

2024

## Probing With Displacements For Variance Reduction And The Effectiveness Of Sketched Krylov Eigenvalue Solvers

Heather Maria Switzer

College of William and Mary - Arts & Sciences, [hmswitzer@wm.edu](mailto:hmswitzer@wm.edu)

Follow this and additional works at: <https://scholarworks.wm.edu/etd>



Part of the [Computer Sciences Commons](#)

---

### Recommended Citation

Switzer, Heather Maria, "Probing With Displacements For Variance Reduction And The Effectiveness Of Sketched Krylov Eigenvalue Solvers" (2024). *Dissertations, Theses, and Masters Projects*. William & Mary. Paper 1727787875.

<https://dx.doi.org/10.21220/s2-phv5-yw33>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact [scholarworks@wm.edu](mailto:scholarworks@wm.edu).

Probing with Displacements for Variance Reduction and the Effectiveness of  
Sketched Krylov Eigenvalue Solvers

Heather M. Switzer

Chester, VA, USA

Associate of Science, Richard Bland College, 2015  
Bachelor of Science, Longwood University, 2018  
Master of Science, The College of William & Mary, 2020

A Dissertation presented to the Graduate Faculty  
of The College of William and Mary in Virginia in Candidacy for the Degree of  
Doctor of Philosophy

Department of Computer Science

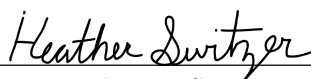
The College of William and Mary in Virginia  
July 2024



## APPROVAL PAGE

This Dissertation is submitted in partial fulfillment of  
the requirements for the degree of

Doctor of Philosophy



---

Heather M. Switzer

Approved by the Committee, July 2024



---

Committee Chair

Professor Andreas Stathopoulos, Computer Science  
College of William & Mary



---

Professor Qun Li, Computer Science  
College of William & Mary



---

Associate Professor Bin Ren, Computer Science  
College of William & Mary



---

Assistant Professor Pradeep Kumar, Computer Science  
College of William & Mary



---

Dr. Jennifer Loe  
Sandia National Laboratory

## ABSTRACT

Scientific Computing is a multidisciplinary field that intersects Computer Science, Mathematics, and some other discipline to address complex problems utilizing computational systems. Numerical Linear Algebra (NLA), a subfield within Scientific Computing, develops and analyzes numerical algorithms for tasks involving linear operators such as matrices and their transformations. This dissertation focuses on developing kernels for two specific areas of NLA.

Firstly, we explore variance reduction methods for trace estimation. In Lattice Quantum Chromodynamics (LQCD), computing the trace of a matrix inverse is crucial for investigating interactions among quarks and gluons in subatomic space. However, directly computing a matrix inverse is computationally expensive, motivating the use of stochastic methods to estimate the trace directly. Higher accuracy models result in a high variance of the trace estimator, necessitating techniques such as probing that leverage the structure of LQCD matrices to reduce this variance. More recently, the development of so-called “disconnected diagrams” in LQCD requires the sum of certain off-diagonal elements in the matrix inverse, rendering classical probing ineffective. In this work, we propose an extension to the probing method that can lessen the variance of the trace estimator in such scenarios.

Secondly, we investigate the viability of a technique that uses randomized subspace projections in Krylov-based iterative methods to approximate a subset of the eigenpairs of a large matrix. A common computational bottleneck for iterative methods comes from the need to orthogonalize the basis being constructed to ensure the accurate extraction of eigenpair approximations. While randomized subspace projections, often referred to as “sketching” methods, were originally introduced to reduce the size of large least-squares problems into more manageable ones, it was later observed that sketching techniques can be integrated into iterative methods to extract information from a non-orthogonal basis with minimal accuracy lost. We investigate the efficiency of these sketching methods using two iterative methods, Lanczos and Generalized Davidson, with and without restarting, within the high-performance software library PRIMME.

# TABLE OF CONTENTS

Acknowledgments	v
Dedication	vi
List of Tables	vii
List of Figures	ix
<b>1 Introduction</b>	<b>2</b>
<b>2 Background</b>	<b>8</b>
2.1 Notation and Conventions . . . . .	8
2.2 Matrix Properties and Decompositions . . . . .	9
2.2.1 The Eigenvalue Problem . . . . .	11
2.2.1.1 The Singular Value Decomposition . . . . .	13
2.3 Variance Reduction Techniques in LQCD . . . . .	14
2.3.1 Finding the Trace of a Matrix Inverse . . . . .	15
2.4 Krylov-based Iterative Methods . . . . .	17
2.4.1 Lanczos . . . . .	20
2.4.1.1 Generalized Davidson . . . . .	26
2.4.2 PRIMME . . . . .	29
2.5 Sketching . . . . .	30

<b>3</b>	<b>Probing for the Trace Estimation of a Permuted Matrix Inverse Corresponding to a Lattice Displacement</b>	<b>32</b>
3.1	Introduction . . . . .	32
3.2	Related Works . . . . .	35
3.2.1	Classical Probing . . . . .	36
3.2.2	Removing Deterministic Bias . . . . .	38
3.2.3	Hierarchical Probing . . . . .	39
3.2.4	Deflation . . . . .	40
3.3	Probing for Permutations . . . . .	41
3.3.1	Coloring with Displacements Algorithm . . . . .	44
3.3.2	Lower Bound on the Number of Colors . . . . .	48
3.3.3	Clearances . . . . .	53
3.3.4	Multiple Displacements . . . . .	53
3.3.5	Tiles . . . . .	54
3.4	Experiments . . . . .	56
3.4.1	Number of Colors Computed . . . . .	57
3.4.2	Comparisons to Other Methods . . . . .	58
3.4.3	Using one coloring for all displacements . . . . .	61
3.5	Chapter Summary . . . . .	65
<b>4</b>	<b>Exploring Krylov Methods with Sketched Rayleigh-Ritz in PRIMME</b>	<b>68</b>
4.1	Introduction . . . . .	68
4.2	Subspace Embeddings . . . . .	72
4.2.1	Sparse Dimension Reduction Map . . . . .	73
4.2.2	Subsampled Random Fourier Transform . . . . .	73
4.3	Sketched Rayleigh-Ritz . . . . .	74
4.3.1	Whitening . . . . .	76

4.3.2	Stabilization . . . . .	77
4.4	Lanczos with Sketched Rayleigh-Ritz . . . . .	78
4.4.1	Thick-Restarted Lanczos with sRR . . . . .	82
4.5	Experiments with Lanczos . . . . .	83
4.5.1	Complexity Analysis of Unrestarted Lanczos . . . . .	85
4.5.2	PRIMME Experiments . . . . .	87
4.5.3	Convergence of the Ritz Pairs . . . . .	89
4.5.4	Run Times . . . . .	92
4.6	Generalized Davidson with Sketched Rayleigh-Ritz . . . . .	94
4.7	Experiments with Generalized Davidson . . . . .	97
4.7.1	Complexity Analysis of Generalized Davidson . . . . .	97
4.7.2	PRIMME Experiments . . . . .	99
4.7.3	Convergence of the Ritz Pairs . . . . .	101
4.7.4	Timing Comparisons . . . . .	103
4.8	Chapter Summary . . . . .	106
<b>5</b>	<b>Conclusion</b>	<b>108</b>
5.1	Summary of Contributions . . . . .	108
<b>A</b>	<b>Proving the Lower Bounds on the Number of Colors Needed For Coloring a Lattice with Displacements</b>	<b>111</b>
A.1	Proof for Theorem 1 . . . . .	111
A.2	Proof for Theorem 2 . . . . .	111
A.3	Proof for Theorem 3 . . . . .	112
A.4	Proof for Theorem 4 . . . . .	112
A.5	Proof for Theorem 5 . . . . .	113
A.5.1	Case $(T, T)$ : . . . . .	114
A.5.2	Case $(S, S)$ : . . . . .	115



A.5.3 Case $(T, S)$ : . . . . .	115
A.6 Proof for Theorem 6 . . . . .	116

## ACKNOWLEDGMENTS

I would like to begin by thanking my doctoral advisor, Dr. Andreas Stathopoulos, for his support and encouragement over the past six years. He drove me to be a better and more confident researcher, and I could not have asked for a better mentor.

I am also grateful to my dissertation committee: Dr. Qun Li, Dr. Bin Ren, Dr. Pradeep Kumar, and Dr. Jennifer Loe. Thank you for your time, flexibility, and critiques when preparing this dissertation.

The College of William & Mary has provided me with an incredibly welcoming and encouraging environment, largely due to the Computer Science department administrators: Vanessa Godwin, Dale Hayes, and Jacquelyn Johnson. Thank you for all your help and advice and for giving me a safe space to talk when needed.

Thank you to those who encouraged me to pursue this degree and built me up when I had no confidence in myself. Dr. Julian Dymacek, Dr. Robert Marmorstein, and Dr. Thomas Wears, I could never have done this without you all, and you have my endless appreciation.

I have met many friends and mentors while pursuing this degree, including Steven Goldenberg, Kenneth Koltermann, Nathan Cooper, and Eloy Romero, to whom I am endlessly grateful. I am also thankful for my emotional support rodents: Peanut Butter and Jelly the gerbils, Cheeto the Syrian hamster, and Ashe the Russian dwarf hamster.

Lastly, I would like to thank my family, especially my parents, Kenneth and Jean Switzer, and my brother, Paul Switzer, for always supporting me and being my pillars during this journey, and my aunt, Dr. Maria Puccio, for being my role model since I was a child.

This dissertation is dedicated to my parents, Paul, Selina, and Biscuit.

## LIST OF TABLES

3.1	Formulas for size of the clique $ C(d, \alpha, \beta) $ , if $k > p$ and $(k + p)$ is even, with $\alpha = \lfloor \frac{k+p}{2} \rfloor$ and $\beta = \lfloor \frac{k-p}{2} \rfloor$ . If $(k + p)$ is odd, use Equation 3.13. . . . .	52
3.2	Tile sizes for each $(p, k)$ -coloring for a $32^3 \times 64$ lattice with the displacement in the first dimension (corresponding to the $z, x, y, t$ dimensions of the application). . . . .	56
3.3	Time (in seconds) to run each $(p, k)$ -coloring with tile sizes outlined in Table 3.2 and the resulting number of colors is shown in Table 3.4. . . . .	57
3.4	The first number is the smallest number of colors achieved for distance- $k$ , displacement $p$ , on the tiles of size as noted in Table 3.2. The second number is the lower bound for that $(p, k)$ from Equation 3.13. . . . .	57
3.5	The estimation of traces and variances for 1,000 RNVs run without probing for different values of $p$ and $k$ compared to probing with displacements and 10 RNVs. . . . .	66
3.6	The average percentage of neighbors at exactly distance- $k$ that do not get eliminated from the trace estimator when using a $(8, 10)$ -coloring to find other displacements. The lattice size used is $32^3 \times 64$ with a tile size of $32^4$ . . . . .	67
4.1	Operation count for each step in the unrestarted Lanczos algorithm in PRIMME when used without sketching . . . . .	87

4.2	Operation count for each step in the unrestarted Lanczos algorithm in PRIMME when used with sketching . . . . .	88
4.3	The list of matrices used for testing the unrestarted Lanczos and Gen- eralized Davidson methods with and without sketching in PRIMME. “Constructed” refers to matrices built ourselves using Matlab. . . . .	89
4.4	Operation count for each step in the Generalized Davidson algorithm in PRIMME when used without sketching . . . . .	99
4.5	Operation count for each step in the Generalized Davidson algorithm in PRIMME when used with sketching . . . . .	100

## LIST OF FIGURES

3.1	Using a shifted 1D Laplacian $A$ with 32 nodes and periodic boundary conditions, Figure 3.1a shows $A^{-1}$ permuted into color-blocks based on a distance-3 coloring before probing vectors are applied. Figure 3.1b shows the result of these color blocks after applying the probing vectors to the shifted Laplacian inverse, $A^{-1} \odot HH^T$ . . . . .	38
3.2	Applying a power and displacement to a matrix representation of a 1D toroidal lattice. The red diagonal represents the locations of the elements corresponding to the wanted displacement. . . . .	42
3.3	The neighborhood $N^2(\mathbf{0}, p, k)$ and how 1D coloring is sufficient when $p \geq k$ . . . . .	49
3.4	The distance- $k$ clique shown in grey of the neighborhood $N^3(\mathbf{0}, p, k)$ which is shown as wireframes, when $p < k$ and $(k + p)$ is even as described in Theorem 4. . . . .	50
3.5	Distance- $k$ cliques of $N^3(\mathbf{0}, p, k)$ when $(k + p)$ is odd as shown in Theorem 5. Set $C(d, \alpha, \beta)$ is the grey set in the center, set $S$ is the red hyper-surface on the right, and $T$ is the blue hyper-surface on the top. . . . .	50
3.6	The ratio of the minimum number of colors achieved with a $(p, k)$ -coloring to the theoretical lower bound in Table 3.4. Each column is a different displacement. . . . .	58

3.7	Speedups of probing with displacements over unprobed Hutchinson using each $(p, k)$ -coloring from Table 3.5 to find $\text{tr}(P_p A^{-1})$ . . . . .	61
3.8	Speedup of probing with displacements with $(p, k)$ -colorings over classical probing with $(0, k)$ -colorings to find $\text{tr}(P_p A^{-1})$ using Equation 3.16. . . . .	62
3.9	The speedups over unprobed Hutchinson for each $(p, k_p)$ -coloring to find $\text{tr}(P_n A^{-1})$ , $\forall n \in \{0, \dots, 8\}$ . . . . .	63
3.10	The relative error (Equation 3.17) for each $(p, k_p)$ -coloring used to find $\text{tr}(P_n A^{-1})$ , $\forall n \in \{0, \dots, 8\}$ . . . . .	64
4.1	Comparisons of convergence between 3-term Lanczos with sRR utilizing whitening and 3-term Lanczos with sRR utilizing stabilization to manage the condition number of the basis. . . . .	81
4.2	The convergence of 3-term Lanczos with classical RR alongside the condition number of the Krylov basis being built. . . . .	82
4.3	Comparisons of convergence between 3-term TRL with classical RR and sRR for a maximum basis size of 100. . . . .	84
4.4	The ratio of operation counts between unrestarted Lanczos with and without sketching. The top plot depicts when the number of eigenpairs sought is 100, and the bottom plot shows when the number of sought eigenpairs is 1,000. . . . .	86
4.5	The convergence of the 1st, 50th, and 100th Ritz values using the 3-term Lanczos method with sRR and the FO Lanczos method with RR when run of the matrices listed in Table 4.3. . . . .	90

4.6	The largest 100 eigenpairs returned to the user after all sought eigenpairs are marked as converged or the maximum basis size of 1,000 was reached. Results are shown for 3-term Lanczos with sketching and FO Lanczos with and without sketching for each matrix listed in Table 4.3 . . . . .	91
4.7	The breakdown in runtime for 3-term Lanczos and FO Lanczos run with and without sketching over 1,000 iterations with timings achieved from the 8-core Intel Xeon CPU E5-4627 v2 on the left and 32-core 960 Xeon Skylake on the right. . . . .	93
4.8	The ratio of operation counts between Generalized Davidson with and without sketching. The top plot depicts when the number of sought eigenpairs is 100, and the bottom plot shows when the number of sought eigenpairs is 1,000. The restarted basis size equals the number of sought eigenpairs. . . . .	98
4.9	The convergence of the targeted Ritz pair when running GD with a maximum basis size of 250 (left) and 1,200 (right) and seeking 100 Ritz pairs. . . . .	101
4.10	Comparisons of the 100 largest magnitude Ritz pairs returned from PRIMME for Generalized Davidson for a maximum basis size of 250 (right) and 1,200 (left). Only the results from matrices <code>QuadraticDiag</code> and <code>Laplacian4D</code> are shown. . . . .	103
4.11	The breakdown in average runtime per iteration for GD run with a maximum basis size of 250 (left) and 1,200 (right) when searching for 100 Ritz pairs. . . . .	104



Probing with Displacements for Variance Reduction and the  
Effectiveness of Sketched Krylov Eigenvalue Solvers

# Chapter 1

## Introduction

Scientific Computing is a field that intersects Mathematics, Computer Science, and some other discipline to develop and analyze algorithms for tackling complex numerical problems. This is done via advancements in high-performance computing, mathematical modeling, and theory. It finds application in diverse fields such as materials science, computational chemistry, computational physics, mechanical engineering, and even in social sciences such as linguistics and economics [60, 42]. Within Scientific Computing, Numerical Linear Algebra (NLA) emerges as a critical subfield that uses matrix operations and techniques to develop algorithms for effectively estimating solutions to problems. Common problems found in this field include the eigenvalue problem, the singular value decomposition, and solving systems of linear equations [98].

Direct matrix methods involving a square matrix  $A$  with  $n$  rows and columns often have computational complexities of  $O(n^3)$ . Notable examples of these direct methods include solving a system of linear equations  $Ax = b$ , accurately finding all solutions to the equation  $Ax = \lambda x$ , and finding  $A$ 's inverse, denoted  $A^{-1}$ . These methods do not scale well for matrices of large sizes, even on high-performance computing systems. Due to this, practitioners often use stochastic and iterative methods to approximate solutions more cost-effectively. This dissertation focuses on two distinct applications of these approximation methods.

The first part of this dissertation explores advancements in variance reduction techniques for computing the trace of a square matrix. The *trace problem*, which involves summing all elements in the main diagonal of a square matrix, is essential in fields such as quantum mechanics [74], signal processing [28], data mining [17], and quantum Monte Carlo [3]. The trace of a matrix  $A$ ,  $\text{tr}(A)$ , has also been shown to equal the sum of the eigenvalues of  $A$ , including all geometric and algebraic multiplicities [82]. Existing research has predominantly focused on determining the trace of a matrix function,  $f(A)$ , denoted as  $\text{tr}(f(A))$ . In most cases,  $f(A)$  is the matrix inverse ( $A^{-1}$ ), the logarithm of a matrix ( $\log A$ ), or the matrix exponential ( $\exp A$ ).

The application for our research lies in the field of Lattice Quantum Chromodynamics (LQCD) [79], where computing the trace of a matrix inverse is a critical part of the simulation and analysis of the interactions between quarks and gluons in subatomic space [65]. In this context, the Dirac operator, used to represent the behavior of quarks in space-time, is discretized into a 4-dimensional regular toroidal lattice, which can then be stored as a matrix. As previously mentioned, directly computing the inverse of a matrix with  $n$  rows and columns has a computational complexity of  $O(n^3)$  and requires  $O(n^2)$  storage in memory. Alternatively, estimating  $\text{tr}(A^{-1})$  can be reformulated as an expectation value problem, and thus stochastic methods, most notably the Hutchinson estimator, can be used to estimate this value [51].

The variance of a stochastic estimator is a measure of the variability of the estimator's mean. A high variance indicates a large dispersion of the results, requiring more samples for an accurate solution. Therefore, the Hutchinson estimator often incorporates variance reduction techniques to enhance accuracy, some of which include deflation [38], preconditioning [13], importance sampling [60], and probing [96].

*Probing* uses a coloring of the graph of the matrix  $A$  to construct *probing vectors*. These probing vectors take advantage of the structure of matrices found in LQCD to guarantee the removal of those off-diagonal elements in  $A^{-1}$  that contribute the most to the variance of the Hutchinson estimator [96]. Graph coloring assigns “colors” to vertices in a graph such

that if two vertices share an edge, they can not share a color [52, 67]. More of a variance reduction can be achieved by constructing probing vectors from a higher-distance coloring, i.e., coloring the graph of  $A^k$ , albeit with an increased computational cost. An extension of this method, *hierarchical probing*, ensures that if probing vectors constructed from a  $k$ -distance coloring does not achieve a specified reduction in variance, probing vectors can be constructed for a higher-distance coloring without discarding any previous work [89].

While these methods have proven effective for estimating  $\text{tr}(A^{-1})$ , the study of *disconnected diagrams* in LQCD necessitates the approximate sum of specific off-diagonal elements in  $A^{-1}$  corresponding to a displacement in one of the lattice dimensions [39]. In simpler terms, rather than seeking  $\text{tr}(A^{-1})$ , we now aim to approximate  $\text{tr}(PA^{-1})$ , where  $P$  represents a permutation matrix. Conventional techniques like probing and hierarchical probing have shown limited effectiveness, as they do not effectively eliminate the largest variance-causing terms from the estimator.

Our work explores novel variance reduction methods to enhance the accuracy of the Hutchinson estimator in the context of disconnected diagrams. Specifically, we introduce a probing technique that accommodates lattice displacements by adapting the graph coloring scheme. Additionally, we conduct a lower-bound analysis to determine the minimum number of colors required to color the graph of  $A$  given a coloring distance  $k$ , a single-dimension lattice displacement  $p$ , and the number of lattice dimensions  $d$ . Finally, experimental evaluations are performed to compare the effectiveness of this new method against existing approaches, leveraging the LQCD software library, Chroma [29].

The second part of this dissertation investigates the application of randomized subspace embedding techniques within Krylov-based iterative methods. Iterative methods are primarily utilized for solving systems of linear equations or approximating the eigenpairs of a matrix  $A \in \mathbb{F}^{n \times n}$ . Given a normal starting vector of length  $n$ , Krylov-based iterative methods, or *Krylov methods* for short, iteratively build a basis  $V \in \mathbb{F}^{n \times d}$  from which approximate solutions can be extracted. Without the cost of the matrix-vector multiplications, iterative methods have a computational complexity of  $O(nd^2)$ , where  $d$

represents the number of iterations performed [98, 82]. The computational complexity of the matrix-vector multiplication cost is  $O(nd)$  for sparse matrices but it grows to  $O(n^2d)$  for dense matrices. Moreover, the more ill-conditioned a matrix, the larger the number of iterations required to meet convergence. Therefore, iterative methods are particularly useful for sparse matrices and when  $d \ll n$ .

Krylov methods build a basis  $V \in \mathbb{F}^{n \times d}$  that spans the Krylov subspace of  $A$ , denoted  $\mathcal{K}_d(A, b)$ . When seeking eigenvalues,  $b \in \mathbb{F}^n$  is typically a normal random vector or an initial guess for an eigenvector (or sometimes a linear combination of the required eigenvectors). Other methods may build a space that deviates from a Krylov space with the goal of achieving faster convergence. Regardless of how  $V$  is built, when seeking a subset of the eigenpairs of  $A$ , the Rayleigh-Ritz method (RR) can extract approximations of these pairs from  $V$  [82]. However, for the RR method to be numerically stable,  $V$  must be orthonormal. Maintaining the orthonormality of the basis during its construction can be achieved using orthogonalization methods such as Householder reflections [50] or Classical Gram-Schmidt [19], each having trade-offs in speed and accuracy. These orthogonalization methods are costly but also require many synchronization points, limiting scalability in parallel computers. For these reasons, frequent applications of orthogonalization and inefficiencies in Level 1 BLAS routines can introduce bottlenecks in computation [22].

In their work [72], Nakatsukasa and Tropp highlighted the potential of incorporating randomized subspace embedding methods, also known as *sketching* methods, into RR for approximating the eigenpairs of a matrix. While sketching was traditionally used to alleviate computational costs associated with solving least-squares problems within iterative methods for linear solvers [11], [72] observed that RR can be reformulated as a least-squares problem, allowing sketching to be used in conjunction with RR. RR with sketching (sRR) enables the extraction of approximate eigenpairs of  $A$  from the basis  $V$  without  $V$  having to be orthonormal, which can alleviate the computational bottleneck of reorthogonalization techniques in Krylov methods. However, sRR introduces its own computational costs, including storing and maintaining a sketched Krylov basis. Additionally, sketching poten-

tially requires more iterations to achieve the desired accuracy of the approximate eigenpairs of  $A$ .

Based on these observations, several questions arise: (1) When does the additional computational overhead of sketching become more economical than maintaining an orthonormal Krylov basis? (2) Do certain Krylov methods work better with sRR than others? (3) On average, how many more iterations does sketching require from the Krylov method to achieve convergence comparable to the classical method?

To address these questions, we extend the C/C++ high-performance software library PRIMME [91] to incorporate the Lanczos iterative method and sRR. We then enable the use of sRR in both Lanczos and the existing Generalized Davidson method. Lanczos was selected for the simplicity of its implementation, as it iteratively constructs a basis  $V$  using only a 3-term recurrence [58], and its optimal convergence properties when approximating a small subset of the extremal eigenpairs of large, sparse, symmetric matrices.

When searching for many eigenpairs, the size of the basis built by Lanczos may become too large to store efficiently, and convergence may begin to stagnate due to instability in the method. Both of these issues can be addressed using basis restarting techniques. Therefore, we consider both the Thick-Restarted Lanczos and Unrestarted Lanczos algorithms used in conjunction with sRR [93].

After examining the Lanczos with sRR, our attention shifts to the Generalized Davidson (GD) iterative solver [26]. GD stands out as a more robust method that expands the basis with the residuals of the first unconverged pairs. Preconditioning techniques can also be used inside GD, expediting convergence. In theory, GD is identical to the Lanczos method without preconditioning.

By implementing sRR within these methods, we aim to provide insight into the trade-offs between computational efficiency, accuracy, and convergence rate associated with sketching techniques compared to their nonsketched counterparts.

The subsequent sections of this dissertation will unfold as follows:

Chapter 2: This chapter will establish essential notation conventions utilized throughout this document and introduce key concepts crucial for understanding the work described.

Chapter 3: Delving into the specifics of the first part of our research, this chapter will go into detail about our work with variance reduction techniques for computing the trace of a permuted matrix inverse in the context of Lattice Quantum Chromodynamics.

Chapter 4: The second part of our research focuses on investigating the efficiency of Krylov methods when coupled with sketched Rayleigh-Ritz. We provide insights into the advantages and limitations of these techniques.

Chapter 5: At the conclusion of this dissertation, we discuss the current status of our work, as well contributions our work offers.

## Chapter 2

# Background

In this chapter, we lay the foundation for background information pertinent to understanding the subsequent chapters of this dissertation. Section 2.1 provides mathematical and matrix notation utilized throughout this document. Following this, in Section 2.2, we define various properties of matrices and introduce the eigenvalue problem along with the singular value decomposition. Section 2.3 gives background information regarding variance reduction techniques used for trace estimation in LQCD, serving as a prelude to Chapter 3. Section 2.4 will provide a general background for Krylov-based iterative methods, with randomized subspace embeddings briefly discussed in Section 2.5. These two sections provide the foundations for Chapter 4. Their respective chapters will explore more in-depth background information, motivation, and related works.

### 2.1 Notation and Conventions

The ring of integers, the field of real numbers, and the field of complex numbers are denoted as  $\mathbb{Z}$ ,  $\mathbb{R}$ , and  $\mathbb{C}$  respectively. Restricting these to the sets of their non-negative values will be indicated using a “+” subscript:  $\mathbb{Z}_+$ ,  $\mathbb{R}_+$ , and  $\mathbb{C}_+$ . Similarly, restricting to their negative values will be denoted with a “-” subscript. The double-struck capital letter  $\mathbb{F}$  will be used as a placeholder to represent a general ring or field throughout this document.



Scalar values and vectors are characterized as italicized lowercase letters, e.g.,  $a$  or  $x$ . Scalars are defined using the notation  $a \in \mathbb{F}$ , signifying the element  $a$  resides in the ring/field  $\mathbb{F}$ . Similarly, vectors are defined with the notation  $x \in \mathbb{F}^n$ , meaning  $x$  is an element of the  $n$ -th dimensional vector space  $\mathbb{F}^n$ . In terms of data structures,  $n$  is the *length* of the vector  $x$ , written using MATLAB-style notation as `len(x)`. Matrices are represented using italicized capital letters, e.g.,  $A$ .  $A \in \mathbb{R}^{n \times m}$  denotes the matrix  $A$  consisting of  $n$  columns and  $m$  rows with all entries being real, with  $n \times m$  being the *size* of  $A$ , written `size(A)` using MATLAB notation. The  $i$ -th entry in a vector  $x$  is denoted as  $x_i$ , and the  $(i, j)$  entry of a matrix  $A$  is denoted  $A_{ij}$ . For consistency, indices will start at one and continue to the variable representing the size, e.g.,  $i = 1, \dots, n$ .

The *transpose* of a matrix or vector is denoted using the superscript  $T$ , i.e.,  $A^T$ , while the superscript  $H$  signifies the *Hermitian transpose*, also known as the conjugate transpose. If  $A \in \mathbb{R}^{n \times n}$  and  $A = A^T$ , it is considered *symmetric*. Similarly, if  $A \in \mathbb{C}^{n \times n}$  and  $A = A^H$ , it is considered *Hermitian*. When all entries of a matrix  $A$  are in the field of real numbers,  $A^T = A^H$ .

*Machine epsilon*, written  $\epsilon_{\text{mach}}$ , is the distance between 1 and the next largest floating point number in a computing system. When using double-precision floating point numbers, typically  $\epsilon_{\text{mach}} \approx 10^{-16}$ .

## 2.2 Matrix Properties and Decompositions

By default, we consider all vectors to be column vectors. Therefore, when dealing with two vectors of the same length,  $x \in \mathbb{F}^n$  and  $y \in \mathbb{F}^n$ , their *Euclidean inner product*, also known as the “dot product”, is  $x^H y$ , and their *Euclidean outer product* is  $x y^H$ . The inner product of two vectors should result in a single scalar value, while the outer product of two vectors of length  $n$  results in an  $n \times n$  matrix.

Associated with this inner product is the *Euclidean norm*, also referred to as the vector 2-norm [98], which is defined as

$$\|x\|_2 = \sqrt{x^H x} = \left( \sum_{i=1}^n |x_i|^2 \right)^{\frac{1}{2}}. \quad (2.1)$$

The Frobenius norm of a matrix  $A \in \mathbb{F}^{m \times n}$  is similarly defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m |A_{ij}|^2}, \quad (2.2)$$

while the 2-norm of a matrix is

$$\|A\|_2 = \max_{\mathbf{0} \neq x \in \mathbb{F}^n} \left( \frac{\|Ax\|_2}{\|x\|_2} \right) = \sigma_{max} \quad (2.3)$$

where  $\sigma_{max}$  represents the maximum singular value of  $A$ . Further details on the singular value decomposition of a matrix will be provided in Subsection 2.2.1.1.

The number of nonzero elements of a matrix  $A$  is  $\mathbf{nnz}(A)$ , with its *sparsity structure* referring to the list of matrix indices containing nonzero elements, without storing the values of those nonzeros. A *sparse matrix* refers to a matrix containing a significant number of zero elements and can use specialized storage and computational techniques to gain improved performance. In contrast, *dense matrices* do not achieve a performance benefit using sparse techniques.

The *trace* of a square matrix  $A$ , denoted  $\mathbf{tr}(A)$ , is defined as the sum of its principle (or “main”) diagonal elements. Specifically, if  $A \in \mathbb{F}^{n \times n}$ ,

$$\mathbf{tr}(A) = \sum_{i=1}^n A_{ii}. \quad (2.4)$$

Moreover,  $\mathbf{tr}(A)$  can be proven to equal the sum of the eigenvalues of  $A$ , including all geometric and algebraic multiplicities [82].

The number of linearly independent columns of matrix  $A$  is termed the *rank* of  $A$ , which also denotes the dimension of the range of  $A$  [82]. If the rank of a matrix  $A \in \mathbb{F}^{m \times n}$  is equal to  $\min(m, n)$ , then the matrix is deemed *full rank*; otherwise, it is considered *rank-deficient*.

### 2.2.1 The Eigenvalue Problem

The eigenvalue problem is prevalent across a variety of scientific disciplines, including computational physics [97], structural engineering [85], and machine learning [18]. Consider the equation

$$Ax = \lambda x, \tag{2.5}$$

where  $A \in \mathbb{F}^{n \times n}$ . Any vector  $x \in \mathbb{F}^n$  and associated scalar  $\lambda \in \mathbb{F}$  that satisfy this equation constitute a *right eigenpair* of  $A$ . Here,  $x$  represents a *right eigenvector* and  $\lambda$  represents an *eigenvalue*. Similarly, any  $(y, \lambda)$  combination that solves the equation

$$A^H y = \bar{\lambda} y, \tag{2.6}$$

is considered a *left eigenpair* of  $A$ . The left and right eigenvectors correspond to the same set of eigenvalues. In this work, when we refer to an “eigenpair” of a matrix, we refer to the right eigenpairs unless otherwise specified.

Accurately computing all eigenpairs of matrix  $A \in \mathbb{F}^{n \times n}$  has a computational complexity of  $O(n^3)$  and a memory storage cost of  $O(n^2)$  [98]. This renders direct methods impractical for large-sized matrices. However, in most cases, users are interested in obtaining only a subset of approximate eigenpairs. This motivation has led to the development of more cost-effective Krylov-based iterative methods.

Krylov-based iterative methods, or “Krylov methods”, are used to iteratively construct a basis  $V \in \mathbb{F}^{n \times d}$  that consists of  $d$  linearly independent columns spanning a  $d$ -dimensional

*Krylov subspace*, denoted

$$\mathcal{K}_d(A, b) = \text{span}\{b, Ab, A^2b, A^3b, \dots, A^{d-1}b\}, \quad (2.7)$$

where  $b \in \mathbb{F}^n$  is some normal vector. Approximations to the eigenpairs of  $A$  and solutions to linear systems can then be extracted from  $\text{span}(V)$ .

The *backward error* of an approximate eigenpair measures how well the estimation satisfies Equation 2.5. If we let  $(\hat{x}, \hat{\lambda})$  be an approximate eigenpair of  $A$  with  $\|\hat{x}\| = 1$ , the backwards error is computed as

$$\text{Backward Error} = \|A\hat{x} - \hat{\lambda}\hat{x}\|/\|A\|. \quad (2.8)$$

The smaller this value is, the better the approximation of the eigenpair. The vector  $A\hat{x} - \hat{\lambda}\hat{x}$  is referred to as the *residual* of that particular eigenpair approximation. Further discussion on this topic is provided in Section 2.4.

*Deflation* is a technique used to eliminate an eigenpair from the range of  $A$ . This may be used to improve the conditioning of the matrix for various problems, e.g., to accelerate the convergence of an iterative method [83]. Suppose  $x_i$  is the right eigenvector to be removed from  $A$ , with  $y_i$  being its corresponding left eigenvector. If  $A$  is a symmetric matrix, *orthogonal deflation* can be used to construct the deflated operator  $\hat{A}$ .

$$\hat{A} = (I - x_i x_i^H) A (I - x_i x_i^H). \quad (2.9)$$

Orthogonal deflation can also be used if  $A$  is nonsymmetric, but  $x_i^H x_j \neq 0$  and the other eigenpairs of  $A$  will be changed. Instead, we can use *spectral deflation* to form the operator

$$\hat{A} = (I - x_i y_i^H) A (I - x_i y_i^H). \quad (2.10)$$

In general, spectral deflation has better theoretical properties, but orthogonal deflation may be more stable [83].

### 2.2.1.1 The Singular Value Decomposition

The singular value decomposition (SVD) factorizes a matrix  $A \in \mathbb{F}^{m \times n}$  into the form

$$A = U\Sigma V^H \quad (2.11)$$

where  $U \in \mathbb{F}^{m \times m}$  and  $V \in \mathbb{F}^{n \times n}$  are both *unitary* matrices, satisfying  $U^H U = U U^H = I$  and  $V^H V = V V^H = I$ , and  $\Sigma \in \mathbb{F}^{m \times n}$  is a rectangular diagonal matrix [54]. The columns of  $U$  and  $V$  constitute the left and right *singular vectors* of  $A$ , respectively. The diagonal entries of  $\Sigma$ , denoted  $\sigma_1, \sigma_2, \dots, \sigma_{\min(n,m)}$ , are the *singular values* of  $A$ , where

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(n,m)}. \quad (2.12)$$

If  $A$  has a matrix rank of  $r$ , then the columns of  $U(:, 1 : r)$  and  $V(:, 1 : r)$  form orthonormal bases for the column and row spaces of  $A$ , respectively. The remaining columns of  $U$ ,  $U(:, r + 1 : m)$ , constitute an orthonormal basis for the left nullspace  $N(A^H)$ , while the remaining columns of  $V$ ,  $V(:, r + 1 : n)$ , form the orthonormal basis for the nullspace  $N(A)$ . If  $u_i$  represents the  $i$ -th left singular vector of  $A$ ,  $v_i$  denotes the  $i$ -th right singular vector, and  $\sigma_i$  signifies the  $i$ -th singular value, then  $(u_i, \sigma_i, v_i)$  is referred to as a *singular triplet* of  $A$ .

In many practical applications,  $A$  is a rectangular matrix with dimensions  $m \times n$ . In these cases, the thin or *economy* SVD is used [98]. If we let  $k = \min(m, n)$ , or the rank of  $A$ , then the economy SVD is depicted as:

$$A_k = U_k \Sigma_k V_k^H, \quad (2.13)$$

where  $U_k \in \mathbb{F}^{m \times k}$  is the first  $k$  columns of  $U$ ,  $\Sigma_k \in \mathbb{F}^{k \times k}$  is a diagonal matrix with the first  $k$  singular values as entries, and  $V \in \mathbb{F}^{n \times k}$  is the matrix containing the first  $k$  columns of  $V$ . Given a rank- $k$  approximation of  $A$  using the  $k$  first singular triplets

$$A \approx A_k = \sum_{j=1}^k \sigma_j u_j v_j^H, \quad (2.14)$$

$A_k$  is the closest matrix approximation to  $A$  of rank  $k$ , i.e., the distance between  $A$  and  $A_k$  in Euclidean space,  $\|A - A_k\|_F$ , is minimized [98].

The SVD can also be used to compute the spectral, L2, or Euclidean *condition number* of a matrix  $A$ , denoted as  $\kappa(A)$ , using

$$\kappa(A) = \|A\| \cdot \|A^{-1}\| = \frac{\sigma_1}{\sigma_{\min(n,m)}}. \quad (2.15)$$

$\kappa(A)$  quantifies the sensitivity of several matrix operations, including matrix-vector multiplications (MatVecs) and solutions of linear systems, to changes in the coefficients of the matrix [23].

The operator  $A$  can also be deflated by eliminating the  $i$ -th singular triplet from its range,

$$\hat{A} = (I - u_i v_i^H) A (I - u_i v_i^H), \quad (2.16)$$

where  $\hat{A}$  is the deflated operator.

## 2.3 Variance Reduction Techniques in LQCD

Lattice Quantum Chromodynamics (LQCD) has emerged as a pivotal approach for addressing problems within Quantum Chromodynamics (QCD) in a non-perturbative manner [79, 65]. QCD, a branch of theoretical physics, studies the interactions between quarks and gluons within subatomic space. LQCD operates by discretizing continuous space-time into a structured 4-dimensional regular lattice, where each point has eight neighboring nodes,

two for each of the four dimensions. In this lattice, the first three dimensions represent space, designated as the  $x$ ,  $y$ , and  $z$  dimensions, while the fourth dimension represents time, labeled as the  $t$  dimension.

In the realm of LQCD, each quark in subatomic space can be labeled with one of three colors (red, green, or blue) and exhibit one of four spin directions, resulting in twelve spin-color combinations. Consequently, each lattice point contains twelve degrees of freedom. When the 4-dimensional lattice is represented as an adjacency matrix, each nonzero index corresponds to a  $12 \times 12$  unitary matrix to account for these spin-color combinations [15]. This matrix is the foundation for numerous numerical computations revolving around linear solves, including trace estimations, integrators, and high-dimensional sampling.

### 2.3.1 Finding the Trace of a Matrix Inverse

A fundamental computation in LQCD involves calculating the trace of the inverse of a matrix with a sparsity structure corresponding to a 4-dimensional toroidal lattice. However, directly computing the inverse of matrix  $A \in \mathbb{F}^{n \times n}$  takes  $O(n^3)$  operations, necessitating the use of stochastic methods. One such method, referred to as the *Hutchinson estimator*, states that

$$\text{tr}(A^{-1}) = \mathbb{E}(z^T A^{-1} z), \quad (2.17)$$

where  $z \in \mathbb{F}^n$  with  $\|z\| = \sqrt{n}$  and independent and identically distributed (i.i.d.) elements.

When evaluating the effectiveness of a stochastic estimator such as 2.17, it is crucial to consider its *variance*, which represents the variability of the estimation. A significant variance indicates a large dispersion of results from the estimator, meaning more samples, and thus computations, are needed to estimate the trace accurately.

In [12], the authors investigate various strategies for constructing vectors  $z$  to minimize the variance of Equation 2.17. One strategy utilizes *Rademacher vectors*, where each element in the vector takes the values  $\pm 1$  with a probability of 0.5 if using real numbers, or  $\pm 1, \pm i$  with a probability of 0.25 if using complex numbers. Using these vectors, the

trace of  $A^{-1}$  can be estimated by using  $s$  randomly generated Rademacher vectors in the Hutchinson estimator,

$$\mathrm{tr}(A^{-1}) \approx \frac{1}{s} \sum_{j=1}^s z_j^T A^{-1} z_j, \quad (2.18)$$

with variance [12]

$$\mathrm{Var}(z^T A^{-1} z) = 2(\|A^{-1}\|^2 - \sum_{i=1}^n A_{ii}^2). \quad (2.19)$$

Utilizing Rademacher vectors in the Hutchinson estimator has yielded half the variance compared to using vectors from the normal distribution.

In Equation 2.18,  $A^{-1}z$  is never explicitly computed. Instead, an iterative method such as Generalized Minimum Residual (GMRES) approximately solves the linear system

$$Ax_j = z_j \text{ where } j = 1, 2, \dots, s \quad (2.20)$$

for  $x_j \in \mathbb{F}^n$  [82]. Solving these linear systems is the most computationally expensive part of the Hutchinson estimator, meaning we want to minimize the number of solutions required to approximate  $\mathrm{tr}(A^{-1})$ . When the vectors  $z$  reduce the estimator's variance, fewer samples  $s$  are needed to get a decent trace approximation, and, therefore, fewer linear solutions are required.

More modern algorithms, such as `Hutch++`, have been presented that use a randomized method to compute a  $(1 \pm \epsilon)$  approximation of  $\mathrm{tr}(A)$  when  $A$  is definite [66, 78]. `Hutch++` was later expanded to work for indefinite matrices in [24]. Epperly et al. introduced `XTrace` and `XNyzTrace`, which use an exchangeability principle to achieve errors in the estimator that are orders of magnitude smaller than those obtained with `Hutch++` [30]. These methods can not be utilized for our purposes because we find  $\mathrm{tr}(A^{-1})$  using Krylov-based iterative methods for  $A^{-1}$ . Due to this, reserving a few  $z$  to compute a good deflation space is unnecessary.

Equation 2.19 suggests that off-diagonal elements of larger magnitude in  $A^{-1}$  contribute most significantly to the variance of the estimator. For many sparse matrices, including



those in LQCD, the elements in  $A_{i,j}^{-1}$  exhibit Green’s function decay, where their magnitudes display exponential decay relative to the distance between nodes  $i$  and  $j$  in the graph of  $A$  [106]. Leveraging this knowledge, *probing* is a method that uses a  $k$ -distance graph coloring [52] of  $A$  to construct a set of orthogonal *probing vectors*. Assuming  $m$  is the number of colors required to perform a  $k$ -distance coloring of the graph of  $A$ , then we can form the matrix  $H \in \mathbb{Z}^{n \times m}$ , where each column in  $H$  is a different probing vector.

Once  $H$  has been constructed, prominent error-causing elements of  $A^{-1}$  are eliminated from the estimator in Equation 2.18 by replacing the operator  $A^{-1}$  with  $A^{-1} \odot HH^T$ , where  $\odot$  denotes the element-wise product. Replacing the operator thereby reduces the estimator’s variance. A higher-distance coloring distance equates to a greater reduction, although at the expense of increased computational cost resulting from more linear solves [96].

One drawback of probing is that the constructed probing vectors are discarded if a specific  $k$ -distance coloring does not achieve a targeted variance reduction. New probing vectors are then built from a higher-distance coloring. *Hierarchical probing* is a technique that extends probing by applying a recursive coloring technique, allowing an increase in the coloring distance without discarding prior work [89, 57]. Further elaboration on these methodologies is provided in Chapter 3.

## 2.4 Krylov-based Iterative Methods

Krylov methods, initially introduced in 1931 by Aleksey Krylov for solving systems of linear equations, were developed on the insight that repeated applications of a matrix  $A$  to a starting vector  $b$  generates a subspace from which valuable information can be extracted. In exact arithmetic, Krylov methods such as Arnoldi and Lanczos explicitly build a *Krylov basis*  $V \in \mathbb{F}^{n \times d}$  where  $V$  serves as a basis for the Krylov subspace  $\mathcal{K}_d(A, b)$ , defined in Equation 2.7 [82]. However, due to numerical instability caused by floating-point operations, the basis  $V$  may deviate from an exact Krylov basis. Once  $V$  is built, it can be

used to extract approximate eigenpairs of  $A$ , or to find approximations of  $f(A)b$  for some matrix function  $f$  [82]. Examples include  $f(A) = A^{-1}$  for a linear system of equations, or  $f(A) = \exp(A)$ , i.e., the matrix exponential.

Several subspace projection methods exist to extract information from the Krylov basis, such as Galerkin [19], Petrov-Galerkin [32], Harmonic Ritz [70], and Refined Ritz [63], which can be utilized for a broad spectrum of problems. Galerkin methods add the constraint that the residuals of the extracted solutions must be orthogonal to the basis  $V$ . This work focuses on the Galerkin Rayleigh-Ritz (RR) method for extracting information.

A Galerkin method is one that has the constraint that the error of an approximate solution orthogonal to a given subspace. In the context of RR, this means that the residual of the extracted approximate eigenpairs are orthogonal to  $V$  [83]. Named after Lord Rayleigh and Walter Ritz, RR achieves orthogonal residuals by extracting approximate eigenpairs of a matrix  $A \in \mathbb{F}^{n \times n}$  from the basis  $V \in \mathbb{F}^{n \times d}$  by solving the smaller, more manageable eigenproblem on the matrix  $V^H A V \in \mathbb{F}^{d \times d}$ . This smaller eigenproblem yields eigenvectors  $\hat{x}_i$  and corresponding eigenvalues  $\hat{\lambda}_i$  of  $V^H A V$  for  $i = 1, 2, \dots, d$ . The resulting estimated eigenpairs of  $A$ , referred to as the *Ritz pairs*, are computed as  $(V \hat{x}_i, \hat{\lambda}_i)$ , and their residuals  $AV \hat{x}_i - V \hat{x}_i \hat{\lambda}_i$  are orthogonal to the basis  $V$  [83]. Algorithm 1 shows the RR procedure.

**Algorithm 1:** The Rayleigh-Ritz Method**Input:** $A \in \mathbb{F}^{n \times n}$  = The matrix we are searching for the eigenpairs of $V \in \mathbb{F}^{n \times d}$  = The Krylov basis of  $A$  $H \in \mathbb{F}^{d \times d} = V^H A V$ **Output:** $X \in \mathbb{F}^{n \times d}$  = The Ritz vectors of  $A$  $\Lambda \in \mathbb{F}^d$  = The Ritz values of  $A$ resNorms  $\in \mathbb{F}^d$  = The residual norms of the eigenpair estimatesRayleigh-Ritz( $A, V, H$ )1  $[Y, \Lambda] = \mathbf{eig}(H)$ ;2  $X = VY$ ;

# Compute the residuals

3 resNorms( $i$ ) =  $\|AX_i - X_i\Lambda_i\| / \|X(:, i)\|$  for  $i = 1, 2, \dots, d$ 4 **return**  $[X, \Lambda, \text{resNorms}]$ ;

Krylov methods also play an important role in approximating matrix functions using matrix polynomials. Given  $A \in \mathbb{F}^{n \times n}$ , a matrix polynomial takes the form

$$p(A) = a_0 I + a_1 A + a_2 A^2 + \cdots + a_d A^{d-1}, \quad (2.21)$$

for some degree  $d$  and with  $I \in \mathbb{Z}^{n \times n}$  being an identity matrix.

According to the Stone-Weierstrass theorem, any continuous function  $f(x)$ , such as  $A^{-1}$ ,  $\exp(A)$ , and  $\log(A)$ , can be approximated using a polynomial  $p(x)$  [95]. The higher the polynomial degree, the better the approximation of  $f(x)$  achieved. Examples include the Bernstein polynomial, the Padé approximation, and the Fourier series [61, 99].

Like most direct methods, computing  $f(A)$  accurately has a cost of  $O(n^3)$ , necessitating the use of the approximations. A Krylov space of degree  $d$  can generate a degree  $d$  polynomial with “optimal” coefficients given by RR or other Galerkin methods.

In this work, we use the RR method to extract approximate eigenpairs of a matrix  $A$  from bases built by the Lanczos and Generalized Davidson Krylov-based iterative methods.

### 2.4.1 Lanczos

In 1951, the Arnoldi algorithm was developed to reduce a dense matrix  $A \in \mathbb{F}^{n \times n}$  into its Hessenberg form

$$H = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n-1} & a_{2,n} \\ 0 & a_{3,2} & a_{3,3} & \cdots & a_{3,n-1} & a_{3,n} \\ 0 & 0 & a_{4,3} & \cdots & a_{4,n-1} & a_{4,n} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n,n-1} & a_{n,n} \end{bmatrix} \quad (2.22)$$

using unitary similarity transforms<sup>1</sup> [10, 82]. If the original matrix  $A$  is Hermitian, its Hessenberg form is tridiagonal. [10] noted that the eigenpairs of  $H \in \mathbb{F}^{d \times d}$ , where  $d$  is

<sup>1</sup>Equation 2.22 shows an *upper Hessenberg* matrix. A *lower Hessenberg* matrix is similar but has zeros above the first super-diagonal, with nonzeros elsewhere.

the number of Arnoldi iterations performed and  $d < n$ , could be used to approximate eigenpairs of  $A$  on the outer parts of the spectrum.

The Lanczos algorithm provides a simplified approach to the Arnoldi method for dealing with symmetric matrices [82]. As shown in Algorithm 2, it employs a 3-term recurrence to construct a basis  $V \in \mathbb{F}^{n \times d}$  alongside a tridiagonal matrix  $H \in \mathbb{F}^{d \times d}$  of the form

$$H = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & \cdots & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{d-1} & \alpha_{d-1} & \beta_d \\ & & & \beta_d & \alpha_d \end{bmatrix} = V^H A V. \quad (2.23)$$

This tridiagonal form of  $H$  implies that that basis  $V$  can be formed with a 3-term recurrence, avoiding the need for full orthogonalization of  $V$ .

Furthermore, the Arnoldi and Lanczos methods satisfy

$$A V_i = V_{i+1} \bar{H} \quad (2.24)$$

where

$$\bar{H} = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & \cdots & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{i-1} & \alpha_{i-1} & \beta_i \\ & & & \beta_i & \alpha_i \\ & & & & \beta_{i+1} \end{bmatrix}. \quad (2.25)$$

This means that the residual norms of the approximate eigenpairs of  $A$  can be computed at the  $i$ -th iteration using the formula

$$\|A V \hat{x}_j - V \hat{x}_j \hat{\lambda}_j\|_2 = \beta_{i+1} e_n^T \hat{x}_j \quad \text{for } j = 1, 2, \dots, i, \quad (2.26)$$

where  $e_n \in \mathbb{N}^n$  is  $n$ -th orthocanonical basis vector.

**Algorithm 2:** The Lanczos Algorithm with Rayleigh-Ritz**Input:**

- $A \in \mathbb{F}^{n \times n}$  = A square, Hermitian matrix  
 $y \in \mathbb{F}^n$  = An initial starting column vector of norm 1  
 $d_{\max}$  = The maximum size of the Krylov basis  
 $e$  = Number of approximate eigenpairs being sought  
 $\text{tol}$  = The convergence tolerance

**Output:**

- $X \in \mathbb{F}^{n \times d}$  = The Ritz vectors of  $A$        $\Lambda \in \mathbb{F}^d$  = The Ritz values of  $A$   
 $\text{resNorms} \in \mathbb{F}^d$  = The residual norms for the eigenpair estimates of  $A$

**Lanczos**( $A, y, d$ )

```

1  $n = \text{size}(A, 1)$ ,     $H = \text{zeros}(n, n)$ ;
2  $V(:, 1) = y$ ;
3  $\hat{w} = AV(:, 1)$ ;
4  $H(1, 1) = \hat{w}^H V(:, 1)$ ;
5  $w = \hat{w} - H(1, 1) * V(:, 1)$ ;
6 for  $i = 2:d_{\max}$  do
7    $H(i, i - 1) = H(i - 1, i) = \|w\|_2$ ;
8    $V(:, i) = w / H(i, i - 1)$ ;
9    $\hat{w} = AV(:, i)$ ;
10   $H(i, i) = \hat{w}^H V(:, i)$ ;
11   $w = \hat{w} - H(i, i)V(:, i) - H(i, i - 1)V(:, i - 1)$ ;
    # Periodically check for convergence
12   $[X, \Lambda, \text{resNorms}] = \text{Rayleigh-Ritz}(A, V, H)$ ;
13  If  $\text{resNorms}(1:e) < \text{tol}$  then return  $[X(:, 1:e), \Lambda(1:e), \text{resNorms}(1:e)]$ 
14  $[X, \Lambda, \text{resNorms}] = \text{Rayleigh-Ritz}(A, V, H)$ ;
15 return  $[X(:, 1:e), \Lambda(1:e), \text{resNorms}(1:e)]$ 

```

The Lanczos process outlined in Algorithm 2 shows the construction of the basis  $V \in \mathbb{F}^{n \times d}$  and its corresponding tridiagonal matrix  $H \in \mathbb{F}^{d \times d}$ . The new basis vector is built and added to  $V$  through the 3-term recurrence shown on line 11. Following the construction of  $V$  and  $H$ , line 14 invokes the Rayleigh-Ritz algorithm (see Algorithm 1) to extract the Ritz pairs of  $A$ . The user should periodically perform RR intermittently while constructing  $V$  to monitor convergence and prevent unneeded iterations, as shown on lines 12 and 13 in Algorithm 2.

Theoretically, the constructed basis  $V \in \mathbb{F}^{n \times d}$  should always be orthonormal, i.e.,  $V^H V = I$ . However,  $V$  loses orthogonality in floating point arithmetic, resulting in repeated directions of converged eigenvectors entering the basis and an increase in  $V$ 's condition number,  $\kappa(V)$ . Due to this, RR may yield inaccurate solutions and many copies of converged eigenpairs or even fictitious eigenvalues (*ghost eigenvectors*). Consequently, convergence will slow down or stagnate, and numerical instability will increase [98]. To address this challenge, practitioners adopt strategies such as *full-orthogonalization*, *selective orthogonalization*, *locking*, and *partial reorthogonalization* [77]. Several orthogonalization methods exist, with two common examples being the Classical Gram-Schmidt (CGS) process [19], Modified Gram-Schmidt (MGS) [82], and Householder reflections [50]. Each method comes with trade-offs in performance and accuracy.

Full-orthogonalization Lanczos explicitly orthogonalizes new basis vectors against all preceding vectors in the basis. In the context of Algorithm 2, this equates to adding an additional step after line 11 inside the for loop that orthogonalizes the vector  $w$  against  $V(:, 1:i)$ . This is identical to the more generalized Arnoldi iterative method when dealing with symmetric matrices, except that Arnold does not perform the 3-term recurrence as the initial orthogonalization step [10].

Selective orthogonalization was developed based on the observation that orthogonality loss can be attributed to the convergence of a Ritz pair, meaning the computational cost of full-orthogonalization Lanczos can be avoided by orthogonalizing incoming basis vectors against the set of converged Ritz vectors [44, 77]. In a 1979 paper by Parlett and Scott



[76], they classified a Ritz pair  $(\hat{x}, \hat{\lambda})$  of  $A \in \mathbb{F}^{n \times n}$  as “good enough” to be marked as converged if

$$\|A\hat{x} - \hat{x}\hat{\lambda}\|_2 \approx \sqrt{\epsilon_{\text{mach}}}\|A\|_2. \quad (2.27)$$

*Soft locking* is a technique that orthogonalizes new basis vectors against the Ritz vectors flagged as converged. This prevents repeated directions of the corresponding Ritz pairs from entering the basis while allowing improvement to their accuracies, as they will still be used in the RR extraction [55, 87]. The Lanczos method can not use soft locking as the  $V$  is not rotated to the eigenvectors. Instead, Lanczos can use *hard locking*, which orthogonalizes all new basis vectors against the converged Ritz vectors and deflates the converged pairs from the search space, so their accuracy never improves but will never become unconverged either.

Finally, partial reorthogonalization was developed on the insight that Lanczos vectors do not need to be entirely orthogonal to produce accurate eigenpair estimates [86]. Instead of orthogonalizing incoming basis vectors against all preceding basis vectors or a set of converged Ritz vectors, they are orthogonalized against a small subset of the most recently added vectors.

*Restarted Lanczos* is used to overcome memory constraint challenges and mitigate orthogonalization issues. When the number of columns in the basis  $V$  reaches a predefined threshold  $d_{\text{max}}$  and not all desired eigenpairs have converged, restarting is performed [83]. The standard Lanczos restarting technique involves discarding the entire basis and reentering the Lanczos process using a new initial vector. Discarding this much convergence information will likely result in slower convergence [44], motivating the development of various alternative restarting strategies. One notable strategy is Thick-Restarted Lanczos (TRL) [103].

TRL explicitly restarts the basis with multiple Ritz vectors, hence the term “thick”. It is equivalent to the Implicitly Restarted Arnoldi (Lanczos) method [20] but simplifies the restarting implementation while enabling more information retention, thereby minimizing

the loss of convergence. In this method, the Ritz vectors used in the restarted basis correspond to the Ritz values sought by the user, allowing the most significant information to be retained.

*Block Lanczos* extends the traditional Lanczos algorithm by replacing the single vector  $b$  in Equation 2.7 with a set of  $k$  orthonormal vectors  $B \in \mathbb{F}^{n \times k}$ , with  $k$  denoting the *block size* [43]. At each iteration in the Krylov method,  $k$  additional vectors are incorporated into the basis. This approach offers several advantages, including increased parallelism, better single-node performance, and enabling the approximation of clustered eigenpairs or multiplicities, which, in theory, can not be accomplished using single vector Lanczos. However, it does increase the total number of operations and requires special treatment when some block directions become colinear.

#### 2.4.1.1 Generalized Davidson

During the late 1970s, the Davidson algorithm was introduced as a variant of Lanczos that allowed for diagonal preconditioning to accelerate convergence [26]. *Preconditioning* is a technique used to accelerate the convergence of an iterative method [83], and an extensive list and explanations of various preconditioning techniques can be found in Ke Chen’s 2005 book [21].

An extension of Davidson, known as Generalized Davidson (GD), was later presented to allow the use of general preconditioners [69]. Without preconditioning, Davidson and GD are equivalent to the Lanczos method in exact arithmetic but differ from the beginning when using floating point operations. When used with preconditioning, Davidson and GD construct bases that no longer correspond to a Krylov subspace but can still drastically accelerate convergence [68].

Unlike conventional Krylov methods such as Lanczos, where the next basis vector results from multiplying the previous one by the input matrix  $A \in \mathbb{F}^{n \times n}$ , GD expands its basis  $V$  using the residual of the first unconverged eigenpair (or  $k$  eigenpairs when using Block GD), essentially “targeting” it. These new basis vector(s) are then orthogonalized against

all existing vectors [82, 19]. Once the number of columns in the basis  $V$  reaches some predefined threshold  $d_{\max}$  without all sought approximate eigenpairs converging, restarting sets  $V$  equal to the first  $r$  Ritz vectors before proceeding. Algorithm 3 is provided for a more detailed depiction of the GD algorithm.

**Algorithm 3:** The Generalized Davidson Algorithm**Input:** $A \in \mathbb{F}^{n \times n}$  = A square, Hermitian matrix $Y \in \mathbb{F}^{n \times e}$  = Matrix of initial orthonormal column(s) $r$  = The size of the restarted basis $e$  = The number of eigenpairs sought after

tol = Convergence tolerance

**Output:** $X \in \mathbb{F}^{n \times d}$  = The Ritz vectors of  $A$  $\Lambda \in \mathbb{F}^d$  = The Ritz values of  $A$ resNorms  $\in \mathbb{F}^d$  = The residual norms for the eigenpair estimates of  $A$ Generalized\_Davidson( $A, Y, r, e, tol$ )

```
1  $V(:, 1:e) = Y; \quad W(:, 1:e) = AY;$ 
2 for  $m = 2, 3, \dots$  do
3    $W(:, m) = AV(:, m);$ 
4    $[X, \Lambda] = \text{Rayleigh-Ritz}(A, V(:, 1:m), V(:, 1:m)^H W(:, 1:m));$ 
5   residuals =  $WX(:, i) - VX(:, i)\Lambda(i)$  for  $i = 1, 2, \dots, m;$ 
6   resNorms( $i$ ) =  $\| \text{residuals}(:, i) \|$  for  $i = 1, 2, \dots, m;$ 
7   target = find(resNorms > tol, "first", 1); # Target 1st unconverged vector
8   if target >  $e$  then
9     return  $[X, \Lambda, \text{resNorms}]$ 
10  if  $m > d$  then
11     $V(:, 1:r) = X(:, 1:r); \quad W(:, 1:r) = AV(:, 1:r);$  # Restart
12     $m = r;$ 
    # Precondition new vector before orthogonalizing it against all
    previous basis vectors
13   $V(:, m+1) = \text{cgs}(V(:, 1:m), \text{Precondition}(\text{residuals}(:, \text{target})));$ 
```

In 1992, Murray et al. proposed an extension of the Davidson algorithm, now known as  $+k$ , which modifies the restarting technique by not only including the first  $r$  Ritz vectors from the current iteration into the restarted basis but also  $k$  Ritz vectors from the previous iteration [71]. Stathopoulos and Saad introduced initial analysis and efficient implementation of this method in [92], which was more recently expanded upon in [104]. Studies in 2005 [88] and 2007 [90] further investigate the effectiveness of GD+k when searching for one or many eigenpairs, respectively. While GD+k does accelerate the convergence of the sought Ritz pairs, the resulting basis no longer corresponds to a Krylov subspace, similar to preconditioning.

While one step of GD is computationally more expensive than that of Lanczos, it does provide several advantages that justify these increased costs. Unrestarted Lanczos may offer optimal convergence when dealing with Hermitian eigenvalue problems but requires unbounded memory storage, and orthogonalization costs grow at the rate  $O(nd^2)$ [88]. Furthermore, clustered eigenvalues result in convergence deterioration of Lanczos, necessitating preconditioning. GD allows for the basis extension using a preconditioning residual, which can accelerate convergence. It is also more general as it allows any vector to be incorporated into the basis, not just Krylov vectors.

### 2.4.2 PRIMME

The PReconditioned Iterative MultiMethod Eigensolver library (PRIMME)<sup>2</sup> is a C99 software package designed to determine a user-specified number of eigenpairs or singular pairs of a matrix  $A$  [91]. There are extensions for the code, enabling it to run in Python, R, Fortran, and Matlab. Although our primary emphasis is on real symmetric eigenproblems, PRIMME can handle eigenproblems for generalized and complex Hermitian problems. PRIMME also provides preconditioning options and supports dynamic methods that can switch eigensolvers given the appropriate parameters.

---

<sup>2</sup>The PRIMME software library is publicly available at <https://github.com/primme/primme>

## 2.5 Sketching

Randomized Numerical Linear Algebra (RNLA) has grown in popularity due to algorithms that offer advantages in speed and reliability for extremely large matrices [62]. These algorithms find application in the least-squares problem [80], preconditioning [34], SVD [31], and orthogonalization [14].

One technique within RNLA is *sketching*, which operates as a dimensionality reduction method using subspace embeddings to lower the computational cost of matrix operations while still providing accurate estimations. In a 2006 paper by Sarlos [84], he notes that a sketching matrix  $S \in \mathbb{F}^{s \times n}$ , commonly built using Sparse Maps [64], the Subsampled Random Fourier Transform (SRFT) [4], or as a random Gaussian, is a subspace embedding for matrix  $A \in \mathbb{F}^{n \times n}$  with distortion factor  $\epsilon \in (0, 1)$  if

$$(1 - \epsilon) \cdot \|Ay\|_2 \leq \|SAy\|_2 \leq (1 + \epsilon) \cdot \|Ay\|_2. \quad (2.28)$$

The Johnson-Lindenstrauss Lemma [53, 25] states that any set of  $n$  points in a high-dimensional Euclidean space can be mapped to  $s$  points in a low-dimensional space while preserving the distance between points within a factor of  $(1 \pm \epsilon)$ , with the condition

$$s \geq \frac{4 \log n}{\epsilon^2/2 - \epsilon^3/3} = O\left(\frac{\log n}{\epsilon^2}\right). \quad (2.29)$$

Originally, sketching was utilized to solve least-squares problems. Consider the minimization problem

$$\text{minimize}_{y \in \mathbb{F}^d} \|By - f\|_2, \quad (2.30)$$

where  $B \in \mathbb{F}^{n \times d}$  with  $n \gg d$  [80]. Sketching can be incorporated into the least-squares problem by first generating a sketching matrix  $S \in \mathbb{F}^{s \times n}$  before applying it to Equation 2.30 to form the smaller  $s \times d$  problem:

$$\text{minimize}_{y \in \mathbb{F}^d} \|S(By - f)\|_2. \quad (2.31)$$

According to Equation 2.28, the residuals of the sketched problem compared to its non-sketched counterpart should only be off by a small constant factor.

In [72], Nakastukasa and Tropp noted that the RR method utilized in Krylov-based iterative solvers could also be cast as a least-squares problem using the variational formula described in [77]

$$\text{minimize}_{H \in \mathbb{F}^{d \times d}} \|AV - VH\|_2, \quad (2.32)$$

with  $V \in \mathbb{F}^{n \times d}$ ,  $A \in \mathbb{F}^{n \times n}$ , and  $H = V^H AV$ . This formulation presents an opportunity to apply sketching techniques. RR with sketching (sRR),

$$\text{minimize}_{\lambda, x} \|S(AV - VM)\|_2, \quad (2.33)$$

where  $M = (SV)^H(SAV)$ , can then be used for the extraction of approximate Ritz pairs from a basis  $V$  without the basis being fully orthonormal, as long as  $\kappa(V) \lesssim \epsilon_{\text{mach}}^{-1}$ . More details will be expounded upon in Chapter 4.

## Chapter 3

# Probing for the Trace Estimation of a Permuted Matrix Inverse Corresponding to a Lattice Displacement

### 3.1 Introduction

The trace approximation of a matrix function,  $f(A)$ , for a large sparse matrix  $A$ , is a computationally challenging problem. Commonly used functions are  $A^{-1}$  and  $\log A$  (used to find the matrix determinant). In this work, we focus on  $f(A) = A^{-1}$ , which has many applications in statistics [51], quantum Monte Carlo [3], and data mining [17]. Our motivating application comes from Lattice Quantum Chromodynamics (LQCD), where the trace of the inverse of an operator discretized on a symmetric, 4-dimensional, toroidal lattice representing space-time is often used to analyze the interactions, properties, and structures of hadrons on a subatomic scale [65]. The trace computations are part of larger-scale Monte Carlo simulations and, therefore, do not require high accuracy but must induce no statistical bias.



Effective methods for computing  $\text{tr}(A^{-1})$  exist for smaller matrices where sparse factorizations are possible or in cases where selective elements of the inverse can be found [9, 27, 59]. However, as the size and density of  $A$  increases, these methods become computationally infeasible, leaving stochastic estimation as the only alternative. A widely used method for this is the Hutchinson trace estimator [51], which takes the form

$$\text{tr}(A^{-1}) \approx \frac{1}{s} \sum_{j=1}^s z_j^H A^{-1} z_j, \quad (3.1)$$

where  $z_j$  for  $j = 1, 2, \dots, s$  are  $s$  i.i.d. random noise vectors (RNVs). The computational complexity of Equation 3.1 is dominated by finding solutions of the linear systems with some iterative method to approximate the Gaussian quadrature  $z_j^H A^{-1} z_j$  at each step. The vectors  $z_j$  for  $j = 1, 2, \dots, s$  can also be constructed using a Rademacher distribution, where each element equals  $\pm 1$  with a probability of 0.5 if using real numbers, and  $\pm 1, \pm i$  with probability 0.25 if using complex numbers, which is the case for LQCD matrices. This choice of vectors results in the estimator having a variance

$$\text{var}(z^H A^{-1} z) = 2(\|A^{-1}\|_F^2 - \sum_{j=1}^n (A_{j,j})^2), \quad (3.2)$$

which is minimum over all random distributions for  $z_j$  [12].

The variance formula in Equation 3.2 indicates that off-diagonal elements with high magnitude in  $A^{-1}$  contribute significant errors to the estimator, resulting in slow convergence. Various techniques have been introduced and studied to reduce the variance of the Hutchinson estimator by choosing vectors that better take advantage of the structure of the matrix [12, 17, 48, 96, 101].

One such technique is classical probing (CP). CP is a general technique that uses a graph coloring of the graph of an adjacency matrix  $A$  to construct structurally orthogonal probing vectors to extract specific non-zero entries of the matrix. For example, multiplying a diagonal matrix with a vector of ones recovers its diagonal. Similarly, when the adjacency

matrix of a graph is  $m$ -colorable, we can also recover the diagonal by multiplying the matrix with  $m$  vectors, each vector having ones in rows with the same color and zero elsewhere. In numerical optimization, probing is applied on the graph of  $A^2$  to compute the Hessian [41]. For trace estimation, CP constructs probing vectors from a coloring of the graph of  $A^k$  or, equivalently, the distance- $k$  coloring of the graph of  $A$ , where  $k \in \mathbb{Z}_+$  [96]. For many sparse matrices, the elements of  $A_{ij}^{-1}$  display a Green's function decay in magnitude with the distance between nodes  $i$  and  $j$ . Although  $A^{-1}$  is not sparse, using these probing vectors in the Hutchinson estimator removes from the variance (Equation 3.2) all elements (edges) of neighbors with distance  $\leq k$ . A drawback of CP is that if a coloring for a certain distance  $k$  does not produce the required variance reduction, a higher distance coloring cannot reuse the quadratures computed with the previous probing vectors.

Hierarchical probing (HP) was introduced to address the reuse issue [89, 57]. HP assigns colors to nodes hierarchically so that two nodes that receive the same color for some coloring distance  $k$  will never share the same color at higher distances. The technique also provided a computationally inexpensive way to produce a distance- $k$  coloring for large  $k$  when the matrix graph is a regular, toroidal lattice. This toroidal structure appears in LQCD matrices, which is also the focus of this work.

Deflation has also been used as a variance reduction technique [81, 38, 47]. While probing techniques capture high-magnitude elements from relatively small lattice distances, the low-rank approximation of  $A^{-1}$  using the lowest magnitude singular triplets of  $A$  typically captures a large part of the magnitude of  $A^{-1}$  at long distances. Thus, the two approaches are complementary and, when used in tandem, can significantly accelerate the Monte Carlo estimator.

In this work, we extend probing for computing the trace of a permutation of  $A^{-1}$ . The motivation comes from LQCD computations of the flavor-separated Generalized Parton functions (GPDs) where the so-called “disconnected diagrams” need to be calculated [39, 6]. This translates to the need to find the sum of certain off-diagonal elements of  $A^{-1}$  that correspond to a displacement along the  $z$  dimension of the 4-dimensional (space-time)

LQCD lattice. The displacement equates to a non-symmetric permutation of the rows of  $A^{-1}$ , where the index of a node  $x$  no longer refers to  $[x_1, x_2, x_3, x_4]$ , but instead  $[x_1, x_2, x_3 + p, x_4]$ . The associated trace problem is more challenging because the variance for  $PA^{-1}$  now includes the main diagonal of  $A^{-1}$ , which is of much larger magnitude than the main diagonal of  $PA^{-1}$ .

We propose an extension of CP that modifies a greedy graph coloring algorithm to consider not the node's original neighborhood but the neighborhood of its displacement. This idea applies to any permutation matrix, and the coloring can be performed hierarchically if desired. For toroidal lattices with displacement applied in one dimension, we prove lower bounds on the number of colors and study the algorithm's effect on variance reduction both theoretically and with LQCD experiments. The method results in orders of magnitude variance reduction over conventional probing methods.

The rest of this chapter is organized as follows: Section 3.2 introduces notation and discusses previous variance reduction techniques. Section 3.3 introduces the coloring algorithm with displacements and studies its properties theoretically. Experimental results are shown in Section 3.4. Section 3.5 summarizes this work and expresses some open questions.

## 3.2 Related Works

In this chapter, we seek the trace of  $PA^{-1}$ , where  $P$  is a permutation matrix corresponding to a single-dimension displacement in a 4-dimensional toroidal lattice, and  $A \in \mathbb{F}^{N \times N}$  is a non-singular matrix representation of that lattice.  $A$  can be complex-valued as in the case of LQCD, but for convenience and without loss of generality, our presentation involves real-valued matrices. Although our main idea applies to any  $P$  and  $A$ , the algorithm and the analysis are relevant to matrices stemming from a regular lattice discretization. Letting  $\mathbb{Z}_n$  be the multiplicative group of integers modulo  $n$ , then a  $d$ -dimensional toroidal lattice is described as

$$\mathbb{Z}_D^d = \mathbb{Z}_{D_1} \times \cdots \times \mathbb{Z}_{D_d}, \quad (3.3)$$

where  $D_j$  is the size of dimension  $j$ . Two lattice nodes  $x$  and  $y$  are connected by an edge if their coordinate vectors  $[x_1, \dots, x_d]$  and  $[y_1, \dots, y_d]$ , satisfy  $\|x - y\|_1 = 1$  (in a modulo sense). In LQCD, the lattice represents the 4-dimensional space-time.

Variance reduction techniques for the Hutchinson trace estimator focus on two approaches: one derives an approximation to  $A^{-1}$  such as from deflation or preconditioning, which we briefly address in Section 3.2.4; the other replaces the Rademacher vectors with ones that better take advantage of the structure of the matrix. The use of orthogonal columns of the Hadamard or Fourier matrix has been proposed [17], which can systematically annihilate specific diagonals of the matrix and thus reduce the variance in (3.2). The variance reduction is monotonic with the number of columns used. Still, this method works no better than using solely RNVs as the patterns of diagonals removed are not typically the heaviest variance-contributing diagonals of  $A^{-1}$ . The following methods attempt to capture these heaviest elements directly.

### 3.2.1 Classical Probing

The inverse of an  $N \times N$  non-singular matrix  $A$  where  $\|A\| < 1$  can be represented by the Neumann series  $A^{-1} = \sum_{k=0}^{\infty} (I - A)^k$  [94]. As this series is convergent, higher powers of  $(I - A)^k$  provide a smaller contribution to  $A^{-1}$ . Many matrices from Partial Differential Equations, LQCD, and other applications display a significant decay in the elements of  $(I - A)^k$  for larger values of  $k$ , further motivating the idea of probing [15, 96, 37]. [100] first introduced a basic form of probing in LQCD, which has become more popular with the name dilution since [33].

The CP method does not directly approximate  $A^{-1}$  but instead locates its highest-magnitude elements using graph coloring. Based on the decay principle above and since  $(I - A)^k$  and  $A^k$  have the same adjacency matrix, it is the first few powers of  $A^k$  that contribute to the highest-magnitude elements of  $A^{-1}$ . Note that the neighborhood of a node  $x$  in the graph of  $A^k$  is the same as the distance- $k$  neighborhood of  $x$  in the graph of

A. Therefore, the computation of  $A^k$  can be avoided by working directly on the graph of  $A$ .

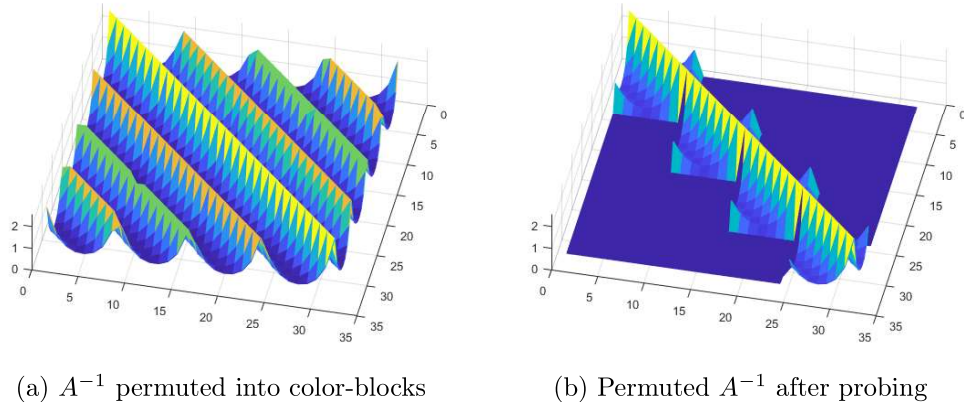
Assume that we have computed a distance- $k$  coloring of the graph of  $A$ , which results in  $m$  colors. Conceptually, if we permuted the nodes with the same color together,  $A^k$  would have  $m$  color blocks along the diagonal, each being diagonal matrices. We construct the following structurally orthogonal probing vectors  $z_j$ ,  $j = 1, 2, \dots, m$ ,

$$z_j(i) = \begin{cases} 1 & \text{if } \text{color}(i) = j \\ 0 & \text{otherwise} \end{cases}. \quad (3.4)$$

Notice that these vectors can recover precisely the trace  $\text{tr}(A^k) = \sum_{j=1}^m z_j^T A^k z_j$  because they annihilate all matrix elements outside the color-blocks along the diagonal of  $A^k$  and because the color-blocks are diagonal matrices themselves. Although the diagonal blocks are dense matrices in  $A^{-1}$ , using these  $z_j$  in the trace estimator (3.1) has the same effect of annihilating all off-diagonal blocks of  $A^{-1}$ , or equivalently, any neighbor at a distance up to  $k$  from *any* node in the same color group. Then, the accuracy of the trace estimation is the summation of the variances described in Equation 3.2 of the diagonal color blocks.

Figure 3.1 displays this effect. Let  $A$  be a 32-node 1D Laplacian matrix with periodic boundary conditions shifted by its smallest non-zero eigenvalue, making it non-singular. A distance-3 coloring of this matrix yields four colors. Consider the permutation vector  $perm$  that lists the indices of all nodes in order of their color label, i.e., nodes with color 1 come first, followed by colors 2, 3, and 4. Plotting  $A^{-1}$  symmetrically permuted by  $perm$  shows the color blocks along the diagonals (Figure 3.1a). Figure 3.1b shows  $A^{-1} \odot HH^T$  permuted the same way, where  $\odot$  refers to the Hadamard product between two matrices, and the columns of  $H$  consist of the four probing vectors from Equation 3.4. It can be seen that every element outside the color blocks along the main diagonal gets annihilated.

Computationally, we can use a greedy coloring algorithm that takes linear time for  $A^k$  and provides close to the optimal number of colors for most matrices with regular sparsity



**Figure 3.1:** Using a shifted 1D Laplacian  $A$  with 32 nodes and periodic boundary conditions, Figure 3.1a shows  $A^{-1}$  permuted into color-blocks based on a distance-3 coloring before probing vectors are applied. Figure 3.1b shows the result of these color blocks after applying the probing vectors to the shifted Laplacian inverse,  $A^{-1} \odot HH^T$ .

patterns. Most of the computation is spent on the iterative method that solves for the  $m$  linear systems  $A^{-1}z_j$ .

CP is a deterministic method. Many applications, such as LQCD, require an unbiased trace estimator (unless the deterministic accuracy can be guaranteed to be well below the statistical significance of the simulation). Moreover, suppose the probing vectors from the distance- $k$  coloring do not provide sufficient accuracy. In that case, we seek ways to either use  $A^{-1} \odot HH^T$  as the matrix of the statistical estimator shown in Equation 3.1 or to extend CP to higher distances. In either case, the work spent on solving  $A^{-1}z_j$  should be re-used and not discarded. This has been explored in [89, 57] as described next.

### 3.2.2 Removing Deterministic Bias

We note that the vectors in Equation 3.4 consist of 0's and 1's in the positions determined by the colors. To remove the deterministic bias from the CP estimation, we can introduce random noise to the vectors  $z_j$  similarly to one step of Hutchinson ( $s = 1$ ). Consider the noise vector  $z_0 \in \mathbb{Z}_2^N$  and apply a Hadamard product between  $z_0$  and each of the probing vectors  $z_j$ ,  $j = 1, \dots, m$ ,

$$V = [z_0 \odot z_1, z_0 \odot z_2, \dots, z_0 \odot z_m]. \quad (3.5)$$

As shown in [89],  $VV^H = HH^T$  have the same non-zero pattern, but using the vectors  $v_j$  in Equation 3.1 imparts no deterministic bias.

Moreover, given a sequence of random vectors,  $z_0^{(i)}, i = 1, \dots, s$ , we can construct the vector sets  $V^{(1)}, \dots, V^{(s)}$  as above. Using these  $s \times m$  vectors in Equation 3.1 is the same as performing  $s$  steps of Hutchinson on the variance reduced matrix  $A^{-1} \odot HH^T$ .

### 3.2.3 Hierarchical Probing

Instead of applying the CP method for a fixed distance  $k$  followed by the Hutchinson stochastic estimator, it is more beneficial to continue with probing to higher distances as long as the elements of  $A^{-1}$  continue to display substantial decay and as long as the work from prior distances can be reused.

Saving computations by reusing previous work is the goal of Hierarchical probing (HP), which was initially proposed for matrices with lattice-type structure [89] and was later extended to arbitrary sparsity patterns [57]. The idea is to enforce a hierarchical coloring, which ensures that probing vectors for smaller distance colorings are contained in the subspace of the vectors generated for higher distances, with all distances being a power of two. Therefore, the trace estimation reuses the already computed quadratures  $z_j^T A^{-1} z_j$  and augments them with those from higher distances.

We can generate a hierarchical coloring on lattices by recursively partitioning a  $d$ -dimensional lattice into  $2^d$  sub-lattices, each receiving a different color. The non-overlapping sub-lattices guarantee that if two nodes share a color at a distance  $k$ , they must also share a color at any smaller distance, and if two nodes do not share a color at a distance  $k$ , they will not share a color at higher distances. Each recursion step doubles the distance between nodes of the same color. The recursion stops when separate colors are assigned to all nodes or when the requested distance is reached. A red-black coloring between recursion steps allows for intermediate colorings as the number of colors increases by a factor of  $2^d$  at each recursion.

Instead of using Equation 3.4, probing vectors for the HP can be generated efficiently as special permutations of the rows and columns of the Hadamard or Fourier matrices. The nested coloring implies a nesting of the subspaces of the probing vectors, which can be used incrementally until the desired accuracy is achieved. Used in its unbiased form (Equation 3.5) with  $s = 1$ , this method proved particularly flexible and practical in real-world LQCD problems [46, 45].

[57] extended HP to work with arbitrary lattice sizes, mainly general sparse matrices. Hierarchical techniques can also be used with the algorithm of this chapter if desired. However, because the number of colors required increases by a factor of 3-4 over the non-hierarchical version, we assume that users can choose a priori the required distance.

### 3.2.4 Deflation

A different way to reduce the estimator's variance is to deflate the lowest singular triplets of  $A$  [38]. Given  $U$  and  $V$ , consisting of approximate left and right singular vectors of the smallest singular values of  $A$ , respectively, we can form the oblique projector  $Q = AV(U^H AV)^{-1}U^H$  and split the trace computation into two parts,

$$\mathbf{tr}(A^{-1}) = \mathbf{tr}(A^{-1}Q) + \mathbf{tr}(A^{-1}(I - Q)). \quad (3.6)$$

Since  $\mathbf{tr}(A^{-1}Q)$  is easily computed as the trace of the small matrix  $(U^H AV)^{-1}U^H V$ , we instead apply the stochastic estimator on the  $\mathbf{tr}(A^{-1}(I - Q))$ , which is expected to have smaller variance. The number of singular vectors needed to provide a significant variance reduction of the estimator depends on the spectral decay of the matrix  $A$  and its size. For significant decay or small matrix size, an iterative SVD solver can compute the singular space of  $A$  with a multigrid method as a preconditioner [38] or using a multigrid eigensolver directly [36]. Still, for large matrix sizes, the cost of computing and applying a large number of singular vectors becomes significant. Because the goal is to approximate  $A^{-1}(I - Q)$  in the Frobenius norm, the accuracy of individual vectors is less critical. [81]



showed that hundreds or thousands of singular vectors from the coarse grid operator of multigrid can be computed efficiently and applied effectively for deflation.

Deflation works complementary to probing. While probing captures heavy elements of  $A^{-1}$  occurring within some distance  $k$  between nodes, deflation captures heavy connections between elements at long-range distances. Therefore, combining the two techniques has shown significant improvements over using one of these methods individually. As deflation does not depend on permutations, we use the same deflation as in [81] and focus solely on the effects of our new probing method.

### 3.3 Probing for Permutations

Consider the problem of finding the trace of  $PA^{-1}$  where  $P$  is a permutation matrix. The problem arises in LQCD, where  $P$  corresponds to one or more displacements in the lattice. We will study this problem shortly, but let us first consider the problem for a general  $P$ .

The question is how to achieve the probing goals for  $PA^{-1}$ . The CP method would take powers of the matrix  $AP^T$ , which does not relate to how information propagates through powers of  $A$  to generate  $A^{-1}$ . In other words, this method may not capture the most significant elements of  $A^{-1}$ , which are at close graph distances for each node, and thus does not satisfy the design goal of probing. Moreover, when  $AP^T$  has a nonsymmetric sparsity structure, the graph coloring problem is poorly defined but can be avoided by coloring the graph of the symmetric part of a matrix. Finally, for our LQCD application on regular lattices, the powers  $(AP^T)^k$  are much denser than the corresponding  $A^k$ , which means a larger number of colors and, thus, probing vectors.

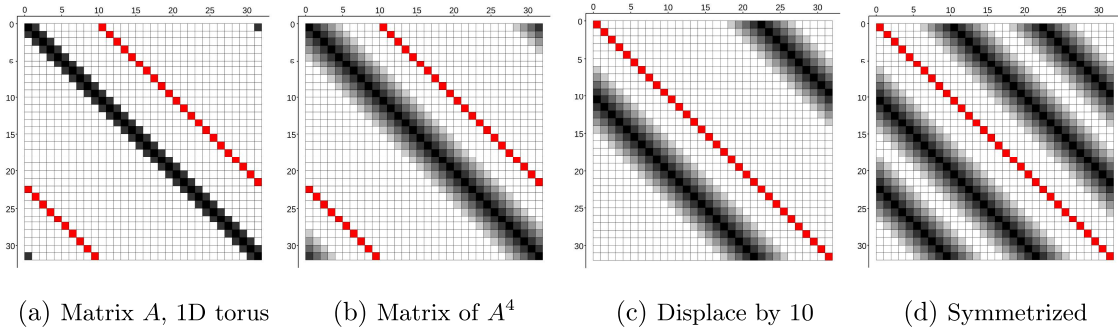
The solution is conceptually simple. Since  $PA^{-1} = P \sum_{k=0}^{\infty} (I - A)^k$ , we can first take powers of the matrix  $A$ , permute them, and then find the coloring on the associated graph of  $PA^k$ , or rather its symmetric part  $PA^k + (PA^k)^T$ . Despite its simplicity, when this method is applied to toroidal lattices stemming from our LQCD application, it creates

connectivity patterns that our HP method cannot handle. However, these patterns allow for a CP-based algorithm specifically tailored for this application.

In LQCD, the application of disconnected diagrams requires the trace of a certain projected operator, which for this discussion can be abstracted as the sum of all the elements of  $A^{-1}$  that correspond to a displacement  $p \in \mathbb{Z}_+^d$ , i.e.,  $\sum_x A_{ij}^{-1}$ , where  $i$  is the index of the lattice node  $x = [x_1, \dots, x_d]$  and  $j$  is the index of node  $x + [p_1, \dots, p_d]$ . Let  $P$  be the permutation matrix that places the required off-diagonal elements onto the main diagonal. In MATLAB, the corresponding permutation index is computed as

$$\text{perm} = \text{Coord2Index}(\text{mod}(\text{Index2Coord}([1:N],D)+p,D), D).$$

The two functions `Coord2Index` and `Index2Coord` are the maps between lattice coordinates and the particular index ordering of the application. The inverse permutation  $P^T$  simply maps a lattice point  $y$  to  $y - [p_1, \dots, p_d]$ . The idea of coloring the graph of  $PA^k + (PA^k)^T$  is shown in Figure 3.2 for the matrix of a 1D periodic lattice of 32 points with  $p = 10$  and  $k = 4$ . The red locations indicate the required diagonal at displacement  $p = 10$ . The grayscale diagonals show the magnitude of the non-zeros of the following matrices: Figure 3.2a of the matrix  $A$ ; Figure 3.2b of the  $A^4$ ; Figure 3.2c of the matrix  $PA^4$ , and Figure 3.2d of the matrix  $PA^4 + (PA^4)^T$ . Based on the aforementioned decay property, coloring this last matrix would eliminate the heaviest elements of the inverse.



**Figure 3.2:** Applying a power and displacement to a matrix representation of a 1D toroidal lattice. The red diagonal represents the locations of the elements corresponding to the wanted displacement.

As with CP, we use a greedy linear time algorithm to color  $PA^k + (PA^k)^T$ . However, working directly on the lattice can speed up the distance- $k$  coloring process. Given a node  $x$  with lattice coordinates  $[x_1, \dots, x_d]$ , we do not find the distance- $k$  neighborhood of  $x$ , but rather the distance- $k$  neighborhoods centered at

$$x^+ = [x_1 + p_1, \dots, x_d + p_d] \text{ and } x^- = [x_1 - p_1, \dots, x_d - p_d]. \quad (3.7)$$

Displacements in the directions of  $p$  and  $-p$  enforce a symmetric matrix structure. We denote the distance- $k$  neighborhood of  $x$  for displacement  $p$  as,

$$\begin{aligned} N^d(x, p, k) &= N^d(x^+, 0, k) \cup N^d(x^-, 0, k) \\ &= \{y : \|y - x^+\|_1 \leq k\} \cup \{y : \|y - x^-\|_1 \leq k\}. \end{aligned} \quad (3.8)$$

During coloring, we exclude  $\{x\}$  from the neighborhood, and the superscript  $d$  is removed when the dimension is implied.

We make three observations. First, the main diagonal of the original  $A^{-1}$ , whose elements are typically of the largest magnitude, is part of the off-diagonal structure of  $PA^{-1}$  and contributes to the estimator variance. However, the  $(x, x)$  elements of this diagonal are now displaced to the  $(x, x^-)$  lattice points in the  $N^d(x, p, 0)$ , so our new method eliminates them immediately for any probing distance. Equivalently, because of the assumed decay, the elements of the next-highest magnitude in  $A^{-1}$  will be in the diagonals closest to the main or at distance  $k = 1$  from it. The decay continues with higher distances  $k$ . Therefore, the new algorithm includes in the neighborhoods  $N^d(x, p, k)$  all original distance- $k$  neighbors of the points  $x^+$  and  $x^-$  as these will have the most significant weight. Finally, although  $k = 0$  removes the old main diagonal (the graph of  $P + P^T$ ), in practice, probing is meaningful for  $k \geq 1$ .

### 3.3.1 Coloring with Displacements Algorithm

Once we have defined the neighborhood of each node in the displacement graph, we can use a simple greedy coloring approach by assigning a color to the first node, then the second, until all nodes have a color [67]. The number of colors translates to the number of iterations in the stochastic estimator. Minimizing this number is not critical, as more vectors/iterations could imply a more significant variance reduction. However, this additional reduction beyond the best distance- $k$  coloring is hard to quantify and may not be more effective than using extra random noise vectors. Thus, the order in which the greedy algorithm visits nodes is important.

We have experimented with some standard visitation orders, such as natural and red-black orderings, a completely random order, and a random red-black where the order of the nodes within a color is random. In addition, we tested a domain decomposition idea, where an independent set of the graph of  $A^i$  was constructed for various  $i$ 's. Then, a breadth-first search was used to add neighborhoods to each of these centers (for  $i = 1$ , this reverts to red-black). After extensive testing, we observed that in most cases, natural and red-black orders achieved the fewest colors. Surprisingly, thousands of runs of the random variants yielded only marginal improvements, and the domain decomposition idea deteriorated with increasing  $i$ . We believe this is due to the well-structured connections of the lattice.

Algorithm 4 shows how to work directly on the lattice  $\mathbb{Z}_D^d$  to apply the greedy distance- $k$  coloring for a displacement vector  $p$  and a user-defined visitation order. It returns a vector `Colors`, which can be used in Equation 3.4 to generate the probing vectors. To avoid re-computing the neighborhood for each lattice point, Algorithm 5 builds first a “stencil” of coordinate offsets that, when added to the coordinates of some point  $x$ , returns the coordinates of the points in  $N(x, p, k)$ . Because every lattice node is of the same degree, it is clear that the maximum number of colors produced by the greedy algorithm is one more than the degree of a node, i.e., colors are less or equal to  $|N(x, p, k)| + 1 = \text{len}(\text{Stencil}(:$

, 1)) + 1. A bit array of this size can be used to record the colors used for each neighborhood and find the first color not in use. The colors returned by Algorithm 4 are used in Equation 3.4 and then Equation 3.5 to generate the unbiased probing vectors to be applied on the displaced inverse  $PA^{-1}$ .

**Algorithm 4:** Displacement Coloring on a  $d$ -Dimensional Lattice

```
Input:  
     $p = d$ -length displacement array     $D = d$ -length array of dimension sizes  
     $k =$  Coloring distance  
Output:  
    Colors = Array of lattice colors  
Create_Coloring( $p, D, k$ )  
1  $N = \text{prod}(D)$ ;    Colors = zeros( $N, 1$ );  
2 Stencil = Create_Stencil( $k, p, 1, \text{zeros}(1, \text{len}(D))$ );  
3 for  $i = \text{Make_Visiting_Order}(N)$  do  
4      $ix = \text{Index2Coord}(i, D)$ ;    # Convert the node index to a lattice coord  
    # Add each stencil offset to  $ix$  to find  $ix$ 's neighborhood  
5     Neighbor_Colors = [];  
6     for  $s = \text{Stencil}$  do  
7          $n = \text{Coord2Index}(\text{mod}(ix + s, D))$ ;  
8         Neighbor_Colors = [Neighbor_Colors, Colors( $n$ )];  
    # Create a logical array to mark which colors are already in use  
9     Colors_In_Use = false(len(Stencil(:, 1)));  
10    for  $c = \text{Neighbor_Colors}$  do  
11        if  $c > 0$  then  
12            Colors_In_Use( $c$ ) = true;  
    # Find the first color not in use and set that to be  $i$ 's color  
13    for  $j = 1 : \text{len}(\text{Colors\_In\_Use})$  do  
14        if  $\sim \text{Colors\_In\_Use}(j)$  then  
15            Colors( $i$ ) =  $j$ ;  
16            break;  
17 return Colors;
```

**Algorithm 5:** Find coordinate offsets for each node in a neighborhood

**Input:**

- $x$  =  $d$ -dimensional array to store an offset
- $k$  = Coloring distance
- $p$  = Displacement array of length  $d$
- dim = Recursion/dimension level

**Output:**

Stencil = A mapping of a lattice coordinate's neighbors

Create\_Stencil( $x, k, p, dim$ )

```

1 if dim == 1 then
2   Stencil = []; # Empty array to hold all neighbor offsets
3 if dim == len(x)+1 then
4   # Append the positively and negatively displaced offset to the stencil
5   return unique([Stencil; x + p; x - p], 'rows');
6   # Find the distance-k neighborhood around x
7 for j = -k : k do
8   x(dim) = j
9   Stencil = [Stencil, Create_Stencil(x, k - |j|, p, dim+1)]
10 return Stencil

```

The size of the distance- $k$   $L_1$  ball on the lattice is  $O(k^d)$ , and the stencil contains two such balls in  $N(x, p, k)$ . To union the two stencil balls, we must remove duplicates when the balls overlap, which can be obtained by sorting the elements. Therefore, generating the stencil has a  $O(k^d d \log k)$  complexity. The dominant part of the complexity is the linear time greedy algorithm, which visits the neighborhoods  $N(x, p, k)$  for each  $x$ , resulting in the algorithm's complexity being  $O(Nk^d)$ .

Although the algorithm we presented is for any  $d$ -dimensional displacement, in practical LQCD problems, the displacement occurs only in the  $z$  space-time direction. For convenience, our theoretical discussion considers the displacement to be in the 1st dimension, i.e.,  $p = p_1$  and  $p_2 = \dots = p_d = 0$ .

### 3.3.2 Lower Bound on the Number of Colors

The chromatic number of a graph must be at least the size of its maximal clique. In our problem, the neighborhood of every lattice node is the union of two  $L_1$  balls, so we seek to identify its maximal clique. Finding the maximal clique is complicated by the wrap-around property of the torus, which adds additional constraints to the coloring. Thus, the results depend not only on  $p$  and  $k$  but also on the size  $D_i$  of each dimension. To avoid this complication, we ignore the toroidal property, which, for sufficiently large  $D_i$ , is equivalent to considering the lattice  $\mathbb{Z}_\infty^d$ , which is infinite in all  $d$  dimensions. By removing these constraints from the coloring algorithm, the maximal clique of the infinite lattice may be smaller, and thus, its size will still be a lower bound to the chromatic number of the finite toroidal lattice. We call the number of colors required to distance- $k$  color the infinite lattice with displacement  $p$ ,  $\text{col}(\mathbb{Z}_\infty^d, p, k)$ .

Without displacement, i.e.  $p = 0$ , each neighborhood  $N(x, 0, k)$  is an  $L_1$  ball of radius  $k$ . Any two points in this ball are at  $L_1$  distance  $2k$  or less. Therefore, the maximal clique of the distance- $k$  graph of  $N(x, 0, k)$  should be the nodes inside the  $L_1$  ball of radius  $\lfloor \frac{k}{2} \rfloor$ . If  $k$  is odd, this  $L_1$  ball is extended by one point in one dimension. The size of this clique provides the lower bound of the chromatic number:

$$\text{col}(\mathbb{Z}_\infty^d, 0, k) = \begin{cases} |N^d(\mathbf{0}, 0, \frac{k}{2})| & \text{if } k \text{ is even} \\ |N^d(\mathbf{0}, 0, \lfloor \frac{k}{2} \rfloor)| + |N^{d-1}(\mathbf{0}, 0, \lfloor \frac{k}{2} \rfloor)| & \text{if } k \text{ is odd} \end{cases}, \quad (3.9)$$

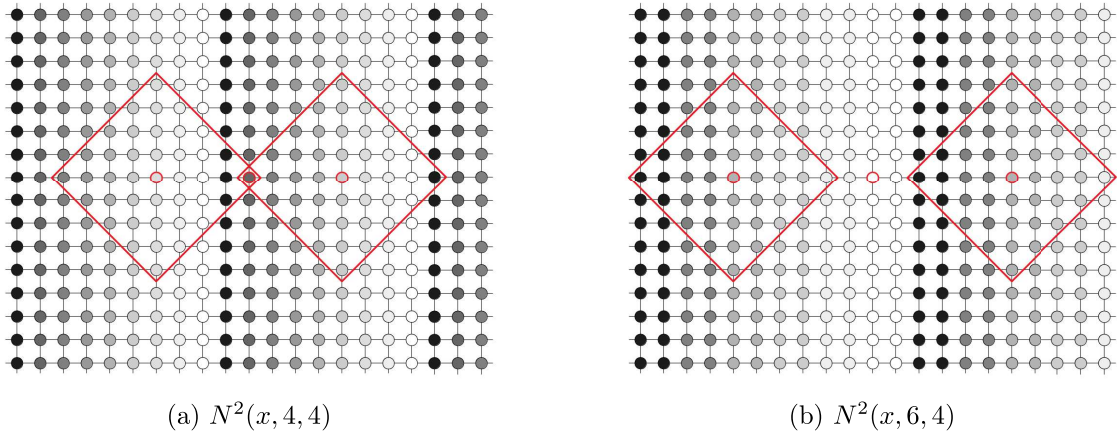


where  $\mathbf{0} = [0, \dots, 0]$  is chosen as a representative neighborhood center. Recurrence relations can be derived to compute this number for any dimension, although general closed forms for an arbitrary number of dimensions are unknown. More details can be found in [16, 89, 57].

With displacement  $p > 0$ , the  $L_1$  balls of a neighborhood  $N(x, p, k)$  are not centered around the node  $x$ , resulting in different coloring patterns. We characterize the number of colors needed first for  $p \geq k$  and then for  $p < k$ . Proofs are given in Appendix A.

**Theorem 1.** *Let  $x \in \mathbb{Z}_\infty^d$ . If  $p \geq k$ , then  $\forall y \neq x$  with  $y_1 = x_1$ , it holds  $y \notin N(x, p, k)$ .*

The above theorem implies that when  $p \geq k$ , all nodes with the same  $x_1$ -coordinate can share the same color, reducing the  $d$ -dimensional coloring problem to a 1D problem. Figure 3.3 shows an example of this. To find the lower bound on the number of colors, we separately consider the two sub-cases,  $p = k$  and  $p > k$ .



**Figure 3.3:** The neighborhood  $N^2(\mathbf{0}, p, k)$  and how 1D coloring is sufficient when  $p \geq k$ .

**Theorem 2.** *If  $p = k$ , then  $\text{col}(\mathbb{Z}_\infty^d, p, k) = 2k + 1$ .*

**Theorem 3.** *If  $p > k$ , then  $\text{col}(\mathbb{Z}_\infty^d, p, k) = \lceil \frac{2p}{p-k} \rceil = \lceil \frac{2k}{p-k} \rceil + 2$ .*

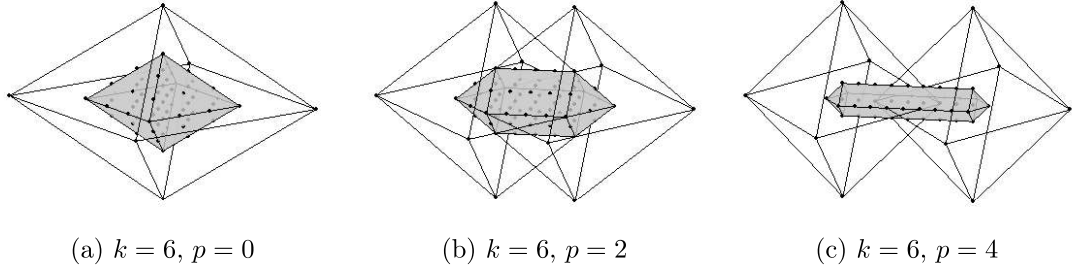
When  $p < k$ , the two  $L_1$  balls centered around  $x^-$  and  $x^+$  overlap. Next, we identify the maximal clique in this neighborhood for which all points are at most at a distance  $k$  considering displacement  $p$ . As before, we center the neighborhood at  $x = \mathbf{0}$ . The case

where  $(k + p)$  is even is considered in Theorem 4, an example of which is shown in Figure 3.4.

**Theorem 4.** Assume  $(k + p)$  is even and  $p < k$ . Let,  $\alpha = \lfloor \frac{k+p}{2} \rfloor$ ,  $\beta = \lfloor \frac{k-p}{2} \rfloor$ , and define the set

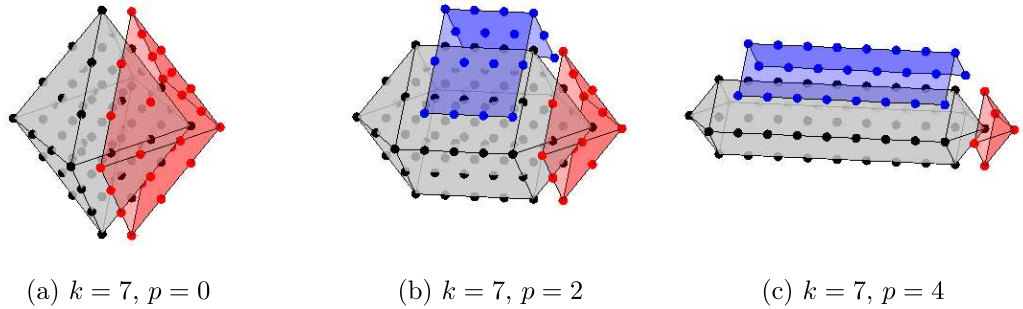
$$C(d, \alpha, \beta) = \left\{ x : \|x\|_1 \leq \alpha \text{ and } \sum_{i=2}^d |x_i| \leq \beta \right\}. \quad (3.10)$$

Then  $\forall x, y \in C(d, \alpha, \beta)$ ,  $x \in N(y, p, k)$ , i.e.,  $C(d, \alpha, \beta)$  constitutes a distance- $k$  clique.



**Figure 3.4:** The distance- $k$  clique shown in grey of the neighborhood  $N^3(\mathbf{0}, p, k)$  which is shown as wireframes, when  $p < k$  and  $(k + p)$  is even as described in Theorem 4.

For  $(k + p)$  is odd, Equation 3.9 shows that when  $p = 0$ , the clique needs to be extended by one hyper-surface. In Theorem 5, we prove that for  $p > 0$ , the clique requires two additional hyper-surfaces as depicted in Figure 3.5.



**Figure 3.5:** Distance- $k$  cliques of  $N^3(\mathbf{0}, p, k)$  when  $(k + p)$  is odd as shown in Theorem 5. Set  $C(d, \alpha, \beta)$  is the grey set in the center, set  $S$  is the red hyper-surface on the right, and  $T$  is the blue hyper-surface on the top.

**Theorem 5.** Assume  $(k + p)$  is odd and  $k > p$ . Define  $C' = C(d, \alpha, \beta) \cup T \cup S$ , where  $C(d, \alpha, \beta)$  is defined in condition 3.10 and

$$T = \{x : -(p - 1) \leq x_1 \leq p \text{ and } 1 \leq x_2 < \beta + 1 \text{ and } \sum_{i=2}^d |x_i| = \beta + 1\}, \quad (3.11)$$

$$S = \{x : p + 1 \leq x_1 \leq \alpha + 1 \text{ and } |x_2| \leq \beta \text{ and } \|x\|_1 = \alpha + 1\}. \quad (3.12)$$

Then  $\forall x, y \in C'$ ,  $x \in N(y, p, k)$ , i.e.,  $C'$  constitutes a distance- $k$  clique.

Finally, to count the number of points in the clique for any combination of  $d$ ,  $p$ , and  $k$ , we can use the recursive algorithm 6. Table 3.1 shows the analytic formulas for the size of  $C(d, p, k)$  obtained by the nested summations of points over all dimensions for lattices with  $d = 1, 2, 3, 4$  when  $k+p$  is even and  $k > p$ . For  $k+p$  odd, we also need to add the size of  $d-1$  dimensional hyper-surfaces  $S$  and  $T$ . It is not hard to see that  $|S| + |T| = |C(d-1, \alpha, \beta)|$ . Therefore, we arrive at the following general lower bound for the number of colors of our algorithm,

$$\text{col}(\mathbb{Z}_\infty^d, p, k) = \begin{cases} 2k + 1 & \text{if } k = p \\ \lceil \frac{2p}{p-k} \rceil & \text{if } k < p \\ |C(d, \alpha, \beta)| & \text{if } k > p, k + p \text{ even} \\ |C(d, \alpha, \beta)| + |C(d-1, \alpha, \beta)| & \text{if } k > p, k + p \text{ odd} \end{cases}. \quad (3.13)$$

**Algorithm 6:** Recursive Function to Find the Lower Bound on Colors Needed

**Input:**

$k$  = Coloring distance

$p$  = Displacement (in the first dimension)

$s$  = The current distance traveled

$d$  = Current dimension level

$\text{min\_Colors}$  = Number of colors needed so far

$\text{Min\_Num\_Colors}(k, p, s, d, \text{min\_Colors})$

```

1 if  $p > k$  then
2    $\text{min\_Colors} = \lceil \frac{2k}{p-k} \rceil + 2;$ 
3   return  $\text{min\_Colors}$ 
4 if  $d == 0$  then
5    $\text{min\_Colors} = \text{min\_Colors} + 1;$ 
6   return  $\text{min\_Colors}$ 
7 if  $d == 1$  then
8    $\text{min\_Colors} = \text{min\_Colors} + 2(\lfloor \frac{k-p}{2} \rfloor - s) + 1;$ 
9   return  $\text{min\_Colors}$ 
10 for  $i = -\lfloor \frac{k-p}{2} \rfloor + s : \lfloor \frac{k-p}{2} \rfloor - s$  do
11    $\text{min\_Colors} = \text{Min\_Num\_Colors}(k, p, s + |i|, d - 1, \text{min\_Colors});$ 
12 return  $\text{min\_Colors}$ 

```

d	Size of the clique $C(d, \alpha, \beta)$ for $k > p$ and $(k + p)$ even
1	$2\alpha + 1$
2	$-2\beta^2 + 4\alpha\beta + 2\alpha + 1$
3	$-\frac{8}{3}\beta^3 + (4\alpha - 2)\beta^2 + (4\alpha + \frac{2}{3})\beta + 2\alpha + 1$
4	$\frac{1}{3}(2(4\beta^3 + 6\beta^2 + 8\beta + 3)\alpha - 6\beta^4 - 8\beta^3 - 6\beta^2 + 2\beta + 3)$

**Table 3.1:** Formulas for size of the clique  $|C(d, \alpha, \beta)|$ , if  $k > p$  and  $(k + p)$  is even, with  $\alpha = \lfloor \frac{k+p}{2} \rfloor$  and  $\beta = \lfloor \frac{k-p}{2} \rfloor$ . If  $(k + p)$  is odd, use Equation 3.13.

### 3.3.3 Clearances

The LQCD application of disconnected diagrams requires the computation of traces for one displacement and multiple displacements (e.g.,  $p = 0, \dots, 8$ ). Using different colorings to individually find each of the traces is computationally prohibitive as we would have to solve a different set of linear systems for each of the nine displacements. Therefore, asking whether the probing vectors from one displacement can be used effectively for others is natural. Theorem 6 shows that if a distance- $k$  coloring generated for displacement  $p$  is used for displacement  $p + \lambda$  or  $p - \lambda$ , then it clears at least distance  $\max(k - \lambda, 0)$ .

**Theorem 6.**  $N(\mathbf{0}, p \pm \lambda, k - \lambda) \subseteq N(\mathbf{0}, p, k)$ , for any  $\lambda \leq k$ .

Based on this theorem, a specific  $(p, k)$ -coloring, i.e., a distance  $k$ -coloring for displacement  $p$ , will also effectively reduce variance for nearby displacements. However, its effectiveness declines for further displacements. Our LQCD experiments show that choosing larger valued  $(p, k)$  pairs is more beneficial.

### 3.3.4 Multiple Displacements

The diminishing clearance achieved from a particular  $(p, k)$ -coloring to farther displacements motivates finding a single distance- $k$  coloring for a graph stemming from multiple displacements. The goal is to spread the effectiveness of a power  $k$  to more values of  $p$  instead of using one  $p$  and a high  $k$  value while still using fewer colors than all displacements individually. Given a list of displacements,  $p_1, p_2, \dots, p_n$ , the neighborhood of a node  $x$  can be constructed as,

$$N(x, [p_1, \dots, p_n], k) = N(x, p_1, k) \cup \dots \cup N(x, p_n, k). \quad (3.14)$$

Algorithm 4 can be modified to do this by calling `Create_Stencil` (shown in Algorithm 5) for multiple different  $p$  vectors and taking the union of the created stencils.

As expected from Theorem 6, we observed that the resulting clique is smaller when the displacements  $p_1, p_2, \dots, p_n$  are successive. When the distance between each displacement  $p_i$  is greater than  $k$ , this method returns a similar number of colors to the total number returned when each of the displacements is applied individually. However, in our LQCD experiments, even successive multiple displacements did not yield improvements in variance over using just one of the higher displacements (say  $p_n$ ) with a distance larger than  $k$ . We believe this is because traces derived from smaller displacements have significantly higher magnitudes, thus requiring less variance reduction. Further discussion on the effectiveness of clearances is discussed in the experiments section.

### 3.3.5 Tiles

Despite the linear complexity of Algorithm 4, practical lattice sizes reach  $64^4$  (and often larger), where the neighborhood size is  $O(k^4)$  (e.g., for  $p = 0, k = 10$  there are 8361 neighbors to visit). It is clear, therefore, that we should avoid running the method every time a new trace problem is solved. One solution is to generate and save in a database colorings for the most useful lattice sizes. However, the regular structure of the lattice results in coloring patterns that repeat across the lattice. These repetitive coloring patterns motivate tiling, where we color a smaller toroidal lattice, the “tile”, and repeat its coloring throughout the larger lattice. Small tiles can be generated at runtime, while several common larger tiles can be saved in the aforementioned database.

The second motivation comes from the effect of lattice size on the number of colors. While our analysis was based on  $\mathbb{Z}_\infty^d$ , with a wrap-around structure, the additional constraints make the number of colors sensitive to the lattice size. For example, the distance-1 coloring of a non-periodic 1D lattice requires two colors, while for the toroidal lattice, we need two colors when  $D_1$  is even and three colors when  $D_1$  is odd. These effects are amplified in higher dimensions and more considerable distances. Interestingly, for a given combination  $(p, k)$ , increasing the lattice size often results in a larger number of colors. Therefore, it is beneficial if a lattice can be composed of smaller tiles.

The tile size must satisfy certain constraints. First, because the periodicity of the tile must match that of the lattice, a hyper-cubic tile must be used. Second, the tile must be large enough to include an entire  $N(x, p, k)$  neighborhood. Otherwise, the neighborhood will wrap around the boundary and thus require more colors than a larger tile would need. To avoid this wrap-around effect, dimensions without displacement should have a length of at least  $2k + 1$ . The dimension with the displacement should have a length of at least  $2(p + k) + 1$ . For example, a  $(p = 8, k = 8)$ -coloring on a 4D lattice would require a tile of size at least  $34 \times 18^3$ .

A third constraint is that the tile dimensions must divide the lattice dimensions to ensure a valid coloring. In LQCD, lattices have dimensions that are a power of two in size, occasionally including a factor of three. Therefore, the minimum size  $34 \times 18^3$  tile of the previous example cannot be used. One solution is to consider tiles with each dimension length being the smallest power of two greater than the minimum required length. Referencing the previous example, the tile size required for the  $(8, 8)$ -coloring on a 4D lattice would be  $64 \times 32^3$ . The drawback of this requirement is that tiles may become too large, and some of their dimensions (in particular, the one with displacement) may be longer than the size of the actual lattice. In such cases, we may limit the tile size in the offending dimension to  $D_i$ . This ensures valid coloring, although possibly with a few more colors, and also standardizes the number of tiles we need to pre-compute and store. In the example above, if the lattice is of size  $32 \times 64^3$ , then the size of the  $(8, 8)$ -coloring tile becomes  $32^4$ .

Table 3.2 shows the sizes of the tiles for different  $(p, k)$ -colorings chosen with the above policy for a 4-dimensional toroidal lattice of size  $32^3 \times 64$ , which is the size of the lattice used in our experiments in the next section. The table shows the displacement in the first direction for clarity, although our LQCD application requires it in the third dimension.

$k$	Displacement								
	0	1	2	3	4	5	6	7	8
1	$4^4$	$8 \times 4^3$	$8 \times 4^3$	$16 \times 4^3$	$16 \times 4^3$	$16 \times 4^3$	$16 \times 4^3$	$32 \times 4^3$	$32 \times 4^3$
2	$8^4$	$8^4$	$16 \times 8^3$	$16 \times 8^3$	$16 \times 8^3$	$16 \times 8^3$	$32 \times 8^3$	$32 \times 8^3$	$32 \times 8^3$
3	$8^4$	$16 \times 8^3$	$16 \times 8^3$	$16 \times 8^3$	$16 \times 8^3$	$32 \times 8^3$	$32 \times 8^3$	$32 \times 8^3$	$32 \times 8^3$
4	$16^4$	$16^4$	$16^4$	$16^4$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$
5	$16^4$	$16^4$	$16^4$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$
6	$16^4$	$16^4$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$
7	$16^4$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$	$32 \times 16^3$
8	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$
9	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$
10	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$	$32^4$

**Table 3.2:** Tile sizes for each  $(p, k)$ -coloring for a  $32^3 \times 64$  lattice with the displacement in the first dimension (corresponding to the  $z, x, y, t$  dimensions of the application).

### 3.4 Experiments

We have implemented our code in both C and MATLAB. The computation of all lattice tiles in Table 3.2 was performed with the C code. All tests were run on the Femto subcluster at William & Mary, where each compute node is a 32-core 960 Xeon Skylake with a clock speed of 2.1GHz. The timings for each of the  $(p, k)$ -colorings on a single thread are shown in Table 3.3, but the code can be easily parallelized. While iterating through each node must be sequential to avoid coloring conflicts, gathering the color labels of a single node's neighbors is a read-only process that can be done independently. For example, the maximum number of neighbors each node can have for an  $(8, 10)$ -coloring is 16,681, allowing for decent speedups. A red-black scheme can also be done in parallel, as the red and black nodes can be separated and colored independently.



$k$	Displacement								
	0	1	2	3	4	5	6	7	8
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.01	0.00	0.01	0.01	0.01	0.02	0.01	0.01
3	0.01	0.02	0.02	0.02	0.03	0.05	0.04	0.04	0.04
4	0.21	0.33	0.39	0.41	0.82	0.82	0.82	0.83	0.83
5	0.44	0.67	0.79	1.70	1.74	1.75	1.75	1.75	1.76
6	0.84	1.22	2.92	3.16	3.29	3.32	3.32	3.34	3.33
7	1.44	4.13	4.93	5.44	5.68	5.79	5.82	5.84	5.85
8	38.88	55.23	64.81	71.35	77.19	76.69	77.84	78.54	77.99
9	61.63	85.06	97.91	108.21	114.82	119.90	121.16	121.22	122.22
10	91.16	121.33	143.77	157.81	167.94	175.82	179.13	180.09	180.38

**Table 3.3:** Time (in seconds) to run each  $(p, k)$ -coloring with tile sizes outlined in Table 3.2 and the resulting number of colors is shown in Table 3.4.

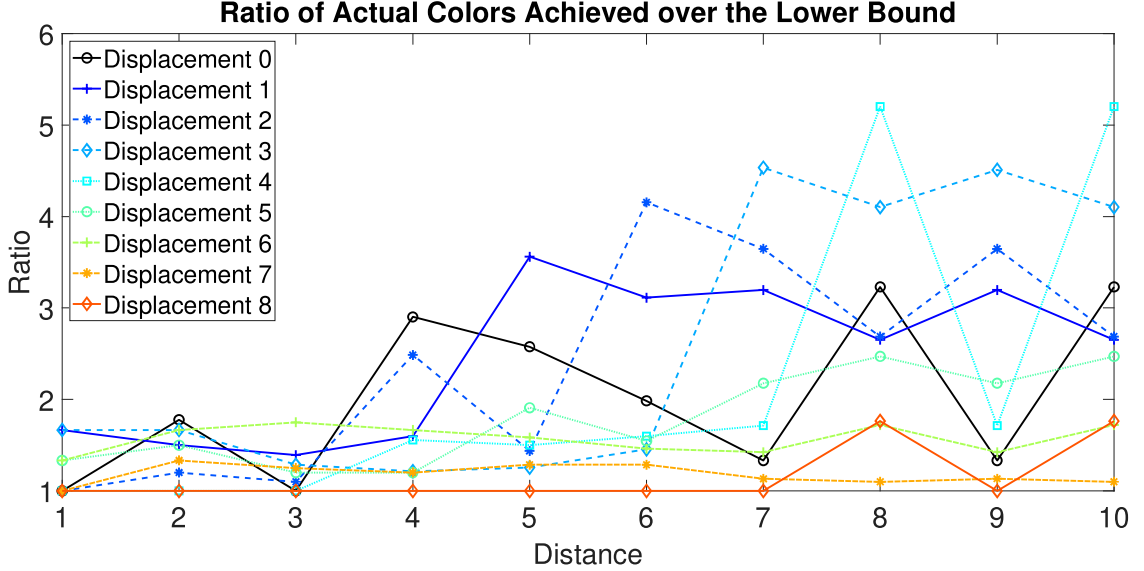
### 3.4.1 Number of Colors Computed

As the number of colors equates to the number of linear systems needing to be solved in Equation 3.1, we are interested in studying how close the number returned by the greedy algorithm is to the theoretical lower bounds summarized in Equation 3.13. As discussed in Subsection 3.3.5, the lower bounds are for lattices without boundary restrictions. We expect variability in the deviation from the lower bound depending on lattice size.

$k$	Displacement								
	0	1	2	3	4	5	6	7	8
1	2/2	5/3	4/4	5/3	3/3	4/3	4/4	3/3	3/3
2	16/9	9/6	6/5	10/6	4/4	6/4	5/3	4/3	3/3
3	16/16	32/23	11/10	9/7	8/8	6/5	7/4	5/4	4/4
4	119/41	64/40	92/37	17/14	14/9	12/10	10/6	6/5	4/4
5	170/66	324/91	96/64	64/51	27/18	21/11	19/12	9/7	6/6
6	256/129	442/142	586/141	128/88	104/65	34/22	19/13	18/14	8/8
7	256/192	815/255	795/218	866/192	192/112	172/79	37/26	17/15	16/16
8	1037/321	976/368	1024/381	1206/294	1254/241	336/136	160/93	33/30	30/17
9	1298/450	2031/579	1024/544	1760/507	1577/370	1556/291	288/160	128/107	52/34
10	2220/681	2462/790	3238/837	1922/720	2082/633	1976/446	1954/341	256/184	264/121

**Table 3.4:** The first number is the smallest number of colors achieved for distance- $k$ , displacement  $p$ , on the tiles of size as noted in Table 3.2. The second number is the lower bound for that  $(p, k)$  from Equation 3.13.

Table 3.4 shows the least amount of colors achieved between natural and red-black orderings for our different  $(p, k)$ -colorings. Next to this number is the theoretical lower bound for each  $(p, k)$  combination where  $p \in \{0, 1, \dots, 8\}$  and  $k \in \{1, 2, \dots, 10\}$ .



**Figure 3.6:** The ratio of the minimum number of colors achieved with a  $(p, k)$ -coloring to the theoretical lower bound in Table 3.4. Each column is a different displacement.

The ratio between the two numbers for all combinations is plotted in Figure 3.6. We observe that when  $p \geq k$ , the achieved number of colors is close to the lower bound as the coloring problem becomes one-dimensional, which provides significantly fewer clique constraints. However, once the two displaced neighborhoods begin to overlap, the number of constraints increases, and we see the toroidal boundary effects of the tiles. In our experiments,  $(p, k)$  combinations with ratios less than three and around 200-300 colors work the best.

### 3.4.2 Comparisons to Other Methods

Based on the tiles outlined in Table 3.2, we generated the probing vectors used in trace estimation experiments using the Chroma library from Jefferson Laboratory [29]. The  $32^3 \times 64$  lattice generated by Chroma uses a Clover fermion action with a quark mass of

-0.239. The gauge configuration is from the same ensemble listed as Ensemble B in [38]; details of which can be found in [105]. As suggested in [38], we use deflation with the 200 largest singular vectors of  $A^{-1}$ , which are computed using the PRIMME library [91]. The solution of each linear system is performed in single precision to relative residual accuracy of 1e-3 with the MG Proto library <sup>1</sup>, an adaptive multigrid solver, inside of Chroma.

We compare our displacement probing method against unprobed Hutchinson and CP without displacement. Because in LQCD each lattice point has 12 degrees of freedom (for all spin-color combinations), all methods perform a probing of these 12 components (called spin-color dilution in the literature [15]). Performing this dilution amounts to taking a Kronecker product of each probing or random vector with a  $12 \times 12$  identity matrix. Thus, each probing or random vector must solve twelve linear systems. We also assume that the matrix  $A$  has already been deflated with 200 singular triplets.

Let  $\mathbf{v}(P_p A^{-1})$  be a shorthand for the variance in Equation 3.2 for the matrix  $P_p A^{-1}$ , where  $P_p$  is the permutation matrix that places the elements of  $A^{-1}$  corresponding to displacement  $p$  in the main diagonal (clearly  $P_0 = I$ ). The unprobed Hutchinson method is run with  $s_1 = 1,000$  Rademacher vectors to estimate the trace and variance of  $P_p A^{-1}$ . For the probing with displacements and the CP methods, let  $H$  be the  $N \times m$  matrix with the required  $m$  probing vectors as columns, and considering  $s_2 = 10$  Rademacher vectors, construct the  $m \times s_2$  vectors  $V^{(1)}, \dots, V^{(s_2)}$  as in Subsection 3.2.2. These are used to estimate the trace and variance for each  $P_p A^{-1}$ .

To compare the methods meaningfully, we must consider their effect under the same number of linear systems solved. For the unprobed Hutchinson method, the computed variance of the  $s_1$  quadrature values computed in Equation 3.1 provides a good estimation of  $\mathbf{v}(P_p A^{-1})$ . Similarly, for the probing variants after  $s_2$  Hutchinson steps, we expect a good estimation of  $\mathbf{v}((P_p A^{-1}) \odot H H^T)$ . However, each of the  $s_2$  stochastic steps of the

---

<sup>1</sup><http://jeffersonlab.github.io/qphix> and [github.com/jeffersonlab/mg](https://github.com/jeffersonlab/mg)

probing variants solves  $m$  linear systems, which implies that the speedup is

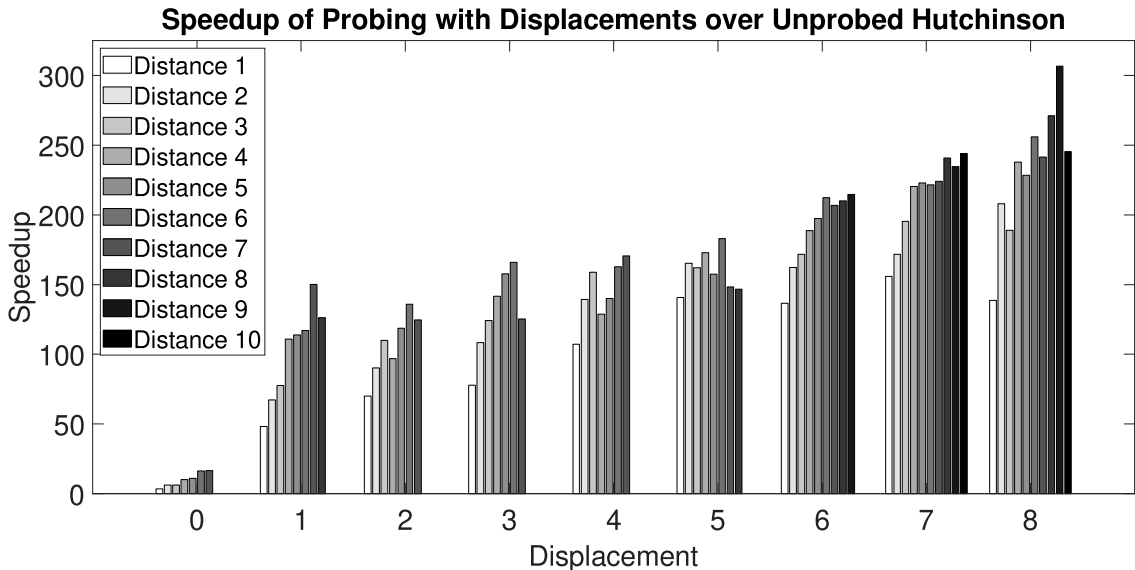
$$\text{Speedup over unprobed Hutchinson} = \frac{\mathfrak{v}(P_p A^{-1})}{m \times \mathfrak{v}((P_p A^{-1}) \odot H H^T)}. \quad (3.15)$$

Table 3.5 shows the detailed results for trace and variance estimations and speedups for Hutchinson and our new method for different combinations of  $p$  and  $k$ . The speedups for probing with displacements over unprobed Hutchinson are also graphed in Figure 3.7. We make a few observations. First, the larger the displacement  $p$ , the more significant the speedup of the new method performs over unprobed Hutchinson. Second, as mentioned before, distance-1 probing has the most considerable impact as it removes the main diagonal of  $A^{-1}$  and the elements at distance-1 away from the main diagonal. Third, the speedup increases with  $k$  but peaks at a certain distance, typically around  $k = 6$  for smaller displacements and around  $k = 9$  for larger displacements. This behavior is expected as the elements of  $A^{-1}$  decay at higher distances, making it less beneficial to probe them directly instead of randomly. Finally, for  $p = 0$ , probing is equivalent to CP and gives a speedup of 16 over unprobed Hutchinson, which is slightly better than our previous HP method, albeit giving up the hierarchical property.

To solve the problem with displacements, practitioners had previously attempted to use CP or HP [5] or a more localized hopping parameter expansion [107]. We want to show the improvements of our method over CP. Let  $m_p$  be the number of probing vectors produced in the  $(p, k)$ -coloring to form  $H_p$ . The  $m_0$  vectors forming  $H_0$  are the CP vectors, which could be used to reduce the estimator's variance for  $P_p A^{-1}$ . The speedup of probing with displacements over CP is then,

$$\text{Speedup over CP} = \frac{\mathfrak{v}((P_p A^{-1}) \odot H_0 H_0^T) \times m_0}{\mathfrak{v}((P_p A^{-1}) \odot H_p H_p^T) \times m_p}. \quad (3.16)$$

In Figure 3.8, we can see this speedup increasing with displacement, although, for small displacements, it decreases with distance. This is because CP builds its neighborhood

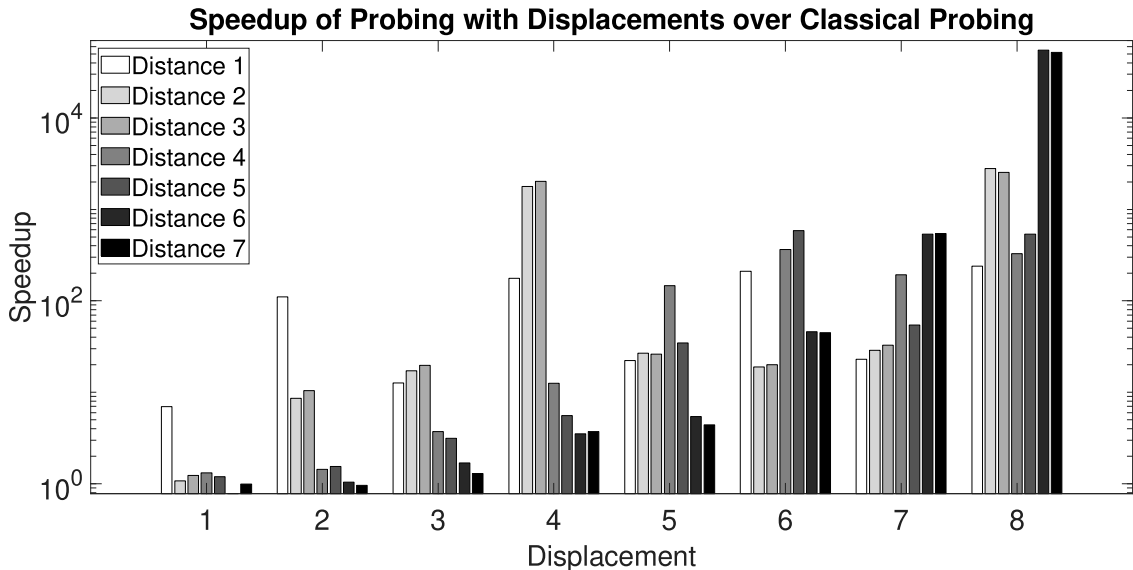


**Figure 3.7:** Speedups of probing with displacements over unprobed Hutchinson using each  $(p, k)$ -coloring from Table 3.5 to find  $\text{tr}(P_p A^{-1})$ .

outward from the new diagonal, so it can only eliminate the original main diagonal when  $k \geq p$ . As the displacement grows, the number of colors the new method needs to achieve a distance- $k$  coloring becomes much smaller. For example, a  $(0, 7)$ -coloring uses 256 colors, while an  $(8, 7)$ -coloring only uses 16. Therefore, even if CP eventually removes the high-magnitude elements, it can take many more probing vectors to do so.

### 3.4.3 Using one coloring for all displacements

Theorem 6 showed that a  $(p_0, k_0)$ -coloring would clear all nodes up to distance  $k = \max(0, k_0 - |p_0 - p|)$  for a displacement of  $p$ . Table 3.6 confirms this experimentally for the  $(8, 10)$ -coloring but also shows how many nodes are *not* annihilated beyond the distance described by the theorem. To obtain this, for each pair of  $(p, k)$ ,  $p = 0, \dots, 8$ ,  $k = 1, \dots, 12$ , we go through every node  $x$  in the lattice and compute the percentage of nodes exactly at distance- $k$  from  $x^+$  or  $x^-$  that share the same color label as  $x$ . These are distance- $k$  neighbors that are not annihilated by the  $(8, 10)$ -coloring. We report the average of this percentage over all  $N$  nodes. When the percentage is 0.00, it means that distance is "cleared", i.e., all nodes of that distance are annihilated from the variance.

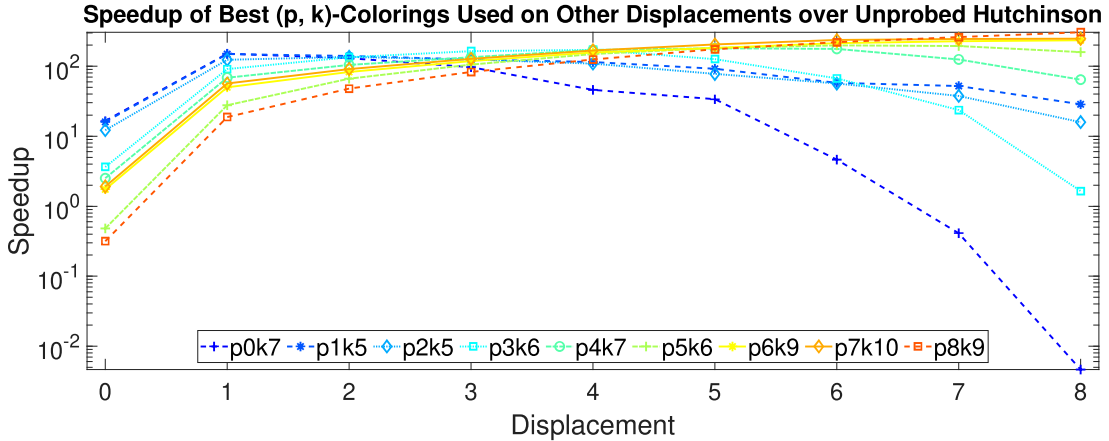


**Figure 3.8:** Speedup of probing with displacements with  $(p, k)$ -colorings over classical probing with  $(0, k)$ -colorings to find  $\text{tr}(P_p A^{-1})$  using Equation 3.16.

The presence of zeros for any  $k \leq 10 - |p - 8|$  confirms Theorem 6. For each  $p$ , we also observe a zero at distances  $4i + (k_0 - |p_0 - p|), \forall i \in \mathbb{Z}_+$  which may be attributed to wrap-around effects and/or the red-black ordering used for the  $(8, 10)$ -coloring. More importantly, however, the percentages of uncleared elements at larger distances are still very small, often less than 1%. This is because a coloring annihilates the distance- $k$  neighbors of all nodes of the same color. For example, if  $x_1$  and  $x_2$  have the same color, some neighbors of  $x_1$  may be further distance neighbors of  $x_2$  but are annihilated for this  $k$ .

Next, we study the effects of this strategy on variance reduction. For each  $p$ , we take the  $(p, k_p)$ -coloring that gives the best speedup over unprobed Hutchinson (from Figure 3.7) and use it to find the variance  $\mathbf{v}((P_n A^{-1}) \odot H_p H_p^T)$  for all other displacements  $n = 0, \dots, 8$ . Figure 3.9 shows nine lines, one for each  $p$ , plotting its speedup over the Hutchinson method for all  $n$ . Each line achieves its maximum speedup at  $n = p$  or for smaller  $p$ , at  $n = p + 1$ . It is unclear why this happens for smaller  $p$ , e.g., most pronounced for the  $(0, 7)$ -coloring, but it may have to do with the symmetrization. More importantly, the speedup does not reduce as steeply away from  $p$  as Theorem 6 would suggest because these colorings work

very well for nearby displacements and still work well for more distant ones as described in Table 3.6.



**Figure 3.9:** The speedups over unprobed Hutchinson for each  $(p, k_p)$ -coloring to find  $\text{tr}(P_n A^{-1})$ ,  $\forall n \in \{0, \dots, 8\}$

The above results help ascertain the efficiency of the approach, but they cannot help determine which coloring to use for performing all displacement experiments. There are two reasons. First, the speedups reported depend on the number of probing vectors used. For example, the  $(8, 9)$ -coloring obtains a speedup of 300 at  $p = 8$  because it uses only 52 colors. Its variance is four times larger than  $(7, 10)$ -coloring, which uses 250 vectors and thus gets a lower speedup of 250. The  $(7, 10)$ -coloring would be better for a more accurate answer.

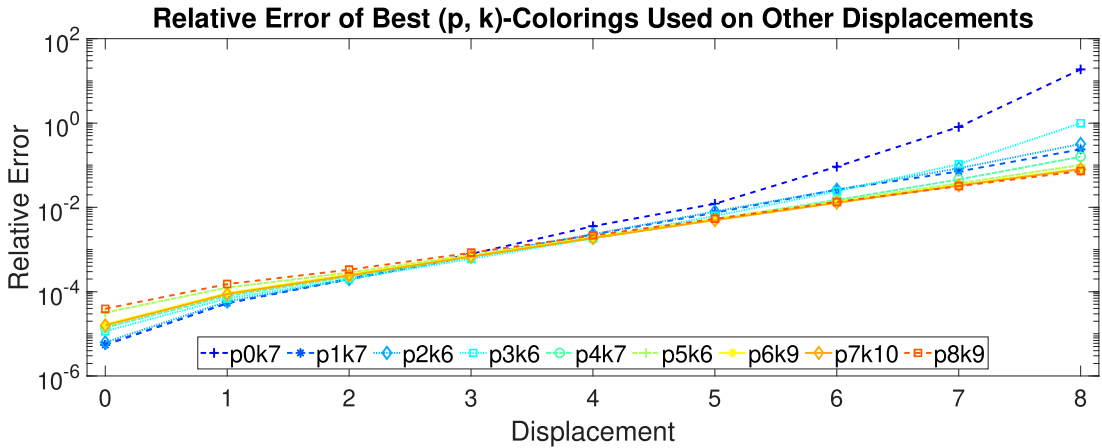
Second, a smaller variance is only meaningful relative to the value of the trace and traces for different displacements vary significantly. In Table 3.5, we see that a variance of 3.275 for the  $(0, 4)$ -coloring gives five digits of accuracy for the trace of  $p = 0$ , while a variance of 2.332 for the  $(8, 9)$ -coloring hardly attains a digit for the trace of  $p = 8$ .

Therefore, to compare colorings over different displacements, we introduce the normalized relative error metric, which normalizes with respect to both the trace and the number of probing vectors needed. As before, for each  $p$  we pick the  $(p, k_p)$ -coloring with the best speedup over unprobed Hutchinson. Let  $m_p$  be the number of colors it requires, and

let  $M$  be the maximum number of colors over all colorings being compared (in this case,  $M = 815$ ). Then, for all  $n = 0, \dots, 9$ , the normalized relative error is given by,

$$\frac{\sqrt{\mathbf{v}((P_n A^{-1}) \odot H_p H_p^T) \frac{m_p}{M}}}{\mathbf{tr}(P_n A^{-1})}. \quad (3.17)$$

The normalization to  $M$  ensures all colorings are compared as if they use the same number of probing vectors. Figure 3.10 shows these results.



**Figure 3.10:** The relative error (Equation 3.17) for each  $(p, k_p)$ -coloring used to find  $\mathbf{tr}(P_n A^{-1})$ ,  $\forall n \in \{0, \dots, 8\}$ .

The fact that the trace decreases significantly in higher displacements provides a much clearer evaluation picture. For  $2 \leq n \leq 6$ , all  $(p, k_p)$ -colorings have similar normalized relative errors. However, the colorings from larger displacements, e.g.,  $(7, 10)$  or  $(8, 9)$ , yield at least 1 to 2.5 digits better accuracy for the same amount of work than colorings from small displacements. Because for displacements less than 4, the errors are already very small, meaning the effort must be focused on the small traces of higher displacements. Therefore, it is best to use the  $(7, 10)$ -coloring for all displacements and increase its distance if needed.



### 3.5 Chapter Summary

In this work, we have extended the idea of probing for variance reduction of the Hutchinson trace estimator to the case of permuted matrices and, in particular, when this permutation corresponds to a lattice displacement  $p$ . This has an important application on disconnected diagrams in LQCD. Our method works by computing a distance- $k$  coloring, not of the original neighborhood of each lattice point  $x$ , but rather the points within a distance  $k$  around centers  $x \pm p$ .

We have provided a lower bound of the number of colors needed for a particular  $(p, k)$ -coloring and discussed the impact of the lattice size on the number of colors achieved. We have also studied theoretically and experimentally the effect of using a single  $(p, k)$ -coloring for displacements other than  $p$ , showing that the variance reduction of using probing with displacements is orders of magnitude lower than solely using random noise vectors or using classical probing that does not consider the displacement. Also, as expected, the trace is smaller as the displacement increases, meaning that a  $(p, k)$ -coloring for larger  $p$  needs to be computed and reused for lower  $p$ . This gives approximately an additional 10-fold speedup for the LQCD application.

A few open problems could be considered further. The node visitation orderings we considered in the greedy coloring approach did not vary substantially in the resulting number of colors, staying within a factor of 3 from the lower bound. It is unclear whether a different ordering can considerably reduce the current number of colors. A second direction is to study the effect of the lattice or tile size on the coloring. Understanding this theoretically rather than experimentally and providing a lower bound on the number of colors based on a finite lattice size could be helpful in understanding the limitations of the current approach. Finally, it is worth extending the theory and algorithms to the case where the decay of the elements in the matrix inverse depends on the L2 distance, which is closer to what LQCD theory predicts for long-range distances.

$p$	$k$	Approx. Trace	1,000 RNVs w/o Probing		10 RNVs w/ Probing		Speedup
			Variance		Colors	Variance	
0	1	6,339,643.7		249,827.7	2	35,075.3	3.56
	2				16	2,502.5	6.24
	3				16	2,501.2	6.24
	4				119	209.6	10.02
	5				170	134.2	10.95
	6				256	59.4	16.43
	7				256	59.1	16.50
1	1	652,636.1		2,341,455.9	5	9,721.2	48.17
	2				9	3,861.4	67.38
	3				32	943.5	77.55
	4				64	330.2	110.78
	5				324	63.4	113.94
	6				442	45.2	117.08
	7				815	19.1	150.13
	8				976	19.0	126.21
2	1	185,764.9		2,360,726.0	4	8,415.7	70.13
	2				6	4,362.0	90.20
	3				11	1,949.6	110.08
	4				92	264.9	96.87
	5				96	207.2	118.70
	6				586	29.6	135.95
	7				795	23.8	124.73
3	1	56,047.8		2,364,612.0	5	6,076.0	77.84
	2				10	2,180.4	108.45
	3				9	2,115.0	124.22
	4				17	982.1	141.63
	5				64	234.1	157.82
	6				128	111.4	165.89
	7				866	21.8	125.41
4	1	17,893.6		2,388,516.5	3	7,420.1	107.30
	2				4	4,285.6	139.33
	3				8	1,880.1	158.81
	4				14	1,323.9	128.87
	5				27	631.8	140.02
	6				104	141.2	162.67
	7				192	72.9	170.66
5	1	6,059.5		2,358,840.1	4	4,186.9	140.85
	2				6	2,379.5	165.22
	3				6	2,425.6	162.08
	4				12	1,137.2	172.86
	5				21	712.8	157.58
	6				34	379.3	182.90
	7				172	92.4	148.40
	8				332	48.4	146.65
6	1	2,183.3		2,392,640.1	4	4,375.9	136.70
	2				5	2,948.7	162.29
	3				7	1,990.8	171.69
	4				10	1,267.6	188.75
	5				19	638.1	197.36
	6				19	592.9	212.40
	7				37	312.5	206.92
	8				160	71.2	210.12
	9				288	38.7	214.56
7	1	836.9		2,378,138.9	3	5,081.5	156.00
	2				4	3,463.9	171.64
	3				5	2,435.2	195.31
	4				6	1,798.8	220.35
	5				9	1,185.9	222.82
	6				18	596.2	221.62
	7				17	624.3	224.07
	8				33	299.3	240.76
	9				128	79.2	234.68
	10				256	38.1	243.95
8	1	339.3		2,381,007.2	3	5,719.0	138.78
	2				3	3,814.9	208.04
	3				4	3,151.8	188.86
	4				4	2,502.7	237.85
	5				6	1,737.5	228.40
	6				8	1,162.5	256.02
	7				16	616.1	241.53
	8				30	292.7	271.18
	9				52	149.2	306.80
	10				264	36.7	245.51

**Table 3.5:** The estimation of traces and variances for 1,000 RNVs run without probing for different values of  $p$  and  $k$  compared to probing with displacements and 10 RNVs.

$k$	Displacement								
	0	1	2	3	4	5	6	7	8
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	4.55	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	9.38	1.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	3.33	3.41	0.63	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	1.40	1.76	0.35	0.00	0.00	0.00	0.00	0.00
7	2.52	0.00	0.78	1.05	0.22	0.00	0.00	0.00	0.00
8	4.69	1.29	0.00	0.49	0.69	0.15	0.00	0.00	0.00
9	2.01	2.56	0.79	0.00	0.33	0.47	0.10	0.00	0.00
10	0.00	1.16	1.64	0.53	0.00	0.24	0.34	0.07	0.00
11	1.66	0.00	0.77	1.14	0.38	0.00	0.18	0.26	0.06
12	3.13	1.05	0.00	0.54	0.83	0.29	0.00	0.13	0.20

**Table 3.6:** The average percentage of neighbors at exactly distance- $k$  that do not get eliminated from the trace estimator when using a  $(8, 10)$ -coloring to find other displacements. The lattice size used is  $32^3 \times 64$  with a tile size of  $32^4$ .

## Chapter 4

# Exploring Krylov Methods with Sketched Rayleigh-Ritz in PRIMME

### 4.1 Introduction

The field of numerical linear algebra (NLA) encompasses a broad range of problems. The most significant of these include solving linear systems of equations, approximating the eigenpairs of a matrix, computing the singular value decomposition (SVD) of a matrix, and using orthogonalization techniques [62]. While advancements in modern computing architectures and high-performance computing can speed up these computations, the increasing size of datasets makes these problems more challenging to solve as the number of computational resources required grows.

To address these increasing problem sizes, practitioners often adopt randomized techniques to approximate solutions, as they are fast and reliable [62]. Randomized methods began rising in popularity starting in the 1980s and have since been used in areas such as solving least-squares problems [80], SVD [31], orthogonalization [14], matrix preconditioning [34], data streaming [8, 7], and finding low-rank matrix approximations [75, 35].

In Chapter 2.5, we discuss one such randomized method, referred to as *sketching*, which uses random subspace embedding matrices  $S \in \mathbb{F}^{s \times n}$  to project high-dimensional  $n \times d$

least-squares problems, with  $d \ll n$ , into low-dimensional  $s \times d$  ones with minimal loss in solution accuracy [84]. Here,  $s$  is referred to as the embedding dimension and is selected using the Johnson-Lindenstrauss Lemma, which states  $s = O(\frac{\log n}{\epsilon^2})$  for some distortion factor  $\epsilon \in (0, 1)$  [53, 25].

In [72], Nakatsukasa and Tropp observed that the Rayleigh-Ritz (RR) method, used to extract approximate eigenpairs of a matrix  $A \in \mathbb{F}^{n \times n}$  from a basis  $V \in \mathbb{F}^{n \times d}$  as described in Chapter 2.4, could be written as the least-squares problem

$$\text{minimize}_{H \in \mathbb{F}^{d \times d}} \|AV - VH\|, \quad (4.1)$$

outlined in [77]. The eigenvalue problem RR solves is on the solution to this minimization, i.e.,  $\hat{H}y = \hat{\lambda}y$ , where  $\hat{H} = V^\dagger AV$  with  $\dagger$  denoting the Moore-Penrose pseudoinverse. This formulation of RR opens it up to the application of sketching by transforming Equation 4.1 into

$$\text{minimize}_{M \in \mathbb{F}^{d \times d}} \|S(AV - VM)\|, \quad (4.2)$$

with solution  $\hat{M} = (SV)^\dagger(SAV)$  [72]. When the basis  $V$  is constructed using a Krylov method, the primary benefit of utilizing sRR is that the approximate eigenpairs of  $A$  can be accurately extracted from the basis without  $V$  being orthonormal.

The RR procedure is a Galerkin method that extracts approximate eigenpairs  $(\hat{x}, \hat{\lambda})$  of  $A$  from  $V$  with the constraint that the residuals of the eigenpair estimates are orthogonal to all columns in  $V$  [32], i.e.,

$$A\hat{x}_i - \hat{x}_i\hat{\lambda}_i \perp V \quad \text{for } i = 1, 2, \dots, d. \quad (4.3)$$

This is done by solving the eigenproblem on  $V^H AV \in \mathbb{F}^{d \times d}$  with an orthogonal  $V$ , where the resulting eigenpairs  $(y, \hat{\lambda})$  correspond to the approximate eigenpairs, or *Ritz pairs*, of  $A$ ,  $(Vy, \hat{\lambda})$ .

To maintain the numerical stability of a Krylov method and ensure accurate extraction of the Ritz pairs from the basis  $V$ ,  $V$  should be orthonormal, i.e.,  $V^H V = I$ . In theory, short-term recurrence methods such as unrestarted Lanczos will ensure  $V$  remains orthonormal during its construction, providing optimal convergence properties when used without restarting. However, floating point errors in practice result in orthogonality loss as eigenpairs converge. Once this happens, repeated directions of converged eigenpairs begin re-entering the basis, which in turn causes an increase in the condition number of  $V$ ,  $\kappa(V)$ . These repeated directions may result in the extraction of inaccurate or fictitious eigenpair approximations from  $V$  during the RR procedure or cause a slowdown or stagnation in convergence if managing to extract accurate eigenpairs. Orthogonality of  $V$  can be maintained by using reorthogonalization techniques such as Classical or Modified Gram-Schmidt [19] or Householder reflections [50], which have a computational complexity of  $O(nd^2)$ . However, frequent application of these methods may result in a computational bottleneck.

When seeking many eigenpairs, short-term recurrence methods such as JDQMR or GD+k are not optimal, as orthogonality will inevitably be lost, and neither technique can match the convergence of unrestarted Lanczos with full orthogonalization [82]. However, seeking many eigenpair approximations using full-orthogonalization Lanczos (FO Lanczos) requires the Krylov method to construct a large basis to build sufficient convergence information, making frequent orthogonalization computationally prohibitive.

Strategies such as partial orthogonalization [44, 77], selective reorthogonalization [86], and locking [55] have been developed to reduce the amount of orthogonalization needed when constructing  $V$ . However, [72] suggests that the cost of orthogonalization can be reduced using sketching, which involves orthogonalizing a lower dimensional space.

As previously mentioned, classical RR extracts the approximate eigenpairs of a matrix  $A \in \mathbb{F}^{n \times n}$  by solving the eigenvalue problem on  $V^H A V \in \mathbb{F}^{d \times d}$ , where  $V \in \mathbb{F}^{n \times d}$  is an orthogonal basis constructed via a Krylov method. If  $V$  is not orthogonal, RR can still

extract accurate eigenpairs by solving the generalized eigenproblem

$$V^H A V y = \hat{\lambda} V^H V y, \quad (4.4)$$

which requires the inversion or factorization of  $V^H V$ , and has a similar computational cost to orthogonalizing  $V$ .

Let  $S \in \mathbb{F}^{s \times n}$  be the sketching matrix to be applied to  $V$ ,  $C = S V \in \mathbb{F}^{s \times d}$  be the sketched basis, and  $D = S A V \in \mathbb{F}^{s \times d}$ . While sRR does not avoid orthogonalization entirely, it does reduce the cost. The eigenpairs  $(y, \hat{\lambda})$  returned from solving the eigenproblem

$$C^H D y = \hat{\lambda} y \quad (4.5)$$

will not correspond to the approximate eigenpairs of  $A$  if  $\kappa(C) > 1$ . However, using the same logic as in Equation 4.4, we can instead solve the generalized eigenvalue problem

$$U^H D y = \hat{\lambda} T y, \quad (4.6)$$

where  $U \in \mathbb{F}^{s \times d}$  and  $T \in \mathbb{F}^{d \times d}$  are the factors from the QR decomposition of  $C$  [40]. Because using an orthogonalization technique such as QR on  $C \in \mathbb{F}^{s \times d}$  is less expensive than on  $V \in \mathbb{F}^{n \times d}$ , sRR efficiently reduces the computational cost associated with frequent orthogonalization of  $V$  when sketching is not used. Furthermore, the residual norms of the Ritz pairs returned from sRR should be within a constant factor of residual norms of its non-sketching counterparts [84]. According to [72], sRR allows the extraction of accurate Ritz pairs as long as  $\kappa(V) \lesssim \epsilon_{\text{mach}}^{-1}$ .

This work investigates the performance benefits of integrating sRR alongside two prevalent Krylov methods – Lanczos and Generalized Davidson (GD) – utilizing the high-performance software library PRIMME [91]. Shown in Algorithm 2, the Lanczos method was chosen due to its simplicity, inexpensive iteration cost, and optimal convergence properties over all matrix-vector-based methods when used without restarting for Hermitian

eigenvalue problems. We begin by exploring the use of sRR with Thick-Restarted Lanczos (TRL), followed by unrestarted Lanczos. After this, we turn our attention to the Generalized Davidson (GD) method, depicted in Algorithm 3, as it is robust across different problem types and allows the integration of preconditioning, which can accelerate convergence.

We implemented the unrestarted Lanczos iterative method and sRR into PRIMME before running performance and convergence tests to compare the effects of utilizing Lanczos and GD with sRR against their non-sketching counterparts. Following this, we offer insights into the observed behavior, considering varying factors such as the density of a matrix’s eigenspectrum, the maximum basis size, and the number of sought-after eigenpairs.

The remainder of the chapter is organized as follows: Section 4.2 introduces additional background information on sketching. Subsequently, we present an overview of the advantages and restrictions of the Lanczos algorithm with sketching, along with comparative results in Sections 4.4 and 4.5. Similarly, the Generalized Davidson algorithm with sketching is discussed in Sections 4.6 and 4.7. Finally, we will summarize this chapter in Section 4.8.

## 4.2 Subspace Embeddings

In Chapter 2.5, we introduced the concept of using randomized subspace embeddings for projecting high-dimensional least-squares problems into lower-dimensional ones with minimal loss in accuracy. We then explained how these random projection variants, referred to as sketched methods, could be utilized when extracting eigenpairs of a matrix from a basis  $V$  constructed via a Krylov method.

Two common approaches are often considered when constructing the randomized subspace embedding, or “sketching”, matrix  $S$ : the *sparse dimension reduction map* and the *subsampling random Fourier transform*.



### 4.2.1 Sparse Dimension Reduction Map

The sparse dimension reduction map [64, 73], also known as *Sparse Maps*, is defined as:

$$S = \frac{1}{\sqrt{s}}[s_1, s_2, \dots, s_n] \in \mathbb{F}^{s \times n}. \quad (4.7)$$

Here, the sketching matrix  $S \in \mathbb{F}^{s \times n}$  consists of statistically independent columns, with each column  $s_i$  comprising exactly  $z$  nonzero elements, meaning  $\mathbf{nnz}(S) = z \cdot n$ . Each nonzero element of  $S$  is either drawn from the Steinhaus distribution<sup>1</sup> when handling complex matrices, or chosen as  $\pm 1$  with 0.5 probability when dealing with real matrices.

Leveraging a software library capable of sparse matrix operations, the computational cost of applying  $S$  to some matrix  $A \in \mathbb{C}^{n \times n}$  is  $O(z * \mathbf{nnz}(A))$ . When applying  $S$  to a basis  $V \in \mathbb{F}^{n \times d}$ , with  $d$  being the maximum basis size, we adhere to the conventions outlined in Nakatsukasa and Tropp's manuscript by setting  $z = \lceil 2 \log(d + 1) \rceil$  for reliability [72].

### 4.2.2 Subsampled Random Fourier Transform

The subsampled random Fourier transform (SRFT) [4, 102] is defined as:

$$S = \sqrt{\frac{n}{s}} D F E \in \mathbb{F}^{s \times n}. \quad (4.8)$$

where:

- $D \in \mathbb{F}^{s \times n}$  is a diagonal projector such that each row only contains one non-zero element,  $\frac{1}{\sqrt{s}}$ . This non-zero column index is chosen independently at random from  $\{1, 2, \dots, n\}$  for each row.
- $F \in \mathbb{F}^{n \times n}$  represents the unnormalized discrete Fourier transform.

---

<sup>1</sup>The Steinhaus distribution is uniform on the complex unit circle  $\{x \in \mathbb{C} \mid |x| = 1\}$

- $E \in \mathbb{F}^{n \times n}$  is a diagonal matrix with independent entries drawn from the Steinhaus distribution when dealing with complex matrices, or  $\pm 1$  with probability 0.5 for real matrices.

The computational cost for applying this  $S$  to a basis  $V \in \mathbb{F}^{n \times d}$  is  $O(nd \log d)$  [102].

### 4.3 Sketched Rayleigh-Ritz

As discussed in Section 4.1, sRR reduces the amount of orthogonalization needed to extract accurate eigenpairs approximations of a matrix  $A \in \mathbb{F}^{n \times n}$  from a basis  $V \in \mathbb{F}^{n \times d}$  by orthogonalizing a low-dimensional space,  $C = SV \in \mathbb{F}^{s \times d}$ , resulting in Equation 4.6.

In this work, we construct the sketching matrix  $S \in \mathbb{F}^{s \times n}$  using the Sparse Maps method described in Section 4.2.1. Following the convention outlined in [72], we set the embedding dimension  $s$  equal to four times the maximum basis size, i.e.,  $4d$ . This results in the distortion factor being  $\epsilon \approx \frac{1}{2}$ .

When seeking many eigenpairs and performing no orthogonalization on  $V$ ,  $\kappa(V)$  will eventually exceed  $\epsilon_{\text{mach}}^{-1}$ , rendering sRR ineffective. While orthogonalization is the only definitive method to address  $V$ 's rapidly increasing conditioning number, there are two strategies sRR can use to extract Ritz pairs even when  $\kappa(V) \geq \epsilon_{\text{mach}}^{-1}$ : whitening [80] and stabilization [72]. Whitening performs a pseudo-orthogonalization technique on  $V$  utilizing the QR decomposition of the sketched basis  $C$ , reducing  $\kappa(V) \approx 1$ . Stabilization, however, does not change the basis  $V$  but instead computes the truncated SVD (tSVD) of  $C$  before using the decomposition to solve a generalized eigenvalue problem. More details about these methods are introduced in Sections 4.3.1 and 4.3.2, respectively.

**Algorithm 7:** Sketched Rayleigh-Ritz

**Input:**

$S \in \mathbb{F}^{s \times n}$  = The sketching matrix

$V \in \mathbb{F}^n$  = The Krylov basis of  $A$

$W \in \mathbb{F}^n$  = The projected Krylov basis ( $W = AV$ )

**Output:**

$X \in \mathbb{F}^{n \times d}$  = The approximate eigenvectors of  $A$

$\Lambda \in \mathbb{F}^d$  = The approximate eigenvalues of  $A$

$\text{resNorms} \in \mathbb{F}^d$  = The sketched residual norms of the eigenpair estimates

Sketched\_RR( $S, V, W$ )

- 1  $C = SV$ ;
- 2  $D = SW$ ;
- # Determine whether we need to use stabilization
- 3 **if**  $\text{cond}(SV) \geq \epsilon_{\text{mach}}^{-1}$  **then**
  - # Remove problematic parts of the basis using the truncated SVD
  - 4  $[U_{\text{SVD}}, \Sigma_{\text{SVD}}, V_{\text{SVD}}^H] = \text{truncated\_SVD}(C)$ ;
  - 5  $[\bar{Y}, \Lambda] = \text{eig}(U_{\text{SVD}}^H D V_{\text{SVD}}, \Sigma_{\text{SVD}})$ ;
  - 6  $Y = V_{\text{SVD}} \bar{Y}$ ;
- 7 **else**
  - 8  $[U, T] = \text{qr}(C, 0)$ ;
  - 9  $[Y, \Lambda] = \text{eig}(U^H D, T)$ ;
- # Compute the sketched residuals
- 10  $\text{resNorms}(i) = \|DY(:, i) - CY(:, i)\Lambda(i)\| / \|CY(:, i)\|$  for  $i = 1, 2, \dots, d$
- 11  $X = VY$ ;
- 12  $X(:, i) = X(:, i) / \|X(:, i)\|$  for  $i = 1, 2, \dots, d$  # Normalize Ritz vectors
- 13 **return**  $[X, \Lambda, \text{resNorms}]$ ;

Algorithm 7 illustrates the sRR process. This algorithm takes in three parameters: the basis  $V \in \mathbb{F}^{n \times d}$ , the projected basis  $W = AV \in \mathbb{F}^{n \times d}$ , and the sketching matrix  $S \in \mathbb{F}^{s \times n}$ . The process begins by determining the condition number of the sketched basis  $C$ , as, according to [80],  $\kappa(C) \approx \kappa(V)$  according to the equation

$$\frac{1 - \epsilon}{1 + \epsilon} \cdot \kappa(C) \leq \kappa(V) \leq \frac{1 + \epsilon}{1 - \epsilon} \cdot \kappa(C). \quad (4.9)$$

If  $\kappa(C) \geq \epsilon_{\text{mach}}^{-1}$ , stabilization is invoked. Otherwise, the economy QR decomposition of  $C$  is used to solve the generalized eigenproblem in Equation 4.6.

Once we obtain the Ritz pairs  $(\hat{x}_i, \hat{\lambda}_i)$  for  $i = 1, 2, \dots, d$  from Equation 4.6, their approximate residual norms can be computed,

$$\text{resNorms} = \|(D\hat{x}_i - \hat{\lambda}_i C\hat{x}_i)\|_2 / \|C\hat{x}_i\|_2 \quad \text{for } i = 1, 2, \dots, d \quad (4.10)$$

as illustrated on line 10 of Algorithm 7, where  $D = SAV \in \mathbb{F}^{s \times d}$ .

While the Ritz vectors are computed the same way as in classical RR, the sketched Ritz vectors,  $\hat{x}_i$  for  $i = 1, 2, \dots, d$ , may need to be normalized, as shown on line 12 of Algorithm 7.

### 4.3.1 Whitening

In 2008, Rokhlin and Tygert introduced the concept of using a randomized embedding to cheaply precondition an overdetermined linear least-squares regression [80]. This preconditioning technique can be used to approximately orthogonalize, or “whiten”, a basis  $V \in \mathbb{F}^{n \times d}$  when  $\kappa(V)$  is large.

Assuming  $S \in \mathbb{F}^{s \times n}$  is the sketching matrix and  $C = SV \in \mathbb{F}^{s \times d}$  is the sketched basis, the economy QR is performed on  $C$  such that  $C = UT$ . Utilizing the upper triangular matrix  $T \in \mathbb{F}^{d \times d}$ , whitening can then be performed on  $V$ :

$$\bar{V} = VT^{-1}. \quad (4.11)$$

The condition number of the whitened basis  $\bar{V}$  will satisfy [72]

$$\kappa(\bar{V}) = \frac{\sigma_{\max}(\bar{V})}{\sigma_{\min}(\bar{V})} \leq \frac{1 + \epsilon}{1 - \epsilon}, \quad (4.12)$$

where  $\sigma_{\max}(\bar{V})$  and  $\sigma_{\min}(\bar{V})$  are the maximum and minimum singular values of  $\bar{V}$ , respectively.

Whitening is a temporary solution to manage the  $\kappa(V)$ . MATLAB experiments, shown in Section 4.4, revealed that whitening only sets  $\kappa(V) \approx 1$  for an iteration or two before  $\kappa(V)$  again exceeds  $\epsilon_{\text{mach}}^{-1}$ . This remains an active area of research with ongoing investigations into other approximate orthogonalization methods using subspace embeddings [34, 14].

### 4.3.2 Stabilization

Referring to the condition number diagnostic in Equation 4.9, it becomes evident that if the sketched basis  $C \in \mathbb{F}^{s \times d}$  is poorly conditioned, the basis  $V \in \mathbb{F}^{n \times d}$  will also exhibit poor conditioning. Instead of pseudo-orthogonalizing  $V$  to reduce its condition number before recomputing  $C$  and solving the sRR problem, an alternative approach can be used: stabilization [72].

Stabilization involves computing the tSVD of  $C$  such that [49]:

$$C = U_0 \Sigma_0 V_0^H + U_1 \Sigma_1 V_1^H \quad (4.13)$$

where  $U_0 \in \mathbb{F}^{n \times d_t}$  and  $U_1 \in \mathbb{F}^{n \times (d-d_t)}$  contain the left singular vectors of  $C$  as columns,  $V_0 \in \mathbb{F}^{n \times d_t}$  and  $V_1 \in \mathbb{F}^{n \times (d-d_t)}$  contain the right singular vectors, and  $\Sigma_0 \in \mathbb{F}^{d \times d_t}$  and  $\Sigma_1 \in \mathbb{F}^{d \times (d-d_t)}$  are diagonal matrices with singular values  $\sigma_i$  as entries in descending order.

In our tSVD implementation,  $U_1$ ,  $\Sigma_1$ , and  $V_1$  consist of all singular triplets where

$$\frac{\sigma_{\max}}{\sigma_i} \gtrsim \epsilon_{\text{mach}}^{-1} \quad \text{for } i = 1, 2, \dots, d, \quad (4.14)$$

ensuring  $\kappa(U_0 \Sigma_0 V_0^H) \lesssim \epsilon_{\text{mach}}^{-1}$ .

The matrices  $U_0$ ,  $\Sigma_0$ , and  $V_0$  can then be used to construct the  $d_t \times d_t$  generalized eigenvalue problem

$$U_0^H D V_0 y = \hat{\lambda} \Sigma_0 y, \quad (4.15)$$

where  $D = SAV \in \mathbb{F}^{s \times d}$ . For efficient performance, the QZ algorithm is recommended for large, sparse matrices [72, 44].

All resulting eigenpairs  $(y_i, \hat{\lambda}_i)$  from Equation 4.15 correspond to the sRR eigenpair  $(V_0 y_i, \hat{\lambda}_i)$ , which in turn yield the approximate Ritz pairs of  $A$ ,  $(V V_0 y_i, \hat{\lambda}_i)$  for  $i = 1, 2, \dots, d_t$ . According to [72], utilizing stabilization should not alter the computational complexity of sRR,  $O(d^3 + nd \log d)$ .

## 4.4 Lanczos with Sketched Rayleigh-Ritz

As highlighted in Section 4.1, maintaining the orthogonality of the basis  $V \in \mathbb{F}^{n \times d}$  is crucial to avoid slowdown or stagnation in convergence as  $\kappa(V)$  increases. However, performing frequent orthogonalization on a large basis can become computationally prohibitive, motivating sketching. sRR shows promise in this regard due to its ability to extract accurate Ritz pairs of  $A$  without requiring  $V$  to be orthogonal, provided that  $\kappa(V) < \epsilon_{\text{mach}}^{-1}$ .

Algorithm 2 outlines the conventional 3-term Lanczos method with classical RR. Adapting this to integrate sRR requires minimal modifications to the algorithm itself, with the primary distinction being that retaining the Hessenberg matrix  $H \in \mathbb{F}^{d \times d}$  is no longer necessary since sRR does not use  $H$  for eigenpair approximations. Instead, we construct the basis  $V$  as usual before invoking the sRR method to extract eigenpairs using Equation 4.6. This updated procedure is illustrated in Algorithm 8.



MATLAB experiments were conducted to assess the efficiency of whitening and stabilization for handling ill-conditioned problems. RR and sRR were performed every iteration to monitor convergence behavior, although in practice, these methods only need to be performed once the basis has been fully constructed. To compare whitening and stabilization, we invoke 3-term Lanczos with sRR using whitening when  $\kappa(V) > \epsilon_{\text{mach}}^{-1}$  and compare it against 3-term Lanczos with sRR using stabilization to remove all singular triplets such that  $\frac{\sigma_1}{\sigma_i} > \epsilon_{\text{mach}}^{-1} \cdot 1E - 2$  for  $i = 1, 2, \dots, d$ .

The input parameters to 3-term Lanczos are as follows:

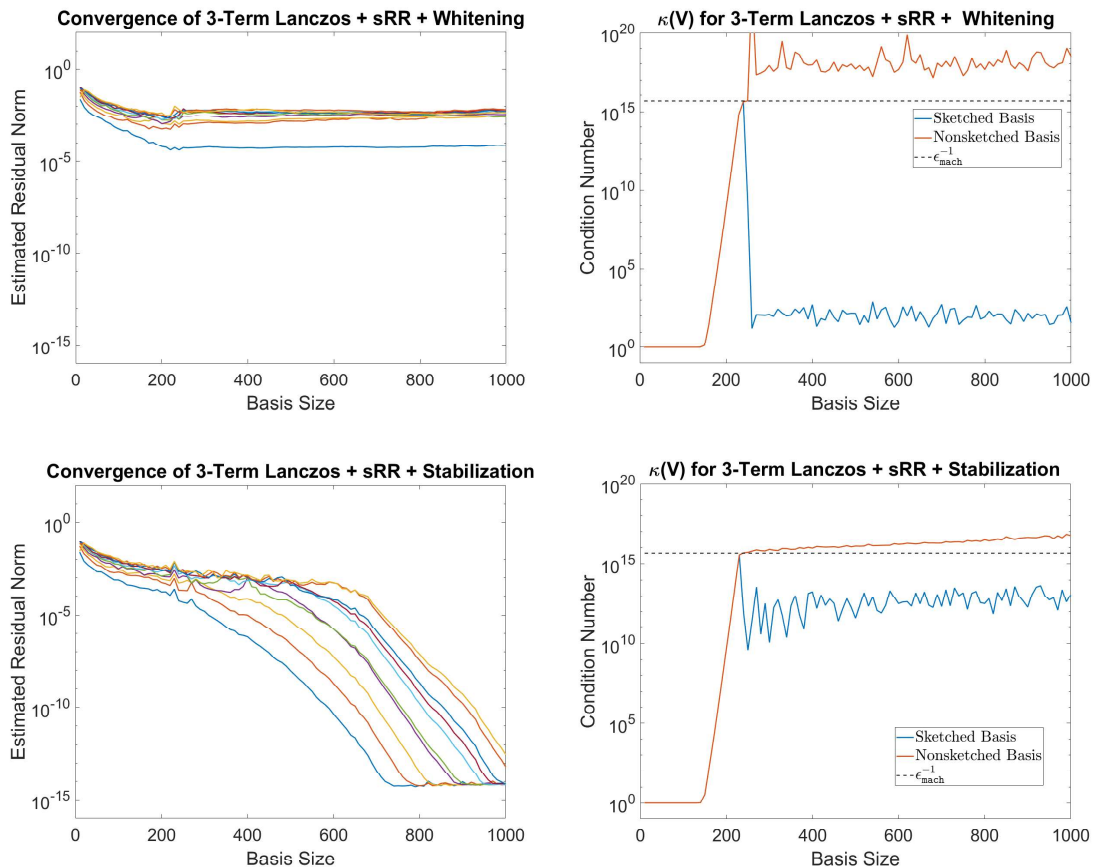
- Input matrix  $A = \text{diag}(\sqrt{1 : 5000})$
- Maximum basis size = 1,000
- Number of largest magnitude eigenvalues sought = 10
- Convergence tolerance =  $\epsilon_{\text{mach}}$
- $\epsilon_{\text{mach}} = 2.2204E-16$
- Subspace embedding dimension  $s = 4 \times$  Maximum basis size = 4,000

Figure 4.1 presents two results from these MATLAB experiments. The left figures illustrate the convergence behavior of the residual norms corresponding to the ten largest magnitude eigenvalues, and the right figures display how the condition number of the basis changes as vectors are being added before (labeled as “Nonsketched Basis”) and after (“Sketched Basis”) whitening or stabilization.

A few observations emerge from these results:

1. Whitening temporarily reduces  $\kappa(V)$  to approximately 1 when  $\kappa(V) \geq \epsilon_{\text{mach}}^{-1}$ . However, orthogonality is quickly lost in subsequent iterations, causing  $\kappa(V)$  to once again exceed  $\epsilon_{\text{mach}}^{-1}$ . Tests were performed to reduce the whitening tolerance (i.e., called whitening once  $\kappa(V) > \epsilon_{\text{mach}}^{-1} \cdot 1E-2$ ); however, this had little effect on the convergence shown in Figure 4.1.



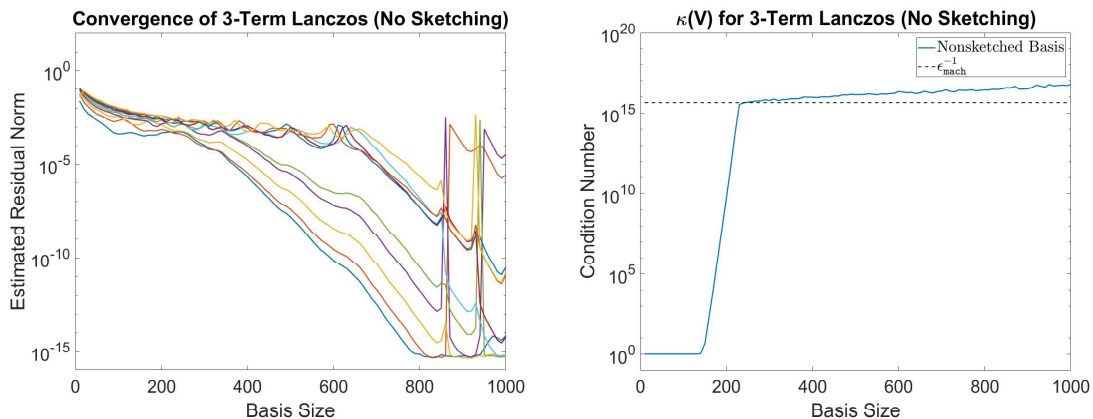


**Figure 4.1:** Comparisons of convergence between 3-term Lanczos with sRR utilizing whitening and 3-term Lanczos with sRR utilizing stabilization to manage the condition number of the basis.

2. Despite whitening reducing the condition number of the sketched basis  $\kappa(C)$  more than stabilization before calling sRR, accurate eigenpairs could still not be extracted from  $C$ , resulting in a stagnation in convergence.

For these reasons, all subsequent sRR experiments will forego the use of whitening and instead perform stabilization when  $\kappa(V) \geq \epsilon_{\text{mach}}^{-1}$ .

To ensure the convergence of 3-term Lanczos with sRR is comparable to that of its non-sketching counterpart, we also conducted MATLAB experiments using 3-term Lanczos with classical RR under the same parameters as those used in Figure 4.1. RR and sRR were performed at every iteration to monitor convergence before the convergence rates were compared.



**Figure 4.2:** The convergence of 3-term Lanczos with classical RR alongside the condition number of the Krylov basis being built.

The results, illustrated in Figure 4.2, show that when  $\kappa(V) < \epsilon_{\text{mach}}^{-1}$ , the convergence rates between the two methods are comparable. However, once  $\kappa(V) \gtrsim \epsilon_{\text{mach}}^{-1}$ , classical RR begins to fail in extracted eigenpair approximations. In the context of Figure 4.2, this occurs around iteration 850 when previously converged eigenpairs become unconverged. Contrasting this, sRR leverages stabilization to manage bases with large condition numbers, allowing for the accurate extraction of eigenpairs even when  $\kappa(V) \gtrsim \epsilon_{\text{mach}}^{-1}$ . This observation reinforces the motivation behind using sRR in applications where many eigenpairs are required.

#### 4.4.1 Thick-Restarted Lanczos with sRR

Thick-restarted Lanczos (TRL) extends the traditional unrestarted Lanczos by periodically setting the basis  $V$  equal to a set of orthogonal Ritz vectors [103]. This approach manages memory and orthogonalization costs by ensuring the basis size never grows excessively large. However, frequent restarting can slow convergence when seeking a large number of eigenpairs, as large basis sizes allow many eigenpairs to converge quickly.

The construction of  $V$  in TRL continues until a user-specified maximum basis size  $d$  is reached. Once this threshold is reached, RR is performed, and the first  $r$  Ritz vectors corresponding to the most significant Ritz pairs form the foundation for the new basis.

Subsequently, Lanczos proceeds with the usual  $V_{d+1}$  vector, which corresponds to the direction of the Ritz pair residuals. However, once the orthogonality of  $V$  is lost, the residuals of the Ritz vectors used for restarting are no longer orthogonal to the actual eigenvectors of  $A$ , causing a stagnation in convergence. When using TRL alongside sRR, the process remains largely the same, except for explicitly orthogonalizing the set of Ritz vectors used for restarting.

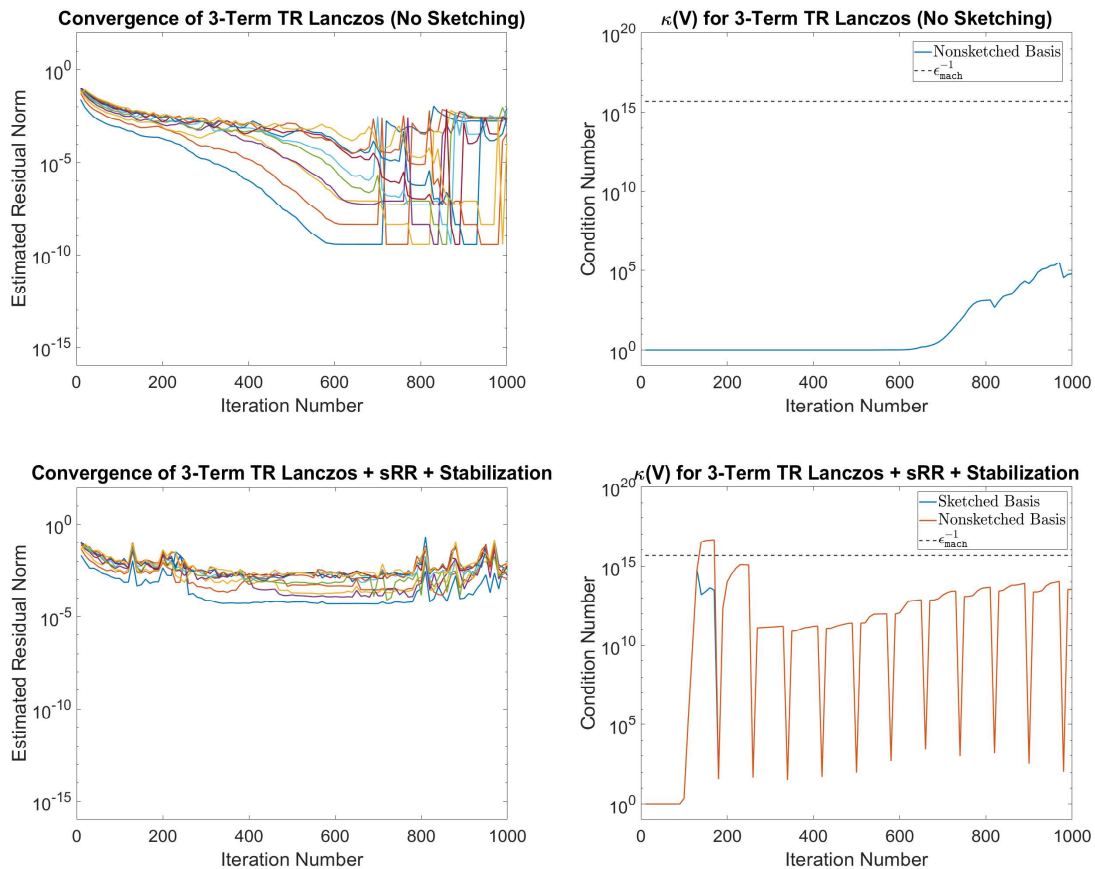
MATLAB experiments were conducted to evaluate the convergence behavior of the Ritz pairs using TRL with sRR compared to classical RR. The parameter setup of these tests mirrored those in Figure 4.1, with a few exceptions. The maximum basis size was set to 100, the restarted basis size to 20, and the maximum number of iterations to 1,000. Restarting was performed once the number of columns in  $V$  reached the maximum basis size of 100. Results from one of these experiments are shown in Figure 4.3, although similar behavior was observed across multiple runs.

The left plots in Figure 4.3 depict the convergence behavior of the ten largest-magnitude Ritz pairs using the 3-term TRL method, while the left plots illustrate the changes in the basis' condition number over time. This experiment highlights that without orthogonalization, convergence stagnates once the basis loses orthogonality for both RR and sRR, just at differing levels.

When sketching, this stagnation occurs earlier as loss of orthogonality in  $V$  causes the residuals from the Ritz pairs extracted from sRR to no longer correspond with the descent direction of the restarted space. Additional experiments validated these results and yielded similar behavior. As a result, we concluded that if sRR is to be utilized with the 3-term Lanczos method, Lanczos must be run unrestarted.

## 4.5 Experiments with Lanczos

In Section 4.4.1, we highlighted that sRR can not be used in conjunction with TRL due to a stagnation in convergence. However, considering the potential for 3-term Lanczos with



**Figure 4.3:** Comparisons of convergence between 3-term TRL with classical RR and sRR for a maximum basis size of 100.

sRR to work unrestarted, even in instances where 3-term Lanczos with classical RR fails (as shown in Figures 4.1 and 4.2), we sought to evaluate whether unrestarted 3-term Lanczos with sRR would perform better in terms of run time compared to full-orthogonalization Lanczos (FO Lanczos) used with RR. To do this, we utilized the C99 software library PRIMME, detailed in Section 2.4.2.

We have integrated the unrestarted 3-term and FO Lanczos methods and sRR into PRIMME. In the previous section, we observed that at a certain point after  $\kappa(V) \gtrsim \epsilon_{\text{mach}}^{-1}$ , whitening fails to extract accurate eigenpair approximations. Furthermore, TRL cannot be used alongside sRR without orthogonalization. Therefore, all evaluations in our study utilize stabilization when the condition number of the basis exceeds  $\epsilon_{\text{mach}}^{-1}$ . Otherwise, we

use the QR decomposition of the sketched basis to solve the generalized eigenproblem depicted in Equation 4.6. Restarting is not utilized.

#### 4.5.1 Complexity Analysis of Unrestarted Lanczos

Before conducting numerical experiments comparing unrestarted Lanczos with and without sketching for various input values, we first performed a complexity analysis to estimate the potential performance improvement we can expect when running 3-term Lanczos with sRR compared to FO Lanczos with RR and understand the computational costs involved. This was done by considering the number of floating-point operations (FLOPs) the PRIMME software will use when running the Lanczos processes. Some minor computations were excluded as their computational costs were negligible.

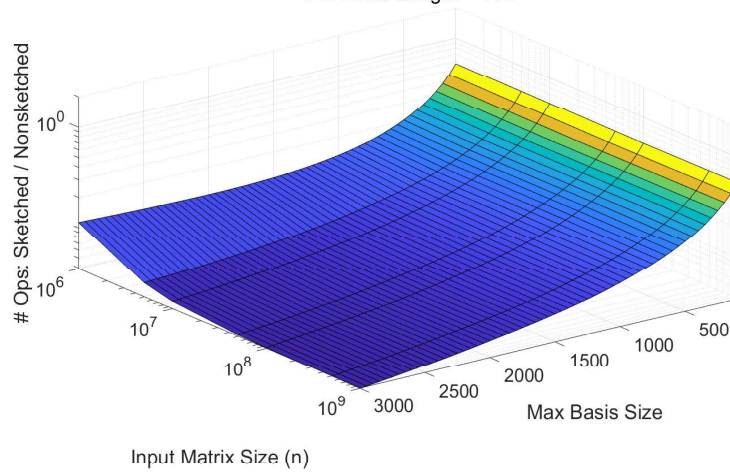
Tables 4.1 and 4.2 present the results of this complexity analysis for FO Lanczos with RR and 3-term Lanczos with sRR, respectively. In these tables,  $n$  denotes the size of the input matrix  $A \in \mathbb{F}^{n \times n}$ ,  $d$  denotes the maximum basis size of  $V \in \mathbb{F}^{n \times d}$ , and the number of eigenpairs sought is denoted by  $e$ . Each table lists the PRIMME operations in the left column with their corresponding FLOP count in the right column. These complexity analyses consider a complete Lanczos cycle as the number of FLOPs needed to build a basis of size  $d$ , perform the eigenpair extraction using RR or sRR, and compute the residuals of the Ritz pairs.

To compare the computational efficiencies between 3-term Lanczos with sRR and FO Lanczos with RR, we compute the ratios of their total cycle operation counts from Tables 4.1 and 4.2. These ratios are illustrated in Figure 4.4.

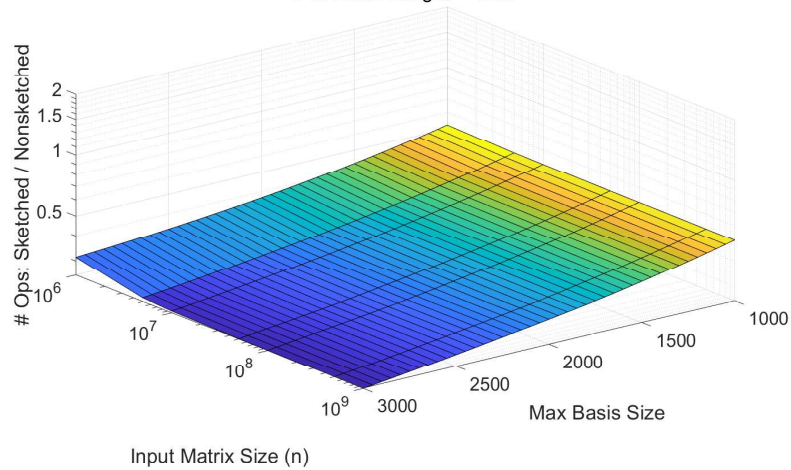
Based on the results in Figure 4.4, theoretically, as the size of the input matrix  $A \in \mathbb{F}^{n \times n}$  increases, unrestarted Lanczos with sketching requires fewer FLOPs than FO Lanczos without sketching. Additionally, sRR should necessitate fewer FLOPs when the maximum basis size is large.

While the results of this analysis favor sRR, there are two considerations to make. Firstly, although one cycle of unrestarted 3-term Lanczos with sRR theoretically requires

**# Ops Ratio for 3-Term Lanczos With Sketching vs FO Lanczos Without Sketching**  
 # of Evals Sought = 100



**# Ops Ratio for 3-Term Lanczos With Sketching vs FO Lanczos Without Sketching**  
 # of Evals Sought = 1000



**Figure 4.4:** The ratio of operation counts between unrestarted Lanczos with and without sketching. The top plot depicts when the number of eigenpairs sought is 100, and the bottom plot shows when the number of sought eigenpairs is 1,000.

Unrestarted Lanczos With RR in PRIMME (No Sketching)	
Operation	Operation Count
<b>At Iteration <math>j</math></b>	
Update the $H$ matrix $H_{j,j-1} = \ V_j\ _2$	$2n$
Normalize the new basis vector $V_j = V_j/H_{j,j-1}$	$n$
Orthogonalize $V_j$ against all preceding vectors	$4nj$
$V_{j+1} = AV_j$	<b>MatVec</b>
Update the $H$ matrix $H_{j,j} = V_{j+1}^H V_j$	$2n$
3-Term Recurrence $V_{j+1} = V_{j+1} - V_{j-1}H_{j,j-1} - V_j H_{j,j}$	$4n$
<b>Rayleigh-Ritz</b>	
Solve the eigenproblem $H\hat{x} = \hat{\lambda}\hat{x}$	$\frac{4}{3}d^3$
$\text{evecs}_i = V\hat{x}_i$ for $i = 1, 2, \dots, e$	$2nde$
Approximate residual norms (Equation 2.26)	$e$
<b>Total Operations:</b> $\sum_{j=1}^d (9n + 4nj + \text{MatVec}) + \frac{4}{3}d^3 + 2nde + e$ $= O(2nd^2 + 2nde + 11nd + \frac{4}{3}d^3 + d \cdot \text{MatVec})$	

**Table 4.1:** Operation count for each step in the unrestarted Lanczos algorithm in PRIMME when used without sketching

fewer FLOPs than unrestarted FO Lanczos with RR, the convergence rates of the two methods may differ, as later experiments indicate that 3-term Lanczos with sRR converges slower on average. Secondly, the cost of a FLOP can vary depending on the specific operation being performed, meaning these FLOP ratios may only partially reflect the actual performance in terms of timings.

#### 4.5.2 PRIMME Experiments

Tests were conducted on BG2 of the Computer Science cluster at William & Mary [2]. Each compute node is an 8-core Intel Xeon CPU E5-4627 v2 with a clock speed of 3.3GHz, and each test instance utilized a single compute node with 8 MPI processes and no multi-threading.

To assess the efficiency of employing sRR with unrestarted Lanczos compared to RR, we consider the following input parameters:

- Maximum basis size = 1,000
- # of Largest Magnitude Eigenpairs Sought = [1; 50; 100; 250; 500]

<b>Unrestarted Lanczos With sRR in PRIMME (With Sketching)</b>	
Operation	Operation Count
<b>At Iteration <math>j</math></b>	
Update the $H$ matrix $H_{j,j-1} = \ V_j\ _2$	$2n$
Normalize the new basis vector $V_j = V_j/H_{j,j-1}$	$n$
Update the sketched basis $C_j = SV_j$	$4n\log(d)$
Update the $Q$ and $R$ factors of $C$ such that $C = QR$	$4dj$
$V_{j+1} = AV_j$	<b>MatVec</b>
Update the $H$ matrix $H_{j,j} = V_{j+1}^H V_j$	$2n$
3-Term Recurrence $V_{j+1} = V_{j+1} - V_{j-1}H_{j,j-1} - V_j H_{j,j}$	$4n$
<b>Sketched Rayleigh-Ritz (with Stabilization)</b>	
Estimate $\kappa(R)$ using Lapack's <code>trcon</code>	$3d^3$
Compute $D = CH$	$8d^3$
Find the tSVD of $C$ [98]	$19d^3$
Compute left-hand-side operator	$16d^3$
Solve the generalized eigenvalue problem	$\frac{4}{3}d^3$
$\hat{x}_i = V_{SVD}\hat{y}_i$ for $i = 1, 2, \dots, e$	$d^2e$
$\mathbf{evecs}_i = V\hat{x}_i$ for $i = 1, 2, \dots, e$	$2nde$
Normalize $\mathbf{evecs}$	$3ne$
Approximate residual norms (Equation 2.26)	$e$
<b>Total Operations:</b> $\sum_{j=1}^d (9n + 4n\log(d) + 4dj + \text{MatVec}) + \frac{142}{3}d^3 + d^2e + 2nde + 3ne + e$ $= O(4nd\log d + 13nd + 2nde + 3ne + \frac{142}{3}d^3 + 2d^2(d+1) + d^2e)$	

**Table 4.2:** Operation count for each step in the unrestarted Lanczos algorithm in PRIMME when used with sketching

- Convergence tolerance = 1E-6
- $\epsilon_{\text{mach}} = 2.2204\text{E-}16$
- Subspace embedding dimension  $s = 4 \times$  Maximum basis size = 4,000

Table 4.3 presents the four matrices used for testing. `f12010` is a matrix obtained from the SuiteSparse Matrix Collection [56], representing a weighted, undirected graph derived from the 2010 census redistricting in Florida, USA. `QuadraticDiag` is a diagonal matrix constructed in MATLAB, where the largest 1,000 elements are squares of the first 1,000 positive integers, with all other elements having a value around  $10^{-3}$ . `Laplacian4D` is a Laplacian on a 4D regular grid with dimensions [32, 32, 32, 32] and no boundary conditions. Finally, `CurlCurl_4` is a matrix from SuiteSparse representing the curl-curl



Matrices Used in PRIMME For Testing			
Matrix Name	# Rows/Columns	Description	Obtained
f12010	484,481	An undirected weighted graph	SuiteSparse
QuadraticDiag	1,000,000	A diagonal matrix with a quadratic decay	Constructed
Laplacian4D	1,048,576	A 4D Laplacian with dimensions $32^4$	Constructed
Cur1Cur1_4	2,380,515	A model reduction problem	SuiteSparse

**Table 4.3:** The list of matrices used for testing the unrestarted Lanczos and Generalized Davidson methods with and without sketching in PRIMME. “Constructed” refers to matrices built ourselves using Matlab.

operator of a second-order Maxwell’s equations with perfect electric conductor boundary conditions.

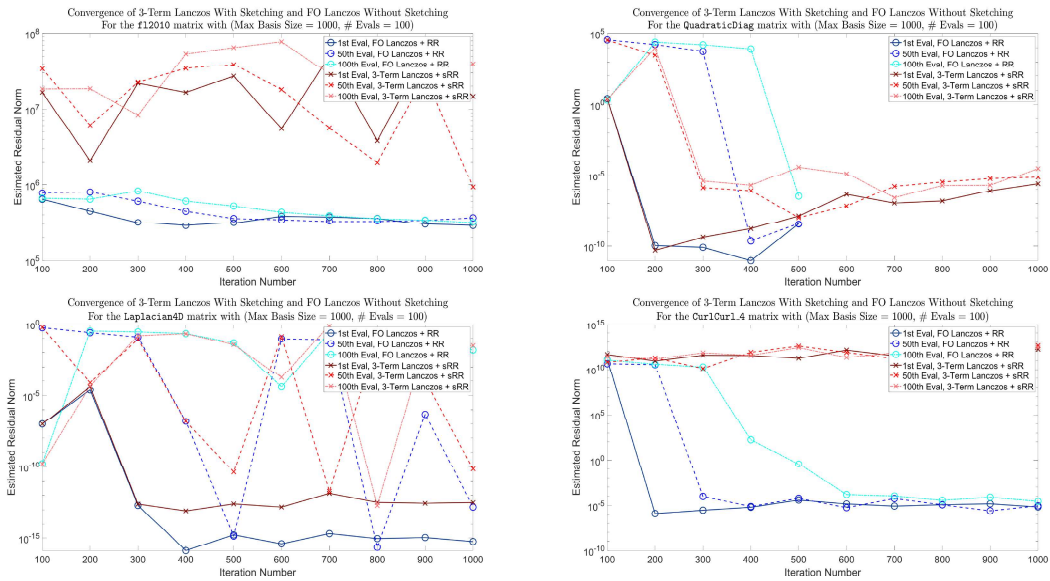
### 4.5.3 Convergence of the Ritz Pairs

We examine first the convergence of the Ritz pairs. We ran both FO Lanczos without sketching and 3-term Lanczos with sketching. Every 100 iterations, the residual norms of the extracted Ritz pairs were approximated as shown in Equation 2.26. If all eigenpairs sought had been marked as converged and the basis size was less than 1,000, Equation 2.8 was used to explicitly compute the residual norms for verification. Given that all Ritz pairs remained converged, the values would be returned to the user. Otherwise, the Lanczos method would continue.

Figure 4.5 illustrates the convergence of the 1st, 50th, and 100th Ritz pairs using the 3-term Lanczos method with sRR and the FO Lanczos with RR.

A couple of observations can be made from these results:

- FO Lanczos converges faster than the traditional 3-term Lanczos as the condition number of the basis is always one, allowing for new information to continuously enter the basis and a more accurate extraction of the Ritz pairs.
- For matrices `QuadraticDiag` and `Laplacian4D`, 3-term Lanczos with sRR shows comparable convergence to FO Lanczos without sketching. For `QuadraticDiag`, the easy-to-compute spectrum helps both methods converge relatively early. However, sRR cannot reach as good accuracy as FO Lanczos and begins stagnating. For `Laplacian4D`, using stabilization in sRR keeps the accuracy for the largest eigenpair



**Figure 4.5:** The convergence of the 1st, 50th, and 100th Ritz values using the 3-term Lanczos method with sRR and the FO Lanczos method with RR when run of the matrices listed in Table 4.3.

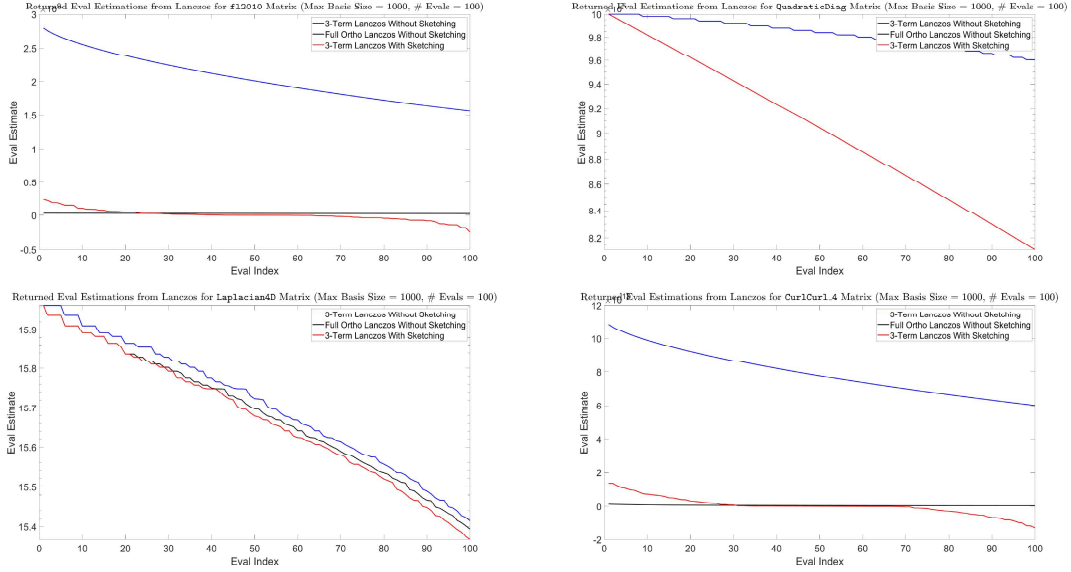
within two orders of magnitude from FO Lanczos. The rest of the eigenpairs are tightly clustered, so convergence is slow, implying less orthogonality loss. Therefore, we see no apparent difference between the two methods.

- For matrices `f12010` and `CurlCurl_4`, loss of orthogonality settles early, so sRR cannot extend the basis with useful directions and stagnates.

These observations highlight that sRR should extract relatively accurate eigenpairs, even in the case of severe orthogonality loss in  $V$ . The problem, however, is that in such cases, Lanczos tends to repeat directions and slow down or stagnate.

To verify and gather a better understanding of the convergence behavior seen in Figure 4.5, we also plot the 100 largest magnitude eigenvalues returned to the user from PRIMME. The values are returned once all 100 have been marked as converged and verified, or the maximum basis size of 1,000 was reached. The results of this can be seen in Figure 4.6.

According to Figure 4.6, returned eigenpairs extracted using sRR from the basis constructed by 3-term Lanczos matched closely with those returned from FO Lanczos with RR for the `QuadraticDiag` and `Laplacian4D` matrices. However, for matrices such as



**Figure 4.6:** The largest 100 eigenpairs returned to the user after all sought eigenpairs are marked as converged or the maximum basis size of 1,000 was reached. Results are shown for 3-term Lanczos with sketching and FO Lanczos with and without sketching for each matrix listed in Table 4.3

f12010 and CurlCurl1\_4, sRR struggled to extract approximate eigenpairs, most likely because the basis  $V$  became too ill-conditioned to satisfy the conditions of the theory that  $\kappa(V) \lesssim \epsilon_{\text{mach}}^{-1}$ .

To increase the accuracy of 3-term Lanczos with sketching, the following strategies should be considered:

1. **Adjust Stabilization Criteria:** Stabilization takes the tSVD of the sketched basis, disregarding all singular triplets following the condition shown in Equation 4.14 before solving the generalized eigenproblem using the remaining triplets. We may be able to increase the accuracy of our Ritz pairs by modifying this condition to discard more singular triplets, e.g.,

$$\frac{\sigma_{\max}}{\sigma_i} \lesssim \epsilon_{\text{mach}}^{-1} \cdot 10^3 \quad \text{for } i = 1, 2, \dots, d. \quad (4.16)$$

Increasing the truncation would likely result in the more accurate extraction of Ritz pairs, but it does not affect the expansion of  $V$  and, thus, convergence.

2. **Increased Orthogonalization:** We can ensure better conditioning of  $V$  using methods such as selective or partial reorthogonalization, resulting in better expansion of  $V$  and the extraction of more accurate Ritz pairs. However, this goes against the purpose of using sRR to minimize orthogonalization.

#### 4.5.4 Run Times

We ran the FO and 3-term Lanczos methods with RR and sRR. However, instead of computing the Ritz pairs and checking their residual norms every 100 iterations, we built the Lanczos basis to a basis size of 1,000 before calling RR/sRR. Each test ran on 8 MPI processes, and the reported times for each of the following categories come from process 0:

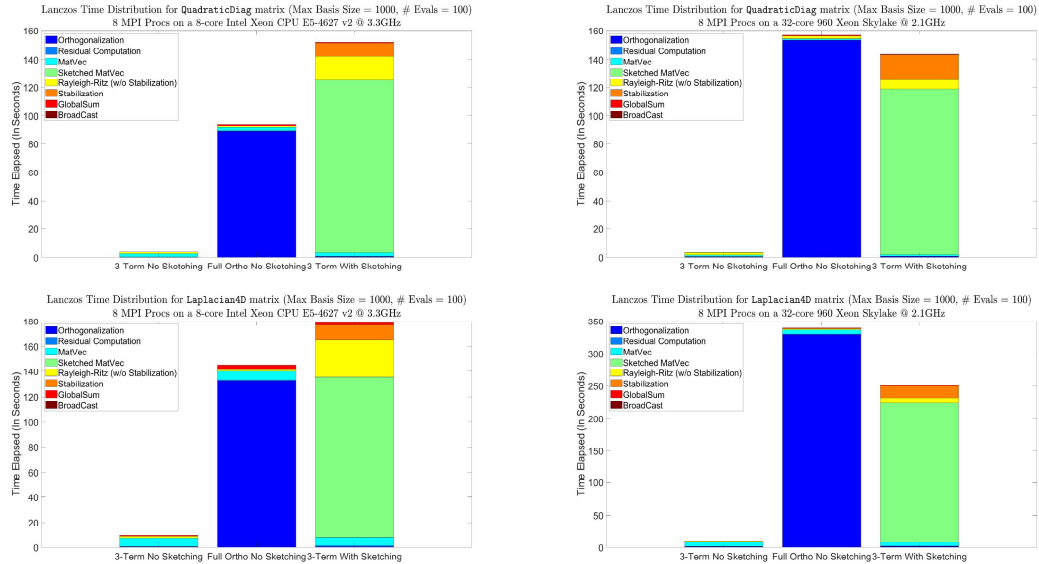
1. Orthogonalization
2. Residual computation
3. MatVec
4. Sparse MatVec (If sketching)
5. RR/sRR
6. Stabilization (If sketching)
7. MPI GlobalSum
8. MPI BroadCast

Some of these categories overlap; for example, the residual computations call MPI GlobalSum.

We also compare timings from two different machines:

- **BG2 Cluster:** 8-core Intel Xeon CPU E5-4627 v2 with a clock speed of 3.3GHz run on 8 MPI processes [2].
- **Femto Cluster:** 32-core 960 Xeon Skylake with a clock speed of 2.1GHz run on 8 MPI processes [1].

We only report timings for two of our four test matrices, `QuadraticDiag` and `Laplacian4D`, as the conclusions are general across all matrices tested.



**Figure 4.7:** The breakdown in runtime for 3-term Lanczos and FO Lanczos run with and without sketching over 1,000 iterations with timings achieved from the 8-core Intel Xeon CPU E5-4627 v2 on the left and 32-core 960 Xeon Skylake on the right.

Figure 4.7 illustrates the breakdown of the time spent during the execution of Lanczos run for 1,000 iterations and extracting the 100 Ritz corresponding to the eigenvalues of largest magnitude. Results are shown only for 100 eigenvalues, as the only noticeable difference in the timing breakdowns between this and 1,000 eigenvalues was the computation of the residual norms. Moreover, the timings for the residual norms were small, so they did not change the overall conclusions. The times are reported over full runs of the Lanczos code, as normalizing the times over the number of performed iterations would not change their comparative ratios.

The time to perform orthogonalization varied significantly between machines, even with identical PRIMME configuration settings and input parameters. While further investigation is needed to understand this behavior fully, we think this is due to differences in architecture and versions of available software libraries. Clearly, if orthogonalization can not be performed efficiently on a given machine, 3-term Lanczos with sRR is more likely to outperform FO Lanczos with RR in terms of run time.

As Figure 4.4 shows, FLOP modeling predicts improved speedups with sRR over RR for problems with larger matrix sizes or basis sizes. However, in practice, the computational bottleneck when sketching comes from the sparse MatVec. Currently, our PRIMME sketching code builds the sketched basis by performing a sparse MatVec on one vector at every iteration. Better performance may be achieved if we apply the sketching matrix just before calling the sRR algorithm at the end so the sparse MatVec is applied to all vectors simultaneously. Using a different sketching matrix, e.g., using SFRT instead of Sparse Maps, may also achieve better performance results, though further investigation is needed to confirm this.

## 4.6 Generalized Davidson with Sketched Rayleigh-Ritz

When seeking many eigenpairs, a large number of iterations are typically needed, and without restarting, the size of the basis becomes impractical in terms of memory and cost of orthogonalization, especially when the size of  $A \in \mathbb{F}^{n \times n}$  is large. TRL did not work because it expanded the basis with the next Lanczos vector, which, in the presence of sketched Ritz pairs, stopped being the gradient direction for any of them. This motivates the transition to Generalized Davidson.

Generalized Davidson (GD), as detailed in Algorithm 3, is an extension of the Lanczos algorithm that expands the basis  $V$  with the residual of the first unconverged Ritz pair (or pairs, in the case of block methods), a technique referred to as “targeting” [69]. In theory, this is equivalent to the Lanczos vector, except the explicitly computed residual

is the gradient of the Rayleigh Quotient, which allows for preconditioning. Therefore, the promise of much faster convergence justifies the cost of explicit orthogonalization and RR at every step. Even without preconditioning, GD also offers a more robust approach for a broader range of problems. Since the residual, or gradient direction, is with respect to the actual Ritz vectors in the basis, unlike TRL, we expect GD to be able to expand the sRR restarted basis effectively. Moreover, the residuals are orthogonal to  $V$  at a much higher numerical accuracy than the 3-term Lanczos vectors, implying that more accurate sketched eigenpair approximations can be achieved and the potential of avoiding orthogonalization. The GD method integrated with sRR is detailed in Algorithm 9.

**Algorithm 9:** The Generalized Davidson Algorithm With sRR**Input:** $A \in \mathbb{F}^{n \times n}$  = A square, Hermitian matrix $Y \in \mathbb{F}^{n \times e}$  = Matrix of initial orthonormal column(s) $r$  = The size of the restarted basis $e$  = The number of eigenpairs sought after

tol = Convergence tolerance

**Output:** $X \in \mathbb{F}^{n \times d}$  = The estimated eigenvectors of  $A$  $\Lambda \in \mathbb{F}^d$  = The estimated eigenvalues of  $A$ resNorms  $\in \mathbb{F}^d$  = The residual norms for the eigenpair estimates of  $A$ Sketched\_Generalized\_Davidson( $A, Y, r, e, tol$ )

```

1  $V(:, 1:e) = Y;$     $W(:, 1:e) = AY;$ 
2  $S = \text{Sparse\_Maps}();$                                      # Build sketching matrix
3 for  $m = 2, 3, \dots$  do
4    $W(:, m) = AV(:, m);$ 
5    $[X, \Lambda, \sim] = \text{Sketched\_RR}(S, V(:, 1:m), W(:, 1:m));$ 
6   residuals =  $WX(:, i) - VX(:, i)\Lambda(i)$    for  $i = 1, \dots, m;$ 
7   resNorms( $i$ ) =  $\| \text{residuals}(:, i) \|$    for  $i = 1, \dots, m;$ 
8   target = find(resNorms > tol, "first", 1); # Target 1st unconverged vector
9   if target >  $e$  then
10    return  $[X, \Lambda, \text{resNorms}]$ 
11   if  $m > d$  then
12     $V(:, 1:r) = X(:, 1:r);$     $W(:, 1:r) = AV(:, 1:r);$            # Restart
13     $m = r;$ 
14   # Precondition new vector before adding it to the basis
    $V(:, m+1) = \text{Precondition}(\text{residuals}(:, \text{target}) / \text{resNorms}(\text{target}));$ 

```



Unlike sketching with Lanczos, where sRR is only performed once the basis has been constructed to extract eigenpair approximations, sketching with GD solves the sRR problem instead of RR every iteration. The Ritz pairs from sRR are then used to expand the basis. Additionally, we do not orthogonalize the residual vector against all previous vectors when expanding the basis; we only normalize the new vector.

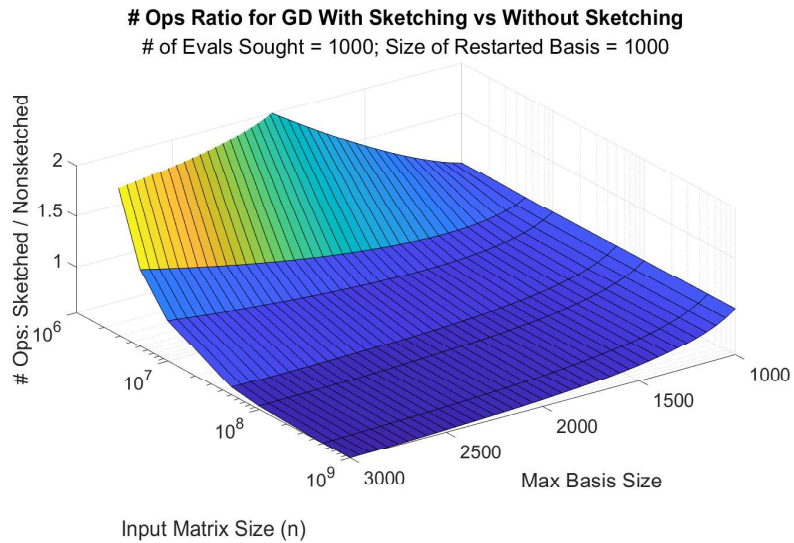
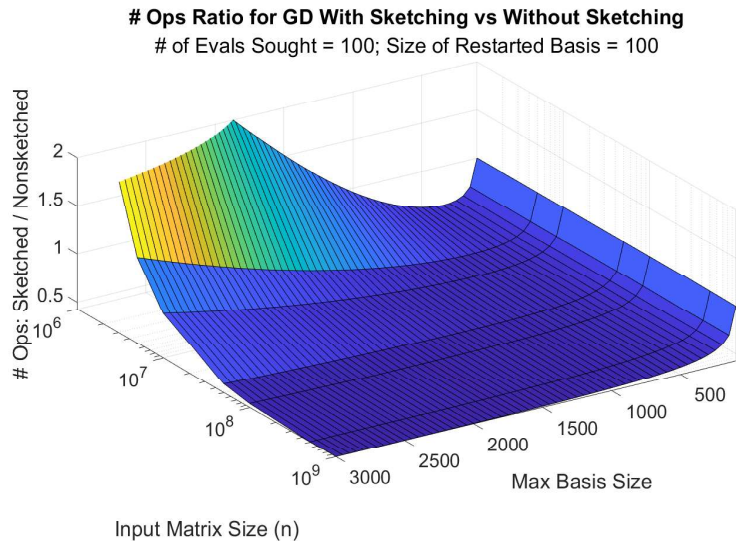
## 4.7 Experiments with Generalized Davidson

We have integrated the ability to call sRR into the pre-existing GD algorithm in PRIMME. Experiments were then run to assess the efficiency of utilizing sRR over RR. GD with RR uses full orthogonalization, while GD with sRR uses no orthogonalization of the long, length- $n$  vectors.

### 4.7.1 Complexity Analysis of Generalized Davidson

As with Lanczos, we first performed a complexity analysis to determine how much performance improvement we can expect. The analysis counted the approximate number of FLOPs each operation should take. Minor computations were disregarded as they had minimal effect on the analysis. Four variables are considered:  $n$ , which is the dimension of the input matrix  $A \in \mathbb{F}^{n \times n}$ , the maximum basis size  $d$ , the number of sought eigenpair estimates  $e$ , and the size of the restarted basis  $r$ . Tables 4.4 and 4.5 show the analysis results for GD without and with sketching, respectively.

When comparing the complexities of the two methods, we evaluate the number of FLOPs performed over a “cycle”, which begins immediately after the basis is restarted and completes after the maximum basis size is reached and restarting is performed. In our benchmarking experiments, we always set  $e = r$  to minimize the number of varying input parameters. We do the same in Figure 4.8, which shows the ratio between the number of FLOPs needed to perform a cycle of GD with sketching over its non-sketching counterpart.



**Figure 4.8:** The ratio of operation counts between Generalized Davidson with and without sketching. The top plot depicts when the number of sought eigenpairs is 100, and the bottom plot shows when the number of sought eigenpairs is 1,000. The restarted basis size equals the number of sought eigenpairs.

Generalized Davidson With RR in PRIMME (No Sketching)	
Operation	Operation Count
<b>At Iteration <math>j</math></b>	
$W_j = AV_j$	MatVec
Update $H = V^H W$	$2nj$
Solve the eigenproblem $H\hat{x} = \hat{\lambda}\hat{x}$	$\frac{4}{3}j^3$
Compute residual and its norm	$4nj + 2n$
Orthogonalize $V_{j+1}$ against all proceeded vectors	$4nj$
<b>At Restart</b>	
$V, W \times = \hat{X}$	$4ndr$
Solve the eigenproblem $V^H AV\hat{x} = \hat{\lambda}\hat{x}$	$\frac{4}{3}r^3$
Compute residual and its norm	$4nr + 2n$
Orthogonalize $V_{r+1}$ against all proceeded vectors	$4nr$
<b>Total Operations:</b> $\sum_{j=r+1}^d (10nj + 2n + \frac{4}{3}j^3 + \text{MatVec})$ $+ 4ndr + 8nr + 2n + \frac{4}{3}r^3$ $= O(n(5d^2 + 4dr + 7d + 6r + 2 - 5r^2) + \frac{d^4 + 2d^3 - r^4 + 2r^3}{3})$ $+ (d - r) \cdot \text{MatVec}$	

**Table 4.4:** Operation count for each step in the Generalized Davidson algorithm in PRIMME when used without sketching

The results of Figure 4.8 show promise in terms of runtime speedup. As the matrix size increases, the ratio of FLOPs between sketched and non-sketching GD seemingly converges to 0.5. However, as noted in Section 4.5, not all FLOPs take the same amount of time to perform; thus, actual benchmarking is needed.

#### 4.7.2 PRIMME Experiments

The following input parameters were considered when comparing the effectiveness of GD with sRR and GD with RR:

- Maximum basis size = [100; 200; 250; 500; 1,200]
- # of Largest Magnitude Eigenpairs Sought = [1; 50; 100; 250; 500]
- Restart size = # Eigenpairs
- Convergence tolerance = 1E-6

Generalized Davidson With sRR in PRIMME (With Sketching)	
Operation	Operation Count
<b>At Iteration <math>j</math></b>	
$W_j = AV_j$	MatVec
Update $C = SV_j$ and $D = SW_j$	$8n \log d$
Update Q and R factors of $C$	$4dj$
Compute left-hand-side operator	$8dj^2$
Solve the generalized eigenvalue problem	$\frac{4}{3}j^3$
Normalize $\hat{X}$	$3je$
Compute residual and its norm	$4nj + 2n$
<b>At Restart</b>	
Orthogonalize $\hat{X}$	$2dr^2$
$V, W, C, D \times = \hat{X}$	$4ndr + 16d^2r$
Compute $\ V_i\ _2^{-1}$ for $i = 1, 2, \dots, r$	$2nr + r$
Normalize the columns of $V, W, C, D$ with $\ V_i\ _2^{-1}$	$2nr + 8dr$
Compute left-hand-side operator	$8dr^2$
Solve the generalized eigenvalue problem	$\frac{4}{3}r^3$
Normalize $\hat{X}$	$3re$
Compute residual and its norm	$4nr + 2n$
Update Q and R factors of $C$	$8dr^2$
<b>Total Operations:</b> $\sum_{r+1}^d (4nj + 8n \log d + 2n + \frac{4}{3}j^3 + 8dj^2 + 4dj + 3je + \text{MatVec})$ $+ 4ndr + 8nr + 2n + 16d^2r + 8dr + \frac{4}{3}r^3 + 18dr^2 + 3re + r$ $= O(2n(2d^2 + 7d - 2r^2 - 7r + 2dr + 4 \log d \cdot (d - r))$ $+ \frac{9d^4 - r^4 + 14d^3 - 10dr^3 + 4r^3}{3} + (d - r) \cdot \text{MatVec})$	

**Table 4.5:** Operation count for each step in the Generalized Davidson algorithm in PRIMME when used with sketching

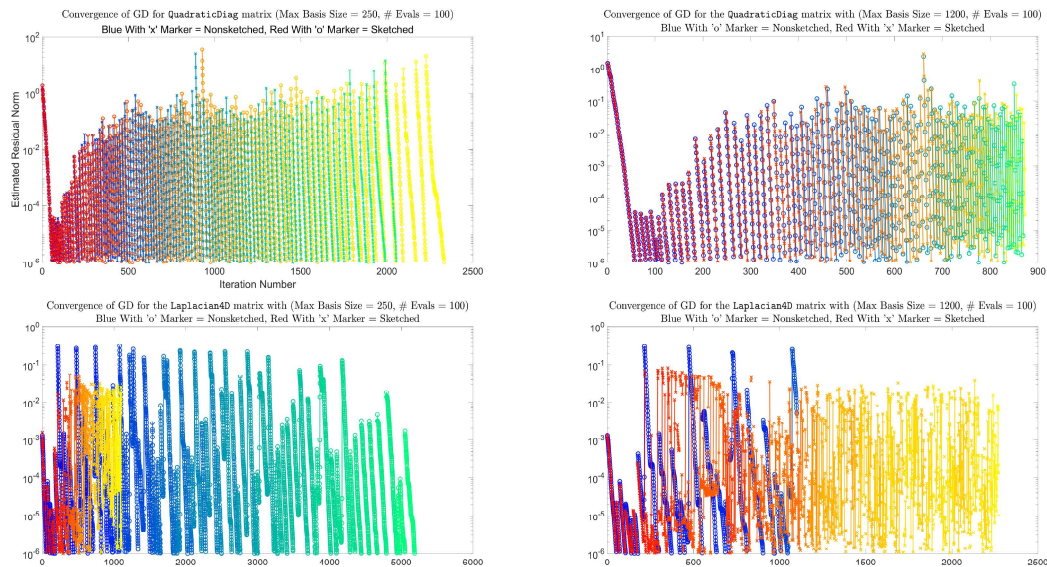
- $\epsilon_{\text{mach}} = 2.2204\text{E-}16$

Tests were performed when the number of sought eigenpairs was less than the maximum basis size and run using BG2 of the Computer Science cluster at William & Mary, where each compute node is an 8-core Intel Xeon CPU E5-4627 v2 with a clock speed of 3.3GHz [2]. As in Section 4.5, each test instance utilized a single compute node with 8 MPI processes and no multithreading. The matrices tested are listed in Table 4.3, though, for the sake of brevity and to avoid redundancy, we only present results for the matrices QuadraticDiag and Laplacian4D. The matrices f12010 and CurlCurl\_4 yielded similar timings; however, neither converged to a single eigenpair.

### 4.7.3 Convergence of the Ritz Pairs

We report results from runs with the maximum basis sizes 250 and 1,200, in both cases seeking the 100 largest magnitude eigenpairs. These two cases are generally representative and can be used to summarize our findings. Because of GD’s targeting strategy, only the residual norm of the targeted pair is plotted at every iteration. The gradient in colors depicts the targeted eigenvalue index.

Figure 4.9 shows the convergence of the Ritz pairs corresponding to the 100 largest magnitude Ritz values for RR and sRR given a maximum basis size of 250 in the left column and a maximum basis size of 1,200 in the right column. The behavior shown between the two matrices is noticeably different.



**Figure 4.9:** The convergence of the targeted Ritz pair when running GD with a maximum basis size of 250 (left) and 1,200 (right) and seeking 100 Ritz pairs.

The results for the QuadraticDiag test using maximum basis sizes of 250 and 1,200 show that the convergence rates of the Ritz pairs were similar between sRR and RR. This is expected due to the well-separated upper spectrum of the matrix. When the maximum basis size is 1,200, the run does not restart, and both methods converge identically. For the 250-basis run, we observed that  $\kappa(V)$  stays small, at least for the first few thousand

iterations. This means that sRR does not need stabilization, which is reflected in our complexity analysis in Table 4.5. However, even though approximations are accurately extracted, convergence shows a slight slowdown after many restarts.

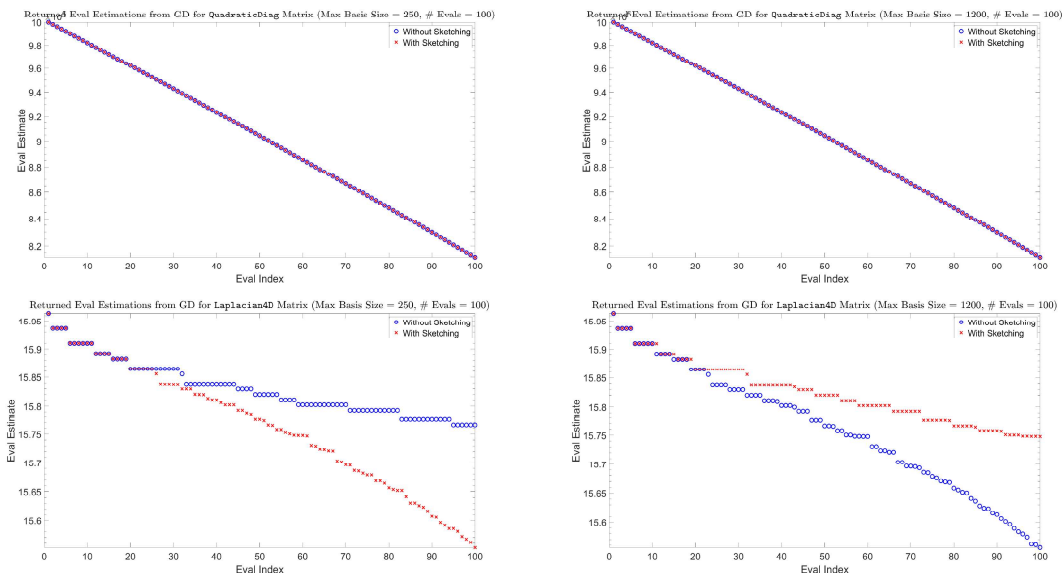
For the `Laplacian4D` matrix, the convergence behavior observed between GD with sRR and RR was noticeably different, illustrating different numerical issues with each method. With a basis size of 250, sRR converged almost 5x faster than RR with the same input parameters. However, when running the same problems with a basis size of 1,200, sRR converged 2x slower than GD with RR.

This behavior can be understood better if we examine the accuracy of the returned eigenvalues after all required eigenpairs converge and the algorithm returns. Figure 4.10 plots this information for both matrices `QuadraticDiag` and `Laplacian4D`.

For the easy spectrum of `QuadraticDiag`, sRR and RR find the same eigenvalues at the required accuracy. For `Laplacian4D`, however, the spectrum has clusters of high multiplicity. It is known that in exact arithmetic, the Lanczos method (or any Krylov method) cannot compute multiple eigenvalues. Floating point arithmetic introduces noise that is slowly amplified towards the missing directions. Interestingly, the method that takes fewer iterations is less susceptible to noise and thus misses most of the multiplicities.

For the 250-vector basis, GD with sRR misses half of the eigenvalues of the 6th cluster, and for more interior values it identifies no more than two eigenvalues per cluster. The slightly better conditioning of  $V$  allows the restarted GD with RR to identify these multiplicities and take the time to fully converge.

For the 1,200-vector basis, the roles are reversed. Maintaining full orthogonality in unrestarted GD with RR makes it very close to Lanczos in exact arithmetic. Since there is no restart to introduce numerical noise, the method fails to find all multiplicities, even starting from the third cluster. This is a situation where doing the right thing does not pay!



**Figure 4.10:** Comparisons of the 100 largest magnitude Ritz pairs returned from PRIMME for Generalized Davidson for a maximum basis size of 250 (right) and 1,200 (left). Only the results from matrices `QuadraticDiag` and `Laplacian4D` are shown.

#### 4.7.4 Timing Comparisons

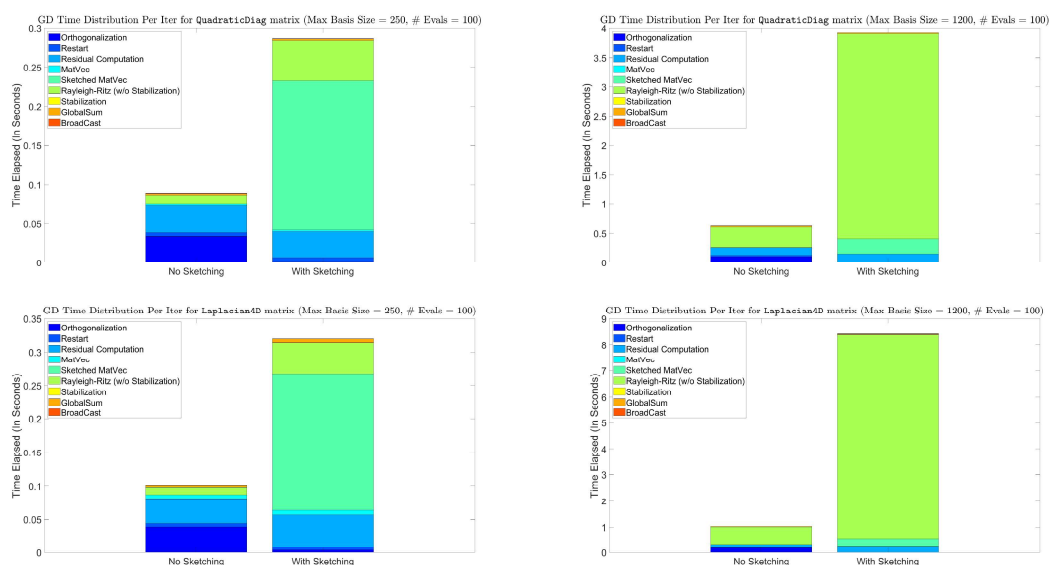
We now turn our attention to comparing the run times of GD with and without the use of sketching. We report the times returned by process 0 using 8 MPI processes on the 8-core Intel Xeon CPU E5-4627 v2 [2] without multithreading. The timed categories are the same as those listed in Section 4.5.4, with the addition of reporting the time the program took to restart the basis:

1. Orthogonalization
2. Restarting the basis
3. Residual computation
4. MatVec
5. Sparse MatVec (If sketching)
6. RR/sRR
7. Stabilization (If sketching)

8. MPI GlobalSum

9. MPI BroadCast

The results of the timing breakdowns are shown in Figure 4.11. In these plots, we average the total time spent in each category by the number of iterations performed as GD with and without sketching, as the two methods took a different number of iterations to converge. The left column shows results when the maximum basis size is 250, whereas the right plots used a maximum basis size of 1,200. The top two figures depict timings from the QuadraticDiag matrix, and the bottom two depict timings from the Laplacian4D matrix.



**Figure 4.11:** The breakdown in average runtime per iteration for GD run with a maximum basis size of 250 (left) and 1,200 (right) when searching for 100 Ritz pairs.

Figure 4.11 shows that the percentage of time spent in different operations highly depends on the maximum basis size. When using RR and the maximum basis size is small, most of the run time is nearly equally divided between orthogonalizing the new basis vectors and computing the residual vector for the basis expansion. It is notable that restarting, which has the same number of FLOPs as these two kernels, takes significantly less time as it is performed with BLAS level 3 GEMM while the others use level 2 and some level 1 BLAS. When the maximum basis size is large, most of the run time for GD



with RR is spent on the RR computation, even though the complexity of RR does not involve the matrix dimension. Clearly, a matrix dimension of 1,000,000 is insufficient for the asymptotic regime where the RR cost should be negligible.

The timing results when using GD with sRR are more interesting. When the basis size is small, the computational bottleneck is clearly the sketched Matvec that gets performed twice each iteration. This is consistent with the 3-term Lanczos with sRR timing results in Figure 4.7. However, when discussing Lanczos with sRR, we stated that the sketched MatVec might be done more efficiently by applying the sketching matrix to multiple vectors simultaneously right before calling sRR. This block operation can not be done with GD, where sRR is computed at every iteration.

When the maximum basis size is large, the computational bottleneck associated with using sRR with GD shifts from sketched MatVec to the sRR computation, as shown in Figure 4.11, and the time per iteration becomes more expensive compared to GD with RR. This behavior is predicted by the complexity analysis. Firstly, given the same matrix size, the cost of sRR increases cubically with the basis size, while the cost of the sketched MatVec increases linearly. Secondly, the complexity of sRR is a constant factor larger than RR, so for matrix sizes where (s)RR dominates, sketching is not beneficial (see also the timings in Figure 4.7).

It is clear from our results above that matrix sizes of 1,000,000 are not sufficiently large to show the computational benefits of sketching, especially with large basis sizes. This is also suggested by the modeling in Figure 4.8. Furthermore, the architectural details of the machine and numerical libraries used are important, as shown in Figure 4.7. For small basis sizes, the challenge is to reduce the time for sketched MatVec, either through better implementation of the sparse embedding or by utilizing a different subspace embedding matrix.

## 4.8 Chapter Summary

This work explores the performance advantages and disadvantages of using sketched Rayleigh-Ritz (sRR) to extract the approximate eigenpairs of a matrix from a basis built using a Krylov-based iterative method (or a Krylov-like method in case of GD). To accomplish this, we implemented sRR and the unrestarted Lanczos Krylov method into the C99 software library, PRIMME, which can be found at the publicly accessible GitHub repository <https://github.com/primme/primme/tree/heather/sketch> [91]. In theory, sRR shows promise in improving runtime by allowing the extraction of Ritz pairs from a non-orthonormal basis. This circumvents the need for consistent reorthogonalization, which can become a bottleneck in traditional iterative methods.

Sections 4.5 and 4.7 show theoretical complexity analyses of both the unrestarted Lanczos and Generalized Davidson (GD) methods, respectively, when used alongside either classical RR or sRR. These analyses were conducted by summing the expected number of FLOPs each method should take in practice. Following this, convergence, accuracy, and timing results are shown for the Lanczos and Generalized Davidson methods using matrices of different eigenspectra densities and sizes.

From these results, several conclusions were made. For the Lanczos method, the complexity analysis showed that the 3-term Lanczos with sRR in PRIMME should require less FLOPS than full-orthogonalization Lanczos when the input matrix and maximum basis sizes are large. However, in our benchmarking tests, we saw that the performance improvement between the two methods in terms of time depended on how efficiently the orthogonalization was performed using full-orthogonalization Lanczos without sketching. Also, while in general full-orthogonalization Lanczos without sketching converges in fewer iterations than 3-term Lanczos with sketching by ensuring the condition number of the basis remains at one, 3-term Lanczos with sketching was able to converge at approximately the same rate in some cases through the use of stabilization.

The results for GD indicated that the removal of full-orthogonalization and the use of sketching did not dramatically raise the basis's condition number, at least for the first few thousand iterations. This means GD with sRR was able to accurately extract eigenpairs from the basis. In terms of runtime, GD with sRR performed significantly slower per iteration than GD with RR, with most of the runtime being dedicated to sparse MatVec when the maximum basis size was small, and the sRR procedure when the maximum basis size was large. However, using an input matrix of larger size, a better implementation of the sparse MatVec, or a different subspace embedding matrix may result in a lower runtime when using GD with sRR. More investigation is needed to verify this.

# Chapter 5

## Conclusion

### 5.1 Summary of Contributions

#### Variance Reduction Methods in LQCD

In the first part of this dissertation, we introduced a novel probing approach for reducing the variance of the Hutchinson estimator when approximating the trace of a permuted matrix inverse. This work is essential for research in LQCD, specifically in the study of disconnected diagrams. Probing for permutations is done by performing a distance- $k$  coloring on the graph of a matrix where every lattice point  $x$  is considered connected to the neighborhoods centered around its corresponding displacements,  $x \pm p$ , where  $p$  is the displacement vector.

Additionally, we provide a lower-bound analysis on the minimum number of colors necessary to perform a distance- $k$  coloring, considering a particular lattice displacement vector  $p$ . We offer insights into how the lattice dimensions may impact the number of colors achieved. Finally, we present experimental results comparing the effectiveness of a given  $(p, k)$ -coloring against classical probing and exclusively using Rademacher vectors in the estimator.

When utilizing probing vectors from  $(p, k)$ -coloring in the Hutchinson estimator compared to solely Rademacher vectors, we achieved up to a 300x speedup in variance (with

$p = 8, k = 9$ ). On average, probing with displacements provided approximately a 100x - 200x speedup. However, when probing with displacements was compared with classical probing to find the trace of a displaced matrix inverse, probing with displacements achieved almost a 100,000x speed up (for  $p = 8, k = 6$ ). The reason for this drastic difference between the speedup with classical compared to solely using Rademacher vectors is explainable. Using classical probing to aid in finding the trace of a displaced matrix inverse will not remove the largest magnitude elements from the inverse when  $p > k$ , resulting in poor variance reduction. However, Rademacher vectors may get “lucky” because they can remove several of these large-magnitude elements that classical probing will not touch. In either case, probing with displacements, in general, is guaranteed to reduce the variance of the Hutchinson estimator more than classical probing or solely using random vectors, with better speedups as the displacements grow larger.

### **The Effectiveness of Krylov Methods Using Sketched Rayleigh-Ritz**

In the second part of this dissertation, we explore using sketched Rayleigh-Ritz (sRR) within the Lanczos and Generalized Davidson Krylov methods to avoid the computational burden of orthogonalizing the basis as eigenpairs begin to converge causing repeated directions to enter the basis. We conduct a comprehensive complexity analysis to predict how much of a performance speedup we can expect when running Lanczos and Generalized Davidson with sRR compared to their nonsketched counterparts for varying input parameters, including the maximum basis size, the size of the restarted basis, the number of approximate eigenpairs sought, and the size of the input matrix.

We then integrated the Lanczos algorithm and sRR into the high-performance parallel PRIMME software library before conducting numerous practical experiments. We report the rates of convergence given varying input parameters. After this, we compare the values of the returned eigenvalues to determine their accuracy and the method’s reliability. Finally, we time various components of the experiment to determine which operations are

the most costly, where most of the run time is spent, and how these timings change based on the problem's parameters.

From our complexity analysis, we found that theoretically, sRR with 3-term Lanczos or Generalized Davidson should require fewer FLOPs per iteration to perform than RR with full-orthogonalization Lanczos and Generalized Davidson, respectively, specifically when the sizes of the input matrix and the maximum basis are large. However, benchmarking these routines identified numerical and performance issues that need to be addressed, such as the type of matrix that performs the random subspace embedding, the application of a sparse MatVec, and the efficiency of the orthogonalization routine. Furthermore, the matrices we are benchmarking range from size 500,000 to 2,000,000, which are too small to give an accurate picture as to when sRR should outperform RR.

## Appendix A

# Proving the Lower Bounds on the Number of Colors Needed For Coloring a Lattice with Displacements

### A.1 Proof for Theorem 1

**Theorem 1.** *Let  $x \in \mathbb{Z}_\infty^d$ . If  $p \geq k$ , then  $\forall y \neq x$  with  $y_1 = x_1$ , it holds  $y \notin N(x, p, k)$ .*

**Proof.** From the definition of  $x^+$  and  $x^-$ , the assumption  $p \geq k$  implies  $|x_1 - x_1^+| = |x_1 - x_1^-| \geq k$ , and  $\sum_{i=2}^d |x_i - x_i^+| = \sum_{i=2}^d |x_i - x_i^-| = 0$ . Let  $y \in \mathbb{Z}_\infty^d$  with  $y \neq x$  and  $y_1 = x_1$ . Since  $|y_1 - x_1^+| \geq k$ ,  $\|y - x^+\|_1 = |x_1 - x_1^+| + \sum_{i=2}^d |x_i - x_i^+| > k$ . The same argument applies for the distance from  $x^-$ . Therefore,  $y \notin N(x, p, k)$ .  $\square$

### A.2 Proof for Theorem 2

**Theorem 2.** *If  $p = k$ , then  $\text{col}(\mathbb{Z}_\infty^d, p, k) = 2k + 1$ .*

**Proof.** From Theorem 1, if  $p = k$ , the coloring problem reduces to coloring in one

dimension. Then the neighborhood definition covers  $4k + 1$  indices in the first dimension, from  $-2k$  to  $+2k$ . The maximal clique size of the  $2k$ -distance graph of these nodes is the 1D unit ball of half the size, which includes indices from  $-k$  to  $+k$ . The size of this clique is exactly  $2k + 1$  nodes. Therefore  $2k + 1$  is also the number of colors needed.  $\square$

### A.3 Proof for Theorem 3

**Theorem 3.** *If  $p > k$ , then  $\text{col}(\mathbb{Z}_\infty^d, p, k) = \lceil \frac{2p}{p-k} \rceil = \lceil \frac{2k}{p-k} \rceil + 2$ .*

**Proof.** As previously noted, if  $p \geq k$ , the coloring problem gets reduced to one dimension. Consider two nodes on  $\mathbb{Z}_\infty^1$  that are exactly  $2p$  links apart, i.e., node  $i$  and  $i + 2p$ . Because  $p > k$ , node  $i + 2p$  does not belong in  $N(i, p, k)$ ; thus, the two nodes can be assigned the same color. Therefore, the problem reduces to coloring a tile of  $2p$  consecutive nodes, which can then be repeated to color the entire  $\mathbb{Z}_\infty^1$  lattice.

Consider  $2p$  consecutive nodes, 0 to  $2p - 1$ . The first  $p - k$  nodes can all be assigned color 1 as they do not belong in each other's neighborhood. The following  $p - k + 1, \dots, 2(p - k)$  nodes have at least one of the nodes in the first group as a neighbor, so they must take a different color, say 2. Similarly, every  $p - k$  group of nodes must take a different color. The last node has neighbors  $i \geq 2p - (p + k) = p - k$ , so it cannot reuse any color, including color 1, because the tile needs to repeat. Then, the total number of colors is the partitioning of  $2p$  nodes in  $p - k$  groups. Note that  $\lceil \frac{2p}{p-k} \rceil = \lceil \frac{2(k+p-k)}{p-k} \rceil = \lceil \frac{2k}{p-k} \rceil + 2$ .  $\square$

### A.4 Proof for Theorem 4

**Theorem 4.** *Assume  $(k + p)$  is even and  $p < k$ . Let,  $\alpha = \lfloor \frac{k+p}{2} \rfloor$ ,  $\beta = \lfloor \frac{k-p}{2} \rfloor$ , and define the set*

$$C(d, \alpha, \beta) = \left\{ x : \|x\|_1 \leq \alpha \text{ and } \sum_{i=2}^d |x_i| \leq \beta \right\}. \quad (\text{A.1})$$



Then  $\forall x, y \in C(d, \alpha, \beta)$ ,  $x \in N(y, p, k)$ , i.e.,  $C(d, \alpha, \beta)$  constitutes a distance- $k$  clique.

**Proof.** First note that  $\alpha - \beta = p$ . Let  $x, y \in C(d, \alpha, \beta)$ . Then the following holds:

$$|x_1| + \sum_{i=2}^d |x_i| \leq \alpha, \quad |y_1| + \sum_{i=2}^d |y_i| \leq \alpha \quad (\text{A.2})$$

$$\sum_{i=2}^d |x_i| \leq \beta, \quad \sum_{i=2}^d |y_i| \leq \beta. \quad (\text{A.3})$$

WLOG assume  $x_1 \leq y_1$ . Then it is sufficient to show that  $x$  belongs in the left neighborhood around  $y^-$ , i.e.,  $x \in N(y^-, 0, k)$  or  $|y_1^- - x_1| + \sum_{i=2}^n |y_i - x_i| \leq k$ .

We distinguish two cases for the distance between  $x_1$  and  $y_1$ .

(a)  $y_1 - x_1 \geq p > 0$ . Using Theorem A.2, we have  $|y_1^- - x_1| + \sum_{i=2}^n |y_i - x_i| = y_1^- - x_1 + \sum_{i=2}^n |y_i - x_i| \leq |y_1| + |x_1| - p + \sum_{i=2}^n |y_i| + \sum_{i=2}^n |x_i| \leq k + p - p = k$ .

(b)  $0 \leq y_1 - x_1 < p$ . Using Equation A.3, we have:  $|y_1^- - x_1| + \sum_{i=2}^n |y_i - x_i| = -y_1 + x_1 + p + \sum_{i=2}^n |y_i - x_i| \leq p + \sum_{i=2}^n |y_i| + \sum_{i=2}^n |x_i| \leq p + k - p = k$ .

□

## A.5 Proof for Theorem 5

**Theorem 5.** Assume  $(k + p)$  is odd and  $k > p$ . Define  $C' = C(d, \alpha, \beta) \cup T \cup S$ , where  $C(d, \alpha, \beta)$  is defined in condition 3.10 and

$$T = \{x : -(p-1) \leq x_1 \leq p \text{ and } 1 \leq x_2 < \beta + 1 \text{ and } \sum_{i=2}^d |x_i| = \beta + 1\} \quad (\text{A.4})$$

$$S = \{x : p + 1 \leq x_1 \leq \alpha + 1 \text{ and } |x_2| \leq \beta \text{ and } \|x\|_1 = \alpha + 1\} \quad (\text{A.5})$$

Then  $\forall x, y \in C'$ ,  $x \in N(y, p, k)$ , i.e.,  $C'$  constitutes a distance- $k$  clique.

**Proof.** For brevity, we denote  $C = C(d, \alpha, \beta)$ . Let  $x, y \in C'$ , i.e.,  $x$  and  $y$  belong in one of the sets  $C$ ,  $T$ , or  $S$ . Because of symmetry, we consider the following pairs of conditions for  $(x, y)$ :  $(C, C)$ ,  $(T, T)$ ,  $(S, S)$ ,  $(C, T)$ ,  $(C, S)$ ,  $(T, S)$ .

Notice that the set  $C$  is the clique obtained by  $k' = k - 1$  and  $p$ . Then, case  $(C, C)$  is covered by Theorem 4, which bounds the (displaced) distance of any two points in  $C$  by  $k' = k - 1 < k$ . This observation can be used to show the similarity of the cases  $(C, T)$  and  $(C, S)$ . Specifically for  $(C, T)$ ,  $x \in C$  and any  $y \in T$  will be exactly at distance 1 from some point in  $C$ , which means  $\|x - y\| \leq k' + 1 = k$ . For  $(C, S)$ , a  $y \in S$  is also at a distance 1 from any point in  $C$  by extending the first dimension.

As in Theorem 4, we assume  $x_1 \leq y_1$  and show that  $x$  belongs in the left neighborhood around  $y^-$ , i.e.,  $x \in N(y^-, 0, k)$  or  $\delta = |y_1^- - x_1| + \sum_{i=2}^d |y_i - x_i| \leq k$ . We also use the following property of absolute values,

$$|f - g| - |f| - |g| = \begin{cases} 0, & \text{if } fg \leq 0, \\ -2 \cdot \min(|f|, |g|), & \text{otherwise.} \end{cases} \quad (\text{A.6})$$

### A.5.1 Case $(T, T)$ :

Using the last two conditions of 3.11, the corresponding part of Equation A.6 for  $x_2, y_2$ , and  $2\beta = k - p - 1$  we have

$$\begin{aligned} \sum_{i=2}^d |y_i - x_i| &\leq |y_2 - x_2| + \sum_{i=3}^d |y_i| + \sum_{i=3}^d |x_i| \\ &\leq |y_2 - x_2| + (\beta + 1 - |y_2|) + (\beta + 1 - |x_2|) \\ &= 2\beta + 2 - 2 \cdot \min(|x_2|, |y_2|) \leq k - p - 1. \end{aligned}$$

Then  $\delta = |y_1^- - x_1| + \sum_{i=2}^d |y_i - x_i| \leq |y_1 - p - x_1| + k - p - 1$ . Using the first condition in 3.11 we have,

If  $y_1 - p \geq x_1$  then  $\delta \leq y_1 - p - x_1 + k - p - 1 \leq (p) - p + (p - 1) + (k - p - 1) = k - 2 < k$ .

If  $y_1 - p < x_1$  or  $y_1 - x_1 < p$  then  $\delta \leq x_1 - y_1 + p + (k - p - 1) \leq p + k - p - 1 < k$ .

### A.5.2 Case $(S, S)$ :

Again we prove  $x \in N(y^-, 0, k)$ . Based on the conditions in 3.12,  $\sum_{i=3}^d |x_i| = \alpha + 1 - x_1 - |x_2|$ , and  $\sum_{i=3}^d |y_i| = \alpha + 1 - y_1 - |y_2|$ , and since  $2\alpha = k + p - 1$  we have,

$$\begin{aligned}
\delta &\leq |y_1^- - x_1| + |y_2 - x_2| + \sum_{i=3}^d |y_i| + \sum_{i=3}^d |x_i| \\
&= |y_1^- - x_1| + |y_2 - x_2| + 2\alpha + 2 - |x_2| - |y_2| - x_1 - y_1 \\
&= k + p + 1 + (|y_1 - p - x_1| - x_1 - y_1) + (|y_2 - x_2| - |x_2| - |y_2|) \\
&\leq k + p + 1 + (p + |y_1 - x_1| - x_1 - y_1) - 2 \cdot \min(|x_2|, |y_2|) \\
&\leq k + 2p + 1 - 2 \cdot \min(|x_1|, |y_1|) - 2 \cdot \min(|x_2|, |y_2|) \\
&= k + 2p + 1 - 2(p + 1) = k - 1 < k.
\end{aligned}$$

### A.5.3 Case $(T, S)$ :

Let  $x \in T$ ,  $y \in S$ . From the defining conditions,  $x_1 \leq p < y_1$ . We work similarly with the previous cases, replacing the  $\sum_{i=3}^d$ , and noting that  $\alpha + \beta = k - 1$ ,

$$\begin{aligned}
\delta &\leq |y_1^- - x_1| + |y_2 - x_2| + (\alpha + 1 - |y_1| - |y_2|) + (\beta + 1 - |x_2|) \\
&= k + 1 + (|y_1 - p - x_1| - |y_1|) + (|y_2 - x_2| - |y_2| - |x_2|) \\
&\leq k + 1 + (|y_1 - p - x_1| - |y_1|).
\end{aligned}$$

If  $y_1 - p \geq x_1$ , then  $|y_1 - p - x_1| - |y_1| = y_1 - p - x_1 - y_1 = -p - x_1 \leq -p + (p - 1) = -1$ .

Thus  $\delta \leq k$ .

If  $y_1 - p \leq x_1$ , then  $|y_1 - p - x_1| - |y_1| = x_1 + p - y_1 - y_1 \leq 2p - 2(p + 1) = -2$ . Thus,

$\delta \leq k - 1 < k$ . □

## A.6 Proof for Theorem 6

**Theorem 6.**  $N(\mathbf{0}, p \pm \lambda, k - \lambda) \subseteq N(\mathbf{0}, p, k)$ , for any  $\lambda \leq k$ .

**Proof.** We consider only the  $p + \lambda$  case as the  $p - \lambda$  has a similar proof. Because of symmetry, we consider only the positive displacements  $x^+$  and  $y^+$  from Equation 3.7. It is sufficient to show that if  $x \in N(\mathbf{0}, p + \lambda, k - \lambda)$ , then  $x \in N(y^+, 0, k)$ . From Equation 3.8 we have  $\sum_{i=2}^n |x_i| + |x_1 - (p + \lambda)| \leq k - \lambda$ . We distinguish the following cases.

- (a) If  $x_1 - p \geq \lambda$ , then also  $x_1 \geq p$ , and thus  $\sum_{i=2}^n |x_i| + x_1 - p - \lambda \leq k - \lambda \Rightarrow \sum_{i=2}^n |x_i| + |x_1 - p| \leq k \Rightarrow x \in N(\mathbf{0}, p, k)$ .
- (b) If  $x_1 < p + \lambda$ , then  $\sum_{i=2}^n |x_i| - x_1 + p \leq k - 2\lambda$ . We distinguish two sub-cases.
  - (b.1) If  $x_1 \leq p$ , then  $\sum_{i=2}^n |x_i| + |x_1 - p| \leq k - 2\lambda \leq k \Rightarrow x \in N(\mathbf{0}, p, k)$ .
  - (b.2) If  $x_1 > p$  and since  $x_1 - p < \lambda$ , then  $\sum_{i=2}^n |x_i| + p - x_1 \leq k - 2\lambda \Rightarrow \sum_{i=2}^n |x_i| + x_1 - p \leq k - 2\lambda + 2(x_1 - p) < k - 2\lambda + 2\lambda = k \Rightarrow x \in N(\mathbf{0}, p, k)$ .

□

# Bibliography

- [1] Femto. <https://www.wm.edu/offices/it/services/researchcomputing/hw/nodes/femto/index.php>. Accessed: 2024-04-30.
- [2] Server and lab machine specifications. <https://support.cs.wm.edu/index.php/specs>. Accessed: 2024-07-01.
- [3] KAPIL AHUJA, BRYAN CLARK, ERIC DE STURLER, DAVID CEPERLEY, AND JEONGNIM KIM. Improved scaling for quantum monte carlo on insulators. *SIAM Journal on Scientific Computing*, 33(4):1837–1859, 2011.
- [4] NIR AILON AND BERNARD CHAZELLE. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. STOC '06, New York, NY, USA, 2006. Association for Computing Machinery.
- [5] C. ALEXANDROU, S. BACCHIO, M. CONSTANTINOU, J. FINKENRATH, K. HADJIYIANNAKOU, K. JANSEN, G. KOUTSOU, H. PANAGOPOULOS, AND G. SPANOUEDES. Complete flavor decomposition of the spin and momentum fraction of the proton using lattice qcd simulations at physical pion mass. *Phys. Rev. D*, 101:094513, May 2020.
- [6] C. ALEXANDROU, M. CONSTANTINOU, K. HADJIYIANNAKOU, K. JANSEN, AND F. MANIGRASSO. Flavor decomposition for the proton helicity parton distribution functions. *Physical Review Letters*, 126:102003, Mar 2021.

- [7] NOGA ALON, PHILLIP B. GIBBONS, YOSSI MATIAS, AND MARIO SZEGEDY. Tracking join and self-join sizes in limited storage. *J. Comput. Syst. Sci.*, 64(3):719–747, may 2002.
- [8] NOGA ALON, YOSSI MATIAS, AND MARIO SZEGEDY. The space complexity of approximating the frequency moments. page 20–29, 1996.
- [9] PATRICK R. AMESTOY, IAIN S. DUFF, JEAN-YVES L’EXCELLENT, YVES ROBERT, FRANÇOIS-HENRY ROUET, AND BORA UÇAR. On computing inverse entries of a sparse matrix in an out-of-core environment. *SIAM Journal on Scientific Computing*, 34(4):A1975–A1999, 2012.
- [10] WALTER E. ARNOLDI. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–29, 1951.
- [11] HAIM AVRON, PETAR MAYMOUNKOV, AND SIVAN TOLEDO. Blendepik: Supercharging lapack’s least-squares solver. *SIAM Journal on Scientific Computing*, 32(3):1217–1236, 2010.
- [12] HAIM AVRON AND SIVAN TOLEDO. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM*, 58(2), apr 2011.
- [13] R. BABICH, J. BRANNICK, R. C. BROWER, M. A. CLARK, T. A. MANTEUFFEL, S. F. MCCORMICK, J. C. OSBORN, AND C. REBBI. Adaptive multigrid algorithm for the lattice wilson-dirac operator. *Physical Review Letters*, 105(20), Nov 2010.
- [14] OLEG BALABANOV AND LAURA GRIGORI. Randomized gram–schmidt process with application to gmres. *SIAM Journal on Scientific Computing*, 44(3):A1450–A1474, 2022.
- [15] GUNNAR S. BALI, SARA COLLINS, AND ANDREAS SCHÄFER. Effective noise reduction techniques for disconnected loops in lattice qcd. *Computer Physics Communications*, 181(9):1570–1583, Sep 2010.

- [16] MATTHIAS BECK AND SINAI ROBINS. *Computing the Continuous Discretely: Integer-Point Enumeration in Polyhedra*. Springer, 2007.
- [17] C. BEKAS, E. KOKIOPOULOU, AND Y. SAAD. An estimator for the diagonal of a matrix. *Appl. Numer. Math.*, 57(11–12):1214–1229, nov 2007.
- [18] MOHAMED-ALI BELABBAS AND PATRICK J. WOLFE. Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences*, 106(2):369–374, 2009.
- [19] Å. BJÖRCK. Numerics of gram-schmidt orthogonalization. *Linear Algebra and its Applications*, 197-198:297–316, 1994.
- [20] D. CALVETTI, L. REICHEL, AND DANNY C. SORESENSEN. An implicit restarted lanczos method for large symmetric eigenvalue problems. volume 2, pages 1–21. Kent State University, Department of Mathematics and Computer Science, 1994.
- [21] KE CHEN. *Matrix Preconditioning Techniques and Applications*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2005.
- [22] CHEMSEDDINE CHOHRRA, PHILIPPE LANGLOIS, AND DAVID PARELLO. Efficiency of reproducible level 1 blas. In *Scientific Computing, Computer Arithmetic, and Validated Numerics: 16th International Symposium, SCAN 2014, Würzburg, Germany, September 21–26, 2014. Revised Selected Papers*, page 99–108, Berlin, Heidelberg, 2015. Springer-Verlag.
- [23] A. K. CLINE, C. B. MOLER, G. W. STEWART, AND J. H. WILKINSON. An estimate for the condition number of a matrix. *SIAM Journal on Numerical Analysis*, 16(2):368–375, 1979.

- [24] ALICE CORTINOVIS AND DANIEL KRESSNER. On randomized trace estimates for indefinite matrices with an application to determinants. *Foundations of Computational Mathematics*, 22:875–903, 2022.
- [25] SANJOY DASGUPTA AND ANUPAM GUPTA. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.
- [26] ERNEST R. DAVIDSON. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *Journal of Computational Physics*, 17(1):87–94, 1975.
- [27] I. S. DUFF, A. M. ERISMAN, AND J. K. REID. *Direct Methods for Sparse Matrices*. Oxford University Press, 01 2017.
- [28] B. DUMITRESCU. *Positive Trigonometric Polynomials and Signal Processing Applications*. Springer Publishing Company, Incorporated, 2007.
- [29] ROBERT G. EDWARDS AND BÁLINT JOÓ. The chroma software system for lattice qcd. *Nuclear Physics B - Proceedings Supplements*, 140:832–834, Mar 2005.
- [30] ETHAN N. EPPERLY, JOEL A. TROPP, AND ROBERT J. WEBBER. Xtrace: Making the most of every sample in stochastic trace estimation. *SIAM Journal on Matrix Analysis and Applications*, 45(1):1–23, 2024.
- [31] XU FENG, WENJIAN YU, AND YAOHANG LI. Faster matrix completion using randomized svd. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 608–615, 2018.
- [32] C.A.J. FLETCHER. *Computational Galerkin Methods*. Scientific Computation. Springer-Verlag New York Inc., 1984.
- [33] JUSTIN FOLEY, K. JIMMY JUGE, ALAN Ó CAIS, MIKE PEARDON, SINÉAD M. RYAN, AND JON-IVAR SKULLERUD. Practical all-to-all propagators for lattice qcd. *Computer Physics Communications*, 172(3):145–162, Nov 2005.



- [34] ZACHARY FRANGELLA, JOEL A. TROPP, AND MADELEINE UDELL. Randomized nyström preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 44(2):718–752, 2023.
- [35] A. FRIEZE, R. KANNAN, AND S. VEMPALA. Fast monte-carlo algorithms for finding low-rank approximations. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 370–378, 1998.
- [36] ANDREAS FROMMER, KARSTEN KAHL, FRANCESCO KNECHTLI, MATTHIAS ROTTMANN, ARTUR STREBEL, AND IAN ZWAAN. A multigrid accelerated eigensolver for the hermitian wilson–dirac operator in lattice qcd. *Computer Physics Communications*, 258:107615, Jan 2021.
- [37] ANDREAS FROMMER, CLAUDIA SCHIMMEL, AND MARCEL SCHWEITZER. Analysis of probing techniques for sparse approximation and trace estimation of decaying matrix functions. *SIAM Journal on Matrix Analysis and Applications*, 42(3):1290–1318, 2021.
- [38] ARJUN SINGH GAMBHIR, ANDREAS STATHOPOULOS, AND KOSTAS ORGINOS. Deflation as a method of variance reduction for estimating the trace of a matrix inverse. *SIAM Journal on Scientific Computing*, 39(2):A532–A558, 2017.
- [39] ARJUN SINGH GAMBHIR, ANDREAS STATHOPOULOS, KOSTAS ORGINOS, BORAM YOON, RAJAN GUPTA, AND SERGEY SYRITSYN. Algorithms for Disconnected Diagrams in Lattice QCD. *PoS, LATTICE2016*:265, 2016.
- [40] WALTER GANDER. Algorithms for the qr decomposition. *Res. Rep*, 80(02):1251–1268, apr 1980.
- [41] ASSEFAW HADISH GEBREMEDHIN, FREDRIK MANNE, AND ALEX POTHEN. What color is your jacobian? graph coloring for computing derivatives. *SIAM Review*, 47(4):629–705, 2005.

- [42] GENE GOLUB AND JAMES M. ORTEGA. Scientific computing: an introduction with parallel computing. USA, 1993. Academic Press Professional, Inc.
- [43] GENE H. GOLUB, FRANKLIN T. LUK, AND MICHAEL L. OVERTON. A block lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Trans. Math. Softw.*, 7(2):149–169, jun 1981.
- [44] GENE H. GOLUB AND CHARLES F. VAN LOAN. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013.
- [45] JEREMY GREEN, NESREEN HASAN, STEFAN MEINEL, MICHAEL ENGELHARDT, STEFAN KRIEG, JESSE LAEUCHLI, JOHN NEGELE, KOSTAS ORGINOS, ANDREW POCHINSKY, AND SERGEY SYRITSYN. Up, down, and strange nucleon axial form factors from lattice qcd. *Phys. Rev. D*, 95:114502, Jun 2017.
- [46] JEREMY GREEN, STEFAN MEINEL, MICHAEL ENGELHARDT, STEFAN KRIEG, JESSE LAEUCHLI, JOHN NEGELE, KOSTAS ORGINOS, ANDREW POCHINSKY, AND SERGEY SYRITSYN. High-precision calculation of the strange nucleon electromagnetic form factors. *Physical Review D*, 92(3), Aug 2015.
- [47] VICTOR GUERRERO, RONALD B. MORGAN, AND WALTER WILCOX. Eigenspectrum noise subtraction methods in lattice qcd. 2010.
- [48] HONGBIN GUO. Computing trace of function of matrix. *Numerical Mathematics*, 9, 01 2000.
- [49] PER CHRISTIAN HANSEN. The truncated svd as a method for regularization. *BIT Numerical Mathematics*, 27(4):534–553, 1987.
- [50] A.S. HOUSEHOLDER. *Principles of Numerical Analysis*. Dover books on mathematics. McGraw-Hill, 1953.

- [51] M.F. HUTCHINSON. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*, 18(3):1059–1076, 1989.
- [52] TOMMY R. JENSEN AND BJARNE TOFT. *Introduction to Graph Coloring*. John Wiley & Sons, Ltd, 1994.
- [53] WILLIAM JOHNSON AND J. LINDENSTRAUSS. Extensions of lipschitz mappings into a hilbert space. *Conference in Modern Analysis and Probability*, 26:189–206, 01 1982.
- [54] V. KLEMA AND A. LAUB. The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, 25(2):164–176, 1980.
- [55] ANDREW KNYAZEV. Hard and soft locking hard and soft locking in iterative methods for symmetric eigenvalue problems. *Eighth Copper Mountain Conference on Iterative Methods*, 04 2004.
- [56] SCOTT P. KOLODZIEJ, MOHSEN AZNAVEH, MATTHEW BULLOCK, JARRETT DAVID, TIMOTHY A. DAVIS, MATTHEW HENDERSON, YIFAN HU, AND READ SANDSTROM. The suitesparse matrix collection website interface. *Journal of Open Source Software*, 4(35):1244, 2019.
- [57] JESSE LAEUCHLI AND ANDREAS STATHOPOULOS. Extending hierarchical probing for computing the trace of matrix inverses, 2020.
- [58] CORNELIUS LANCZOS. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45:255–282, 1950.
- [59] LIN LIN, CHAO YANG, JUAN C. MEZA, JIANFENG LU, LEXING YING, AND WEINAN E. Selinv—an algorithm for selected inversion of a sparse symmetric matrix. 37(4), 2011.

- [60] JUN S. LIU. *Monte Carlo Strategies in Scientific Computing*. Springer New York, New York, NY, 2004.
- [61] G.G. LORENTZ. *Bernstein Polynomials*, volume 8. Chelsea Publishing Company, 1986.
- [62] PER-GUNNAR MARTINSSON AND JOEL A. TROPP. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020.
- [63] D.F. MCALLISTER, G.W. STEWART, AND W.J. STEWART. On a rayleigh-ritz refinement technique for nearly uncoupled stochastic matrices. *Linear Algebra and its Applications*, 60:1–25, 1984.
- [64] XIANGRUI MENG AND MICHAEL W. MAHONEY. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, STOC '13*, page 91–100, New York, NY, USA, 2013. Association for Computing Machinery.
- [65] H. B. MEYER. *Lattice QCD: A Brief Introduction*, pages 1–34. Springer International Publishing, Cham, 2015.
- [66] RAPHAEL A. MEYER, CAMERON MUSCO, CHRISTOPHER MUSCO, AND DAVID P. WOODRUFF. *Hutch++: Optimal Stochastic Trace Estimation*, pages 142–155.
- [67] JOHN MITCHEM. On Various Algorithms for Estimating the Chromatic Number of a Graph. *The Computer Journal*, 19(2):182–183, 05 1976.
- [68] RONALD B. MORGAN. Preconditioning eigenvalues and some comparison of solvers. *Journal of Computational and Applied Mathematics*, 123:101–115, 2000.
- [69] RONALD B. MORGAN AND DAVID S. SCOTT. Generalizations of davidson’s method for computing eigenvalues of sparse symmetric matrices. *SIAM Journal on Scientific and Statistical Computing*, 7(3):817–825, 1986.

- [70] RONALD B MORGAN AND MIN ZENG. Harmonic projection methods for large non-symmetric eigenvalue problems. *Numerical linear algebra with applications*, 5(1):33–55, 1998.
- [71] CHRISTOPHER W. MURRAY, STEPHEN C. RACINE, AND ERNEST R. DAVIDSON. Improved algorithms for the lowest few eigenvalues and associated eigenvectors of large matrices. *Journal of Computational Physics*, 103(2):382–389, 1992.
- [72] YUJI NAKATSUKASA AND JOEL A. TROPP. Fast & accurate randomized algorithms for linear systems and eigenvalue problems, 2022.
- [73] JELANI NELSON AND HUY L. NGUYÊN. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 117–126, 2013.
- [74] ROLAND OMNÈS. *Consistent Histories and the Interpretation of Quantum Mechanics*, pages 215–224. Springer Netherlands, Dordrecht, 1995.
- [75] CHRISTOS H. PAPADIMITRIOU, HISAO TAMAKI, PRABHAKAR RAGHAVAN, AND SANTOSH VEMPALA. Latent semantic indexing: a probabilistic analysis. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS '98*, page 159–168, New York, NY, USA, 1998. Association for Computing Machinery.
- [76] B. N. PARLETT AND D. S. SCOTT. The lanczos algorithm with selective orthogonalization. *Mathematics of Computation*, 33(145):217–238, 1979.
- [77] BERESFORD N. PARLETT. *The Symmetric Eigenvalue Problem*. Society for Industrial and Applied Mathematics, 1998.
- [78] DAVID PERSSON, ALICE CORTINOVIS, AND DANIEL KRESSNER. Improved variants of the hutch++ algorithm for trace estimation. *SIAM Journal on Matrix Analysis and Applications*, 43(3):1162–1185, 2022.

- [79] CLAUDIA RATTI AND RENE BELLWIED. *Introduction to Lattice QCD*, pages 3–22. Springer International Publishing, Cham, 2021.
- [80] VLADIMIR ROKHLIN AND MARK TYGERT. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105(36):13212–13217, 2008.
- [81] ELOY ROMERO, ANDREAS STATHOPOULOS, AND KOSTAS ORGINOS. Multigrid deflation for lattice qcd. *Journal of Computational Physics*, 409:109356, May 2020.
- [82] YOUSEF SAAD. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.
- [83] YOUSEF SAAD. *Numerical Methods for Large Eigenvalue Problems*. Society for Industrial and Applied Mathematics, 2011.
- [84] TAMAS SARLOS. Improved approximation algorithms for large matrices via random projections. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 143–152, 2006.
- [85] ALEXANDER P SEYRANIAN, ERIK LUND, AND NIELS OLHOFF. Multiple eigenvalues in structural optimization problems. *Structural optimization*, 8(4):207–227, 1994.
- [86] HORST D. SIMON. The lanczos algorithm with partial reorthogonalization. *Mathematics of Computation*, 42(165):115–142, 1984.
- [87] ANDREAS STATHOPOULOS. Locking issues for finding a large number of eigenvectors of hermitian matrices. Technical report, William & Mary, 2005.
- [88] ANDREAS STATHOPOULOS. Nearly optimal preconditioned methods for hermitian eigenproblems under limited memory. part i: Seeking one eigenvalue. *SIAM Journal on Scientific Computing*, 29(2):481–514, 2007.

- [89] ANDREAS STATHOPOULOS, JESSE LAEUCHLI, AND KOSTAS ORGINOS. Hierarchical probing for estimating the trace of the matrix inverse on toroidal lattices. *SIAM Journal on Scientific Computing*, 35(5):S299–S322, 2013.
- [90] ANDREAS STATHOPOULOS AND JAMES R. MCCOMBS. Nearly optimal preconditioned methods for hermitian eigenproblems under limited memory. part ii: Seeking many eigenvalues. *SIAM Journal on Scientific Computing*, 29(5):2162–2188, 2007.
- [91] ANDREAS STATHOPOULOS AND JAMES R. MCCOMBS. Primme: Preconditioned iterative multimethod eigensolver—methods and software description. *ACM Trans. Math. Softw.*, 37(2), apr 2010.
- [92] ANDREAS STATHOPOULOS AND YOUSEF SAAD. Restarting techniques for the (jacobi-)davidson symmetric eigenvalue methods. *Electronic Transactions on Numerical Analysis*, 7:163–181, 1998.
- [93] ANDREAS STATHOPOULOS, YOUSEF SAAD, AND KESHENG WU. Dynamic thick restarting of the davidson, and the implicitly restarted arnoldi methods. *SIAM Journal on Scientific Computing*, 19(1):227–245, 1998.
- [94] G. W. STEWART. *Matrix Algorithms*. Society for Industrial and Applied Mathematics, 1998.
- [95] M. H. STONE. The generalized weierstrass approximation theorem. *Mathematics Magazine*, 21(4):167–184, 1948.
- [96] JOK M. TANG AND YOUSEF SAAD. A probing method for computing the diagonal of a matrix inverse. *Numerical Linear Algebra with Applications*, 19(3):485–501, 2011.
- [97] JOS THIJSEN. *Computational Physics*. Cambridge University Press, 2 edition, 2007.
- [98] LLOYD N. TREFETHEN AND DAVID BAU, III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

- [99] RICHARD S. VARGA. Extrapolation methods: theory and practice. *Numerical Algorithms*, 4:305, 1993.
- [100] WALTER WILCOX. Noise methods for flavor singlet quantities. In *Numerical Challenges in Lattice Quantum Chromodynamics*, Andreas Frommer, Thomas Lippert, Björn Medeke, and Klaus Schilling, editors, pages 127–141, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
- [101] MEI NING WONG AND FRED J. HICKERNELL. Quasi-monte carlo sampling for computing the trace of a function of a matrix, 2002.
- [102] FRANCO WOOLFE, EDO LIBERTY, VLADIMIR ROKHLIN, AND MARK TYGERT. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366, 2008.
- [103] KESHENG WU AND HORST SIMON. Thick-restart lanczos method for large symmetric eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 22(2):602–616, 2000.
- [104] LINGFEI WU, FEI XUE, AND ANDREAS STATHOPOULOS. Trpl+k: Thick-restart preconditioned lanczos+k method for large symmetric eigenvalue problems. *SIAM Journal on Scientific Computing*, 41(2):A1013–A1040, 2019.
- [105] BORAM YOON, RAJAN GUPTA, TANMOY BHATTACHARYA, MICHAEL ENGELHARDT, JEREMY GREEN, BÁLINT JOÓ, HUEY-WEN LIN, JOHN NEGELE, KOSTAS ORGINOS, ANDREW POCHINSKY, AND ET AL. Controlling excited-state contamination in nucleon matrix elements. *Physical Review D*, 93(11), Jun 2016.
- [106] MARTIN P. W. ZERNER. Directional decay of the Green’s function for a random nonnegative potential on  $\mathbf{Z}^d$ . *The Annals of Applied Probability*, 8(1):246 – 280, 1998.
- [107] RUI ZHANG, HUEY-WEN LIN, AND BORAM YOON. Probing nucleon strange and charm distributions with lattice qcd. *Physical Review D*, 104(9), Nov 2021.