}.$$

Hence, applying Lemma 2.2.8,

$$\begin{aligned}
\mathbb{P}\Big[(i,j) \notin P\Big] &= \mathbb{P}\Big[\big|[S \circ VWV^\top]_{i,j} - [T]_{i,j}\big| < (\alpha/2) c_1 \sqrt{r-m}\Big] \\
&\leq \mathbb{P}\Big[\big|x_i^\top x_j - [T]_{i,j}\big| < (\alpha/2)\|v_i\|_2 \|v_j\|_2 \sqrt{r-m}\Big] < \frac{\alpha}{2}.
\end{aligned}$$

Now, by Markov's inequality, we have that

$$\mathbb{P}\left[\sum_{(i,j) \in M} \mathbb{1}[(i,j) \notin P] \geq \frac{1}{\alpha} \cdot \frac{\alpha}{2} |M|\right] \leq \alpha.$$

Since $|M| \geq 2c_1 ds$, this then implies that

$$\mathbb{P}\big[E\big] = \mathbb{P}\Big[|P| \geq c_1 ds\Big] \geq \mathbb{P}\left[|P| \geq \frac{|M|}{2}\right] \geq 1 - \alpha.$$

This proves the result. □

With Lemma 2.2.10, the proof if Theorem 2.2.3 is straightforward. The basic idea is that if $r = d$ and $\alpha$ is constant then $\|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2$ is typically $\Omega(sd^2)$. However, by simple computation, it is not hard to see that $\|S \circ A\|_{\mathsf{F}}^2 \leq \|A\|_{\mathsf{F}}^2$ is typically $O(d^3)$. Thus, if we set $d = O(s/\varepsilon)$, we typically will have that $\|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2 > \varepsilon \|S \circ A\|$ as long as $m \leq c_1 d = O(s/\varepsilon)$.

*Proof of Theorem 2.2.3.* Fix $\gamma \in (0,1)$ and let $c_1, c_2, c_3$ be the constants from

Lemma 2.2.10. We will make the following assignments:[5]

$$\alpha = \frac{1}{25}, \quad b_1 = 50, \quad b_2 = \frac{c_2\alpha^2(1-c_1)}{6b_1}, \quad b_3 = \max\left\{2, \frac{(1-c_1)b_2\alpha^2}{c_3}\right\}, \quad C_1 = c_1 b_2.$$

Fix $\varepsilon > 0$. We will show that if

$$m \le C_1\frac{s}{\varepsilon}, \qquad \frac{b_2}{\varepsilon} \in \mathbb{Z}, \qquad \varepsilon < \frac{1}{b_3} \le \frac{1}{2}, \tag{2.11}$$

then there is a distribution on matrices such that for any sparsity pattern with between $\gamma s$ and $s$ entries per row,

$$\mathbb{P}\Big[\|A - \widetilde{A}\|_{\mathsf{F}} \le (1+2\varepsilon)\,\|A - S\circ A\|_{\mathsf{F}}\Big] \le \frac{1}{25}. \tag{2.12}$$

The assumption $b_2/\varepsilon \in \mathbb{Z}$ will subsequently be removed.

Towards this end, assume Equation (2.11), and set

$$d = b_2\frac{s}{\varepsilon}, \quad r = d, \quad A = G^\top G, \quad G \in \mathbb{R}^{r\times d}, G_{ij} \sim \mathcal{N}(0,1) \text{ iid for } 1 \le i \le r, 1 \le j \le d.$$

Define events,

$$E = \Big\{\|S\circ A - \widetilde{A}\|_{\mathsf{F}}^2 \ge c_2\alpha^2 ds(r-m)\Big\}, \qquad F = \Big\{\|A - S\circ A\|_{\mathsf{F}}^2 \le b_1 d^3\Big\}.$$

By assumption, $m \le C_1 s/\varepsilon = c_1 d$ and

$$r - m \ge (1-c_1)d = (1-c_1)\cdot b_2\frac{s}{\varepsilon} > (1-c_1)b_2\frac{1}{b_3} = \frac{c_3}{\alpha^2}.$$

Therefore, applying Lemma 2.2.10, we have that $\mathbb{P}[E] \ge 49/50$.

It is easy to show that $\mathbb{E}[\|A - S\circ A\|_{\mathsf{F}}^2] \le d^3$ (see [6, Appendix B.3] for a derivation), so since $b_1 = 50$, an application of Markov's inequality implies $\mathbb{P}[F] \ge 49/50$.

Note that

$$6\varepsilon b_1 d^3 \le 6\varepsilon b_1 d\cdot b_2\frac{s}{\varepsilon}\cdot\frac{r-m}{1-c_1} \le c_2\alpha^2 ds(r-m).$$

[5]We have labeled the constants so that if $c_i$ depends on $c_j$, then $i > j$.

By a union bound, both $E$ and $F$ hold with probability at least $24/25$. Then since $S \circ A$ and $\widetilde{A}$ have disjoint support and $\varepsilon < 1/2$, as noted in Equation (2.5),

$$\|S \circ A - \widetilde{A}\|_{\mathsf{F}}^2 \geq (1 + 6\varepsilon)\|A - S \circ A\|_{\mathsf{F}}^2 \geq (1 + 2\varepsilon)^2 \|A - S \circ A\|_{\mathsf{F}}^2.$$

We now relax the assumption $c_5/\varepsilon \in \mathbb{Z}$. For arbitrary $\varepsilon > 0$, define $\tilde{\varepsilon} = b_2/\lceil b_2/\varepsilon \rceil$. Then $b_2/\tilde{\varepsilon} \in \mathbb{Z}$ and $\tilde{\varepsilon} \leq \varepsilon$. Moreover, if $\varepsilon \leq b_2/2$, then $\varepsilon \leq b_2\tilde{\varepsilon}/(b_2 - \tilde{\varepsilon}) \leq 2\tilde{\varepsilon}$. Set $c = \min\{1/b_3, b_2/2\}$. Therefore, since $1/\varepsilon \leq 1/\tilde{\varepsilon}$, if

$$m \leq C_1 \frac{s}{\varepsilon}, \qquad \varepsilon < c, \tag{2.13}$$

then the conditions in Equation (2.11) hold (with $\tilde{\varepsilon}$), and so we have the result in Equation (2.12) (with $\tilde{\varepsilon}$). Since $\varepsilon \leq 2\tilde{\varepsilon}$, this implies there is a distribution on matrices and sparsity patterns for which the result of the theorem holds.

Thus, the proof is complete with $C = 2C_1$. □

### 2.2.7 Numerical Experiments

In this section, we provide several numerical experiments which illustrate the performance of Algorithm 3. These problems are modeled after similar problems from the literature. Code to reproduce the figures can be found at `https://github.com/tchen-research/fixed_sparsity_matrix_approximation`.

#### Model problem

We consider the matrix $A = M^{-1}$, where $M = \text{tridiag}(-1, 4, -1)$. This class of matrices exhibits exponential decay away from the diagonal and was used in experiments in past work [22, 72].

We take $A$ to be $1000 \times 1000$ and, for varying values of $b \geq 0$, we set $S$ to be a symmetric banded matrix of maximum total bandwidth $2b+1$; i.e. $[S]_{i,j} = \mathbb{1}(|i-j| \leq b)$. We then compute the approximation error $\|A - \widetilde{A}\|_{\mathsf{F}}$ and recovery error $\|S \circ A - \widetilde{A}\|_{\mathsf{F}}$ for the output of Algorithm 3 run using a varying number of matvec queries $m$.

The results are illustrated in Figure 2.1. Here the convergence of Algorithm 3 is matched well by the upper bound in Theorem 2.2.1 for the expected squared error (note that the plot shows the error, not the expected squared error).

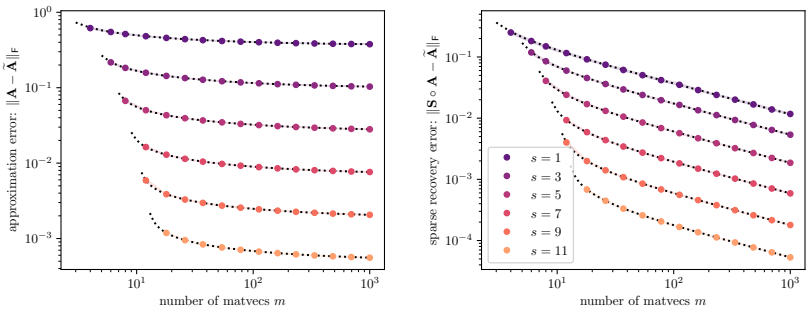



Figure 2.1: Approximation of model problem matrix $A = \mathrm{tridiag}(-1, 4, -1)^{-1}$ by a matrix of total bandwidth $s$ for varying values of $s$. The solid circles indicate the root mean squared error of Algorithm 3 over 20 independent runs of the algorithm, and the shaded region indicates the 10%-90% range. The dotted lines are the $\sqrt{s/(m-s-1)}\|A - S \circ \widetilde{A}\|_{\mathsf{F}}$ (left) and $\sqrt{1+s/(m-s-1)}\|A - S \circ \widetilde{A}\|_{\mathsf{F}}$ (right).

### Trefethen Primes

We let $A = M^{-1}$, where $M$ be the $1000 \times 1000$ matrix whose entries are zero everywhere except for the primes $2, 3, 5, 7, \ldots, 7919$ along the main diagonal and the number 1 in all the positions $[B]_{i,j}$ with $|i–j| \in \{1, 2, 4, 8, \ldots, 512\}$.[6] An example (somewhat different from our example) involving this matrix was used in

[6] In problem 7 of the "A Hundred-dollar, Hundred-digit Challenge" in SIAM News, readers are asked to compute the (1,1) entry of a larger but analogously defined matrix $A$ to 100 digits of accuracy [162].

[138].

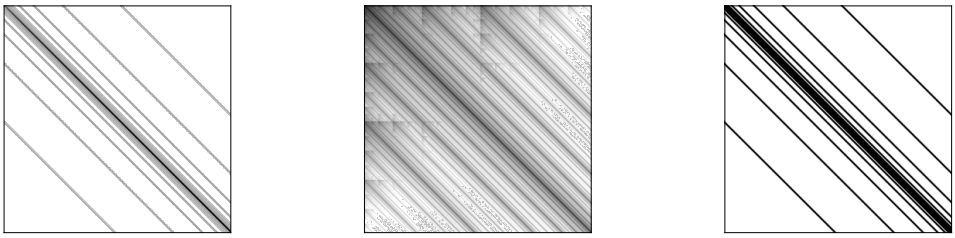


Figure 2.2: *Left*: Log-scale of the nonzero entries of $M$, which range in magnitude from 1 to 7919. *Middle*: Log-scale of the nonzero entries of $A$. *Right*: Sample sparsity pattern $S$ corresponding to $b = 5$.

For $b > 0$, we defined a sparsity pattern $S$ to be such that, for each $t \in \{1, 2, 4, 8, \ldots, 512\}$, $[S]_{i,j} = 1$ whenever $|i - j \pm t| \leq b$ and zero otherwise. In other words, the sparsity pattern consists of bandwidth $2b + 1$ bands centered nonzero entries of $M = A^{-1}$. This is illustrated in Figure 2.2.

The results are shown in Figure 2.3. Here, the convergence of Algorithm 3 is somewhat better than the upper bound Theorem 2.2.1. This is because many rows of the sparsity pattern have far fewer than $s$ entries. From the proof of Theorem 2.2.1, it is clear how to obtain an exact characterization of the expected squared error.

### 2.2.8 Outlook on sparse approximation

This work raises a number of interesting practical and theoretical questions. We now comment on three.

It is clear that there is potential for combining coloring methods with algorithms such as Algorithm 3 in order to avoid paying for large off-sparsity entries in parts

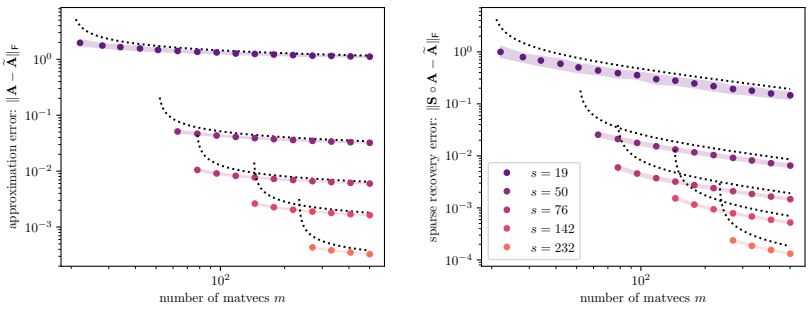



Figure 2.3: Approximation of "Trefethen primes" inverse matrix by a multi-banded matrix for varying values of $s$. The solid circles indicate the root mean squared error of Algorithm 3 over 100 independent runs of the algorithm, and the shaded region indicates the 10%-90% range. The dotted lines are the $\sqrt{s/(m-s-1)}\|A - S \circ \widetilde{A}\|_{\mathsf{F}}$ (left) and $\sqrt{1+s/(m-s-1)}\|A - S \circ \widetilde{A}\|_{\mathsf{F}}$ (right).

of a matrix while simultaneously maintaining the robustness to noise enjoyed by Algorithm 3. One approach to combining these two paradigms is to use adaptive queries to identify portions of the matrix with large-mass, and then find a coloring based on this information. We believe further study in this direction may yield algorithms that work better in many practical situations. Of course, our lower bound Theorem 2.2.3 shows that only constant factors can be improved for some families of problem instances.

The present paper focuses only on the Frobenius norm. It would be valuable to understand the analogous problem in other norms such as the matrix 2-norm. For other norms, $S \circ A$ is not necessarily the best approximation to $A$ with sparsity $S$. For instance, if $A = [1, 1; 1, 1]$ and $S = [1, 0; 0, 0]$, then

$$\operatorname*{argmin}_{X=S\circ X} \|A - X\|_2 = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \neq \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = S \circ A.$$

Loosely speaking, minimizing the operator-norm approximation error requires the rows of the error matrix to be small and unaligned, whereas the Frobenius norm problem only requires them to be small.

Finally, it is of broad interest to understand algorithms and lower bounds for richer structures of matrices, such as the sum of sparse and low-rank matrices and matrices with hierarchical low-rank structure. This paper is a starting point for investigating these problems, and we hope that future work will explore them in greater depth.

CHAPTER 3

# HIERARCHICAL MATRIX RECOVERY AND APPROXIMATION

Hierarchical low-rank approximations lie at the heart of many widely used numerical methods, including the Fast Multipole Method (FMM) [77, 183, 184, 185] and have applications ranging from partial differential equation (PDE) solvers and control problems to the non-uniform Fourier transform [27, 12, 176]. Hierarchical matrix recovery has received significant attention due to both practical and theoretical importance [111, 116, 117, 106, 32, 35, 105]. For example, black-box HODLR approximation methods can be used to obtain fast direct solvers in settings where we have access to efficient matrix-vector products, e.g., via an FMM or iterative method [111, 117]. Black-box methods are also central in the field of *operator or PDE learning*, where $A$ is an unknown differential operator and one can access matrix-vector products through physical experiments or simulations involving different forcing functions [33, 31, 151].

In this chapter[1], which, we discuss the theoretical bounds and stability guarantees for hierarchical matrix recovery and approximation from matrix-vector products. We propose a projection-based variant of peeling for hierarchical matrix recovery in Section 3.1. Then, in Section 3.2, we derive an algorithm and accompanying theory for hierarchical matrix approximation from matrix-vector products. In this work, we resolve the open question about the stability of peeling introduced as an important theoretical challenge in [35] by providing some of the first relative

[1]This chapter consists of sections which are excerpts from two papers [84, 45], The first section contains two hierarchical matrix recovery algorithms which I developed with Alex Townsend in [84]. The second section contains excerpts from [45], to which I contributed ideas for upper and lower bounds and algorithms. This paper was joint with Tyler Chen, Feyza Duman Keles, Cameron Musco, Chris Musco, and David Persson. Tyler Chen performed the numerical experiments in this section, and Tyler Chen and David Persson primarily derived the theory for noisy low-rank approximation.

guarantees for the stable behavior that is observed in practice.

## 3.1 Hierarchical matrix recovery from matrix-vector products

In Sections 3.1.1 and 3.1.4, we describe randomized algorithms that recover $N \times N$ rank-$k$ HODLR matrices from $\mathcal{O}((k+p)\log(N))$ queries and HSS matrices from a small multiple of $k$ queries with high probability. There are existing algorithms for HODLR matrix recovery using $\mathcal{O}(k\log(N))$ queries based on a recursive elimination strategy [111, 117]. Instead of using recursive elimination, our recovery algorithms use a recursive projection procedure that carefully projects the input query vectors, as well as outputs. Therefore, we think of our recovery algorithm for HODLR matrices as a QR-variant of recursive elimination [111, 117]. We suspect our algorithm to be more theoretically stable than peeling due to the advantages of projection over elimination, though both algorithms are observed to be stable in practice.

In Section 3.1.1, we describe a randomized algorithm for HSS matrix recovery, where we restrict to a particular representation of HSS matrices, and in Section 3.1.4, we derive a stable algorithm for HODLR recovery by incrementally making the matrix structure more complicated. Finally, in Section 3.1.6, we consider related problems, such as recovering matrices when the matrix-vector products are error-prone and recovering hierarchical matrices whose blocks are only numerically low-rank.

### 3.1.1 p-HSS matrix recovery

In this section, we derive a recovery algorithm for $\mathcal{H}_{N,k}$, the family of $N \times N$ rank-$k$ p-HSS matrices, where $N$ and $k$ are known in advance.

We first count the parameters that define the structure of $H_{N,k}$. There are

$$\#\text{ parameters} = \underbrace{4k\frac{N}{2}}_{U,V,W,Z} + \underbrace{k^2(N/2^{\ell-1}-4)}_{\text{off-diag. blks}} + \underbrace{2^{2\ell}\frac{N}{2^\ell}}_{\text{diag. blks}} = N\left(2k + \frac{k^2}{2^{\ell-1}} - \frac{4k^2}{N} + 2^\ell\right) \tag{3.1}$$

defining $H_{N,k}$. However, to apply the bound in Equation (2.1), we require that $\mathcal{H}_{N,k}$ is a linearly parametrized family, so that the equations from matrix-vector product queries are linear in the parameters of $H_{N,k}$. This is not the case. However, if we recover $U, V, W$, and $Z$ first, the equations will be linear in the remaining parameters. Therefore, by Equation (2.1), we need at least $\lceil \frac{k^2}{2^{\ell-1}} - \frac{4k^2}{N} + 2^\ell \rceil$ more matrix-vector products to fully recover $H_{N,k}$. Since $k \leq 2^\ell \leq 2k$ and in general, $N \gg k^2$, $4k$ queries is more than enough.

If $H_{N,k}$ is symmetric, there are $kN$ parameters defining $U$ and $V$, $k^2(N/2^\ell - 2)$ parameters in the off-diagonal blocks, and $2^{\ell-1}(2^\ell+1)N/2^\ell = 2^{-1}(2^\ell+1)N$ parameters in the diagonal blocks. Thus, we have

$$\#\text{ parameters of symmetric } H_{N,k} = \underbrace{kN}_{U,V} + \underbrace{k^2(N/2^\ell - 2)}_{\text{off-diag. blks}} + \underbrace{\frac{(2^\ell+1)N}{2}}_{\text{diag. blocks}}, \tag{3.2}$$

and symmetric rank-$k$ p-HSS recovery requires at least $\lceil \frac{k^2}{2^\ell} - \frac{2k^2}{N} + \frac{2^\ell+1}{2} \rceil$ queries, in addition to the number of queries needed to recover $U$ and $V$. Using $k \leq 2^\ell \leq 2k$ and $N \gg k^2$ as before, $3k$ queries is more than enough.

**Existing approaches**

There are a few existing algorithms for general HSS matrix recovery. Peeling algorithms utilize the same recursive elimination strategy as existing algorithms for HODLR recovery [117, 106], and require $\mathcal{O}((k+p)\log_2(N))$ matrix-vector products. These algorithms yield an upper bound on QC(HSS rank-$k$) and QC(HSS rank-$k$) of $\mathcal{O}(k\log_2(N))$. In theory, peeling algorithms can be numerically unstable because the pivoting strategy in elimination relies on the hierarchical structure of the matrix, rather than the magnitude of its entries.

A recent HSS recovery algorithm, which we refer to as the Levitt–Martinsson HBS algorithm, does not rely on peeling, and instead only requires $\mathcal{O}(k)$ matrix-vector products, as suggested by the lower bound Equation (2.1) [105]. Thus, QC(HSS rank-$k$) $= \mathcal{O}(k)$. It leverages a telescoping factorization of an HSS matrix and recursively recovers its entries in an order based on the hierarchical structure. Our algorithm also achieves $\mathcal{O}(k)$ matrix-vector products and exploits hierarchical structure for a projection-based method for recovery. Essentially, our algorithm solves a linear system in the parameters of a p-HSS matrix given in Equation (3.1), which is generated from $\mathcal{O}(k)$ matrix-vector queries. From the existence of the Levitt–Martinsson HBS algorithm [105] and a result by Otto [136], we know that using the same number of matrix-vector products as the Levitt–Martinsson HBS algorithm, our linear system has a unique solution with probability 1. In practice, we observe that the linear system has a unique solution with even fewer matrix-vector products than this. We believe this is due to some oversampling in the Levitt–Martinsson algorithm. While our algorithm and that of [105] require the same number of matrix-vector products and are both projection-based, we recover the entries of a p-HSS matrix all at once, exploiting the sparsity structure and

magnitude of entries in the linear system. In contrast, [105] recovers a general HSS matrix in sequential steps based on the telescoping factorization.

Our numerical results in Section 3.1.3 suggest that our algorithm performs better than that of [105] in terms of accuracy, especially when the rank of the p-HSS matrix is low. However, one trade-off here is that the Levitt–Martinsson HBS algorithm in [105] achieves linear complexity and is more efficient for large ranks. Our algorithm can be more computationally expensive due to our QR strategy for solving a large linear system in the parameters of an HSS matrix. In practice, we use a multifrontal multithreaded sparse QR factorization causes our algorithm's computational time to grow like $\mathcal{O}(N^{1.2})$ [58]. We also observe in Section 3.1.6 that our linear system strategy is a more accurate solution to related recovery problems: when the HSS matrix has only numerically low-rank blocks and when matrix-vector products are error-prone, i.e., outputs of matrix-vector products are noisy. Overall, our two algorithms are complementary, as that of [105] is better suited to the telescoping factorization structure for an HSS matrix. In contrast, ours is designed to recover an HSS structure in a format that stores the parameters defining each subblock of the HSS matrix. We also emphasize that the algorithm of [105] is suited to the general class of HSS matrices, whereas our algorithm the slightly more restrictive class of p-HSS matrices (Definition 1.4.7).

We now describe a randomized algorithm to recover $H_{N,k}$. We do so by progressively increasing the complexity of the p-HSS structure from symmetric $H_{4,1}$ (see Section 3.1.1) to general symmetric $H_{N,k}$ (see Section 3.1.1) before extending to p-HSS matrices and so-called restricted HSS matrices (see Section 3.1.2).

### Symmetric rank-1 p-HSS matrices

We begin by considering the recovery of symmetric p-HSS matrices, which is conceptually easier to explain. In turn, we consider symmetric $H_{4,1}$ (see Section 3.1.1), symmetric $H_{N,1}$ where $N$ is a power of 2 (see Section 3.1.1), and the recovery of symmetric $H_{N,k}$ (see Section 3.1.1). We emphasize that our recovery algorithm is not necessarily optimal for a matrix as small as $4 \times 4$; however, we consider this initial pet example to illustrate the algorithm.

### Recovering $\mathbf{H_{4,1}}$

Consider the recovery of a $4 \times 4$ symmetric rank-1 p-HSS matrix, which can be expressed as

$$H_{4,1} = \left[\begin{array}{cc|cc} a_1 & a_2 & \multicolumn{2}{c}{\multirow{2}{*}{$vu^\top$}} \\ a_2 & a_3 & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$uv^\top$}} & a_4 & a_5 \\ & & a_5 & a_6 \end{array}\right], \qquad a_1, \ldots, a_6 \in \mathbb{R}, \quad u, v \in \mathbb{R}^2. \tag{3.3}$$

Hence, $H_{4,1}$ is defined by 6 parameters in the entries $a_1, \ldots, a_6$ and 4 more parameters in the vectors $u$ and $v$. We can stably recover $uv^\top$ by noting that

$$H_{4,1} \begin{bmatrix} x_1 \\ x_2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} * \\ * \\ uv^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{bmatrix}, \qquad H_{4,1} \begin{bmatrix} 0 \\ 0 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} vu^\top \begin{bmatrix} y_3 \\ y_4 \end{bmatrix} \\ * \\ * \end{bmatrix}, \tag{3.4}$$

where $*$ denotes irrelevant entries. Thus, we compute matrix-vector products with $uv^\top$ and $vu^\top$ by querying $H_{4,1}$. We use the randomized SVD to stably recover $u$ and $v$ in a total of $2 + p$ queries. Finally, we use two matrix-vector products of the

form:

$$H_{4,1}\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} a_1 g_1 + a_2 g_2 \\ a_2 g_1 + a_3 g_2 \\ a_4 g_3 + a_5 g_4 \\ a_5 g_3 + a_6 g_4 \end{bmatrix} + \left[\begin{array}{cc|c} 0 & 0 & \multirow{2}{*}{$vu^\top$} \\ 0 & 0 & \\ \hline \multirow{2}{*}{$uv^\top$} & & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \end{array}\right] \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix}, \tag{3.5}$$

where $g_1, \ldots, g_4$ are standard Gaussian i.i.d. random numbers, to set up an $8 \times 6$ linear system for $a_1, \ldots, a_6$. The exact rectangular linear system is given by

$$\begin{bmatrix} g_1 & g_2 & 0 & 0 & 0 & 0 \\ 0 & g_1 & g_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_3 & g_4 & 0 \\ 0 & 0 & 0 & 0 & g_3 & g_4 \\ g_1' & g_2' & 0 & 0 & 0 & 0 \\ 0 & g_1' & g_2' & 0 & 0 & 0 \\ 0 & 0 & 0 & g_3' & g_4' & 0 \\ 0 & 0 & 0 & 0 & g_3' & g_4' \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} H_{4,1}\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} - \left[\begin{array}{cc|c} 0 & 0 & \multirow{2}{*}{$vu^\top$} \\ 0 & 0 & \\ \hline \multirow{2}{*}{$uv^\top$} & & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \end{array}\right] \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \\ H_{4,1}\begin{bmatrix} g_1' \\ g_2' \\ g_3' \\ g_4' \end{bmatrix} - \left[\begin{array}{cc|c} 0 & 0 & \multirow{2}{*}{$vu^\top$} \\ 0 & 0 & \\ \hline \multirow{2}{*}{$uv^\top$} & & \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \end{array}\right] \begin{bmatrix} g_1' \\ g_2' \\ g_3' \\ g_4' \end{bmatrix} \end{bmatrix}, \tag{3.6}$$

which can trivially be decoupled into two $4 \times 3$ rectangular linear systems problems to solve for $a_1, a_2, a_3$ and $a_4, a_5, a_6$. Here, $g_1', \ldots, g_4'$ are also standard iid Gaussians. This means that we can recover $H_{4,1}$ with a total of $4 + p$ matrix-vector product queries. Of course, $4 + p > 4$, so the naive algorithm of recovering $H_{4,1}$ column-by-column is better here; however, these ideas extend to recovering $H_{N,1}$ for larger $N$.

**Recovering an N × N rank-1 p-HSS matrix, where N is a power of 2**

A similar procedure for $H_{4,1}$ (see Section 3.1.1) works for the recovery of $H_{N,1}$, where $N$ is a power of 2. Note that for any $x, y \in \mathbb{R}^{N/2}$ we have

$$H_{N,1} \begin{bmatrix} x \\ \mathbf{0}_{N/2} \end{bmatrix} = \begin{bmatrix} * \\ uv^\top x \end{bmatrix}, \qquad H_{N,1} \begin{bmatrix} \mathbf{0}_{N/2} \\ y \end{bmatrix} = \begin{bmatrix} vu^\top y \\ * \end{bmatrix},$$

where $\mathbf{0}_{N/2}$ is the zero vector of length $N/2$ and $*$ is a vector of length $N/2$ with arbitrary entries. Thus, we can access matrix-vector products with $uv^\top$ and $vu^\top$ by directly querying $H_{N,1}$. This means we can apply the randomized SVD to stably recover $u$ and $v$ in a total of $2 + p$ queries.

By setting $k = 1$ in Equation (3.2) and counting the parameters other than $u$ and $v$, there are $2N - 2$ parameters remaining to recover in $H_{N,1}$. We construct a rectangular linear system whose solution is the remaining parameters in a similar way to Equation (3.6). Each matrix-vector query with a random i.i.d. standard Gaussian vector yields $N$ linear equations in the remaining parameters, so two matrix-vector products yield a $2N \times (2N - 2)$ rectangular linear system. The rectangular linear system has a recurring "staircase" structure due to the HSS structure (see Figure 3.1, left), and trivially decouples into two $N \times (N - 1)$ rectangular linear systems. We observe that the multifrontal multithreaded sparse QR factorization [58] takes less than 5 seconds to solve the rectangular linear system for $N = 2^{18} = 262{,}144$, and has a computational time complexity of about $\mathcal{O}(N^{1.2})$ (see Figure 3.1, right).

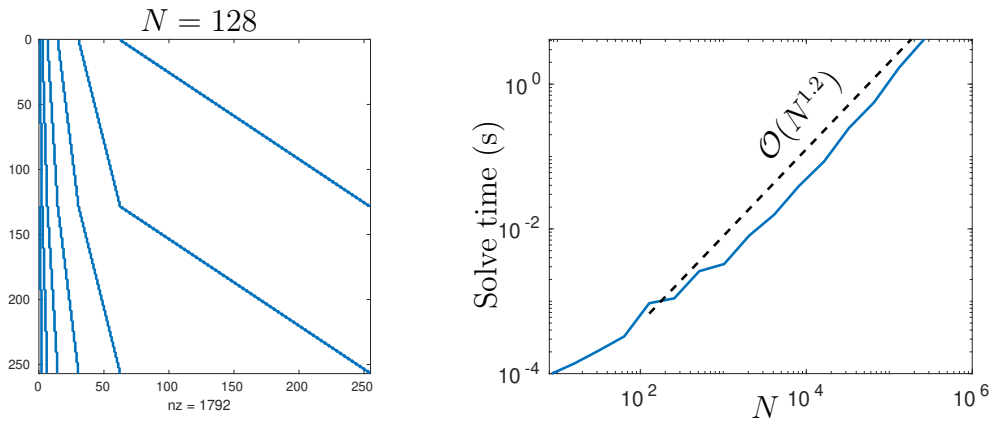



Figure 3.1: Left: The structure of the $2N \times (2N-2)$ sparse linear system that are satisfied by the parameters of a symmetric $H_{N,1}$, excluding $u_1$ and $v_1$ for $N = 128$. Right: The computational time required to solve the rectangular linear system using '\' in MATLAB for the remaining parameters of $H_{N,1}$, which executes the multifrontal multithreaded sparse QR factorization [58]. From these experiments, we observe that the computational time scales like $\mathcal{O}(N^{1.2})$.

**Recovering a symmetric, rank-k p-HSS matrix**

Our algorithm for recovering $H_{N,k}$ generalizes our approach for $H_{N,1}$ (see Section 3.1.1). In total, it requires $5k + p$ matrix-vector products to recover $H_{N,k}$, including the $N/2 \times k$ matrices $U$ and $V$ in the largest off-diagonal blocks and the $k \times k$ matrices in each subblock. First, we recover the matrix $UV^\top$ by noticing that for any vectors $x, y \in \mathbb{R}^{N/2}$,

$$H_{N,k} \begin{bmatrix} x \\ \mathbf{0}_{N/2} \end{bmatrix} = \begin{bmatrix} * \\ UV^\top x \end{bmatrix}, \qquad H_{N,k} \begin{bmatrix} \mathbf{0}_{N/2} \\ y \end{bmatrix} = \begin{bmatrix} VU^\top y \\ * \end{bmatrix}.$$

Thus, we query $UV^\top$ and $VU^\top$ by using matrix-vector products with $H_{N,k}$, allowing us to recover $UV^\top$ with the randomized SVD in $2k + p$ queries with high probability.

We construct a linear system for the remaining parameters that define $H_{N,k}$. Since we have already recovered $UV^\top$, a matrix-vector product of the form $H_{N,k}x$

yields the following $N$ linear equations:

$$\begin{bmatrix} H_{11}x(i_1) + V(i_1,:)H_{12}V(i_2,:)^\top x(i_2) \\ V(i_2,:)H_{21}^\top V(i_1)^\top x(i_1) + H_{22}x(i_2) \\ H_{33}x(i_3) + U(i_1,:)H_{34}U(i_2,:)^\top x(i_4) \\ U(i_2,:)H_{43}^\top U(i_1)^\top x(i_3) + H_{44}x(i_4) \end{bmatrix} = H_{N,k}x - \begin{bmatrix} UV^\top \begin{bmatrix} x(i_1) \\ x(i_2) \end{bmatrix} \\ VU^\top \begin{bmatrix} x(i_3) \\ x(i_4) \end{bmatrix} \end{bmatrix},$$

where $i_1 = 1 : N/4$, $i_2 = N/4{+}1 : N/2$, $i_3 = N/2{+}1 : 3N/4$, and $i_4 = 3N/4{+}1 : N$. Since $\mathrm{vec}(AXB^\top) = (B \otimes A)\mathrm{vec}(X)$, we write this as

$$[A_1\ A_2\ A_3\ A_4]\begin{bmatrix} \mathrm{vec}(H_{12}) \\ \mathrm{vec}(H_{21}) \\ \mathrm{vec}(H_{34}) \\ \mathrm{vec}(H_{43}) \end{bmatrix} + \begin{bmatrix} \mathrm{vec}(H_{11}x(i_1)) \\ \mathrm{vec}(H_{22}x(i_2)) \\ \mathrm{vec}(H_{33}x(i_3)) \\ \mathrm{vec}(H_{44}x(i_4)) \end{bmatrix} = \mathrm{vec}(H_{N,k}x) - \mathrm{vec}\left( \begin{bmatrix} UV^\top \begin{bmatrix} x(i_1) \\ x(i_2) \end{bmatrix} \\ VU^\top \begin{bmatrix} x(i_3) \\ x(i_4) \end{bmatrix} \end{bmatrix} \right),$$

where we write $A_1 = (x(i_2)^\top V(i_2,:)) \otimes V(i_1,:)$, $A_2 = (x(i_1)^\top V(i_1,:)) \otimes V(i_2,:)$, $A_3 = (x(i_4)^\top U(i_2,:)) \otimes U(i_1,:)$, and $A_4 = (x(i_3)^\top U(i_1,:)) \otimes U(i_2,:)$. Since $H_{jj}$ for $1 \le j \le 4$ are themselves HSS matrices, $N$ equations can be constructed recursively. For example, when $N = 16$ and $k = 2$, $H_{N,k}$ takes the following form:

$$H_{16,2} = \left[\begin{array}{cc|cc} \begin{bmatrix} b_9 & b_{10} & b_{11} & b_{12} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{11} & b_{12} & b_{13} & b_{14} \\ b_{12} & b_{13} & b_{14} & b_{15} \end{bmatrix} & V(1:4,:)\begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} V(5:8,:)^\top & \multicolumn{2}{c}{\multirow{2}{*}{$VU^\top$}} \\ V(5:8,:)\begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix}^\top V(1:4,:)^\top & \begin{bmatrix} b_{16} & b_{17} & b_{18} & b_{19} \\ b_{17} & b_{18} & b_{19} & b_{20} \\ b_{18} & b_{19} & b_{20} & b_{21} \\ b_{19} & b_{20} & b_{21} & b_{22} \end{bmatrix} & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$UV^\top$}} & \begin{bmatrix} b_{23} & b_{24} & b_{25} & b_{26} \\ b_{24} & b_{25} & b_{26} & b_{27} \\ b_{25} & b_{26} & b_{27} & b_{28} \\ b_{26} & b_{27} & b_{28} & b_{29} \end{bmatrix} & U(1:4,:)\begin{bmatrix} b_5 & b_6 \\ b_7 & b_8 \end{bmatrix} U(5:8,:)^\top \\ & & U(5:8,:)\begin{bmatrix} b_5 & b_6 \\ b_7 & b_8 \end{bmatrix}^\top U(1:4,:)^\top & \begin{bmatrix} b_{30} & b_{31} & b_{32} & b_{33} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{32} & b_{33} & b_{34} & b_{35} \\ b_{33} & b_{34} & b_{35} & b_{36} \end{bmatrix} \end{array}\right],$$

where $U, V \in \mathbb{R}^{8\times 2}$. A matrix-vector product $H_{16,2}x$ for $x \in \mathbb{R}^{16}$ gives us 16 equations in the unknowns $b_1, \ldots, b_{36}$. Given the repetitive structure of $H_{16,2}$, we

focus on the first four equations produced by this query to find the pattern of the linear system. The top left $4 \times 4$ subblock of the linear system matrix, called $L_1$, corresponds to the coefficients of $b_1, \ldots, b_4$ in the first four equations of the linear system:

$$L_1 = \begin{bmatrix} \sum_{j=5}^{8} x_j v_{1,j} v_{1,1} & \sum_{j=5}^{8} x_j v_{2,j} v_{1,1} & \sum_{j=5}^{8} x_j v_{1,j} v_{2,1} & \sum_{j=5}^{8} x_j v_{2,j} v_{2,1} \\ \sum_{j=5}^{8} x_j v_{1,j} v_{1,2} & \sum_{j=5}^{8} x_j v_{2,j} v_{1,2} & \sum_{j=5}^{8} x_j v_{1,j} v_{2,2} & \sum_{j=5}^{8} x_j v_{2,j} v_{2,2} \\ \sum_{j=5}^{8} x_j v_{1,j} v_{1,3} & \sum_{j=5}^{8} x_j v_{2,j} v_{1,3} & \sum_{j=5}^{8} x_j v_{1,j} v_{2,3} & \sum_{j=5}^{8} x_j v_{2,j} v_{2,3} \\ \sum_{j=5}^{8} x_j v_{1,j} v_{1,4} & \sum_{j=5}^{8} x_j v_{2,j} v_{1,4} & \sum_{j=5}^{8} x_j v_{1,j} v_{2,4} & \sum_{j=5}^{8} x_j v_{2,j} v_{2,4} \end{bmatrix}.$$

Note that this matrix is the same as the one generated by the Kronecker product given by $V(1:4,:) \otimes (x(5:8)^\top V(5:8,:))$. For the rest of the equations corresponding to multiplication by a $4 \times 4$ off-diagonal subblock, there is an analogous Kronecker product, as this is the same product up to changing the indexing of $U, V$, and $x$.

To find the remaining coefficients in the first 4 equations, we note that the only other nonzero coefficients of the parameters $b_i$ result from multiplication by the diagonal block with seven parameters: $b_9, \ldots, b_{15}$. The corresponding subblock of the linear system matrix, which is $4 \times 7$ and starts from the parameter $b_9$, looks like this:

$$L_2 = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & 0 & 0 & 0 \\ 0 & x_1 & x_2 & x_3 & x_4 & 0 & 0 \\ 0 & 0 & x_1 & x_2 & x_3 & x_4 & 0 \\ 0 & 0 & 0 & x_1 & x_2 & x_3 & x_4 \end{bmatrix}.$$

Putting these nonzero blocks together and setting the rest of the coefficients to 0 gives the first 4 rows of the linear system: $\begin{bmatrix} L_1 & L_2 & \mathbf{0_{4,25}} \end{bmatrix}$. As with $L_1$, the

structure of $L_2$ is the same for the rest of the linear system, up to changing the indexing of $x$.

### The linear system for $H_{N,k}$

The p-HSS recovery strategy of forming a linear system and solving for parameters works for any p-HSS matrix of arbitrary size and partitioning of the hierarchy. However, to illustrate our algorithm, we continue to assume that $N$ is a power of 2. We generate the rank-$k$ linear system according to the analogous formula using Kronecker products. Now that we have a general form for the linear system given by one matrix-vector product, we find the number of queries needed to generate a linear system with more equations than the parameters of $H_{N,k}$. We claim that this number is at least $3k$. This is the same analysis as in Equation (3.2); we have already recovered $U$ and $V$, and we also assume the diagonal blocks each have $2^{2\ell}$ degrees of freedom because we know by Lemma 2.1.4 that symmetry does not reduce the number of queries. So, there are $k^2(N/2^\ell - 2) + 2^\ell \cdot N$ parameters. Then by Equation (2.1), the number of matrix-vector queries to guarantee more equations than unknowns is $\lceil \frac{k^2}{2^\ell} - \frac{2k^2}{N} + 2^\ell \rceil \leq 3k$.

Thus, we perform $3k$ matrix-vector products and generate $3Nk$ equations. We solve this linear system with the multifrontal multithreaded sparse QR factorization. From the degrees of freedom perspective, $3k$ is a lower bound on the number of queries that will give a unique solution. We also observe that $3k$ queries are enough to uniquely determine $H_{N,k}$. Moreover, this solution has little error because the right-hand side of the least-squares problem is in the column space of the left-hand side. Thus, only the condition number of the linear system upper bounds the solution's relative error rather than its square. We observe that the

relative error of the solution grows slowly with $N$ in Section 3.1.3, demonstrating the stability of the p-HSS recovery algorithm in practice.

#### General rank-$k$ p-HSS recovery

If $H_{N,k}$ is not symmetric, the same linear system strategy works. Recovery of $U, V, W$, and $Z$ requires $2(k+p)$ queries with $A$ and $2k$ queries with $A^\top$ via the randomized SVD. Then, we solve for the remaining parameters by constructing a linear system from matrix-vector products. By Equation (3.1), there are now $k^2(N/2^{\ell-1} - 4) + 2^\ell N$ parameters defining $H_{N,k}$, so the number of queries to guarantee more equations than unknowns is $\lceil \frac{k^2}{2^{\ell-1}} - \frac{4k^2}{N} + 2^\ell \rceil \leq 4k$ using the bound $k \leq 2^\ell \leq 2k$ and the fact that $N \gg k^2$. Thus, the total number of matrix-vector products with $A$ is $6k + 2p$ and with $A^\top$ is $2k$.

### 3.1.2 Restricted symmetric, rank-$k$ HSS recovery

We often deal with a more specific symmetric rank-$k$ HSS matrix, which we call a restricted HSS matrix $A^{\text{res}}_{N,k}$. This also arises as one step of our HODLR recovery algorithm. In this case, we already know the $N/2 \times k$ matrices $U$ and $V$, the row and column spaces defining $A^{\text{res}}_{N,k}$. In addition, $A^{\text{res}}_{N,k}$'s diagonal blocks $A^{\text{res}}_{ii}$ have a more specific structure, rather than being general $2^\ell \times 2^\ell$ blocks. The top half diagonal blocks are $A^{\text{res}}_{ii} = V^{(i)} A^{\text{res}}_i V^{(i)\top}$, and the bottom half diagonal blocks are $A^{\text{res}}_{jj} = U^{(j)} A^{\text{res}}_j U^{(j)\top}$ where $A^{\text{res}}_i$, $A^{\text{res}}_j$ are symmetric $k \times k$ matrices and $V^{(i)}$ and $U^{(j)}$ correspond to the $i$th and $j$th restrictions of $U$ and $V$, respectively. To recover a restricted HSS matrix, we construct a linear system similar to that in Section 3.1.1; however, it requires only $2k$, rather than $3k$, matrix-vector products to be solved.

To see this, we count the degrees of freedom in $A^{\text{res}}_{N,k}$ and divide by the number of linearly independent equations given by a matrix-vector product. The analysis is almost identical to that in Section 3.1.1, and the only difference is that the diagonal blocks $A^{\text{res}}_{ii}$ contribute $k^2$ instead of $2^{2m}$ degrees of freedom. We have $k^2(N/2^\ell - 2)$ parameters in the off-diagonal blocks and $Nk^2/2^\ell$ parameters in the diagonal blocks. Thus, the number of queries that guarantee more equations than unknowns is

$$\left\lceil \frac{k^2}{2^\ell} - \frac{2k^2}{N} + \frac{k^2}{2^\ell} \right\rceil \leq 2k,$$

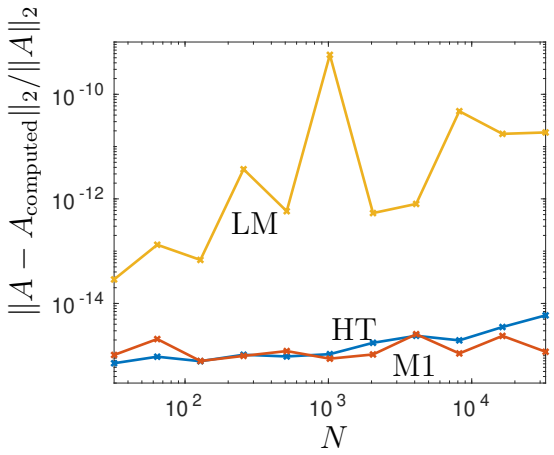
where the last inequality follows by using the bound $k \leq 2^\ell$. Thus, we use $2k$ matrix-vector products to recover a symmetric restricted HSS matrix.

**Asymptotic Complexity**

We find the asymptotic behavior of $T_{\text{pHSS}}$, the time required to recover a rank-$k$ p-HSS matrix $A$ using the algorithm in Section 3.1.1. Let $T_A$ denote the time to apply $A$ or $A^\top$ to a vector, and $T_{\text{flop}}$ denote the time for a floating point operation. To recover $A$, one applies $A$ to $6k+2p$ vectors and $A^\top$ to $2k$ vectors, totaling a cost of $(8k + 2p)T_A$. The remaining cost is incurred by solving a $4Nk$ by $Nk^2/2^{\ell-1} - 4k^2 + 2^\ell N$ linear system for the parameters of $A$. Using the QR factorization to solve the associated least-squares problem costs $\mathcal{O}(T_{\text{flop}}N^3k^3)$. Then overall, $T_{\text{pHSS}} = \mathcal{O}(T_A(k+p) + T_{\text{flop}}N^3k^3)$.

### 3.1.3 Numerical results

In this section, we compare the performance of our p-HSS recovery algorithm to those of Martinsson in [117] and Levitt and Martinsson in [105] using the same




| $N$ | M1 | LM | HT |
|---|---|---|---|
| 32 | 90 | 33 | 15 |
| 64 | 112 | 33 | 15 |
| 128 | 134 | 33 | 15 |
| 256 | 156 | 33 | 15 |
| 512 | 178 | 33 | 15 |
| 1024 | 200 | 33 | 15 |
| 2048 | 222 | 33 | 15 |
| 4096 | 244 | 33 | 15 |

Figure 3.2: Left: The relative error in recovering an $N \times N$ symmetric rank-1 generic HSS matrix. M1 refers to the recursive elimination strategy in [117], a peeling algorithm of Martinsson. Our algorithm, denoted as HT, is described in Section 3.1.1. LM denotes the Levitt–Martinsson HBS algorithm in [105]. The errors in the spectral norm were calculated via 20 iterations of the power method. Right: Number of matrix-vector products used by each algorithm. While LM has a much larger observed recovery error than M1, it only uses $\mathcal{O}(k)$ matrix-vector products. Our algorithm also only uses $\mathcal{O}(k)$ queries and achieves the same accuracy as M1. In contrast, M1 needs $\mathcal{O}((k+p)\log_2(N))$ queries.

error measurement as in those papers. We measure relative error using $||H_{N,k} - H_{N,k}^{\text{recov}}||/||H_{N,k}||$ in the spectral norm via 20 iterations of the power method. To approximate the norm of a matrix $X$, we apply the power method to $X^\top X$. All of the experiments in this section and later on in Section 3.1.5 were written in MATLAB and carried out on a Xeon E5-2698 processor with a single core and 256 GB of memory.

We measure the success of the three algorithms in several ways. First, in terms of number of matrix-vector queries, [117] requires $\mathcal{O}((k+p)\log_2(N))$ queries, whereas both [105] and our p-HSS recovery algorithm only require $\mathcal{O}(k)$. On the other hand, our algorithm requires the most floating point operations due to the multifrontal multithreaded QR factorization. In contrast, the Levitt–Martinsson algorithm and Martinsson peeling algorithm require $\mathcal{O}(Nk^2)$ and $\mathcal{O}(N(\log_2(N))^2k^2)$ floating point operations, respectively. However, due to the

sparsity structure of the linear system, in practice, our algorithm's computational time complexity is $\mathcal{O}(N^{1.2})$.

A recovery algorithm should also be accurate. As shown in Figure 3.2, all three algorithms are reasonably accurate, and the performance of the algorithms in [117] and [105] are close to the numerical results in their papers. We observe that our algorithm and the peeling algorithm of [117] are comparable and more accurate than the HBS algorithm of [105]. Moreover, while our algorithm and the peeling algorithm of [117] achieve high accuracy, our method requires far fewer matrix-vector products.

Finally, our algorithm and the recent algorithm of [105] are both projection-based, as opposed to the peeling algorithm of [117], which employs a strategy of recursive elimination. Projection is advantageous compared to peeling because projection limits floating point error, whereas peeling can, in principle, be more numerically unstable. However, despite its use of projection, the algorithm in [105] is observed to be the least accurate. One explanation for this is the algorithm's recursive strategy of recovering the telescoping factorization of an HSS matrix, starting from the level closest to the diagonal. This process follows the sequential order of the telescoping factorization and can potentially propagate floating point errors. In fact, the recovered matrix may not have an exact HSS structure when the recovery algorithm is performed in floating point arithmetic. In contrast, the linear system strategy used in our HSS recovery algorithm solves for all of the parameters of an HSS matrix at once. It enforces the HSS structure even in floating point arithmetic.

It is worth noting that the Levitt–Martinsson algorithm [105] recovers an HSS matrix only in the form of its telescoping factors, which is an equivalent char-

acterization of an HSS matrix. In contrast, our algorithm recovers precisely the parameters that define each off-diagonal block in a p-HSS matrix. In this sense, our two algorithms are complementary, as they are suited to two different characterizations of HSS matrices, although the Levitt–Martinsson algorithm works on a slightly larger class. One of the algorithms may be preferable depending on how one stores an HSS matrix.

### 3.1.4 HODLR recovery

We now describe our algorithm for HODLR recovery. We denote an $N \times N$, symmetric, generic rank-$k$ HODLR matrix as $B_{N,k}$, where again $N = 2^n$. By generic, we mean that $B_{N,k}$ is defined by random parameters. That is, the low-rank factors of the off-diagonal blocks are matrices with i.i.d. random Gaussian entries, and the diagonal blocks also have i.i.d. random Gaussian entries. We then discuss how to recover a symmetric rank-1 HODLR matrix defined by parameters that are not truly random, i.e., the column and row spaces of different subblocks have some correlation. This technique has multiple stages, and we pare our matrix down by recovering parameters until all that is left is a restricted HSS matrix (see Section 3.1.2). We first recover the subblocks that do not correlate with one another by projecting inputs as in Section 3.1.4, then recover additional information by projecting outputs, and finally recover the rest using the symmetric rank-$k$ HSS algorithm described in Section 3.1.1. Finally, we extend our algorithm to nonsymmetric rank-$k$ complex-valued HODLR matrices of general size (see Section 3.1.4).

Both our algorithm and existing peeling algorithms utilize the randomized SVD algorithm for recovering low-rank blocks. It is worth noting that one may be able to save a factor of 2 here by using the Nyström method. However, in this paper,

we care more about the complexity $\mathcal{O}((k+p)\log_2(N))$ than constants.

### Symmetric and generic rank-1 HODLR recovery

We can visualize $B_{N,1}$'s structure (see Figure 1.2) where we force the matrix to be symmetric by setting $Z_i = U_i$ and $W_i = V_i$ for all $i$. Additionally, because $B_{N,1}$ is rank-1 HODLR, we label each of the matrices $U_i$ and $V_i$ as $u_i$ and $v_i$ to emphasize that they are vectors. The off-diagonal blocks are rank-1 and are defined by random parameters. That is, each rank-1 subblock is defined by generating random vectors $u_i$ and $v_i$ of the proper size.

Our algorithm recovers $B_{N,1}$ in levels, starting with the largest off-diagonal blocks, then recursing on smaller subblocks. At level 1, we recover the two off-diagonal blocks of size $N/2$ by finding $u_0$ and $v_0$. We do so with the exact same technique as described in Section 3.1.1, using $2+p$ matrix-vector products and applying the randomized SVD.

Then, we concatenate $u_0$ and $v_0$ in a "blacklist" vector $b_1$ of length $N$, i.e.,

$$b_1 = \begin{bmatrix} | \\ v_0 \\ | \\ | \\ u_0 \\ | \end{bmatrix}.$$

Moving on to level 2, we now recover $u_1, v_1, u_2$, and $v_2$. We perform matrix-vector products with analogously constructed alternating input vectors: $[x_1, 0, x_2, 0]^\top$ and $[0, y_1, 0, y_2]^\top$. Importantly, we modify the inputs $x_1$ and $x_2$ by replacing each with

its projection onto the orthogonal space of the corresponding block of $b_1$, which contains $u_0$ and $v_0$. We can call these new, orthogonalized vectors $x_1'$ and $x_2'$. Thus, we set:

$$x_1' \perp b_1(1 : N/4) \iff x_1' \perp v_0(1 : N/4),$$
$$x_2' \perp b_1(N/2 + 1 : 3N/4) \iff x_2' \perp u_0(1 : N/4).$$

This step is necessary because when we perform the product with $B_{N,1}$, $x_1$ and $x_2$ are orthogonal to the corresponding column spaces of the subblocks that were previously recovered. That is,

$$B_{N,1} \begin{bmatrix} x_1' \\ 0 \\ x_2' \\ 0 \end{bmatrix} = \begin{bmatrix} * \\ u_1 v_1^\top x_1' + (v_0 u_0^\top)(1 : N/4, N/4 + 1 : N/2) x_2' \\ * \\ (u_0 v_0^\top)(N/4 + 1 : N/2, 1 : N/4) x_1' + u_2 v_2^\top x_2' \end{bmatrix} = \begin{bmatrix} * \\ u_1 v_1^\top x_1' \\ * \\ u_2 v_2^\top x_2' \end{bmatrix}.$$

Thus, projection "zeroes" out the matrix in the desired subblocks, so that we can isolate the actions of $u_1 v_1^\top$ and $u_2 v_2^\top$ on random vectors. We can do the same trick with $[0, y_1, 0, y_2]^\top$ to isolate the actions of $v_1 u_1^\top$ and $v_2 u_2^\top$ on inputs of our choice.

Because $B_{N,1}$ is a generic HODLR matrix, if $x_1$ and $x_2$ are random, then the projected random vectors $x_1'$ and $x_2'$ are nonzero with probability 1. Then, one can still use the randomized SVD algorithm to recover the blocks at this level. This will take $2 + p$ matrix-vector products with $B_{N,1}$.

At level $\ell$, we construct two types of input vectors: one with alternating blocks of vectors $x_i$ and zeros, and another which alternates zeros and vectors $y_i$, where the blocks are of size $2^{n-\ell}$. Then, we project the blocks $x_i$ and $y_i$ to be orthogonal to the corresponding blocks of the blacklist vectors $b_1, \ldots, b_{\ell-1}$. When $B_{N,1}$ is applied to these alternating vectors, the projected inputs "zero out" $B_{N,1}$'s recov-

ered subblocks, and isolate the actions of the level-$\ell$ subblocks. We then use the randomized SVD algorithm to recover these blocks with high probability.

One question arises: At what level does this algorithm stop working? Once the length of the inputs $x_i$ and $y_i$ is less than or equal to the number of vectors it needs to be orthogonal to, i.e. the size of the blacklist, the projection step forces them to be zero vectors. Even slightly before this level, when the inputs $x_i$ and $y_i$ belong to a subspace of dimension less than $1 + p$, the randomized SVD algorithm cannot oversample the inputs, and we observe some loss of accuracy in the randomized SVD. In the generic HODLR case, the input subspace is of dimension $2^{n-\ell} - \#(\text{blacklist})$. A new vector is added to the blacklist at each level, so that at level $\ell$, there are $\ell - 1$ vectors in the blacklist. Thus, when $\ell$ becomes large enough so that $2^{n-\ell} - \ell + 1 < 1 + p$, we must use a different strategy to recover the remaining blocks. We can calculate the value of $\ell$ at which this occurs:

$$w := \frac{W(2^{n+p}\log(2))}{\log(2)} - p,$$

where $W$ is the Lambert W-function. Because the levels range from $1, \ldots, \log_2(N)$, we have the upper bound of

$$w \leq \log_2(N). \tag{3.7}$$

Then, using the lower bound on the Lambert W-function $W(x) \geq \log(x) - \log(\log(x))$ for $x \geq e$, we can also bound $w$ from below as

$$\begin{aligned} w &\geq \frac{\log(2^{n+p}\log(2)) - \log(\log(2^{n+p}\log 2))}{\log(2)} - p \\ &\geq n - 1 - \log_2(\log(2^{n+p}\log(2))) + \frac{\log(\log(2))}{\log(2)} \\ &\geq \log_2(N) - \log_2(\log_2(N) + p) - 1 \\ &= n - \log_2(n + p) - 1 \end{aligned} \tag{3.8}$$

**Algorithm 2** Generic Symmetric Rank-1 HODLR Recovery

**Input** Function handle `matvec`: $x \mapsto Ax$ and $N = 2^n$, the size of $A$

**Output** Vectors $U_1, V_1, \ldots, U_w, V_w$ that store $u$ and $v$ factors for every level's blocks

**Set** $w = \frac{W(2^{n+p}\log(2))}{\log(2)} - p$ and $r = 1 + p$, where $p$ is a fixed parameter, e.g., 10.

**for** level $\ell = 1, \ldots, w-1$ **do**

**Set** bsize $= N/2^\ell$, $U_\ell = [\,]$, $V_\ell = [\,]$, and $X = \left[\texttt{randn}(\text{bsize}, r); \texttt{zeros}(\text{bsize}, r); \cdots; \texttt{randn}(\text{bsize}, r); \texttt{zeros}(\text{bsize}, r)\right]$

**if** $\ell > 1$ **then**

*Project blocks of $X$ onto the corresponding orthogonal space of the blacklist. $I$ is the corresponding index set and $BL$ is a matrix storing stacked blacklist vectors.*

**for** each nonzero block of the form $X(I,:)$ **do**

**Set** $Q_1 = \texttt{orth}(\text{BL}(I,:))$, $X(I,:) \leftarrow X(I,:) - Q_1 Q_1^\top X(I,:)$

**end for**

**end if**

*Query $A$ with projected $X$. Apply rSVD to outputs to obtain each $u$ at level $\ell$.*

**Set** $W = \texttt{matvec}(X)$, $Y = [\,]$

**for** each nonzero block of the form $X(I,:)$ **do**

**Set** $J = I + \text{bsize}$, $u = \texttt{orth}(W(J,:), 1)$, $U_\ell \leftarrow [U; \texttt{zeros}(\text{bsize}, 1); u]$

**Set** $Q_2 = \texttt{orth}(\text{BL}(J,:))$, $y \leftarrow u - Q_2 Q_2^\top u$, $Y \leftarrow [Y; \texttt{zeros}(\text{bsize}, 1); y]$

**end for**

*Query $A$ with projected $Y$. Apply rSVD to outputs to obtain each $v$ at level $\ell$.*

**Set** $V_\ell = \texttt{matvec}(Y)$

**for** each nonzero block $V_\ell(I)$ **do**

**Set** $V_\ell(I) \leftarrow V_\ell(I)/(Y(I + \text{bsize})^\top U(I + \text{bsize}))$

**end for**

**Append** $[U_\ell(1:\text{bsize}); V_\ell(1:\text{bsize}); \ldots; U_\ell(2^\ell \text{bsize} + 1 : N); V_\ell(2^\ell \text{bsize} + 1 : N)]$ as a column to the matrix BL.

**end for**

Let $\tilde{A}$ be the matrix recovered so far using $U_1, V_1, \ldots U_{w-1}, V_{w-1}$.

**Set** bsize $= 2^{n-w+1}$, $X = \texttt{randn}(N, \text{bsize} + p)$, $W = \texttt{matvec}(X) - \tilde{A}X$

**for** each block of the form $W(I)$ of size bsize in $W$ **do**

$U_w(I) = \texttt{orth}(W(I))$

**end for**

$V_w = \texttt{matvec}(U_w) - \tilde{A}U_w$,

It remains to recover the diagonal blocks of size $2^{n-w+1} \times 2^{n-w+1}$, of which there are $N/2^{n-w+1} = 2^{w-1}$. We call the matrix we have recovered so far $B'_{N,1}$, which has zeros in these diagonal subblocks that we have not yet recovered. Now, it suffices to recover the matrix $B_{N,1} - B'_{N,1}$, whose only nonzero subblocks are the unknown diagonal subblocks. One also effectively has access to a matrix-vector product with $B_{N,1} - B'_{N,1}$ and a given vector $x$ by taking $B_{N,1}x - B'_{N,1}x = (B_{N,1} - B'_{N,1})x$.

We recover the diagonal blocks $B_{N,1_{ii}}$ with $2^{n-w+1} + p$ matrix-vector products. We construct an $N \times (2^{n-w+1} + p)$ input matrix $X$ with random Gaussian entries. Then the product $(B_{N,1} - B'_{N,1})X$ isolates the actions of each diagonal block $B_{N,1_{i,i}}$ on a $2^{n-w+1} \times (2^{n-w+1} + p)$ random Gaussian input matrix $X_i$, for $1 \leq i \leq 2^{w-1}$. Then we can apply the randomized SVD to recover diagonal blocks. Here, we perform enough matrix-vector products to recover each diagonal block column-by-column; however, we prefer to use random Gaussian inputs, so instead we treat each diagonal block as a rank-$2^{n-w+1}$ matrix and apply the randomized SVD.

$$\begin{bmatrix} B_{N,1_{11}} & 0 & & 0 \\ 0 & B_{N,1_{22}} & & \\ & & \ddots & 0 \\ 0 & & 0 & B_{N,1_{2^{w-1},2^{w-1}}} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_{2^{w-1}} \end{bmatrix} = \begin{bmatrix} B_{N,1_{11}} X_1 \\ B_{N,1_{22}} X_2 \\ \vdots \\ B_{N,1_{2^{w-1},2^{w-1}}} X_{2^{w-1}} \end{bmatrix}.$$

Overall, we recover $B_{N,1}$ with $2 + p$ matrix-vector products at each level until level $w$, then $2^{n-w+1} + p$ products for the diagonal blocks. The bounds Equation (3.8) and Equation (3.7) yield an upper bound on the total number of required matrix-vector queries:

$$\#\text{ matrix-vector products} = (2+p)(w-1) + 2^{n-w+1} < (6+p)\log_2(N) + 5p.$$

### General symmetric rank-1 HODLR recovery

We now turn to the case where the parameters defining a rank-1 HODLR matrix are not random. We denote this matrix by $C_{N,1}$. We recover $u_0$ and $v_0$ with the same $2 + p$ matrix-vector products as in Section 3.1.4. Again, we store $u_0$ and $v_0$ in blacklist vector. At subsequent levels, we orthogonalize the inputs to the corresponding parts of $u_0$ and $v_0$ and recover subblocks by the same technique as in Section 3.1.4, projecting inputs onto the orthogonal space of the blacklist.

In Section 3.1.4, we recovered the vectors $u$ and $v$ defining each rank-1 subblock by projecting inputs, modifying the input to $x'$. Because $u$ and $v$ were random and $x'$ was orthogonal to blacklist vectors with no correlation to $v$, the output $uv^\top x'$ was nonzero with probability 1. Thus, we recovered $u$ and $v$ using the randomized SVD.

However, in the general HODLR case, some blocks may "fail", i.e., their outputs equal 0. Of course, in practice, the outputs will not be exactly zero, so one can instead consider failure to be when the outputs are below some tolerance. For simplicity, we describe the exact arithmetic case where the outputs are exactly 0. This prevents us from invoking the randomized SVD. If we want to observe the action of $u$ on a random input $x'$, and $x'$ is orthogonal to $k$ vectors of the blacklist which we call $b_{u1}, \ldots, b_{uk}$, it is possible that $v$ is a linear combination of $b_{u1}, \ldots, b_{uk}$. Then, $uv^\top x' = 0$. This can also happen if $uv^\top$ is a block of zeros. So, projecting inputs may not recover all of $C_{N,1}$. However, even if a block fails, we can still recover finer blocks that do not fail, as the projected inputs zero out prior failed subblocks. Thus, we recover what we can of $C_{N,1}$ with this technique, which again takes $(2+p)(w-1)$ queries. Next, we return to those failed blocks and fully recover them with another pass, so that we recover all of $C_{N,1}$.

We now project outputs instead of inputs, using a second pass of $2+p$ matrix-vector products at each level with the failed blocks. More precisely, for a failed subblock $uv^\top$ at level $\ell$, we perform matrix-vector products with the same alternating inputs as in Section 3.1.4. Let $uv^\top$ act on the input $x_u$. This time, we do not project $x_u$ onto the orthogonal space of the blacklist vectors, $b_{u,1}, \ldots, b_{u,k}$. Instead, the product yields:

$$uv^\top x_u + \sum_{i=1}^{k} b_{u,i}(*)b_{v,i}(*)^\top x_i, \tag{3.9}$$

where $*$ represents proper indexing of the $k$ blacklist vectors. We define $\text{proj}_b$ as projecting subsets of a vector onto the corresponding subsets of vectors in blacklist $b$. Projecting the output onto $b^\perp$, we zero out the second term in Equation (3.9) and obtain $uv^\top x_u - \text{ proj }_b uv^\top x_u = (uv^\top - \text{ proj }_b uv^\top)x_u$, a rank-1 matrix applied to $x_u$. Thus, we can recover the matrix $uv^\top - \text{proj}_b uv^\top$ with high probability using the randomized SVD, revealing the failed blocks up to a linear combination of the blacklist vectors. We repeat this for the failed blocks at each level, taking at most $(2+p)(w-1)$ matrix-vector products. This step at the final level $w$ also gives us the data of the symmetric diagonal blocks that are not a linear combination of the blacklist vectors.

As in Section 3.1.4, to recover the rest of $C_{N,1}$, we recover $C_{N,1} - C'_{N,1}$, where $C'_{N,1}$ is the matrix we have recovered so far. The subblocks in $C_{N,1} - C'_{N,1}$ are projections of $C_{N,1}$'s blocks onto the blacklist at that subblock's level. If there are $k$ vectors in the blacklist, $C_{N,1} - C'_{N,1}$ is a restricted rank-$k$ HSS matrix. By Section 3.1.1, we recover $C_{N,1} - C'_{N,1}$ in at most $2k$ matrix-vector products. The size of the blacklist is at most $w - 1 \leq \log_2(N) - 1$, where equality holds if a vector is appended to the blacklist at every level until the diagonal blocks. In total, we have a loose upper bound on the number of matrix-vector products used to recover

$C_{N,1}$:

$$\#\text{ matrix-vector products} \leq 2(2+p)(w-1) + 2(\log_2(N)-1) \leq (6+2p)\log_2(N).$$

To illustrate this algorithm and the possible ways blocks can pass or fail, we recover $C_{8,1}$ as an example below.

**Example 3.1.1.** *The structure of $C_{8,1}$ is below. We first recover $u_0$ and $v_0$ and store them in the blacklist $b$. At level 2, suppose the blocks with an "$\times$" failed and the block with a "$\checkmark$" passed when we applied to the projected inputs.*

$$C_{8,1} = \left[\begin{array}{cc|cc} C_{11} & v_1u_1^\top \checkmark & \multicolumn{2}{c}{\multirow{2}{*}{$v_0u_0^\top$}} \\ u_1v_1^\top \times & C_{22} & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$u_0v_0^\top$}} & C_{33} & v_2u_2^\top \times \\ & & u_2v_2^\top \times & C_{44} \end{array}\right].$$

*The failed blocks tell us that for some of the projected inputs, which we will write as $x_1'$, $x_2'$, and $y_2'$, we have that $u_1v_1^\top x_1' = v_2u_2^\top x_2' = u_2v_2^\top y_2' = 0$, so we cannot recover $u_1v_1^\top$ or $u_2v_2^\top$ via the randomized SVD. Then, we perform matrix-vector products without projecting inputs:*

$$C_{8,1}\begin{bmatrix} x_1 \\ 0 \\ x_2 \\ 0 \end{bmatrix} = \begin{bmatrix} C_{11}x_1 + v_0(1:4)u_0(1:4)^\top x_2 \\ u_1v_1^\top x_1 + v_0(5:8)u_0(1:4)^\top x_2 \\ u_0(1:4)v_0(1:4)^\top x_1 + C_{33}x_2 \\ u_0(5:8)v_0(1:4)^\top x_1 + u_2v_2^\top x_2 \end{bmatrix}, \quad C_{8,1}\begin{bmatrix} 0 \\ y_1 \\ 0 \\ y_2 \end{bmatrix} = \begin{bmatrix} v_1u_1^\top y_1 + v_0(1:4)u_0(5:8)^\top y_2 \\ C_{22}y_1 + v_0(5:8)u_0(5:8)^\top y_2 \\ u_0(1:4)v_0(5:8)^\top y_1 + v_2u_2^\top y_2 \\ u_0(5:8)v_0(5:8)^\top y_1 + C_{44}y_2 \end{bmatrix}.$$

*We orthogonalize outputs to the corresponding parts of the blacklist, $b$. For clarity, we write out what $proj_b$ means for each block:*

$$\begin{bmatrix} (C_{11} - proj_{u_0(1:4)}(C_{11}))x_1 \\ (u_1v_1^\top - proj_{u_0(1:4)}(u_1v_1^\top))x_1 \\ (C_{33} - proj_{v_0(1:4)}(C_{33})x_2 \\ (u_2v_2^\top - proj_{v_0(1:4)}(u_2v_2^\top))x_2 \end{bmatrix}, \quad \begin{bmatrix} (v_1u_1^\top - proj_{u_0(5:8)}(v_1u_1^\top))y_1 \\ (C_{22} - proj_{u_0(5:8)}(C_{22})y_1 \\ (v_2u_2^\top - proj_{v_0(5:8)}(v_2u_2^\top))y_2 \\ (C_{44} - proj_{v_0(5:8)}(C_{44})y_2 \end{bmatrix}.$$

*If these projected outputs are nonzero, we use the randomized SVD to recover matrices of the form $u_i v_i - proj_b(u_i v_i)$. If they are zero, the column and row spaces of these blocks are linear combinations of the corresponding vectors in b. In either case, the blocks of the matrix $C_{8,1} - C'_{8,1}$ are given by linear combinations of the vectors in b:*

$$C_{8,1} - C'_{8,1} = \left[\begin{array}{cc|cc} proj_b C_{11} & proj_b(v_1 u_1^\top) & \multicolumn{2}{c}{\multirow{2}{*}{0}} \\ proj_b(u_1 1_1^\top) & proj_b C_{22} & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{0}} & proj_b C_{33} & proj_b(v_2 u_2^\top) \\ & & proj_b(u_2 v_2^\top) & proj_b C_{44} \end{array}\right].$$

*Because the size of the blacklist b is 1, this is a rank-1 restricted HSS matrix. Thus, we learn it with 2 matrix-vector products using the linear system in Section 3.1.2.*

### General HODLR recovery

One can generalize the algorithm in Section 3.1.4 to more general HODLR recovery.

**Rank $k$ Symmetric HODLR** The symmetric rank-$k$ HODLR recovery algorithm generalizes that of Section 3.1.4. Let $E_{N,k}$ be a symmetric, $N \times N$, rank-$k$ HOLDR matrix. We use the notation of Figure 1.2, where $U_i, V_i$ are $N \times k$ matrices defining each block of $E_{N,k}$, and symmetry forces $Z_i = U_i$ and $W_i = V_i$ for all $i$. We recover $U_0$ and $V_0$ with high probability by the same technique as in Section 3.1.1. An $N \times k$ blacklist matrix $B_1$ concatenates $U_0$ and $V_0$:

$$B_1 = \begin{bmatrix} V_0 \\ U_0 \end{bmatrix}.$$

As before, we recurse on the structure of $E_{N,k}$ to recover blocks a level at a time. At level $\ell$, we construct an $N \times (k+p)$ input matrix and an $N \times k$ input matrix

given by

$$\begin{bmatrix} X_1 \\ \mathbf{0_{2^{n-\ell},k+p}} \\ \vdots \\ X_\ell \\ \mathbf{0_{2^{n-\ell},k+p}} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathbf{0_{2^{n-\ell},k}} \\ Y_1 \\ \vdots \\ \mathbf{0_{2^{n-\ell},k}} \\ Y_\ell \end{bmatrix},$$

where each of the blocks $X_i$ is $2^{n-\ell} \times (k+p)$ and $Y_i$ is $2^{n-\ell} \times k$. We replace $X_i$ and $Y_i$ with their projections onto the orthogonal space of the corresponding parts of the $m$ blacklist matrices, $B_1, \ldots B_m$. That is, we orthogonalize each column vector of $X_i$ and $Y_i$ to $k$ corresponding blacklist vectors. Then, products with these inputs zero out the blocks we have recovered, isolating the actions of the level-$\ell$ blocks. If these outputs are nonzero, the randomized SVD yields each level-$\ell$ block, using $2k+p$ matrix-vector products. We repeat this process until the block size is equal to the size of the blacklist, at which point projecting inputs sets them to 0.

If a subblock applied to a projected input outputs 0, it is considered a "failed" block as before. We mark it as failed and continue to recover what we can at the rest of the levels by projecting inputs. Then, we perform another pair of matrix-matrix products at each level $\ell$ with failed subblocks, as well as with the diagonal blocks. We project outputs, rather than inputs, onto the orthogonal space of the blacklist. We thus deduce these subblocks up to linear combinations of blacklist vectors.

If $E'_{N,k}$ is the matrix containing everything we have recovered thus far, we note that the subblocks of $E_{N,k} - E'_{N,k}$ are either zero blocks if we have completely recovered them, or nonzero linear combinations of the blacklist at the time of

recovering them, and thus of rank at most the size of the blacklist at that level. For example, a failed block at level 2 must be a linear combination of the $k$ vectors in $B_1$, so it is at most rank $k$ (some of the scalars in the linear combination may be 0).

We can bound on the rank of each of these subblocks by the size of the final blacklist. Thus, we view $E_{N,k} - E'_{N,k}$ as a restricted HSS matrix of rank equal to the size of the blacklist. If the blacklist is of size $m$, we perform $2m$ matrix-vector products to recover what remains of $E_{N,k}$ as described in Section 3.1.2.

We determine an upper bound on the number of matrix-vector queries to recover $E_{N,k}$. This reduces to a bound on the level at which we stop partitioning $E_{N,k}$. This happens when the dimension of the input space is less than $k+p$, i.e., at level $L$, where $2^{n-L} - (L-1) < k+p$. Because $L$ is a level, we trivially have the bound:

$$L \leq \log_2(N). \tag{3.10}$$

To derive a lower bound on $L$, we want an upper bound on $m$. The blacklist is largest if we append $k$ vectors to it at every level before $L$:

$$2^{n-L} - (L-1)k < k+p \implies L > \frac{W\left(\frac{2^{n+\frac{p}{k}}\log(2)}{k}\right)}{\log(2)} - \frac{p}{k},$$

where $W$ is the Lambert-$W$ function.

In the first step, where we project inputs, we do $2k+p$ matrix-vector queries at each level from 1 to $L$. Then, when we project outputs, we do at most the same number of queries. In the final step, using the algorithm in Section 3.1.2, we recover a restricted HSS matrix of rank at most the size of the blacklist, which is at most $k\log_2(N)$ by Equation (3.10). This will take $2k\log_2(N)$ queries. Thus, we

can bound the number of matrix-vector products to recover $E_{N,k}$:

$$\#\text{ matrix-vector products} \leq (6k + 2p)\log_2(N).$$

**Nonsymmetric complex-valued HODLR** If $A$ is not symmetric, when projecting inputs, we perform $2(k+p)$ matrix-vector products with $A$ and $2k$ matrix-vector products with $A^\top$ at each level. We apply the analogous randomized SVD result for complex matrices [85] to recover low-rank blocks. Then, we do at most the same number of queries for both $A$ and $A^\top$ at each level when we project outputs. We are left with a rank-$k\log_2(N)$ HSS matrix. Note that the same algorithm in Section 3.1.2 applies for generating the linear system to recover this HSS matrix, as it does not exploit the symmetry of the diagonal blocks. Thus, we use $2k\log_2(N)$ additional queries to generate the linear system. In sum, we require at most $(10k + 4p)\log_2(N) = \mathcal{O}((k+p)\log_2(N))$ matrix-vector products.

**HODLR matrices of any size** If the underlying HODLR matrix $A$ is of size $N \times N$ where $N$ is not a power of 2, then one can instead recover a version of $A$ padded with zeros to make its dimensions a power of 2. Let $\tilde{N} = 2^{\lceil \log_2(N) \rceil}$, $N_1 = \lfloor \tilde{N} - N \rfloor$, and $N_2 = \lceil \tilde{N} - N \rceil$. Then, the matrix is given by

$$B = \begin{bmatrix} \mathbf{0}_{N_1,N_1} & \mathbf{0}_{N_1,N} & \mathbf{0}_{N_1,N_2} \\ \mathbf{0}_{N,N_1} & A & \mathbf{0}_{N,N_2} \\ \mathbf{0}_{N_2,N_1} & \mathbf{0}_{N_2,N} & \mathbf{0}_{N_2,N_2} \end{bmatrix} \in \mathbb{R}^{\tilde{N}\times\tilde{N}}$$

is a rank-$k$ HODLR matrix. Here, $\mathbf{0}_{m,n}$ is the zero matrix of size $m \times n$. Instead of recovering $A$ directly, we recover $B$ and then remove the zero padding. Since we have only have access to $x \mapsto Ax$ and $x \mapsto A^\top x$, we infer the matrix-vector

products with $B$ and $B^\top$ as follows:

$$Bx = \begin{bmatrix} \mathbf{0}_{N_1,1} \\ Ay \\ \mathbf{0}_{N_2,1} \end{bmatrix}, \quad B^\top x = \begin{bmatrix} \mathbf{0}_{N_1,1} \\ A^\top y \\ \mathbf{0}_{N_2,1} \end{bmatrix}, \quad y = \begin{bmatrix} x_{N_1+1} \\ \vdots \\ x_{N_1+N} \end{bmatrix}.$$

Recovering $A$ is thus reduced to recovering an $\tilde{N} \times \tilde{N}$ HODLR matrix. This will require at most $(9k+2p)\lceil \log_2(N) \rceil = \mathcal{O}((k+p)\lceil \log_2(N) \rceil)$ matrix-vector products.

### Asymptotic Complexity

We determine the asymptotic complexity of the general HODLR recovery algorithm. It suffices to consider the generic HODLR algorithm, as the general HODLR algorithm described in Section 3.1.4 treats a general HODLR matrix as the sum of a generic HODLR matrix and a restricted HSS matrix, and the complexity of the p-HSS recovery algorithm is already described in Section 3.1.2.

To compute $T_{\text{HODLR}}$, the time to recover a generic HODLR matrix, we first compute $T_\ell$, the time it takes to recover level $\ell$. As in Section 3.1.2, we write this in terms of $T_A$ and $T_{\text{flop}}$. At level $\ell$, one multiplies $A$ by $2(k+p)$ inputs and $A^\top$ by $2k$ inputs, totaling a cost of $T_A(4k+2p)$. These inputs are projected so that their nonzero blocks are orthogonal to the corresponding blocks of the blacklist vectors. At level $\ell$, blocks are size $N/2^\ell$, and there are $\mathcal{O}(k\ell)$ blacklist vectors. Then, the cost of forming the projection matrix is $\mathcal{O}(T_{\text{flop}} N k^2 \ell^2)$. The cost of projecting $2(k+p)$ inputs is $\mathcal{O}(T_{\text{flop}} k(k+p)\ell N)$. Finally, the cost of the QR factorization done as part of the randomized SVD is $\mathcal{O}(T_{\text{flop}} N(k+p)^2)$. Adding all of this together yields:

$$T_\ell = \mathcal{O}(T_A(k+p) + T_{\text{flop}}(Nk^2\ell^2 + Nk(k+p)\ell + N(k+p)^2)). \tag{3.11}$$

Summing Equation (3.11) over all levels $\ell = 1, \ldots, \log_2(N)$ yields:

$$T_{\text{HODLR}} = \mathcal{O}(T_A(k+p)\log_2(N) + T_{\text{flop}} N k^2 (\log_2(N))^3). \tag{3.12}$$

For comparison, Martinsson's peeling algorithm has asymptotic complexity $\mathcal{O}(T_A((k+p)\log_2 N)) + T_{\text{flop}}(N(\log_2 N)^2 k^2)$.

### 3.1.5 Numerical Results

We plot the relative error of the HODLR recovery algorithms for increasingly large matrices. In particular, Figure 3.3 shows the results when our recovery algorithm is applied to a generic HODLR matrix. We also implement Martinsson's algorithm in [117] for comparison. To recover a matrix $A$, our algorithm generates $A_{\text{computed}}$. As before, we measure relative error using 20 iterations of the power method.

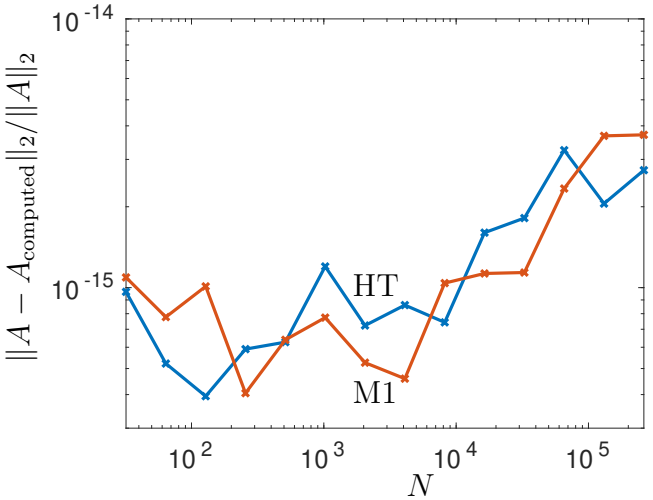



Figure 3.3: The relative error in recovering an $N \times N$ symmetric rank-1 random HODLR matrix. M1 refers to the recursive elimination strategy in [117] and our algorithm (HT) is described in Section 3.1.4. Excellent recovery accuracies are observed, and the potential numerical instability of recursive elimination is not seen.

It suffices to consider the numerical results for HSS and generic HODLR ma-

Table 3.1: HODLR recovery with error-prone matrix-vector products where the matrix-vector products are perturbed by Gaussian random vectors with expected norms of $10^{-4}$. In this situation, both recursive elimination (M1) and recursive projection (HT) provide accurate HODLR recovery, where the error is measured as $\|A - A_{\text{computed}}\|_2$. (Similar accuracies are achieved for larger values of $N$.)

| $N$ | 32 | 64 | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|---|---|
| M1 | $5.3\times10^{-4}$ | $5.6\times10^{-4}$ | $5.3\times10^{-4}$ | $4.4\times10^{-4}$ | $4.5\times10^{-4}$ | $5.0\times10^{-4}$ |
| HT | $5.3\times10^{-4}$ | $5.3\times10^{-4}$ | $4.4\times10^{-4}$ | $3.2\times10^{-4}$ | $3.8\times10^{-4}$ | $2.5\times10^{-4}$ |

Table 3.2: HSS recovery with error-prone matrix-vector products where the matrix-vector products are perturbed by Gaussian random vectors with expected norms of $10^{-4}$. In this situation, both recursive elimination (M1) and recursive projection (HT) provide accurate HSS recovery, where the error is measured as $\|A - A_{\text{computed}}\|_2$. (Similar accuracies are achieved for larger values of $N$.)

| $N$ | 32 | 64 | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|---|---|
| M1 | $5.3\times10^{-4}$ | $5.6\times10^{-4}$ | $5.3\times10^{-4}$ | $4.4\times10^{-4}$ | $4.5\times10^{-4}$ | $5.0\times10^{-4}$ |
| LM | $5.3\times10^{-4}$ | $5.6\times10^{-4}$ | $5.3\times10^{-4}$ | $4.4\times10^{-4}$ | $4.5\times10^{-4}$ | $5.0\times10^{-4}$ |
| HT | $5.3\times10^{-4}$ | $5.3\times10^{-4}$ | $4.4\times10^{-4}$ | $3.2\times10^{-4}$ | $3.8\times10^{-4}$ | $2.5\times10^{-4}$ |

trices because the algorithm described in Section 3.1.4 treats a general HODLR matrix as the sum of a generic HODLR matrix and an HSS matrix. The accuracy of our HSS recovery algorithm was discussed and illustrated in Section 3.1.3 and Figure 3.2. In Figure 3.3, we observe that our generic HODLR recovery algorithm performs just as well as Martinsson's peeling algorithm in [117], and both are very accurate. In addition, our algorithm's projection strategy makes it theoretically stable.

### 3.1.6 Applications to numerically rank-$k$ matrices and related problems

The HODLR and HSS matrix recovery algorithms outlined in Section 3.1.4 and Section 3.1.1 are not observed to be sensitive to noisy matrix-vector products and can

Table 3.3: Recovery of HODLR matrix with off-diagonal blocks of numerical rank 10. Here, M denotes the Martinsson's peeling algorithm, and H denotes our algorithm. Error is measured as $\|A - A_{\text{computed}}\|_2 / \|A\|_2$.

| N | 2048 | | 4096 | | 8192 | | 16384 | | 32768 | | 65536 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M | 1.6 $10^{-13}$ | × | 1.5 $10^{-13}$ | × | 1.5 $10^{-13}$ | × | 1.5 $10^{-13}$ | × | 1.4 $10^{-13}$ | × | 1.4 $10^{-13}$ | × |
| H | 1.9 $10^{-13}$ | × | 2.0 $10^{-13}$ | × | 2.0 $10^{-13}$ | × | 2.0 $10^{-13}$ | × | 1.9 $10^{-13}$ | × | 1.8 $10^{-13}$ | × |

Table 3.4: Recovery of HSS matrices with off-diagonal blocks of numerical rank 10. We observe that all three algorithms perform well, and our projection-based algorithm (H) has about one digit of precision more than Martinsson's peeling algorithm (M1), and two digits more than Martinsson's recent projection-based algorithm (M2). Error is measured as $\|A - A_{\text{computed}}\|_2 / \|A\|_2$.

| N | 2048 | | 4096 | | 8192 | | 16384 | | 32768 | | 65536 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M1 | 1.6 $10^{-13}$ | × | 1.5 $10^{-13}$ | × | 1.5 $10^{-13}$ | × | 1.4 $10^{-13}$ | × | 1.5 $10^{-13}$ | × | 1.6 $10^{-13}$ | × |
| M2 | 4.8 $10^{-12}$ | × | 1.8 $10^{-11}$ | × | 3.3 $10^{-11}$ | × | 8.7 $10^{-11}$ | × | 3.2 $10^{-10}$ | × | 9.2 $10^{-10}$ | × |
| H | 2.5 $10^{-14}$ | × | 4.6 $10^{-14}$ | × | 6.0 $10^{-14}$ | × | 7.2 $10^{-14}$ | × | 8.6 $10^{-14}$ | × | 1.2 $10^{-13}$ | × |

be applied in related contexts. In these experiments, error in the spectral norm is calculated via 20 iterations of the power method.

**Error-prone matrix-vector products.** Suppose we are more limited in the accuracy of our matrix-vector products. More precisely, consider a perturbation factor $\varepsilon$. Instead of applying a HODLR matrix $A$ to a vector $x_i$, our algorithm works with input-output pairs $(x_i, Ax_i + \varepsilon w_i)$, where $w_i$ is a length-$N$ vector whose entries are drawn from a standard random Gaussian distribution. Then we observe that both our p-HSS and HODLR algorithms outperform the existing algorithms. The results of this error-prone matrix-vector products recovery problem in Table 3.1 and Table 3.2 were made with the setting $\varepsilon = 10^{-5}$.

**Numerically rank-$k$ HSS and HODLR.** Suppose that the low-rank blocks of

an HSS or HODLR matrix are not exactly rank-$k$, but rather numerically rank-$k$. A matrix has numerical rank $k$ if its $k$ largest singular values are above a certain tolerance. We can apply our recovery algorithms to such matrices, as well as the existing algorithms in [117] and [105]. The results from these experiments are shown in Table 3.3 and Table 3.4. In this setting, the oversampling parameter $p$ plays an even more important role, as the probability of success of randomized SVD producing an accurate approximation to a numerically rank-$k$ matrix depends only on $p$. However, just as in the peeling algorithm of [117], it suffices to take $p = 5$ or 10.

## 3.2 Near-optimal hierarchical matrix approximation

We now consider the harder problem of near-optimal HODLR approximation to a matrix $A$ that can only be accessed through black-box matrix-vector product queries $x \mapsto Ax$ and matrix-transpose-vector product queries $x \mapsto A^\top x$.

In most practical situations, $A$ is not *exactly* HODLR. This has resulted in broad concern about whether peeling algorithms applied to non-HODLR matrices might produce inaccurate approximations [111, 28, 84, 30, 32]. Notably, a recent SIAM Linear Algebra Best Paper [35] poses an algorithmic challenge which roughly asks whether a near-HODLR matrix be approximated at nearly the same cost as algorithms for recovering an exactly HODLR matrix. Our work addresses this challenge. In particular, we study the following HODLR approximation problem, which makes no assumptions about $A$.

**Problem 3.** *Given an $n \times n$ matrix $A$, accessible only by matvec queries, a rank parameter $k \geq 1$, and an accuracy parameter $\Gamma \geq 1$, find a* HODLR($k$) *matrix $\widetilde{H}$*

*such that*

$$\|A - \widetilde{H}\|_{\mathsf{F}} \leq \Gamma \cdot \min_{H \in \text{HODLR}(k)} \|A - H\|_{\mathsf{F}}.$$

Interesting parameter regimes for Problem 3 include when $\Gamma = (1+\varepsilon)$ for $\varepsilon \in (0,1)$, or when $\Gamma = O(n^c)$ for a small constant $c$.

Problem 3 can be trivially solved for $\Gamma = 1$ using $n$ matrix-vector products. Via multiplication by an identity matrix, we can recover all entries of $A$ and then compute an exactly optimal HODLR approximation by computing optimal rank-$k$ approximations to $A$'s top right and bottom left $(n/2)\times(n/2)$ blocks, and recursing on the top left and bottom right blocks. However, outside this baseline and the case when $A$ is exactly HODLR, we are unaware of any non-trivial results on Problem 3. This is despite the vast literature on HODLR matrix approximation and on efficient matrix-vector query methods for vanilla low-rank approximation [49, 146, 51, 167, 10, 166, 11].

### 3.2.1 Main results

Our main contribution is an efficient algorithm for solving Problem 3.

**Theorem 3.2.1.** *Fix a rank parameter $k$ and accuracy parameter $\beta \in (0,1)$. Let $L = \lceil \log_2(n/k) \rceil$. There exists a non-adaptive[2] algorithm (Algorithm 3) which, in expected squared error, solves Problem 3 to accuracy $\Gamma = (1+\beta)^{L+1}$ using $O(k/\beta^3 \cdot L)$ matvec queries to $A$. Apart from matvec queries, the algorithm requires $O(n \cdot \textit{poly}(\log(n), k, \beta))$ additional runtime.*

We highlight two interesting instantiations of Theorem 3.2.1. For any $\varepsilon > 0$, we obtain accuracy $(1+\varepsilon)$ with $O(k \log^4(n/k)/\varepsilon^3)$ total matvec queries by setting

$\beta = O(\varepsilon/\log(n/k))$. Alternatively, for any constant $c > 0$, we obtain accuracy $n^c$ with $O(k\log(n/k))$ total matvec queries by setting $\beta = 2^c - 1 = \Theta(1)$. This matches the complexity of existing methods for solving the *recovery problem*, i.e., the case when $A$ is exactly HODLR.

Theorem 3.2.1 is obtained through a variant of the popular peeling algorithm for exact HODLR matrix recovery. This algorithm obtains an approximation via a "top down" approach. Specifically, it computes low-rank approximations to $A$'s top right and bottom left blocks via randomized sketching, implicitly subtracts the results from $A$, and then recurses on the upper left and lower right blocks. A major technical challenge for applying the peeling algorithm to Problem 3 is how to control error that can accumulate across levels of recursion when $A$'s top right and lower left blocks are not *exactly* rank-$k$ (i.e., when the matrix is not exactly HODLR). This potential accumulation of error has been highlighted in several prior works, including in the earliest work on the peeling method [111, 28, 30], and has been the key challenge in extending peeling to the solve the HODLR approximation problem.

We resolve this long-standing challenge using two new techniques. First, we prove that if peeling is implemented with the so-called *Generalized Nyström Method* for low-rank approximation [49, 130], sufficient oversampling leads to an algorithm that requires $kL/\text{poly}(\beta)$ matrix-vector products to achieve error $\Gamma = (1+\beta)^{L+1}$. Our proof requires a novel perturbation analysis of sketching for low-rank approximation, and brings to light a surprising fact: the same result cannot be obtained if the more standard *Randomized SVD* method [85] is used for low-rank approximation within peeling. Second, we obtain our final result (with a better polynomial dependence on $\beta$) by combining peeling with a new "randomly perforated"

sketching distribution that allows for even stronger control of error buildup across recursive levels. Details of the existing peeling algorithm are given in Section 1.4.1, and our improvements are described in Section 3.2.3.

We complement our upper bound from Theorem 3.2.1 with a nearly matching lower-bound.

**Theorem 3.2.2.** *For any $\varepsilon > 0$, any (possibly adaptive and randomized) algorithm that solves Problem 3 with error $\Gamma = (1+\varepsilon)$ and probability $\geq 1/25$ requires $\Omega(k\log(n/k) + k/\varepsilon)$ matvec queries with $A$.*[3] *Moreover, any algorithm that solves Problem 3 for any finite $\Gamma$ and any non-zero probability, requires $\Omega(k\log(n/k))$ matvec queries.*

Theorem 3.2.2 establishes that our $O(k\log(n/k))$ query result to achieve error $n^c$ from Theorem 3.2.1 cannot be improved in terms of the number of matvec queries, although it might be possible to obtain better error (e.g., a $\log(n)$ or even constant factor approximation) with the same number of matvecs. Additionally, Theorem 3.2.2 establishes that our $O(k\log^4(n/k)/\varepsilon^3)$ query result for $(1+\varepsilon)$ error cannot be improved by more than log factors and a $1/\varepsilon^2$ factor. Interestingly, the lower bound shows a separation between the complexity of vanilla low-rank approximation and hierarchical low-rank approximation in terms of dependence on $\varepsilon$. The best known low-rank approximation algorithms use just $\tilde{O}(k/\varepsilon^{1/3})$ matrix-vector products to achieve relative error $(1+\varepsilon)$ [10, 124].[4] In contrast, Theorem 3.2.2 shows that a sublinear dependence on $1/\varepsilon$ *cannot* be achieved for HODLR matrix approximation.

[3]Formally, for a fixed constant $c$, we show that there is a distribution over inputs $A \in \mathbb{R}^{n\times n}$ for $n \geq ck/\epsilon$ such that any (possibly randomized) algorithm using $O(k\log(n/k) + k/\epsilon)$ matvecs succeeds with probability $< 1/25$ over the randomness of the algorithm and the choice of input.

[4]The best known lower bound for vanilla $k$-rank approximation is $\Omega(k + 1/\varepsilon^{1/3})$ matrix-vector products [10]. Combining the $k$ and $\varepsilon$ terms, as we do in our Theorem 3.2.2 for hierarchical approximation, is an open question.

The proof of Theorem 3.2.2 builds on a growing body of work on lower bounds for adaptive matrix-vector product algorithms [153, 37, 48], which require techniques beyond those used to prove lower bounds, e.g., in the non-adaptive sketching setting. Our proof draws specifically on a recent result on the number of matrix-vector products required to obtain an optimal block diagonal approximation to a matrix $A$ [6].

**Organization.** The remainder of the section is organized as follows. In Section 1.4.1, we describe the classic peeling algorithm for HODLR matrix recovery, and discuss the challenges in applying it to the HODLR matrix approximation problem (Problem 3). In Section 3.2.3, we discuss the techniques we use to analyze our variant of peeling, including our new perturbation bound for low-rank approximation. Then, in Section 3.2.4, we describe the exact implementation of our algorithm and introduce notation required for our main analysis. The final proof of Theorem 3.2.1 is given in Section 3.2.7. In Section 3.2.9, we prove our lower bound, Theorem 3.2.2. Finally, in Section 3.2.10 we show the results of some numerical experiments.

### 3.2.2 Peeling in the presence of error

The peeling algorithm described in Section 1.4.1 assumes that the matrix $A$ is exactly HODLR($k$). In particular, Observation 2 does not hold if the off-diagonal blocks at the previous level are not recovered exactly. This is illustrated in Figure 3.4.

As noted in [111], "it is a natural concern that whether the error from low-rank

decompositions on top levels accumulates in the peeling steps." In particular, if all of the error at a given level propagated to the next level through perturbations on the desired sketches, the error could double at each level; i.e., result in an *exponential* blow-up of the error with respect to the number of levels. In fact, as we demonstrate in Section 3.2.11, certain variants of the peeling algorithm can exhibit such a failure mode on hard instances. Thus, understanding the propagation of error from one level to the next is critical in the design and analysis of peeling algorithms. In Section 3.2.3 we discuss the techniques we use to control and analyze this error propagation.

### 3.2.3 Techniques

As discussed in Section 3.2.2, when the peeling algorithm is applied to a matrix $A$ that is not exactly HODLR, errors at previous levels can pollute the matrix-vector products at a given level. The aim of this paper is to show that these errors can controlled in such a way that the peeling algorithm can still solve Problem 3 to high accuracy. To do this we must understand: (1) how matrix-vector query algorithms for low-rank approximation are impacted by noise in their matvecs and (2) how this noise can be controlled in the setting of peeling.[5] In the next two subsections we outline the high-level ideas we use to address each question.

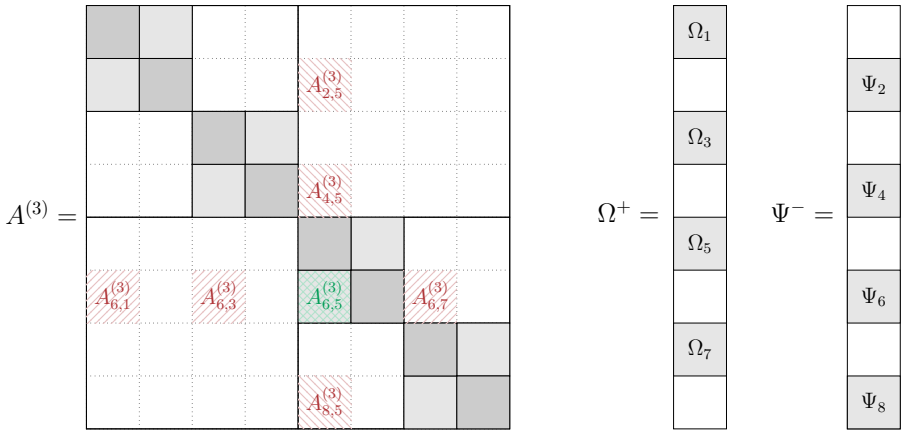



Figure 3.4: State of the hierarchical matrix at the start of the $\ell = 3$ level of peeling. The matrix $A^{(3)}$ denotes $A$ after subtracting off-diagonal blocks that were (approximately) recovered at the first two levels. We would like to use the sketches $A^{(3)}_{6,5}\Omega_5$ and $(A^{(3)}_{6,5})^\top\Psi_6$ to obtain a low-rank approximation to the off-diagonal block $A^{(3)}_{6,5}$. However, if the off-diagonal blocks at previous levels were not recovered exactly (since these blocks may not be exactly rank-$k$), then after subtraction, these blocks will not be exactly zero in $A^{(3)}$. We thus obtain perturbed sketches of the form $A^{(3)}_{6,5}\Omega_5 + A^{(3)}_{6,1}\Omega_1 + A^{(3)}_{6,3}\Omega_3 + A^{(3)}_{6,7}\Omega_7$ and $(A^{(3)}_{6,5})^\top\Psi_6 + (A^{(3)}_{2,5})^\top\Psi_2 + (A^{(3)}_{4,5})^\top\Psi_4 + (A^{(3)}_{8,5})^\top\Psi_8$.

### A perturbation bound for low-rank approximation

To understand how matvec query errors impact low-rank approximation algorithms, including RSVD and the Generalized Nyström Method (see Section 1.3), we develop the following perturbation bound, which we believe will be of general interest. The bound is proven in Section 3.2.7.

**Theorem 3.2.3.** *Let $B \in \mathbb{R}^{m_1 \times m_2}$, $\Omega \in \mathbb{R}^{m_2 \times s}$, $E_1 \in \mathbb{R}^{m_1 \times s}$, and $E_2 \in \mathbb{R}^{s \times m_2}$,*

[5] In [32], a similar approach is taken to analyze an infinite dimensional variant of the peeling algorithm for the task of approximating the Green's function of an elliptic partial differential (PDE) operator by a HODLR operator. However, [32] relies strongly on the fact that the off-diagonal blocks of the Green's function have rapid spectral decay and are thus low-rank, up to exponentially small error [18]. This allows the overall error to be controlled, even if the error can grow exponentially across levels. In contrast, we make no assumptions about $A$.

*and let* $Q = \text{orth}(B\Omega + E_1)$. *Write* $B$ *in its singular value decomposition as*

$$B = U\Sigma V^\top = \begin{bmatrix} U_{\text{top}} & U_{\text{bot}} \end{bmatrix} \begin{bmatrix} \Sigma_{\text{top}} & \\ & \Sigma_{\text{bot}} \end{bmatrix} \begin{bmatrix} V_{\text{top}}^\top \\ V_{\text{bot}}^\top \end{bmatrix},$$

*where* $U$ *and* $V$ *are orthonormal and* $\Sigma$ *is diagonal, and the "top" blocks represent the top* $k$ *columns; i.e.* $U_{\text{top}} \in \mathbb{R}^{m_1 \times k}, V_{\text{top}} \in \mathbb{R}^{m_2 \times k}$, *and* $\Sigma_{\text{top}} \in \mathbb{R}^{k \times k}$. *Define also*

$$\Omega_{\text{top}} = V_{\text{top}}^\top \Omega, \quad \Omega_{\text{bot}} = V_{\text{bot}}^\top \Omega.$$

*Then, assuming* $\text{rank}(\Omega_{\text{top}}) = k$,

$$\|B - Q(Q^\top B + E_2)_k\|_{\mathsf{F}} \leq \|E_1 \Omega_{\text{top}}^\dagger\|_{\mathsf{F}} + 2\|E_2\|_{\mathsf{F}} + \left(\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 + \|\Sigma_{\text{bot}}\Omega_{\text{bot}}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2\right)^{1/2}.$$

Theorem 3.2.3 is most naturally a perturbation bound for the RSVD method, which, as discussed in Section 1.3, first computes an orthonormal basis $Q$ for $B\Omega$ where $\Omega$ is a random sketching matrix, and then outputs $Q(Q^\top B)_k$ – the best rank-$k$ approximation to $B$ lying in the span of $Q$. The error term $E_1$ captures error that occurs during the initial sketching step, i.e., due to noise in computing the matrix-vector products $B\Omega$. Similarly, $E_2$ captures noise in the second projection step. Theorem 3.2.3 can be used to recover the standard bounds for RSVD implemented with exact matrix-vector products (i.e., where $E_1$ and $E_2$ are zero). In particular, in this case the RSVD Frobenius error is bounded by:

$$\begin{aligned} \|B - Q(Q^\top B)_k\|_{\mathsf{F}} &\leq \left(\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 + \|\Sigma_{\text{bot}}\Omega_{\text{bot}}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2\right)^{1/2} \\ &= \left(\|B - B_k\|_{\mathsf{F}}^2 + \|\Sigma_{\text{bot}}\Omega_{\text{bot}}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2\right)^{1/2}. \end{aligned}$$

When the sketching matrix $\Omega$ is Gaussian, we can observe that $\Omega_{\text{bot}}$ and $\Omega_{\text{top}}$ are independent Gaussian matrices since $V_{\text{top}}$ and $V_{\text{bot}}$ are orthogonal to each other. This allows us to apply a standard bound (see Theorem 3.2.9) to argue that when $s = O(k/\beta)$, $\mathbb{E}[\|\Sigma_{\text{bot}}\Omega_{\text{bot}}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2] \leq \beta\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 = \beta\|B - B_k\|_{\mathsf{F}}^2$, which ultimately gives that $\mathbb{E}[\|B - Q(Q^\top B)_k\|_{\mathsf{F}}] \leq (1+\beta)\|B - B_k\|_{\mathsf{F}}$.

Theorem 3.2.3 can also be used as a perturbation bound for the Generalized Nyström Method. In this case, $E_2$ is used to account for the error due to using an approximate projection onto $Q$ via a sketched regression problem, as well as the errors in the matvecs with $A^\top$ used to set up the regression problem. We discuss this further in Section 3.2.3. As with RSVD, Theorem 3.2.3 matches the current best known bounds for Generalized Nyström implemented with exact matvecs.

### Low-rank approximation in the peeling algorithm

**Randomized SVD.** Now consider implementing peeling with RSVD as the base low-rank approximation algorithm. In this case, $E_1$ and $E_2$ will be nonzero since we cannot compute exact matrix-vector products with an off-diagonal low-rank block $B$ due to noise from previous levels (i.e., we cannot exactly compute $B\Omega$ and $Q^\top B$). Fortunately, the error $E_1$ that arises in peeling has a particular structure that can be used to bound the $\|E_1\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}$ term in Theorem 3.2.3. In particular, recalling Figure 3.4, we will have

$$E_1 = \sum_j M_j\Omega_j = M\widetilde{\Omega}, \qquad M = \begin{bmatrix} M_1 & M_2 & \cdots \end{bmatrix}, \qquad \widetilde{\Omega} = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \vdots \end{bmatrix}, \tag{3.13}$$

where the $M_j$ are fixed matrices that depend on the recovery error at the previous levels of peeling, and the $\Omega_j$ are independent Gaussian matrices.[6] $\widetilde{\Omega}$ is itself a Gaussian matrix that is independent from $\Omega_{\text{top}}$, allowing us to bound the $\mathbb{E}[\|E_1\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2]$ term in Theorem 3.2.3 by $\mathbb{E}[\|E_1\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2] = \mathbb{E}[\|M\widetilde{\Omega}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2] \leq \beta\|M\|_{\mathsf{F}}^2$ when $s = O(k/\beta)$, again using Theorem 3.2.9. I.e., with a large enough sketch size, we can drive down the error term $E_1$ to be arbitrarily small, and ensure that it does not accumulate too much across the levels of peeling.

[6] In Figure 3.4, the $M_j\Omega_j$ terms when $B = A_{6,5}^{(3)}$ are $A_{6,1}^{(3)}\Omega_1$, $A_{6,3}^{(3)}\Omega_3$, and $A_{6,7}^{(3)}\Omega_7$.

Unfortunately, we cannot handle the $\|E_2\|_{\mathsf{F}}$ term in Theorem 3.2.3 in a similar way. We have $E_2 = \sum_j M_j Q_j$ for orthogonal matrices $Q_j$ computed for each off-diagonal block at the current level. The $Q_j$ and $M_j$ matrices may be arbitrarily correlated, and so it is not clear that $\|E_2\|_{\mathsf{F}}$ can be made small compared to the noise from the previous levels, $\|M\|_{\mathsf{F}}$. In fact, as discussed in Section 3.2.11, theoretical and empirical evidence suggests the existence of hard input instances where standard peeling with RSVD can suffer exponential error blow-up across levels.

**Generalized Nyström.** Our first key observation is that when peeling is implemented with the Generalized Nyström method as the base low-rank approximation algorithm, rather than RSVD, the $\|E_2\|_{\mathsf{F}}$ error term in Theorem 3.2.3 can in fact be bounded. For peeling implemented with Generalized Nyström, $E_2$ encapsulates two sources of error: the first is due to the fact that Generalized Nyström does not exactly return $Q(Q^\top B)_k$, but rather computes $X = \operatorname{argmin}_Z \|\Psi^\top B - \Psi^\top Q Z\|_{\mathsf{F}}$ for a random sketching matrix $\Psi$ and returns $QX_k$. We can think of $X$ as an approximation to $Q^\top B = \operatorname{argmin}_Z \|B - QZ\|_{\mathsf{F}}$. It is well known that this source of error can be controlled by setting the size of the sketch $\Psi$ large enough – this leads to the standard analysis of Generalized Nyström from the literature [167].

The second source of error in $E_2$ is due to noise in computing the matrix-vector products $\Psi^\top B$. Fortunately, analogously to how we bounded $E_1$ for RSVD in Section 3.2.3, we can leverage the fact that in peeling, this noise has the structure of equation (3.13) – it is the product of a fixed matrix $M$ (the recovery error at previous levels of peeling) and a random Gaussian matrix $\widetilde{\Psi}$ that is independent of $\Psi$. Using this structure, we are able to bound its impact on the final approximation quality. In particular, we use Theorem 3.2.3 to give the following bound for the

Generalized Nyström Method with Gaussian errors as in Equation (3.13). This bound is proven in Section 3.2.7.

**Theorem 3.2.4.** *Let $B \in \mathbb{R}^{m_1 \times m_2}$, $M \in \mathbb{R}^{m_1 \times p}$, and $N \in \mathbb{R}^{q \times m_2}$ be fixed matrices, and let $\Omega \in \mathbb{R}^{m_2 \times s_{\mathrm{R}}}$, $\widetilde{\Omega} \in \mathbb{R}^{p \times s_{\mathrm{R}}}$, $\Psi \in \mathbb{R}^{m_1 \times s_{\mathrm{L}}}$, and $\widetilde{\Psi} \in \mathbb{R}^{q \times s_{\mathrm{L}}}$ be independent matrices with entries drawn independently from $\mathcal{N}(0,1)$. Define*

$$Q := \mathrm{orth}(B\Omega + M\widetilde{\Omega}), \qquad X := (\Psi^\top Q)^\dagger (\Psi^\top B + \widetilde{\Psi}^\top N).$$

*Then, provided $s_{\mathrm{R}} > 2k+1$ and $s_{\mathrm{L}} > 2s_{\mathrm{R}} + 1$*

$$\mathbb{E}\left[\|B - QX_k\|_{\mathsf{F}}^2\right] \le E_1 + E_2 + 2\sqrt{E_1 E_2},$$

*where*

$$E_1 := \left(1 + \frac{k}{s_{\mathrm{R}} - k - 1}\right)\|B - B_k\|_{\mathsf{F}}^2$$
$$E_2 := \frac{18k}{s_{\mathrm{R}} - k - 1}\|M\|_{\mathsf{F}}^2 + \frac{8s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1}\|N\|_{\mathsf{F}}^2 + \frac{32 s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1}\|B - B_k\|_{\mathsf{F}}^2.$$

We can see from Theorem 3.2.4 that, for fixed noise matrices $M$ and $N$, if we set $s_{\mathrm{R}}$ large enough compared to $k$ and $s_{\mathrm{L}}$ large enough compared to $s_{\mathrm{R}}$, we can drive the expected error of the rank-$k$ approximation $QX_k$ arbitrarily close to the error of the best possible rank-$k$ approximation $B_k$. Formally, in Section 3.2.7 (Theorem 3.2.13) we use Theorem 3.2.4 to show that for any $\beta \in (0,1)$, if we set $s_{\mathrm{R}} = O(k/\beta^2)$ and $s_{\mathrm{L}} = O(s_{\mathrm{R}}/\beta^2) = O(k/\beta^4)$, then at each level of peeling, our approximation error blows up by at most a $(1+\beta)$ factor. Over $L+1$ levels, we obtain final accuracy $\Gamma = (1+\beta)^{L+1}$. This matches our main result, Theorem 3.2.1, but with a slightly worse dependence on $\beta$: we require $O(k/\beta^4 \cdot L)$ rather than $O(k/\beta^3 \cdot L)$ total matvecs. In Section 3.2.3 we show how to give the improved $\beta$ dependence by using a novel sketching approach to further control the noise terms $M$ and $N$.

We note that the algorithm described above, which simply implements standard peeling with Generalized Nyström is particularly simple and performs well experimentally – see Section 3.2.10. It would be interesting to understand if our bounds on $s_{\text{R}}$ and $s_{\text{L}}$ are tight – we suspect that they are not. In particular, even in the case of exact matrix-vector products, we suspect that $s_{\text{L}} = O(k/\beta^2)$ suffices to achieve a $(1+\beta)$ approximation, even though current best known bound is $O(k/\beta^3)$. Giving an improved bound in this setting would very likely yield an improved bound in our setting.

### Randomly perforated Gaussian sketching

We next discuss how we can further improve the query complexity of peeling with Generalized Nyström-based low-rank approximation by altering the sketching distribution to explicitly reduce matrix-vector product errors that arise due to inexact recovery within the peeling algorithm.

Referring back to Figures 1.3 and 3.4, we observe that at each level the peeling algorithm employs two random sketching matrices on the right: $\Omega^+$ and $\Omega^-$ and two on the left: $\Psi^+$ and $\Psi^-$. For sake of exposition we focus here just on the right sketches, $\Omega^+$ and $\Omega^-$, which are both block matrices with $2^\ell$ blocks, alternating between random Gaussian blocks (i.e., $\Omega_1$, $\Omega_2$, $\Omega_3$, ...) and zero blocks. In particular, $\Omega^+$ has even blocks set to zero, while $\Omega^-$ has odd blocks set to zero.

This "perforated" structure arises naturally from Observation 1 and Observation 2. Intuitively, since $\Omega^+$ has even blocks set to zero, it has no interaction with the diagonal blocks at level $\ell$ with even indices. Thus, it can be used to simultaneously produce right sketches of every bottom-left off-diagonal block at level $\ell$, without incurring any error due to the unrecovered on-diagonal blocks.

As discussed in Section 3.2.2 and illustrated in Figure 3.4, when used in the peeling algorithm applied to a non-HODLR matrix, the sketch does incur error due to inexact recovery at the previous levels. In particular, referring to Figure 3.4, when approximating a given off-diagonal block at level $\ell$ (e.g., $A^{(3)}_{6,5}$ in the figure), the sketch incurs error from $2^{\ell-1}-1$ blocks that were only approximately recovered in previous levels of peeling (i.e., $A^{(3)}_{6,1}$, $A^{(3)}_{6,3}$, and $A^{(3)}_{6,7}$ in the figure).

Our key idea to reduce this error is simple: we will increase the sparsity of $\Omega^+$ and $\Omega^-$, so that a higher fraction of blocks are set to zero. Thus, when recovering each block at level $\ell$, we will incur error due to a smaller number of inexactly recovered blocks from the previous levels. We still need non-zero blocks for each of the $2^\ell$ off-diagonal blocks that are recovered at level $\ell$, so our sparser matrices will have a large number of block columns than before. See Figure 3.5 for an illustration.

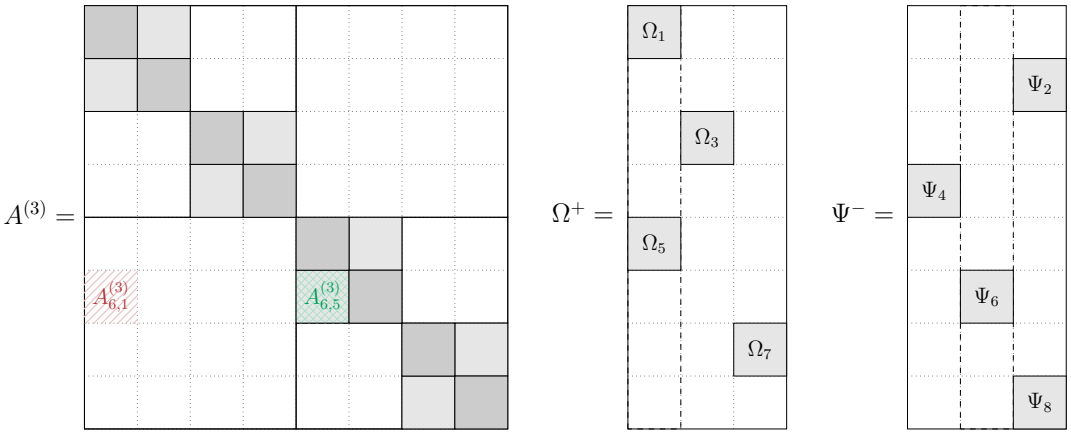



Figure 3.5: As in Figures 1.3 and 3.4, we aim to obtain the sketches $A^{(3)}_{j\pm1,j}\Omega^+_j$ and $(A^{(3)}_{j\pm1,j})^\top\Psi^-_{j\pm1}$ from the sketches $A^{(3)}\Omega^+$ and $(A^{(3)})^\top\Psi^-$. The key observation is that by using *randomly perforated Gaussian sketches*, which decrease the number of nonzero blocks per block-column (while increasing the number of column. This results in less error. We illustrate the error for block $A^{(3)}_{6,5}$. Note that we obtain sketches $A^{(3)}_{6,5}\Omega_5 + A^{(3)}_{6,1}\Omega_1$ and $(A^{(3)}_{6,5})^\top\Psi_6$. Thus, the error is decreased compared to Figure 3.4.

Of course, just decreasing the number of inexactly recovered blocks that introduce error into each of our sketches may not decrease the magnitude of this error if the error is all concentrated on a few blocks. Thus, we will chose the nonzero blocks of our sketches randomly, ensuring that the expected error when recovering each block at level $\ell$ is small. Formally, our modified sketches $\Omega^+, \Omega^-, \Psi^+$, and $\Psi^-$ will be $2^\ell \times t$ block matrices, with a single Gaussian block placed randomly in each odd (resp. even) block row. They can be thought of as block Kronecker products of Gaussian matries with *Count-sketch-like* sparse sketching matrices – see Section 3.2.4 for a more complete description. We call our sketches *randomly perforated Gaussian sketches*. By increasing the number of block columns $t$ in our randomly perforated sketches, we decrease the expected matvec error introduced by inexactly recovered blocks at each level. In particular, we show that the expected noise added to each matvec query is decreased by a factor of $1/t$ in the squared Frobenius norm. We describe our variant of Generalized Nyström-based peeling with perforated sketching in Algorithm 3 and analyze it in Section 3.2.7 (Theorem 3.2.13), using the perturbation bound for Generalized Nyström described in Section 3.2.3. Ultimately, this approach gives our main result Theorem 3.2.1.

We note that perforated sketching can also be combined with vanilla RSVD-based peeling, giving a similar bound to Theorem 3.2.1. We describe the resulting algorithm in Section 3.2.6. We numerically compare our main Generalized Nyström Method-based algorithm with the RSVD-based algorithm in Section 3.2.10.

### 3.2.4 The Generalized Nyström Method Peeling algorithm

We now describe our the variant of the peeling algorithm, which is summarized as Algorithm 3. In Section 3.2.4 we describe the notation we use for the algorithm

and provide the pseudocode for Algorithm 3. Then, in sections Section 3.2.4 and Section 3.2.4 we describe how the two key stages of the Generalized Nyström Method are approximately implemented in the peeling algorithm and introduce notation we will use in the analysis. Finally, in Section 3.2.5 we discuss how to recover the diagonal blocks at the final level.

### Notation for sketching distributions

The sketching matrices the peeling algorithm use to recover the matrices in Equation (3.17) depends on the parity of the column index $j$. To handle the two cases simultaneously in our analysis, we will introduce the following notation. Fix a value $j \in \{1, \ldots, d\}$. We will use "$\pm$" and "$\mp$" to indicate addition or subtraction depending on the parity of $j$:

$$\pm := \begin{cases} + & j \text{ odd} \\ - & j \text{ even} \end{cases}, \qquad \mp := \begin{cases} - & j \text{ odd} \\ + & j \text{ even} \end{cases}. \tag{3.14}$$

For instance, $j \pm 1$ means $j + 1$ if $j$ is odd and $j - 1$ if $j$ is even and $\xi_j^{\pm}$ means $\xi_j^+$ if $j$ is odd and $\xi_j^-$ if $j$ is even.

**Definition 3.2.5.** *Let $X \in \mathbb{R}^{p \times v}$ and $Y \in \mathbb{R}^{pu \times t}$. We define the product $X \bullet Y \in \mathbb{R}^{pu \times vt}$ by*

$$\underbrace{\begin{bmatrix} x_{1,1} & \cdots & x_{1,v} \\ \vdots & & \vdots \\ x_{p,1} & \cdots & x_{p,v} \end{bmatrix}}_{X} \bullet \underbrace{\begin{bmatrix} Y_1 \\ \vdots \\ Y_p \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} x_{1,1}Y_1 & \cdots & x_{1,v}Y_1 \\ \vdots & & \vdots \\ x_{p,1}Y_p & \cdots & x_{p,v}Y_p \end{bmatrix}}_{X \bullet Y},$$

*where $Y_1, \ldots, Y_p \in \mathbb{R}^{u \times t}$.*

Note that " $\bullet$ " is a block row-wise Kronecker product; i.e. the block-rows of

$X \bullet Y$ are obtained as the Kronecker product of the rows of $X$ and the block-rows of $Y$.

**Definition 3.2.6.** *We say* $\xi \sim \text{CountSketch}(d, t)$ *if* $\xi \in \{0,1\}^{d \times t}$ *is such that each row of* $\xi$ *independently has exactly one nonzero entry in a uniformly random column. We respectively denote the* $(j, i)$ *entry of* $\xi$ *as* $\xi_{j,i}$.

**Definition 3.2.7.** *We say* $\xi^+, \xi^- \sim \text{PerfCountSketch}(d, t)$ *if*

$$\xi^+ = \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots \end{bmatrix}^\top \bullet \xi, \qquad \xi^- = \begin{bmatrix} 0 & 1 & 0 & 1 & \cdots \end{bmatrix}^\top \bullet \xi,$$

*where* $\xi \sim \text{CountSketch}(d, t)$. *We respectively denote the* $(j, i)$ *entry of* $\xi^+$ *and* $\xi^-$ *as* $\xi^+_{j,i}$ *and* $\xi^-_{j,i}$.

We are now prepared to describe the main "randomly perforated Gaussian" sketching distribution that we will use in our variant of the peeling algorithm.

**Definition 3.2.8.** *We say* $\xi^+, \xi^-, \Omega^+, \Omega^- \sim \text{RandPerfGaussian}(n, d, s, t)$ *if*

$$\Omega^+ = \xi^+ \bullet \begin{bmatrix} \Omega_1 \\ \vdots \\ \Omega_d \end{bmatrix}, \qquad \Omega^- = \xi^- \bullet \begin{bmatrix} \Omega_1 \\ \vdots \\ \Omega_d \end{bmatrix},$$

*where* $\xi^+, \xi^- \sim \text{PerfCountSketch}(d, t)$ *and each* $\Omega_i \in \mathbb{R}^{n/d \times s}$ *is independent with entries drawn independently from* $\mathcal{N}(0, 1)$.

The sketching matrices $\Omega^+$ and $\Psi^-$ shown in Figure 3.5 illustrate the sparsity structure of the randomly perforated Gaussian sketching distribution Definition 3.2.8 with $t = 3$. In the case that $t = 1$, then the randomly perforated Gaussian sketching matrices are of the form used by the standard peeling algorithm, as illustrated in Figure 1.3.

**Notation for iteration and pseudocode**

At level $\ell$, the peeling algorithm aims to approximate the $2^\ell$ low-rank off-diagonal blocks of $A$, each of which is of size $n/2^\ell$. The resulting approximation to these blocks is placed in a matrix $H^{(\ell)}$. This is repeated for $L := \lceil \log_2(n/k) \rceil \leq O(\log(n/k))$ levels, at which points the blocks are of size at most $k$. At the final level $\ell = L$, the algorithm also obtains an approximation $\widehat{H}$ to the $2^L$ diagonal blocks. The final approximation is then expressed in terms of the approximations at each level as

$$\widetilde{A} = H^{(1)} + \cdots + H^{(L)} + \widehat{H}. \tag{3.15}$$

Suppose we are at level $\ell$, and let $A^{(\ell)}$ be the matrix $A$ at step $\ell$ after subtracting the approximations $H^{(\ell-1)}, \ldots, H^{(1)}$ from the previous levels; that is

$$A^{(\ell)} := A - (H^{(\ell-1)} + \cdots + H^{(1)}).$$

Partition $A^{(\ell)}$ into a $2^\ell \times 2^\ell$ block matrix with blocks of size $(n/2^\ell) \times (n/2^\ell)$:

$$A^{(\ell)} = \begin{bmatrix} A^{(\ell)}_{1,1} & A^{(\ell)}_{1,2} & \cdots & A^{(\ell)}_{1,d} \\ A^{(\ell)}_{2,1} & A_{2,2} & \cdots & A^{(\ell)}_{2,2^\ell} \\ \vdots & \vdots & & \vdots \\ A^{(\ell)}_{d,1} & A^{(\ell)}_{2^\ell,2} & \cdots & A^{(\ell)}_{2^\ell,2^\ell} \end{bmatrix}. \tag{3.16}$$

We will use the Generalized Nyström Method to simultaneously approximate the relevant factors

$$A^{(\ell)}_{2,1},\ A^{(\ell)}_{1,2},\ A^{(\ell)}_{4,3},\ A^{(\ell)}_{3,4},\ \ldots,\ A^{(\ell)}_{2^\ell-1,2^\ell}, \tag{3.17}$$

with low-rank matrices $H^{(\ell)}_1 \approx A^{(\ell)}_{2,1}$, $H^{(\ell)}_2 \approx A^{(\ell)}_{1,2}$, ..., $H^{(\ell)}_d \approx A^{(\ell)}_{2^\ell-1,2^\ell}$. The low-rank

factors obtained will then be placed in the matrix

$$H^{(\ell)} = \text{block-tridiag}\begin{pmatrix} H_2^{(\ell)} & 0 & H_4^{(\ell)} & 0 & \cdots & H_{2^\ell}^{(\ell)} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ H_1^{(\ell)} & 0 & H_3^{(\ell)} & 0 & \cdots & H_{2^\ell-1}^{(\ell)} \end{pmatrix}, \tag{3.18}$$

and the algorithm proceeds to the next level.

The diagonal factors $A_{j,j}^{(\ell)}$ may be large, and the sketching matrices used by the peeling algorithm are constructed explicitly to avoid touching these diagonal blocks. Assuming the algorithm has not accumulated much error, then besides the diagonal blocks and sub/super diagonal blocks we aim to recover Equation (3.17), the other blocks will be on the scale of the best possible error. In particular, as described in Section 1.4.1, if $A$ is exactly HODLR, then these blocks are all zero.

At the final level $\ell = L$, we must also approximate the on-diagonal blocks. Let $\widehat{A}^{(L)}$ be the matrix after removing our approximation to the off-diagonal blocks of $A^{(L)}$; i.e. $\widehat{A}^{(L)} := A - (H^{(L)} + \cdots + H^{(1)})$. Partition $\widehat{A}^{(L)}$ into a $2^L \times 2^L$ block matrix with blocks of size $(n/2^L) \times (n/2^L)$:

$$\widehat{A}^{(L)} = \begin{bmatrix} \widehat{A}_{1,1}^{(L)} & \widehat{A}_{1,2}^{(L)} & \cdots & \widehat{A}_{1,2^L}^{(L)} \\ \widehat{A}_{2,1}^{(L)} & \widehat{A}_{2,2} & \cdots & \widehat{A}_{2,2^L}^{(L)} \\ \vdots & \vdots & & \vdots \\ \widehat{A}_{2^L,1}^{(L)} & \widehat{A}_{2^L,2}^{(L)} & \cdots & \widehat{A}_{2^L,2^L}^{(L)} \end{bmatrix}. \tag{3.19}$$

We note that $\widehat{A}_{i,j}^{(L)} = A_{i,j}^{(L)}$ for all $i \neq j \pm 1$; indeed, the approximation $H^{(L)}$ to the off-diagonal low-rank blocks at level $L$ only updates the blocks $A_{j\pm1,j}^{(L)}$. We will approximate the blocks $\widehat{A}_{j,j}^{(L)}$ with factors $\widehat{H}_j$ which we place in the matrix

$$\widehat{H} = \text{block-diag}\Big(\widehat{H}_1 \quad \widehat{H}_2 \quad \widehat{H}_3 \quad \cdots \quad \widehat{H}_{2^L}\Big). \tag{3.20}$$

We now have the notation needed to fully describe Algorithm 3. In Sections 3.2.4 to 3.2.5 we describe in more detail the main conceptual pieces of the algorithm for the approximation problem and introduce some more notation which we will use in our analysis.

**Algorithm 3** Generalized Nyström Method Peeling algorithm for HODLR approximation

**procedure** GeneralizedNyströmPeeling($A, k, s_{\mathrm{R}}, t_{\mathrm{R}}, s_{\mathrm{L}}, t_{\mathrm{L}}$)
  Set $L = \lceil \log_2(n/k) \rceil$ ▷ Final level blocks of size at most $k$
  **for** $\ell = 1, 2, \ldots, L$ **do**
    Allocate and partition $H^{(\ell)}$ as in Equation (3.18) ▷ blocks of size $n/2^\ell \times n/2^\ell$
    $\xi^+, \xi^-, \Omega^+, \Omega^- \sim \text{RandPerfGaussian}(n, 2^\ell, s_{\mathrm{R}}, t_{\mathrm{R}})$ ▷ as in Definition 3.2.8
    $\zeta^+, \zeta^-, \Psi^+, \Psi^- \sim \text{RandPerfGaussian}(n, 2^\ell, s_{\mathrm{L}}, t_{\mathrm{L}})$ ▷ as in Definition 3.2.8
    Compute $A^{(\ell)}\Omega^{\pm}$ ▷ $2s_{\mathrm{R}}t_{\mathrm{R}}$ matvecs with $A$
    Compute $(\Psi^{\pm})^\top A^{(\ell)}$ ▷ $2s_{\mathrm{L}}t_{\mathrm{L}}$ matvecs with $A^\top$
    **for** $j = 1, 2, \ldots, 2^\ell$ **do**
      Set $\rho$ such that $\xi^{\pm}_{j,\rho} = 1$
      Set $\sigma$ such that $\zeta^{\mp}_{j\pm1,\sigma} = 1$
      $Q_j^{(\ell)} = \text{orth}(Y_j^{(\ell)})$ ▷ $Y_j^{(\ell)}$ is $(j \pm 1, \rho)$ block of $A^{(\ell)}\Omega^{\pm}$
      $X_j^{(\ell)} = \left((\Psi^{\mp}_{j\pm1})^\top Q_j^{(\ell)}\right)^\dagger Z_j^{(\ell)}$ ▷ $Z_j^{(\ell)}$ is $(\sigma, j)$ block of $(\Psi^{\pm})^\top A$
      $H_j^{(\ell)} = Q_j^{(\ell)}(X_j^{(\ell)})_k$ ▷ $H_j^{(\ell)}$ is $(j \pm 1, j)$-th block of $H^{(\ell)}$
    **end for**
  **end for**
  Allocate and partition $\widehat{H}$ as in Equation (3.20) ▷ blocks of size $n/2^L \times n/2^L$
  $\widehat{\zeta}^+, \widehat{\zeta}^-, \widehat{\Psi}^+, \widehat{\Psi}^- \sim \text{RandPerfGaussian}(n, 2^L, s_{\mathrm{L}}, t_{\mathrm{L}})$ ▷ as in Definition 3.2.8
  $\widehat{\zeta} = \widehat{\zeta}^+ + \widehat{\zeta}^-$, $\widehat{\Psi} = \widehat{\Psi}^+ + \widehat{\Psi}^-$
  Compute $\widehat{\Psi}^\top \widehat{A}^{(L)}$ ▷ $s_{\mathrm{L}}t_{\mathrm{L}}$ matvecs with $A^\top$
  **for** $j = 1, 2, \ldots, 2^L$ **do**
    Set $\sigma$ such that $\zeta_{j,\sigma} = 1$
    $\widehat{X}_j = \left((\Psi_j)^\top\right)^\dagger \widehat{Z}_j$ ▷ $\widehat{Z}_j$ is $(\sigma, j)$ block of $(\widehat{\Psi})^\top \widehat{A}^{(L)}$
    $\widehat{H}_j = \widehat{X}_j$ ▷ $\widehat{H}_j$ is $(j, j)$-th block of $\widehat{H}$
  **end for**
  **return** $\widetilde{A} = H^{(1)} + \cdots + H^{(L)} + \widehat{H}$
**end procedure**

**Range approximation**

We first describe how to obtain an approximate range $Q_j^{(\ell)}$ for each $A_{j\pm1,j}$ at level $\ell$. Towards this end, let

$$\xi^+, \xi^-, \Omega^+, \Omega^- \sim \text{RandPerfGaussian}(n, d, s_{\text{R}}, s_{\text{R}})$$

as defined in Definition 3.2.8. We will access $A$ via the sketches $A^{(\ell)}\Omega^+$ and $A^{(\ell)}\Omega^-$ which each can be computed using $s_{\text{R}}t_{\text{R}}$ matvecs with $A$.

Fix $j$ and let $\rho = \rho(j)$ be the (unique) index such that $\xi^{\pm}_{j,\rho} = 1$. We will try to recover a good approximation to the range of $A^{(\ell)}_{j\pm1,j}$ using the information from the $(j \pm 1, \rho)$ block of the sketch $A\Omega^{\pm}$. This is illustrated in Figure 3.5. In particular, we will use the sketch

$$Y_j^{(\ell)} := A^{(\ell)}_{j\pm1,:}\Omega^{\pm}_{:,\rho} = \sum_{i=1}^{d} A^{(\ell)}_{j\pm1,i}(\xi^{\pm}_{i,\rho}\Omega_i).$$

It will be useful to write

$$Y_j^{(\ell)} = A^{(\ell)}_{j\pm1,j}\Omega_j + E_j^{(\ell)}, \qquad E_j^{(\ell)} := \sum_{i\neq j} \xi^{\pm}_{i,\rho} A^{(\ell)}_{j\pm1,i}\Omega_i. \tag{3.21}$$

We will then set

$$Q_j^{(\ell)} = \text{orth}(Y_j^{(\ell)}) = \text{orth}(A^{(\ell)}_{j\pm1,j}\Omega_j + E_j^{(\ell)}), \tag{3.22}$$

which will still be an approximate top subspace of $A^{(\ell)}_{j\pm1,j}$, provided the noise term $E_j^{(\ell)}$ is not too large. In particular, note that for the recovery problem where $A$ is exactly HODLR, the noise term $E_j^{(\ell)}$ is zero. This is because in this setting $A^{(\ell)}_{j\pm1,i} = 0$ for $i \neq j \pm 1$ and $\xi^{\pm}_{j\pm1,\rho} = 0$ so that there is no contributions from the on-diagonal block $A^{(\ell)}_{j,j}$.

**Low-rank approximation from given subspace**

We now describe how to obtain the low-rank approximations $H_j^{(\ell)}$ to each $A_{j\pm1,j}^{(\ell)}$ at level $\ell$, given the approximate left subspaces $Q_j^{(\ell)}$ computed by the procedure described in Section 3.2.4.

$$\zeta^+, \zeta^-, \Psi^+, \Psi^- \sim \text{RandPerfGaussian}(n, d, s_{\text{L}}, t_{\text{L}})$$

as defined in Definition 3.2.8. We will access $A$ via the sketches $(A^{(\ell)})^\top \Psi^+$ and $(A^{(\ell)})^\top \Psi^-$ which each can be computed using $s_{\text{L}} t_{\text{L}}$ matvecs with $A^\top$.

Fix $j$ and let $\sigma = \sigma(j)$ be the (unique) index such that $\zeta^{\mp}_{j\pm1,\sigma} = 1$. We will try to recover a low-rank approximation of $A_{j\pm1,j}^{(\ell)}$ whose column space is $Q_j^{(\ell)}$ from the $(\sigma, j)$ block of the sketch $(\Psi^{\mp})^\top A^{(\ell)}$. This is illustrated in Figure 3.5. In particular, we will use the sketch

$$Z_j^{(\ell)} := (\Psi^{\mp}_{:,\sigma})^\top A^{(\ell)}_{:,j} = \sum_{i=1}^{d} (\zeta^{\mp}_{i,\sigma} \Psi_i)^\top A^{(\ell)}_{i,j}.$$

Similar to above, we therefore have

$$Z_j^{(\ell)} = (\Psi^{\mp}_{j\pm1})^\top A^{(\ell)}_{j\pm1,j} + F_j^{(\ell)}, \qquad F_j^{(\ell)} := \sum_{i \neq j\pm1} \zeta^{\mp}_{i,\sigma} (\Psi_i)^\top A^{(\ell)}_{i,j}. \tag{3.23}$$

Now, we obtain obtain the right factors by

$$X_j^{(\ell)} := \left((\Psi^{\mp}_{j\pm1})^\top Q_j^{(\ell)}\right)^\dagger Z_j^{(\ell)} = \operatorname*{argmin}_X \|(\Psi_{j\pm1})^\top Q_j^{(\ell)} X - Z_j^{(\ell)}\|_{\mathsf{F}}. \tag{3.24}$$

Finally, we obtain an approximation $H_j^{(\ell)} = Q_j^{(\ell)} (X_j^{(\ell)})_k$ to $A_{j\pm1,j}^{(\ell)}$.

If $F_j^{(\ell)}$ is small, then $Q_j^{(\ell)} X_j^{(\ell)}$ will be nearly the best approximation to $A_{j\pm1,j}^{(\ell)}$ with range equal to the column span of $Q_j^{(\ell)}$. Similar to before, if $A$ is exactly HODLR, the noise term $F_j^{(\ell)}$ is zero.

### 3.2.5 Recovering the diagonal blocks

The strategy we use to recover the diagonal blocks similar to the off-diagonal blocks. However, since the blocks have at most $k$ columns, there is no need to obtain the left subspace.[7]

Towards this end, let

$$\widehat{\zeta}^{+}, \widehat{\zeta}^{-}, \widehat{\Psi}^{+}, \widehat{\Psi}^{-} \sim \text{RandPerfGaussian}(n, d, s_{\text{L}}, t_{\text{L}})$$

as defined in Definition 3.2.8. Next, define

$$\widehat{\zeta} = \widehat{\zeta}^{+} + \widehat{\zeta}^{-}, \qquad \widehat{\Psi} = \widehat{\Psi}^{+} + \widehat{\Psi}^{-}$$

Fix $j$ and let $\sigma = \sigma(j)$ be the (unique) index such that $\xi_{j,\sigma} = 1$. We will try to recover a low-rank approximation of $\widehat{A}^{(L)}_{j,j}$ the $(\sigma, j)$ block of the sketch $(A^{(L+1)})^{\top}\widehat{\Psi}$. In particular, we will use the sketch

$$\widehat{Z}_j := (\widehat{\Psi}_{:,\sigma})^{\top} \widehat{A}^{(L)}_{:,j} = \sum_{i=1}^{d} \xi_{i,\sigma} (\Psi_i)^{\top} \widehat{A}^{(L)}_{i,j}$$

Similar to above, we therefore have

$$\widehat{Z}_j = (\Psi_j)^{\top} \widehat{A}^{(L)}_{j,j} + \widehat{F}_j, \qquad \widehat{F}_j := \sum_{i \neq j} \xi_{i,\sigma} (\Psi_i)^{\top} \widehat{A}^{(L)}_{i,j}. \tag{3.25}$$

We obtain the diagonal blocks by

$$\widehat{X}_j := \left((\Psi_j)^{\top}\right)^{\dagger} \widehat{Z}_j = \operatorname*{argmin}_{H} \|(\Psi_j)^{\top} H - \widehat{Z}_j\|_{\mathsf{F}}. \tag{3.26}$$

Finally, we obtain an approximation $\widehat{H}_j = \widehat{X}_j$ to $\widehat{A}^{(L)}_{j,j}$.

[7] This is also true for the off-diagonal blocks in levels where $n/2^{\ell} \leq s_{\text{R}}$. In such cases $Q_j^{(\ell)}$ will be (square) orthogonal, so it could be set to the identity a priori. We do not separate this case to simplify the notation in our analysis.

### 3.2.6 A randomized-SVD peeling algorithm

As discussed in Section 3.2.11, a simple RSVD-based peeling algorithm with truncation cannot perform well, as there is no way of controlling errors made in the projection step. However, by using a similar randomized perforation technique as described in Section 3.2.3, one can implement an RSVD-based algorithm which can solve Problem 3 for arbitrary $\Gamma > 1$. We now define the additional notation needed. The full algorithm is described in Algorithm 4.

At level $\ell$, the RSVD-based peeling algorithm will obtain left subspaces $Q_j^{(\ell)}$ as in Section 3.2.4. However, rather than solving a regression problem like Algorithm 3, the algorithm will attempt to directly compute $(Q_j^{(\ell)})^\top A_{j\pm1,j}^{(\ell)}$. To control the error, we first sample $\zeta^+, \zeta^- \sim \text{PerfCountSketch}(d, t_{\text{R}})$ and define sketching matrices

$$\Psi^+ = \xi^+ \bullet Q^{(\ell)}, \qquad \Psi^- = \xi^- \bullet Q^{(\ell)}, \qquad Q^{(\ell)} := \begin{bmatrix} Q_1^{(\ell)})^\top & \cdots & Q_d^{(\ell)})^\top \end{bmatrix}^\top$$

By setting $t_{\text{R}}$ large, the expected squared error for each block can be driven arbitrarily small.

As with Algorithm 3, at the final level, we do not need to sketch $A$. However, to limit the error when trying to perform the projections (onto the identity), we sample $\widehat{\xi} \sim \text{CountSketch}(d, t_{\text{R}})$ and use the sketching matrix

$$\widehat{\Psi} = \widehat{\xi} \bullet (1 \otimes I).$$

Here 1 is the all-ones vector and "$\otimes$" is denotes the Kronecker product so that $1 \otimes I$ is the stacked identity matrix.

**Algorithm 4** Randomized SVD Peeling algorithm for HODLR approximation

**procedure** RandomizedSVDPeeling($A, k, s_{\mathrm{L}}, t_{\mathrm{L}}, s_{\mathrm{R}}$)
  Set $L = \lceil \log_2(n/k) \rceil$ ▷ Final level blocks of size at most $k$
  **for** $\ell = 1, 2, \ldots, L$ **do**
    Allocate and partition $H^{(\ell)}$ as in Equation (3.18) ▷ blocks of size $n/2^\ell \times n/2^\ell$
    Sample $\xi^+, \xi^-, \Omega^+, \Omega^- \sim \mathrm{RandPerfGaussian}(n, 2^\ell, s_{\mathrm{R}}, t_{\mathrm{R}})$ ▷ as in Definition 3.2.8
    Compute $A^{(\ell)}\Omega^\pm$ ▷ $2s_{\mathrm{R}}s_{\mathrm{R}}$ matvecs with $A$
    **for** $j = 1, 2, \ldots, 2^\ell$ **do**
      $Q_j^{(\ell)} = \mathrm{orth}(Y_j^{(\ell)})$ ▷ $Y_j^{(\ell)}$ is $(j \pm 1, \rho)$ block of $A^{(\ell)}\Omega^\pm$
    **end for**
    Sample $\zeta^+, \zeta^- \sim \mathrm{PerfCountSketch}(d, t_{\mathrm{L}})$ ▷ as in Definition 3.2.7
    Set $\Psi^\pm = \xi^\pm \bullet Q^{(\ell)}$
    Compute $(\Psi^\pm)^\top A^{(\ell)}$ ▷ $2s_{\mathrm{L}}t_{\mathrm{L}}$ matvecs with $A^\top$
    **for** $j = 1, 2, \ldots, 2^\ell$ **do**
      Extract $X_j^{(\ell)}$ ▷ $X_j^{(\ell)}$ is $(j, \sigma)$ block of $(\Psi^\pm)^\top A$
      $H_j^{(\ell)} = Q_j^{(\ell)}(X_j^{(\ell)})_k$ ▷ $H_j^{(\ell)}$ is $(j \pm 1, j)$-th block of $H^{(\ell)}$
    **end for**
  **end for**
  Allocate and partition $\widehat{H}$ as in Equation (3.20) ▷ blocks of size $n/2^L \times n/2^L$
  Sample $\widehat{\zeta} \sim \mathrm{CountSketch}(2^\ell, t_{\mathrm{L}})$ ▷ as in Definition 3.2.6
  Set $\widehat{\Psi} = \widehat{\zeta} \bullet ([1, \ldots, 1]^\top \otimes I)$
  $\widehat{\Psi}^\top \widehat{A}^{(L)} = A^\top \widehat{\Psi} - (H^{(L)} + \cdots + H^{(1)})\Psi$ ▷ $s_{\mathrm{L}}t_{\mathrm{L}}$ matvecs with $A^\top$
  **for** $j = 1, 2, \ldots, 2^L$ **do**
    Extract $\widehat{X}_j$ ▷ $\widehat{X}_j$ is $(j, \sigma)$ block of $\Psi^\top \widehat{A}^{(L)}$
    $\widehat{H}_j = \widehat{X}_j$ ▷ $\widehat{H}_j$ is $(j, j)$-th block of $\widehat{H}$
  **end for**
  **return** $\widetilde{A} = H^{(1)} + \cdots + H^{(L)} + \widehat{H}$
**end procedure**

### 3.2.7 Analysis

We are now prepared to prove our main accuracy guarantee Theorem 3.2.13 for Algorithm 3. The simplified Theorem 3.2.1 immediately follows.

In Section 3.2.7 we first prove the perturbation bounds Theorems 3.2.3 and 3.2.4. Then, in Section 3.2.8 we use these bounds to prove our main approximation guarantees. Our analysis relies on applying Theorem 3.2.4 to each off-diagonal low-rank block at each level. Assuming we have obtained near-optimal low-rank approximations to the off-diagonal low-rank blocks at the previous levels, we will show that we can obtain a near-optimal approximation to the off-diagonal low-rank blocks at the current level.

### Perturbation bound for the RSVD

We begin with proving Theorem 3.2.3.

**Theorem 3.2.3.** *Let $B \in \mathbb{R}^{m_1 \times m_2}$, $\Omega \in \mathbb{R}^{m_2 \times s}$, $E_1 \in \mathbb{R}^{m_1 \times s}$, and $E_2 \in \mathbb{R}^{s \times m_2}$, and let $Q = \mathrm{orth}(B\Omega + E_1)$. Write $B$ in its singular value decomposition as*

$$B = U\Sigma V^\top = \begin{bmatrix} U_{\mathrm{top}} & U_{\mathrm{bot}} \end{bmatrix} \begin{bmatrix} \Sigma_{\mathrm{top}} & \\ & \Sigma_{\mathrm{bot}} \end{bmatrix} \begin{bmatrix} V_{\mathrm{top}}^\top \\ V_{\mathrm{bot}}^\top \end{bmatrix},$$

*where $U$ and $V$ are orthonormal and $\Sigma$ is diagonal, and the "top" blocks represent the top $k$ columns; i.e. $U_{\mathrm{top}} \in \mathbb{R}^{m_1 \times k}, V_{\mathrm{top}} \in \mathbb{R}^{m_2 \times k}$, and $\Sigma_{\mathrm{top}} \in \mathbb{R}^{k \times k}$. Define also*

$$\Omega_{\mathrm{top}} = V_{\mathrm{top}}^\top \Omega, \quad \Omega_{\mathrm{bot}} = V_{\mathrm{bot}}^\top \Omega.$$

*Then, assuming* $\mathrm{rank}(\Omega_{\mathrm{top}}) = k$,

$$\|B - Q(Q^\top B + E_2)_k\|_{\mathsf{F}} \leq \|E_1 \Omega_{\mathrm{top}}^\dagger\|_{\mathsf{F}} + 2\|E_2\|_{\mathsf{F}} + \left(\|\Sigma_{\mathrm{bot}}\|_{\mathsf{F}}^2 + \|\Sigma_{\mathrm{bot}} \Omega_{\mathrm{bot}} \Omega_{\mathrm{top}}^\dagger\|_{\mathsf{F}}^2\right)^{1/2}.$$

*Proof.* Consider the matrix $C := (I - QQ^\top)B + Q(Q^\top B + E_2)$. The best rank-$k$ Frobenius-norm approximation to $C$ with range contained in range$(Q)$ is $Q(Q^\top C)_k$ [79, Theorem 3.5]. Using this, the triangle inequality, and the relations $\|B - C\|_{\mathsf{F}} =$

$\|E_2\|_{\mathsf{F}}$ and $Q^\top C = Q^\top B + E_2$ we obtain

$$\begin{aligned}\|B - Q(Q^\top B + E_2)_k\|_{\mathsf{F}} &= \|B - Q(Q^\top C)_k\|_{\mathsf{F}} \\ &\leq \|B - C\|_{\mathsf{F}} + \|C - Q(Q^\top C)_k\|_{\mathsf{F}} \\ &\leq \|B - C\|_{\mathsf{F}} + \|C - Q(Q^\top B)_k\|_{\mathsf{F}} \\ &\leq 2\|B - C\|_{\mathsf{F}} + \|B - Q(Q^\top B)_k\|_{\mathsf{F}} \\ &= 2\|E_2\|_{\mathsf{F}} + \|B - Q(Q^\top B)_k\|_{\mathsf{F}}. \end{aligned} \tag{3.27}$$

Now let $M := \Omega_{\text{top}}^\dagger V_{\text{top}}^\top$. Note that $\text{rank}((B\Omega + E_1)M) \leq k$ and let $P$ denote the orthogonal projector onto $\text{range}((B\Omega + E_1)M)$. Therefore, using [79, Theorem 3.5] and the triangle inequality, we obtain

$$\begin{aligned}\|B - Q(Q^\top B)_k\|_{\mathsf{F}} &\leq \|(I - P)B\|_{\mathsf{F}} \\ &\leq \|(I - P)B\Omega M\|_{\mathsf{F}} + \|(I - P)B(I - \Omega M)\|_{\mathsf{F}} \\ &\leq \|(I - P)B\Omega M\|_{\mathsf{F}} + \|B(I - \Omega M)\|_{\mathsf{F}}. \end{aligned} \tag{3.28}$$

Now, by the arguments in [54, Chapter 5] we can bound each term (3.28). First note that have

$$(I - P)B\Omega M = (I - P)(B\Omega + E_1)M - (I - P)E_1 M = -(I - P)E_1 M.$$

Hence,

$$\|(I - P)B\Omega M\|_{\mathsf{F}} \leq \|E_1 M\|_{\mathsf{F}} = \|E_1 \Omega_{\text{top}}^\dagger\|_{\mathsf{F}}. \tag{3.29}$$

Furthermore, for the second term note that

$$B(I - \Omega M) = BV_{\text{bot}}V_{\text{bot}}^\top(I - \Omega M) = BV_{\text{bot}}V_{\text{bot}}^\top - BV_{\text{bot}}V_{\text{bot}}^\top \Omega M.$$

By Pythagoras theorem we have,

$$\begin{aligned}\|B(I - \Omega M)\|_{\mathsf{F}} &= \left(\|BV_{\text{bot}}V_{\text{bot}}^\top\|_{\mathsf{F}}^2 + \|BV_{\text{bot}}V_{\text{bot}}^\top \Omega M\|_{\mathsf{F}}^2\right)^{1/2} \\ &= \left(\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 + \|\Sigma_{\text{bot}}\Omega_{\text{bot}}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2\right)^{1/2} \end{aligned} \tag{3.30}$$

Combining (3.27) to (3.30) yields the desired inequality. □

### Probabilistic bounds

We will repeatedly make use about the following well-known fact about the Frobenius norm of Gaussian matrices and their pseudoinverses, proofs of which can be found throughout the literature; see for instance [85, Proposition A.5].

**Theorem 3.2.9.** *Let $X$ be $u \times v$, and $G \in \mathbb{R}^{v\times q}$ and $H \in \mathbb{R}^{p\times q}$ be independent with entries independently drawn from $\mathcal{N}(0,1)$. Then, if $q > p+1$,*

$$\mathbb{E}\Big[\|XGH^\dagger\|_{\mathsf{F}}^2\Big] = \|X\|_{\mathsf{F}}^2\,\mathbb{E}\Big[\|H^\dagger\|_{\mathsf{F}}^2\Big] = \frac{p}{q-p-1}\|X\|_{\mathsf{F}}^2.$$

We proceed with proving the following lemma, which relates to the error of the projection step of the Generalized Nyström Method.

**Lemma 3.2.10.** *Let $B \in \mathbb{R}^{m_1\times m_2}$, $Q \in \mathbb{R}^{m_1\times s_{\mathrm{R}}}$ with orthonormal columns, and $N \in \mathbb{R}^{q\times m_2}$ be fixed matrices and $\Psi \in \mathbb{R}^{m_1\times s_{\mathrm{L}}}$ and $\widetilde{\Psi} \in \mathbb{R}^{q\times s_{\mathrm{L}}}$ have entries drawn independently from $\mathcal{N}(0,1)$. Then, provided $s_{\mathrm{L}} > s_{\mathrm{R}}+1$,*

$$\mathbb{E}\Big[\|Q^\top B - (\Psi^\top Q)^\dagger(\Psi^\top B + \widetilde{\Psi}^\top N)\|_{\mathsf{F}}^2\Big] = \frac{s_{\mathrm{R}}}{s_{\mathrm{L}}-s_{\mathrm{R}}-1}\Big(\|B - QQ^\top B\|_{\mathsf{F}}^2 + \|N\|_{\mathsf{F}}^2\Big).$$

*Proof.* Extend $Q$ to an orthogonal matrix $[Q\ \widetilde{Q}]$. Write $\Psi_1 = \Psi^\top Q$ and $\Psi_2 = \Psi^\top\widetilde{Q}$. Since $Q$ and $\widetilde{Q}$ are orthogonal, $\Psi_1 \in \mathbb{R}^{s_{\mathrm{L}}\times s_{\mathrm{R}}}$ and $\Psi_2 \in \mathbb{R}^{s_{\mathrm{L}}\times m_1 - s_{\mathrm{R}}}$ are independent with entries drawn independently from $\mathcal{N}(0,1)$.

Since $[Q\ \widetilde{Q}]$ is orthogonal, $QQ^\top + \widetilde{Q}\widetilde{Q}^\top$ is the identity, and

$$\Psi^\top B = \Psi^\top(QQ^\top + \widetilde{Q}\widetilde{Q}^\top)B = \Psi_1 Q^\top B + \Psi_2\widetilde{Q}^\top B.$$

Therefore, recalling our definition of $\Psi_1$,

$$\Psi_1^\dagger(\Psi^\top B + \widetilde{\Psi}^\top N) = \Psi_1^\dagger\Psi_1 Q^\top B + \Psi_1^\dagger\Psi_2\widetilde{Q}^\top B + \Psi_1^\dagger\widetilde{\Psi}^\top N.$$

Since $s_{\mathrm{L}} \geq s_{\mathrm{R}}$, with probability 1, $\Psi_1^\dagger \Psi_1 = I$. Thus, $Q^\top B - (\Psi^\top B + \widetilde{\Psi}^\top N) = -\Psi_1^\dagger \Psi_2 \widetilde{Q}^\top \mathbf{B} - \mathbf{\Psi}_1^\dagger \widetilde{\Psi}^\top N$.

Next, as in [142, Lemma A.2(i)], for deterministic matrices $X, Y, Z$ and a Gaussian matrix $\Psi_2$,

$$\mathbb{E}\Big[\|X\Psi_2 Y + Z\|_{\mathsf{F}}^2\Big] = \mathbb{E}\Big[\|X\|_{\mathsf{F}}^2 \|Y\|_{\mathsf{F}}^2 + \|Z\|_{\mathsf{F}}^2\Big]. \tag{3.31}$$

Therefore, using that $\Psi_1, \Psi_2$ and $\widehat{\Psi}$ are independent and applying Equation (3.31) and Theorem 3.2.9 we obtain

$$\begin{aligned}
\mathbb{E}\Big[\|\Psi_1^\dagger(\Psi^\top B + \widetilde{\Psi}^\top N) - Q^\top B\|_{\mathsf{F}}^2\Big] &= \mathbb{E}\Big[\|\Psi_1^\dagger \Psi_2 \widetilde{Q}^\top B + \Psi_1^\dagger \widetilde{\Psi}^\top N\|_{\mathsf{F}}^2\Big] \\
&= \mathbb{E}\Big[\mathbb{E}\Big[\|\Psi_1^\dagger \Psi_2 \widetilde{Q}^\top B + \Psi_1^\dagger \widetilde{\Psi}^\top N\|_{\mathsf{F}}^2 \Big| \mathbf{\Psi}_1, \widetilde{\Psi}\Big]\Big] \\
&= \mathbb{E}\Big[\|\Psi_1^\dagger\|_{\mathsf{F}}^2 \|\widetilde{Q}^\top B\|_{\mathsf{F}}^2 + \|\Psi_1^\dagger \widetilde{\Psi}^\top N\|_{\mathsf{F}}^2\Big]. \\
&= \frac{s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1}\|\widetilde{Q}^\top B\|_{\mathsf{F}}^2 + \frac{s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1}\|N\|_{\mathsf{F}}^2.
\end{aligned}$$

Noting that $\|\widetilde{Q}^\top B\|_{\mathsf{F}}^2 = \|B - QQ^\top B\|_{\mathsf{F}}^2$ yields the desired result. □

Finally, we use Theorem 3.2.3 and lemma 3.2.10 to prove Theorem 3.2.4.

**Theorem 3.2.4.** *Let $B \in \mathbb{R}^{m_1 \times m_2}$, $M \in \mathbb{R}^{m_1 \times p}$, and $N \in \mathbb{R}^{q \times m_2}$ be fixed matrices, and let $\Omega \in \mathbb{R}^{m_2 \times s_{\mathrm{R}}}$, $\widetilde{\Omega} \in \mathbb{R}^{p \times s_{\mathrm{R}}}$, $\Psi \in \mathbb{R}^{m_1 \times s_{\mathrm{L}}}$, and $\widetilde{\Psi} \in \mathbb{R}^{q \times s_{\mathrm{L}}}$ be independent matrices with entries drawn independently from $\mathcal{N}(0,1)$. Define*

$$Q := \mathrm{orth}(B\Omega + M\widetilde{\Omega}), \qquad X := (\Psi^\top Q)^\dagger (\Psi^\top B + \widetilde{\Psi}^\top N).$$

*Then, provided $s_{\mathrm{R}} > 2k + 1$ and $s_{\mathrm{L}} > 2s_{\mathrm{R}} + 1$*

$$\mathbb{E}\Big[\|B - QX_k\|_{\mathsf{F}}^2\Big] \leq E_1 + E_2 + 2\sqrt{E_1 E_2},$$

*where*

$$\begin{aligned}
E_1 &:= \left(1 + \frac{k}{s_{\mathrm{R}} - k - 1}\right)\|B - B_k\|_{\mathsf{F}}^2 \\
E_2 &:= \frac{18k}{s_{\mathrm{R}} - k - 1}\|M\|_{\mathsf{F}}^2 + \frac{8s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1}\|N\|_{\mathsf{F}}^2 + \frac{32 s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1}\|B - B_k\|_{\mathsf{F}}^2.
\end{aligned}$$

*Proof.* Let $B$ have SVD as partitioned in Theorem 3.2.3 and define $\Omega_{\text{top}} = V_{\text{top}}^\top \Omega$ and $\Omega_{\text{bot}} = V_{\text{bot}}^\top \Omega$. Note that, by the unitary invariance of Gaussian random matrices, $\Omega_{\text{top}} \in \mathbb{R}^{k \times s_{\text{R}}}$ and $\Omega_{\text{bot}} \in \mathbb{R}^{m_2 - k \times s_{\text{R}}}$ have independent entries drawn from $\mathcal{N}(0,1)$ and are independent of one another and of $\widetilde{\Omega}$. Note that $\Omega_{\text{top}}$ has rank $k$ with probability one, and Theorem 3.2.3 asserts

$$\|B - QX_k\|_{\mathsf{F}} \leq \underbrace{\|M\widetilde{\Omega}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}}_{A} + \underbrace{2\|Q^\top B - X\|_{\mathsf{F}}}_{B} + \underbrace{\left(\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 + \|\Sigma_{\text{bot}}\Omega_{\text{bot}}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2\right)^{1/2}}_{C}. \tag{3.32}$$

We will set $E_1 = \mathbb{E}[C^2]$ and $E_2$ will be an upper bound for $\mathbb{E}[(A+B)^2]$. Since $\mathbb{E}[(A+B)^2] \leq 2\,\mathbb{E}[A^2 + B^2]$ we may set $E_2$ to be an upper bound for $2\,\mathbb{E}[A^2 + B^2]$. Indeed, by the linearity of expectation, Cauchy–Schwarz, and the fact that $(x+y)^2 \leq 2(x^2+y^2)$,

$$\begin{aligned}
\mathbb{E}[(A+B+C)^2] &= \mathbb{E}[(A+B)^2] + \mathbb{E}[C^2] + 2\,\mathbb{E}[(A+B)C] \\
&\leq \mathbb{E}[(A+B)^2] + \mathbb{E}[C^2] + 2\sqrt{\mathbb{E}[(A+B)^2]\,\mathbb{E}[C^2]} \\
&\leq E_2 + E_1 + 2\sqrt{E_2 E_1}.
\end{aligned}$$

It now remains to compute $\mathbb{E}[A^2]$, $\mathbb{E}[B^2]$, and $\mathbb{E}[C^2]$.

First, using Theorem 3.2.9 we have that

$$\mathbb{E}[A^2] = \mathbb{E}\left[\|M\widetilde{\Omega}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2\right] = \frac{k}{s_{\text{R}} - k - 1}\|M\|_{\mathsf{F}}^2; \tag{3.33}$$

$$\mathbb{E}[C^2] = \left(1 + \frac{k}{s_{\text{R}} - 1}\right)\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 = \left(1 + \frac{k}{s_{\text{R}} - 1}\right)\|B - B_k\|_{\mathsf{F}}^2. \tag{3.34}$$

(3.34) shows that $\mathbb{E}[C^2] = E_1$, as required. We will now proceed with showing that $E_2$ is an upper bound for $2\mathbb{E}[A^2 + B^2]$.

Since $QB$ is the best Frobenius-norm approximation to $B$ with range contained in range$(Q)$ [79, Lemma 2.2], using and Theorem 3.2.3 with the inequality $(x+$

$y)^2 \leq 2x^2 + 2y^2$ gives

$$\begin{aligned}\|B - QQ^\top B\|_{\mathsf{F}}^2 &\leq \|B - Q(Q^\top B)_k\|_{\mathsf{F}}^2 \\ &\leq 2\|M\Omega\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2 + 2\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 + 2\|\Sigma_{\text{bot}}\Omega_{\text{bot}}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2. \qquad (3.35)\end{aligned}$$

Applying Lemma 3.2.10, (3.35) and Theorem 3.2.9 yields

$$\begin{aligned}\mathbb{E}[B^2] &= \frac{4s_{\text{R}}}{s_{\text{L}} - s_{\text{R}} - 1}\Big(\mathbb{E}\Big[\|B - QQ^\top B\|_{\mathsf{F}}^2\Big] + \|N\|_{\mathsf{F}}^2\Big) \\ &\leq \frac{4s_{\text{R}}}{s_{\text{L}} - s_{\text{R}} - 1}\left(2\,\mathbb{E}\Big[\|M\widetilde{\Omega}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2\Big] + 2\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 + 2\,\mathbb{E}\Big[\|\Sigma_{\text{bot}}\Omega_{\text{bot}}\Omega_{\text{top}}^\dagger\|_{\mathsf{F}}^2\Big] + \|N\|_{\mathsf{F}}^2\right) \\ &= \frac{4s_{\text{R}}}{s_{\text{L}} - s_{\text{R}} - 1}\left(\frac{2k}{s_{\text{R}} - k - 1}\|M\|_{\mathsf{F}}^2 + 2\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 + \frac{2k}{s_{\text{R}} - k - 1}\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 + \|N\|_{\mathsf{F}}^2\right).\end{aligned}$$

Using our choices of $k, s_{\text{R}}$, and $s_{\text{L}}$ ensures $\frac{k}{s_{\text{R}}-k-1} \leq 1$ and $\frac{s_{\text{R}}}{s_{\text{L}}-s_{\text{R}}-1} \leq 1$. Hence,

$$\mathbb{E}[B^2] \leq \frac{8k}{s_{\text{R}} - k - 1}\|M\|_{\mathsf{F}}^2 + \frac{16s_{\text{R}}}{s_{\text{L}} - s_{\text{R}} - 1}\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 + \frac{4s_{\text{R}}}{s_{\text{L}} - s_{\text{R}} - 1}\|N\|_{\mathsf{F}}^2. \qquad (3.36)$$

Combining (3.33) and (3.36) and noting $\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 = \|B - B_k\|_{\mathsf{F}}^2$ yields

$$2\,\mathbb{E}[A^2 + B^2] \leq \frac{18k}{s_{\text{R}} - k - 1}\|M\|_{\mathsf{F}}^2 + \frac{32s_{\text{R}}}{s_{\text{L}} - s_{\text{R}} - 1}\|\Sigma_{\text{bot}}\|_{\mathsf{F}}^2 + \frac{8s_{\text{R}}}{s_{\text{L}} - s_{\text{R}} - 1}\|N\|_{\mathsf{F}}^2 := E_2,$$

as required. □

### 3.2.8 Analysis of Algorithm 3

We are now prepared to analyze Algorithm 3.

**Definition 3.2.11.** *For each $\ell = 1, 2, \ldots, L$, let $\mathcal{F}_\ell$ be the sigma algebra representing the information known to the algorithm at the start of the $\ell$-th step.*

We begin by proving the key approximation guarantee for a off-diagonal low-rank block at level $\ell$. This shows that, even in the presence of noise, we can obtain a near-optimal low-rank approximation to each of the off-diagonal blocks (and the

on-diagonal blocks at the final level $\ell = L$). This can be viewed as a version of Theorem 3.2.4 adapted to the errors appearing within the peeling algorithm.

**Lemma 3.2.12.** *Fix a rank parameter $k$ and let $\beta \in (0,1)$. Suppose $t_{\mathrm{L}} \geq 1$ and $s_{\mathrm{R}}$, $t_{\mathrm{L}}$, and $s_{\mathrm{R}}$ are such that*

$$\frac{k}{s_{\mathrm{R}} - k - 1} \leq \frac{\beta}{20}, \quad \frac{k}{s_{\mathrm{R}} - k - 1} \cdot \frac{1}{t_{\mathrm{R}}} \leq \frac{\beta^2}{24^2}, \quad \frac{s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1} \leq \frac{\beta^2}{24^2}.$$

*Then for each $\ell = 1, 2, \ldots, L$ the off-diagonal low-rank factors obtained by Algorithm 3 are nearly optimal in the sense that*

$$\mathbb{E}\Big[\|A^{(\ell)}_{j\pm1,j} - Q^{(\ell)}_j (X^{(\ell)}_j)_k\|^2_{\mathsf{F}} \Big| \mathcal{F}_\ell\Big] \leq \|A^{(\ell)}_{j\pm1,j} - (A^{(\ell)}_{j\pm1,j})_k\|^2_{\mathsf{F}} + \beta E^{(\ell)}_j.$$

*where*

$$E^{(\ell)}_j := \max\left\{ \|A^{(\ell)}_{j\pm1,j} - (A^{(\ell)}_{j\pm1,j})_k\|^2_{\mathsf{F}}, \sum_{\substack{i=1 \\ i\neq j, j\pm1}}^{2^\ell} \frac{1}{2}\|A^{(\ell)}_{i,j}\|^2_{\mathsf{F}}, \sum_{\substack{i=1 \\ i\neq j, j\pm1}}^{2^\ell} \frac{1}{2}\|A^{(\ell)}_{j\pm1,i}\|^2_{\mathsf{F}} \right\}.$$

*Moreover, at the final level $\ell = L$, the on-diagonal factors obtained by Algorithm 3 are nearly exact in the sense that*

$$\mathbb{E}\Big[\|\widehat{A}^{(L)}_{j,j} - \widehat{X}_j\|^2_{\mathsf{F}} \Big| \mathcal{F}_{L+1}\Big] \leq \beta \sum_{\substack{i=1 \\ i\neq j}}^{2^L} \|\widehat{A}^{(L)}_{i,j}\|^2_{\mathsf{F}}.$$

*Proof.* We begin with the first result. Let $d = 2^\ell$ and let $\Omega^\pm$ and $\Psi^\pm$ be the sketches used at level $\ell$ of Algorithm 3. Fix $j$ and recall that $\rho$ and $\sigma$ are the unique indices so that $\xi^\pm_{j,\rho} = 1$ and $\zeta^\mp_{j\pm1,\sigma} = 1$. Define

$$M = \begin{bmatrix} \xi^\pm_{1,\rho} A^{(\ell)}_{j\pm1,1} & \cdots & \xi^\pm_{j-1,\rho} A^{(\ell)}_{j\pm1,j-1} & \xi^\pm_{j+1,\rho} A^{(\ell)}_{j\pm1,j+1} & \cdots & \xi^\pm_{d,\rho} A^{(\ell)}_{j\pm1,d}, \end{bmatrix}$$
$$N = \begin{bmatrix} \zeta^\mp_{1,\sigma} A^{(\ell)}_{1,j} & \cdots & \zeta^\mp_{j\pm1-1,\sigma} A^{(\ell)}_{j\pm1-1,j} & \zeta^\mp_{j\pm1+1,\sigma} A^{(\ell)}_{j\pm1+1,j} & \cdots & \zeta^\mp_{d,\sigma} A^{(\ell)}_{d,j}, \end{bmatrix}$$

and the random Gaussian matrices

$$\widetilde{\Omega} = \begin{bmatrix} (\Omega_1)^\top & \cdots & (\Omega_{j-1})^\top & (\Omega_{j+1})^\top & \cdots & (\Omega_d)^\top \end{bmatrix}^\top$$
$$\widetilde{\Psi} = \begin{bmatrix} (\Psi_1)^\top & \cdots & (\Psi_{j\pm1-1})^\top & (\Psi_{j\pm1+1})^\top & \cdots & (\Psi_d)^\top \end{bmatrix}^\top.$$

Then we can write

$$E_j^{(\ell)} = M\widetilde{\Omega}, \qquad F_j^{(\ell)} = \widetilde{\Psi}^\top N,$$

where $E_j^{(\ell)}$ and $F_j^{(\ell)}$ are are the additive perturbation terms as defined in Equations (3.21) and (3.23). Furthermore, note that $\widetilde{\Omega}$ and $\widetilde{\Psi}$ are independent of $\Omega_j$ and $\Psi_{j\pm1}$. Recall that $Q_j^{(\ell)}$ is an orthonormal basis for range$(A_{j\pm1,j}\Omega_j + E_j^{(\ell)})$ and $X_j^{(\ell)} = (\Psi_{j\pm1}^\top Q_j^{(\ell)})^\dagger(\Psi_j^\top A_{j\pm1,j} + \widetilde{\Psi}^\top N)$.

The specified conditions on $k$, $s_{\mathrm{R}}$, and $s_{\mathrm{L}}$ ensure that $s_{\mathrm{L}} \geq 2s_{\mathrm{R}} + 1$ and $s_{\mathrm{R}} \geq 2k + 1$. Therefore, Theorem 3.2.4 yields

$$\mathbb{E}\Big[\|A_{j\pm1,j}^{(\ell)} - Q_j^{(\ell)}(X_j^{(\ell)})_k\|_{\mathsf{F}}^2 \Big| \mathcal{F}_\ell, \xi, \zeta\Big] \leq E_1 + E_2 + 2\sqrt{E_1 E_2}, \tag{3.37}$$

where

$$E_1 := \left(1 + \frac{k}{s_{\mathrm{R}} - k - 1}\right) \|A_{j\pm1,j}^{(\ell)} - (A_{j\pm1,j}^{(\ell)})_k\|_{\mathsf{F}}^2,$$
$$E_2 := \frac{18k}{s_{\mathrm{R}} - k - 1}\|M\|_{\mathsf{F}}^2 + \frac{8s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1}\|N\|_{\mathsf{F}}^2 + \frac{32s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1}\|A_{j\pm1,j}^{(\ell)} - (A_{j\pm1,j}^{(\ell)})_k\|_{\mathsf{F}}^2.$$

By our choice of $\xi^\pm$, the $\xi_{i,\rho}^\pm$ are independent for different $i$ and $\mathbb{E}[\xi_{i,\rho}^\pm]$ is zero or $1/t_{\mathrm{R}}$ depending on the parity of $\rho$. Thus, $\mathbb{E}[\xi_{i,\rho}^\pm] \leq 1/t_{\mathrm{R}}$. Furthermore, note that $\mathbb{E}[\xi_{j\pm1,\rho}^\pm] = 0$ by construction. Thus,

$$\mathbb{E}\Big[\|M\|_{\mathsf{F}}^2\Big] = \sum_{i\neq j} \mathbb{E}[\xi_{i,\rho}^\pm]\|A_{j\pm1,i}^{(\ell)}\|_{\mathsf{F}}^2 \leq \sum_{\substack{i=1 \\ i\neq j, j\pm1}}^{d} \|A_{j\pm1,i}^{(\ell)}\|_{\mathsf{F}}^2.$$

By a similar argument,

$$\mathbb{E}\Big[\|N\|_{\mathsf{F}}^2\Big] = \sum_{i\neq j\pm1} \mathbb{E}[\zeta_{i,\sigma}^\mp]\|A_{i,j}^{(\ell)}\|_{\mathsf{F}}^2 \leq \frac{1}{t_{\mathrm{L}}} \sum_{\substack{i=1 \\ i\neq j, j\pm1}}^{d} \|A_{i,j}^{(\ell)}\|_{\mathsf{F}}^2.$$

Hence, by the law of total expectation, the Cauchy-Schwarz inequality, and (3.37) we have

$$\mathbb{E}\Big[\|A_{j\pm1,j}^{(\ell)} - Q_j^{(\ell)}(X_j^{(\ell)})_k\|_{\mathsf{F}}^2 \Big| \mathcal{F}_\ell\Big] \leq E_1 + E_2' + 2\sqrt{E_1 E_2'}, \tag{3.38}$$

where

$$E_2' := \frac{18k}{s_{\mathrm{R}} - k - 1} \cdot \frac{1}{t_{\mathrm{R}}} \sum_{\substack{i=1 \\ i \neq j, j\pm 1}}^{d} \|A_{j\pm 1,i}^{(\ell)}\|_{\mathsf{F}}^2 + \frac{8s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1} \cdot \frac{1}{t_{\mathrm{L}}} \sum_{\substack{i=1 \\ i \neq j, j\pm 1}}^{d} \|A_{i,j}^{(\ell)}\|_{\mathsf{F}}^2 + \frac{32s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1} \|A_{j\pm 1,j}^{(\ell)} - (A_{j\pm 1,j}^{(\ell)})_k\|_{\mathsf{F}}^2.$$

Using that

$$\max\left\{ \|A_{j\pm 1,j}^{(\ell)} - (A_{j\pm 1,j}^{(\ell)})_k\|_{\mathsf{F}}^2, \frac{1}{2} \sum_{\substack{i=1 \\ i \neq j, j\pm 1}}^{d} \|A_{j\pm 1,i}^{(\ell)}\|_{\mathsf{F}}^2, \frac{1}{2} \sum_{\substack{i=1 \\ i \neq j, j\pm 1}}^{d} \|A_{i,j}^{(\ell)}\|_{\mathsf{F}}^2 \right\} \leq E_j^{(\ell)}$$

and our assumptions on $t_{\mathrm{L}}, t_{\mathrm{R}}, s_{\mathrm{L}}$, and $s_{\mathrm{R}}$ allow us to bound

$$E_1 + E_2' + 2\sqrt{E_1 E_2'} \leq \|A_{j\pm 1,j}^{(\ell)} - (A_{j\pm 1,j}^{(\ell)})_k\|_{\mathsf{F}}^2 + \beta E_j^{(\ell)}. \tag{3.39}$$

Combining (3.39) with (3.38) yields the first inequality.

The proof of the second result is similar to the analysis of the Generalized Nyström method Theorem 3.2.4. First define

$$\widehat{N} = \begin{bmatrix} \widehat{\zeta}_{1,\rho} \widehat{A}_{1,j}^{(L)} & \cdots & \widehat{\zeta}_{j-1,\rho} \widehat{A}_{j-1,j}^{(L)} & \widehat{\zeta}_{j+1,\rho} \widehat{A}_{j+1,j}^{(L)} & \cdots & \widehat{\zeta}_{2^L,\rho} \widehat{A}_{2^L,j}^{(L)} \end{bmatrix}.$$

and the random Gaussian matrix

$$\widetilde{\Psi} = \begin{bmatrix} (\widehat{\Psi}_1)^\top & \cdots & (\widehat{\Psi}_{j-1})^\top & (\widehat{\Psi}_{j+1})^\top & \cdots & (\widehat{\Psi}_{2^L})^\top \end{bmatrix}^\top.$$

Similar to above we have

$$\mathbb{E}\big[\|\widehat{N}\|_{\mathsf{F}}^2\big] \leq \frac{1}{t_{\mathrm{L}}} \sum_{\substack{i=1 \\ i \neq j}}^{2^L} \|\widehat{A}_{i,j}^{(L)}\|_{\mathsf{F}}^2.$$

Since $s_{\mathrm{L}} \geq s_{\mathrm{R}} > 2k+1$ and $k \geq n/2^L$ we can apply Theorem 3.2.9 to obtain

$$\begin{aligned}
\mathbb{E}\Big[\|\widehat{A}_{j,j}^{(L)} - \widehat{H}_j\|_{\mathsf{F}}^2 \Big| \mathcal{F}_{L+1}\Big] &= \mathbb{E}\Big[\|(\widehat{\Psi}_j^{\top})^{\dagger}\widetilde{\Psi}^{\top}\widehat{N}\|_{\mathsf{F}}^2 \Big| \mathcal{F}_{L+1}\Big] \\
&= \frac{n/2^L}{s_{\mathrm{L}} - n/2^L - 1}\mathbb{E}\big[\|\widehat{N}\|_{\mathsf{F}}^2\big] \\
&\leq \frac{k}{s_{\mathrm{R}} - k - 1}\frac{1}{t_{\mathrm{L}}}\sum_{\substack{i=1 \\ i\neq j}}^{2^L}\|\widehat{A}_{i,j}^{(L)}\|_{\mathsf{F}}^2 \\
&\leq \frac{\beta}{20}\sum_{\substack{i=1 \\ i\neq j}}^{2^L}\|\widehat{A}_{i,j}^{(L)}\|_{\mathsf{F}}^2 \leq \beta\sum_{\substack{i=1 \\ i\neq j}}^{2^L}\|\widehat{A}_{i,j}^{(L)}\|_{\mathsf{F}}^2,
\end{aligned}$$

as required. □

Finally, we prove our main theorem.

**Theorem 3.2.13.** *Fix a rank parameter $k$ and let $\beta \in (0,1)$. Suppose $t_{\mathrm{L}} \geq 1$ and $s_{\mathrm{R}}$, $t_{\mathrm{L}}$, and $s_{\mathrm{R}}$ are such that*

$$\frac{k}{s_{\mathrm{R}} - k - 1} \leq \frac{\beta}{20}, \quad \frac{k}{s_{\mathrm{R}} - k - 1}\cdot\frac{1}{t_{\mathrm{R}}} \leq \frac{\beta^2}{24^2}, \quad \frac{s_{\mathrm{R}}}{s_{\mathrm{L}} - s_{\mathrm{R}} - 1} \leq \frac{\beta^2}{24^2}.$$

*Let $L = \lceil \log_2(n/k) \rceil$. Then, using $2Ls_{\mathrm{R}}t_{\mathrm{R}}$ products with $A$ and $(2L+1)s_{\mathrm{L}}t_{\mathrm{L}}$ products with $A^{\top}$, Algorithm 3 outputs a* HODLR($k$) *matrix $\widetilde{A}$ such that*

$$\mathbb{E}\Big[\|A - \widetilde{A}\|_{\mathsf{F}}\Big] \leq (1+\beta)^{L+1}\cdot\min_{H\in\mathrm{HODLR}(k)}\|A - H\|_{\mathsf{F}}.$$

Before we prove Theorem 3.2.13, we make several remarks. First, observe that Theorem 3.2.1 is a special case of Theorem 3.2.13 with parameters

$$s_{\mathrm{R}} = O(k/\beta), \quad s_{\mathrm{R}} = O(1/\beta), \quad s_{\mathrm{L}} = O(s_{\mathrm{R}}/\beta^2) = O(k/\beta^3), \quad t_{\mathrm{L}} = 1. \tag{3.40}$$

Another interesting parameter setting is

$$s_{\mathrm{R}} = O(k/\beta^2), \quad s_{\mathrm{R}} = 1, \quad s_{\mathrm{L}} = O(k/\beta^4), \quad t_{\mathrm{L}} = 1, \tag{3.41}$$

which corresponds to an implementation of the standard peeling algorithm with the Generalized Nyström Method (i.e. does not use randomly perforated sketching).

*Proof of Theorem 3.2.13.* First, note that at each of the $L$ levels we use $2s_{\mathrm{R}}t_{\mathrm{R}}$ matrix-vector products with $A$ and $2s_{\mathrm{L}}t_{\mathrm{L}}$ matrix-vector products with $A^{\top}$. When recovering the diagonal we use an additional $s_{\mathrm{L}}t_{\mathrm{L}}$ products with $A^{\top}$. This results in the specified cost.

We now analyze the error. For each $\ell = 1, 2, \ldots, L$ partition $A^{(\ell)}$ as in Equation (3.16) and $H^{(\ell)}$ as in Equation (3.18). Define,

$$\mathsf{opt}(\ell) := \left( \sum_{j=1}^{2^\ell} \|A^{(\ell)}_{j\pm1,j} - (A^{(\ell)}_{j\pm1,j})_k\|_{\mathsf{F}}^2 \right)^{1/2}.$$

By the definition of HODLR matrices the on-diagonal blocks at level $L$ are of size at most $k$. Therefore, since the different levels are disjoint,

$$\min_{H \in \mathrm{HODLR}(k)} \|A - H\|_{\mathsf{F}}^2 = \mathsf{opt}(1)^2 + \cdots + \mathsf{opt}(L)^2. \tag{3.42}$$

Next, define the error incurred in the off-diagonal low-rank blocks at level $\ell$,

$$\mathsf{err}(\ell) := \left( \sum_{j=1}^{2^\ell} \|A^{(\ell)}_{j\pm1,j} - H^{(\ell)}_j\|_{\mathsf{F}}^2 \right)^{1/2}.$$

Define also the error of the on-diagonal blocks at the final level,

$$\widehat{\mathsf{err}} := \left( \sum_{j=1}^{2^L} \|A^{(L)}_{j,j} - \widehat{H}_j\|_{\mathsf{F}}^2 \right)^{1/2}.$$

Since the approximations at each level are disjoint (see Equation (3.15)) we have

$$\|A - \widetilde{A}\|_{\mathsf{F}}^2 = \mathsf{err}(1)^2 + \cdots + \mathsf{err}(L)^2 + \widehat{\mathsf{err}}^2. \tag{3.43}$$

We claim that it suffices to prove that for each $\ell = 1, 2, \ldots, L$,

$$\mathbb{E}\big[\mathsf{err}(\ell)^2\big] \leq \mathsf{opt}(\ell)^2 + \beta \cdot (1+\beta)^{\ell-1} \cdot \big(\mathsf{opt}(1)^2 + \cdots + \mathsf{opt}(\ell)^2\big). \tag{3.44}$$

and that at the final level

$$\mathbb{E}\big[\widehat{\mathsf{err}}^2\big] \leq \beta \cdot (1+\beta)^{L} \cdot \big(\mathsf{opt}(1)^2 + \cdots + \mathsf{opt}(L)^2\big). \tag{3.45}$$

Indeed, since $\mathsf{err}(\ell')^2 \geq 0$, together Equations (3.44) and (3.45) imply that

$$\begin{aligned}
&\mathbb{E}\big[\mathsf{err}(1)^2 + \cdots + \mathsf{err}(L)^2 + \widehat{\mathsf{err}}^2\big] \\
&\qquad \leq \Big(1 + \beta \cdot \big(1 + (1+\beta) + \cdots + (1+\beta)^L\big)\Big) \cdot \Big(\mathsf{opt}(1)^2 + \cdots + \mathsf{opt}(L)^2\Big) \\
&\qquad = (1+\beta)^{L+1} \cdot \Big(\mathsf{opt}(1)^2 + \cdots + \mathsf{opt}(L)^2\Big).
\end{aligned}$$

In light of Equations (3.42) and (3.43), this is the desired result.

We begin by proving Equation (3.44) by induction. First note that Equation (3.44) at level $\ell = 1$ we have incurrred no error from previous levels and Lemma 3.2.12 gives $\mathbb{E}[\mathsf{err}(1)] \leq (1+\beta)\mathsf{opt}(1)$. Now suppose the analog of Equation (3.44) holds for each level up to $\ell - 1$. Then, since $\mathsf{err}(\ell')^2 \geq 0$,

$$\begin{aligned}
&\mathbb{E}\big[\mathsf{err}(1)^2 + \cdots + \mathsf{err}(\ell-1)^2\big] \\
&\qquad \leq \big(1 + \beta \cdot (1 + (1+\beta) + \cdots + (1+\beta)^{\ell-2})\big) \cdot \big(\mathsf{opt}(1)^2 + \cdots + \mathsf{opt}(\ell-1)^2\big) \\
&\qquad = (1+\beta)^{\ell-1} \cdot \big(\mathsf{opt}(1)^2 + \cdots + \mathsf{opt}(\ell-1)^2\big). \qquad (3.46)
\end{aligned}$$

Applying Lemma 3.2.12 for each $j = 1, 2, \ldots 2^\ell$ and summing over $j$ we find

$$\mathbb{E}\big[\mathsf{err}(\ell)^2\big] = \mathbb{E}\big[\mathbb{E}\big[\mathsf{err}(\ell)^2 \big| \mathcal{F}_\ell\big]\big] \leq \mathbb{E}\left[\mathsf{opt}(\ell)^2 + \beta \sum_{j=1}^{2^\ell} E_j^{(\ell)}\right]. \qquad (3.47)$$

Observe that the sum over all blocks of $A^{(\ell)}$ except the on-diagonal blocks and the off-diagonal low-rank blocks at level $\ell$ is expressed as

$$\sum_{j=1}^{2^\ell} \sum_{\substack{i=1 \\ i \neq j, j\pm 1}}^{2^\ell} \|A_{i,j}^{(\ell)}\|_{\mathsf{F}}^2 = \sum_{j=1}^{2^\ell} \sum_{\substack{i=1 \\ i \neq j, j\pm 1}}^{2^\ell} \|A_{j\pm1,i}^{(\ell)}\|_{\mathsf{F}}^2.$$

Therefore,

$$\sum_{j=1}^{2^\ell} E_j^{(\ell)} \le \sum_{j=1}^{2^\ell} \left( \|A_{j\pm1,j}^{(\ell)} - (A_{j\pm1,j}^{(\ell)})_k\|_{\mathsf{F}}^2 + \sum_{\substack{i=1 \\ i\neq j,j\pm1}}^{2^\ell} \frac{1}{2}\|A_{i,j}^{(\ell)}\|_{\mathsf{F}}^2 + \sum_{\substack{i=1 \\ i\neq j,j\pm1}}^{2^\ell} \frac{1}{2}\|A_{j\pm1,i}^{(\ell)}\|_{\mathsf{F}}^2 \right)$$

$$= \mathsf{opt}(\ell)^2 + \sum_{j=1}^{2^\ell} \sum_{\substack{i=1 \\ i\neq j,j\pm1}}^{2^\ell} \|A_{i,j}^{(\ell)}\|_{\mathsf{F}}^2$$

$$= \mathsf{opt}(\ell)^2 + \big(\mathsf{err}(1)^2 + \cdots + \mathsf{err}(\ell-1)^2\big). \tag{3.48}$$

In the final equality above we have used that the $(i,j)$ blocks for $i \neq j, j \pm 1$ contain exactly the errors made at previous levels. Hence, taking the expectation of Equation (3.48), using Equation (3.46), and plugging into Equation (3.47) gives

$$\mathbb{E}\big[\mathsf{err}(\ell)^2\big] \le \mathsf{opt}(\ell)^2 + \beta\big(\mathsf{opt}(\ell)^2 + (1+\beta)^{\ell-1}\big(\mathsf{opt}(1)^2 + \cdots + \mathsf{opt}(\ell-1)^2\big)\big)$$

$$\le \mathsf{opt}(\ell)^2 + \beta(1+\beta)^{\ell-1}\big(\mathsf{opt}(1)^2 + \cdots + \mathsf{opt}(\ell)^2\big),$$

which is Equation (3.44).

The proof of Equation (3.45) follows analogously. By Lemma 3.2.12 we have

$$\mathbb{E}\big[\widehat{\mathsf{err}}^2\big] \le \beta \sum_{j=1}^{2^L} \sum_{\substack{i=1 \\ i\neq j}}^{2^L} \mathbb{E}\big[\|\widehat{A}_{i,j}^{(L)}\|_{\mathsf{F}}^2\big] = \beta \sum_{\ell=1}^{L} \mathbb{E}\left[\mathsf{err}(\ell)^2\right] \le \beta(1+\beta)^L(\mathsf{opt}(1)^2+\cdots+\mathsf{opt}(L)^2),$$

where we used that $\sum_{j=1}^{2^L}\sum_{i=1,i\neq j}^{2^L} \|\widehat{A}_{i,j}^{(L)}\|_{\mathsf{F}}^2$ is precisely the errors made at previous levels and (3.46). This gives the desired inequality. □

### 3.2.9 Lower Bounds

In this section, we prove that our main result on HOLDR matrix approximation (Theorem 3.2.1) is close to optimal in terms of the number of matrix-vector products needed to solve Problem 3.

**Lower bound for exact recovery**

We begin with a simple lower bound in the setting where $A$ is exactly HODLR, in which case solving Problem 3 requires *exactly* recovering $A$. For this setting, we can appeal to prior work by Halikias and Townsend [84] on the query complexity of recovering matrices from *linearly parameterized matrix families*. In particular, any set $B_1, \ldots, B_m$ of *linearly independent* base matrices induces a linearly parameterized family $\mathcal{L}$ consisting of linear combinations of the base matrices; i.e.

$$\mathcal{L} = \left\{ \sum_{i=1}^{m} \theta_i B_i : \theta = [\theta_1, \ldots, \theta_m] \in \mathbb{R}^m \right\}$$

Halikias and Townsend observe (see [84], Lemma 2.2) that recovering a matrix $A \in \mathcal{L}$ requires at least $\lceil m/n \rceil$ matvec queries with $A$. In particular, any matrix-vector product with $A$ or $A^\top$ yields $n$ linear equations in entries of $\theta$. Since recovering $A$ amounts to determining the corresponding $\theta$, at least $m$ such equations are needed to uniquely determine $A$. We leverage this observation to prove the following:

**Theorem 3.2.14.** *Any algorithm that can recover any $n \times n$ matrix $A \in$ HODLR($k$) from adaptive matvec queries must use at least $\lceil k \log_2(n/k) \rceil$ queries. Therefore, solving Problem 3 requires $\Omega(k \log(n/k))$ matvec queries for any finite approximation factor $\Gamma$.*

*Proof.* The proof follows from observing that, while HODLR($k$) is not itself a linearly parameterized family, it contains a linearly parameterized family $\mathcal{L}$. In particular, consider the subset of HODLR($k$) matrices where every rank-$k$ block in the matrix is zero everywhere except in its first $k$ columns. The first $k$ columns in each block can be chosen to have entries with any value such that the block is rank-$k$, and the matrix is therefore HODLR($k$). As such, $\mathcal{L}$ is equal to the set of matrices that can be written as a linear combination of $B_1, \ldots, B_m$, where each $B_i$

is a matrix that is zero everywhere, but has a single 1 in one of the first $k$ columns in one of the rank-$k$ blocks of the HODLR structure. Recall that for an $n \times n$ matrix to be HODLR($k$), it must be that $n = n_{\text{base}} \cdot 2^p$ for integers $n_{\text{base}} \in (k/2, k]$ and $p = \log_2(n/n_{\text{base}})$.[8] It can be checked that the total number of base matrices, $m$, equals:

$$m = 2n_{\text{base}}n + nk(p-1) > nkp \geq nk\log_2(n/k)$$

So, by Lemma 2.2 in [84], we require $\lceil nk\log_2(n/k)\rceil/n\rceil = \lceil k\log_2(n/k)\rceil$ matvec queries to exactly recover a given $A \in \mathcal{L}$. Since $\mathcal{L} \subset$ HODLR($k$), the theorem follows. We note that to solve Problem 3 for finite $\Gamma$, we must exactly recover all $A \in$ HODLR($k$). □

**Lower bound for approximation**

Next, we prove a lower bound in the setting where $A$ is not exactly HODLR, and we seek to solve Problem 3 with error $\Gamma = (1+\varepsilon)$ for some $\varepsilon \in (0, 1)$. A natural approach for such a lower bound might be to leverage lower bounds for near-optimal rank-$k$ approximation, since HODLR approximation is a strictly harder problem. However, as discussed in Section 3.2.1, the best known lower bound for Frobenius-norm error rank-$k$ approximation in the matrix-vector query model is just $O(k + 1/\varepsilon^{1/3})$ [11]. Such a lower bound does not show that the multiplicative relationship between $k$ and poly$(1/\varepsilon)$ in our upper bound, Theorem 3.2.1, is necessary.

We obtain a stronger lower bound by instead proving a reduction from the problem of *fixed-pattern sparse matrix approximation*, which was recently studied in [6]. That work proves tight lower bounds in the matrix-vector query model

[8] If $n \leq k/2$ then the bound is vacuously true.

for approximating matrices by a wide variety of matrix classes with fixed sparsity patterns, including diagonal matrices, banded matrices, and more. Below, we state a special case of the lower bound from [6] for approximation by *block diagonal matrices*. This lower bound will be used to obtain our lower bound for HODLR approximation. For integers $n$ and $b$, such that $b$ divides $n$, we let $\mathcal{B}(n, b)$ denote the set of $n \times n$ block-diagonal matrices with blocks of size $b \times b$. We have the following:

**Lemma 3.2.15** (Corollary of Thm. 2 from [6][9])**.** *There are absolute constants $c, C > 0$ such that the following holds: For any $\varepsilon > 0$ and any positive integers $b, n$ such that $b$ divides $n$ and $n \geq cb/\varepsilon$, there is a distribution over $n \times n$ matrices $A$ such that any algorithm which accesses $A$ with $\leq Cb/\varepsilon$ adaptive matvec queries and returns an approximation $\widetilde{B} \in \mathcal{B}(n, b)$ must have $\|A - \widetilde{B}\|_{\mathsf{F}} \geq (1 + \varepsilon) \min_{B \in \mathcal{B}(n,b)} \|B - A\|_{\mathsf{F}}$ with probabilty $\geq 24/25$. The probability is taken over the randomness of $A$, as well as possible randomness in the algorithm.*

Lemma 3.2.15 establishes that, even to succeed with small positive probability, any algorithm for computing a near-optimal block-diagonal approximation to an arbitrary matrix $A$ requires $\Omega(b/\varepsilon)$ matvec queries to $A$. Intuitively this result is useful because block diagonal approximation is an *easier* problem than HODLR approximation. In particular, it is not hard to verify that $\mathcal{B}(n, 2k) \subset \text{HODLR}(k)$; matrices in $\mathcal{B}(n, 2k)$ are zero except on the diagonal and off-diagonal low-rank blocks of the final level of HODLR($k$) matrices. Formally, we prove that, if we had an algorithm for finding a near-optimal HODLR approximation to a given matrix, the result could be post-processed via projection onto $\mathcal{B}(n, 2k)$ to obtain a near-optimal block-diagonal approximation. If we found the near-optimal HODLR

[9]The result in [6] is stated for a particular choice of $n_0 = \Theta(b/\epsilon)$, but it can be seen to hold for all $n > n_0$ as well by simply padding the matrices in the hard input distribution with zeros to enlarge their size.

approximation with $o(k/\epsilon)$ matvec queries, we would violate Lemma 3.2.15. This approach results in the following lower bound:

**Theorem 3.2.16.** *There are absolute constants $c, C > 0$ such that the following holds: For any $k$, $\varepsilon > 0$, and $n \geq ck/\varepsilon$,[10] there is a distribution over $n \times n$ matrices $A$ such that any algorithm which accesses $A$ with $\leq Ck/\varepsilon$ adaptive matvec queries fails to solve Problem 3 with probability $\geq 24/25$.*

*Proof.* First note that we may assume $\epsilon \leq 1$, as the result already holds by Theorem 3.2.14 for larger values of $\epsilon$. Let $n = n_{\text{base}} 2^p$ for $n_{\text{base}} \in \lfloor k/2 + 1 \rfloor, \ldots, k$ and $p \geq 0$. Let $b = 2n_{\text{base}}$ and observe that $b \geq k$, and for $c \geq 2$, $b \leq n$. Let $S \in \{0,1\}^{n \times n}$ be an indicator matrix for a block-diagonal sparsity pattern with $n/b$ blocks of size $b$. I.e., $S$ is zero everywhere except that, for each $t \in 0, \ldots, n/b - 1$, $S_{i,j} = 1$ for $i, j \in \{bt + 1, \ldots, bt + b\}$. Observe that for any block diagonal matrix $B \in \mathcal{B}(n, b)$, $B \circ S = B$, where "$\circ$" denotes the entrywise product. Let $\bar{S}$ denote the compliment of $S$: for all $i, j$, $\bar{S}_{i,j} = 1 - S_{i,j}$. Observe that for any matrix $H$, we can write:

$$\|A - H\|_{\mathsf{F}}^2 = \|A \circ S - H \circ S\|_{\mathsf{F}}^2 + \|A \circ \bar{S} - H \circ \bar{S}\|_{\mathsf{F}}^2. \tag{3.49}$$

Define

$$B^* = A \circ S \operatorname*{argmin}_{B \in \mathcal{B}(n,b)} \|A - B\|_{\mathsf{F}}^2, \qquad H^* = \operatorname*{argmin}_{H \in \text{HODLR}(k)} \|A - H\|_{\mathsf{F}}^2.$$

Since $\mathcal{B}(n, b) \subset \text{HODLR}(k)$, we have that $\|A - H^*\|_{\mathsf{F}} \leq \|A - B^*\|_{\mathsf{F}}$.

Our proof approach is to show that, if we can solve Problem 3 with error $\Gamma = (1 + \varepsilon)$ using $m$ matrix-vector products, i.e. if we can find some $\widetilde{H} \in \text{HODLR}(k)$

[10] Recall that HODLR($k$) matrices are only defined for dimensions $n$ of the form $n = n_{\text{base}} 2^p$ for $n_{\text{base}} \leq k$ and $p \geq 0$. Formally, the theorem holds for any such $n$ that is $\geq ck/\varepsilon$.

that satisfies

$$\|A - \widetilde{H}\|_{\mathsf{F}} \leq (1+\varepsilon)\|A - H^*\|_{\mathsf{F}}, \tag{3.50}$$

then $\widetilde{H} \circ S$ is a near-optimal block diagonal approximation to $A$. Concretely, we will show that

$$\|A - \widetilde{H} \circ S\|_{\mathsf{F}} \leq (1+\varepsilon)\|A - B^*\|_{\mathsf{F}}. \tag{3.51}$$

Accordingly, we reach a contradiction to Lemma 3.2.15 unless $m = \Omega(k/\varepsilon) = \Omega(b/\varepsilon)$ matrix-vector products were used to compute $\widetilde{H}$.

To see why (3.50) implies (3.51), the main observation is that:

$$\|A \circ \bar{S} - \widetilde{H} \circ \bar{S}\|_{\mathsf{F}}^2 \geq \|A - H^*\|_{\mathsf{F}}^2. \tag{3.52}$$

(3.52) follows from the fact that the entries in the bottom level of a HODLR($k$) matrix (those corresponding to the ones in $S$) can be set arbitrarily without violating the HODLR property. Accordingly, by adjusting those entries to match $A$ exactly, we can obtain a HODLR approximation with error equal to $\|A \circ \bar{S} - \widetilde{H} \circ \bar{S}\|_{\mathsf{F}}^2$. The optimal approximation $H^*$ cannot have larger error.

Then, using (3.50), (3.49), and (3.52), we have:

$$\begin{aligned}(1+\varepsilon)^2\|A - H^*\|_{\mathsf{F}}^2 \geq \|A - \widetilde{H}\|_{\mathsf{F}}^2 &= \|A \circ S - \widetilde{H} \circ S\|_{\mathsf{F}}^2 + \|A \circ \bar{S} - \widetilde{H} \circ \bar{S}\|_{\mathsf{F}}^2 \\ &\geq \|A \circ S - \widetilde{H} \circ S\|_{\mathsf{F}}^2 + \|A - H^*\|_{\mathsf{F}}^2.\end{aligned}$$

We conclude that

$$\|A \circ S - \widetilde{H} \circ S\|_{\mathsf{F}}^2 \leq (2\varepsilon + \varepsilon^2)\|A - H^*\|_{\mathsf{F}}^2 \leq (2\varepsilon + \varepsilon^2)\|A - B^*\|_{\mathsf{F}}^2.$$

Moreover, $\|A - \widetilde{H} \circ S\|_{\mathsf{F}}^2 = \|A \circ S - \widetilde{H} \circ S\|_{\mathsf{F}}^2 + \|A - A \circ S\|_{\mathsf{F}}^2$, and recall that $A \circ S = B^*$, So:

$$\|A - \widetilde{H} \circ S\|_{\mathsf{F}}^2 \leq (2\varepsilon + \varepsilon^2)\|A - B^*\|_{\mathsf{F}}^2 + \|A - B^*\|_{\mathsf{F}}^2 = (1+\varepsilon)^2\|A - B^*\|_{\mathsf{F}}^2.$$

Taking a square root on both sides proves (3.51), and as explained above, Theorem 3.2.16 follows. □

We conclude the section by noting that Theorem 3.2.2 follows from combining Theorem 3.2.14 and Theorem 3.2.16. An interesting question for future work is to prove a lower bound that multiplicatively combines $k$, $\log(n/k)$, and $1/\varepsilon$; e.g., to prove that $m = \Omega(k \log(n/k)/\varepsilon)$ matrix-vector product queries are necessary to solve Problem 3. This is currently beyond the reach of our current approach and that of [6]. In particular, the lower bound instance in [6] is based on a random Wishart matrix, which has found applications in a number of prior results on lower bounds for adaptive matrix-vector query algorithms [37]. It can be checked that a constant factor near-optimal HODLR approximation for such a matrix can be found by simply returning a near-optimal block diagonal approximation for block size $O(k)$. Since that can be done with $O(k)$ matrix-vector products using the algorithm from [6], we cannot hope to use the Wishart instance to prove a lower bound that combines $k$ and $\log(n/k)$.

### 3.2.10 Numerical experiments and examples

In this section we provide several examples which provide insight into the behavior of peeling-based algorithms. The code used to generate our figures is available at `https://github.com/redacted_for_submission`.

In our experiments we consider two parameter choices for the Generalized Nyström Method based method Algorithm 3 and two parameter choices for a Randomized SVD based method Algorithm 4 described in Section 3.2.6. The parameter values are summarized in Table 3.5. The aim of our numerical experiments is to

provide some basic insight into how the various parameters of peeling algorithms impact their performance. There are many reasonable parameter combinations, and a comprehensive understanding of the best choice of parameters is far beyond the scope of this paper, but would be an interesting topic for future work.

| name | style | algorithm | $s_{\text{R}}$ | $t_{\text{R}}$ | $s_{\text{L}}$ | $t_{\text{L}}$ | # matvecs |
|---|---|---|---|---|---|---|---|
| GN1 | —— | Algorithm 3 | $k/\beta$ | 1 | $k/\beta^2$ | 1 | $O(k/\beta^2)$ |
| GN2 | ..... | Algorithm 3 | $k/\beta$ | $1/\beta$ | $k/\beta^2$ | 1 | $O(k/\beta^2)$ |
| RSVD1 | - - - | Algorithm 4 | $k/\beta$ | 1 | $\sim$ | 1 | $O(k/\beta)$ |
| RSVD2 | —·- | Algorithm 4 | $k/\beta$ | $1/\beta$ | $\sim$ | $1/\beta$ | $O(k/\beta^2)$ |

Table 3.5: Parameter choices used in our numerical experiments.

The RSVD1 and GN1 methods do not use random perforated sketches and can therefore be viewed as standard implementations of the peeling algorithm (e.g. as described in [111, 117]). The RSVD1 method uses the same sketching dimensions as required to obtain a $(1 + O(\beta))$-optimal rank-$k$ approximation to a matrix [85]. Similarly, the GN1 method uses the sketching dimensions required for the Generalized Nyström Method (without truncation to rank-$k$) to produce a low-rank approximation with error less than $(1 + O(\beta))$ times the optimal rank-$k$ approximation to a matrix [167].[11] The choice of paramaters for GN1 is also equivalent (after rescaling $\beta$) to the parameters Equation (3.41) needed for Theorem 3.2.13 to guarantee convergence for Generalized Nyström peeling method without perforation.

The RSVD2 and GN2 methods use the randomly perforated sketches described in Section 3.2.3. In particular, the GN2 method adds perforation to the sketch used to obtain the approximate range. Compared to GN1, this does not increase the asymptotic cost but improves the bound for $\Gamma$ which can be obtained from The-

[11] As we discuss in Section 3.2.3, it is believed that this also produces a $(1 + O(\beta))$-optimal approximation with truncation.

orem 3.2.13. Interestingly, in our numerical experiments, the accuracy of GN1 and GN2 is very similar. The RSVD2 method adds perforation to both sketches. This increases the asymptotic cost over RSVD1. However, as illustrated in Section 3.2.11, regardless of $\beta$, RSVD1 cannot solve Problem 3 for arbitrary $\Gamma > 1$. On the other hand, RSVD2 can.

For an approximation $\widetilde{A}$ to $A$, we will consider the relative and absolute errors defined by:

$$\text{relative error: } \frac{\|A - \widetilde{A}\|_{\mathsf{F}} - \|A - A^\star\|_{\mathsf{F}}^2}{\|A - A^\star\|_{\mathsf{F}}^2}, \qquad \text{absolute error: } \|A - \widetilde{A}\|_{\mathsf{F}},$$

where $A^\star := \min_{H \in \text{HODLR}(k)} \|A - H\|_{\mathsf{F}}^2$ is the best possible HODLR($k$) approximation to $A$. When $\widetilde{A}$ is HODLR($k$), the relative error corresponds to the smallest value of $\varepsilon$ for which Problem 3 is solved with $\Gamma = (1 + \varepsilon)$.

**Poisson's equation**

In this example, we take $A$ as the discretized solution operator to a differential equation. In particular, we consider the 2-dimensional Poisson's equation

$$\frac{\partial^2}{\partial x^2} u(x, y) + \frac{\partial^2}{\partial y^2} u(x, y) = f(x, y)$$

on a periodic domain $(x, y) \in [0, 1]^2$ (i.e., with boundary conditions $u(x, 0) = u(x, 1)$ and $u(0, y) = u(1, y)$. We discretize the problem on a uniform $t \times t$ grid $x_i = i/t$, $y_j = j/t$ for $i, j = 0, 1, \ldots, t - 1$ (so that $n = t^2$). The matrix $A$ is defined as the $n \times n$ linear map taking forcing data $f = \{f(x_i, y_j)\}_{i,j=1}^t \in \mathbb{R}^n$ to the solution approximate $u = \{u_{i,j}\}_{i,j=1}^t$. Matrix-vector products with $A$ (and $A^\top$) can be efficiently computed using a FFT-based 2D Poisson solver. Specifically, given $f \in \mathbb{R}^n = \mathbb{R}^{t \times t}$, we define $A$ by

$$Af = \text{IDFT}_2(D \, \text{DFT}_2(f)), \tag{3.53}$$

where $D : \mathbb{R}^{t\times t} \to \mathbb{R}^{t\times t}$ is the diagonal map with diagonal entries $D_{i,j} = -1/(\kappa_i^2 + \kappa_j^2)$, where the harmonics $\kappa_i$ are defined by $\kappa_i = i$ if $i \leq t/2 - 1$ and $\kappa_i = m - t$ if $i > t/2 - 1$. Here $\mathrm{FFT}_2$ and $\mathrm{IFFT}_2$ are respectively the 2-dimensional Discrete Fourier Transform and Inverse Discrete Fourier Transform. In our experiment we set $t = 32$ so that $n = t^2 = 1024$.

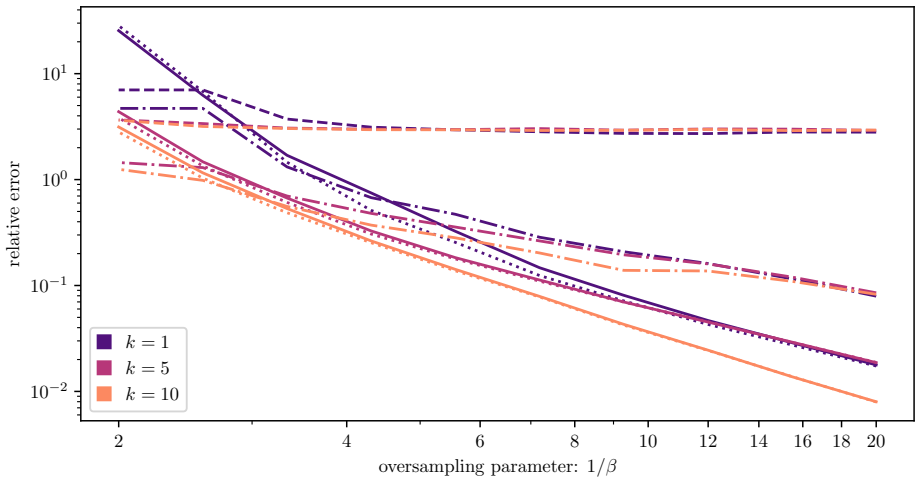



Figure 3.6: Relative error of peeling algorithms from Table 3.5 on discrete inverse Poisson matrix described in Equation (3.53) as a functino of the the oversampling parameter $\beta$ for several choices of the rank-parameter $k$. *Legend*: GN1 ( —— ), GN2 ( ····· ), RSVD1 ( - - - ), RSVD2 ( –·– ). *Takeaway*: Both Generalized Nyström Method-based variants seem to perform similarly, and produce an increasingly accurate output as $1/\beta$ increases. On the other hand, the standard randomized SVD-based variant, RSVD1, stagnates. The RSVD2 variant, which uses randomly perforated sketches, converges.

In Figure 3.6 we show the relative error of the algorithms from Table 3.5 for various ranks $k$ as a function of the oversampling parameters $\beta$ (averaged over 20 trials). As expected, as $\beta \to 0$, the relative errors of GN1, GN2, and RSVD1 all converge to zero. On the other hand, RSVD1 stagnates. In the left panel of Figure 3.7 we show the absolute error. Here we see that while the RSVD1 algorithm has a much higher relative error than the other algorithms, the absolute error is not significantly larger. Finally, in the right panel of Figure 3.7 we show

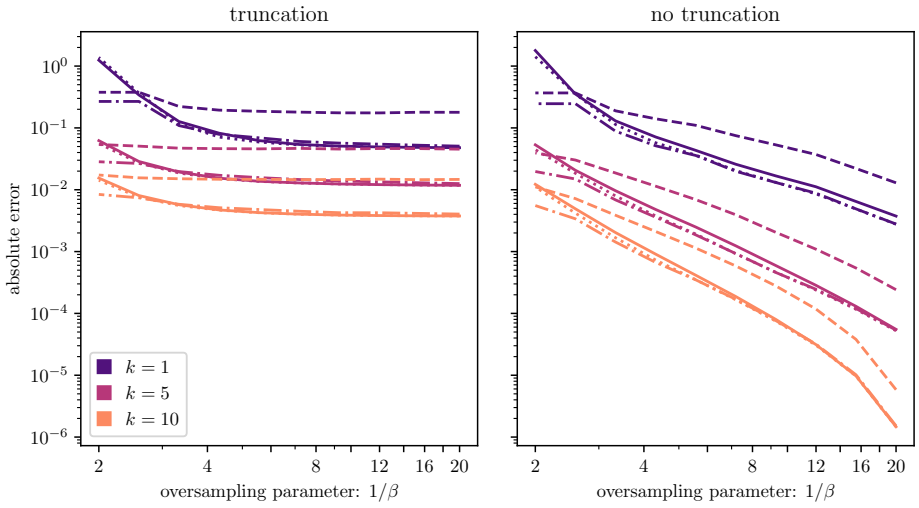



Figure 3.7: Absolute error of peeling algorithms from Table 3.5 on discrete inverse Poisson matrix described in Equation (3.53) as a function of the oversampling parameter $\beta$ with and without truncation to rank-$k$ for several choices of the rank-parameter $k$. *Legend*: GN1 ( —— ), GN2 ( ····· ), RSVD1 ( - - - ), RSVD2 ( –·– ). *Takeaway*: Without truncation, all of the algorithms can perform significantly better. However, the resulting approximations are not HODLR($k$), and are therefore more expensive to store and work with.

the absolute error of the algorithms without truncation to rank-$k$. This results in a much better approximations for the same number of matrix-vector products, but the resulting approximations have HODLR rank larger than $k$. The best tradeoff between HODLR rank and matvecs depends on the problem at hand and the computing environment.

### Kernel Matrix

In this example we consider a kernel matrix $A$ defined by

$$[A]_{i,j} = \begin{cases} 1/\|x_i - x_j\|_2 & i \neq j \\ 0 & i = j \end{cases}, \tag{3.54}$$

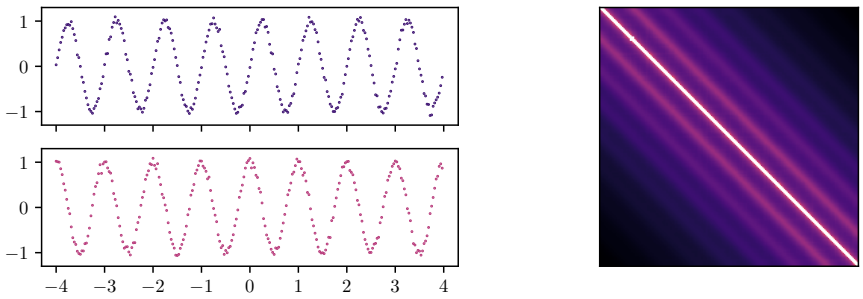


Figure 3.8: *Top/Bottom Left*: $(x_i, y_i)$ and $(x_i, z_i)$, ordered by $x$-value. *Right*: Log-magnitude of entries of $A$ defined in Equation (3.54). In both plots we subsample the data to 256 points for visual clarity.

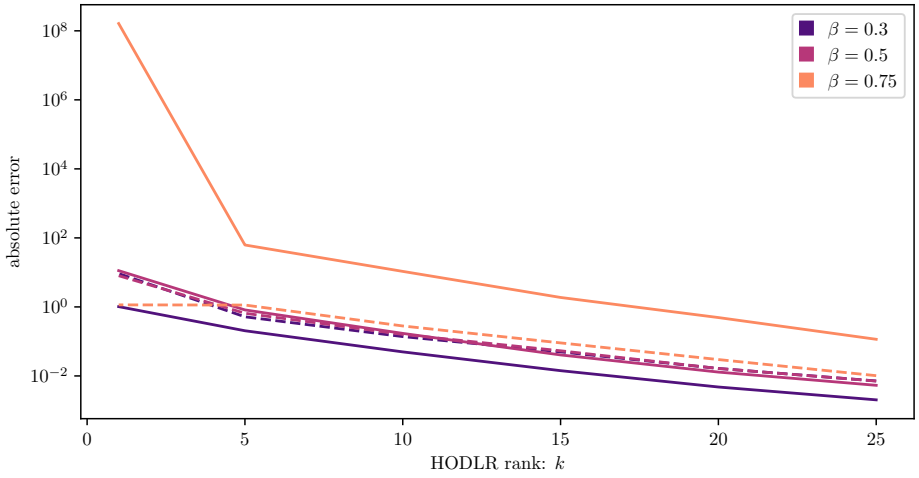



Figure 3.9: Absolute error of peeling algorithms from Table 3.5 on the kernel matrix described in Equation (3.54) as a function of the rank-parameter $k$ for several values of $\beta$. *Legend*: GN1 ( —— ), RSVD1 ( --- ). *Takeaway*: While RSVD1 may not produce near-optimal low-rank approximations, it can sometimes produce good approximations. In addition, when the sketch size used for the regression problem is too large, the Generalized Nyström Method does not produce highly accurate low-rank approximations.

where $\{x_i\}_{i=1}^n$ are a collection of points. Matrix-vector products with matrices such as $A$ can be efficiently performed using algorithms such as the Fast Multipole Method [77].

In our numerical experiments, we set $n = 16384$ and sample points $x_i =$

$(x_i, y_i, z_i)$ by taking $x_i$ as uniformly spaced points in $[-4, 4]$, $y_i = \sin(2\pi x_i) + 0.05\xi_i$, and $z_i = \cos(2\pi x_i) + 0.05\zeta_i$, where $\xi_i$ and $\zeta_i$ are independent standard normal random variables. Products with $A$ are performed using the Flatiron Institute's FMM3D code [7]. The left panel of Figure 3.10 shows a sample of the points we use, and the right panel shows the magnitude of the entries of the kernel matrix Equation (3.54) induced by these points. In Figure 3.11 we show the absolute error of the GN1 and RSVD1 methods as a function of the target rank, for several values of $\beta$.

### 3.2.11 Hard instances for RSVD-based peeling

In this section we provide two examples which illustrate that a RSVD-based implementation of the peeling algorithm, which truncates the low-rank approximations to rank-$k$ at every level, cannot solve Problem 3 for certain values of $\Gamma$.

We emphasize that the purpose of this section is simply to illustrate a potential failure mode which must be addressed by an algorithm provably solving Problem 3. In particular, these hard instances are not hard instances for more practical RSVD-based variants which do not truncate.

#### An illustrative example

Our first hard instance illustrates that all of the error from a one level can propagate to the next level. This example provides clear intuition for why the Randomized SVD-based peeling algorithm (with sketches of size $s_{\mathrm{R}}$ and truncation to rank-$k$ at every level) cannot solve Problem 3 for arbitrary $\Gamma > 1$.

Define, for some $\eta \gg 1$,

$$X = \begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix}, \qquad Y = \eta \begin{bmatrix} 0 & 0 \\ 0 & I_k \end{bmatrix}.$$

Construct the matrix[12]

$$A = \left[\begin{array}{cc|cc} 0 & X & Y & X \\ X & 0 & X & 0 \\ \hline Y & X & 0 & X \\ X & 0 & X & 0 \end{array}\right].$$

For notational convenience, we will write equality for the limits as $\eta \to \infty$. In particular, when $\eta \to \infty$, the randomized SVD (with any $s_{\mathrm{R}} \geq k$) will recover optimal rank-$k$ approximations to the bottom left and top right blocks.

We then remove the low-rank components we found at the first level to obtain

$$\left[\begin{array}{cc|cc} 0 & X & Y & X \\ X & 0 & X & 0 \\ \hline Y & X & 0 & X \\ X & 0 & X & 0 \end{array}\right] - \left[\begin{array}{cc|cc} 0 & 0 & Y & 0 \\ 0 & 0 & 0 & 0 \\ \hline Y & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] = \left[\begin{array}{cc|cc} 0 & X & 0 & X \\ X & 0 & X & 0 \\ \hline 0 & X & 0 & X \\ X & 0 & X & 0 \end{array}\right].$$

Note, however, that since the off-diagonal low-rank blocks of $A$ are not rank-$k$, we fail to zero out the off-diagonal blocks at the first level.

At the next level, the randomized SVD (with any $s_{\mathrm{R}} \geq k$) will exactly recover an orthonormal basis $Q$ containing the range of $X$. In particular, we perform a

[12]Note that the hard instance depends on the truncation rank-$k$ used by the algorithm. If we allow the algorithm to use an adaptively chosen rank or do not use truncation (both of which are often done in practice) then it would no longer be a hard instance.

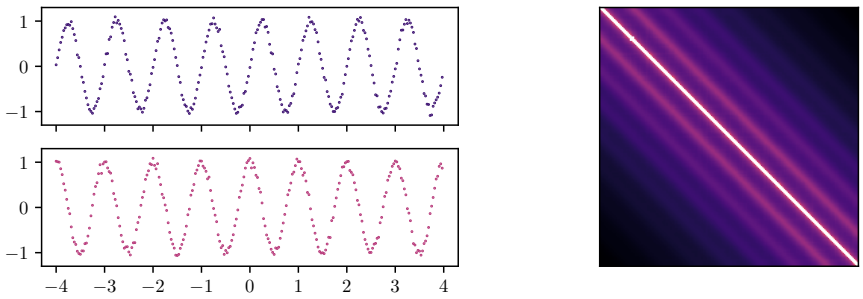


Figure 3.10: *Top/Bottom Left*: $(x_i, y_i)$ and $(x_i, z_i)$, ordered by $x$-value. *Right*: Log-magnitude of entries of $A$ defined in Equation (3.54). In both plots we subsample the data to 256 points for visual clarity.

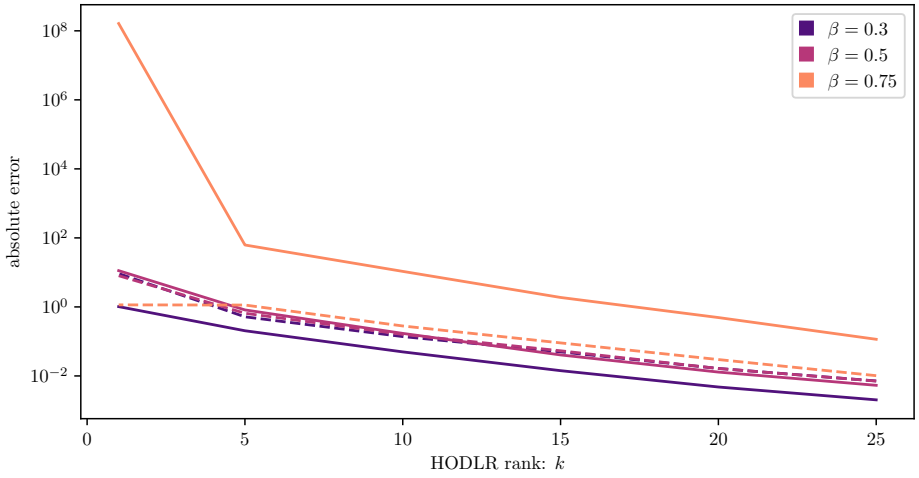



Figure 3.11: Absolute error of peeling algorithms from Table 3.5 on the kernel matrix described in Equation (3.54) as a function of the rank-parameter $k$ for several values of $\beta$. *Legend*: GN1 ( —— ), RSVD1 ( --- ). *Takeaway*: While RSVD1 may not produce near-optimal low-rank approximations, it can sometimes produce good approximations. In addition, when the sketch size used for the regression problem is too large, the Generalized Nyström Method does not produce highly accurate low-rank approximations.

product

$$\left[\begin{array}{cc|cc} 0 & X & 0 & X \\ X & 0 & X & 0 \\ \hline 0 & X & 0 & X \\ X & 0 & X & 0 \end{array}\right] \left[\begin{array}{c} \Omega_1^+ \\ 0 \\ \hline \Omega_3^+ \\ 0 \end{array}\right] = \left[\begin{array}{c} 0 \\ X(\Omega_1^+ + \Omega_3^+) \\ \hline 0 \\ X(\Omega_1^+ + \Omega_3^+) \end{array}\right].$$

We then compute[13] $Q = \mathrm{orth}(X(\Omega_1^+ + \Omega_3^+))$ and

$$\left[\begin{array}{cc|cc} 0 & Q^\top & 0 & Q^\top \end{array}\right] \left[\begin{array}{cc|cc} 0 & X & 0 & X \\ X & 0 & X & 0 \\ \hline 0 & X & 0 & X \\ X & 0 & X & 0 \end{array}\right] = \left[\begin{array}{cc|cc} 2Q^\top X & 0 & 2Q^\top X & 0 \end{array}\right],$$

and analogously for the super-diagonal off-diagonal blocks. Since $QQ^\top X = X$, our rank-$k$ approximation to each of the off-diagonal blocks at the second level is $2X$. In particular, we see that all of the error from the first level propagates to the second level.

Assuming we exactly recover the diagonals (if we do not, this can only increase the error), the final approximation is

$$\widetilde{A} = \left[\begin{array}{cc|cc} 0 & 2X & Y & 0 \\ 2X & 0 & 0 & 0 \\ \hline Y & 0 & 0 & 2X \\ 0 & 0 & 2X & 0 \end{array}\right].$$

As long as $\eta > 1$, the best HODLR approximation to $A$ is

$$A^\star = \left[\begin{array}{cc|cc} 0 & X & Y & 0 \\ X & 0 & 0 & 0 \\ \hline Y & 0 & 0 & X \\ 0 & 0 & X & 0 \end{array}\right].$$

Therefore we find $\|A - A^\star\|_{\mathsf{F}}^2 = 4\|X\|_{\mathsf{F}}^2$ while $\|A - \widetilde{A}\|_{\mathsf{F}}^2 = 8\|X\|_{\mathsf{F}}^2$.

[13] Here we assume $\mathrm{orth}(\cdot)$ is a deterministic function.

**Exponential growth**

The example in Section 3.2.11 shows that there are problems for which a Randomized SVD-based peeling algorithm cannot solve Problem 3 for small (constant) $\Gamma$. We now exhibit a problem instance for a Randomized SVD-based peeling algorithm which suggests that such algorithms cannot even guarantee better than an $O(n)$-factor approximation. This is exponentially large in the number of levels, and to the best of our knowledge, is the first instance demonstrating an exponential instability in the algorithm.

This is the first problem instance for which a variant of the peeling algorithm actually incurs a significant propagation of error from level-to-level that we are aware of. It remains an open question whether other variants of the peeling algorithm (e.g. using the randomized SVD without truncation) can fail for the approximation problem.

For any integer $L > 0$ let $n = 2^L$ and define the $n \times n$ matrix $A_n$ by

$$[A_n]_{i,j} = \begin{cases} 1 & j = 0,\ i \text{ odd} \\ \eta & j = 1,\ i = 2^1, 2^2, \ldots, 2^L \\ 0 & \text{otherwise} \end{cases}. \tag{3.55}$$

For each of the nonzero off-diagonal blocks, note that the second column is orthogonal to the first column. Hence, the optimal rank-1 HODLR approximation to $A_n$ is just $A_n$ with the first column set to zero. This means the error of the optimal approximation is from the $n/2 - 1$ ones in the first column excluding the $(1, 1)$ entry on the diagonal.

We set $\eta = 10^8$ and run an implementation of the peeling algorithm using the Randomized SVD with truncation to rank-1 at each level to obtain a HODLR rank-1 matrix $\widehat{A}_n$. This is repeated for increasing values of $n$. In Figure 3.12

we plot (as a function of $n$) the quantity $\|A_n - \widehat{A}_n\|_{\mathsf{F}}/\sqrt{n/2-1} - 1$, which is the smallest value of $\Gamma$ for which the output $\widehat{A}_n$ solves Problem 3 (averaged over 20 trials). This experiment suggests that $\|A_n - \widehat{A}_n\|_{\mathsf{F}}/\sqrt{n/2-1} - 1 = \Theta(n)$ as $n \to \infty$. Since $n = 2^L$, this is exponentially bad (with a constant base 2) in the number of levels $L$.

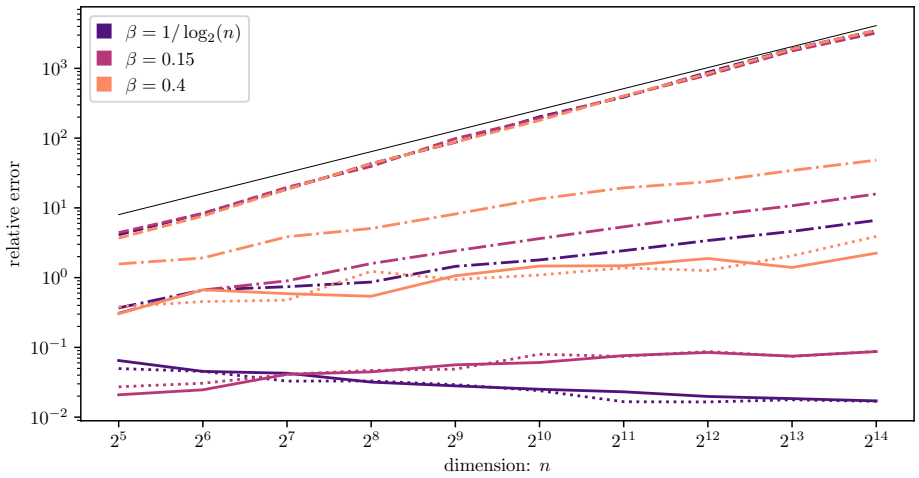



Figure 3.12: Relative error of peeling algorithms on hard instance described in Equation (3.55) for several different values of $\beta$. *Legend*: GN1 ( —— ), GN2 ( ····· ), RSVD1 ( - - - ), RSVD2 ( —·- ). *Takeaway*: The RSVD1 variant has error growing as $n$ (dotted reference line); i.e. exponential growth at every level. When the sketch size increases sufficiently quickly with $n$, the Generalized Nyström Method-based variants produce an error bounded independent of $n$.

### 3.2.12 Outlook

We have presented an algorithm provably solving the HODLR approximation problem in the matrix-vector query model. As far as we can tell, this is the first result on the approximation problem in the matrix-vector query model for any hierarchical matrix family. While this paper focuses on HODLR approximation for clarity of exposition, we expect the ideas used in our analysis can be extended to algo-

rithms for other hierarchical families. The most natural would extension would be to the coloring-based variant of the peeling algorithm for $\mathcal{H}$-matrices introduced in [106]. The $\mathcal{H}$ format is a generalization of HODLR which allows for a more general tree structure and recursive blocks off of the diagonal, and is significantly more efficient than HODLR for multi-dimensional problems.

Our work raises a number of interesting theoretical questions on approximation algorithms for hierarchical matrices:

- For any constant $c > 0$, we show that Problem 3 can be solved to accuracy $\Gamma = n^c$ with $O(k \log(n/k))$ matrix-vector queries. Up to constant, this is the best possible query complexity; as we prove in Theorem 3.2.2, exactly recovering a HODLR matrix requires $O(k \log(n/k))$ queries. It remains open whether there exist matvec query algorithms which solve Problem 3 to higher accuracy (e.g. $\Gamma = \log(n)$) with $O(k \log(n/k))$ queries.
- Hierarchical Semi-Separable (HSS) matrices are an important subfamily of HODLR matrices. The low-rank factors of HSS matrices at different levels are related in such a way that they can be stored and manipulated more efficiently than general HODLR matrices. In particular, there are a number of algorithms for recovering an exactly HSS matrix that require $O(k)$ matrix-vector products [105, 84]. It is therefore natural to ask whether there exists an HSS *approximation* algorithm producing $(1+\varepsilon)$-optimal HSS approximation using $O(\text{poly}(k, 1/\varepsilon))$ matvecs? A major difficulty is that, in contrast with HODLR matrices, we are unaware of a simple characterization of the best HSS approximation to a given matrix.
- Operator learning aims to learn representations of operators mapping functions to functions [113, 100, 36, 99, 73]. A number of recent works study the problem of approximating certain classes of infinite dimensional linear oper-

ators [33, 32] by hierarchically structured operators. Understanding how our analysis and theoretical techniques extend to the infinite dimensional setting may yield stronger theoretical guarantees for some problems in operator learning.

- Our structural perturbation bound Theorem 3.2.3 for low-rank approximation suggests fundamental differences between the behaviors of the Randomized SVD and Generalized Nyström Method in the presence of noisy matrix-vector products. The bound also highlights limitations in our current understanding of the impact of truncation on the Generalized Nyström Method. The best known bounds for the method with truncation are worse than without truncation, but we are unaware of any convincing evidence that such bounds are sharp. Further theoretical and numerical studies of these methods would be of interest.

CHAPTER 4

# DATA-EFFICIENCY OF ELLIPTIC PDE LEARNING

## 4.1 Introduction

Many scientific breakthroughs have come from deriving new partial differential equations (PDEs) from first principles to model real-world phenomena and simulating them on a computer to make predictions. However, many crucial problems currently lack an adequate mathematical formulation. It is not clear how to derive PDEs to describe how turbulence sheds off the wing of a hypersonic aircraft, how E. coli bacteria swim in unison to form an active fluid, or how atomic particles behave with long-range interactions. Rather than working from first principles, scientists are now looking to derive PDEs from real-world data using deep learning techniques [95].

The success of deep learning in language models, visual object recognition, and drug discovery is well known [103]. The emerging field of PDE learning hopes to extend this to discovering new physical laws by supplying deep learning models with experimental or observational data [95, 145]. PDE learning commonly seeks to recover features such as symmetries, conservation laws, solution operators, and the parameters of a family of hypothesized PDEs. In most deep-learning applications, a large amount of data is needed, which is often unrealistic in engineering and biology. However, PDE learning can be shockingly data-efficient in practice [113]. In particular, surprisingly little data is used to learn the solution operator, which maps the forcing term to the solution of the PDE.

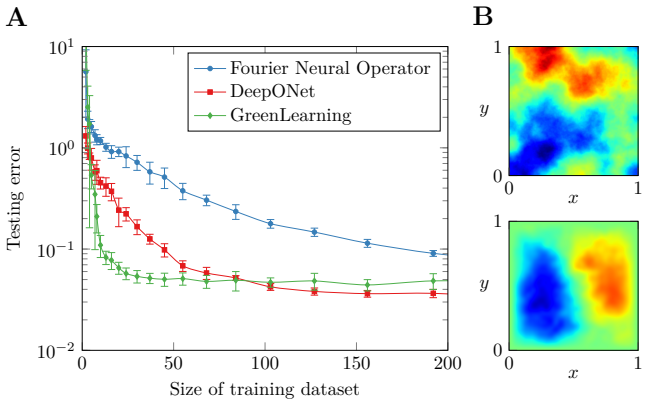



Figure 4.1: Elliptic PDE learning methods can be data-efficient. (A) Performance of three deep learning techniques in approximating the solution operator of the 2D Poisson equation with zero Dirichlet boundary condition on the domain $[0, 1]^2$. On small datasets, DeepONet and GreenLearning attain exponential decay of the testing error, while Fourier Neural Operator (FNO) attains algebraic decay. (B) A forcing term (top) and corresponding predicted solution (bottom) to the 2D Poisson equation by a FNO.

In this chapter[1], we provide a theoretical explanation of this behavior by showing that, for $\epsilon > 0$ sufficiently small, one can recover an $\epsilon$-approximation to the solution operator of a three-dimensional (3D) elliptic PDE with a training dataset of size about $\mathcal{O}(\log^5(1/\epsilon))$. Elliptic PDEs, such as the steady state heat equation, are ubiquitous in physics and model diffusion phenomena. Solution operators can produce surrogate data for data-intensive machine learning approaches such as learning reduced order models for design optimization in engineering, uncovering physics in climate models, and PDE recovery [95].

To illustrate the observed data-efficiency of PDE learning, we compare the performance of three techniques [108, 113, 30] for recovering the solution operator associated with the 2D Poisson equation in Figure 5.2. We vary the size of the training dataset, consisting of random forcing terms and corresponding solutions

[1]This section contains the excerpts from the main text and supplementary text of [32], to which I contributed theory for the stability analysis of the peeling algorithm. This work was joint with Nicolas Boullé and Alex Townsend. Nicolas Boullé performed the numerical experiments.

obtained by a numerical solver. We then evaluate the accuracy of the predicted solutions on a testing dataset with new forcing terms. The three methods are based on deep learning and differ in their neural network architectures. While the Fourier Neural Operator [108] exploits the fast Fourier transform for computationally efficient training, DeepONet [113] and GreenLearning [30] achieve a faster convergence rate on small training datasets. Here, DeepONet employs a complex network architecture with many parameters. In contrast, GreenLearning leverages prior knowledge that the solution operator is an integral operator and the approximation power of rational neural networks [34]. Green's function learning is observed to be the most data-efficient in Figure 5.2, as for a fixed training dataset size, it achieves the smallest testing error. All methods plateau due to discretization errors, and the training procedure gets stuck in a local minimum of the loss landscape rather than finding the global minimum. The rapid decay of testing errors prior to the plateau motivates our main result.

There is a lack of understanding for the efficiency of PDE learning methods with limited training data [113]. This work provides theoretical insights by constructing a provably data-efficient algorithm, showing that one can achieve exponential convergence when learning solution operators of elliptic PDEs.

Consider an unknown uniformly elliptic PDE in three dimensions, defined on a bounded domain $\mathcal{D} \subset \mathbb{R}^3$ with Lipschitz smooth boundary, with variable coefficients of the form:

$$\mathcal{L}u = -\nabla \cdot (A(x)\nabla u) = f, \quad x \in \mathcal{D}, \quad u|_{\partial\mathcal{D}} = 0, \tag{4.1}$$

where the coefficient matrix $A$ has bounded coefficient functions and is symmetric positive definite for all $x \in \mathcal{D}$. The weak assumptions on $\mathcal{D}$ and $A(x)$ allow for corner singularities and low regularity of the coefficients. The training data consists

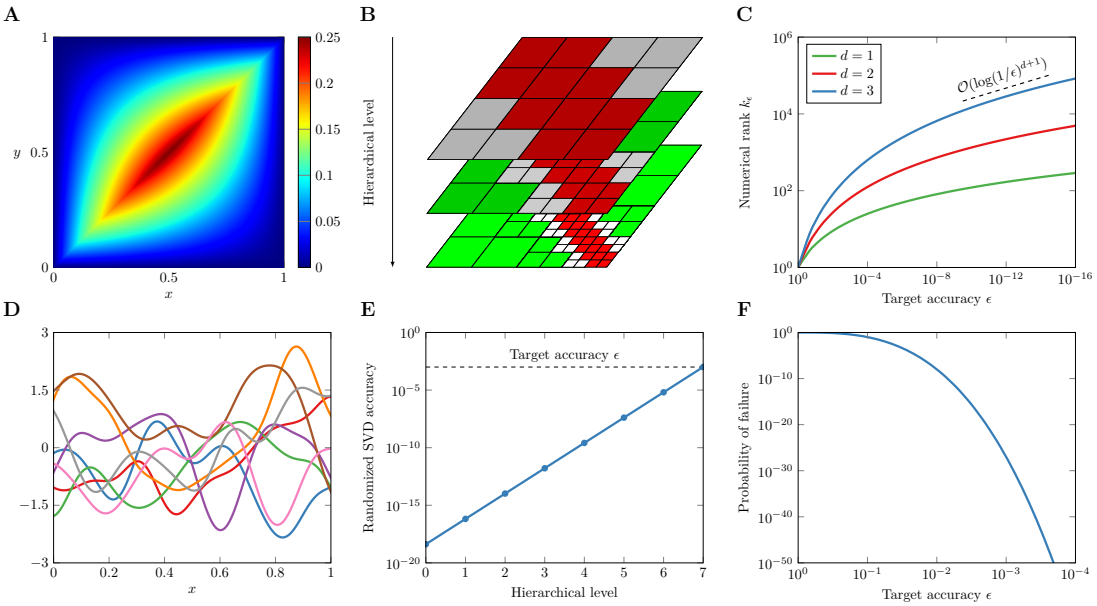



Figure 4.2: Properties of elliptic PDEs can be exploited to construct a provably data-efficient algorithm for recovering solution operators. (A) The Green's function associated with the 1D Poisson equation, which is the kernel of the solution operator. (B) We use the multi-scale (hierarchical) structure of a Green's function [111]. (C) On well-separated domains, the Green's function has rapidly decaying singular values [18], so it is efficiently recovered by the randomized singular value decomposition (SVD) [85]. (D) Forcing terms for the training dataset are randomly sampled from a Gaussian process. (E) The accuracy of the randomized SVD is carefully adapted on each hierarchical level to counterbalance the potential accumulation of errors in the reconstruction process. (F) An upper bound on the probability of failure of the reconstruction algorithm as a function of $\epsilon$.

of pairs of random forcing terms $f_1, \ldots, f_N$ and corresponding solutions $u_1, \ldots, u_N$ such that $\mathcal{L}u_j = f_j$ for $1 \leq j \leq N$. Deep learning techniques use this data to predict solutions to Equation (4.1) at new forcing terms by recovering the action of the solution operator $\mathsf{F}$, which is given by

$$\mathsf{F}(f) = \int_{\mathcal{D}} G(x, y) f(y) \, \mathrm{d}y, \tag{4.2}$$

where $G$ is the associated Green's function. For example, we visualize in Figure 4.2A the Green's function associated with the 1D Poisson equation. The random forcing terms in the training dataset are sampled from a Gaussian process (GP), i.e., they follow a multivariate Gaussian distribution when sampled on a

grid, and the covariance kernel determines the correlation between the function's entries and its smoothness.

Recent work [33] proves that for any $\epsilon > 0$ and 3D elliptic PDEs, a large number of input-output training pairs of size about $\mathcal{O}(\epsilon^{-6})$ is sufficient to recover an $\epsilon$-approximation $\tilde{\mathsf{F}}$ to $\mathsf{F}$ such that

$$\|\mathsf{F} - \tilde{\mathsf{F}}\|_2 \leq \epsilon \|\mathsf{F}\|_{\mathrm{HS}},$$

where $\|\cdot\|_2$ is the solution operator norm and $\|\cdot\|_{\mathrm{HS}}$ is the Hilbert–Schmidt norm. Once the $\epsilon$-approximation to $\mathsf{F}$ has been constructed, $\tilde{\mathsf{F}}$ can be used to study the stability and regularity of solutions of the PDE. For example, to see whether small perturbations of the input function lead to small changes in the output solution, or whether the solution has certain smoothness or decay properties for all forcing terms. Moreover, $\tilde{\mathsf{F}}$ can be used in numerical methods for approximating the solution of the PDE. By discretizing the input function and applying $\tilde{\mathsf{F}}$ as a surrogate for $\mathsf{F}$, one can obtain a numerical solution of the PDE that approximates the true solution. The integral kernel associated with the Hilbert–Schmidt operator $\tilde{\mathsf{F}}$ is also of interest, as it is an approximation to the Green's function, which can be exploited to recover linear conservation laws, symmetries, boundary effects, and dominant modes [30].

Our main result dramatically improves the required amount of training data to construct an $\epsilon$-approximation to $\mathsf{F}$ by exploiting the hierarchical structure of $G$ [18] and randomized linear algebra techniques [85, 120]. We derive a randomized algorithm that provably succeeds with exceptionally high probability and needs a training dataset size of only $\mathcal{O}(\log(1/\epsilon)^5[\log(\log(1/\epsilon)) + \log(1/\Gamma_\epsilon)]^4)$ input-output pairs.

**Theorem 4.1.1.** *Let $\epsilon > 0$ be sufficiently small, and $\mathsf{F}$ be the solution operator*

*associated with a 3D uniformly elliptic PDE of the form in Equation* (4.1)*. There exists a randomized algorithm that constructs an* $\epsilon$*-approximation* $\tilde{\mathsf{F}}$ *to* $\mathsf{F}$ *such that*

$$\|\mathsf{F} - \tilde{\mathsf{F}}\|_2 \leq \epsilon \|\mathsf{F}\|_{\mathrm{HS}},$$

*using* $\mathcal{O}(\log(1/\epsilon)^5[\log(\log(1/\epsilon)) + \log(1/\Gamma_\epsilon)]^4)$ *input-output pairs with probability* $\geq 1 - e^{-\log(1/\epsilon)^3}$.

The main contribution of Theorem 4.2.2 is a theoretical upper bound on the amount of training data required in elliptic PDE learning problems, which should deepen our understanding of existing deep learning techniques. Hence, the exponential convergence rate in Theorem 4.2.2 matches the one observed in the deep learning experiments of Figure 5.2A. We believe that this learning rate is near-optimal, as it exploits the multi-scale structure of Green's functions (see Figure 4.2B,C) and depends on the training dataset. The factor $0 < \Gamma_\epsilon \leq 1$ measures the quality of the training dataset at probing the dominant modes of the PDE, and a technical definition is available in the SI Appendix. We emphasize that the error bound must include a factor that quantifies the quality of the training dataset. If the forcing terms are too smooth, then $\Gamma_\epsilon$ is small. In contrast, choosing the covariance kernel of the GP such that the sampled functions are oscillatory usually ensures that $\Gamma_\epsilon$ is reasonable for learning $G$. In short, a small number of sufficiently diverse forcing terms is required (see Figure 4.2D).

The algorithm constructed in the proof of Theorem 4.2.2 achieves an approximation error measured in the solution operator norm. This mimics the typical measurement of accuracy of PDE learning techniques by comparing true and predicted solutions on a testing dataset of square-integrable forcing terms. Additionally, Theorem 4.2.2 employs random input-output pairs, where the forcing terms are sampled from a GP, so there is always some probability of failure. Fortu-

nately, we show this probability is exceptionally small. For $\epsilon < 10^{-3}$, failure is a once-in-a-cosmic-epoch event (see Figure 4.2F).

Theorem 4.2.2 is challenging to prove, and the whole argument is in the SI Appendix. The proof relies on the fact that the solution operator associated with a 3D elliptic PDE is an integral operator in the form of Equation (4.2). Firstly, the Green's functions related to 3D elliptic operators are square-integrable and have a bounded decay rate away from the diagonal of $\mathcal{D} \times \mathcal{D}$ [78]. Secondly, they possess a hierarchical structure [18] in the sense that they have rapidly decaying singular values when restricted to off-diagonal parts of the domain (green blocks in Figure 4.2B). We leverage the hierarchical structure, which has been historically exploited by fast solvers, in a data-driven context where the PDE is unknown. Combining these properties enables a generalization of the randomized SVD [85] known as the peeling algorithm [111] to simultaneously learn the off-diagonal blocks at any level of the hierarchy.

While the peeling algorithm is traditionally used to recover hierarchical matrices efficiently from matrix-vector products, we generalize it to approximate infinite-dimensional integral operators. To do so, we leverage insights from recent work that extends the peeling algorithm to arbitrary hierarchical partitions and dimensions [106]. This gives us a strategy to recover the Green's function level-by-level. However, proving the stability of peeling is an open question in numerical linear algebra. This is because the approximation errors from one level can potentially accumulate exponentially at later levels, thus degrading the convergence rate [33, 111].

We overcome this theoretical obstacle in the infinite-dimensional context by requiring an adaptive approximation accuracy at each level of the hierarchy. The

peeling algorithm ensures that the large-scale features of a Green's function are first learned to high accuracy by the randomized SVD. Then, we progressively decrease the accuracy requirement at subsequent levels, ensuring an overall $\epsilon$-approximation on each level of the partition at the end (see Figure 4.2E). The rapidly decaying singular values of the Green's function on off-diagonal parts of the domain (see Figure 4.2C) enable us to maintain a near-optimal exponential convergence rate with respect to the size of the training dataset. We then construct a global $\epsilon$-approximant by neglecting $G$ near the diagonal of the domain.

As one usually employs deep learning techniques to learn solution operators, our theoretical contributions can also lead to practical benefits. We believe that future training datasets benefit from taking into account prior knowledge of the PDE to improve the quality of the forcing terms at learning the solution operator. Similar ideas have already been employed in the field of visual object recognition through data-augmentation techniques. There is also an opportunity to design neural network architectures with hierarchical structures to capture the long-range interactions in PDE models. Finally, enforcing a different accuracy at different scales might improve the computational efficiency of existing PDE learning approaches.

In summary, we constructed a randomized algorithm that provably achieves an exponential convergence rate for approximating the solution operator associated with 3D elliptic PDEs in terms of the size of the training dataset. This provides a theoretical explanation for the observed performance of recent deep learning techniques in PDE learning. The proof techniques can be adapted to include elliptic PDEs in any dimension and time-dependent PDEs [33]. Recovering solution operators associated with hyperbolic PDEs, like wave equations, remains a significant open challenge. Moving forward, we plan to ramp up PDE learning techniques to

handle noisy experimental data, deal with data from emerging transient dynamics, and enforce conservation laws onto our solutions.

## 4.2 Theoretical results on Green's functions

We consider a second-order uniformly elliptic partial differential operator $\mathcal{L} : \mathcal{H}^2(\mathcal{D}) \cap \mathcal{H}_0^1(\mathcal{D}) \to L^2(\mathcal{D})$ on a bounded domain $\mathcal{D} \subset \mathbb{R}^d$ in spatial dimension $d = 3$ with a Lipschitz smooth boundary (see [82, Def. 6.2.33]). Here, $\mathcal{H}^k(\mathcal{D})$ denotes the $k$th Sobolev space, $H_0^k(\mathcal{D})$ is the closure of the infinitely differentiable functions compactly supported in $\mathcal{D}$ in $H^k(\mathcal{D})$, and $L^2(\mathcal{D})$ is the space of square-integrable functions over $\mathcal{D}$. We assume that the operator $\mathcal{L}$ takes the following divergence form:

$$\mathcal{L}u = -\nabla \cdot (A(x)\nabla u), \quad x \in \mathcal{D}, \quad u|_{\partial\mathcal{D}} = 0. \tag{4.3}$$

Here, the coefficient matrix $A(x) \in \mathbb{R}^{d\times d}$ is a symmetric positive definite matrix for every $x \in \mathcal{D}$ with $\kappa_C = \sup\{\lambda_{\max}(x)/\lambda_{\min}(x) \mid x \in \mathcal{D}\} < \infty$, and $A$ has bounded coefficient functions, *i.e.*, $A_{ij} \in L^\infty(\mathcal{D})$ for $1 \le i, j \le d$. Also, $\lambda_{\min}(x)$ and $\lambda_{\max}(x)$ denote the smallest and largest eigenvalues of $A(x)$. Under these conditions, it is known [78] that there is a Green's function $G : \mathcal{D} \times \mathcal{D} \to \mathbb{R} \cup \{\infty\}$ associated with Equation (4.3) so that the solution operator associated with $\mathcal{L}$ can be written as

$$\mathsf{F}(\phi)[x] = \int_{\mathcal{D}} G(x, y) f(y)\, \mathrm{d}y, \quad f \in L^2(\mathcal{D}). \tag{4.4}$$

In three dimensions, $G$ is square integrable [78], and its associated integral operator is a Hilbert–Schmidt (HS) operator [90].

PDE learning techniques aim to approximate the solution operator associated with an unknown PDE, such as Equation (4.3), from pairs of input-output func-

tions $\{(f_j, u_j)\}_{j=1}^N$ [95]. The forcing terms $f_j$ are usually sampled from a Gaussian process (GP) with a user-prescribed covariance kernel [108, 30, 113], and the associated solutions $u_j$ are acquired by solving the equation $\mathcal{L}u_j = f_j$ using a black-box solver (*i.e.*, through direct numerical simulations or by performing a physical experiment). In the past few years, several approaches have employed neural networks to approximate the solution operator in a wide range of problems [113, 108, 30, 172, 73, 99, 109]. While these methods have been very successful in practice, theoretical results are limited and mostly focus on the type and complexity of neural network architectures needed to approximate a given solution operator [113, 99, 102].

In this work, we focus on characterizing the sample complexity associated with solution operators of elliptic problems in three dimensions to understand the amount of training data needed to reach a given tolerance $0 < \epsilon < 1$. A number of studies have provided algebraic upper bounds of the form of $N = \mathcal{O}(\epsilon^{-2d})$ on the sample complexity for elliptic problems in three dimensions [33, 59, 44]. This bound naturally extends to time-dependent (parabolic) problems by changing the norm in which the error is measured to an $L^1$-norm to obtain a sample complexity of $\mathcal{O}(\epsilon^{-(3+d)/2}\log(1/\epsilon))$ [33]. However, the properties of Green's functions, such as their low-rank hierarchical structures [18, 33], and deep learning experiments [113, 30] suggest an exponentially smaller (poly-logarithmic) sample complexity of $\mathcal{O}(\log(1/\epsilon)^\alpha)$, for some power $\alpha \geq 1$. Recent work shows that such bounds can be attained by employing a recovery algorithm based on a sparse Cholesky factorization [151, 152]. However, the proof technique requires queries of the solution operator at deterministic piecewise polynomial inputs, which is not close to the setting employed by state-of-the-art deep learning approaches. A key difficulty in improving the polynomial error bound from [33] to a poly-logarithmic

sample complexity comes from the hierarchical structure of the Green's function. Hence, while one can exploit the hierarchical structure to greatly reduce the number of training pairs using a recursive peeling algorithm [111, 116, 105, 106], approximation errors may accumulate exponentially during the procedure and significantly deteriorate the error bound. Additionally, designing a provably stable algorithm for recovering $n \times n$ hierarchical matrices from $\mathcal{O}(\log n)$ matrix-vector products is an open problem in numerical linear algebra [33, Sec. 5.1].

In this paper, we generalize the peeling algorithm for hierarchical matrices to infinite dimensions and control the potential accumulation of errors by adaptively refining the tolerance. Our main theorem (see Theorem 4.2.2) provides a polylogarithmic upper bound for the sample complexity of solution operators associated with elliptic problems in three dimensions. We note that one has to express the error in the solution operator norm $\|\cdot\|$, defined in Definition 4.2.1. This metric is consistent with the usual way of measuring the model error on a test set in deep learning approaches [113, 108, 30, 172, 73, 99, 109].

**Definition 4.2.1** (Operator and HS norms)**.** *Let* $\mathsf{F} : L^2(\mathcal{D}) \to L^2(\mathcal{D})$ *be the solution operator associated with* $\mathcal{L}$ *and Green's function kernel* $G$*. Its operator norm* $\|\cdot\|_2$ *and HS-norm* $\|\cdot\|_{\mathrm{HS}}$ *are defined as [90, Chapt. 3]*

$$\|\mathsf{F}\|_2 = \sup\{\|\mathsf{F}(f)\|_{L^2(\mathcal{D})} \mid f \in L^2(\mathcal{D}),\, \|f\|_{L^2(\mathcal{D})} = 1\}, \qquad \|\mathsf{F}\|_{\mathrm{HS}} = \|G\|_{L^2(\mathcal{D}\times\mathcal{D})}.$$

The operator norm is a generalization of the spectral norm $\|\cdot\|_2$ for matrices to Hilbert–Schmidt operators and could alternatively be defined as the largest singular value of $\mathsf{F}$, while the HS-norm is a generalization of the Frobenius norm.

**Theorem 4.2.2.** *Let* $0 < \epsilon < 1$ *be sufficiently small,* $\mathcal{D} \subset \mathbb{R}^d$ *be a bounded Lipschitz domain in dimension* $d = 3$*, and* $\mathcal{L}$ *be an elliptic partial differential*

*operator in the form given by Equation* (4.3)*. If* $\mathsf{F} : L^2(\mathcal{D}) \to L^2(\mathcal{D})$ *is the solution operator associated with* $\mathcal{L}$*, then there is a randomized algorithm that constructs an approximation* $\tilde{\mathsf{F}}$ *of* $\mathsf{F}$ *using* $N = \mathcal{O}(\log(1/\epsilon)^{d+2}[\log(\log(1/\epsilon)) + \log(1/\Gamma_\epsilon)]^{d+1})$ *input-output pairs* $\{(f_j, u_j)\}_{j=1}^N$*, such that*

$$\|\mathsf{F} - \tilde{\mathsf{F}}\|_2 \leq \epsilon \|\mathsf{F}\|_{\mathrm{HS}},$$

*with probability greater than* $1 - e^{-\log(1/\epsilon)^d}$*. The factor* $0 < \Gamma_\epsilon \leq 1$ *is a measure of the quality of the forcing terms at approximating the eigenfunctions of* $\mathcal{L}$ *(see Equation* (4.28)*).*

The proof of Theorem 4.2.2 is summarized in Algorithm 5 and occupies the rest of this Supplementary Information Text. It exploits two theoretical properties of Green's functions: (1) the low-rank structure on well-separated domains (see Section 4.2) and (2) their decay away from the singularity along the diagonal of the domain (see Section 4.2.1). In Section 4.2.2, we review existing results for the discrete analogue of Green's function recovery from input-output pairs, which is equivalent to hierarchical matrix recovery from matrix-vector products. We describe the randomized singular value decomposition (see Section 4.2.2) and the peeling algorithm in Section 4.2.4 for reconstructing hierarchical low-rank matrices from matrix-vector products [111, 76, 17, 81]. We also include a perturbation analysis of the randomized singular value decomposition under additive perturbation errors in Section 4.2.3. In Section 4.4, we extend these results to the continuous case of Green's function recovery. We describe the previously established randomized singular value decomposition for Hilbert–Schmidt operators (see Section 4.4), then use this result to develop an infinite-dimensional analogue of the peeling algorithm (see Section 4.4). These ideas are finally combined in Sections 4.4.1 to 4.4.2 to show that one can stably recover Green's functions associated with el-

liptic operators of the form of Equation (4.3) using a poly-logarithmic number of input-output functions (*i.e.*, a spectrally efficient learning rate).

**Algorithm 5** Learning the solution operator via input-output functions.

**Input:** Black-box numerical solver associated with $\mathcal{L}$, GP covariance kernel $K$, tolerance $0 < \epsilon < 1$

**Output:** Approximation $\tilde{\mathsf{F}}$ of the solution operator $\mathsf{F}$ within relative error $\epsilon$

1: Determine the number of hierarchical levels $N_\epsilon$ using Green's functions' off-diagonal decay (Equation (4.9))
2: Set initial tolerance to $\epsilon_1 = \epsilon^p \ll \epsilon$
3: **for** $L = 1 : N_\epsilon$ **do**
4: Sample the GP with covariance kernel $K$ (Section 4.4)
5: Use peeling to sketch the Green's function on new admissible domains (Sections 4.2.4 and 4.4)
6: Approximate the Green's function on new admissible domains using the randomized SVD with tolerance $\epsilon_L$ (Section 4.4)
7: Increase the randomized SVD tolerance (Proposition 4.4.2)
8: **end for**
9: Pad the approximant $\tilde{\mathsf{F}}$ with zeros over the remaining non-admissible domains (Section 4.4.2)

### Low-rank structure on well-separated domains

Green's functions of elliptic operators have low numerical rank on well-separated domains $X, Y \subset \mathcal{D}$. More precisely, we say that $X$ and $Y$ satisfy a strong admissible condition if

$$\operatorname{dist}(X, Y) \geq \rho \max\{\operatorname{diam} X, \operatorname{diam} Y\}, \tag{4.5}$$

where $\rho > 0$ is an arbitrary constant measuring the relative distance between $X$ and $Y$. Let $0 < \epsilon < 1$ be a target relative accuracy. Bebendorf & Hackbusch [18, Thm. 2.8] proved that when the pair $(X, Y)$ is strongly admissible, there exists a separable approximation $G_k(x, y) = \sum_{i=1}^{k} u_i(x) v_i(y)$ of the Green's function such that

$$\|G - G_k\|_{L^2(X \times Y)} \leq \epsilon \|G\|_{L^2(X \times \hat{Y})},$$

where $k \leq k_\epsilon$ and $k_\epsilon = M \log(1/\epsilon)^{d+1}$ for some constant $M > 0$ that depends on $\mathcal{L}$. Here, $Y \subset \hat{Y} \subset \mathcal{D}$ denotes a domain slightly larger than $Y$ (see [18, Thm. 2.8] for the precise definition of $\hat{Y}$). Using the Eckart–Young–Mirsky theorem [64, 125], we deduce that Green's functions associated with elliptic operators have numerical rank bounded by $k_\epsilon$ on well-separated domains. As noted in [33], this property enables the use of randomized numerical linear algebra techniques for approximating Green's functions from input-output pairs. A pair $(X, Y) \subset \mathcal{D} \times \mathcal{D}$ that does not satisfy Equation (4.5) is called non-admissible.

This property leads to a hierarchical partition of $\mathcal{D} \times \mathcal{D}$ into admissible and non-admissible domains [76, 17, 81]. Without loss of generality, we assume that the domain of the hierarchical partition is $\mathcal{D} = [0, 1]^d$. Otherwise, one may rescale and shift $\mathcal{D}$ such that $\mathcal{D} \subset [0, 1]^d$ and consider the intersection of the partition of $[0, 1]^d$ and $\mathcal{D}$. Since $G$ is not of low rank on the initial domain $\mathcal{D} \times \mathcal{D}$, the domain $[0, 1]^d$ is dyadically partitioned into $2^d$ smaller subdomains given by halving each of the $d$ intervals $[0, 1]$. This leads to $N^{(L)} \leq 6^d 2^{dL}$ admissible domains at the level $1 \leq L \leq N$ of the hierarchical partition, where $N \geq 1$ is the number of hierarchical levels. Here, we chose an admissibility constant $\rho = 1/\sqrt{d}$ in Equation (4.5) so that non-admissible domains are neighboring boxes. We note that the rescaling procedure is equivalent to embedding $\mathcal{D}$ into a cube whose side length depends on the diameter of $\mathcal{D}$. Then, one constructs a hierarchical partition of the cube and intersects each component with $\mathcal{D}$ to obtain the desired partition. Therefore, the constant in front of the learning rate estimate in Theorem 4.2.2 depends implicitly on the geometry of $\mathcal{D}$ such as its diameter. A domain with poor aspect ratio leads to a suboptimal partition and therefore we do not expect to have the best constant in Theorem 4.2.2.

### 4.2.1 Off-diagonal decay

We determine the number of hierarchical levels using a second property of Green's functions associated with elliptic operators, known as off-diagonal decay, which controls the magnitude of Green's functions on non-admissible domains. Following [78, Thm. 1.1], we know that a Green's function in three dimensions decays away from the diagonal:

$$G(x,y) \leq \frac{c_{\kappa_C}}{|x-y|}\|G\|_{L^2(\mathcal{D}\times\mathcal{D})}, \quad x,y \in \mathcal{D},\, x \neq y, \tag{4.6}$$

where $c_{\kappa_C}$ is a constant depending on the spectral condition number of the operator $\mathcal{L}$ defined in Equation (4.3), and $|\cdot|$ denotes Euclidean distance. For a non-admissible pair $X \times Y \subset \mathcal{D} \times \mathcal{D}$ such that $\operatorname{dist}(X,Y) < \rho \max\{\operatorname{diam} X, \operatorname{diam} Y\}$, the $L^2$-norm of $G$ over $X \times Y$ can be bounded by integrating Equation (4.6), following the argument of [33, Sec. 4.2]. Let $r = (2+\rho)\max\{\operatorname{diam} X, \operatorname{diam} Y\}$ be a constant depending on the size of $X$ and $Y$, then the $L^2$-norm of the Green's function associated with $\mathcal{L}$ satisfies

$$\|G\|_{L^2(X\times Y)} \leq C_\rho c_{\kappa_C} r^2 \|G\|_{L^2(\mathcal{D}\times\mathcal{D})}, \tag{4.7}$$

where $C_\rho = \sqrt{2}\pi/\sqrt{3(2+\rho)^3}$ is an integration constant.

Using the hierarchical partition of $\mathcal{D}\times\mathcal{D}$ introduced in Section 4.2, we find that the maximum diameter of the non-admissible domains $X \times Y$ decays exponentially fast with the number of hierarchical levels as

$$\max\{\operatorname{diam} X, \operatorname{diam} Y\} \leq \frac{\sqrt{d}}{2^{N+1}}, \tag{4.8}$$

since $X$ and $Y$ are contained in cubes of side length $1/2^N$. We select $\rho = 1/\sqrt{d}$ and the number of hierarchical levels, $N_\epsilon$, such that $G$ has relative $L^2$-norm bounded

by $0 < \epsilon < 1$ on each non-admissible domain of the partition. Finally, combining Equations (4.7) and (4.8) yields

$$N_\epsilon = \frac{1}{2}\log_2(1/\epsilon) + \frac{1}{2}\log_2\left(c_{\kappa_C}\pi\sqrt{6\sqrt{2}+\sqrt{6}}\right) - 1 \sim \frac{1}{2\log 2}\log(1/\epsilon), \quad \epsilon \to 0. \tag{4.9}$$

**Remark 5** (Generalization to other systems)**.** *Elliptic operators [62] in dimensions 1 and 2 and elliptic operators with lower order terms [16] also admit a Green's function with a hierarchical structure and off-diagonal decay. The rate of off-diagonal decay in these cases results in a different constant $N_\epsilon$, so $N_\epsilon$ is dimension dependent. Thus, we suspect that the results presented in this work and and a version of Theorem 4.2.2 generalize with slightly different constants to these systems.*

### 4.2.2 Fast recovery of hierarchical matrices from matrix-vector products

A hierarchical off-diagonal low rank (HODLR) matrix $\mathbf{H}$ is a block-structured matrix which often arises from integral or differential equation problems [27, 80, 83], *e.g.*, by discretizing a Green's function (see Section 4.2). The matrix is hierarchically partitioned into sub-blocks, where the blocks located away from the diagonal are of low rank. A pair of admissible blocks $(X, Y)$ satisfies a so-called weakly admissible condition analogous to Equation (4.5) with $\rho$ if $\text{dist}(X, Y) > 0$. The number of levels in the hierarchy determines the sizes of the blocks. This section describes a "peeling" recovery algorithm [111] for approximating HODLR matrices from matrix-vector products with random test vectors using the randomized singular value decomposition [116].

### Randomized singular value decomposition

We first focus on recovering a low-rank matrix using matrix-vector products with test vectors. The randomized singular value decomposition (SVD) is one of the most popular algorithms for constructing a low-rank approximant of a large matrix from matrix-vector products with random Gaussian test vectors [85]. While the probabilistic error analysis performed by Halko et al. applies when the input vectors are standard Gaussian, the randomized SVD has been recently analyzed for Gaussian input vectors with correlated entries determined by a general covariance matrix [33, 29]. As we shall see later in Section 4.4, this generalization enables the application of the randomized SVD in infinite dimensions to compute low-rank approximants of Hilbert–Schmidt integral operators and can be used to approximate off-diagonal low-rank blocks of Green's functions (see Section 4.2.1 and [33]).

Let $\mathbf{A}$ be an $m \times n$ real matrix, where $m \geq n$, with SVD given by $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$, where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ have orthonormal columns, and $\mathbf{\Sigma}$ is an $m \times n$ rectangular diagonal matrix containing the singular values $\sigma_1(\mathbf{A}) \geq \cdots \geq \sigma_n(\mathbf{A}) \geq 0$ of $\mathbf{A}$. Let $k \geq 1$ be the target rank, $p \geq 2$ an oversampling parameter, and $\mathbf{\Omega} \in \mathbb{R}^{n\times(k+p)}$ a random test matrix with independent and identically distributed (i.i.d.) columns following a multivariate Gaussian distribution with mean 0 and covariance matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$, *i.e.*, $\mathbf{\Omega}(:, j) \sim \mathcal{N}(0, \mathbf{K})$ for $1 \leq j \leq k + p$. It is convenient to partition the reduced SVD of $\mathbf{A}$ as follows:

$$\mathbf{A} = \begin{array}{c} \begin{matrix} k & n-k \end{matrix} \\ \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} k & n-k \end{matrix} \\ \begin{bmatrix} \mathbf{\Sigma}_1 & \\ & \mathbf{\Sigma}_2 \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} n \end{matrix} \\ \begin{bmatrix} \mathbf{V}_1^* \\ \mathbf{V}_2^* \end{bmatrix} \end{array} \begin{matrix} k \\ n-k \end{matrix},$$

where the matrices $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ are diagonal and contain the first $k$ and last $n - k$ singular values of $\mathbf{A}$, respectively. In addition, we decompose the test vector $\mathbf{\Omega}$ in

the basis defined by the right singular vectors of $\mathbf{A}$ as $\mathbf{\Omega}_1 = \mathbf{V}_1^*\mathbf{\Omega}$ and $\mathbf{\Omega}_2 = \mathbf{V}_2^*\mathbf{\Omega}$.

The randomized SVD uses the following two-stage procedure for constructing a low-rank approximant to the matrix $\mathbf{A}$:

1. Form the matrix product $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$. This step requires $k+p$ matrix-vector products with $\mathbf{A}$.
2. Project the matrix $\mathbf{A}$ onto the range of $\mathbf{Y}$. One achieves this by constructing a matrix $\mathbf{Q}$ with orthonormal columns and the same column space as that of $\mathbf{Y}$. Then, we project $\mathbf{A}$ onto the range of $\mathbf{Y}$ as follows: $\tilde{\mathbf{A}} = \mathbf{Q}\mathbf{Q}^*\mathbf{A} = \mathbf{P}_\mathbf{Y}\mathbf{A}$, where $\mathbf{P}_\mathbf{Y}$ is the orthogonal projection matrix on the range of $\mathbf{Y}$ defined as $\mathbf{P}_\mathbf{Y} = \mathbf{Y}\mathbf{Y}^\dagger$. Here, $\mathbf{Y}^\dagger$ is the Moore–Penrose pseudo-inverse of $\mathbf{Y}$. Assuming that the matrix $\mathbf{A} = \mathbf{A}^*$, this projection step also requires $k+p$ matrix-vector products with $\mathbf{A}$ as $\tilde{\mathbf{A}} = \mathbf{Q}(\mathbf{A}\mathbf{Q})^*$; otherwise, it requires $k+p$ matrix-vector products with $\mathbf{A}^*$.

The approximation error in the Frobenius norm, $\|\cdot\|_\mathrm{F}$, between the matrix $\mathbf{A}$ and the computed low-rank approximant $\tilde{\mathbf{A}}$ can be characterized as follows [33, Thm. 1]:

$$\|\mathbf{A} - \mathbf{P}_\mathbf{Y}\mathbf{A}\|_\mathrm{F} \leq \sqrt{1 + t^2 s^2 \frac{3}{\gamma_k} \frac{k(k+p)}{p+1} \frac{\mathrm{tr}(\mathbf{K})}{\lambda_1}} \left( \sum_{j=k+1}^{n} \sigma_j^2(\mathbf{A}) \right)^{1/2},$$

which holds with probability $\geq 1 - t^{-p} - [se^{-(s^2-1)/2}]^{k+p}$, for arbitrary numbers $s, t \geq 1$. Here, $\mathrm{tr}(\mathbf{K})$ denotes the trace of the matrix $\mathbf{K}$, *i.e.*, the sum of its eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$. The quantity $0 \leq \gamma_k \leq 1$ measures the quality of the covariance matrix for approximating the right singular vectors of $\mathbf{A}$ and is defined as $\gamma_k = k/(\lambda_1 \,\mathrm{tr}((\mathbf{V}_1^*\mathbf{K}\mathbf{V}_1)^{-1}))$. Note that when $k = p$, one can simplify the bound

by choosing $t = e$ and $s = 2$ to obtain the following error bound [33, Eq. 38]:

$$\|\mathbf{A} - \mathbf{P}_{\mathbf{Y}}\mathbf{A}\|_{\mathrm{F}} \leq \left(1 + 20\sqrt{\frac{k}{\gamma_k}\frac{\mathrm{tr}(\mathbf{K})}{\lambda_1}}\right)\left(\sum_{j=k+1}^{n}\sigma_j^2(\mathbf{A})\right)^{1/2}, \tag{4.10}$$

which holds with probability $\geq 1 - 2e^{-k}$. This error bound allows us to recover a HODLR matrix $\mathbf{H}$ with high probability from matrix-vector products by approximating it block-by-block. Later in Section 4.2.4, we combine the randomized SVD with the peeling algorithm to greatly reduce the number of test input vectors needed by exploiting the hierarchical structure of $\mathbf{H}$. However, employing the peeling procedure leads to a potential accumulation of errors, which can only be controlled by a careful perturbation analysis of the randomized SVD.

### 4.2.3 Perturbation analysis of the randomized singular value decomposition

In this section, we assume that matrix-vector products with $\mathbf{A}$ introduce additive perturbation errors in the sample and projection steps of the randomized SVD, such that

$$\mathbf{Y}_{\text{noisy}} = \mathbf{A}\boldsymbol{\Omega} + \mathbf{E} = \mathbf{Y} + \mathbf{E}, \quad \text{and} \quad \tilde{\mathbf{A}} = \mathbf{P}_{\mathbf{Y}_{\text{noisy}}}\mathbf{A} + \mathbf{Q}\mathbf{E}_P^*. \tag{4.11}$$

Here, $\mathbf{Y}$ denotes the noiseless matrix-vector products with $\mathbf{A}$ which we cannot access, $\mathbf{E} \in \mathbb{R}^{m\times(k+p)}$ is the additive perturbation with $\|\mathbf{E}\|_{\mathrm{F}} \leq \epsilon$ for a given $\epsilon > 0$, and $\mathbf{E}_P \in \mathbb{R}^{n\times(k+p)}$ is the projection error satisfying $\|\mathbf{E}_P\|_{\mathrm{F}} \leq \epsilon$. While the original theoretical results on the randomized SVD are formulated in the noise-free setting [85, 120, 112, 146, 148], a large number of studies in matrix perturbation theory have considered bounding the approximation error between

the low-rank approximant and exact leading singular vectors of $\mathbf{A}$ when the observed matrix $\hat{\mathbf{A}}$ follows a "signal-plus-noise" model, *i.e.*, it contains noise as $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{E}$ [110, 175, 57, 174, 186]. Standard bounds in the literature can be refined by introducing additional assumptions on the perturbation error matrix $\mathbf{E}$, such as specific distribution of its entries [3, 135, 147, 165, 39, 2, 40, 41, 66, 69, 104]. Recently, these results have been exploited to derive explicit perturbation errors in the context of the randomized SVD [187].

Our setting differs from previous work because we do not have access to the perturbed sketch through matrix-vector products, as error is introduced additively after sketching (see Equation (4.11)). Therefore, we cannot use sketching to "learn" the perturbation error and obtain efficient error bounds with classical matrix perturbation theory results. Our aim is to analyze the error between the matrix $\mathbf{A}$ and its approximation $\tilde{\mathbf{A}}$ computed by the randomized SVD as

$$\|\mathbf{A} - \tilde{\mathbf{A}}\|_{\mathrm{F}} \leq \|\mathbf{A} - \mathbf{P}_{\mathbf{Y}_{\mathrm{noisy}}}\mathbf{A}\|_{\mathrm{F}} + \|\mathbf{Q}\mathbf{E}_P^*\|_{\mathrm{F}} \leq \|\mathbf{A} - \mathbf{P}_{\mathbf{Y}_{\mathrm{noisy}}}\mathbf{A}\|_{\mathrm{F}} + \epsilon. \tag{4.12}$$

Following Equation (4.12), we focus on analyzing the perturbed projection error term $\|\mathbf{A} - \mathbf{P}_{\mathbf{Y}_{\mathrm{noisy}}}\mathbf{A}\|_{\mathrm{F}}$, where $\mathbf{Y}_{\mathrm{noisy}} = \mathbf{Y} + \mathbf{E}$. This term can be directly bounded using the triangular inequality as $\|\mathbf{A} - \mathbf{P}_{\mathbf{Y}_{\mathrm{noisy}}}\mathbf{A}\|_{\mathrm{F}} \leq \|\mathbf{A} - \mathbf{P}_{\mathbf{Y}}\mathbf{A}\|_{\mathrm{F}} + \|(\mathbf{P}_{\mathbf{Y}} - \mathbf{P}_{\mathbf{Y}_{\mathrm{noisy}}})\mathbf{A}\|_{\mathrm{F}}$, and then using orthogonal projector error analysis [46, 155, 157, 156] to analyze the projection error $(\mathbf{P}_{\mathbf{Y}} - \mathbf{P}_{\mathbf{Y}_{\mathrm{noisy}}})$. However, this approach introduces technical difficulties whenever the additive error $\mathbf{E}$ has a non-zero component in the space spanned by the leading singular vectors of $\mathbf{A}$. Instead, we choose to directly analyze Equation (4.12) to estimate the error using a first-order expansion when $\epsilon$ is sufficiently small. The following proposition is an analogue of [85, Thm. 9.1] and provides a deterministic error bound for the randomized SVD with perturbed samples.

**Proposition 4.2.3** (Deterministic error bound)**.** *Let* $\mathbf{A}$ *be an* $m \times n$ *matrix with* $m \geq n$ *and* $1 \leq k \leq n$ *be a target rank, and* $p \geq 2$ *an oversampling parameter. Choose a test vector* $\mathbf{\Omega} \in \mathbb{R}^{n\times(k+p)}$ *and construct the perturbed sample matrix as* $\mathbf{Y}_{noisy} = \mathbf{Y} + \mathbf{E}$*, where* $\mathbf{Y} = \mathbf{A\Omega}$ *and* $\mathbf{E} \in \mathbb{R}^{m\times(k+p)}$ *satisfies* $\|\mathbf{E}\|_{\mathrm{F}} \leq \epsilon$*. Assuming that* $\mathbf{\Omega}_1 = \mathbf{V}_1^*\mathbf{\Omega}$ *is full rank and* $\epsilon > 0$ *is sufficiently small, then the approximation error satisfies*

$$\|\mathbf{A} - \mathbf{P}_{\mathbf{Y}_{noisy}}\mathbf{A}\|_{\mathrm{F}}^2 \leq \|\mathbf{\Sigma}_2\|_{\mathrm{F}}^2 + \|(\mathbf{\Sigma}_2\mathbf{\Omega}_2 + \mathbf{E}_2)(\mathbf{\Omega}_1 + \mathbf{\Sigma}_1^{-1}\mathbf{E}_1)^\dagger\|_{\mathrm{F}}^2, \tag{4.13}$$

*where* $\mathbf{E}_1 = \mathbf{U}_1^*\mathbf{E}$ *and* $\mathbf{E}_2 = \mathbf{U}_2^*\mathbf{E}$*.*

*Proof.* The proof of Proposition 4.2.3 closely follows the proof of [85, Thm. 9.1]. We first argue that the left singular vectors of $\mathbf{A}$ do not play any role by defining the auxiliary matrix $\tilde{\mathbf{A}}$ and perturbed sample matrix $\tilde{\mathbf{Y}}_{\text{noisy}}$ as

$$\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{A} = \begin{bmatrix}\mathbf{\Sigma}_1\mathbf{V}_1^*\\ \mathbf{\Sigma}_2\mathbf{V}_2^*\end{bmatrix}, \quad \text{and} \quad \tilde{\mathbf{Y}}_{\text{noisy}} = \tilde{\mathbf{A}}\mathbf{\Omega} + \mathbf{U}^*\mathbf{E} = \begin{bmatrix}\mathbf{\Sigma}_1\mathbf{\Omega}_1 + \mathbf{E}_1\\ \mathbf{\Sigma}_2\mathbf{\Omega}_2 + \mathbf{E}_2\end{bmatrix}.$$

Using the unitary invariance of the Frobenius norm and [85, Prop. 8.4], we have

$$\|\mathbf{A}-\mathbf{P}_{\mathbf{Y}_{\text{noisy}}}\mathbf{A}\|_{\mathrm{F}} = \|\mathbf{U}^*(\mathbf{I}-\mathbf{P}_{\mathbf{Y}_{\text{noisy}}})\mathbf{U}\tilde{\mathbf{A}}\|_{\mathrm{F}} = \|(\mathbf{I}-\mathbf{P}_{\mathbf{U}^*\mathbf{Y}_{\text{noisy}}})\tilde{\mathbf{A}}\|_{\mathrm{F}} = \|\tilde{\mathbf{A}}-\mathbf{P}_{\tilde{\mathbf{Y}}_{\text{noisy}}}\tilde{\mathbf{A}}\|_{\mathrm{F}}.$$

It is then sufficient to show that the following inequality holds: $\|\tilde{\mathbf{A}}-\mathbf{P}_{\tilde{\mathbf{Y}}_{\text{noisy}}}\tilde{\mathbf{A}}\|_{\mathrm{F}}^2 \leq \|\mathbf{\Sigma}_2\|_{\mathrm{F}}^2+\|(\mathbf{\Sigma}_2\mathbf{\Omega}_2+\mathbf{E}_2)(\mathbf{\Omega}_1+\mathbf{\Sigma}_1^{-1}\mathbf{E}_1)^\dagger\|_{\mathrm{F}}^2$. To achieve this, we first remark that, since $\mathbf{\Omega}_1$ is full rank, it has linearly independent columns and $\mathbf{\Omega}_1\mathbf{\Omega}_1^*$ is invertible such that $\mathbf{\Omega}_1^\dagger \coloneqq \mathbf{\Omega}_1^*(\mathbf{\Omega}_1\mathbf{\Omega}_1^*)^{-1}$ is well defined. Since $1 \leq k \leq \operatorname{rank}(\mathbf{A})$, the $k$th singular value of $\mathbf{A}$ is strictly positive and $\mathbf{\Sigma}_1$ is invertible. Let $d_{\mathbf{\Omega}_1}$ be the function on $k \times k$ matrices defined as $d_{\mathbf{\Omega}_1}(\mathbf{X}) = \det\left[(\mathbf{\Omega}_1 + \mathbf{\Sigma}_1^{-1}\mathbf{X})(\mathbf{\Omega}_1 + \mathbf{\Sigma}_1^{-1}\mathbf{X})^*\right]$ for $\mathbf{X} \in \mathbb{R}^{k\times k}$. $d_{\mathbf{\Omega}_1}$ is continuous and strictly positive at $\mathbf{X} = 0$. Hence, for $\epsilon$ sufficiently small, $d_{\mathbf{\Omega}_1}(\mathbf{E}_1) > 0$ and $(\mathbf{\Omega}_1 + \mathbf{\Sigma}_1^{-1}\mathbf{E}_1)^\dagger$ is well defined. This also implies that $\mathbf{\Sigma}_1\mathbf{\Omega}_1 + \mathbf{E}_1$

has full rank and, if $\mathbf{W}$ denotes the matrix containing the first $k$ columns of $\tilde{\mathbf{Y}}_{\text{noisy}}$ and zero afterwards, then

$$\operatorname{range}(\mathbf{W}) = \operatorname{range}\left(\begin{array}{c} k+p \\ \left[\begin{array}{c} \mathbf{I} \\ \mathbf{0} \end{array}\right] \end{array}\right) \begin{array}{c} k \\ n-k \end{array}, \quad \mathbf{W} = \begin{array}{c} k+p \\ \left[\begin{array}{c} \boldsymbol{\Sigma}_1\boldsymbol{\Omega}_1 + \mathbf{E}_1 \\ \mathbf{0} \end{array}\right] \end{array} \begin{array}{c} k \\ n-k \end{array}.$$

The range of $\mathbf{W}$ spans the same subspace as the first $k$ left singular vectors of the auxiliary matrix $\tilde{\mathbf{A}}$. We then use perturbation theory for orthogonal projectors to treat the matrix $\tilde{\mathbf{Y}}_{\text{noisy}}$ as a perturbation of $\mathbf{W}$, and introduce a matrix $\mathbf{Z}$ to flatten the first $k$ rows of the auxiliary sample matrix as

$$\mathbf{Z} = \tilde{\mathbf{Y}}_{\text{noisy}}(\boldsymbol{\Sigma}_1\boldsymbol{\Omega}_1 + \mathbf{E}_1)^\dagger = \tilde{\mathbf{Y}}_{\text{noisy}}(\boldsymbol{\Omega}_1 + \boldsymbol{\Sigma}_1^{-1}\mathbf{E}_1)^\dagger\boldsymbol{\Sigma}_1^{-1} = \begin{bmatrix} \mathbf{I} \\ \mathbf{F} \end{bmatrix}$$

and

$$\mathbf{F} = (\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2 + \mathbf{E}_2)(\boldsymbol{\Omega}_1 + \boldsymbol{\Sigma}_1^{-1}\mathbf{E}_1)^\dagger\boldsymbol{\Sigma}_1^{-1}.$$

We then follow the proof of [85, Thm. 9.1] to obtain the following inequality $\|(\mathbf{I} - \mathbf{P}_{\tilde{\mathbf{Y}}_{\text{noisy}}})\mathbf{A}\|_{\text{F}}^2 \leq \|\boldsymbol{\Sigma}^*(\mathbf{I}-\mathbf{P}_{\mathbf{Z}})\boldsymbol{\Sigma}\|_{\text{F}} \leq \|\mathbf{F}\boldsymbol{\Sigma}_1\|_{\text{F}}^2 + \|\boldsymbol{\Sigma}_2\|_{\text{F}}^2$. We have $\|\mathbf{F}\boldsymbol{\Sigma}_1\|_{\text{F}}^2 = \|(\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2 + \mathbf{E}_2)(\boldsymbol{\Omega}_1 + \boldsymbol{\Sigma}_1^{-1}\mathbf{E}_1)^\dagger\|_{\text{F}}^2$, which shows that Equation (4.13) holds. □

One can easily extract explicit upper bounds from Proposition 4.2.3 when the perturbation error is located in a subspace that is orthogonal to the first $k$ left singular vectors of $\mathbf{A}$, contained in the matrix $\mathbf{U}_1$. Moreover, the multiplicative factor $\boldsymbol{\Sigma}_1$ in the right-hand side of Equation (4.13) implies that the component of the error in the space generated by the $j$th singular vector must have magnitude bounded by the $j$th singular value of $\mathbf{A}$. Otherwise, one cannot hope to recover a good rank $k$ approximation, as the perturbation is too large in the direction of the $j$th singular vector. To alleviate this issue, we derive a probabilistic bound for the perturbed randomized SVD by combining Proposition 4.2.3 with a first-order

expansion of a perturbed pseudo-inverse matrix when the perturbation magnitude $\epsilon$ is sufficiently small.

**Theorem 4.2.4** (Probabilistic error bound)**.** *Let* $\mathbf{A}$ *be an* $m \times n$ *matrix with* $m \geq n$*,* $1 \leq k \leq n$ *a target rank,* $p \geq 2$ *an oversampling parameter, and* $\mathbf{\Omega}$ *be an* $n \times (k+p)$ *Gaussian matrix, where each column is i.i.d. and drawn from a multivariate Gaussian distribution with mean zero and covariance matrix* $\mathbf{K}$*. Assume that matrix-vector products with* $\mathbf{A}$ *and* $\mathbf{A}^*$ *introduce an additive error of* $\mathbf{E}$*, with* $\|\mathbf{E}\|_{\mathrm{F}} \leq \epsilon$ *for sufficiently small* $0 < \epsilon < 1$*. Then, the approximation error between the matrix* $\mathbf{A}$ *and its low-rank approximant* $\tilde{\mathbf{A}}$ *constructed by the randomized SVD satisfies*

$$\|\mathbf{A} - \tilde{\mathbf{A}}\|_{\mathrm{F}} \leq \sqrt{1 + t^2 s^2 \frac{3}{\gamma_k} \frac{k(k+p)}{p+1} \frac{\mathrm{tr}(\mathbf{K})}{\lambda_1}} \left( \sum_{j=k+1}^{n} \sigma_j^2(\mathbf{A}) \right)^{1/2} + \mathcal{O}(\epsilon), \tag{4.14}$$

*with probability* $\geq 1 - t^{-p} - [se^{-(s^2-1)/2}]^{k+p}$*.*

*Proof.* Combining Equation (4.12) and proposition 4.2.3 yields

$$\|\mathbf{A} - \tilde{\mathbf{A}}\|_{\mathrm{F}} \leq \|\mathbf{A} - \mathbf{P}_{\mathbf{Y}_{\mathrm{noisy}}} \mathbf{A}\|_{\mathrm{F}} + \epsilon \leq \|\mathbf{\Sigma}_2\|_{\mathrm{F}} + \|(\mathbf{\Sigma}_2 \mathbf{\Omega}_2 + \mathbf{E}_2)(\mathbf{\Omega}_1 + \mathbf{\Sigma}_1^{-1} \mathbf{E}_1)^{\dagger}\|_{\mathrm{F}} + \epsilon. \tag{4.15}$$

Let $\mathbf{X}$ be the $k \times (k+p)$ matrix such that $\epsilon \mathbf{X} = \mathbf{\Sigma}_1^{-1} \mathbf{E}_1$. By the same argument of the proof of Proposition 4.2.3, the matrix $\mathbf{\Omega}_1 + \epsilon \mathbf{X}$ has full rank for $\epsilon$ sufficiently small. Then, the matrix $(\mathbf{\Omega}_1 + \epsilon \mathbf{X})(\mathbf{\Omega}_1 + \epsilon \mathbf{X})^*$ is invertible, and we can compute a first-order expansion of the pseudo-inverse using an expansion for the matrix inverse as

$$\begin{aligned}
(\mathbf{\Omega}_1 + \epsilon \mathbf{X})^{\dagger} &= (\mathbf{\Omega}_1 + \epsilon \mathbf{X})^* [(\mathbf{\Omega}_1 + \epsilon \mathbf{X})(\mathbf{\Omega}_1 + \epsilon \mathbf{X})^*]^{-1} \\
&= (\mathbf{\Omega}_1 + \epsilon \mathbf{X})^* [\mathbf{\Omega}_1 \mathbf{\Omega}_1^* + \epsilon (\mathbf{X} \mathbf{\Omega}_1^* + \mathbf{\Omega}_1 \mathbf{X}^* + \epsilon \mathbf{X} \mathbf{X}^*)]^{-1} \\
&= \mathbf{\Omega}_1^{\dagger} + \mathcal{O}(\epsilon).
\end{aligned}$$

Therefore, the second term in the right-hand side of Equation (4.15) satisfies $\|(\mathbf{\Sigma}_2\mathbf{\Omega}_2+\mathbf{E}_2)(\mathbf{\Omega}_1+\mathbf{\Sigma}_1^{-1}\mathbf{E}_1)^\dagger\|_{\mathrm{F}} = \|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}+\mathcal{O}(\epsilon)$, since $\|\mathbf{E}_2\|_{\mathrm{F}} \leq \|\mathbf{E}\|_{\mathrm{F}} \leq \epsilon$ and $\|\mathbf{E}_1\|_{\mathrm{F}} \leq \epsilon$. We conclude that the approximation error is bounded for $\epsilon$ sufficiently small as $\|\mathbf{A}-\tilde{\mathbf{A}}\|_{\mathrm{F}} \leq \|\mathbf{\Sigma}_1\|_{\mathrm{F}} + \|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}} + \mathcal{O}(\epsilon)$, which we can bound using the proof of [33, Thm. 1]. □

### 4.2.4 Peeling algorithm for weakly admissible hierarchical matrices

We now consider a symmetric $n \times n$ hierarchical matrix $\mathbf{H}$, whose off-diagonal blocks are of low rank, and hence can be approximated by the randomized SVD.

A hierarchical partition of $\mathbf{H}$ is obtained by halving the index set of the rows and columns at each level. Therefore, if $\mathbf{H}$ is a symmetric rank-$k$ $n \times n$ HODLR matrix, it has $N = \lfloor \log_2(n) \rfloor$ levels, and the following block structure at the first level:

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 \\ \mathbf{H}_2^\top & \mathbf{H}_4 \end{pmatrix}, \tag{4.16}$$

where $\mathbf{H}_2$ and $\mathbf{H}_2^\top$ are considered off-diagonal blocks and are of rank at most $k$. The off-diagonal blocks are colored in green in Equation (4.16) and are usually called admissible blocks as they satisfy a weakly admissible condition analogue to Equation (4.5). The blocks along the diagonal, colored in red in Equation (4.16), are considered non-admissible. Further hierarchical levels are obtained by recursively subdividing the first two non-admissible blocks, $\mathbf{H}_1$ and $\mathbf{H}_4$, to refine the matrix

along the diagonal and obtain more admissible blocks with rank $k$ as follows:

$$\begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 \\ \mathbf{H}_2^\top & \mathbf{H}_4 \end{pmatrix} \rightarrow \left(\begin{array}{cc|c} \mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{H}_2 \\ \mathbf{H}_{12}^\top & \mathbf{H}_{14} & \\ \hline \multicolumn{2}{c|}{\mathbf{H}_2^\top} & \begin{matrix} \mathbf{H}_{41} & \mathbf{H}_{42} \\ \mathbf{H}_{42}^\top & \mathbf{H}_{44} \end{matrix} \end{array}\right) \tag{4.17}$$

$$\rightarrow \left(\begin{array}{cc|c|c} \mathbf{H}_{111} & \mathbf{H}_{112} & \mathbf{H}_{12} & \multirow{2}{*}{$\mathbf{H}_2$} \\ \mathbf{H}_{112}^\top & \mathbf{H}_{114} & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$\mathbf{H}_{12}^\top$}} & \begin{matrix} \mathbf{H}_{141} & \mathbf{H}_{142} \\ \mathbf{H}_{142}^\top & \mathbf{H}_{144} \end{matrix} & \\ \hline \multicolumn{3}{c|}{\multirow{2}{*}{$\mathbf{H}_2^\top$}} & \begin{matrix} \mathbf{H}_{411} & \mathbf{H}_{412} \\ \mathbf{H}_{412}^\top & \mathbf{H}_{414} \end{matrix} & \multirow{2}{*}{$\mathbf{H}_{42}$} \\ \hline \multicolumn{3}{c|}{} & \multirow{2}{*}{$\mathbf{H}_{42}^\top$} & \begin{matrix} \mathbf{H}_{441} & \mathbf{H}_{442} \\ \mathbf{H}_{442}^\top & \mathbf{H}_{444} \end{matrix} \end{array}\right). \tag{4.18}$$

Here, green matrices are of rank $k$ while red matrices need to be further partitioned to reveal off-diagonal low-rank structure.

The peeling algorithm [117, 111] uses a recursive elimination procedure to efficiently utilize each input-output pair to recover $\mathbf{H}$ in $\mathcal{O}(\log n)$ matrix-vector products. The main idea is to recover low-rank blocks of $\mathbf{H}$ level-by-level, *i.e.*, recover $\mathbf{H}$ one level at a time, starting with the largest blocks, working towards the diagonal, and finally recovering the finest block size at a chosen stopping point.

We refer the reader to a description of the peeling algorithm in Section 1.4.1.

**Remark 6.** *The peeling technique described in this section easily generalizes to a non-symmetric HODLR matrix* $\mathbf{H}$ *by requiring two additional matrix-vector products with* $\mathbf{H}^\top$ *at each hierarchical level. This is because the randomized*

*SVD requires matrix-vector products with the transpose of each admissible block, which doubles the total number of matrix-vector products to recover* $\mathbf{H}$ *to at most* $4(k+p)N$.

As it is known in the literature [111], the peeling algorithm is not stable and can potentially introduce errors that exponentially accumulate as one goes down the levels. We characterize the magnitude of these accumulating errors for a weakly admissible HODLR matrix. However, the analysis generalizes easily to strongly admissible hierarchical matrices (see Section 4.3).

**Proposition 4.2.5** (Perturbation error from sketch)**.** *Let* $1 \leq L \leq N$ *be a hierarchical level and assume that each of the off-diagonal blocks of* $\mathbf{H}$ *at lower levels* $1 \leq \ell \leq L-1$ *has been approximated to within an absolute accuracy of* $0 < \tilde{\epsilon}_\ell < 1$. *Let* $\mathbf{H}_i \in \mathbb{R}^{n/2^L \times n/2^L}$ *be an admissible block at level* $L$ *and* $\mathbf{x}_i \in \mathbb{R}^{n/2^L \times k}$ *an input matrix. Consider the exact sketch* $\mathbf{Z}_L = \mathbf{H}_i \mathbf{x}_i$, *and the perturbed sketch* $\tilde{\mathbf{Z}}_L$ *obtained by peeling from the input*[2] $\mathbf{X}_L = [\mathbf{0}; \mathbf{x}_1; \cdots; \mathbf{0}; \mathbf{x}_i; \mathbf{0}; \cdots; \mathbf{x}_{2^{L-1}}] \in \mathbb{R}^{n \times k}$, *where the Euclidean norm of each column of the matrices* $\mathbf{x}_i$ *is bounded by a constant* $C > 0$. *Then, the approximation error satisfies*

$$\|\mathbf{Z}_L - \tilde{\mathbf{Z}}_L\|_{\mathrm{F}} \leq C\sqrt{k} \sum_{\ell=1}^{L-1} 2^{\frac{L-1-\ell}{2}} \tilde{\epsilon}_\ell.$$

*Proof.* It suffices to prove this result for the upper left-most off-diagonal block at level $L$, which we denote as $\mathbf{H}_{1^{(L-1)}2}$, where $1^{(L-1)}$ represents $L-1$ concatenated 1's. This is because the perturbation argument for one block applies symmetrically to any other admissible block. Let $\mathbf{X}_L = [\mathbf{0}; \mathbf{x}_1; \mathbf{0}; \cdots; \mathbf{x}_{2^{L-1}}] \in \mathbb{R}^{n \times k}$ be the input matrix, such that $\|\mathbf{x}_j\|_{\mathrm{F}} \leq C\sqrt{k}$ for $1 \leq j \leq 2^{L-1}$. We consider the exact sketch

[2]This notation denotes a matrix whose components $\mathbf{0}, \mathbf{x}_1, \mathbf{0}, \ldots, \mathbf{0}, \mathbf{x}_{2^{L-1}} \in \mathbb{R}^{n/2^L \times k}$ are stacked vertically.

$\mathbf{Z}_L = \mathbf{H}_{1^{(L-1)}2}\mathbf{x}_1$ of the block $\mathbf{H}_{1^{(L-1)}2}$ with $\mathbf{x}_1$. Then, the perturbed sketch obtained by peeling satisfies:

$$\begin{aligned}\tilde{\mathbf{Z}}_L = \mathbf{Z}_L &+ \begin{pmatrix}\mathbf{I}_L & \mathbf{0}\end{pmatrix}(\mathbf{H}_{1^{(L-2)}2} - \tilde{\mathbf{H}}_{1^{(L-2)}2})[\mathbf{0};\mathbf{x}_2] \\ &+ \cdots \\ &+ \begin{pmatrix}\mathbf{I}_L & \mathbf{0}\end{pmatrix}(\mathbf{H}_2 - \tilde{\mathbf{H}}_2)[\mathbf{0};\mathbf{x}_{2^{L-2}+1};\cdots;\mathbf{0};\mathbf{x}_{2^{L-1}}],\end{aligned}$$

where $\mathbf{I}_L$ is the $n/2^L \times n/2^L$ identity matrix, *i.e.*, $\mathbf{I}_L$'s dimensions are the block size at level $L$. Moreover, the approximant $\tilde{\mathbf{H}}_{1^{(\ell)}2}$ of the admissible block $\mathbf{H}_{1^{(\ell)}2}$ at level $1 \leq \ell \leq L-1$ satisfies $\|\mathbf{H}_{1^{(\ell)}2} - \tilde{\mathbf{H}}_{1^{(\ell)}2}\|_{\mathrm{F}} \leq \tilde{\epsilon}_\ell$. Then, the sketch error is bounded by

$$\|\mathbf{Z}_L - \tilde{\mathbf{Z}}_L\|_{\mathrm{F}} \leq \sum_{\ell=1}^{L-1}\tilde{\epsilon}_i\sqrt{\sum_{j=1}^{2^{L-1-\ell}}\|\mathbf{x}_{2^{L-1-\ell}+j}\|_{\mathrm{F}}^2} \leq C\sqrt{k}\sum_{\ell=1}^{L-1}2^{\frac{L-1-\ell}{2}}\tilde{\epsilon}_\ell.$$

We note that the sketch error is zero at the first hierarchical level. □

Proposition 4.2.5 also characterizes the perturbation error for the projection step of the randomized SVD in the peeling algorithm. Then, if we consider the exact sketch $\mathbf{Z}_L^P = \mathbf{H}_i\mathbf{y}_i$ of an off-diagonal block $\mathbf{H}_i$ at level $L$, and the perturbed sketch $\tilde{\mathbf{Z}}_L^P$ obtained by peeling from the input $\mathbf{Y}_L = [\mathbf{y}_1;\mathbf{0};\cdots;\mathbf{0};\mathbf{y}_i;\mathbf{0};\cdots;\mathbf{0}] \in \mathbb{R}^{n\times k}$, whose components $\mathbf{y}_1,\ldots,\mathbf{y}_{2^{L-1}}$ have orthonormal columns. The approximation error satisfies

$$\|\mathbf{Z}_L^P - \tilde{\mathbf{Z}}_L^P\|_{\mathrm{F}} \leq \sqrt{k}\sum_{\ell=1}^{L-1}2^{\frac{L-1-\ell}{2}}\tilde{\epsilon}_\ell.$$

## 4.3 Peeling algorithm for strongly admissible partitions

We now generalize the peeling algorithm introduced in Section 4.2.4 to hierarchical matrices with arbitrary structure. As we saw in Section 4.2, Green's functions

associated with elliptic operators satisfy a strong admissibility condition, which implies that sub-blocks intersecting the diagonal are no longer considered admissible and may not have low numerical rank. As an example, for a symmetric block matrix $\mathbf{H}$ in Equation (4.16), the blocks $\mathbf{H}_2$ and $\mathbf{H}_2^\top$ are not admissible, and we instead recursively partition $\mathbf{H}$ as follows:

$$\begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 \\ \mathbf{H}_2^\top & \mathbf{H}_4 \end{pmatrix} \rightarrow \left(\begin{array}{cc|cc} \mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{H}_{21} & \mathbf{H}_{22} \\ \mathbf{H}_{12}^\top & \mathbf{H}_{14} & \mathbf{H}_{23} & \mathbf{H}_{24} \\ \hline \mathbf{H}_{21}^\top & \mathbf{H}_{23}^\top & \mathbf{H}_{41} & \mathbf{H}_{42} \\ \mathbf{H}_{22}^\top & \mathbf{H}_{24}^\top & \mathbf{H}_{42}^\top & \mathbf{H}_{44} \end{array}\right) \tag{4.19}$$

$$\rightarrow \left(\begin{array}{cc|cc|cc} \mathbf{H}_{111} & \mathbf{H}_{112} & \mathbf{H}_{121} & \mathbf{H}_{122} & \multicolumn{2}{c|}{\mathbf{H}_{21}} & \multicolumn{2}{c}{\mathbf{H}_{22}} \\ \mathbf{H}_{112}^\top & \mathbf{H}_{114} & \mathbf{H}_{123} & \mathbf{H}_{124} & & & & \\ \hline \mathbf{H}_{121}^\top & \mathbf{H}_{123}^\top & \mathbf{H}_{141} & \mathbf{H}_{142} & \mathbf{H}_{231} & \mathbf{H}_{232} & \multicolumn{2}{c}{\multirow{2}{*}{$\mathbf{H}_{24}$}} \\ \mathbf{H}_{122}^\top & \mathbf{H}_{124}^\top & \mathbf{H}_{142}^\top & \mathbf{H}_{144} & \mathbf{H}_{233} & \mathbf{H}_{234} & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$\mathbf{H}_{21}^\top$}} & \mathbf{H}_{231}^\top & \mathbf{H}_{233}^\top & \mathbf{H}_{411} & \mathbf{H}_{412} & \mathbf{H}_{421} & \mathbf{H}_{422} \\ & & \mathbf{H}_{232}^\top & \mathbf{H}_{234}^\top & \mathbf{H}_{412}^\top & \mathbf{H}_{414} & \mathbf{H}_{423} & \mathbf{H}_{424} \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$\mathbf{H}_{22}^\top$}} & \multicolumn{2}{c|}{\multirow{2}{*}{$\mathbf{H}_{24}^\top$}} & \mathbf{H}_{421}^\top & \mathbf{H}_{423}^\top & \mathbf{H}_{441} & \mathbf{H}_{442} \\ & & & & \mathbf{H}_{422}^\top & \mathbf{H}_{424}^\top & \mathbf{H}_{442}^\top & \mathbf{H}_{444} \end{array}\right). \tag{4.20}$$

Hierarchical matrices partitioned as in Equation (4.19) satisfy a strong admissible condition (see Equation (4.5)) in spatial dimension $d = 1$. Partitions in higher dimensions work similarly but are harder to visualize and possess a larger bandwidth of non-admissible blocks at each level. Constructing the test vectors $\mathbf{X}_L$ and $\mathbf{Y}_L$, where $1 \leq L \leq N$, is crucial to reducing the number of matrix-vector products in the randomized SVD when recovering $\mathbf{H}$. If one naively employs different test vectors at each admissible block, then the number of matrix-vector products grows exponentially with the number of levels rather than linearly.

Fortunately, peeling has been recently generalized to hierarchical matrices with arbitrary structure [106]. The nature of the hierarchical partition (weakly or strongly admissible) determines the structure and number of alternating inputs in the test matrices at each hierarchical level to maintain the linear growth of the number of matrix-vector products needed with respect to the number of levels. The peeling algorithm recovers a matrix with a strongly admissible partition in exactly the same way as for weakly admissible, except the input vectors have slightly different structures to isolate the actions of the low-rank sub-blocks. This structure is derived by the generalized coloring algorithm technique in [106], which we now describe.

The peeling algorithm aims to sketch all the admissible blocks of a given level $1 \leq L \leq N$ of a hierarchical matrix $\mathbf{H}$. We generate a set of test vectors such that for each admissible block at level $L$, a test vector satisfies a set of sketching constraints associated with that block. More specifically, for a given admissible block, we require a test vector that samples the block and avoids contribution from non-admissible blocks and admissible blocks of the same level. As an example, if we want to sketch the block $\mathbf{H}_{231}^{\top}$ in Equation (4.19), the test vector must vanish for the blocks $\mathbf{H}_{233}^{\top}$, $\mathbf{H}_{411}$, $\mathbf{H}_{412}$, $\mathbf{H}_{421}$, and $\mathbf{H}_{422}$. Of course, one could use as many test vectors as admissible blocks, but then the number of inputs would grow exponentially with the number of levels. To resolve this issue, [106] constructs inputs that satisfy the constraints for several different admissible blocks simultaneously by defining a constraint incompatibility graph. The vertices of the graph correspond to the constraint set for a particular admissible block, and vertices are connected if their constraints conflict with one another. A vertex coloring algorithm finds the minimal number of colors (chromatic number) denoted as $\chi(\mathbf{H}_L) \in \mathbb{N}$ for all the constraint sets at level $L$. For each color $1 \leq j \leq \chi(\mathbf{H}_L)$, there is a test vector

$\mathbf{\Omega}_L^{(j)}$ that satisfies the constraints of all the vertices colored by $j$. A key insight is that the chromatic number is bounded by a constant which is independent of the level of the partition and size of the matrix [106], meaning that one can use a constant number of input matrices at each level. In particular, if $\mathbf{H}$ satisfies a strongly admissible condition in dimension $d \geq 1$, we have the bound $\chi(\mathbf{H}_L) \leq 6^d$ for each level $L$ [106]. As a comparison, the chromatic number for the HODLR matrix with a weakly admissible partition analyzed in Section 4.2.4 is two.

We illustrate the input matrices for the strongly hierarchical matrix in Equation (4.19), whose chromatic number is bounded above as $\chi(\mathbf{H}_L) \leq 6$. At the first level, *i.e.*, the second matrix in Equation (4.19), there are four test vectors of the form:

$$\mathbf{X}_1^{(1)} = \begin{pmatrix} \mathbf{G}_1 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{X}_1^{(2)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{G}_2 \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{X}_1^{(3)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{G}_3 \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{X}_1^{(4)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{G}_4 \end{pmatrix},$$

where each $\mathbf{G}_i$ is a matrix of size $n/4 \times (k+p)$. Here, one can obtain sketches of the blocks $\mathbf{H}_{21}^\top$ and $\mathbf{H}_{22}^\top$ in Equation (4.19) using $\mathbf{X}_2^{(1)}$. With similar notation, six

test vectors are needed to sample all the admissible blocks at the second level:

$$\mathbf{X}_2^{(1)} = \begin{pmatrix} \mathbf{G}_1 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{G}_{11} \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{X}_2^{(2)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{G}_6 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{G}_{12} \end{pmatrix}, \quad \mathbf{X}_2^{(3)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{G}_7 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix},$$

$$\mathbf{X}_2^{(4)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{G}_8 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{X}_2^{(5)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{G}_9 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{X}_2^{(6)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{G}_{10} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}.$$

From this level and until the stopping point, only six test vectors are needed at each level. The following corollary generalizes the error analysis for the peeling algorithm performed on a weakly admissible hierarchical matrix to a strongly admissible partition.

**Corollary 4.3.1** (Accumulation of errors in strongly admissible partitions)**.** *Suppose the assumptions and notation in Proposition 4.2.5 hold for a hierarchical matrix* $\mathbf{H}$ *with a strongly admissible partition in dimension* $d \geq 1$*. Then, the sketch and projection errors during the peeling algorithm at level* $1 \leq L \leq N$ *satisfy the*

*following bounds:*

$$\|\mathbf{Z}_L - \tilde{\mathbf{Z}}_L\|_{\mathrm{F}} \leq (6^d - 3^d)C\sqrt{k}\sum_{\ell=1}^{L-1} 2^{\frac{L-1-\ell}{2}}\tilde{\epsilon}_\ell$$

*and*

$$\|\mathbf{Z}_L^P - \tilde{\mathbf{Z}}_L^P\|_{\mathrm{F}} \leq (6^d - 3^d)\sqrt{k}\sum_{\ell=1}^{L-1} 2^{\frac{L-1-\ell}{2}}\tilde{\epsilon}_\ell.$$

*Proof.* Let $1 \leq L \leq N$ be a hierarchical level. With the notation of Proposition 4.2.5, an input matrix $\mathbf{X}_L$ queries the action of a level-$L$ sub-block $\mathbf{H}(I_\alpha, I_\beta)$, where $\alpha$ and $\beta$ are level-$L$ nodes in the hierarchical tree and $I_\alpha, I_\beta \subset [0,1]^d$ are their corresponding index sets (see [81] for a description of index sets and hierarchical trees). Unfortunately, it also captures the actions of approximation errors in intersecting sub-blocks of $\mathbf{H} - \mathbf{H}^{(L-1)}$, due to using peeling at previous levels. As an example, if one tries to sketch the admissible block $\mathbf{H}_{421}^\top$ in Equation (4.19) with a test vector, then the matrix-vector product might contain a contribution from $\mathbf{H}_{22}^\top - \tilde{\mathbf{H}}_{22}^\top$ and $\mathbf{H}_{24}^\top - \tilde{\mathbf{H}}_{24}^\top$. We derive an upper bound on the number of level-$\ell$ intersecting blocks for $1 \leq \ell \leq L-1$. Each of these intersecting blocks is the restriction of $\mathbf{H} - \mathbf{H}^{(L-1)}$ to an admissible block of the form $I_{\alpha^{(\ell)}} \times I_\beta$ from a previous level $\ell$, where $\alpha^{(\ell)}$ is the level-$\ell$ subdomain that was eventually subdivided to get $\alpha$. That is, $\alpha^{(L-1)}$ is the parent of the $\alpha$, $\alpha^{(L-2)}$ is the parent of $\alpha$'s parent, and so on. We then count the number of level-$\ell$ nodes $\beta$ for which $I_{\alpha^{(\ell)}} \times I_\beta$ is an admissible domain. First, $I_{\alpha^{(\ell)}} \times I_\beta$ is an admissible domain if and only if the nodes $\alpha^{(\ell)}$ and $\beta$ belong to each other's interaction lists [106]. Moreover, the size of the interaction list of any node is bounded above by $6^d - 3^d$ [106], so this is also an upper bound on the number of intersecting blocks from level $\ell$. Finally, each previously learned block from level $\ell$ contributes $\tilde{\epsilon}_\ell$ to the sum of accumulated errors for $1 \leq \ell \leq L-1$ (see Proposition 4.2.5), and we have just shown that there are at most $6^d - 3^d$ of such intersecting blocks at each level $\ell$. Combining this

with Proposition 4.2.5, we derive the following bound on the perturbation error in the sketch step of the randomized SVD:

$$\|\mathbf{Z}_L - \tilde{\mathbf{Z}}_L\|_{\mathrm{F}} \leq \sum_{\ell=1}^{L-1} (6^d - 3^d)\tilde{\epsilon}_\ell \sqrt{\sum_{j=1}^{2^{L-1-\ell}} \|\mathbf{x}_{2^{L-1-\ell}+j}\|_{\mathrm{F}}^2} \leq (6^d - 3^d) C\sqrt{k} \sum_{\ell=1}^{L-1} 2^{\frac{L-1-\ell}{2}} \tilde{\epsilon}_\ell.$$

Similarly, we obtain a bound on the projection perturbation error:

$$\|\mathbf{Z}_L^P - \tilde{\mathbf{Z}}_L^P\|_{\mathrm{F}} \leq \sum_{\ell=1}^{L-1} (6^d - 3^d)\tilde{\epsilon}_\ell \sqrt{\sum_{j=1}^{2^{L-1-\ell}} \|\mathbf{y}_{2^{L-1-\ell}+j}\|_{\mathrm{F}}^2} \leq (6^d - 3^d)\sqrt{k} \sum_{\ell=1}^{L-1} 2^{\frac{L-1-\ell}{2}} \tilde{\epsilon}_\ell.$$

□

## 4.4 Stable recovery of Green's functions

This section generalizes the discussion of the randomized SVD and its perturbation analysis described in Section 4.2.3 to the continuous analogue of Hilbert–Schmidt operators, rather than matrices. We then use these results to derive the error analysis for the generalization of the peeling algorithm in infinite dimensions, showing that one can stably recover Green's functions on admissible domains. The key idea is to avoid the exponential accumulation of errors with each hierarchical level in the peeling procedure by taking advantage of the fast decay of a Green's function's singular values using a different target rank in the randomized SVD at each level. We then derive a probabilistic error bound for using infinite-dimensional peeling to approximate Green's functions on admissible domains from input-output pairs using the randomized SVD. Finally, we present a global error bound for the approximant in the operator norm that achieves a relative error of $\epsilon > 0$ on the entire domain.

**The randomized singular value decomposition for Hilbert–Schmidt operators**

The randomized SVD introduced in Section 4.2.2 has been recently generalized to compute low-rank approximants to Hilbert–Schmidt operators [33, 29]. In this section, we consider the particular setting of using the randomized SVD to approximate a Green's function $G$ over an admissible domain $X \times Y \subset \Omega \times \Omega$. Analogous to sampling random input vectors, we sample random input functions defined on $Y$ from a Gaussian process $\mathcal{GP}(0, K_Y)$, where $K_Y : Y \times Y \to \mathbb{R}$ is the covariance kernel. We construct the secondary kernel $K_Y$ by transforming a global kernel $K : \Omega \times \Omega \to \mathbb{R}$, defined on the entire domain as follows. First, we assume that $K$ is a continuous symmetric positive-definite kernel with bounded trace, *i.e.*,

$$\operatorname{tr}(K) = \int_\Omega K(x, x)\,\mathrm{d}x = \sum_{i=1}^{\infty} \lambda_i < \infty,$$

where $\lambda_1 \geq \lambda_2 \geq \cdots > 0$ are the eigenvalues of $K$. Using Mercer's theorem [123], there exists an orthonormal basis $\{e_i\}_i$ of $L^2(\Omega)$ of eigenfunctions of $K$, such that

$$K(x, y) = \sum_{i=1}^{\infty} \lambda_i e_i(x) e_i(y), \quad x, y \in \Omega,$$

where the convergence is absolute and uniform, and the eigenfunctions are continuous. We can then rescale, shift, restrict the $e_i$, and orthonormalize the family using Gram–Schmidt algorithm to generate an orthonormal family $\{e_{Y,i}\}_i$ in $L^2(Y)$. We then define the kernel $K_Y$ following its Mercer decomposition as

$$K_Y(x, y) = \sum_{i=1}^{\infty} \lambda_i e_{Y,i}(x) e_{Y,i}(y), \quad x, y \in Y. \tag{4.21}$$

This procedure guarantees that the secondary kernels have the same eigenvalues of $K$, and, in particular, that the trace is unchanged, which is crucial in the next sections to bound the norm of the random functions sampled from the associated GP.

As described in Section 4.2, $G$ has low numerical rank on $X \times Y$, and the singular values associated with the corresponding Hilbert–Schmidt integral operator (see Equation (4.4)) decay exponentially fast. Let $k \geq 1$ be a target rank and $k_\epsilon$ an oversampling parameter (see Section 4.2). We sample random functions $f_1, \ldots, f_{k+k_\epsilon}$ from the Gaussian process $\mathcal{GP}(0, K_Y)$, and define the random quasimatrix[3] $\Omega_{X\times Y} = \begin{bmatrix} f_1 & \cdots & f_{k+k_\epsilon} \end{bmatrix}$. Analogous to the discrete case [85], we consider the $k \times (k + k_\epsilon)$ matrix $\Omega_{X\times Y,1} = (\langle v_{i,X\times Y}, f_j \rangle)_{i,j}$, whose columns are i.i.d. and follow a multivariate Gaussian distribution with covariance matrix $\mathbf{C}_{X\times Y}$ [33, Lem. 1], defined as

$$[\mathbf{C}_{X\times Y}]_{ij} = \int_{Y\times Y} v_{i,X\times Y}(x) K_Y(x,y) v_{j,X\times Y}(y) \,\mathrm{d}x \,\mathrm{d}y, \quad 1 \leq i,j \leq k.$$

Here, $v_{i,X\times Y}$ is the $i$th singular vector of $G$ restricted to $X \times Y$. Following [33, Sec. 4.1.2], we sketch the HS operator at the random functions $f_1, \ldots, f_{k+k_\epsilon}$ and apply the randomized SVD to obtain an approximant $\tilde{G}$ to $G$ on the domain $X \times Y$. Then, with a target rank $k$ and an oversampling parameter $k_\epsilon$, the following error bound holds:

$$\|G - \tilde{G}_{X\times Y}\|^2_{L^2(X\times Y)} \leq \left(1 + t^2 s^2 \frac{6k_\epsilon}{\gamma_{k,X\times Y}} \sum_{j=1}^{\infty} \frac{\lambda_j}{\lambda_1}\right) \epsilon^2 \|G\|^2_{L^2(X\times \hat{Y})}$$

with probability $\geq 1 - t^{-k_\epsilon} - e^{-s^2(k+k_\epsilon)}$ for any $s, t \geq 1$. The covariance kernel quality, $\gamma_{k,X\times Y}$, is defined as $\gamma_{k,X\times Y} = k/(\lambda_1 \operatorname{tr}(\mathbf{C}^{-1}_{X\times Y}))$. We can then choose $t = \mathcal{O}(1)$ and $s = \mathcal{O}(1)$ such that

$$\|G - \tilde{G}_{X\times Y}\|_{L^2(X\times Y)} = \mathcal{O}(k_\epsilon^{1/2} \gamma_{k,X\times Y}^{-1/2} \epsilon) \|G\|_{L^2(X\times \hat{Y})} \tag{4.22}$$

holds with probability $\geq 1 - e^{-k_\epsilon}$. We emphasize that the perturbation error analysis for the randomized SVD stated in Theorem 4.2.4 generalizes to infinite dimensions so that Equation (4.22) holds whenever the magnitude of the perturbations is smaller than the target tolerance.

[3] See [161] for an introduction to the quasimatrix notation.

**Peeling for Green's functions**

We now generalize the peeling algorithm to recover Green's functions over hierarchically partitioned domains using the randomized SVD. In this section, we follow the model of HODLR matrix recovery algorithms [111, 117] to extend the peeling algorithm to the continuous case of Green's function recovery in dimensions $d \in \{1, 2, 3\}$. We recover $G$ over levels of the domain recursively, starting with the largest hierarchical level. First, the Green's function's behavior over all of the admissible domains at this level is recovered and stored in the function $G^{(1)}$. To get to the next level, we subdivide by halving the domain in all directions. Now, we seek to recover the function $G - G^{(1)}$ on any new admissible domains to obtain $G^{(2)}$. We dyadically partition any non-admissible domains and repeat the process to recover $G_2 = G - G^{(1)} - G^{(2)}$ over the next level's admissible domains. We repeat until we reach a pre-determined stopping point close to the diagonal.

Let $1 \leq L \leq N_\epsilon$ be a hierarchical level. We now describe the procedure for recovering $G_L = G - \sum_{\ell=1}^{L} G^{(\ell)}$ on each admissible subdomain of this level. The coloring algorithm easily applies to generating input functions over $\Omega$, as it was originally written in this generality. We first apply the graph coloring algorithm, described in Section 4.3, to determine the chromatic number $\chi(G_L)$ associated with the different constraint sets for sketching $G_L$. As established in [106], we know that for the strongly admissible partition in dimension $d$, we have the following bound: $\chi(G_L) \leq 6^d$. Each color number $1 \leq j \leq \chi(G_L)$ of the constraint graph gathers $n_j \geq 1$ constraints, which correspond to a set of admissible subdomains $\{X_i^j \times Y_i^j\}_{1 \leq i \leq n_j} \subset \Omega \times \Omega$. We then construct test functions $f_j \in L^2(\Omega)$ such that

$$\text{Supp}(f_j) \subset \cup_{i=1}^{n_j} Y_i^j, \quad \text{where} \quad f_j|_{Y_i^j} \sim \mathcal{GP}(0, K_{Y_i^j}), \quad 1 \leq i \leq n_j,$$

and $K_{Y_i^j}$ is defined in Equation (4.21). Given a target rank $k$ and an oversampling

parameter $k_\epsilon$ for learning the Green's function with the randomized SVD over the admissible domains, we generate the input quasimatrix $F_L^{(j)}$, where $1 \leq j \leq \chi(G_L)$, as follows:

$$F_L^{(j)} = \begin{bmatrix} f_{j,1} & \cdots & f_{j,k+k_\epsilon} \end{bmatrix}, \quad \text{where} \quad f_{j,l} \sim f_j \quad \text{for} \quad 1 \leq l \leq k + k_\epsilon.$$

Analogous to performing a matrix-vector product, sketching the Hilbert–Schmidt integral operator associated with $G_L$ with the quasimatrix $F_L^{(j)}$ yields $U_L^{(j)} = \begin{bmatrix} u_{j,1} & \cdots & u_{j,k+k_\epsilon} \end{bmatrix}$, where

$$\begin{aligned} u_{j,l}(x) &= \int_\Omega G_L(x,y) f_{j,l}(y)\,\mathrm{d}y \\ &= \int_\Omega G(x,y) f_{j,l}(y)\,\mathrm{d}y - \sum_{\ell=1}^{L} \int_\Omega G^{(\ell)}(x,y) f_{j,l}(y)\,\mathrm{d}y, \quad 1 \leq l \leq k + k_\epsilon. \end{aligned}$$

We now restrict the output functions $u_{j,l} \in L^2(\Omega)$ to isolate the action of $G$ on the admissible subdomains $\{X_i^j \times Y_i^j\}_{1 \leq i \leq n_j}$ associated with the $j$th color. Let $\mathcal{R}_{X_i^j} : L^2(\Omega) \to L^2(X_i^j)$ be the restriction operator to the subdomain $X_i^j \subset \Omega$. Then,

$$\begin{aligned} [\mathcal{R}_{X_i^j} u_{j,l}](x) &= \mathcal{R}_{X_i^j} \int_\Omega G_L(x,y) f_{j,l}(y)\,\mathrm{d}y \\ &= \sum_{i'=1}^{n_j} \int_{Y_{i'}^j} \mathcal{R}_{X_i^j} G_L(x,y) f_{j,l}(y)\,\mathrm{d}y \\ &= \int_{Y_i^j} \mathcal{R}_{X_i^j} G(x,y) f_{j,l}|_{Y_i^j}(y)\,\mathrm{d}y, \end{aligned}$$

where the final equality holds because the coloring algorithm ensures that there is only one $Y_i^j$ such that $X_i^j \times Y_i^j$ is an admissible domain. Any other domain of the form $X_i^j \times Y_{i'}^j$ for $i' \neq i$ belongs to a larger subdomain of $\Omega \times \Omega$ over which $G$ has already been recovered at a previous level. Thus, assuming exact recovery of peeling at previous hierarchical levels, $G_L$ is zero over such domains. Having isolated the action of $G_L$ on one admissible domain, we can now construct its approximation using the randomized SVD [33]. Following this method for each color

$1 \leq j \leq \chi(G_L)$ of the graph coloring algorithm, we recover an approximation for $G$ over the admissible domains satisfying the corresponding sampling constraints. This results in a total of $2(k + k_\epsilon)\chi(G_L) \leq 4 \times 6^d k_\epsilon$ input-output function pairs at each hierarchical level $1 \leq L \leq N_\epsilon$. We emphasize that one application of the randomized SVD requires $2(k + k_\epsilon)$ pairs to perform the sketching and projection steps. Finally, if $X_j \times Y_j$ is an admissible domain at level $L$, with $1 \leq j \leq N^{(L)}$, we denote by $\Omega_L^{(j)} \subset Y_j$ the associated random quasimatrix defined as

$$\Omega_L^{(j)} = \begin{bmatrix} f_{j,1}|_{Y_j} & \cdots & f_{j,k+k_\epsilon}|_{Y_j} \end{bmatrix}, \quad f_{j,l}|_{Y_j} \sim \mathcal{GP}(0, K_{Y_j}). \tag{4.23}$$

In practice, the number of random quasimatrices at level $L$ is much smaller than $N^{(L)}$, the number of admissible domains at that level.

### 4.4.1 Error analysis of the infinite-dimensional peeling algorithm

For a given target accuracy $\epsilon > 0$, we aim to reconstruct a Green's function $G$ corresponding to a self-adjoint Hilbert–Schmidt operator $H$, given in hierarchical format with $N_\epsilon \geq 1$ hierarchical levels in the domain and chromatic number bounded by $6^d$, to within relative error $\epsilon$. The reconstruction algorithm combines the randomized SVD and the peeling algorithm described in Sections 4.2.4, 4.3 and 4.4 and only relies on sketches of $H$ with random test functions. In particular, if $\mathbf{H}$ is a matrix, we compute matrix-vector products using random input vectors sampled from a multivariate Gaussian distribution with zero mean and covariance matrix $K$. On the other hand, if $H : L^2(\Omega) \to L^2(\Omega)$ is a Hilbert–Schmidt operator in infinite dimensions, one can compute low-rank approximants using the HS randomized SVD described in Section 4.4 by evaluating the operator at random

functions sampled from a Gaussian process with zero mean and the appropriate covariance kernel $K_Y$ described in Equation (4.21).

More specifically, if $G_i$ represents the restriction of $G$ to some admissible domain $I_{\alpha_i} \times I_{\beta_i}$, where $I_{\alpha_i}, I_{\beta_i} \subset [0,1]^d$, then we define $H_i$ as the HS operator corresponding to the restriction $G_i$. We employ this notation for the rest of the section. Because $G$ is a Green's function, any such $H_i$ has exponentially fast decaying singular values. That is, for $0 < \epsilon < 1$, $H_i$ has numerical rank $k \leq k_\epsilon = M \log(1/\epsilon)^{d+1}$, where $d \geq 1$ is an integer and $M > 0$ is a constant independent of $\epsilon$. By the Eckart–Young theorem, the tail of the singular values of $H_i$ satisfies the following inequality:

$$\left( \sum_{j>k} \sigma_j(H_i)^2 \right)^{1/2} \leq \epsilon \|H\|_{\mathrm{HS}}. \tag{4.24}$$

**Assumption 1** (Summary of assumptions)**.** *We summarize the assumptions on the self-adjoint Hilbert-Schmidt operator $H$ as follows for ease of reference:*

1. *$H$ is a solution operator under the same assumptions as $\mathsf{F}$ in Equation (4.4), with corresponding Green's function kernel $G$.*
2. *The domain of $G$ is hierarchically partitioned with $N_\epsilon \sim \frac{1}{2}\log_2(1/\epsilon)$ levels and chromatic number bounded by $6^d$.*
3. *For $\epsilon > 0$ sufficiently small, $H_i$ has numerical rank $k \leq k_\epsilon = M \log(1/\epsilon)^{d+1}$, where $M > 0$ is a constant.*

The following proposition gives a sufficient condition on the magnitude of the sketch perturbations to apply the perturbation estimate for the randomized SVD derived in Section 4.2.3.

**Proposition 4.4.1** (Perturbed randomized SVD error)**.** *Let the target tolerance $0 < \epsilon < 1$ be sufficiently small and $H_i$ be the restriction of $H$ to an admissible domain $X \times Y$. Suppose that sketching $H_i$ using a quasimatrix $\Omega_i$ with $k + k_\epsilon$ columns*

*sampled i.i.d. from* $\mathcal{GP}(0, K_Y)$ *returns a perturbed sample* $\tilde{Z} = H_i\Omega_i + E$*, where the norm of the perturbation quasimatrix* $E$ *satisfies* $\|E\|_{\mathrm{HS}} \leq \gamma_k^{1/2}\sqrt{k+k_\epsilon}\epsilon^{1+\delta}\|H\|_{\mathrm{HS}}$*, for some* $\delta > 0$*. Then, under the condition that* $\|\Omega_{i,1}\|_{\mathrm{HS}} \geq \lambda_1\gamma_k^{1/2}\epsilon^{\delta/2}$*, the randomized SVD constructs an approximation* $\tilde{H}_i$ *of* $H_i$ *using* $k+k_\epsilon$ *input-output pairs satisfies*

$$\|H_i - \tilde{H}_i\|_{\mathrm{HS}} = \mathcal{O}(\epsilon\sqrt{k_\epsilon/\gamma_k})\|H\|_{\mathrm{HS}},$$

*with probability* $\geq 1 - e^{-k_\epsilon}$*, where* $k \leq k_\epsilon = M\log(1/\epsilon)^{d+1}$*. Here,* $\Omega_{i,1} = V_1^*\Omega_i \in \mathbb{R}^{k\times(k+k_\epsilon)}$*, where* $V_1$ *is the quasimatrix containing the first* $k$ *right singular functions of* $H_i$*.*

*Proof.* The proof relies on the deterministic error bound in Proposition 4.2.3 for the randomized SVD with perturbed inputs. Here, one must control the decay rate of the $k$th singular value of $H_i$, denoted by $\sigma_k$, by providing a lower bound to ensure that the matrix $\mathbf{\Sigma}_1\mathbf{E}_1$ in Equation (4.13) has small norm so that one can perform a Taylor expansion of the pseudo-inverse. In the following, we argue that if the tail of the singular values of $H_i$ decays exponentially fast, then $\sigma_k$ must contribute to a fraction of the norm of $H_i$. For a given $0 < \epsilon < 1$, by combining [18, Thm. 2.8] and the Eckart–Young–Mirsky theorem for Hilbert–Schmidt operators, there exists $k \leq k_\epsilon = M\log(1/\epsilon)^{d+1}$ such that $H_i$ has numerical rank $k$. Let $\Sigma_2$ be the diagonal quasimatrix containing the singular values $\sigma_{k+1} \geq \sigma_{k+2} \geq \cdots \geq 0$ of $H_i$. We let $k \leq k_\epsilon$ be the unique integer satisfying the following two inequalities:

$$\|\Sigma_2\|_{\mathrm{HS}}^2 \leq \epsilon^2\|H\|_{\mathrm{HS}}^2 \quad \text{and} \quad \sigma_k^2 + \|\Sigma_2\|_{\mathrm{HS}}^2 > \epsilon^2\|H\|_{\mathrm{HS}}^2. \tag{4.25}$$

We aim to apply the perturbation analysis argument of Section 4.2.3 for the randomized SVD. We therefore must be able to estimate the term $\|\Sigma_1^{-1}E_1\|_{\mathrm{HS}}$, where $\Sigma_1$ is the diagonal matrix containing the $k$ largest singular values of $H_i$. We

apply the randomized SVD with a target rank of $k$ and an oversampling parameter of $k_\epsilon$, resulting in $k + k_\epsilon$ sketches of $H_i$ such that $E_1 = U_1^* E$ satisfies $\|E_1\|_{\text{HS}} \leq \|E\|_{\text{HS}} \leq \gamma_k^{1/2}\sqrt{k+k_\epsilon}\epsilon^{1+\delta}\|H\|_{\text{HS}} \leq \sqrt{2M}\gamma_k^{1/2}\log(1/\epsilon)^{\frac{d+1}{2}}\epsilon^{1+\delta}\|H\|_{\text{HS}}$. Then, we derive a lower bound on $\sigma_k$ to show that $\|\Sigma_1^{-1}E_1\|_{\text{HS}}$ converges to zero as $\epsilon$ goes to zero. We consider the following two cases:

1. If $\sigma_k \geq \epsilon\|H\|_{\text{HS}}/\sqrt{2}$, then $\|\Sigma_1^{-1}\|_{\text{HS}} \leq \sqrt{2k}/\epsilon\|H\|_{\text{HS}}$, and we have $\|\Sigma_1^{-1}E_1\|_{\text{HS}} \leq 2M\gamma_k^{1/2}\log(1/\epsilon)^{d+1}\epsilon^\delta$.
2. Otherwise, if $\sigma_k < \epsilon\|H\|_{\text{HS}}/\sqrt{2}$, then Equation (4.25) implies that

$$\sum_{j>k}\sigma_j^2 > \frac{\epsilon^2}{2}\|H\|_{\text{HS}}^2. \tag{4.26}$$

   We apply [18, Thm. 2.8] and the Eckart–Young–Mirsky theorem again with the accuracy $\epsilon/2$ to obtain a $k' \leq k_{\epsilon/2} = M\log(2/\epsilon)^{d+1} \leq 2^{d+1}k_\epsilon$ such that

$$\sum_{j>k'}\sigma_j^2 \leq \frac{\epsilon^2}{4}\|H\|_{\text{HS}}^2. \tag{4.27}$$

   We then combine Equations (4.26) and (4.27) to obtain the following lower bound on $\sigma_k$:

$$k'\sigma_k^2 \geq \sum_{j=k+1}^{k'}\sigma_j^2 = \sum_{j>k}\sigma_j^2 - \sum_{j>k'}\sigma_j^2 \geq \frac{\epsilon^2}{4}\|H\|_{\text{HS}}^2.$$

   In the end, we find that $\|\Sigma_1^{-1}E_1\|_{\text{HS}} \leq 2^{d/2+2}M^{3/2}\gamma_k^{1/2}\log(1/\epsilon)^{3(d+1)/2}\epsilon^\delta$.

In both cases, we showed that $\|\Sigma_1^{-1}E_1\|_{\text{HS}} = o(\|\Omega_{i,1}\|_{\text{HS}})$. Then, we can perform a first order expansion for $\epsilon$ sufficiently small to obtain the randomized SVD bound (see Section 4.2.3 and eq. (4.22)):

$$\|H_i - \tilde{H}_i\|_{\text{HS}} = \mathcal{O}(k_\epsilon^{1/2}\gamma_k^{-1/2}\epsilon)\|H\|_{\text{HS}},$$

which holds with probability $\geq 1 - e^{-k_\epsilon}$. We note that the error due to the projection step is negligible since we assumed this error is smaller than the target tolerance. □

Proposition 4.4.1 guarantees that one can apply the randomized SVD perturbation analysis whenever the magnitude of the perturbation is smaller than the target randomized SVD accuracy. We will now analyze the effect of the accumulation of errors in the peeling algorithm on the randomized SVD accuracy at the higher hierarchical levels. Following Proposition 4.2.5, the peeling algorithm introduces an accumulation of errors because admissible blocks from previous hierarchical levels perturb the sketches at the current level. To counterbalance this effect, we employ the randomized SVD at the first level with a higher target accuracy $\epsilon_1 = \epsilon^r$, where $r > 1$ is an exponent to be determined, and progressively decrease the target accuracy $\epsilon_L$ at hierarchical levels $1 \leq L \leq N_\epsilon$, such that $\epsilon_1 < \cdots < \epsilon_{N_\epsilon}$, to reach a relative error between the approximant and the Green's function of at most $\epsilon/\log(1/\epsilon)$ on each admissible subdomain $X \times Y \subset \Omega \times \Omega$ of the partition. Let $k_L \leq k_{\epsilon_L}$ be the target randomized SVD rank at level $1 \leq L \leq N_\epsilon$. We introduce the following covariance quality measure for the peeling algorithm:

$$\Gamma_\epsilon = \min_{1 \leq L \leq N_\epsilon} \left\{ \min_{X_L \times Y_L \text{is admissible}} \gamma_{k_L, X_L \times Y_L} \right\}. \tag{4.28}$$

We consider the following probability events, which provide bounds on the $j$th random quasimatrix at level $L$, $\Omega_L^{(j)}$, sampled from $\mathcal{GP}(0, K_{Y_j})$ (see Equation (4.23)):

$$A_\Omega \coloneqq \bigcap_{L=1}^{N_\epsilon} \bigcap_{j=1}^{N^{(L)}} \left\{ \|\Omega_L^{(j)}\|_{\text{HS}}^2 \leq 8 k_{\epsilon_1} \operatorname{tr}(K) \right\}, \qquad B_\Omega \coloneqq \bigcap_{L=1}^{N_\epsilon} \bigcap_{j=1}^{N^{(L)}} \left\{ \|\Omega_{L,1}^{(j)}\|_{\text{HS}}^2 \geq \lambda_1 \epsilon^\delta \Gamma_\epsilon \right\}, \tag{4.29}$$

where $\delta = 1/(N_\epsilon - 1)$, and $\Omega_{L,1}^{(j)}$ is the matrix containing the inner products of the first right singular vectors of $G$ restricted to the admissible domain $X \times Y$ and the random functions in $\Omega_L^{(j)}$ (see Section 4.4). We analyze the probability that $A_\Omega$ and $B_\Omega$ occur in Section 4.4.1. Then, Proposition 4.4.2 estimates the required exponent $r$ and decay rate of the target accuracy.

**Proposition 4.4.2** (Adaptive rate of the target accuracy)**.** *Let* $0 < \epsilon < 1$ *suf-*

*ficiently small and select the target accuracy for the randomized SVD at level* $1 \leq L \leq N_\epsilon$ *as follows:*

$$\epsilon_L = \mathcal{O}(k_{\epsilon_1}^{3/2}\Gamma_\epsilon^{-1}\epsilon^{-1/(N_\epsilon-1)}\epsilon_{L-1}), \quad \epsilon_1 = \epsilon^r,$$

*where* $r = (d+2)\log(\log(1/\epsilon)) + \log(1/\Gamma_\epsilon)$*, and* $k_{\epsilon_1} = M\log(1/\epsilon_1)^{d+1}$*. The factor* $\Gamma_\epsilon$ *is defined as the minimum of the covariance quality* $\gamma_{\epsilon_L}$ *over all hierarchical levels. Then, conditioning on* $A_\Omega$*,* $B_\Omega$*, and assuming that the randomized SVD succeeds on each admissible block, the error between the Green's function and its approximant* $\tilde{G}$ *returned by Algorithm 5 using the randomized SVD and the peeling algorithm satisfies*

$$\|G - \tilde{G}\|_{L^2(X\times Y)} \leq \frac{\epsilon}{\log(1/\epsilon)}\|G\|_{L^2(\Omega\times\Omega)},$$

*on each admissible domain* $X \times Y \subset \Omega \times \Omega$ *of the hierarchical partition.*

*Proof.* Let $1 \leq L \leq N_\epsilon$ be a hierarchical level and denote by $\tilde{\epsilon}_\ell$ the absolute error between the Green's function and its approximant on each of the admissible domains at the previous levels $1 \leq \ell \leq L-1$. According to Corollary 4.3.1, the approximation error of the perturbed sketches (randomized SVD sketch and projection sketch) at level $L$ are bounded by $E_L$ and $E_L^P$ as

$$\|Z_L - \tilde{Z}_L\|_{\mathrm{F}} \leq E_L, \tag{4.30a}$$

$$E_L \coloneqq \sqrt{2}(6^d - 3^d)C\sqrt{k_{\epsilon_1}}\sum_{\ell=1}^{L-1} 2^{\frac{L-1-\ell}{2}}\tilde{\epsilon}_\ell \tag{4.30b}$$

$$= \sqrt{2}E_{L-1} + \sqrt{2}(6^d - 3^d)C\sqrt{k_{\epsilon_1}}\tilde{\epsilon}_{L-1}, \tag{4.30c}$$

$$\|Z_L^P - \tilde{Z}_L^P\|_{\mathrm{F}} \leq E_L^P, \tag{4.30d}$$

$$E_L^P \coloneqq \sqrt{2}(6^d - 3^d)\sqrt{k_{\epsilon_1}}\sum_{\ell=1}^{L-1} 2^{\frac{L-1-\ell}{2}}\tilde{\epsilon}_\ell \tag{4.30e}$$

$$= \sqrt{2}E_{L-1}^P + \sqrt{2}(6^d - 3^d)\sqrt{k_{\epsilon_1}}\tilde{\epsilon}_{L-1}, \tag{4.30f}$$

where $C = 2\sqrt{2}\,\mathrm{tr}(K)^{1/2} k_{\epsilon_1}^{1/2}$ following Equation (4.29). Note that we bounded the target rank $k$ at level $\ell$ by the initial largest target rank $k_{\epsilon_1}$ since we start the peeling algorithm with the smallest target accuracy $\epsilon^r = \epsilon_1 \leq \cdots \leq \epsilon_{N_\epsilon}$. Then, Equation (4.30) yields

$$\max\{E_L, E_L^P\} \leq \sqrt{2}\max\{E_{L-1}, E_{L-1}^P\} + 4(6^d - 3^d)\,\mathrm{tr}(K)^{1/2} k_{\epsilon_1} \tilde{\epsilon}_{L-1}. \tag{4.31}$$

Now, following Proposition 4.4.1, one can apply the randomized SVD perturbation analysis whenever the sketch perturbations are asymptotically smaller than the target accuracy. That is, if $E_L$ and $E_L^P$ satisfy $\max\{E_L, E_L^P\} = \Gamma_\epsilon^{1/2}\epsilon^\delta \epsilon_L$ for some $\delta > 0$. The target randomized SVD accuracy, $\epsilon_L$, and resulting absolute approximation error, $\tilde{\epsilon}_L$, are related by the following equation (see Proposition 4.4.1):

$$\tilde{\epsilon}_L = \mathcal{O}(k_{\epsilon_L}^{1/2} \gamma_{\epsilon_L}^{-1/2} \epsilon_L) = \mathcal{O}(k_{\epsilon_1}^{1/2} \Gamma_\epsilon^{-1/2} \epsilon_L), \tag{4.32}$$

where $k_{\epsilon_L} = M\log(1/\epsilon_L)^{d+1} \leq k_{\epsilon_1}$ and $\gamma_{\epsilon_L}^{-1/2} \leq \Gamma_\epsilon^{-1/2}$. Combining Equations (4.31) and (4.32) yields

$$\epsilon_L = \mathcal{O}\left(\frac{\sqrt{2}\epsilon^\delta + 4(6^d - 3^d)\,\mathrm{tr}(K)^{1/2} k_{\epsilon_1}^{3/2} \Gamma_\epsilon^{-1}}{\epsilon^\delta} \epsilon_{L-1}\right) = \mathcal{O}(k_{\epsilon_1}^{3/2} \Gamma_\epsilon^{-1} \epsilon^{-\delta} \epsilon_{L-1}). \tag{4.33}$$

Therefore, after iterating Equation (4.33) over $1 \leq \ell \leq L$, we obtain the following estimate for the target randomized SVD accuracy at level $L$: $\epsilon_L = \mathcal{O}(k_{\epsilon_1}^{3(L-1)/2} \Gamma_\epsilon^{-(L-1)} \epsilon^{-(L-1)\delta} \epsilon_1)$. Moreover, the initial randomized SVD target rank is bounded by $k_{\epsilon_1} = M\log(1/\epsilon_1)^{d+1} = M r^{d+1}\log(1/\epsilon)^{d+1}$ since $\epsilon_1 = \epsilon^r$ by definition. Finally, with a choice of $\delta = 1/(N_\epsilon - 1)$, at the final level $N_\epsilon \sim \log(1/\epsilon)/(2\log 2)$, the absolute accuracy satisfies

$$\tilde{\epsilon}_{N_\epsilon} = \mathcal{O}\left(\epsilon^{r-1} k_{\epsilon_1}^{3N_\epsilon/2} \Gamma_\epsilon^{-N_\epsilon}\right) = \mathcal{O}\left(\epsilon^{r-1-\frac{1}{2\log 2}\left[\frac{3}{2}\log(k_{\epsilon_1}) + \log(1/\Gamma_\epsilon)\right]}\right).$$

Then, we select $r$ such that $\tilde{\epsilon}_{N_\epsilon} = \mathcal{O}(\epsilon/\log(1/\epsilon))$, *i.e.*,

$$\begin{aligned} r - 1 - \frac{3}{4\log 2}\log(M) - \frac{3}{4\log 2}(d+1)\log(r) - \\ \frac{3}{4\log 2}(d+1)\log(\log(1/\epsilon)) - \frac{1}{2\log 2}\log(1/\Gamma_\epsilon) \geq 1 + \frac{\log(\log(1/\epsilon))}{\log(1/\epsilon)}. \end{aligned} \tag{4.34}$$

We then choose $r = (d+2)\log(\log(1/\epsilon)) + \log(1/\Gamma_\epsilon)$ so that Equation (4.34) holds for $\epsilon$ sufficiently small, which concludes the proof. □

Proposition 4.4.2 estimates the randomized SVD target accuracy, $\epsilon_L$, at each hierarchical level needed to recover the Green's function to within $\epsilon/\log(1/\epsilon)$ relative error. One can then provide an upper bound on the number of input-output pairs needed by Algorithm 5 to approximate the Green's function on each admissible domain.

**Corollary 4.4.3** (Number of input-output pairs)**.** *Assume that the randomized SVD is successful on each admissible block of the Green's function up to level $N_\epsilon$. Then, the total number of input-output pairs required to approximate each off-diagonal block of the Green's function to within relative error $\epsilon/\log(1/\epsilon)$ is bounded by $\mathcal{O}(\log(1/\epsilon)^{d+2}[\log(\log(1/\epsilon)) + \log(1/\Gamma_\epsilon)]^{d+1})$ as $\epsilon \to 0$.*

*Proof.* Following Proposition 4.4.2, we choose a target rank and an oversampling parameter for the randomized SVD bounded by $k_{\epsilon_L} \leq k_{\epsilon_1} = M\log(1/\epsilon_1)^{d+1}$ at each hierarchical level. Moreover, for a given level $1 \leq L \leq N_\epsilon$, the randomized SVD requires a number of input-output pairs that is bounded by $4k_{\epsilon_L}\chi(G_L) \leq 4 \times 6^d k_{\epsilon_1}$ to approximate the Green's function on all the level-$L$ admissible subdomains (see Section 4.4). Then, the total number of input-output pairs after $N_\epsilon \sim \log(1/\epsilon)/(2\log 2)$ hierarchical levels is bounded by

$$\begin{aligned} N \leq 4 \times 6^d N_\epsilon k_{\epsilon_1} &= \mathcal{O}(r^{d+1}\log(1/\epsilon)^{d+2}) \\ &= \mathcal{O}(\log(1/\epsilon)^{d+2}[\log(\log(1/\epsilon)) + \log(1/\Gamma_\epsilon)]^{d+1}). \end{aligned}$$

□

### Probabilistic error analysis

The previous section provided an upper bound on the number of input-output pairs required by Algorithm 5 to construct an approximant to the Green's function, conditionally upon the success of the randomized SVD at each hierarchical level. In this section, we provide a lower bound on the probability of success of Algorithm 5. We begin with the following lemma (see [33, Lem. 4] for a proof), which gives a Chernoff-type bound [47] for the HS-norm of the random input functions. This controls the constant $C$ in Proposition 4.2.5 to bound the norm of the sketch error during peeling.

**Lemma 4.4.4** (Chernoff bound for Gaussian processes)**.** *Let $\Omega_{X\times Y}$ be a random quasimatrix with $\ell \geq 1$ i.i.d. columns in $L^2(Y)$ for $X \times Y \subset \Omega \times \Omega$, with each column following a Gaussian process $\mathcal{GP}(0, K_Y)$, where $K_Y$ is defined in Equation* (4.21)*. For all $s \geq 1$,*

$$\mathbb{P}\left\{\|\Omega_{X\times Y}\|_{\mathrm{HS}}^2 \leq \ell s^2 \operatorname{tr}(K)\right\} \geq 1 - \left[se^{-(s^2-1)/2}\right]^{\ell},$$

*where $\operatorname{tr}(K) = \operatorname{tr}(K_Y)$ is the sum of the eigenvalues of the kernel $K$.*

We can then bound the norm of the random quasimatrices used in the peeling algorithm of Section 4.4.1 with Lemma 4.4.4 and estimate the probability of the event $A_\Omega$ defined in Equation (4.29).

**Proposition 4.4.5** (Global upper bound of the forcing terms)**.** *Let $1 \leq L \leq N_\epsilon$ be a hierarchical level and $1 \leq j \leq N^{(L)} \leq 6^d 2^{dL}$, where $N^{(L)}$ is the number of admissible domains at level $L$. Let $\Omega_L^{(j)}$ be a random quasimatrix used at level $L$ of the peeling algorithm with i.i.d. columns defined in Section 4.4. Then,*

$$\mathbb{P}(A_\Omega) = \mathbb{P}\left(\bigcap_{L=1}^{N_\epsilon}\bigcap_{j=1}^{N^{(L)}}\left\{\|\Omega_L^{(j)}\|_{\mathrm{HS}}^2 \leq 8k_{\epsilon_1}\operatorname{tr}(K)\right\}\right) \geq 1 - \frac{1}{4}e^{-\log(1/\epsilon)^d}.$$

*Proof.* Let $k_L \leq k_{\epsilon_L}$ be the target rank at level $L$ such that the quasimatrix $\Omega_L^{(j)}$ has $k_L + k_{\epsilon_L}$ columns, where $k_{\epsilon_L} = M \log(1/\epsilon_L)^{d+1}$ is the oversampling parameter. Applying Lemma 4.4.4 for $s \geq 2$ yields

$$\mathbb{P}\left\{\|\Omega_L^{(j)}\|_{\mathrm{HS}}^2 \leq (k_L + k_{\epsilon_L}) s^2 \operatorname{tr}(K)\right\} \geq 1 - \left[s e^{-(s^2-1)/2}\right]^{k_L + k_{\epsilon_L}} \geq 1 - \left[s e^{-(s^2-1)/2}\right]^{k_\epsilon},$$

as $k_\epsilon \leq k_{\epsilon_L}$ for $1 \leq L \leq N_\epsilon$ and $\operatorname{tr}(K) = \operatorname{tr}(K_Y)$. We denote by $A_\Omega^{(s)}$ the event

$$A_\Omega^{(s)} = \bigcap_{L=1}^{N_\epsilon} \bigcap_{j=1}^{N^{(L)}} \left\{\|\Omega_L^{(j)}\|_{\mathrm{HS}}^2 \leq (k_L + k_{\epsilon_L}) s^2 \operatorname{tr}(K)\right\}.$$

We can compute a lower bound for $\mathbb{P}(A_\Omega^{(s)})$ using the independence of the random samples from the Gaussian process as

$$\begin{aligned} \mathbb{P}(A_\Omega^{(s)}) &= \prod_{L=1}^{N_\epsilon} \prod_{j=1}^{N^{(L)}} \mathbb{P}\left\{\|\Omega_L^{(j)}\|_{\mathrm{HS}}^2 \leq (k_L + k_{\epsilon_L}) s^2 \operatorname{tr}(K)\right\} \\ &\geq \left(1 - \left[s e^{-(s^2-1)/2}\right]^{k_\epsilon}\right)^{6^d 2^{d(N_\epsilon+1)}} \\ &\geq \exp\left\{6^d 2^{d(N_\epsilon+1)} \log\left(1 - e^{-k_\epsilon[(s^2-1)/2 - \log(s)]}\right)\right\}, \end{aligned} \tag{4.35}$$

where $k_\epsilon[(s^2-1)/2 - \log(s)] \to \infty$ as $\epsilon \to 0$ since $(s^2-1)/2 > \log(s)$ for $s \geq 2$. Let $s = 2$, $k_\epsilon = M \log(1/\epsilon)^{d+1}$, and define the constant $C_1 = M[3/2 - \log(2)] > 0$ such that

$$\log\left(1 - e^{-k_\epsilon[3/2 - \log(2)]}\right) = \log\left(1 - e^{-C_1 \log(1/\epsilon)^{d+1}}\right) \geq -2e^{-C_1 \log(1/\epsilon)^{d+1}}, \tag{4.36}$$

where we used the inequality $\log(1-u) \geq -u/(1-u) \geq -2u$ for $|u| < 1/2$. Moreover, since $N_\epsilon \sim \log(1/\epsilon)/(2\log 2)$ (see Equation (4.9)), there exists a constant $C_2 > 0$ such that $6^d 2^{d(N_\epsilon+1)} \leq C_2 \epsilon^{-2d}$ for sufficiently small $\epsilon$. Therefore, combining Equations (4.35) and (4.36) yields

$$\begin{aligned} \mathbb{P}(A_\Omega^{(2)}) \geq \exp\left\{-C_2 e^{-C_1 \log(1/\epsilon)^{d+1} + 2d\log(1/\epsilon)}\right\} &\geq \exp\left\{-\frac{1}{4} e^{-\log(1/\epsilon)^d}\right\} \\ &\geq 1 - \frac{1}{4} e^{-\log(1/\epsilon)^d}, \end{aligned}$$

where we used the fact that $C_2 e^{-C_1 \log(1/\epsilon)^{d+1}+2d\log(1/\epsilon)} \leq e^{-\log(1/\epsilon)^d}/4$ for sufficiently small $\epsilon$, and the inequality $e^{-u} \geq 1-u$ for $|u|<1$. We then obtain the following bound,

$$\mathbb{P}\left(\bigcap_{L=1}^{N_\epsilon}\bigcap_{j=1}^{N^{(L)}}\left\{\|\Omega_L^{(j)}\|_{\mathrm{HS}}^2 \leq 4(k_L+k_{\epsilon_L})\operatorname{tr}(K)\right\}\right) \geq 1-\frac{1}{4}e^{-\log(1/\epsilon)^d}.$$

Finally, we note that bounding $\|\Omega_L^{(i)}\|_{\mathrm{HS}}^2$ by $4(k_L+k_{\epsilon_L})\operatorname{tr}(K)$ implies the inequality $\|\Omega_L^{(i)}\|_{\mathrm{HS}}^2 \leq 8k_{\epsilon_1}\operatorname{tr}(K)$, because $k_L \leq k_{\epsilon_L} \leq k_{\epsilon_1}$, which achieves the proof. □

The next lemma gives a lower bound on the Frobenius norm of a matrix with i.i.d. columns sampled from a multivariate Gaussian distribution.

**Lemma 4.4.6** (Chernoff lower bound)**.** *Let $\mathbf{\Omega}_1 \in \mathbb{R}^{k\times\ell}$ be a random matrix with $\ell \geq k \geq 1$, where each column is sampled from a multivariate Gaussian distribution with mean zero and covariance matrix $\mathbf{C} \in \mathbb{R}^{k\times k}$. Then, for $0<c<1$ we have*

$$\mathbb{P}(\|\mathbf{\Omega}_1\|_{\mathrm{HS}}^2 \geq c\lambda_k(\mathbf{C})\ell) \geq 1-\left[e^{-\log(c)+c-1}\right]^{-\ell/2},$$

*where $\lambda_k(\mathbf{C})$ is the smallest eigenvalue of $\mathbf{C}$.*

*Proof.* Let $\mathbf{Z} = \|\mathbf{\Omega}_1\|_{\mathrm{HS}}^2 = \operatorname{tr}(\mathbf{A})$, where the scatter matrix $\mathbf{A} = \mathbf{\Omega}_1\mathbf{\Omega}_1^*$ follows the Wishart distribution $W_k(\mathbf{C},\ell)$ with scale matrix $\mathbf{C} \in \mathbb{R}^{k\times k}$. We denote the eigenvalues of $\mathbf{C}$ by $\lambda_1(\mathbf{C}) \geq \cdots \geq \lambda_k(\mathbf{C})$. Following [127, Sec. 3], the moment generating function of $\mathbf{Z}$ is

$$\mathbb{E}[e^{t\mathbf{Z}}] = \det(\mathbf{I}-2t\mathbf{C})^{-\ell/2}.$$

Let $c>0$ and $t<0$, using Markov's inequality we have

$$\begin{aligned}\mathbb{P}(\mathbf{Z} < c\lambda_k(\mathbf{C})\ell) &= \mathbb{P}(e^{t\mathbf{Z}/\lambda_k(\mathbf{C})} > e^{tc\ell}) \leq \det\left(\mathbf{I}-\frac{2t}{\lambda_k(\mathbf{C})}\mathbf{C}\right)^{-\ell/2} e^{-tc\ell} \\ &= e^{-\ell\left[\frac{1}{2}\log\det\left(\mathbf{I}-\frac{2t}{\lambda_k(\mathbf{C})}\mathbf{C}\right)+tc\right]} \\ &\leq e^{-\ell\left[\frac{k}{2}\log(1-2t)+tc\right]},\end{aligned} \tag{4.37}$$

where the last inequality comes from the following relation:

$$\log\det\left(I-\frac{2t}{\lambda_k(\mathbf{C})}\mathbf{C}\right)=\sum_{i=1}^{k}\log\left(1-2t\frac{\lambda_i(\mathbf{C})}{\lambda_k(\mathbf{C})}\right)\geq k\log(1-2t).$$

We then choose $t=1-k/c$ to minimize the right-hand side of Equation (4.37), which yields

$$\mathbb{P}(\mathbf{Z}<c\lambda_k(\mathbf{C})\ell)\leq\left[e^{\log(k/c)+(c/k-1)}\right]^{-\ell k/2}\leq\left[e^{\log(c)+1-c}\right]^{\ell/2}.$$

□

We can now estimate the probability of $B_\Omega$ in Equation (4.29) using Lemma 4.4.6.

**Proposition 4.4.7** (Global lower bound of the forcing terms)**.** *Let $1\leq L\leq N_\epsilon$ be a hierarchical level and $1\leq j\leq N^{(L)}\leq 6^d2^{dL}$, where $N^{(L)}$ is the number of admissible domains at level $L$. Let $\Omega_L^{(j)}$ be a random quasimatrix used at level $L$ of the peeling algorithm with i.i.d. columns defined in Section 4.4 and $\Omega_{L,1}^{(j)}$ be the matrix containing the inner products with right singular vectors of $G$ restricted to the corresponding admissible domain (see Section 4.4). Then,*

$$\mathbb{P}(B_\Omega)=\mathbb{P}\left(\bigcap_{L=1}^{N_\epsilon}\bigcap_{j=1}^{N^{(L)}}\left\{\|\Omega_{L,1}^{(j)}\|_{\mathrm{HS}}^2\geq\lambda_1\epsilon^\delta\Gamma_\epsilon\right\}\right)\geq 1-\frac{1}{4}e^{-\log(1/\epsilon)^d},$$

*where $\delta=1/(N_\epsilon-1)$ (see Proposition 4.4.2).*

*Proof.* Let $1\leq L\leq N_\epsilon$, $1\leq j\leq N^{(L)}$, and $\Omega_{L,1}^{(j)}\in\mathbb{R}^{k_L\times(k_L+k_{\epsilon_L})}$ be the random matrix associated with the quasimatrix $\Omega_L^{(j)}$. Let $0<c<1$, following Lemma 4.4.6, we have

$$\mathbb{P}\left(\|\Omega_{L,1}^{(j)}\|_{\mathrm{HS}}^2\geq c\lambda_1\Gamma_\epsilon\right)\geq 1-e^{k_\epsilon[\log(c)-c+1]/2},$$

since $\lambda_{k_L}(\mathbf{C})(k_L + k_{\epsilon_L}) \geq \lambda_1 \Gamma_\epsilon$ and $k_L + k_{\epsilon_L} \geq k_\epsilon$. Therefore, choosing $c = \epsilon^\delta$ yields

$$\begin{aligned}
\mathbb{P}(B_\Omega) &= \mathbb{P}\left( \bigcap_{L=1}^{N_\epsilon} \bigcap_{j=1}^{N^{(L)}} \left\{ \|\Omega_{L,1}^{(j)}\|_{\text{HS}}^2 \geq \lambda_1 \epsilon^\delta \Gamma_\epsilon \right\} \right) \\
&\geq \left(1 - e^{k_\epsilon[-\delta \log(1/\epsilon) - \epsilon^\delta + 1]/2}\right)^{6^d 2^{d(N_\epsilon+1)}} \\
&\geq \exp\left\{ 6^d 2^{d(N_\epsilon+1)} \log\left(1 - e^{k_\epsilon[-\delta \log(1/\epsilon) - \epsilon^\delta + 1]/2}\right) \right\} \\
&\geq \exp\left\{ -6^d 2^{d(N_\epsilon+1)+1} e^{k_\epsilon[-\delta \log(1/\epsilon) - \epsilon^\delta + 1]/2} \right\},
\end{aligned}$$

where we used the inequality $\log(1-u) \geq -2u$ for $|u| < 1/2$. Let $C_2 > 0$ be a constant such that $6^d 2^{d(N_\epsilon+1)} \leq C_2 \epsilon^{-2d}$ for sufficiently small $\epsilon$. Then,

$$\begin{aligned}
\mathbb{P}(B_\Omega) &\geq \exp\left\{ -2C_2 e^{2d \log(1/\epsilon) - M \log(1/\epsilon)^{d+1}(\delta \log(1/\epsilon) + \epsilon^\delta - 1)} \right\} \\
&\geq \exp\left\{ -\frac{1}{4} e^{-\log(1/\epsilon)^d} \right\} \geq 1 - \frac{1}{4} e^{-\log(1/\epsilon)^d},
\end{aligned}$$

where we used the fact that $2C_2 e^{2d \log(1/\epsilon) - M \log(1/\epsilon)^{d+1}(\delta \log(1/\epsilon) + \epsilon^\delta - 1)} \leq e^{-\log(1/\epsilon)^d}/4$ for sufficiently small $\epsilon$, and the inequality $e^{-u} \geq 1 - u$ for $|u| < 1$. □

We combine Proposition 4.4.5 and the probability bounds for the Hilbert–Schmidt randomized SVD (cf. Section 4.4) to obtain a global probability bound for the algorithm that uses the peeling procedure with the randomized SVD.

**Proposition 4.4.8** (Probabilistic bound)**.** *Let $0 < \epsilon < 1$ be sufficiently small. There is a randomized algorithm that constructs an approximation to the Green's function using $N = \mathcal{O}(\log(1/\epsilon)^{d+2}[\log(\log(1/\epsilon)) + \log(1/\Gamma_\epsilon)]^{d+1})$ input-output pairs $\{(f_j, u_j)\}_{j=1}^N$ such that*

$$\|G - \tilde{G}\|_{L^2(X \times Y)} \leq \frac{\epsilon}{\log(1/\epsilon)} \|G\|_{L^2(\Omega \times \Omega)}, \tag{4.38}$$

*holds on all admissible domains $X \times Y$ in the hierarchical partition of $G$ with $N_\epsilon$ levels with probability $\geq 1 - e^{-\log(1/\epsilon)^d}$.*

*Proof.* We first note that Equation (4.38) holds if the randomized SVD is successful at all the admissible blocks of the hierarchical partition of the Green's function. Therefore, if $\mathcal{E}_\epsilon$ denotes the following event:

$$\mathcal{E}_\epsilon = \{\|G - \tilde{G}\|_{L^2(X\times Y)} \leq \frac{\epsilon}{\log(1/\epsilon)}\|G\|_{L^2(\Omega\times\Omega)}, \text{ for all admissible domains } X \times Y\},$$

then we have $\mathbb{P}(\mathcal{E}_\epsilon) \geq \mathbb{P}(\cap_{L=1}^{N_\epsilon} A_L)$, where $A_L$ is the event that all the applications of the randomized SVD are successful for the admissible domains in the level $1 \leq L \leq N_\epsilon$ of the hierarchical partition. Since the probability of success of the randomized SVD depends on the norm of the random input vectors due to the peeling procedure, we condition on the event $A_\Omega \cap B_\Omega$, where again $A_\Omega$ and $B_\Omega$ are defined in Equation (4.29), to obtain

$$\mathbb{P}(\mathcal{E}_\epsilon) \geq \mathbb{P}\left(\bigcap_{L=1}^{N_\epsilon} A_L \mid A_\Omega \cap B_\Omega\right)\mathbb{P}(A_\Omega \cap B_\Omega). \tag{4.39}$$

By Propositions 4.4.5 and 4.4.7, we have the inequalities $\mathbb{P}(A_\Omega) \geq 1 - e^{-\log(1/\epsilon)^d}/4$ and $\mathbb{P}(B_\Omega) \geq 1 - e^{-\log(1/\epsilon)^d}/4$. We therefore focus on deriving a lower bound for $\mathbb{P}(\cap_{L=1}^{N_\epsilon} A_L \mid A_\Omega \cap B_\Omega)$.

Let $2 \leq L \leq N_\epsilon$ be a hierarchical level. As in Section 4.4.1, due to the peeling procedure and accumulation of the perturbation errors, the probability of success of the randomized SVD bound given by Proposition 4.4.1 on the admissible blocks at level $L$ depends on the success of the randomized SVDs at all previous levels $1 \leq \ell \leq L-1$. If $p_L = \mathbb{P}(\cap_{\ell=1}^{L} A_\ell \mid A_\Omega \cap B_\Omega)$ denotes the probability of success of the randomized SVD at each level up to $L$ given $A_\Omega \cap B_\Omega$, we can estimate $p_L$

using conditional probability as follows:

$$p_L = \mathbb{P}\left(\bigcap_{\ell=1}^{L} A_\ell \mid A_\Omega \cap B_\Omega\right) = \mathbb{P}\left(A_L \mid \bigcap_{\ell=1}^{L-1} A_\ell, A_\Omega \cap B_\Omega\right) \mathbb{P}\left(\bigcap_{\ell=1}^{L-1} A_\ell \mid A_\Omega \cap B_\Omega\right) \tag{4.40}$$

$$= \mathbb{P}\left(A_L \mid \bigcap_{\ell=1}^{L-1} A_\ell, A_\Omega \cap B_\Omega\right) p_{L-1}, \tag{4.41}$$

where $p_1 = \mathbb{P}(A_1 \mid A_\Omega \cap B_\Omega)$. We can now use the HS randomized SVD probability bound (see Equation (4.22)) to derive a lower bound for $p_L$. First, denote by $N^{(L)}$ the number of admissible domains at level $L$, and consider the event $B_L^i$ that the approximation of $G$ obtained by the randomized SVD on the $i$th admissible domain $X \times Y$ with target accuracy $\epsilon_L > 0$ satisfies Equation (4.22), *i.e.*,

$$B_L^i = \left\{ \|G - \tilde{G}\|_{L^2(X\times Y)} \leq \mathcal{O}\left(k_{\epsilon_L}^{1/2} \Gamma_\epsilon^{-1/2} \epsilon_L\right) \|G\|_{L^2(\Omega\times\Omega)} \right\}.$$

This is an application of the randomized SVD with a target rank of $k_L \leq k_{\epsilon_L}$ and an oversampling parameter of $k_{\epsilon_L} \geq k_\epsilon$, where $k_\epsilon = M \log(1/\epsilon)^{d+1}$. Therefore, using the HS randomized SVD probability bound (see Equation (4.22)), we have

$$\mathbb{P}\left(B_L^i \mid \bigcap_{\ell=1}^{L-1} A_\ell, A_\Omega \cap B_\Omega\right) \geq 1 - e^{-k_\epsilon}. \tag{4.42}$$

Moreover, combining Equations (4.40) and (4.42), and using Boole's inequality for the union of events yields

$$\begin{aligned} p_L &= \mathbb{P}\left(\bigcap_{i=1}^{N^{(L)}} B_L^i \mid \bigcap_{\ell=1}^{L-1} A_\ell, A_\Omega \cap B_\Omega\right) p_{L-1} \\ &= \left(1 - \mathbb{P}\left(\bigcup_{i=1}^{N^{(L)}} \bar{B}_L^i \mid \bigcap_{\ell=1}^{L-1} A_\ell, A_\Omega \cap B_\Omega\right)\right) p_{L-1} \\ &\geq \left(1 - \sum_{i=1}^{N^{(L)}} \mathbb{P}\left(\bar{B}_L^i \mid \bigcap_{\ell=1}^{L-1} A_\ell, A_\Omega \cap B_\Omega\right)\right) p_{L-1} \\ &\geq (1 - N^{(L)} e^{-k_\epsilon}) p_{L-1}, \end{aligned} \tag{4.43}$$

where $\bar{B}_L^i$ denotes the complementary event of $B_L^i$ such that $\mathbb{P}(\bar{B}_L^i \mid \cap_{\ell=1}^{L-1} A_\ell, A_\Omega \cap B_\Omega) \leq e^{-k_\epsilon}$, and $p_1 \geq 1 - N^{(1)} e^{-k_\epsilon}$ by Equation (4.42). Iterating Equation (4.43) over $1 \leq L \leq N_\epsilon$ gives the following probability bound:

$$p_{N_\epsilon} \geq \prod_{L=1}^{N_\epsilon} (1 - N^{(L)} e^{-k_\epsilon}), \tag{4.44}$$

where the number of admissible domains at level $L$ is bounded by $N^{(L)} \leq 6^d 2^{dL}$. Therefore, we take the logarithm of Equation (4.44) to determine a lower bound on $\log(p_{N_\epsilon})$,

$$\begin{aligned}
\log(p_{N_\epsilon}) &\geq \sum_{L=1}^{N_\epsilon} \log(1 - N^{(L)} e^{-k_\epsilon}) \\
&= -\sum_{L=1}^{N_\epsilon} \sum_{i=1}^{\infty} \frac{N^{(L)i} e^{-ik_\epsilon}}{i} \\
&\geq -\sum_{i=1}^{\infty} \frac{6^{di} e^{-ik_\epsilon}}{i} \sum_{L=1}^{N_\epsilon} 2^{idL} = -\sum_{i=1}^{\infty} \frac{6^{di} e^{-ik_\epsilon}}{i} \frac{2^{id(N_\epsilon+1)} - 1}{2^{id} - 1} \\
&\geq -\sum_{i=1}^{\infty} \frac{6^{di} 2^{id(N_\epsilon+1)} e^{-ik_\epsilon}}{i} = \log(1 - 6^d 2^{d(N_\epsilon+1)} e^{-k_\epsilon}).
\end{aligned} \tag{4.45}$$

Here, we used the Taylor series of $\log(1-u)$ for $|u| < 1$. Moreover, for sufficiently small $\epsilon$, we have $6^d 2^{d(N_\epsilon+1)} e^{-k_\epsilon} \leq C_1 \epsilon^{-d} e^{-M \log(1/\epsilon)^{d+1}}$ for some constant $C_1 > 0$. Finally, taking the exponential of Equation (4.45) gives,

$$p_{N_\epsilon} \geq 1 - C_1 \epsilon^{-d} e^{-M \log(1/\epsilon)^{d+1}} = 1 - C_1 e^{-\log(1/\epsilon)[M \log(1/\epsilon)^d - d]} \geq 1 - \frac{1}{2} e^{-\log(1/\epsilon)^d},$$

for $\epsilon$ sufficiently small. We conclude with Equation (4.39) as

$$\mathbb{P}(\mathcal{E}_\epsilon) \geq p_{N_\epsilon} \mathbb{P}(A_\Omega \cap B_\Omega) \geq \left(1 - \frac{1}{2} e^{-\log(1/\epsilon)^d}\right)^2 \geq 1 - e^{-\log(1/\epsilon)^d},$$

where we used $\mathbb{P}(A_\Omega \cap B_\Omega) \geq \mathbb{P}(A_\Omega) + \mathbb{P}(B_\Omega) - 1$. □

We note that the bound given by Equation (4.38) on the probability of failure of the algorithm decays super-algebraically with respect to the accuracy $\epsilon$ and is illustrated in Fig. 2 of the main text.

### 4.4.2 Global error in the spectral norm

Let $\mathsf{F} : L^2(\Omega) \to L^2(\Omega)$ be the Hilbert–Schmidt integral operator associated with the Green's function of the elliptic operator $\mathcal{L}$ in three dimensions such that

$$\mathsf{F}(f)[x] = \int_\Omega G(x, y) f(y) \, \mathrm{d}y, \quad f \in L^2(\Omega), \, x \in \Omega.$$

Previous works [33, 33] have expressed the error between the learned and exact Green's functions either in the $L^2$-norm or $L^1$-norm by exploiting the connection between the Hilbert–Schmidt norm of the integral operator $\mathsf{F}$ and the $L^2$-norm of the Green's function, since $\|\mathsf{F}\|_{\mathrm{HS}} = \|G\|_{L^2(\Omega\times\Omega)}$. As observed in Section 4.4.1, the peeling algorithm employed in this work introduces a perturbation error on the sketch of the Green's function that depends on the previous hierarchical levels. In particular, relative errors on admissible domains from the past hierarchical levels add up during the procedure. Hence, the algorithm introduced in Section 4.2.2, which is based on peeling and the randomized SVD, only guarantees a relative error of $\log(1/\epsilon)^{-1}\epsilon$ on each admissible domain $X \times Y \subset \Omega \times \Omega$, *i.e.*,

$$\|G - \tilde{G}\|_{L^2(X\times Y)} \leq \log(1/\epsilon)^{-1}\epsilon \|G\|_{L^2(\Omega\times\Omega)}. \tag{4.46}$$

However, since the number of admissible domains increases exponentially with the hierarchical levels $N_\epsilon$, measuring the error in the $L^2$-norm on the entire domain would significantly deteriorate the error bound, and therefore the number of input-output pairs required to approximate $G$ within $\epsilon$. To circumvent this issue, we exploit the properties of the operator norm of Hilbert–Schmidt operators. In this section, we prove the following proposition, which provides an error bound expressed in the operator norm for the Green's function approximated by Algorithm 5. The proof is deferred to the end of the section.

**Proposition 4.4.9** (Approximation error in the HS operator norm)**.** *Let* $\mathsf{F}$ :

*$L^2(\Omega) \to L^2(\Omega)$ be the Hilbert–Schmidt operator associated with the Green's function $G$ of the elliptic operator $\mathcal{L}$ in three dimensions. Let $\tilde{G} : \Omega \times \Omega \to \mathbb{R}$ be the kernel constructed by Algorithm 5 and $\tilde{\mathsf{F}}$ its associated Hilbert–Schmidt integral operator, defined as*

$$\tilde{\mathsf{F}}(f)[x] = \int_\Omega \tilde{G}(x,y) f(y)\,\mathrm{d}y, \quad f \in L^2(\Omega),\, x \in \Omega.$$

*Then, $\|\mathsf{F} - \tilde{\mathsf{F}}\|_2 \leq \epsilon \|\mathsf{F}\|_{\mathrm{HS}}$.*

We remark that Proposition 4.4.9 implies that the following equation holds for any $f \in L^2(\Omega)$:

$$\left\| \int_\Omega (G - \tilde{G}) f(y)\,\mathrm{d}y \right\|_{L^2(\Omega)} \leq \epsilon \|G\|_{L^2(\Omega\times\Omega)} \|f\|_{L^2(\Omega)}.$$

Therefore, obtaining a relative error in the operator norm between the exact and learned Green's functions guarantees that the range of the Hilbert–Schmidt operator $\mathsf{F}$, *i.e.*, the solution operator associated with the partial differential operator $\mathcal{L}$, is well approximated. Additionally, expressing the error in the operator norm aligns with the methodology employed by current machine learning techniques for learning solution operators associated with PDEs, such as DeepONets [113, 172] and neural operators [99, 108]. In particular, they measure the approximation error of the solution operator on a set of testing pairs. This is equivalent to estimating the error in the spectral norm on the finite dimensional subspace spanned by the $f_j$. We emphasize that measuring the approximation error in the spectral norm is relevant to the setup of state-of-the-art deep learning techniques. We begin by recalling a property for the maximum singular value of an HS operator (analogous to block diagonal matrices).

**Lemma 4.4.10** (Operator norm of block diagonal HS operators)**.** *Let $G_1$, $G_2 : \Omega\times\Omega \to \mathbb{R}$ be two kernels such that Supp$(G_1) \subset D_1 \times D_2$ and Supp$(G_2) \subset D_3 \times D_4$,*

*where* $D_1 \cap D_3 = \emptyset$ *and* $D_2 \cap D_4 = \emptyset$. *Let* $\mathsf{F}$, $\mathsf{F}_1$, *and* $\mathsf{F}_2$ *be the Hilbert–Schmidt integral operators associated with the kernels* $G_1 + G_2$, $G_1$, *and* $G_2$, *respectively. Then,* $\|\mathsf{F}\|_2 = \max\{\|\mathsf{F}_1\|_2, \|\mathsf{F}_2\|_2\}$.

*Proof.* We start the proof by showing that the first inequality $\|\mathsf{F}\|_2 \geq \max\{\|\mathsf{F}_1\|_2, \|\mathsf{F}_2\|_2\}$ holds:

$$\begin{aligned}\|\mathsf{F}\|_2 = \sup_{\|f\|_{L^2(\Omega)}=1} \|\mathsf{F}(f)\|_{L^2(\Omega)} &\geq \sup_{\substack{\|f\|_{L^2(\Omega)}=1,\\ \mathrm{Supp}(f)\subset D_2}} \|\mathsf{F}(f)\|_{L^2(\Omega)} \\ &= \sup_{\|f\|_{L^2(D_2)}=1} \|\mathsf{F}_1(f)\|_{L^2(D_1)} \\ &= \|\mathsf{F}_1\|_2,\end{aligned}$$

where we identified the operator $\mathsf{F}_1$ with the operator defined on $D_1 \times D_2$, since $\mathrm{Supp}(G_1) \subset D_1 \times D_2$. The same argument applies to $\mathsf{F}_2$ so that $\|\mathsf{F}\|_2 \geq \max\{\|\mathsf{F}_1\|_2, \|\mathsf{F}_2\|_2\}$. For the other inequality, let $f \in L^2(\Omega)$ such that $\|f\|_{L^2(\Omega)} = 1$, and define $f_1 = f|_{D_2}$ and $f_2 = f|_{D_4}$ as the restrictions of $f$ to the domains $D_2$ and $D_4$, respectively. Then,

$$\begin{aligned}\|f_1\|^2_{L^2(D_2)} + \|f_2\|^2_{L^2(D_4)} &= \int_{D_2} |f_1(x)|^2 \,\mathrm{d}x + \int_{D_4} |f_2(x)|^2 \,\mathrm{d}x \\ &\leq \int_{\Omega} |f(x)|^2 \,\mathrm{d}x \\ &= \|f\|^2_{L^2(\Omega)} = 1.\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\|\mathsf{F}(f)\|_{L^2(\Omega)}^2 &= \left\| \int_\Omega (G_1 + G_2)(\cdot, y) f(y)\, \mathrm{d}y \right\|_{L^2(\Omega)}^2 \\
&= \left\| \int_{D_2} G_1(\cdot, y) f_1(y)\, \mathrm{d}y + \int_{D_4} G_2(\cdot, y) f_2(y)\, \mathrm{d}y \right\|_{L^2(\Omega)}^2 \\
&= \left\| \int_{D_2} G_1(\cdot, y) f_1(y)\, \mathrm{d}y \right\|_{L^2(D_1)}^2 + \left\| \int_{D_4} G_2(\cdot, y) f_2(y)\, \mathrm{d}y \right\|_{L^2(D_3)}^2 \\
&= \|\mathsf{F}_1(f_1)\|_{L^2(D_1)}^2 + \|\mathsf{F}_2(f_2)\|_{L^2(D_3)}^2 \\
&\leq \|\mathsf{F}_1\|_2^2 \|f_1\|_{L^2(D_2)}^2 + \|\mathsf{F}_2\|_2^2 \|f_2\|_{L^2(D_4)}^2.
\end{aligned}$$

Taking the supremum of both sides over all possible $f \in L^2(\Omega)$ yields the second inequality as

$$\|\mathsf{F}\|_2^2 \leq \max\{\|\mathsf{F}_1\|_2^2, \|\mathsf{F}_2\|_2^2\}(\|f_1\|_{L^2(D_2)}^2 + \|f_2\|_{L^2(D_4)}^2) \leq \max\{\|\mathsf{F}_1\|_2^2, \|\mathsf{F}_2\|_2^2\}.$$

□

Lemma 4.4.10 immediately generalizes to an arbitrary sum of Hilbert–Schmidt integral operators with disjoint kernel supports.

**Corollary 4.4.11** (Spectral norm of sums of HS operators)**.** *Let $n \geq 1$ and $G_1, \ldots, G_n : \Omega \times \Omega \to \mathbb{R}$ be $n$ kernels such that Supp$(G_1) \subset C_1 \times D_1$, …, Supp$(G_1) \subset C_n \times D_n$, where $D_i \cap D_j = \emptyset$ and $C_i \cap C_j = \emptyset$ for $1 \leq i \neq j \leq n$. Let $\mathsf{F} : L^2(\Omega) \to L^2(\Omega)$ be the Hilbert–Schmidt integral operator associated with the kernel $\sum_{i=1}^n G_i$, and $\mathsf{F}_i$ be the Hilbert–Schmidt integral operator associated with the kernel $G_i$ for $1 \leq i \leq n$. Then, $\|\mathsf{F}\|_2 = \max_{1 \leq i \leq n} \|\mathsf{F}_i\|_2$.*

*Proof.* The proof follows directly from Lemma 4.4.10 by induction. In particular, we consider the HS integral operators associated with the two kernels $H_1 = \sum_{i=1}^{n-1} G_i$ and $H_2 = G_n$ and apply Lemma 4.4.10. □

We are now ready to prove Proposition 4.4.9.

*Proof of Proposition 4.4.9.* Let $\tilde{G} : \Omega \times \Omega \to \mathbb{R}$ be the kernel constructed by Algorithm 5 such that the error with the Green's function $G$ satisfies

$$\|G - \tilde{G}\|_{L^2(X\times Y)} = \begin{cases} \log(1/\epsilon)^{-1}\epsilon\|G\|_{L^2(\Omega\times\Omega)}, & \text{if } X \times Y \text{ is an admissible domain,} \\ \epsilon\|G\|_{L^2(\Omega\times\Omega)}, & \text{otherwise.} \end{cases}$$

We apply Corollary 4.4.11 to compute an upper bound on the operator norm of the HS operator associated with the kernel $G - \tilde{G}$. The procedure consists of decomposing the kernel into a sum of kernels over the number of hierarchical levels and colored admissible domains (see Section 4.3), such that each element in the sum is a block diagonal kernel whose norm can be estimated by Corollary 4.4.11. As an illustration, we assume for simplicity that the kernel $H = G - \tilde{G}$ is written in the weakly admissible hierarchical form with $L = 2$ levels as

$$H = \left(\begin{array}{cc|cc} H_{11} & H_{12} & \multicolumn{2}{c}{\multirow{2}{*}{$H_2$}} \\ H_{12}^\top & H_{14} & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$H_2^\top$}} & H_{41} & H_{42} \\ & & H_{42}^\top & H_{44} \end{array}\right),$$

where the green blocks have HS-norm bounded by $\log(1/\epsilon)^{-1}\epsilon\|G\|_{\mathrm{HS}}$ while the red blocks have HS-norm bounded by $\epsilon\|G\|_{\mathrm{HS}}$. Then, we can estimate the operator

norm of the associated Hilbert–Schmidt operator by decomposing $H$ as

$$H = \left(\begin{array}{cc|cc} 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{$H_2$}} \\ 0 & 0 & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$H_2^\top$}} & 0 & 0 \\ & & 0 & 0 \end{array}\right) + \left(\begin{array}{cc|cc} 0 & H_{12} & \multicolumn{2}{c}{\multirow{2}{*}{$0$}} \\ H_{12}^\top & 0 & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$0$}} & 0 & H_{42} \\ & & H_{42}^\top & 0 \end{array}\right)$$
$$+ \left(\begin{array}{cc|cc} H_{11} & 0 & \multicolumn{2}{c}{\multirow{2}{*}{$0$}} \\ 0 & H_{14} & & \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$0$}} & H_{41} & 0 \\ & & 0 & H_{44} \end{array}\right).$$

Hence, the norm of the individual components in the sum are bounded by the norm of each block following Corollary 4.4.11, and the norm of the HS operator associated with $H$ is the maximum of the norms of the components. Therefore, in this simple example, we have

$$\|H\|_2 \leq \|H_2\|_2 + \max\{\|H_{12}\|_2, \|H_{42}\|_2\} + \max\{\|H_{11}\|_2, \|H_{14}\|_2, \|H_{41}\|_2, \|H_{44}\|_2\}.$$

In the strongly admissible case with $N_\epsilon \sim \log(1/\epsilon)/(2\log 2)$ levels, we decompose the operator $\mathscr{H} \coloneqq \mathsf{F} - \tilde{\mathsf{F}}$ as

$$\mathscr{H} = \mathscr{H}_{\text{diag}} + \sum_{L=1}^{N_\epsilon} \sum_{j=1}^{\chi(G_L)} \sum_{i=1}^{n_j} \mathscr{H}|_{X_i^{j,(L)} \times Y_i^{j,(L)}},$$

where $\mathscr{H}_{\text{diag}}$ denote the non-admissible part of the operator $\mathscr{H}$ and $X_i^{j,(L)} \times Y_i^{j,(L)}$ is an admissible domain at level $L$ colored by $1 \leq j \leq \chi(G_L)$. By construction, for a fixed $1 \leq L \leq N_\epsilon$ and $1 \leq j \leq \chi(G_L)$, the restrictions of the kernel of $\mathscr{H}$ to the domains $\{X_i^{j,(L)} \times Y_i^{j,(L)}\}_{1 \leq i \leq n_j}$ have disjoint support. Hence, applying Corollary 4.4.11 and using the property that the operator norm is bounded by the

HS-norm yield,

$$\begin{aligned}\|\mathscr{H}\|_2 &\leq \|\mathscr{H}_{\text{diag}}\|_2 + \sum_{L=1}^{N_\epsilon}\sum_{j=1}^{\chi(G_L)} \max_{1\leq i\leq n_j}\{\|\mathscr{H}|_{X_i^{j,(L)}\times Y_i^{j,(L)}}\|_2\} \\ &\leq 6^d\epsilon\|\mathsf{F}\|_{\text{HS}} + 6^d\epsilon\|\mathsf{F}\|_{\text{HS}} \\ &\leq 2\times 6^d\epsilon\|\mathsf{F}\|_{\text{HS}}.\end{aligned}$$

Here, we used the fact that $N_\epsilon \leq \log(1/\epsilon)$ for sufficiently small $\epsilon$, and $\chi(G_L) \leq 6^d$. Finally, without loss of generality, we can rescale the target accuracy $\epsilon$ in Algorithm 5 to $\tilde{\epsilon} \coloneqq \epsilon/(2\times 6^d)$ by increasing the number of input-output pairs by a constant factor, which concludes the proof. □

## Methods

In this section, we describe the deep learning experiments used to generate Fig. 1 of the main text. We compare the performance of three neural network architectures, namely DeepONet (DON) [113], Fourier Neural Operator (FNO) [108], and Green's function learning (GreenLearning) [30], at approximating the solution operator associated with the two-dimensional Poisson equation defined on the domain $\Omega = [0,1]^2$ with homogeneous Dirichlet boundary conditions:

$$-\nabla^2 u = f, \qquad u|_{\partial\Omega} = 0. \tag{4.47}$$

These methods learn an approximant $\tilde{\mathsf{F}}$ to the solution operator $\mathsf{F}$ associated with Equation (4.47), represented by a neural network (NN). Our main motivation is to study the behavior of these PDE learning techniques as the number of training data varies. Hence, we want to understand why specific neural network approaches are data-efficient and others are not. We reproduce the setup of [114] that performs a fair comparison between FNO and DON and provide

details of the experiments below. The code is publicly available on GitHub at `https://github.com/NBoulle/pde-learning`.

**Neural networks architecture.** The FNO architecture is a succession of 4 Fourier layers, which perform convolutions in the Fourier space using the fast Fourier transform (FFT), with the ReLU activation functions. The DON employed in this work has a much larger number of trainable parameters and consists of a product of a "branch network", which is a convolution NN with two 2D convolution layers, and a "trunk network", which is a standard fully connected NN with 4 hidden layers and 128 neurons per layer. GreenLearning enforces prior knowledge of the solution operator associated with Equation (4.47) by approximating directly the associated Green's function. Solutions to Equation (4.47) are predicted by integrating the NN representation of the Green's function against the forcing terms using a trapezoidal rule. We employ a rational neural network [34] with 4 hidden layers and 50 neurons per layer. Rational NNs offer theoretical and practical advantages over ReLU NNs, as they can capture the singularities of Green's function located along the diagonal [34, 30]. DON is implemented in TensorFlow [1] using the DeepXDE library [115], while FNO and GreenLearning are implemented in PyTorch [140]. The deep learning experiments were performed on a workstation with a GPU (NVIDIA GeForce RTX 3080 Ti).

**Training and testing datasets.** We train the three neural networks on pairs of training data $\{(f_j, u_j)\}_{j=1}^{N}$, where we vary $N$ between 2 and 1000. In Fig. 1 of the main text, we report the resulting approximation error $\|\mathsf{F} - \tilde{\mathsf{F}}\|_2$ for $N \leq 200$ to visualize the exponential convergence rate before the plateau due to discretization errors. Here, $\|\mathsf{F} - \tilde{\mathsf{F}}\|_2$ is approximated by the mean relative error on

the testing dataset, which consists of $N_{\text{test}} = 200$ pairs of input-output data $\{(f_j^{(\text{test})}, u_j^{(\text{test})})\}_{j=1}^{N_{\text{test}}}$, as

$$\|\mathsf{F} - \tilde{\mathsf{F}}\|_2 \approx \frac{1}{N_{\text{test}}} \sum_{j=1}^{N_{\text{test}}} \frac{\|u_j^{(\text{test})} - \tilde{F}(f_j^{(\text{test})})\|_{L^2([0,1]^2)}}{\|u_j^{(\text{test})}\|_{L^2([0,1]^2)}}.$$

For consistency, we train the different networks ten times and report the mean and standard deviation of the testing errors in Fig. 1. Following [114, 108], the forcing terms are sampled from the Gaussian process $\mathcal{GP}(0, (-\Delta + 9I)^{-2})$ with zero Neumann boundary conditions on the Laplacian. In particular, we sample 1200 forcing terms from the Gaussian process and obtain the associated solutions by solving numerically the Poisson equation. Then, we perform a training/testing split of 1000/200. When we vary the size of the training dataset, we randomly select a portion of the 1000 training pairs to train the models and evaluate them on the 200 testing pairs to avoid train/test contamination. Equation (4.47) is solved on a $421 \times 421$ uniform grid, which is then downsampled to a $29 \times 29$ grid when training and testing the NNs.

CHAPTER 5

# TRANSPOSE-FREE LINEAR ALGEBRA MOTIVATED BY ADJOINT-FREE OPERATOR LEARNING

This chapter[1] centers on transpose-free numerical linear algebra, which has several motivations, the first of which is adjoint-free PDE learning. In the context of linear PDEs, the adjoint operator is essentially a dual operator that can sometimes be interpreted as the operator that arises when changing the direction of time or reversing the direction of space in the original PDE. The adjoint operator arises frequently in linear sensitivity analysis. In practice, acquiring data from the adjoint operator can be impossible when the underlying PDE is unknown. From a theoretical point-of-view, previous studies mostly focused on learning the solution operator of self-adjoint elliptic PDEs [35, 32] in divergence form defined as

$$Lu \coloneqq -\mathrm{div}(\mathbf{A}(x)\nabla u) = f, \quad x \in \Omega \subset \mathbb{R}^d, \tag{5.1}$$

where the coefficient matrix $\mathbf{A}$ is symmetric and satisfies the uniform ellipticity condition. However, non-self-adjoint PDEs arise naturally when considering time-dependent problems such as the heat equation or advection-diffusion equation, i.e., $Lu = -\mathrm{div}(\mathbf{A}(x)\,\nabla\, u) + \mathbf{c}(x)\cdot\nabla\, u$. In this context, it seems essential to require a solver for the adjoint to obtain information about the left and right singular functions of the PDE [33]. Curiously, there is a lack of emphasis in practical works on the need for adjoint equation solvers, as many methods seem to succeed without them. To understand the phenomenon of adjoint-free operator learning, we turn

[1]This chapter contains an excerpt from [31] that describes bounds for transpose-free low-rank approximation. I contributed ideas and proofs for these results, working closely with Nicolas Boullé, Sam Otto, and Alex Townsend. Sam Otto derived much of the adjoint-free operator learning theory, which is not included in this thesis to keep the focus on numerical linear algebra. Nicolas Boullé performed the numerical experiments. This chapter also contains a result on the necessity of the transpose action for least squares problems, which have not yet been published and was developed in collaboration with Michiel Hochstenbach.

to the discrete version of the problem, which boils down to what information can be recovered about a matrix from only forward matrix-vector products.

A second motivation for transpose-free linear algebra arises from the fact that in many settings, it is not natural to access matrix-vector products with the transpose. For example, in ordinary differential equations, a matrix may be an approximated Jacobian from finite differences [43]. In other cases, a particular storage format for a matrix, like a sparse format, may lend itself better to multiplication on one side than on the other. In compressed sensing, physical reasons allow one to sample linear combinations of columns but not rows [158]. Finally, in inverse problems arising in imaging, the operators $A$ and $A^\top$ are implemented in two different packages; see, e.g., [87] for more details on the unmatched transpose.

In the following sections, we consider transpose-free low-rank recovery and least squares problems. First, we show that low-rank approximation is fundamentally limited when one does not have access to the transpose, and we quantify the extent to which this holds when one knows in advance that the matrix is close to symmetric. Then, we demonstrate one cannot solve a least squares problem using only sketches from the right by showing that the problem is not uniquely determined by this information.

## 5.1 Transpose-free low-rank recovery

Let $\mathcal{A} : \mathcal{H} \to \mathcal{H}'$ be an operator between Hilbert spaces. Suppose that one can only access $\mathcal{A}$ via the forward and adjoint queries $f \mapsto \mathcal{A}f$ and $g \mapsto \mathcal{A}^\top g$, where $f \in \mathcal{H}$ and $g \in \mathcal{H}'$ are inputs. We consider the problem of approximating $\mathcal{A}$ efficiently from data, using as few inputs $\{f_i\}_{i=1}^N$ as possible. In this paper, we ask

the following question:

Is it possible to recover $\mathcal{A}$ when one can only query $\mathcal{A}$, and not $\mathcal{A}^\top$?

As a toy example, consider the discrete problem of recovering an $N \times N$ rank-one matrix $A = uv^\top$, where $u, v \in \mathbb{R}^N$. The randomized SVD [85] and generalized Nyström [130, 167] methods recover $A$ in just two queries: one with $A$, and one with $A^\top$. However, if one cannot query $A^\top$, one can only observe $uv^\top x = (v^\top x)u$. Therefore, to recover $A$, one needs $N$ matrix-vector products with $A$. Thus, in general, the action of the adjoint is essential to efficient low-rank matrix recovery.

The situation is more complicated for other classes of structured matrices. Consider the recovery of an $N \times N$ Toeplitz matrix $T$ from matrix-vector products. Unlike in the low-rank case, this can be done using just two matrix-vector products with $T$: $Te_1$ and $Te_N$, where $e_i \in \mathbb{R}^n$ is the $i$th elementary basis vector. In this case, access to the action of $T^\top$ is not required, even though $T$ is not symmetric. These examples suggest that depending on what prior information is known about the matrix, the adjoint may or may not be needed in a recovery algorithm [84].

The infinite-dimensional generalization of the matrix recovery problem arises naturally in operator learning [36]. Learning mappings between function spaces has widespread applications in science and engineering, as one can use data to either efficiently approximate existing scientific models or even discover new ones entirely. Moreover, just as low-rank matrices arise naturally in data science [169], operators that occur in physics also have known mathematical properties. Thus, to be as efficient and accurate as possible, operator learning techniques seek to exploit prior knowledge about the operator.

This section aims to bridge the gap between the theoretical requirement for the adjoint in PDE learning and its omission in practice by providing theoretical guar-

antees on operator learning in the adjoint-free setting. Our goal is to understand when and why neural network models can recover operators without access to data about the adjoint operator. To this end, we provide a thorough characterization of various contexts where one can leverage additional assumptions to quantify the accuracy of the adjoint-free reconstruction.

In the finite-dimensional setting of low-rank matrix recovery, we prove that the quality of the reconstructed matrix is fundamentally limited without access to the adjoint. However, we show that the quality of the approximation improves when we have more information about the left and right singular vectors. This suggests that no clever technique from linear algebra can be leveraged in the analogous adjoint-free operator learning problem unless we have prior information.

This section is organized as follows. We begin in Section 5.1.1 with motivational examples for analyzing the sample complexity of non-self-adjoint operator learning. Then, in subsequent sections, we gradually strengthen the assumptions about the operator we wish to learn and analyze the quality of our reconstruction in each case. In Section 5.1.2, we consider the finite-dimensional case of a low-rank operator recovery problem. Given additional information about how close an unknown low-rank matrix is to symmetric, we provide lower and upper bounds (see Theorems 5.1.3 and 5.1.4) on the size of the set of possible matrices satisfying given sketching constraints.

### 5.1.1 Motivational Examples of Non-Self-Adjoint Operator Learning

As a first motivational example for the analysis of non-self-adjoint operators, we consider the problem of estimating the sample complexity of learning parabolic PDEs (generalizing the heat equation) in the following form:

$$\mathcal{P}u \coloneqq u_t - \operatorname{div}(\mathbf{A}(x,t)\nabla u) = f(x,t), \quad x \in \Omega,\, t \in [0,T], \quad 0 < T < \infty,$$

where $\Omega \subset \mathbb{R}^d$ is a bounded spatial domain with Lipschitz smooth boundary and $A(x,t) \in \mathbb{R}^{d\times d}$ is a symmetric positive definite matrix with bounded coefficient functions satisfying the uniform parabolicity condition. Here, we are interested in estimating the number of training pairs $\{(f,u)\}$ needed to learn the solution operator associated with $\mathcal{P}$, i.e., the Green's function [68], to within a target tolerance $\epsilon > 0$. The authors of [33] construct an algorithm that provably converges to the solution operator at an algebraic rate with respect to the number of training pairs. However, a key assumption required to approximate the solution operator is that one can evaluate the adjoint $\mathcal{P}^*$ of the parabolic operator defined as

$$\mathcal{P}^*u = -u_t - \operatorname{div}(\mathbf{A}(x,t)^\top \nabla u).$$

In this work, we ask whether the requirement for the adjoint is an essential assumption and why neural operators do not require the adjoint in practical applications. A second example emerges from the stationary convection-diffusion equation with variable coefficients in the form:

$$\mathcal{L}u \coloneqq -\operatorname{div}(\mathbf{A}(x)\nabla u) + \mathbf{c}\cdot\nabla u, \quad x \in \Omega \subset \mathbb{R}^d, \tag{5.2}$$

where the lower order coefficient vector $\mathbf{c}$ contains functions in $L^p(\Omega)$ for some $p > d$ [97]. Here, one can interpret $\mathcal{L}$ as a perturbation of the self-adjoint partial

differential operator $L$ defined in Equation (5.1). In particular, the magnitude of $\mathbf{c}$ influences the difference between the solution operator and its adjoint, and our ability to approximate it from training pairs of source terms and solutions.

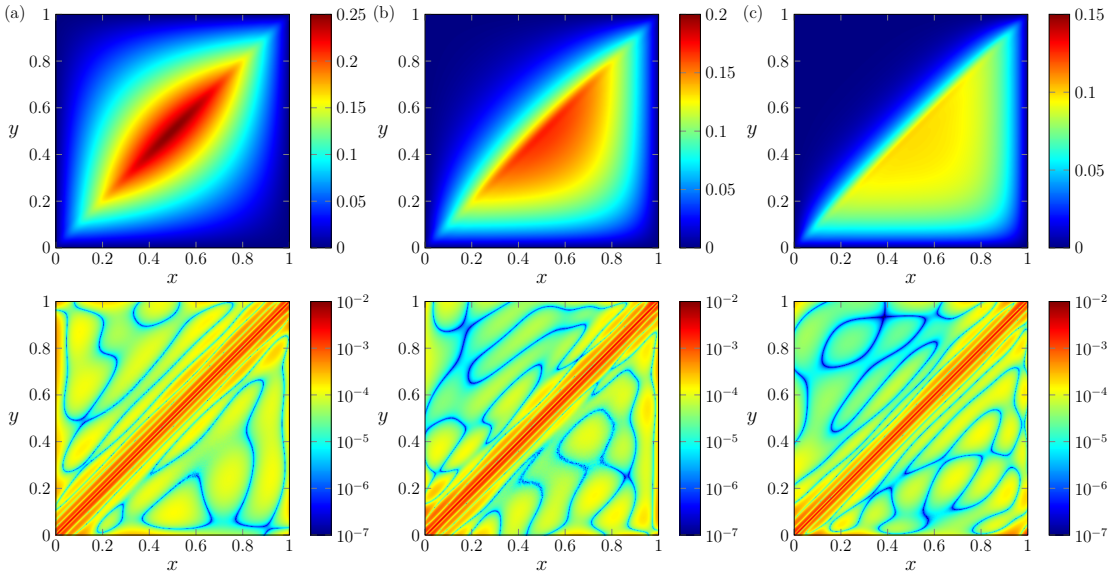



Figure 5.1: Green's functions learned by a rational neural network (top row) along with the absolute error with the exact Green's function (bottom row) for the stationary convection-diffusion equation, with coefficients (a) $c = 0$, (b) $c = 5$, and (c) $c = 10$.

We perform a deep learning experiment to approximate Green's function associated with the one-dimensional stationary convection-diffusion equation with homogeneous Dirichlet boundary conditions on $\Omega = [0, 1]$:

$$-\frac{d^2u}{dx^2} + c\frac{du}{dx} = f, \quad u(0) = u(1) = 0, \quad x \in [0, 1]. \tag{5.3}$$

We employ a rational neural network [34] to approximate the Green's function associated with Equation (5.3) using the Green's function learning technique introduced by [30]. We sample 25 random functions from a Gaussian process with squared-exponential kernel and length-scale parameter $\ell = 0.03$ and solve Equation (5.3) using a Chebyshev spectral collocation method implemented in the Chebfun software system [63]. The source terms $f$ and solutions $u$ are then sampled on

a uniform grid with 200 points, and the neural network is trained in the TensorFlow library [1] using a combination of Adam [98] and L-BFGS [38] optimization algorithms. The learned Green's function is then evaluated at a higher resolution on a $400 \times 400$ grid and compared with the analytical expression for the exact Green's function given by:

$$G(x,y) = \frac{(1-e^{c(x-1)})(1-e^{-cy})}{c(1-e^{-c})} H(x-y) + \frac{(-1+e^{cx})(e^{-cy}-e^{-c})}{c(1-e^{-c})} H(y-x),$$

where $H$ is the Heaviside step function.

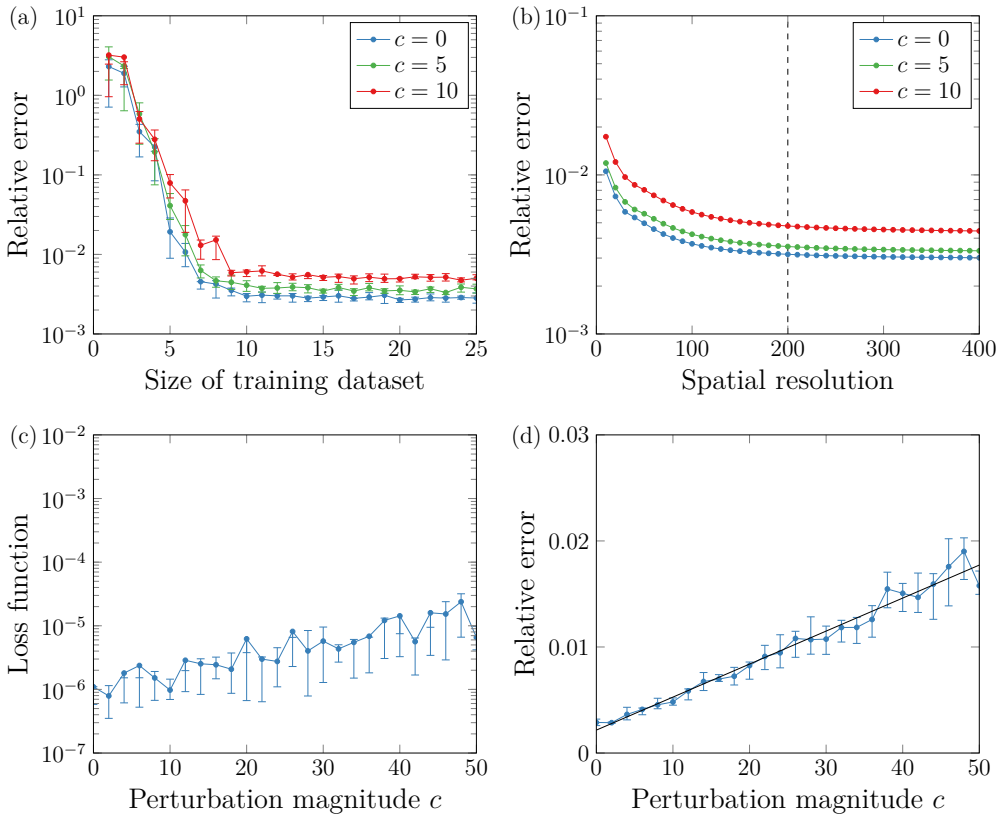



Figure 5.2: (a) Relative errors for learning the Green's function of the advection-diffusion equation. The graph displays the mean error over ten runs, along with error bars representing the first and third quartiles. (b) Relative errors of the Green's function after training using input-output pairs sampled on a grid with resolution $s = 200$ (dashed line) and evaluated at lower and higher resolutions. (c)-(d) Evolution of the loss function after training and relative error as the magnitude of the perturbation increases. The black line in (d) represents the linear least squares approximation and achieves $R^2 = 0.8$.

We report in Figure 5.1(a-c) the learned Green's functions of the stationary convection-diffusion equation (5.3) with respective convection parameters $c = 0$, $c = 5$, and $c = 10$, along with the absolute error with the exact Green's function in the bottom row. We observe that even though the difference between the Green's function and its adjoint increases between $c = 0$ and $c = 10$, the rational neural network can approximate the Green's function within $0.3 - 0.5\%$ relative error. Additionally, the approximation errors are mainly located around the diagonal $x = y$ of the domain $[0, 1]^2$ (see the bottom row in Figure 5.1), where the Green's function has derivative discontinuity.

In Figure 5.2, we study the dependence of the error on the size of the training dataset, the spatial grid resolution, and the convection coefficient $c$. To do this, we approximate the Green's function associated with Equation (5.3) using 25 training pairs sampled on a grid with resolution $s = 200$ and report the relative error when evaluating the Green's function at different resolutions from $s = 10$ to $s = 400$. Similar to the Fourier neural operator [99], the rational neural network is capable of zero-shot super-resolution, even when learning highly non-self-adjoint operators, as the approximation error does not degrade when testing the network at a higher resolution.

The relative error plotted in Figure 5.2(a) for different convection coefficients shows two distinct regimes as the size of the training dataset is increased. There is an initial exponential decay of the error up to 10 training pairs, followed by a stagnation at small relative error. These results agree with previous experiments performed in [32] and suggest that one can approximate non-self-adjoint operators with few training data using deep learning. However, our experiments show that the plateau in relative error increases with the magnitude of the convection coeffi-

cient $c$, suggesting that there is a systematic component of the error that increases with the non-self-adjointness of the operator we seek to learn.

To study the systematic error introduced by non-self-adjoiness, we progressively increase the magnitude of the perturbation in Equation (5.3) and report the corresponding loss function after training and relative errors for the learned Green's function in Figure 5.2(c)-(d). Using a linear least squares regression ($R^2 = 0.78$), we observe that the relative error increases linearly with the magnitude of the perturbation. At the same time, the loss function after training remains relatively small between $10^{-6}$ and $10^{-5}$. The discrepancy between the magnitude of the loss function and the relative error is because the loss function is a relative mean-squared error, while the error reported in Figure 5.2(d) is measured as a relative $L^2$-error between the exact and learned Green's functions. Finally, we observe in Figure 5.2(d) that the variance of the errors also increases with the perturbation magnitude. These numerical experiments motivate our theoretical analysis of learning non-self-adjoint solution operators associated with PDEs in the rest of the paper. Moreover, these numerical results lead us to introduce an adjoint-free operator learning method that provably converges with the size of the training dataset, and therefore does not suffer from the plateau observed in Figure 5.2(a).

### 5.1.2 Reconstruction of Low-Rank Matrices under Sketching Constraints

In this section, we analyze a finite-dimensional variant of the main problem. Existing sample complexity bounds for learning Green's functions of linear PDEs, which consider self-adjoint elliptic PDEs [35, 32] or parabolic PDEs [33], assume access

to the adjoint operator. The proofs of these results exploit randomized numerical linear algebra techniques, such as the randomized singular value decomposition (SVD) [85, 120, 29], to construct low-rank approximants of the Green's function on well-separated subdomains. This motivates our investigation into the analogous problem of adjoint-free recovery of low-rank matrices from matrix-vector products.

The randomized SVD is an algorithm that computes a near-best low-rank approximant to a matrix $F \in \mathbb{R}^{n\times n}$ from matrix-vector products with a random input matrix $X \in \mathbb{R}^{n\times s}$ using a two-stage procedure. First, one sketches the matrix $F$ at $X$ to obtain $Y = FX$ and orthonormalizes $Y$ to form a basis $Q$, which approximates the range of $F$. Then, one constructs the approximant $\hat{F} = QQ^*F = Q(F^*Q)^*$ by sketching the adjoint, $F^*$, of $F$. In a landmark paper, [85] showed that the approximant $\hat{F}$ is a near-best approximant to $F$ with high probability.

If $F$ is too large to be stored or given as a streaming model [129, 49, 180], $F = H_1+H_2+H_3+\cdots$, one might not be able to view the matrix twice as in the randomized SVD [120]. Several single-view algorithms have been proposed to compute an approximate SVD of $F$, which visit the matrix only once, such as the Nyström method [134, 74, 107, 167]. However, to our knowledge, every low-rank approximation algorithm based on sketching requires access to $F^\top$ [120, 84]. This leads to a natural question: is there an algorithm that constructs a low-rank approximant to $F$ without its adjoint? Recently, this question was answered negatively. It was proven that there are infinitely many rank-$k$ matrices $F$ satisfying the same matrix-vector products $FX = Y$ and $F^\top W = Z$, where $X \in \mathbb{R}^{n\times k_1}, W \in \mathbb{R}^{n\times k_2}$, if $\min(k_1, k_2) < k$ and $\max(k_1, k_2) < n$ [84]. This result extends to complex-valued matrices, so one needs $k$ queries to $F$ and $F^\top$ each for the matrix recovery problem to have a unique solution. Even if one has as many as $n - 1$ matrix-vector

products with $F$ and none with $F^\top$, $F$ is not uniquely determined. Thus, this section considers the space of these infinitely many possible rank-$k$ matrices when the recovery problem is underspecified and does not have a unique solution.

We aim to understand how close one can get to recovering an unknown low-rank matrix $F \in \mathbb{C}^{n\times n}$ from matrix-vector products with an input matrix $X \in \mathbb{C}^{n\times s}$, i.e., without access to its adjoint. It has already been shown that when one does not have access to $F^\top$, the possible row spaces of a matrix satisfying the same sketch constraints as $F$ can be arbitrarily far apart in the Riemannian metric on the Grassmannian manifold [137]. Thus, we assume some additional structure on our space of possible matrices, using the following notion of near-symmetry.

**Definition 5.1.1** (Near-symmetry)**.** *Let $F \in \mathbb{C}^{n\times n}$ be a rank-$k$ matrix with singular value decomposition $F = U_F S_F V_F^\top$. We say that $F$ is $\delta$-near-symmetric if its left and right singular subspaces are $\delta$-close, i.e., there exists a $k \times k$ orthogonal matrix $Q$ such that*

$$\|U_F^\top V_F - Q\|_2 \leq \delta.$$

We show that one cannot recover an accurate low-rank approximant to $F$ unless $F$ is near-symmetric. This analysis indicates that the adjoint is essential for low-rank recovery algorithms.

### Reconstruction of Near-Symmetric Matrices

We consider an unknown rank-$k$ matrix $F \in M_n(\mathbb{C})$ with singular value decomposition $F = U_F S_F V_F^\top$ and aim to construct an approximant $A$ to $F$ satisfying the sketch constraint $AX = FX$, where the test matrix $X \in \mathbb{C}^{n\times s}$ has linearly independent columns and $\operatorname{rank}(FX) = k$. By construction, we have $k \leq s \leq n$.

Almost every matrix $X \in \mathbb{C}^{n\times s}$ with respect to the Lebesgue measure satisfies the condition $\operatorname{rank}(FX) = k$ [137, Lem. 2.4], meaning that the queries almost surely reveal the rank of $F$ and its range. In randomized numerical linear algebra, the test matrix $X$ is typically chosen to be a random matrix following a standard Gaussian distribution [120], but other random embeddings, such as subsampled trigonometric transforms [181] or coordinate samplings [177, 164, 101, 74], may also be used.

We assume that $F$ is $\delta$-near-symmetric (see Definition 5.1.1), but we only have access to partial information regarding the symmetry of $F$, namely that $F$ is $\epsilon$-near-symmetric for some $\epsilon \geq \delta$. To quantify the resulting uncertainty about $F$, we study the set of possible matrices one could recover given this prior knowledge. We denote this set

$$\Omega^{\epsilon}_{F,X} = \{A \in M_n(\mathbb{C})\colon \operatorname{rank}(A) = k,\ AX = FX,\ \exists Q \in O(k),\ \|U_A^{\top} V_A - Q\|_2 \leq \epsilon\}, \tag{5.4}$$

where $A = U_A S_A V_A^{\top}$ is the singular value decomposition of $A$, $O(k)$ is the group of $k \times k$ orthogonal matrices, and $\|\cdot\|_2$ denotes the spectral norm. This set might be nonempty even when $\epsilon < \delta$, but to ensure that $F \in \Omega^{\epsilon}_{F,X}$, we must have $\epsilon \geq \min_{Q\in O(k)} \|U_F^* V_F - Q\|_2$. The minimum exists because $O(k)$ is compact.

**Remark 7** (Low-rank recovery algorithms and $\Omega^{\epsilon}_{F,X}$)**.** *Given some tolerance $\epsilon$, $\Omega^{\epsilon}_{F,X}$ is the set of $\epsilon$-near-symmetric matrices that can be returned by any low-rank recovery algorithm when approximating $F$, such as the randomized SVD [85, 120] or the Nyström method [134]. One can find a symmetric approximation in the set using Nyström method by querying $A$ in place of $A^*$.*

The size of $\Omega^{\epsilon}_{F,X}$ is measured by its diameter in the spectral norm and determines the maximum accuracy of any reasonable reconstruction. If the diameter

is large, one cannot estimate $F$ accurately, as one cannot distinguish between any candidate matrix in $\Omega^{\epsilon}_{F,X}$. This is because any matrix in $\Omega^{\epsilon}_{F,X}$ satisfies the sketching constraint and is near-symmetric. On the other hand, a small diameter guarantees the fidelity of the reconstruction. We aim to bound the size of $\Omega^{\epsilon}_{F,X}$, i.e., determine how far apart any two matrices in $\Omega^{\epsilon}_{F,X}$ can be from each other, with respect to $\epsilon$, which measures our prior knowledge of $F$'s symmetry. We first provide an upper bound on the diameter of the set $\Omega^{\epsilon}_{F,X}$.

The upper bound relies on a preliminary lemma. Lemma 5.1.2 provides an orthogonal change of basis, bringing a matrix $U$ with orthonormal columns close to another matrix $V$ in the spectral norm sense. The difference is bounded by the proximity of $U^*V$ to an orthogonal matrix. For the rest of this section, $\sigma_{\max}$ and $\sigma_{\min}$ respectively denote the largest and smallest nonzero singular values of a matrix.

**Lemma 5.1.2.** *Let $U$ and $V$ be two $n \times k$ matrices with orthonormal columns and $U^\top V = Q_l \Sigma Q_r^\top \in \mathbb{C}^{k\times k}$ be a complete SVD. Then, the orthonormal matrix $Q_0 = Q_l Q_r^\top$ satisfies*

$$\|V - UQ_0\|_2^2 = 2\left(\min_{Q:\ Q^\top Q = I} \|Q - U^\top V\|_2\right) = 2\left(1 - \sigma_{\min}(U^\top V)\right).$$

*Proof.* We first consider the SVD of the matrix $U^*V$ as $U^*V = Q_l \Sigma Q_r^*$ and introduce the orthonormal matrix $Q_0 = Q_l Q_r^\top$. Let $d = \min_{Q:\ Q^\top Q = I} \|Q - U^*V\|_2$ denote the distance between the $k \times k$ matrix $U^*V$ and the set of orthogonal matrices.

In general, from a result for unitarily invariant norms [70, Thm. 1], if $A$ is a square matrix with $\|A\|_2 \leq 1$ and has complete SVD $A = U\Sigma V^*$, then

$$\min_{Q:\ Q^\top Q = I} \|Q - A\|_2 = 1 - \sigma_{\min}(A) \tag{5.5}$$

is achieved by $Q = UV^*$. Applying Equation (5.5) to $U^*V$ yields a characterization of $d$ with the smallest singular value of $U^*V$ as $\sigma_{\min}(U^*V) = 1 - d$. Let $x \in \mathbb{R}^k$ be a unit vector, then $v = Vx$ and $u = UQ_0x$ have norm 1 because $U$ and $V$ have orthonormal columns. Moreover, $u^\top v = x^\top Q_0^\top U^\top V x = x^\top Q_r Q_l^\top Q_l \Sigma Q_r^\top x = x^\top Q_r \Sigma Q_r^\top x \geq \sigma_{\min}(U^\top V) = 1 - d$, with equality when $x = Q_r e_k$. Using similar triangles in the $(u, v)$-plane, we readily obtain

$$\frac{1 - u^\top v}{\|u - v\|} = \frac{\|u - v\|}{2},$$

which implies that$\|Vx - UQ_0x\|^2 = \|v - u\|^2 = 2(1 - u^\top v) \leq 2d$, with equality when $x = Q_r e_k$. Finally, taking the supremum over $x$, $\|V - UQ_0\|_2^2 = \sup_{x:\ \|x\|=1} \|Vx - UQ_0x\|^2 = 2d$, which concludes the proof. □

We are now ready to state Theorem 5.1.3, which provides an upper bound on the diameter of the set $\Omega_{F,X}^\epsilon$ defined in Equation (5.4).

**Theorem 5.1.3** (Upper bound)**.** *Let $0 \leq \delta \leq \epsilon < 1$, $F \in M_n(\mathbb{C})$ be a $\delta$-near-symmetric rank-$k$ matrix, and $X \in \mathbb{C}^{n \times s}$ be a test matrix with $s \geq k$ orthonormal columns such that* $\operatorname{rank}(FX) = k$*. Let $F = U_0 \Sigma_0 V_0^*$ be a slim SVD of $F$, and introduce the constant $c = \sigma_{\max}(X^*V_0)/\sigma_{\min}(X^*V_0)^2$. If $c\big(\sqrt{2\epsilon} + \sqrt{2\delta}\big) < 1$, then*

$$\sup_{A,B \in \Omega_{F,X}^\epsilon} \|A - B\|_2 \leq 4\|FX\|_2 \left[ \frac{c^2\big(\sqrt{2\epsilon} + \sqrt{2\delta}\big)}{1 - c\big(\sqrt{2\epsilon} + \sqrt{2\delta}\big)} \right].$$

*Proof.* Recalling the $X$ has orthonormal columns and letting $\Phi = XX^*V_0$, we observe that

$$F = F\Phi(V_0^*\Phi)^{-1}V_0^*.$$

This is easily verified using the SVD of $F$. We note that $V_0^*\Phi = V_0^*XX^*V_0$ is invertible because $\operatorname{rank}(FX) = \operatorname{rank}(F) = k$.

Suppose that $A \in \Omega^{\epsilon}_{F,X}$ and let $A = U_A \Sigma_A V_A^*$ be a slim SVD. Then $A = F\Phi(V_A^*\Phi)^{-1}V_A^*$, as one can verify using the SVD of $A$ and the identity $F\Phi = A\Phi$. For any invertible $k \times k$ matrix $Q$ (later on $Q$ will be orthogonal), $A = F\Phi(QV_A^*\Phi)^{-1}QV_A^*$. By the triangle inequality,

$$\|F - A\|_2 \leq \|F\Phi\|_2 \|(V_0^*\Phi)^{-1}\|_2 \|V_A Q^* - V_0\|_2 + \|F\Phi\|_2 \|(V_0^*\Phi)^{-1} - (QV_A^*\Phi)^{-1}\|_2. \tag{5.6}$$

We use a perturbation argument to show that $V_A^*\Phi$ is invertible, to choose $Q$, and to bound each term in the above equation.

By Lemma 5.1.2 and Equation (5.5), there exist orthogonal matrices $Q_0$ and $Q_A$ satisfying $\|V_0 - U_0 Q_0\| \leq \sqrt{2\delta}$ and $\|V_A - U_A Q_A\| \leq \sqrt{2\epsilon}$. Since $\mathrm{Range}(U_A) = \mathrm{Range}(U_0)$ there is an orthogonal matrix $\tilde{Q}$ such that $U_A Q_A = U_0 \tilde{Q}$. Letting $Q = Q_0 \tilde{Q}$ we obtain

$$\|V_A Q^* - V_0\|_2 = \|V_A - V_0 Q\|_2 \leq \|V_A - U_A Q_A\| + \|U_0 \tilde{Q} - V_0 Q_0 \tilde{Q}\|_2 \leq \sqrt{2\epsilon} + \sqrt{2\delta} \tag{5.7}$$

A classical perturbation bound for the difference between the inverse of two matrices $S$ and $T$ is [96, Ch. 1, Eq. 4.24]

$$\|S^{-1} - T^{-1}\|_2 \leq \frac{\|S - T\|_2 \|T^{-1}\|_2^2}{1 - \|S - T\|_2 \|T^{-1}\|_2}.$$

Applying this with $S = QV_A^*\Phi$ and $T = V_0^*\Phi$ gives

$$\|(QV_A^*\Phi)^{-1} - (V_0^*\Phi)^{-1}\|_2 \leq \frac{\left(\sqrt{2\epsilon} + \sqrt{2\delta}\right)\|\Phi\|_2 \|(V_0^*\Phi)^{-1}\|_2^2}{1 - \left(\sqrt{2\epsilon} + \sqrt{2\delta}\right)\|\Phi\|_2 \|(V_0^*\Phi)^{-1}\|_2},$$

where

$$\|(QV_A^*\Phi) - (V_0^*\Phi)\|_2 \leq \|V_A Q^* - V_0\|_2 \|\Phi\|_2 \leq \left(\sqrt{2\epsilon} + \sqrt{2\delta}\right)\|\Phi\|_2,$$

by Equation (5.7). A quick computation shows that

$$\|\Phi\|_2 \|(V_0^*\Phi)^{-1}\|_2 = \|X^* V_0\|_2 \|(V_0^* X X^* V_0)^{-1}\|_2 = \frac{\sigma_{\max}(X^* V_0)}{\sigma_{\min}(X^* V_0)^2} = c,$$

and that $\|F\Phi\|_2 \leq \|FX\|_2\|X^*V_0\|_2 = \|FX\|_2\|\Phi\|_2$. The condition that $c\big(\sqrt{2\epsilon} + \sqrt{2\delta}\big) < 1$ ensures that both $QV_A^*\Phi$ and $V_A^*\Phi$ are invertible. Using these results in Equation (5.6) and collecting terms yields

$$\begin{aligned}\|F - A\|_2 &\leq \|FX\|_2\big(\sqrt{2\epsilon} + \sqrt{2\delta}\big)\left[c + \frac{c^2}{1 - c\big(\sqrt{2\epsilon} + \sqrt{2\delta}\big)}\right] \\ &\leq 2\|FX\|_2\left[\frac{c^2\big(\sqrt{2\epsilon} + \sqrt{2\delta}\big)}{1 - c\big(\sqrt{2\epsilon} + \sqrt{2\delta}\big)}\right].\end{aligned}$$

Applying the triangle inequality $\|A-B\|_2 \leq \|F-A\|_2+\|F-B\|_2$ with $A, B \in \Omega_{F,X}^{\epsilon}$ completes the proof. □

To lower bound the diameter of $\Omega_{F,X}^{\epsilon}$, we must show the existence of two matrices in the set that are at least some distance apart. We generate these matrices by perturbing $F$. Our argument makes use of the gap between $\epsilon$ and $\delta$, reflecting the gap in our prior knowledge about the near-symmetry of $F$.

**Theorem 5.1.4** (Lower bound)**.** *Let $0 \leq \delta \leq \epsilon < 1$, $F \in M_n(\mathbb{C})$ be a $\delta$-near-symmetric rank-$k$ matrix, and $X \in \mathbb{C}^{n\times s}$ be a test matrix with $k \leq s < n$ orthonormal columns such that* $\operatorname{rank}(FX) = k$*. Then, the diameter of $\Omega_{F,X}^{\epsilon}$ is lower bounded as follows:*

$$\sup_{A,B\in\Omega_{F,X}^{\epsilon}} \|A - B\|_2 \geq 2\left(\frac{\sigma_{\min}(F)^2}{\sigma_{\max}(F)}\right)\frac{\arccos(1-\epsilon) - \arccos(1-\delta)}{\pi/2 + \arccos(1-\epsilon) - \arccos(1-\delta)}. \quad (5.8)$$

*Proof.* We begin the proof by selecting a matrix $E \in M_n(\mathbb{C})$, satisfying $EX = 0$ and

$$\|E\|_2 = \left(\frac{\sigma_{\min}(F)}{\sigma_{\max}(F)}\right)\frac{\arccos(1-\epsilon) - \arccos(1-\delta)}{\pi/2 + \arccos(1-\epsilon) - \arccos(1-\delta)}, \quad (5.9)$$

which is well defined since $\epsilon \geq \delta$. The constraint $EX = 0$ is satisfied by choosing the rows of $E$ in $\operatorname{Range}(X)^{\perp}$, which is nontrivial because $s < n$. Letting $B = F(I + E) = U_B\Sigma_B V_B^{\top}$, we aim to show that $B \in \Omega_{F,X}^{\epsilon}$. First, we observe that

Range$(B)$ = Range$(F)$, that is Range$(U_B)$ = Range$(U_F)$, because Range$(B) \subset$ Range$(F)$ and $k = \text{rank}(F) \geq \text{rank}(B) \geq \text{rank}(BX) = \text{rank}(FX) = k$. Following Equation (5.5), we must show that

$$1 - \sigma_{\min}(U_B^\top V_B) \leq \epsilon.$$

The identity $\sigma_i(U_B^\top V_B) = \cos(\theta_i(U_B, V_B))$, where $\theta_i(U_B, V_B)$ denotes the $i$th principal angle between subspaces Range$(U_B)$ and Range$(V_B)$ (see [26]), yields

$$1 - \sigma_{\min}(U_B^\top V_B) = 1 - \cos(\theta_{\max}(U_B, V_B)),$$

because $x \mapsto \cos(x)$ is a decreasing function over the interval $[0, \pi/2]$. Therefore, it suffices to show that $\theta_{\max}(U_B, V_B) \leq \arccos(1 - \epsilon)$. Thanks to the main result of [144], the largest principal angle $\theta_{\max}$ is a unitarily-invariant metric on the Grassmannian consisting of $k$-dimensional subspaces of $\mathbb{R}^n$. In particular, it satisfies the triangle inequality:

$$\theta_{\max}(U_B, V_B) = \theta_{\max}(U_F, V_B) \leq \theta_{\max}(U_F, V_F) + \theta_{\max}(V_F, V_B).$$

Combining the assumption on $F$ and Equation (5.5), we have

$$1 - \delta \leq \sigma_{\min}(U_F^\top V_F) = \cos(\theta_{\max}(U_F, V_F)) \implies \theta_{\max}(U_F, V_F) \leq \arccos(1 - \delta),$$

as $\theta \mapsto \arccos(\theta)$ is a decreasing function. Therefore,

$$\theta_{\max}(U_B, V_B) \leq \arccos(1 - \delta) + \theta_{\max}(V_F, V_B).$$

Using Wedin's theorem [174], we obtain

$$\begin{aligned}
\theta_{\max}(V_F, V_B) &\leq \frac{\pi}{2} \sin(\theta_{\max}(V_F, V_B)) \\
&\leq \frac{\pi}{2} \| \sin(\Theta(V_F, V_B)) \|_{\mathsf{F}} \\
&\leq \frac{(\pi/2)\sigma_{\max}(F)\|E\|_2}{\sigma_{\min}(F) - \sigma_{\max}(F)\|E\|_2},
\end{aligned}$$

which means that

$$\theta_{\max}(U_B, V_B) \leq \arccos(1-\delta) + \frac{(\pi/2)\sigma_{\max}(F)\|E\|_2}{\sigma_{\min}(F) - \sigma_{\max}(F)\|E\|_2}.$$

Inserting the expression for $\|E\|_2$ given by Equation (5.9) yields $\theta_{\max}(U_B, V_B) \leq \arccos(1-\epsilon)$, which shows that $B \in \Omega^{\epsilon}_{F,X}$. Since the same argument shows that $F(I-E) \in \Omega^{\epsilon}_{F,X}$, we obtain a lower bound on the diameter as follows:

$$\operatorname{diam}(\Omega^{\epsilon}_{F,X}) \geq 2\left(\frac{\sigma_{\min}(F)^2}{\sigma_{\max}(F)}\right)\frac{\arccos(1-\epsilon) - \arccos(1-\delta)}{\pi/2 + \arccos(1-\epsilon) - \arccos(1-\delta)},$$

which concludes the proof. □

**Remark 8** (Orthonormal test matrices)**.** *Almost every test matrix $X \in \mathbb{C}^{n\times s}$ (with respect to the Lebesgue measure) has linearly independent columns. Thus, these columns can be orthonormalized using the Gram–Schmidt process. Therefore, without loss of generality, we can assume that the test matrix in Theorems 5.1.3 and 5.1.4 has orthonormal columns. More formally, if $\tilde{X} \in \mathbb{C}^{n\times s}$ is a matrix with linearly independent columns, recovering a matrix $A$ from the matrix-vector products $A\tilde{X} = Y$ is equivalent to recovering $AQ = YR^{\dagger}$, where $\tilde{X} = QR$ is a $QR$ factorization. Thus, our results extend to the general matrix recovery model from matrix-vector products.*

**Remark 9.** *(Sharpness for symmetric $F$ and areas for improvement) There are limitations to our upper and lower bounds on the diameter of $\Omega^{\epsilon}_{F,X}$ that we hope will be resolved by future works. First, our upper bound becomes infinite as $\epsilon$ increases to a finite value, whereas our lower bound saturates as $\epsilon$ is increased. On the other hand, our lower bound vanishes when $\epsilon = \delta$, while the upper bound takes a positive value. Despite these issues, when $F$ is symmetric, i.e., when $\delta = 0$, our bounds yield*

$$\Theta(\sqrt{\epsilon}) = \frac{2c^{-1}\arccos(1-\epsilon)}{\pi/2 + \arccos(1-\epsilon)} \leq \sup_{A,B\in\Omega^{\epsilon}_{F,X}} \|A-B\|_2 \leq \frac{4\|FX\|c^2\sqrt{2\epsilon}}{1-c\sqrt{2\epsilon}} = \mathcal{O}(\sqrt{\epsilon}),$$

*as $\epsilon \to 0$, meaning that they are asymptotically sharp in this regime. Sharpening our bounds when $F$ is asymmetric is particularly challenging because little is known about the properties of $\Omega^{\epsilon}_{F,X}$. To our knowledge, our work is the first to define such a set, and we hope that future works will investigate this set more thoroughly.*

The upper and lower bounds reveal that the uncertainty about $F$ given queries of its action is directly related to the uncertainty about the symmetry of its left and right singular subspaces. For example, our ability to recover a symmetric rank-$k$ matrix using $k \leq s < n$ queries is fundamentally limited by our prior knowledge about the proximity of Range($F$) and Range($F^*$) because there are many asymmetric matrices with the same rank that satisfy the same sketching constraints. As described above, generic $n \times s$ test matrices $X$ are capable of revealing the range of $F$, meaning that the uncertainty about $F$ really comes from a lack of prior knowledge about the range of $F^*$. This highlights why the action of the adjoint is essential for efficient matrix recovery without any prior information about an operator's symmetry, or more generally about the range of its adjoint.

## 5.2 Non-uniqueness of orthonormal least-squares

Suppose we are given the least-squares problem $Qx = b$, where $Q \in \mathbb{R}^{m \times n}$ has orthonormal columns. If one has access to matrix-vector products with both $Q$ and $Q^\top$, one need only compute $Q^\top b = x$, so only one matrix-vector product is needed.

However, suppose one has access to matrix-vector products with $Q$, but not $Q^\top$. We demonstrate that this is not enough information to uniquely determine the least squares problem. We construct another least squares problem, $By = b$, where

the solution $y$ always at least some distance from $x$, and $B$ is also well-conditioned.

**Proposition 5.2.1.** *Let $Q \in \mathbb{R}^{m\times n}$ have orthonormal columns, $b \in \mathbb{R}^m$, and $x \in \mathbb{R}^n$ such that $Qx = b$. Let $X \in \mathbb{R}^{n\times k}$, where $k < n$, be a query matrix with orthonormal columns such that $x \notin col(X)$. Then there exists a constant $\delta > 0$, a matrix $B \in \mathbb{R}^{m\times n}$, and a vector $y \in \mathbb{R}^n$ such that $QX = BX$, $By = b$, and $\kappa_2(B) \leq 2$, but $\|x - y\| = \delta > 0$.*

*Proof.* We denote the columns of $X$ as $x_1, \ldots, x_k$, and the outputs of matrix-vector multiplication as $Qx_1 = y_1, \ldots, Qx_k = y_k$. Let $v \in \mathbb{R}^n$ be a unit vector such that $v^\top X = 0$. Generically we have that $v^\top x \neq 0$, as we assumed that $x \notin \mathrm{col}(X)$. (If $v^\top x \neq 0$, this is a fortunate situation in which we queried with input vectors whose span contains the solution we are looking for; for this reason this is not further considered.) Let $\delta$ be any constant such that $\frac{1}{2}\,|v^\top x| > \delta > 0$ and $\delta \neq$.

Define the matrix $B$ as follows: $B = Q + \frac{1}{2}\, y_1 v^\top = Q\,(I + \frac{1}{2}\, x_1 v^\top)$. Because $x_1$ and $v$ are unit vectors, $\|\frac{1}{2}\, x_1 v^\top\| = \frac{1}{2}$. we can bound $\kappa_2(B)$ using Weyl's inequality:

$$\sigma_{\max}(B) \leq \frac{3}{2} \quad \text{and} \quad \sigma_{\min}(B) \geq \frac{1}{2} \implies \kappa_2(B) \leq 3.$$

Also, $B$ also has linearly independent columns and $QX = BX$ by construction. It remains to compute the solution $y$ to $By = b$. To do so, we write the QR decomposition of $B$ in terms of $Q$. By the rank-one update formula for the QR decomposition, we have that $B = Q_1 R_1$, where

$$Q_1 = Q J_{n-1} \cdots J_1 G_1 \cdots G_{n-1} \quad \text{and} \quad R_1 = G_{n-1}^\top \cdots G_1^\top H_1$$

and $H_1 = J_1^\top \cdots J_{n-1}^\top (I + \frac{1}{2} x_1 v^\top)$. Here, $G_i$ and $J_i$ are Givens rotations. The solutions to the least squares problems $Qx = b$ and $By = b$ are given by

$$x = Q^\top b,$$

$$y = R_1^{-1} Q_1^\top b = H_1^{-1} G_1 \cdots G_{n-1} G_{n-1}^\top \cdots G_1^\top J_1^\top \cdots J_{n-1}^\top Q^\top b = H_1^{-1} J_1^\top \cdots J_{n-1}^\top Q^\top b.$$

Further simplifying $H_1^{-1}$ using the equation for $H_1$ yields

$$\begin{aligned} y &= (J_1^\top \cdots J_{n-1}^\top (I + \tfrac{1}{2} x_1 v^\top))^{-1} J_1^\top \cdots J_{n-1}^\top Q^\top b = (I + \tfrac{1}{2} x_1 v^\top)^{-1} Q^\top b \\ &= \left( I - \frac{\frac{1}{2} x_1 v^\top}{1 + \frac{1}{2} v^\top x_1} \right) Q^\top b = \left( I - \tfrac{1}{2} x_1 v^\top \right) Q^\top b \end{aligned}$$

by the Sherman–Morrison formula for the inverse of a rank-one update to a matrix and the fact that $v \in \text{null}(X)$.

We can now use these equations for $x$ and $y$ to bound their distance.

$$\begin{aligned} \|x - y\| &= \left\| Q^\top b - \left( I - \tfrac{1}{2} x_1 v^\top \right) Q^\top b \right\| \\ &= \left\| \tfrac{1}{2} x_1 v^\top Q^\top b \right\| = \tfrac{1}{2} \left\| x_1 v^\top x \right\| = \tfrac{1}{2} \left| v^\top x \right| > \delta > 0. \end{aligned}$$

□

We assume that $x \notin \text{col}(X)$. If $x \in \text{col}(X)$, then the solution to the least squares problem for both $A$ and $B$ satisfying the same matrix-vector products must be $x$, i.e., $y = x$. This is because one got very lucky and queried $Q$ with vectors $x_i$ whose span contains $x$. Algorithmically, after performing $k$ matrix-vector products to obtain $QX$, one would notice that $b$ is a linear combination of the output vectors, i.e., $b = \sum_{i=1}^n \alpha_i Q x_i = Qx \implies x = \sum_{i=1}^n \alpha_i x_i$ by linearity.

In short, this result implies that one cannot solve a least squares problem using only forward matrix-vector products. Moreover, the result is independent of the algorithm or inputs used.

# CHAPTER 6

# CONCLUSIONS

Many important numerical linear algebra routines are designed for matrix-free setting, as fast matrix-vector products are often computationally efficient in the presence of structure, and randomized sketching is an increasingly common tool in numerical linear algebra. Moreover, exciting applications to PDE learning have also motivated new problems in structured matrix recovery and approximation from matrix-vector products. In this work, we laid a foundation for structured matrix recovery by deriving algorithms and theory for many of these tasks. This framework dovetails nicely with a new focus in our broader linear algebra community on stability in randomized numerical linear algebra.

We began with the general notion of query complexity in Chapter 2, and considered the recovery and approximation problems for several common matrix structures, such as tridiagonal, symmetric, Toeplitz, Hankel, low-rank, and sparse. We proceeded to consider the important class of hierarchical matrices, which accurately approximate the solution operators of some PDEs. We resolved the open question of whether one can guarantee the success of peeling-type algorithms, as well as demonstrated hard matrix cases for some of these algorithms in Chapter 3. Peeling algorithms have been observed to work very accurately in practice, however to the best of our knowledge, ours is the first theoretical guarantee for the accuracy of peeling when the matrix of interest does not have exact hierarchical structure. We also extended peeling to an infinite-dimensional analogue in Chapter 4, and provided a theoretical guarantee for the sample complexity of recovering Green's functions corresponding to uniformly elliptic PDEs. This result justifies the observed data-efficient phenomenon one encounters in operator learning.

Finally, in Chapter 5, we removed the assumption of access to matrix-transpose-vector products, and reconsider what is possible to learn about a matrix with only forward access to queries. We quantified the importance of the transpose in the problem of low-rank approximation and least-squares problems.

Many open problems remain in this area, and the present work mentions several future directions worth investigating. Within numerical linear algebra, the transpose-free setting of Chapter 5 raises the question: which linear algebra tasks fundamentally require the transpose, and which do not? We may also ask how important the structure of the inputs is to matrix recovery, and if one can derive similar guarantees when one cannot choose freely choose inputs, which may be closer to the application of the PDE learning setting. Input-output pairs from e.g., Krylov methods are also worth studying. More broadly, one hopes to better understand the query complexity of different matrix families by resolving: what is the largest class of structured matrices that has low query complexity? Finally, it remains to close a theoretical gap between recovery and approximation. For a fixed class of matrices, do these problems require about the same number of queries?

In the realm of operator learning, there many more questions to consider. It would be ideal to derive lower bounds for the sample complexity of different PDE learning problems. Lower bounds from the hierarchical matrix setting do not immediately extend to this case because not every hierarchical matrix corresponds to the Green's function of a PDE. Resolving this connection amounts to solving an inverse problem, i.e., understanding how the off-diagonal blocks of a hierarchical matrix correspond to the coefficients of a PDE. One may also investigate the perturbation theory for Green's functions to determine how to perturb one hierarchical matrix in a particular way to get another which corresponds to a different

PDE. This is also related to the question of whether PDE discovery (i.e., coefficient discovery) is a strictly harder problem than learning a solution operator.

Hierarchical matrix techniques should also extend to the more challenging problem of hyperbolic PDE learning. This problem is significantly harder than elliptic PDE learning, which we considered in Chapter 4, because the hierarchical structure of the Green's function is not known in advance. Recent work has employed a strategy for testing each block individually for low-rank structure, and then recovering one low-rank block at a time, resulting in an $\varepsilon^{-6}$ in the sample complexity [171]. One hopes to exploit ideas from peeling, where many blocks are recovered simultaneously, to dramatically reduce this constant. This also motivates a related problem in numerical linear algebra, where one may wish to find a near-optimal hierarchical partitioning of a given matrix.

Finally, thus far, theory for operator learning techniques have been restricted to the setting of linear operators. Next, we must tackle the question of whether one can provide guarantees or rigorous algorithms for the task of nonlinear PDE learning.

Matrix recovery and operator learning are conceptually closely related, and interesting phenomena in one area may motivate fascinating problems in the other. For example, the data-efficiency of elliptic PDE learning led us to work to understand the stability of the peeling algorithm, an important tool in numerical linear algebra. As each of these fields evolves, many more intriguing questions are bound to arise. We see the present work as laying a foundation of algorithms and theory that will equip us to tackle these future problems.

# BIBLIOGRAPHY


[1] M. Abadi, A. Agarwal, P. Barham, and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from `https://www.tensorflow.org`.

[2] E. Abbe, J. Fan, K. Wang, and Y. Zhong. Entrywise eigenvector analysis of random matrices with low expected rank. *Ann. Stat.*, 48(3):1452, 2020.

[3] R. Ahlswede and A. Winter. Strong converse for identification via quantum channels. *IEEE Trans. Inf. Theory*, 48(3):569–579, 2002.

[4] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson–Lindenstrauss transform. In *Proc. STOC*, pages 557–563, 2006.

[5] N. Ailon and B. Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.

[6] N. Amsel, T. Chen, F. D. Keles, D. Halikias, C. Musco, and C. Musco. Fixed-sparsity matrix approximation from matrix-vector products, 2024. Preprint, available at `https://arxiv.org/abs/2402.09379`.

[7] T. Askham, A. Barnett, Z. Gimbutas, L. Greengard, L. Lu, J. Magland, D. Malhotra, M. Rachh, V. Rokhlin, M. O'Neil, and F. Vico. Flatiron institute fast multipole libraries. `https://github.com/flatironinstitute/FMM3D`.

[8] B. Austin, E. Roman, and X. Li. Resilient matrix multiplication of hierarchical semi-separable matrices. In *Proc. FTXS*, FTXS '15, 2015.

[9] K. Do Ba, P. Indyk, E. Price, and D. P. Woodruff. Lower bounds for sparse recovery. In *Proc. 21st ACM-SIAM Symp. Discrete Algorithms (SODA)*, pages 1190–1197, 2010.

[10] A. Bakshi, K. L. Clarkson, and D. P. Woodruff. Low-rank approximation with $1/\epsilon^{1/3}$ matrix-vector products. In *Proc. 54th Annu. ACM Symp. Theory Comput. (STOC)*, pages 1130–1143, 2022.

[11] A. Bakshi and S. Narayanan. Krylov methods are (nearly) optimal for low-rank approximation. In *Proc. IEEE Symp. Found. Comput. Sci. (FOCS)*, November 2023.

[12] J. Ballani and D. Kressner. *Matrices with Hierarchical Low-Rank Structures*, pages 161–209. Springer, 2016.

[13] J. Barnes and P. Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(6096):446–449, 1986.

[14] R. H. Bartels and G. W. Stewart. Solution of the matrix equation $AX + XB = C$. *Commun. ACM*, 15(9):820–826, 1972.

[15] R. A. Baston and Y. Nakatsukasa. Stochastic diagonal estimation: Probabilistic bounds and an improved algorithm, 2022. Preprint, available at `https://arxiv.org/abs/2201.10684`.

[16] M. Bebendorf. Efficient inversion of the Galerkin matrix of general second-order elliptic operators with nonsmooth coefficients. *Math. Comput.*, 74(251):1179–1199, 2005.

[17] M. Bebendorf. *Hierarchical Matrices*. Springer, 2008.

[18] M. Bebendorf and W. Hackbusch. Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with $L^\infty$-coefficients. *Numer. Math.*, 95(1):1–28, 2003.

[19] C. Bekas, E. Kokiopoulou, and Y. Saad. An estimator for the diagonal of a matrix. *Appl. Numer. Math.*, 57(11–12):1214–1229, 2007.

[20] M. Benzi, P. Boito, and N. Razouk. Decay properties of spectral projectors with applications to electronic structure. *SIAM Rev.*, 55(1):3–64, 2013.

[21] M. Benzi and N. Razouk. Decay bounds and $O(n)$ algorithms for approximating functions of sparse matrices. *Electron. Trans. Numer. Anal.*, 28:16–39, 2007.

[22] M. Benzi and V. Simoncini. Decay bounds for functions of Hermitian matrices with banded or Kronecker structure. *SIAM J. Matrix Anal. Appl.*, 36(3):1263–1282, 2015.

[23] M. Benzi and M. Tûma. A comparative study of sparse approximate inverse preconditioners. *Appl. Numer. Math.*, 30(2–3):305–340, 1999.

[24] D. Bertsimas, M. S. Copenhaver, and R. Mazumder. Certifiably optimal low rank factor analysis. *J. Mach. Learn. Res.*, 18(1):907–959, 2017.

[25] D. A. Bini and L. Gemignani. Bernstein–Bézoutian matrices. *Theor. Comput. Sci.*, 315(2–3):319–333, 2004.

[26] A. Björck and G. H. Golub. Numerical methods for computing angles between linear subspaces. *Math. Comput.*, 27(123):579–594, 1973.

[27] S. Börm, L. Grasedyck, and W. Hackbusch. Introduction to hierarchical matrices with applications. *Eng. Anal. Bound. Elem.*, 27(5):405–422, 2003.

[28] W. Boukaram, G. Turkiyyah, and D. Keyes. Randomized GPU algorithms for the construction of hierarchical matrices from matrix-vector operations. *SIAM J. Sci. Comput.*, 41(4):C339–C366, 2019.

[29] N. Boullé and A. Townsend. A generalization of the randomized singular value decomposition. In *Proc. Int. Conf. Learn. Representations (ICLR)*, 2022.

[30] N. Boullé, C. J. Earls, and A. Townsend. Data-driven discovery of Green's functions with human-understandable deep learning. *Sci. Rep.*, 12(1):1–9, 2022.

[31] N. Boullé, D. Halikias, S. E. Otto, and A. Townsend. Operator learning without the adjoint. *Journal of Machine Learning Research*, 25(364):1–54, 2024.

[32] N. Boullé, D. Halikias, and A. Townsend. Elliptic pde learning is provably data-efficient. *Proc. Natl. Acad. Sci. USA*, 120(39), 2023.

[33] N. Boullé, S. Kim, T. Shi, and A. Townsend. Learning Green's functions associated with time-dependent partial differential equations. *J. Mach. Learn. Res.*, 23(218):1–34, 2022.

[34] N. Boullé, Y. Nakatsukasa, and A. Townsend. Rational neural networks. In *Adv. Neural Inf. Process. Syst.*, volume 33, 2020.

[35] N. Boullé and A. Townsend. Learning elliptic partial differential equations with randomized linear algebra. *Found. Comput. Math.*, 23:709–739, 2023.

[36] N. Boullé and A. Townsend. A mathematical guide to operator learning. In S. Mishra and A. Townsend, editors, *Handbook of Numerical Analysis, Vol. 25: Numerical Analysis Meets Machine Learning*, volume 25 of *Handb. Numer. Anal.*, pages 83–125. Elsevier, 2024.

[37] M. Braverman, E. Hazan, M. Simchowitz, and B. Woodworth. The gradient complexity of linear regression. In *Proc. 33rd Conf. Learn. Theory*, volume 125 of *Proc. Mach. Learn. Res.*, 2020.

[38] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, 1995.

[39] T. T. Cai and A. Zhang. Rate-optimal perturbation bounds for singular subspaces with applications to high-dimensional statistics. *Ann. Stat.*, 46(1):60–89, 2018.

[40] J. Cape, M. Tang, and C. E. Priebe. Signal-plus-noise matrix models: Eigenvector deviations and fluctuations. *Biometrika*, 106(1):243–250, 2019.

[41] J. Cape, M. Tang, and C. E. Priebe. The two-to-infinity norm and singular subspace geometry with applications to high-dimensional statistics. *Ann. Stat.*, 47(5):2405–2439, 2019.

[42] A. A. Casulli, D. Kressner, and L. Robol. Computing functions of symmetric hierarchically semiseparable matrices, 2024. Preprint, available at `https://arxiv.org/abs/2402.17369`.

[43] T. F. Chan, L. De Pillis, and H. Van Der Vorst. Transpose-free formulations of Lanczos-type methods for nonsymmetric linear systems. *Numer. Algorithms*, 17:51–66, 1998.

[44] K. Chen, C. Wang, and H. Yang. Deep Operator Learning Lessens the Curse of Dimensionality for PDEs. *arXiv preprint arXiv:2301.12227*, 2023.

[45] T. Chen, F. D. Keles, D. Halikias, C. Musco, C. Musco, and D. Persson. Near-optimal hierarchical matrix approximation from matrix-vector products. In *Proc. 2025 ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2025.

[46] Y. M. Chen, X. S. Chen, and W. Li. On perturbation bounds for orthogonal projections. *Numer. Algorithms*, 73(2):433–444, 2016.

[47] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Stat.*, pages 493–507, 1952.

[48] S. Chewi, J. De Dios Pont, J. Li, C. Lu, and S. Narayanan. Query lower

bounds for log-concave sampling. In *Proc. IEEE Symp. Found. Comput. Sci. (FOCS)*, November 2023.

[49] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In *Proc. 41st Annu. ACM Symp. Theory Comput. (STOC)*, pages 205–214, 2009.

[50] K. L. Clarkson and D. P. Woodruff. Low-rank approximation and regression in input sparsity time. *J. ACM*, 63(6):1–45, 2017.

[51] M. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu. Dimensionality reduction for $k$-means clustering and low rank approximation. In *Proc. STOC*, pages 163–172, 2015.

[52] T. F. Coleman and J.-Y. Cai. The cyclic coloring problem and estimation of sparse Hessian matrices. *SIAM J. Algebraic Discrete Methods*, 7(2):221–235, 1986.

[53] T. F. Coleman and J. J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM J. Numer. Anal.*, 20(1):187–209, 1983.

[54] M. Connolly. *Probabilistic rounding error analysis for numerical linear algebra*. PhD thesis, University of Manchester, August 2023. Available at `https://research.manchester.ac.uk/en/studentTheses/probabilistic-rounding-error-analysis-for-numerical-linear-algebr`.

[55] A. R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *IMA J. Appl. Math.*, 13(1):117–119, 1974.

[56] G. Dasarathy, P. Shah, B. N. Bhaskar, and R. D. Nowak. Sketching sparse matrices, covariances, and graphs via tensor products. *IEEE Trans. Inf. Theory*, 61(3):1373–1388, 2015.

[57] C. Davis and W. M. Kahan. The rotation of eigenvectors by a perturbation. III. *SIAM J. Numer. Anal.*, 7(1):1–46, 1970.

[58] T. A. Davis. Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization. *ACM Trans. Math. Softw.*, 38(1):1–22, 2011.

[59] M. V. de Hoop, N. B. Kovachki, N. H. Nelsen, and A. M. Stuart. Convergence

rates for learning linear operators from noisy data. *SIAM-ASA J. Uncertain. Quantif.*, 11(2):480–513, 2023.

[60] S. Demko, W. F. Moss, and P. W. Smith. Decay rates for inverses of band matrices. *Math. Comput.*, 43(168):491–499, 1984.

[61] P. Dharangutte and C. Musco. A tight analysis of hutchinson's diagonal estimator. In *Symp. Simplicity Algorithms (SOSA)*, pages 353–364. SIAM, 2023.

[62] G. Dolzmann and S. Müller. Estimates for Green's matrices of elliptic systems by $L^p$ theory. *Manuscripta Math.*, 88:261–273, 1995.

[63] T. A. Driscoll, N. Hale, and L. N. Trefethen. *Chebfun Guide*. Pafnuty Publications, 2014.

[64] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[65] Y. C. Eldar and G. Kutyniok. *Compressed Sensing: Theory and Applications*. Cambridge Univ. Press, Cambridge; New York, 2012.

[66] J. Eldridge, M. Belkin, and Y. Wang. Unperturbed: Spectral analysis beyond Davis–Kahan. In *Proc. Algorithmic Learn. Theory*, pages 321–358, 2018.

[67] E. N. Epperly and J. A. Tropp. Efficient error and variance estimation for randomized matrix computations, 2023. Preprint, available at `https://arxiv.org/abs/2207.06342`.

[68] L. C. Evans. *Partial Differential Equations*. American Mathematical Society, 1998.

[69] J. Fan, W. Wang, and Y. Zhong. An $\ell_\infty$ eigenvector perturbation bound and its application to robust covariance estimation. *J. Mach. Learn. Res.*, 18(207):1–42, 2018.

[70] K. Fan and A. J. Hoffman. Some metric inequalities in the space of matrices. *Proc. Am. Math. Soc.*, 6(1):111–116, 1955.

[71] S. Foucart and H. Rauhut. *A Mathematical Introduction to Compressive Sensing*. Springer New York, 2013.

[72] A. Frommer, C. Schimmel, and M. Schweitzer. Analysis of probing techniques for sparse approximation and trace estimation of decaying matrix functions. *SIAM J. Matrix Anal. Appl.*, 42(3):1290–1318, 2021.

[73] C. R. Gin, D. E. Shea, S. L. Brunton, and J. N. Kutz. DeepGreen: Deep learning of Green's functions for nonlinear boundary value problems. *Sci. Rep.*, 11(1):1–14, 2021.

[74] A. Gittens. *Topics in Randomized Numerical Linear Algebra*. PhD thesis, California Institute of Technology, 2013.

[75] I. Gohbert, T. Kailath, and V. Olshevsky. Fast gaussian elimination with partial pivoting for matrices with displacement structure. *Math. Comp.*, 64(212):1557–1576, 1995.

[76] L. Grasedyck and W. Hackbusch. Construction and arithmetics of H-matrices. *Computing*, 70(4):295–334, 2003.

[77] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 135(2):280–292, 1987.

[78] M. Grüter and K.-O. Widman. The Green function for uniformly elliptic equations. *Manuscr. Math.*, 37(3):303–342, 1982.

[79] M. Gu. Subspace iteration randomization and singular value problems. *SIAM J. Sci. Comput.*, 37(3):A1139–A1173, 2015.

[80] W. Hackbusch. A sparse matrix arithmetic based on H-matrices. part i: Introduction to H-matrices. *Computing*, 62(2), 1999.

[81] W. Hackbusch. $\mathcal{H}^2$-Matrices. In *Hierarchical Matrices: Algorithms and Analysis*, pages 203–240. Springer, 2015.

[82] W. Hackbusch. *Elliptic Differential Equations: Theory and Numerical Treatment*. Springer, 2nd edition, 2017.

[83] W. Hackbusch, B. Khoromskij, and S. A. Sauter. On $\mathcal{H}^2$-Matrices. In *Lectures on Applied Mathematics*, pages 9–29. Springer Berlin Heidelberg, 2000.

[84] D. Halikias and A. Townsend. Structured matrix recovery from matrix-vector products. *Numer. Linear Algebra Appl.*, 31(1), 2023.

[85] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, 2011.

[86] E. Hallman, I. C. F. Ipsen, and A. K. Saibaba. Monte carlo methods for estimating the diagonal of a real symmetric matrix. *SIAM J. Matrix Anal. Appl.*, 44(1):240–269, 2023.

[87] P. C. Hansen, K. Hayami, and K. Morikuni. GMRES methods for tomographic reconstruction with an unmatched back projector. *J. Comput. Appl. Math.*, 413:114352, 2022.

[88] T. Hartland, G. Stadler, M. Perego, K. Liegeois, and N. Petra. Hierarchical off-diagonal low-rank approximation of hessians in inverse problems, with application to ice sheet model initialization. *Inverse Probl.*, 39(8), 2023.

[89] G. Heinig and K. Rost. Algebraic methods for Toeplitz-like matrices and operators. In *Algebraic Methods for Toeplitz-like Matrices and Operators*. Springer, Basel, 1984.

[90] T. Hsing and R. Eubank. *Theoretical Foundations of Functional Data Analysis, with an Introduction to Linear Operators*. John Wiley & Sons, 2015.

[91] S. Jiang, H. Pham, D. P. Woodruff, and R. Zhang. Optimal sketching for trace estimation. In *Adv. Neural Inf. Process. Syst.*, volume 34, pages 23741–23753, 2021.

[92] T. Kailath and A. H. Sayed. Displacement structure: Theory and applications. *SIAM Rev.*, 37(3):297–386, 1995.

[93] W. A. Kalender and W. A. Kalender. *Computed Tomography: Fundamentals, System Technology, Image Quality, Applications; [Including 64-Slice Spiral CT]*. Publicis Publ, Erlangen, 3., rev. ed edition, 2011.

[94] M. Kapralov, H. Lawrence, M. Makarov, C. Musco, and K. Sheth. Toeplitz low-rank approximation with sublinear query complexity. *arXiv preprint arXiv:2211.11328*, 2022.

[95] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nat. Rev. Phys.*, 3(6):422–440, 2021.

[96] T. Kato. *Perturbation Theory for Linear Operators*. Springer, 1980.

[97] S. Kim and G. Sakellaris. Green's function for second order elliptic equations with singular lower order coefficients. *Commun. Partial. Differ. Equ.*, 44(3):228–270, 2019.

[98] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*, 2015.

[99] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *J. Mach. Learn. Res.*, 24(1), 2023.

[100] N. B. Kovachki, S. Lanthaler, and A. M. Stuart. Operator learning: Algorithms and analysis, 2024. Preprint, available at `https://arxiv.org/abs/2402.15715`.

[101] S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the Nyström method. *J. Mach. Learn. Res.*, 13(1):981–1006, 2012.

[102] S. Lanthaler, S. Mishra, and G. E. Karniadakis. Error estimates for DeepONets: A deep learning framework in infinite dimensions. *Trans. Math. Appl.*, 6(1):tnac001, 2022.

[103] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[104] L. Lei. Unified $\ell_{2\to\infty}$ eigenspace perturbation theory for symmetric random matrices. *arXiv preprint arXiv:1909.04798*, 2019. Preprint, available at `https://arxiv.org/abs/1909.04798`.

[105] J. Levitt and P.-G. Martinsson. Linear-complexity black-box randomized compression of rank-structured matrices. *SIAM J. Sci. Comput.*, 46(3):A1747–A1763, 2024.

[106] J. Levitt and P.-G. Martinsson. Randomized compression of rank-structured matrices accelerated with graph coloring. *J. Comput. Appl. Math.*, 451:116044, 2024.

[107] H. Li, G. C. Linderman, A. Szlam, K. P. Stanton, Y. Kluger, and M. Tygert. Algorithm 971: An implementation of a randomized algorithm for principal component analysis. *ACM Trans. Math. Softw.*, 43(3):1–14, 2017.

[108] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *Int. Conf. Learn. Representations*, 2021.

[109] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, and A. Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Adv. Neural Inf. Process. Syst.*, 33:6755–6766, 2020.

[110] V. B. Lidskii. On the proper values of a sum and product of symmetric matrices. *Akad. Nauk SSSR*, 75:769–772, 1950.

[111] L. Lin, J. Lu, and L. Ying. Fast construction of hierarchical matrix representation from matrix-vector multiplication. *J. Comput. Phys.*, 230(10):4071–4087, 2011.

[112] M. Lopes, N. B. Erichson, and M. Mahoney. Error estimation for sketched SVD via the bootstrap. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 6382–6392, 2020.

[113] 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. *Nat. Mach. Intell.*, 3(3):218–229, 2021.

[114] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, and G. E. Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Comput. Methods Appl. Mech. Eng.*, 393:114778, 2022.

[115] L. Lu, X. Meng, Z Mao, and G. E. K. DeepXDE: A deep learning library for solving differential equations. *SIAM Rev.*, 63(1):208–228, 2021.

[116] P.-G. Martinsson. A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix. *SIAM J. Matrix Anal. Appl.*, 32(4):1251–1274, 2011.

[117] P.-G. Martinsson. Compressing rank-structured matrices via randomized sampling. *SIAM J. Sci. Comput.*, 38(4):A1959–A1986, 2016.

[118] P.-G. Martinsson and V. Rokhlin. A fast direct solver for boundary integral equations in two dimensions. *J. Comput. Phys.*, 205(1):1–23, 2005.

[119] P.-G. Martinsson, V. Rokhlin, and M. Tygert. A fast algorithm for the

inversion of general toeplitz matrices. *Comput. Math. Appl.*, 50:741–752, 2005.

[120] P.-G. Martinsson and J. A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numer.*, 29:403–572, 2020.

[121] S. Massei, L. Robol, and D. Kressner. hm-toolbox: MATLAB software for HODLR and HSS matrices. *SIAM J. Sci. Comput.*, 42(2):C43–C68, 2020.

[122] X. Meng and M. W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proc. STOC*, pages 91–100, 2013.

[123] J. Mercer. Functions of positive and negative type, and their connection the theory of integral equations. *Proc. R. Soc. A*, 209(441-458):415–446, 1909.

[124] R. Meyer, C. Musco, and C. Musco. On the unreasonable effectiveness of Single Vector Krylov Methods for Low-Rank Approximation. In *Proc. SODA*, 2024.

[125] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Q. J. Math.*, 11(1):50–59, 1960.

[126] R. Molinaro, Y. Yang, B. Engquist, and S. Mishra. Neural inverse operators for solving PDE inverse problems. In *Proc. 40th Int. Conf. Mach. Learn.*, 2023.

[127] R. J. Muirhead. *Aspects of Multivariate Statistical Theory*. Wiley, 1982.

[128] C. Musco, C. Musco, and D. P. Woodruff. Simple heuristics yield provable algorithms for masked low-rank approximation. In *Proc. ITCS*, pages 6:1–6:20, 2021.

[129] Shanmugavelayutham Muthukrishnan et al. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, 2005.

[130] Y. Nakatsukasa. Fast and stable randomized low-rank matrix approximation. *arXiv preprint arXiv:2009.11392*, 2020.

[131] Y. Nakatsukasa and T. Park. Personal communication, 2022.

[132] Y. Nakatsukasa and T. Park. Randomized low-rank approximation for symmetric indefinite matrices. *SIAM J. Matrix Anal. Appl.*, 44(3):1370–1392, 2023.

[133] J. Nelson and H. L. Nguyên. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Proc. FOCS*, pages 117–126, 2013.

[134] E. J. Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Math.*, 54:185–204, 1930.

[135] R. Oliveira. Sums of random hermitian matrices and an inequality by rudelson. *Electron. Commun. Probab.*, 15:203–212, 2010.

[136] S. E. Otto. *A Note on Recovering Matrices in Linear Families from Generic Matrix-Vector Products*, May 2023.

[137] S. E. Otto, A. Padovan, and C. W. Rowley. Model reduction for nonlinear systems by balanced truncation of state and gradient covariance. *SIAM J. Sci. Comput.*, 45(5):A2325–A2355, 2023.

[138] T. Park and Y. Nakatsukasa. Approximating sparse matrices and their functions using matrix-vector products, 2024. Preprint, available at `https://arxiv.org/abs/2310.05625`.

[139] D. S. Parker. Random butterfly transformations with applications in computational linear algebra. Technical Report CSD-950023, UCLA, 1995.

[140] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, K. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *Proc. NIPS Workshops*, 2017.

[141] B. A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Comput.*, 6(1):147–160, 1994.

[142] D. Persson, N. Boullé, and D. Kressner. Randomized Nyström approximation of non-negative self-adjoint operators, 2024.

[143] E. Price and D. P. Woodruff. $(1 + \epsilon)$-approximate sparse recovery. In *Proc. IEEE Symp. Found. Comput. Sci. (FOCS)*, October 2011.

[144] L. Qiu, Y. Zhang, and C.-K. Li. Unitarily invariant metrics on the Grassmann space. *SIAM J. Matrix Anal. Appl.*, 27(2):507–531, 2005.

[145] M. Raissi, A. Yazdani, and G. E. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[146] V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. *SIAM J. Matrix Anal. Appl.*, 31(3):1100–1124, 2009.

[147] M. Rudelson. Random vectors in the isotropic position. *J. Funct. Anal.*, 164(1):60–72, 1999.

[148] A. K. Saibaba. Randomized subspace iteration: Analysis of canonical angles and unitarily invariant norms. *SIAM J. Matrix Anal. Appl.*, 40(1):23–48, 2019.

[149] T. Sarlos. Improved approximation algorithms for large matrices via random projections. In *Proc. IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 143–152, 2006.

[150] F. Schäfer, M. Katzfuss, and H. Owhadi. Sparse cholesky factorization by Kullback–Leibler minimization. *SIAM J. Sci. Comput.*, 43(3), 2021.

[151] F. Schäfer and H. Owhadi. Sparse recovery of elliptic solvers from matrix-vector products. *SIAM J. Sci. Comput.*, 46(2):A998–A1025, 2024.

[152] F. Schäfer, T. J. Sullivan, and H. Owhadi. Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity. *Multiscale Model. Simul.*, 19(2):688–730, 2021.

[153] M. Simchowitz, A. El Alaoui, and B. Recht. Tight query complexity lower bounds for PCA via finite sample deformed wigner law. In *Proc. 50th Annu. ACM SIGACT Symp. Theory Comput. (STOC)*, page http://dx.doi.org/10.1145/3188745.3188796, 2018.

[154] A. Stathopoulos, J. Laeuchli, and K. Orginos. Hierarchical probing for estimating the trace of the matrix inverse on toroidal lattices. *SIAM J. Sci. Comput.*, 35(5):S299–S322, 2013.

[155] G. Stewart and J.-G. Sun. *Matrix Perturbation Theory*. Elsevier, 1990.

[156] G. W. Stewart. On the perturbation of pseudo-inverses, projections and linear least squares problems. *SIAM Rev.*, 19(4):634–662, 1977.

[157] J.-G. Sun. The stability of orthogonal projections. *J. Grad. Sch. (in Chinese)*, 1(2):123–133, 1984.

[158] X. Sun, D. P. Woodruff, G. Yang, and J. Zhang. Querying a matrix through matrix-vector products. *ACM Trans. Algorithms*, 17(4):1–19, 2021.

[159] J. Sylvester. Sur l'équation en matrices $px = xq$. *C. R. Acad. Sci. Paris*, 99(2):67–71, 1884.

[160] J. M. Tang and Y. Saad. A probing method for computing the diagonal of a matrix inverse. *Numer. Linear Algebra Appl.*, 19(3):485–501, 2011.

[161] A. Townsend and L. N. Trefethen. Continuous analogues of matrix factorizations. *Proc. R. Soc. A*, 471(2173):20140585, 2015.

[162] N. Trefethen. A hundred-dollar, hundred-digit challenge. *SIAM News*, 35(1), 2002.

[163] R. K. Tripathy and I. Bilionis. Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *J. Comput. Phys.*, 375:565–588, 2018.

[164] J. A. Tropp. Improved analysis of the subsampled randomized hadamard transform. *Adv. Adapt. Data Anal.*, 3(01n02):115–126, 2011.

[165] J. A. Tropp. User-friendly tail bounds for sums of random matrices. *Found. Comput. Math.*, 12(4):389–434, 2012.

[166] J. A. Tropp and R. J. Webber. Randomized algorithms for low-rank matrix approximation: Design, analysis, and applications. *arXiv preprint arXiv:2306.12418*, 2023. Preprint, available at `https://arxiv.org/abs/2306.12418`.

[167] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Practical sketching algorithms for low-rank matrix approximation. *SIAM J. Matrix Anal. Appl.*, 38(4):1454–1485, 2017.

[168] L. Tunçel, S. A. Vavasis, and J. Xu. Computational complexity of decom-

posing a symmetric matrix as a sum of positive semidefinite and diagonal matrices. *Found. Comput. Math.*, 2023.

[169] M. Udell and A. Townsend. Why are big data matrices approximately low rank? *SIAM J. Math. Data Sci.*, 1(1):144–160, 2019.

[170] Y. Urano. A fast randomized algorithm for linear least-squares regression via sparse transforms. Master's thesis, New York University, 2013.

[171] C. Wang and A. Townsend. Operator learning for hyperbolic partial differential equations, 2023. Preprint, available at `https://arxiv.org/abs/2312.17489`.

[172] S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Sci. Adv.*, 7(40):eabi8605, 2021.

[173] A. Waters, A. Sankaranarayanan, and R. Baraniuk. SpaRCS: Recovering low-rank and sparse matrices from compressive measurements. In *Adv. Neural Inf. Process. Syst.*, volume 24, 2011.

[174] P.-Å. Wedin. Perturbation bounds in connection with singular value decomposition. *BIT Numer. Math.*, 12(1):99–111, 1972.

[175] H. Weyl. Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung). *Math. Ann.*, 71(4):441–479, 1912.

[176] H. Wilber, E. N. Epperly, and A. H. Barnett. A superfast direct inversion method for the nonuniform discrete Fourier Transform, 2024. Preprint, available at `https://arxiv.org/abs/2404.13223`.

[177] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Adv. Neural Inf. Process. Syst.*, volume 13, 2000.

[178] T. Wimalajeewa, Y. C. Eldar, and P. K. Varshney. Recovery of sparse matrices via matrix sketching. *arXiv preprint arXiv:1311.2448*, 2013.

[179] T. Wimalajeewa, Y. C. Eldar, and P. K. Varshney. Recovery of sparse matrices via matrix sketching, 2013. Preprint, available at `https://arxiv.org/abs/1311.2448`.

[180] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1–2):1–157, 2014.

[181] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. A fast randomized algorithm for the approximation of matrices. *Appl. Comput. Harmon. Anal.*, 25(3):335–366, 2008.

[182] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numer. Linear Algebra Appl.*, 17(6):953–976, 2010.

[183] C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis. Improved fast Gauss transform and efficient kernel density estimation. In *Proc. 9th IEEE Int. Conf. Comput. Vis.*, page 464, 2003.

[184] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *J. Comput. Phys.*, 196(2):591–626, 2004.

[185] R. Yokota, H. Ibeid, and D. Keyes. Fast multipole method as a matrix-free hierarchical low-rank approximation. In *Eigenvalue Probl.: Algorithms, Softw. Appl. Petascale Comput.*, pages 267–286, 2017.

[186] Y. Yu, T. Wang, and R. J. Samworth. A useful variant of the Davis–Kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015.

[187] Y. Zhang and M. Tang. Perturbation analysis of randomized SVD and its applications to high-dimensional statistics. *arXiv preprint arXiv:2203.10262*, 2022. Preprint, available at `https://arxiv.org/abs/2203.10262`.

[188] H. Zheng, W. Nie, A. Vahdat, K. Azizzadenesheli, and A. Anandkumar. Fast sampling of diffusion models via operator learning. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.