Algorithms for Large Scale Problems in Eigenvalue and SVD Computations and in Big Data Applications

Lingfei Wu

Hefei, China

Bachelor of Science, Anhui University, 2007
Master of Science, University of Science and Technology of China, 2010

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

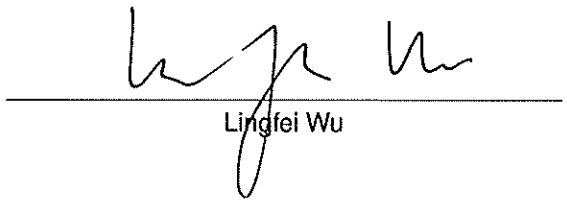
Department of Computer Science

The College of William and Mary
August 2016

APPROVAL PAGE

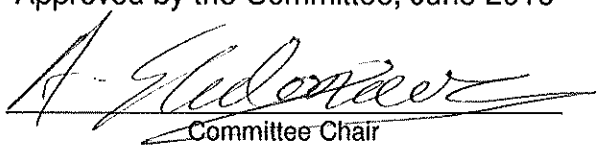
This Dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

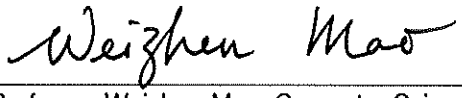


Lingfei Wu

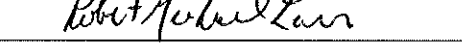
Approved by the Committee, June 2016



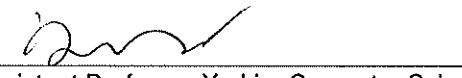
Committee Chair
Professor Andreas Stathopoulos, Computer Science
The College of William and Mary



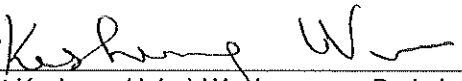
Professor Weizhen Mao, Computer Science
The College of William and Mary



Associate Professor Robert Michael Lewis, Computer Science
The College of William and Mary



Assistant Professor Xu Liu, Computer Science
The College of William and Mary



Senior Staff Scientist Kesheng (John) Wu, Lawrence Berkeley National Laboratory
The College of William and Mary

ABSTRACT

As ”big data” has increasing influence on our daily life and research activities, it poses significant challenges on various research areas. Some applications often demand a fast solution of large, sparse eigenvalue and singular value problems; In other applications, extracting knowledge from large-scale data requires many techniques such as statistical calculations, data mining, and high performance computing. In this dissertation, we develop efficient and robust iterative methods and software for the computation of eigenvalue and singular values. We also develop practical numerical and data mining techniques to estimate the trace of a function of a large, sparse matrix and to detect in real-time blob-filaments in fusion plasma on extremely large parallel computers.

In the first work, we propose a hybrid two stage SVD method for efficiently and accurately computing a few extreme singular triplets, especially the ones corresponding to the smallest singular values. The first stage achieves fast convergence while the second achieves the final accuracy. Furthermore, we develop a high-performance preconditioned SVD software based on the proposed method on top of the state-of-the-art eigensolver PRIMME. The method can be used with or without preconditioning, on parallel computers, and is superior to other state-of-the-art SVD methods in both efficiency and robustness.

In the second study, we provide insights and develop practical algorithms to accomplish efficient and accurate computation of interior eigenpairs using refined projection techniques in non-Krylov iterative methods. By analyzing different implementations of the refined projection, we propose a new hybrid method to efficiently find interior eigenpairs without compromising accuracy. Our numerical experiments illustrate the efficiency and robustness of the proposed method.

In the third work, we present a novel method to estimate the trace of matrix inverse that exploits the pattern correlation between the diagonal of the inverse of the matrix and that of some approximate inverse. We leverage various sampling and fitting techniques to fit the diagonal of the approximation to that of the inverse. Our method may serve as a standalone kernel for providing a fast trace estimate or as a variance reduction method for Monte Carlo in some cases. An extensive set of experiments demonstrate the potential of our method.

In the fourth study, we provide first results on applying outlier detection techniques to effectively tackle the fusion blob detection problem on extremely large parallel machines. We present a real-time region outlier detection algorithm to efficiently find and track blobs in fusion experiments and simulations. Our experiments demonstrated we can achieve linear time speedup up to 1024 MPI processes and complete blob detection in two or three milliseconds.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I am indebted to my advisor, Professor Andreas Stathopoulos, for providing me exceptional guidance, encouragement, and support throughout my Ph.D. study. He taught me the first course in computational methods, aroused my interests in scientific computing, and gave me an invaluable opportunity to work with him on many attractive and challenging topics. Our discussions on eigenvalue problems, singular value problems, and may other branches of numerical linear algebra were tremendously enlightening and insightful, which has nourished me to become a more independent researcher. He has been very patient and dedicated to help me improve in academic writing, and given me much freedom to do my own research in machine learning and data mining. Without him, this work would not have been possible. He is not only an extraordinary advisor, but also a nice and humorous friend. I have been deeply inspired by his serious attitude, strong beliefs and values, both in my research and life.

I am very grateful to Dr. John Wu at Lawrence Berkeley National Lab, with whom I worked during my internship in 2014 summer. His advice and opinions have always been important and insightful. Chapter 6 of this dissertation is based on the research that I have performed under his supervision. I especially want to thank Jie Chen at IBM T.J. Watson Research Center, for his valuable guidance and help on my machine learning and data mining research during my internship at IBM. I also thank all of the members of my dissertation committee for their valuable comments and guidance, which greatly improved this work.

As part of the Computational Science research group at College of William and Mary, I am very lucky to work with Jesse Harrison Laeuchli and Eloy Romero Alcalde. I am grateful for their support, collaboration, and friendship. I would like to thank my classmates and coauthors and many other friends for their company and support. I also thank all stuff members in the computer science department, especially Vanessa Godwin, Jacqulyn Johnson, and Dale Hayes for their kindly and significant help during my graduate study.

Finally, I own special thanks to my wife, Fangli Xu, who has always stood by my side. Her love, encouragement, enthusiasm and optimism have given me great happiness, strength, and confidence through my pursuit of academic career. Everything I have achieved today is due to her everlasting support in our daily life, especially in taking care of our baby girl Anna X Ling. There is no single word that I can use in this world to express my deep gratitude to her.

To My Parents, Xianhao Ling and Keyun Wu,

To My Wife and Daughter, Fangli Xu and Anna X Ling.

致我的父母亲，凌贤好和吴克云，

致我的妻子和女儿，许芳丽和凌许湘婉。

LIST OF TABLES

LIST OF FIGURES

Algorithms for Large Scale Problems in Eigenvalue and SVD Computations and in Big Data Applications

Chapter 1

Introduction

As ”big data” has increasing influence on our daily life and research activities, it poses significant challenges on various research areas. Some applications often demand a fast solution of large, sparse eigenvalue and singular value problems; In other applications, extracting knowledge from large-scale data requires many techniques such as statistical calculations, data mining, and high performance computing. In this dissertation, we develop efficient and robust iterative methods and software for the computation of eigenvalue and singular values. We also develop practical numerical and data mining techniques to estimate the trace of a function of a large, sparse matrix and to detect in real-time blob-filaments in fusion plasma on extremely large parallel computers.

The numerical solution of large, sparse, Hermitian or real symmetric eigenvalue problems is one of the most computationally intensive tasks in a variety of applications. Such applications arise in a large number of areas such as structural engineering, quantum chromodynamics, material science, dynamical systems, machine learning and data mining [45, 105, 128, 122, 62, 53]. Depending on the application, one is interested in seeking a few of the smallest or largest eigenpairs, or some eigenpairs in the interior of the spectrum. When the size of the problem is relatively small (say order 10^3), dense matrix methods such as the QR method can find all eigenvalues efficiently. In cutting edge research and industry, eigenvalue problems involve matrices of size routinely more than a

million and even of order of a billion [96]. Moreover, the matrices are often sparse, containing a significant number of entries that are zeros. For such large, sparse matrices, the dense matrix methods are usually intractable due to limited memory constraints and computational costs. Also, these problems typically demand only a small portion of the eigenvalues. Instead, iterative methods that rely on sparse matrix-vector operations are the only means of addressing these large-scale problems.

Another ubiquitous computational kernel in science and engineering is the singular value decomposition (SVD) of a matrix. The problem of computing the SVD can be formulated as an equivalent Hermitian eigenvalue problem. Many applications require a few of the largest singular values of a large, sparse matrix A and the associated left and right singular vectors (all together we call them singular triplets). These applications are from diverse areas, such as pattern recognition, social network analysis, image processing, textual database searching, and control theory [45, 105, 128, 122]. A smaller, but increasingly important set of applications require a few smallest singular triplets. Examples include total least squares minimization problem, computation of pseudospectrum, determination of range, null space and rank, and low rank approximation of matrix inverse [45, 127, 44, 135, 136, 141]. The main focus of this research is to develop efficient and robust iterative methods and software for solving difficult large, sparse eigenvalue and singular value problems.

A significant amount of research has been devoted to developing numerical analysis techniques for performing statistical calculations efficiently, in order to cope with challenges of big data arising in various areas such as social science and health informatics [47, 56, 139, 137]. For instance, a common large graph mining task is to count the number of triangles in a graph, which is very expensive with millions or billions of edges. Since an exact count of triangles is not needed in many cases, randomized algorithms for approximating the number of triangles have been developed by employing some form of Monte-Carlo simulation [6]. In this research, we focus on the need to estimate the trace of a function of a large, sparse, square matrix where the function is computed implicitly

through matrix vector products. These problems are common in many statistical applications, in image restoration [64], in uncertainty quantification [16], and in lattice quantum chromodynamics [116]. We pursue practical numerical techniques to address this computationally challenging task, specifically shedding light on computing the trace of the inverse of a large, sparse matrix.

In the era of big data, data mining techniques and high-performance computing are two key technologies for extracting knowledge from vary large volumes of a wide variety of data. Furthermore, it is important to perform real-time analysis for providing immediate feedback to the control system in many safety critical applications [61, 143, 142, 144]. Therefore, high-performance computing, grid computing and in-memory analytics are essential to accelerate processing of the huge data sets. In our research, we explore data mining techniques and high-performance computing to help physicists understand the dynamics of fusion plasma in fusion experiments or numerical simulations by leveraging extremely large parallel machines.

The research we are pursuing in this dissertation has led to several important contributions. Our first contribution is to develop a new SVD method, PHSVDS, which significantly advances the current state-of-the-art in singular value problem solving. In addition, we develop a high-performance software, PRIMME SVDS, that implements our PHSVDS method based on the state-of-the-art eigensolver package PRIMME for both largest and smallest singular values. The second contribution is to propose a new hybrid method for implementing refined projection to accurately and efficiently seek a few interior eigenvalues in non-Krylov eigensolvers. The third contribution is to present a novel method to estimate the trace of the matrix inverse by exploiting the pattern correlation between the diagonal elements of the matrix inverse and some approximation. The fourth contribution is a high-performance outlier detection and tracking method for finding blob-filaments in real fusion experiments or numerical simulations by exploring an HPC cluster.

1.1 A State-of-The-Art Preconditioned SVD Solver Software

Assume $A \in \Re^{m \times n}$ is a large sparse matrix with full column rank and $m \geq n$. The (economy size) singular value decomposition of A can be written as:

$$A = U \Sigma V^T \tag{1.1}$$

where $\Sigma = diag(\sigma_1, \ldots, \sigma_n) \in \Re^{n \times n}$ contains the singular values of A, $0 \text{`} \sigma_1 \leq \ldots \leq \sigma_n$, $U = [u_1, \ldots, u_n] \in \Re^{m \times n}$ is an orthonormal set of the left singular vectors and $V = [v_1, \ldots, v_n] \in \Re^{n \times n}$ is the unitary matrix of the right singular vectors. We will be looking for the smallest $k \ll n$ singular triplets $(\sigma_i, u_i, v_i), i = 1, \ldots, k$.

It is well known that the computation of the smallest singular triplets presents challenges both to the speed of convergence and the accuracy of iterative methods. In this research, we compute extreme singular triplets of a large, sparse matrix. Especially, we focus on the most difficult problem of finding the smallest singular triplets.

There are two approaches to compute the singular triplets (σ_i, u_i, v_i) by using a Hermitian eigensolver. Using MATLAB notation, the first approach seeks eigenpairs of the augmented matrix $B = [0\ A^T; A\ 0] \in \Re^{(m+n) \times (m+n)}$, which has eigenvalues $\pm \sigma_i$ with corresponding eigenvectors $([v_i; u_i], [-v_i; u_i])$, as well as $m - n$ zero eigenvalues [42, 43, 26]. The main advantage of this approach is that iterative methods can potentially compute the smallest singular values accurately, i.e., with residual norm close to $O(\|A\| \epsilon_{mach})$. However, convergence of eigenvalue iterative methods is slow since it is a highly interior eigenvalue problem, and even the use of iterative refinement or inverse iteration involves a maximally indefinite matrix [99]. For restarted iterative methods convergence is even slower, irregular, and often the required eigenvalues are missed since the Rayleigh-Ritz projection method does not effectively extract the appropriate information for interior eigenvectors [91, 92, 65].

The second approach computes eigenpairs of the normal equations matrix $C = A^T A \in$

$\Re^{n\times n}$ which has eigenvalues σ_i^2 and associated eigenvectors v_i. If $\sigma_i \neq 0$, the corresponding left singular vectors are obtained as $u_i = \frac{1}{\sigma_i} A v_i$. C is implicitly accessed through successive matrix-vector multiplications. The squaring of the singular values works in favor of this approach with Krylov methods, especially with largest singular values since their relative separations increase. Although the separations of the smallest singular values become smaller, we show in this thesis that this approach still is faster than Krylov methods on B because it avoids indefiniteness. On the other hand, squaring the matrix limits the accuracy at which smallest singular triplets can be obtained. Therefore, this approach is typically followed by a second stage of iterative refinement for each needed singular triplet [100, 32, 19]. However, this one-by-one refinement does not exploit information from other singular vectors and thus it is not as efficient as an eigensolver applied on B with the estimates of the first stage [140, 141, 137].

In this work, we propose a novel SVD approach that can take advantage of preconditioning and of any well designed eigensolver to compute both largest and smallest singular triplets. Accuracy and efficiency is achieved through a hybrid, two-stage meta-method, PHSVDS. In the first stage, PHSVDS solves the normal equations up to the best achievable accuracy. If further accuracy is required, the method switches automatically to an eigenvalue problem with the augmented matrix. Thus it combines the advantages of the two stages, faster convergence and accuracy, respectively. For the augmented matrix, solving the interior eigenvalue is facilitated by a proper use of the good initial guesses from the first stage and an efficient implementation of the refined projection method. We also discuss how to precondition PHSVDS and to cope with some issues that arise.

In addition, we present a high-performance software, PRIMME_SVDS, that implements our hybrid method based on the state-of-the-art eigensolver package PRIMME for both largest and smallest singular values. PRIMME_SVDS fills a gap in production level software for computing the partial SVD, especially with preconditioning. The numerical experiments demonstrate its superior performance compared to other state-of-the-art methods and its good parallel performance under strong and weak scaling.

1.2 Efficient Computation of Interior Eigenvalue Problem

The standard symmetric eigenvalue problem is to find an eigenpair (λ, x) (with $x \neq 0$) that satisfies

$$Ax = \lambda x, \tag{1.2}$$

where A is a $n \times n$ symmetric matrix. Here λ is an eigenvalue, and x is the associated eigenvector. In this research, we are interested in seeking a few interior eigenpairs. These interior eigenvalues are far enough toward the middle of the spectrum and we assume that it is impossible to factorize the matrix due to its large size and limited memory. This is a very difficult problem.

The Rayleigh-Ritz (RR) method is the best-known method for extracting the approximate eigenvectors from a subspace. This method is optimal in terms of seeking extreme eigenpairs [99], but it may give poor approximations of interior eigenvectors even in the symmetric case [91, 92, 112]. When applying the RR method for interior eigenpairs, the convergence of an iterative method can be very irregular, which makes it difficult to select appropriate vectors to approximate the interior eigenvalues [99]. Moreover, spurious approximate eigenvalues can occur when the selected vector is a combination of nearby eigenvectors. In order to circumvent these difficulties, the refined and harmonic projection methods are two alternatives for the interior eigenvalue problems [91, 65].

In this research, we provide insights and develop practical algorithms to accomplish efficient and accurate computation of interior eigenpairs using refined projection techniques in non-Krylov methods. We firstly compare different implementations of the refined projection, and analyze their numerical accuracy and computational costs. Based on the advantages of different approaches, we propose a new hybrid method to effectively find interior eigenpairs without compromising accuracy. We also provide more insights by investigating the effects of single and multiple user shifts for robustly seeking more than one eigenvalues. Our numerical experiments illustrate the efficiency and robustness of the proposed method.

1.3 A Novel Trace Estimator of Large Implicit Matrix

Computing the trace of a matrix A that is given explicitly is a straightforward operation. However, for numerous applications we need to compute the trace of a matrix that is given implicitly by its action on a vector x, i.e., Ax. Specifically, many applications are interested in computing the trace of a function of a matrix $f(A)$. The matrix A is large, and often sparse, so its action $f(A)x$ is typically computed through iterative methods.

Because it is challenging to compute $f(A)$ explicitly, the Monte Carlo (MC) approach has become the standard method for computing the trace [6, 12]. The original method is due to Hutchinson [64] and this method has been improved in many ways over the last two decades [12, 50, 129, 17, 90, 6, 116]. Given a matrix A along with an associated function $f(.)$, the sought quantity is estimated by

$$Tr(f(A)) \approx \frac{1}{s}\sum_{j=1}^{s} z_j^T f(A) z_j, \tag{1.3}$$

where z_j is a Z_2 random vector whose entries are independent and identically distributed Rademacher random variables with $Pr(z_j(i) = \pm 1) = 1/2$. The $z_j^T f(A) z_j$ is an unbiased estimator of $Tr(f(A))$, which is an important property required in some applications [116, 140]. The convergence of the estimator is determined by the standard deviation of Monte Carlo, which is often large and only decreases with the square root of the number of samples s.

In this research, we present a novel method to estimate the trace of the matrix inverse by exploiting the pattern correlation between the diagonal of the inverse of the matrix and the diagonal of some approximate inverse that can be computed inexpensively. We begin by considering a number of inexpensively computed approximations and then leverage various sampling and fitting techniques to fit the diagonal of the approximation to the diagonal of the inverse. Depending on the quality of the approximate inverse, our method may serve as a standalone kernel for providing a fast trace estimate with a small number

of samples. Furthermore, the method can be used as a variance reduction method for Monte Carlo in some cases. This is decided dynamically by our algorithm. An extensive set of experiments with various technique combinations on several matrices from some real applications demonstrate the potential of our method.

1.4 A Fast and Scalable Outlier Detection Method

To extract knowledge from the massive amounts of data available, data mining techniques are frequently used. Many traditional data mining techniques attempt to find patterns occurring frequently in the data. In this work, we explore outlier detection approaches to discover patterns occurring infrequently. Outlier detection is employed in a variety of applications such as fraud detection, time-series monitoring, medical care, and public security [61, 23, 53]. A well-known definition of "outlier" is given in [53]: "a data object that deviates significantly from the rest of the objects, as if it were generated by a different mechanism". In some cases, outliers are treated as errors or noise to be eliminated; while in many other cases, outliers can lead to the discovery of important information in the data.

Outlier detection is an important task in many safety critical environments since the outlier indicates abnormal running conditions from which significant performance degradation may well result. An outlier in these applications demands to be detected in real-time and a suitable feedback is provided to alarm the control system. Moreover, the size of ever increasing amounts of data sets dictates the needs for fast and scalable outlier detection methods. In this research, we apply the outlier detection techniques to effectively tackle the fusion blob detection problem on extremely large parallel machines. The blob-filaments are detected as outliers by constantly monitoring specific features of the experimental or simulation data and comparing the real-time data with these features.

With increased global energy needs, magnetic fusion could be a viable future energy that is inexhaustible, clean, and safe. The success of magnetically-confined fusion reactors, like the International Thermonuclear Experimental Reactor (ITER) [7], demand

steady-state plasma confinement which is challenged by the blob-filaments driven by the edge turbulence. A blob-filament (or blob) is a magnetic-field-aligned plasma structure that appears near the edge of the confined plasma, and has significantly higher density and temperature than the surrounding plasma [35]. Blobs are particularly important to study since they convect filaments of plasma outwards towards the containment wall, which results in substantial heat loss, degradation of the magnetic confinement, and erosion of the containment wall. By identifying and tracking these blob-filaments from fusion plasma data streams, physicists can improve their understanding of the dynamics and interactions of such coherent structures (blobs) with edge turbulence [143, 142, 144].

In this research, we provide first results on applying outlier detection techniques to effectively tackle the fusion blob detection problem on extremely large parallel machines. We present a real-time region outlier detection algorithm to efficiently find blobs in fusion experiments and simulations. In addition, we propose an efficient scheme to track the movement of region outliers over time. We have implemented our algorithms with hybrid MPI/OpenMP and demonstrated the effectiveness and efficiency of the proposed blob detection and tracking methods with a set of fusion simulation data. Our tests illustrate that we can achieve linear time speedup up to 1024 MPI processes and complete blob detection in two or three milliseconds using Edison, a Cray XC30 system at NERSC.

1.5 Contributions

We list the specific contributions of our research. The contributions of our SVD research are as follows:

- We present a novel preconditioned two-stage method, PHSVDS, to compute both largest and smallest singular triplets of large sparse matrices accurately and efficiently.
- We propose a dynamic scheme to automatically switch between either stage of

PHSVD for preconditioning by measuring the run-time performance based on the normal equations approach and the augmented approach to achieve optimal performance.

- We present our development in PRIMME in order to provide a state-of-the-art robust, high performance SVD solver supporting accurate computation of extreme singular triplets, for both square and rectangular matrices, with or without preconditioning. In particular, we show that, our SVD solver can take advantage of a good preconditioner, if present, in order to effectively tackle very large-scale real-world problems.

- Our extensive numerical experiments show that PHSVDS can be considerably more efficient than existing state-of-the-art methods when computing a few of the smallest singular triplets, even without a preconditioner. With a good preconditioner, the PHSVDS method can be much more efficient and more robust than current best methods. In addition, we demonstrate the good scalability of our SVD software, PRIMME_SVDS, for solving large-scale problems in various real applications under different parameter settings.

Our research on the efficient computation of the interior eigenpairs is promising and we summarize our main findings and contributions as follows:

- We first analyze different implementations of the refined projection method and present a new efficient hybrid method to compute interior eigenvalues accurately in non-Krylov methods. We illustrate the efficiency and robustness of the proposed hybrid method through an extensive set of experiments. To the best of our knowledge, this is the first implementation of an efficient and accurate method for refined projection in non-Krylov iterative methods.

- We further investigate the effect of single and multiple shifts in the refined procedure to compare their convergence behaviors when computing several interior eigenval-

ues. In addition, a new shifting strategy for selecting the appropriate approximate eigenvalue is presented.

Our research on estimating the trace of the inverse of a large matrix has yielded several important findings. We summarize them below:

- We present a novel method to estimate the trace of the matrix inverse by exploiting the pattern correlation between the diagonal elements of the inverse of the matrix and some approximation.
- Based on a dynamic evaluation of variances of different MC methods, the proposed method can be viewed as a preprocessing variance reduction method followed by MC in cases when the variance reduces sufficiently.
- The presented method may serve as a standalone kernel for providing a fast trace estimation of the matrix inverse with a small number of samples when a relative low accuracy is demanded.
- Our extensive experiments show that we typically obtain trace estimates with much better accuracy than other competing methods, and in some cases the variance is sufficiently reduced to allow for further improvements through an MC.

Our work for a real-time outlier detection method in the fusion blob detection problem has resulted in the following contributions:

- We formulate the blob detection and tracking problems as identifying different spatial region outliers over time in terms of the spatial-temporal fusion plasma data streams.
- We propose a two-phase region outlier detection method for finding blob-filaments. In the first phase, we apply a distribution-based outlier detection scheme to identify blob candidate points. In the second phase, we adopt a fast two-pass *connected component labeling* (CCL) algorithm from [131] to find different region outliers on an irregular mesh.

- We develop a high-performance blob detection approach to meet real-time feedback requirements by exploiting many-core architectures in a large cluster system. This is the first work to achieve real-time blob detection in *only a few milliseconds*.
- We propose a scheme to efficiently track the movement of region outliers by linking the centers of the region outlier over consecutive frames.
- We have implemented our blob detection algorithm with hybrid MPI/OpenMP, and demonstrated the effectiveness and efficiency of our implementation with a set of data from the XGC1 fusion simulations. Our tests show that we can achieve *linear time speedup* and complete blob detection in *two or three milliseconds* using a cluster at NERSC. In addition, we demonstrate that our method is more robust than recently developed state-of-the-art blob detection methods in [29, 94].

1.6 Organization

This proposal is organized as follows. Chapter 2 describes the motivation of our state-of-the-art SVD solver, developments, enhancements and extensive experimental results. In Chapter 3, we present our research results for analyzing the refined projection methods in non-Krylov eigenmethods. In Chapter 4, we present our new trace estimator for approximating the trace of large matrix inverse. In Chapter 5, we present our real-time blob detection method to efficiently find and track blob-filaments in fusion experiments and numerical simulations. Chapter 6 summarizes our research work and gives future work.

Chapter 2

A Start-of-The-Art Singular Value Solver Software

2.1 Introduction

The Lanczos bidiagonalization (LBD) method [45, 42] is accepted as an accurate and more efficient method for seeking singular triplets (especially smallest), and numerous variants have been proposed [78, 68, 74, 8, 9, 10, 69]. LBD builds the same subspace as Lanczos on matrix C, but since it works on A directly, it avoids the numerical problems of squaring. However, the Ritz vectors often exhibit slow, irregular convergence when the smallest singular values are clustered. To address this problem, harmonic projection [74, 8], refined projection [68], and their combinations [69] have been applied to LBD. Despite remarkable algorithmic progress, current LBD methods are still in development, with only few existing MATLAB implementations that serve mainly as a testbed for mathematical research. We show that a two stage approach based on a well designed eigenvalue code can be more robust and efficient for a few singular triplets. Most importantly, our approach can use preconditioning through the eigensolver, which is not directly possible with LBD but becomes crucial because of the difficulty of the problem even for medium matrix sizes.

The Jacobi-Davidson type SVD method, JDSVD [59, 60], is based on an inner-outer

iteration and can also use preconditioning. It obtains the left and right singular vectors directly from a projection of B on two subspaces and, although it avoids the numerical limitations of matrix C, it needs a harmonic [91, 98, 112] or a refined projection method [65, 122] to avoid irregular Rayleigh-Ritz convergence. JDSVD often has difficulty computing the smallest singular values of a rectangular matrix, especially without preconditioning, due to the presence of zero eigenvalues of B.

SVDIFP is a recent extension to the inverse free preconditioned Krylov subspace method, [46], for the singular value problem [80]. The implementation includes the robust incomplete factorization (RIF) [18] for the normal equations matrix, but other preconditioners can also be used. To circumvent the intrinsic difficulties of filtering out the zero eigenvalues of B, the method works with the normal equations matrix C, but computes directly the smallest singular values of A and not the eigenvalues of C. Thus good numerical accuracy can be achieved but, as we show later, at the expense of efficiency. Moreover, the design of SVDIFP is based on restarting with a single vector, which is not effective when seeking more than one singular values.

In this research we present a preconditioned hybrid two-stage method, PHSVDS, that achieves both efficiency and accuracy for both largest and smallest singular values under limited memory. In the first stage, the proposed method PHSVDS solves an extreme eigenvalue problem on C up to the user required accuracy or up to the accuracy achievable by the normal equations. If further accuracy is required, PHSVDS switches to a second stage where it utilizes the eigenvectors and eigenvalues from C as initial guesses to a Jacobi-Davidson method on B, which has been enhanced by a refined projection method. The appropriate choices for tolerances, transitions, selection of target shifts, and initial guesses are handled automatically by the method. We also discuss how to precondition PHSVDS and to cope with possible issues that can arise. Our extensive numerical experiments show that PHSVDS, implemented on top of the eigensolver PRIMME [118], can be considerably more efficient than all other methods when computing a few of the smallest singular triplets, even without a preconditioner. With a good preconditioner, the

PHSVDS method can be much more efficient and robust than the JDSVD and SVDIFP methods.

In addition, we present a high-performance software, PRIMME_SVDS, that implements our hybrid method based on the state-of-the-art eigensolver package PRIMME for both largest and smallest singular values. PRIMME_SVDS fills a gap in production level software for computing the partial SVD, especially with preconditioning. The numerical experiments demonstrate its superior performance compared to other state-of-the-art methods and its good parallel performance under strong and weak scaling.

In Section 2.2 we motivate the two stage SVD method based on the convergence of other Krylov methods to the smallest magnitude eigenvalue of B and C. In Section 2.3, we develop the components of the two stage SVD method. In Section 2.4, we describe how to precondition PRIMME_SVDS, and how to dynamically inspect the quality of preconditioning at the two different stages. In Section 2.5, we present extensive experiments that corroborate our conclusions. In Section 2.6, we present our development of PRIMME_SVDS on top of PRIMME. In Section 2.7, we demonstrate the performance of our PRIMME_SVDS software in parallel settings. Section 2.8 concludes and gives our future work.

We denote by $\|.\|$ the 2-norm of a vector or a matrix, by A^T the transpose of A, by I the identity matrix, $\kappa(A) = \frac{\sigma_n}{\sigma_1}$, and by $K_m(A, v) = span\{v, Av, \ldots, A^{m-1}v\}$ the m-dimensional Krylov subspace generated by A and the initial vector v.

2.2 Motivation for a two stage strategy

We first introduce some basic iterative methods for SVD, and discuss some of the features in PRIMME that facilitate the development of a flexible SVD solver. Then, we study both the asymptotic convergence of unpreconditioned Krylov methods applied on C and B, and the quality of the subspaces built by different methods. Even without taking into account the way we extract information from the subspaces, we arrive at the conclusion that a

hybrid strategy should be preferred.

We first need to understand whether an eigensolver on C or on B is preferable in terms of convergence and accuracy to iterative solvers that solve the SVD directly. To address this, we introduce the basic SVD iterative methods and study the asymptotic convergence and the quality of the Krylov subspaces built by different methods. We conclude that an appropriate choice of eigensolvers and eigenvalue problem yields methods that are faster and equally accurate as the best SVD methods, especially in the presence of limited memory.

2.2.1 The LBD, JDSVD, and SVDIFP methods

The LBD method, [42, 43], starts with unit vectors p_1 and q_1 and after k steps produces the following decomposition as a partial Lanczos bidiagonalization of A:

$$\begin{aligned} AP_k &= Q_k B_k, \\ A^T Q_k &= P_k B_k^T + r_k e_k^T, \end{aligned} \tag{2.1}$$

where the r_k is the residual vector at k-th step, e_k the $k-$th orthocanonical vector,

$$B_k = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ & \alpha_2 & \ddots & \\ & & \ddots & \beta_{k-1} \\ & & & \alpha_k \end{pmatrix} = Q_k^T A P_k,$$

and Q_k and P_k are orthonormal bases of the Krylov subspaces $K_k(AA^T, q_1)$, and $K_k(A^T A, p_1)$ respectively. With properly chosen starting vectors, LBD produces mathematically the same space as the symmetric Lanczos method on B or C [69, 74].

To approximate singular triplets of A, LBD solves the small singular value problem on B_k, and uses the corresponding Ritz approximations from Q_k and P_k as left and right singular vectors. To address the rapid loss of orthogonality of the columns of Q_k and P_k in finite precision arithmetic, full [43], partial [78], or one-sided reorthogonalization [8, 69]

strategies have been applied to variants of LBD. With appropriate implementation, these can result in a backward stable algorithm for both singular values and vectors [15, 69]. Because, all these solutions become expensive when k is large, restarted LBD versions have been studied [74, 8, 68, 78]. The goal is twofold: restart with sufficient subspace information to maintain a good convergence, and identify the appropriate Ritz information to restart with. The former problem is tackled with implicit or thick restarting [120]. The latter problem is tackled with combinations of harmonic and refined projection methods. For example, IRLBA [9] uses a thick restarted block LBD with harmonic projection, while IRRHLB [69] first computes harmonic Ritz vectors, and then uses their Rayleigh quotients in a refined projection to extract refined Ritz vectors from P_k and Q_k.

The JDSVD method [59] extends the Jacobi-Davidson method and its correction equation for singular value problems by exploiting the special structure of the augmented matrix B. Similarly to LBD, JDSVD computes singular values, not eigenvalues, of the projection matrix, and the left and right singular vectors from separate spaces. Because good quality approximations are important not only for restarting but also in the correction equation, various projection methods can benefit JDSVD. We introduce only the standard choice where the test and search space are the same.

Let U and V be the bases of the left and right search spaces. Computing a singular triplet (θ, c, d) of $H = U^T AV$ yields (θ, Uc, Vd) as the Ritz approximation of a corresponding singular triplet of A. Alternatively, the $u = Uc$ and $v = Vc$ can be computed as harmonic or refined singular triplets. Then JDSVD obtains corrections s and t for u and v by solving (approximately) the following correction equation:

$$\begin{pmatrix} P_u & 0 \\ 0 & P_v \end{pmatrix} \begin{pmatrix} -\theta I_m & A \\ A^T & -\theta I_n \end{pmatrix} \begin{pmatrix} P_u & 0 \\ 0 & P_v \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} Av - \theta u \\ A^T u - \theta v \end{pmatrix} \tag{2.2}$$

where $P_v = I_n - vv^T, P_u = I_m - uu^T$. The left and right corrections s, t are then orthogonalized against and appended to U and V respectively. JDSVD uses thick restarting [120, 132] but retains also Ritz vectors from the previous iteration, similarly to the locally

optimal Conjugate Gradient recurrence [115]. Most importantly, the JDSVD method can take advantage of preconditioning when solving (2.2).

The SVDIFP method [80] extends the EIGIFP method [46]. Given an approximation (x_i, ρ_i) at the i-th step of the outer method, it builds a Krylov space $\mathcal{V} = K_k(M(C-\rho_i I), x_i)$, where M is a preconditioner for C. To avoid the numerical problems of projecting on C, SVDIFP computes the smallest singular values of $A\mathcal{V}$, by using a two sided projection similarly to the LBD. Because the method focuses only the right singular vector, the left singular vectors can be quite inaccurate.

2.2.2 Asymptotic convergence of Krylov methods on C and B

When seeking largest singular values, it is accepted that Krylov methods on C are faster than on B [59, 74, 80, 19]. The argument is straightforward.

Theorem 2.1 *Let $\gamma_B = \frac{\sigma_n - \sigma_{n-1}}{\sigma_{n-1}+\sigma_n}$ and $\gamma_C = \frac{\sigma_n^2 - \sigma_{n-1}^2}{\sigma_{n-1}^2 - \sigma_1^2}$ be the gap ratios of the largest eigenvalue of matrices B and C, respectively. Then, for the largest eigenvalue, the asymptotic convergence of Lanczos on C is 2 times faster than Lanczos on B.*

Proof: The asymptotic convergence rate is the square root of the gap ratio. Then:

$$\gamma_C = \frac{(\sigma_n - \sigma_{n-1})(\sigma_n + \sigma_{n-1})^2}{(\sigma_n + \sigma_{n-1})(\sigma_{n-1}^2 - \sigma_1^2)} = \gamma_B \frac{(\sigma_n + \sigma_{n-1})^2}{(\sigma_{n-1}^2 - \sigma_1^2)} > \gamma_B \frac{4\sigma_{n-1}^2}{(\sigma_{n-1}^2 - \sigma_1^2)} = \frac{4\gamma_B}{1 - (\frac{\sigma_1}{\sigma_{n-1}})^2}.$$

Therefore, for $\sigma_1 \approx 0$, the asymptotic convergence rate $\sqrt{\gamma_C} > 2\sqrt{\gamma_B}$. In the less interesting case $\sigma_1 \to \sigma_{n-1}$, Lanczos on C is arbitrarily faster than on B. ■

For smallest singular values the literature is less clear, although methods that work on C have been avoided for numerical reasons. In previous experiments we have observed much faster convergence with approaches on C than on B [135]. To obtain some intuition, we perform a basic asymptotic convergence analysis of Krylov methods working on C or on B trying to compute the smallest magnitude eigenvalue.

Lemma 2.1 *Let the union of two intervals:* $K = [-a, -b] \cup [c, d]$, $-b < 0 < c$, *and* $p_k(x)$ *the optimal degree* k *polynomial that is as small as possible on* K *and* $p_k(0) = 1$. *Let* $\epsilon_k = \max_{x \in K} |p_k(x)|$, *and* $\rho = \lim_{k \to \infty} \epsilon_k^{1/k}$. *Then asymptotically:*

$$\rho \simeq 1 - \sqrt{\frac{bc}{da}}.$$

Proof: This is an application of Theorem 5 in [110]. ■

This ρ translates to an upper bound for the asymptotic convergence rate of any Krylov solver applied to an indefinite matrix whose spectrum lies in the interval K. Thus it can also be used for the convergence rate to the smallest positive eigenvalue of the augmented matrix B. Assume σ_1 is a simple eigenvalue of B and thus σ_1^2 is a simple eigenvalue of C. Define its gap ratio in C as, $\gamma = \frac{\sigma_2^2 - \sigma_1^2}{\sigma_n^2 - \sigma_2^2}$, and assume $\gamma \ll 1$.

Theorem 2.2 *Consider the spectrum of the matrix* $B - \sigma_1 I$, *which lies (except for the zero eigenvalue) in the two intervals:* $K = [-\sigma_n - \sigma_1, -2\sigma_1] \cup [\sigma_2 - \sigma_1, \sigma_n - \sigma_1]$. *The asymptotic convergence rate for any Krylov solver that finds* σ_1 *is bounded by:*

$$\rho = 1 - \sqrt{\gamma \, \frac{2\sigma_1}{\sigma_2 + \sigma_1} \, \frac{\sigma_n^2 - \sigma_2^2}{\sigma_n^2 - \sigma_1^2}}.$$

Proof: Clearly, the optimal polynomial $p_k(x)$ of Lemma 2.1 is the best polynomial for finding σ_1. Applying the Lemma for the specific bounds for this interval we get:

$$\frac{bc}{ad} = \frac{2\sigma_1(\sigma_2 - \sigma_1)}{(\sigma_n + \sigma_1)(\sigma_n - \sigma_1)} = \frac{2\sigma_1}{\sigma_2 + \sigma_1} \, \frac{\sigma_2^2 - \sigma_1^2}{\sigma_n^2 - \sigma_1^2} = \frac{2\sigma_1}{\sigma_2 + \sigma_1} \, \frac{\sigma_2^2 - \sigma_1^2}{\sigma_n^2 - \sigma_2^2} \, \frac{\sigma_n^2 - \sigma_2^2}{\sigma_n^2 - \sigma_1^2}.$$

■

Lemma 2.2 *The bound of the asymptotic convergence rate to* σ_1^2 *of Lanczos on* C *is approximately:* $q = 1 - 2\sqrt{\gamma}$.

Proof: The bound on the rate of convergence of Lanczos for σ_1^2 is approximated as $e^{-2\sqrt{\gamma}}$

[99, p. 280]. Taking the first order approximation from Taylor series around 0, we obtain $e^{-2\sqrt{\gamma}} = 1 - 2\sqrt{\gamma} + O(\gamma)$. ■

Theorem 2.3 *A Krylov method on C that computes σ_1^2 has always faster asymptotic convergence rate than a Krylov method on B that finds σ_1, by a factor of*

$$\tau = \frac{1 - \sqrt{\gamma}\sqrt{\frac{2\sigma_1}{\sigma_2+\sigma_1}\,\frac{\sigma_n^2-\sigma_2^2}{\sigma_n^2-\sigma_1^2}}}{1 - 2\sqrt{\gamma}}. \tag{2.3}$$

Proof: For the method on C to be faster it must hold $\tau > 1$ or $\frac{2\sigma_1}{\sigma_2+\sigma_1}\,\frac{\sigma_n^2-\sigma_2^2}{\sigma_n^2-\sigma_1^2} < 4$. Basic manipulations lead to the condition $(4 - 2\frac{\sigma_n^2-\sigma_2^2}{\sigma_n^2-\sigma_1^2})\sigma_1 >= -4\sigma_2$. Since $\frac{\sigma_n^2-\sigma_2^2}{\sigma_n^2-\sigma_1^2} < 1$ and all $\sigma_i > 0$, the above condition always holds. ■

First, we observe that if σ_1 is very close to 0, the normal equations approach becomes arbitrarily faster than the augmented one, as long as σ_2 remains bounded away from 0. Second, it is not hard to see that $\tau = 1 + O(\sqrt{\sigma_2 - \sigma_1})$, which means that the two approaches become similar with highly clustered eigenvalues. In that case, however, using a block method would increase the gap ratios and the gains from the approach on C would be larger again.

Most importantly, the above asymptotic convergence rates reflect optimal methods applied to C and B and an extraction of the best information from the subspaces. In practice, memory and computational requirements necessitate the restarting of iterative methods, which results in significant convergence slow down. For extremal eigenvalues, combinations of thick restarting with the locally optimal conjugate gradient directions have been shown to almost fully restore the convergence of the unrestarted Lanczos method. The GD+k and extensions to LOBPCG are such nearly-optimal methods [120, 115]. For interior eigenvalues, practical Krylov methods not only have a hard time achieving this convergence, but also have problems extracting the best eigenvectors from the subspace. Therefore, we expect in practice the normal equations to be significantly faster than any approach based on B.

2.2.3 Comparison of subspaces from Lanczos, LBD and JDSVD

We extend the discussion on Lanczos to include two native SVD methods, and infer the relative differences between their convergence by studying the subspace they build. A higher dimensional Krylov subspace implies faster convergence, assuming eigenvector approximations can be extracted effectively from the subspace. We compare LBD, JDSVD, and Lanczos (or equivalently unpreconditioned GD) on C and on B.

Suppose u_1, v_1 are left and right initial guesses. After k iterations ($2k$ matvecs), Lanczos working on the normal equations matrix C builds:

$$V_k = K_k(A^T A, v_1). \tag{2.4}$$

The LBD method builds both left and right Krylov spaces [8]:

$$U_k = K_k(AA^T, Av_1), \qquad V_k = K_k(A^T A, v_1). \tag{2.5}$$

The JDSVD method also builds two subspaces, each being a direct sum of two Krylov spaces of half the dimension [59]:

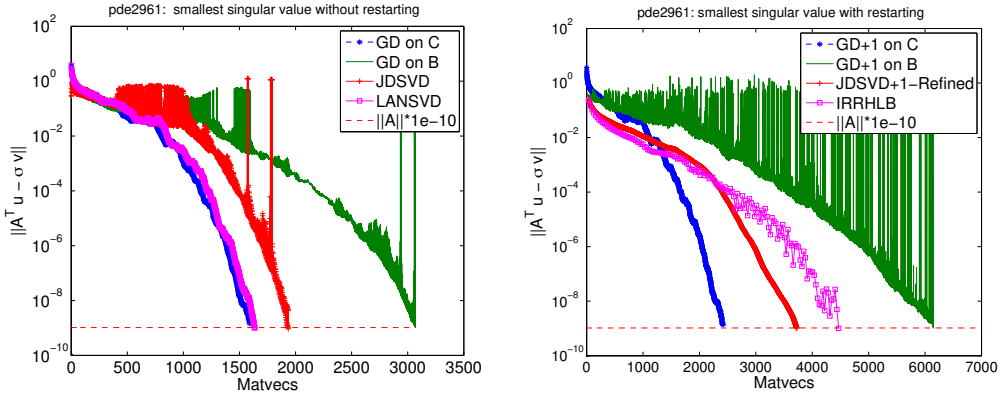
$$U_k = K_{\frac{k}{2}}(AA^T, u_1) \oplus K_{\frac{k}{2}}(AA^T, Av_1), \quad V_k = K_{\frac{k}{2}}(A^T A, v_1) \oplus K_{\frac{k}{2}}(A^T A, A^T u_1) \tag{2.6}$$

Lanczos working on B builds $K_k(B, [v_1; u_1])$ which does not correspond exactly to the spaces above in general. In the special case of $u_1 = 0$, the subspace is given below:

$$\begin{pmatrix} U_k \\ V_k \end{pmatrix} = \begin{pmatrix} 0 \\ K_{\frac{k}{2}}(A^T A, v_1) \end{pmatrix} \oplus \begin{pmatrix} K_{\frac{k}{2}}(AA^T, Av_1) \\ 0 \end{pmatrix}. \tag{2.7}$$

Clearly, Lanczos working on C and LBD build the same Krylov subspace for right singular vectors. The LBD method also builds the Krylov subspace for left singular vectors, and while that helps generate the bidiagonal projection, it does not improve convergence over Lanczos on C. On the other hand, Lanczos on B and JDSVD build a k vector sub-

space, but this comes from a direct sum of Krylov spaces of $k/2$ dimension. Thus, they are expected to take twice the number of iterations of LBD in the worst case. The JDSVD subspace can be richer than that of Lanczos on B because JDSVD handles the left and right search spaces independently for arbitrary initial guesses.



(a) unrestarted (b) restarted

Figure 2.1: Comparing convergence speed of eigenmethods on C, B, LBD, and JDSVD in both unrestarted and restarted case for matrix pde2961. LANSVD implements LBD without restarting [78] while IRRHLB is currently the most advanced LBD method with implicit restarting [69].

Figure 2.1(a) demonstrates the relative convergence behavior of these unrestarted methods seeking the smallest singular value of a sample matrix. Only the outer iteration of JDSVD is used (inner iterations = 0). The results agree with the above analysis. The convergence speed of LBD is the same as GD on C. JDSVD is slower than LBD or GD but faster than GD on B which is about twice as slow as LBD.

These methods will inevitably be used with restarting. Because LBD, JDSVD, and GD on B extract interior spectral information from the subspaces, critical directions may be dropped during restarting, causing significant convergence slow downs and irregular behavior. The use of harmonic or refined Ritz projections during restart help ameliorate this problem up to a point. However, the problem is still an interior one. In contrast, GD+k on C should see a far smaller effect on its convergence.

Figure 2.1(b) reflects the above and the advantage of solving an extreme eigenproblem with GD+k. The only disadvantage is the limited accuracy because of the squared

conditioning of C. Thus, a natural idea is to apply another phase to refine the accuracy until user requirements are satisfied. Instead of iterative refinement, we claim that a second stage eigensolver on B is more efficient.

2.2.4 Necessary eigensolver features

Our goal is to develop a method that solves large, sparse singular value problems with unprecedented efficiency, robustness, and accuracy. This requires eigensolvers with a certain set of features. First, the eigensolver should be able to use preconditioning because very slow convergence is a limiting factor for seeking smallest singular triplets. Second, large problem size suggests the use of advanced restarting techniques so that limiting memory does not impede convergence. Third, the eigensolver of the second stage should be able to exploit the good quality of the several singular vectors and singular values computed in the first stage. Fourth, the eigensolver on B can benefit from refined or Harmonic Ritz procedures for computing interior eigenvalues.

The state-of-the-art package PRIMME (PReconditioned Iterative MultiMethod Eigensolver) [118] implements the GD+k and JDQMR methods that satisfy most of the above requirements and provides a host of additional features. With a few modifications we have developed our method on top of PRIMME. However, appropriate eigensolvers from other packages could also be used.

2.3 Developing The Two Stage Strategy

We develop PHSVDS, a two-stage SVD meta-method that first gets a fast solution of the eigenvalue problem on C to the best accuracy possible, and then resolves the remaining accuracy with an eigensolver on B. We discuss and automate issues of accuracy, convergence tolerance, initial guesses, and interior eigenvalues of B.

2.3.1 The first stage of PHSVDS

Although an eigensolver on C can be much faster than other methods, the residual norms of the eigenvalues involve $\|C\| = \|A\|^2$. Thus achieving the required numerical accuracy may not be possible.

Let (σ, u, v) be a targeted singular triplet of A and $(\tilde{\sigma}^2, \tilde{v})$ the approximating Ritz pair from an eigenmethod working on C. Using the approximation $\tilde{u} = A\tilde{v}/\tilde{\sigma}$, we can write the following four residuals:

$$r_v = A\tilde{v} - \tilde{\sigma}\tilde{u}, \quad r_u = A^T\tilde{u} - \tilde{\sigma}\tilde{v}, \quad r_C = C\tilde{v} - \tilde{\sigma}^2\tilde{v}, \quad r_B = B \begin{bmatrix} \tilde{v} \\ \tilde{u} \end{bmatrix} - \tilde{\sigma} \begin{bmatrix} \tilde{v} \\ \tilde{u} \end{bmatrix}. \tag{2.8}$$

Typically a singular triplet is considered converged when $\|r_v\|$ and $\|r_u\|$ are less than a given tolerance. Since our eigenvalue methods work on C and B we need to relate the above quantities. First, it is easy to see that $r_C = A^T(A\tilde{v}) - \tilde{\sigma}^2\tilde{v} = \tilde{\sigma}A^T\tilde{u} - \tilde{\sigma}^2\tilde{v} = \tilde{\sigma}r_u$. To relate to the norm of the residual of the second stage note that $\|r_B\|^2 = (\|r_v\|^2 + \|r_u\|^2)/(\|\tilde{v}\|^2 + \|\tilde{u}\|^2)$. If the Ritz vector is normalized, $\|\tilde{v}\| = 1$, we also obtain $\|\tilde{u}\| = \|A\tilde{v}/\tilde{\sigma}\| = 1$ and $r_v = 0$. Bringing it all together (see also [135, 80]),

$$\|r_u\| = \frac{\|r_C\|}{\tilde{\sigma}} = \|r_B\|\sqrt{2}. \tag{2.9}$$

Given a user requirement $\|r_u\| < \delta \, \|A\|$, the normal equations and the augmented methods should be stopped when $\|r_C\| < \delta \, \tilde{\sigma} \, \|A\|$ and $\|r_B\| < \delta \, \|A\|/\sqrt{2}$ respectively. A common stopping criterion for eigensolvers is $\|r_C\| < \delta_C\|C\|$, so we must provide $\delta_C = \delta \, \tilde{\sigma}/\|A\|$. In floating point arithmetic this may not be achievable since $\|r_C\|$ can only be guaranteed to achieve $O(\|C\|\epsilon_{mach})$ [99]. Thus, we use

$$\delta_C = \max(\delta \, \tilde{\sigma}/\|A\|, \epsilon_{mach}) \tag{2.10}$$

as the criterion for the normal equations.

First, note that for the largest σ_n, $\delta_C = \delta$ and thus full residual accuracy is achievable with the normal equations. Since $\sigma \approx \tilde{\sigma}$, based on the Bauer-Fike bound, $|\sigma^2 - \tilde{\sigma}^2| \approx |\sigma - \tilde{\sigma}|(2\tilde{\sigma}) \leq \|r_C\| < \delta_C \|A\|^2 = \tilde{\sigma}\delta\|A\|$ and thus $|\sigma - \tilde{\sigma}| \leq \delta\|A\|/2$ so the singular values are as accurate as can be expected.

This does not hold for smaller, and in particular the smallest few, eigenvalues. Thus, if the user requires $\delta < \|A\|\epsilon_{mach}/\tilde{\sigma}$, PHSVDS first makes full use of the first stage and then switches to the second stage working on B to resolve the remaining accuracy of $O(\tilde{\sigma}/\|A\|) < \kappa(A)^{-1}$. For not too ill conditioned matrices, most of the time is then spent on the more efficient first stage.

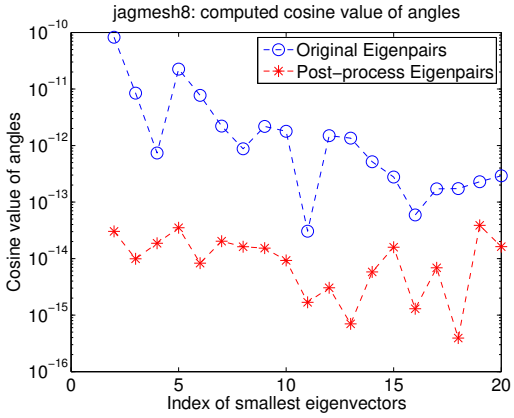
A second, more subtle issue involves the accuracy of the Ritz vectors from C which are used as initial guesses to B. We have observed that even though their residual norms are below the desired tolerance, the convergence of the interior eigenvalues in B is sometimes (but not often) irregular, with long plateaus, and might not be able to reach machine precision. This occurs when the eigenvalues are highly clustered. On the other hand, it does not occur when only one eigenvalue is sought, which implies that it has to do with the sensitivity of interior eigenvalues to the nearby eigenvectors that we pass as initial guesses [92]. Therefore, before we start stage two, we perform a complete Rayleigh Ritz procedure with the converged eigenvectors of C. Providing the new Ritz vectors as initial guesses completely cures this problem.

To understand the problem as well as the solution, consider the decomposition of the smallest Ritz vector $\tilde{u}_1$ on the exact eigenvectors of C, $\tilde{u}_1 = c_1 u_1 + \sum_{i=2}^{n} c_i u_i$. On exit from the first stage, its residual satisfies $\|r_{\tilde{u}_1}\| < \|C\|\delta_C$, and from Bauer-Fike it also holds, $\sqrt{1 - c_1^2} < \|C\|\delta_C$ and $c_i \leq \|C\|\delta_C$. Therefore, if we omit second and higher order terms, the Rayleigh quotient and the residual of $\tilde{u}_1$ can be written as:

$$\mu = \frac{\tilde{u}_1^T A \tilde{u}_1}{\tilde{u}_1^T \tilde{u}_1} = \lambda_1 + \sum_{i=2}^{n} (\frac{c_i}{c_1})^2 \lambda_i, \qquad r_{\tilde{u}_1} = A\tilde{u}_1 - \mu\tilde{u}_1 \approx \sum_{i=2}^{n} c_i(\lambda_i - \lambda_1)u_i. \tag{2.11}$$

As a result of the convergence behavior of iterative methods, the c_i tend to be larger

for nearby eigenpairs, and fall drastically as i increases. Then, from (2.11), the accuracy of μ is dominated by nearby $(c_i/c_1)^2$. The post-processing Rayleigh-Ritz uses the k nearby converged Ritz vectors, recomputes the projection with less floating point errors, and rearranges the directions to produce smaller $c_i, i = 2, \ldots, k$, and thus better Ritz values. Of course, the additional Rayleigh-Ritz cannot improve the residual norms without the incorporation of new information in the basis. Even in floating point arithmetic, the improvements are minimal. This is evident also in (2.11) where $r_{\tilde{u}_1}$ depends on the c_i linearly, so the effect of improving the nearby c_i is small. Figure 2.2 shows these effects on a matrix that presented the original problem.



Smallest eigenvalue accuracy	
Original	3.1E-13
Post-processing	1.9E-16
Smallest residual norm	
Original	4.1E-13
Post-processing	1.3E-13

Figure 2.2: The cosine of angles ($c_i = \tilde{u}_1^T u_i, i = 2, \ldots, 19$) between the smallest exact eigenvectors and the smallest Ritz vector before and after pre-processing. The table compares the accuracy of the Rayleigh quotient and the residual norm for $\tilde{u}_1$ before and post-processing for matrix jagmesh8.

In the second stage, the improvements on c_i translate to a starting search space of better quality, and thus better Ritz (or harmonic Ritz) pairs for the interior eigenvalue problem. Algorithm 1 summarizes the interaction with the first stage eigensolver.

Algorithm 1 PHSVDS first stage

1: Call an eigesolver on C to compute the required $(\tilde{\sigma}_i^2, \tilde{v}_i)$ eigenpairs
2: Eigensolver convergence test uses dynamic tolerance $\delta_C = \max\left(\delta_{user}\frac{\tilde{\sigma}_i}{\tilde{\sigma}_n}, \epsilon_{mach}\right)$, where $\tilde{\sigma}_n$ and $\tilde{\sigma}_i$ are its current largest and targeted Ritz values
3: If requested, perform Rayleigh-Ritz on the returned Ritz vector basis

2.3.2 The second stage of PHSVDS

An eigensolver such as GD+k can be used also to compute interior eigenvalues of B close to a given set of shifts. However, the availability of accurate initial guesses and shifts from the first phase suggest that an inner-outer iterative eigenmethod, such as JDQMR, might be preferable.

We argue that solving an eigenvalue problem on matrix B with an approximate eigenspace as initial guess is a better approach than iterative refinement [32, 19]. First, with iterative refinement, eigenvectors are improved one by one without any synergy from the nearby subspace information. In contrast, a subspace eigensolver provides global convergence to all desired pairs. Second, an inner-outer eigensolver such as JDQMR stops the inner linear solver dynamically and near-optimally to avoid exiting too early (which increases the number of outer iterations) or iterating too long (which increases the number of inner iterations). We are not familiar with similar implementations for iterative refinement. Third, iterative refinement for clustered interior eigenvalues may not be able to converge to the desired high accuracy due to the lack of proper deflation strategies [4], both at the linear solver and at the outer iteration. Naturally, a well designed eigensolver that employs locking and blocking techniques is more robust to address these problems. Finally, we point out that the correction equation of the Jacobi-Davidson method applied on B^T,

$$(I - ww^T)(B^T - \mu I)(I - ww^T)\tilde{t} = \tilde{\sigma}w - B^T w, \tag{2.12}$$

where $w^T = \begin{bmatrix} u^T v^T \end{bmatrix}$ is equivalent to the iterative refinement proposed in [32] ([59]). Therefore, JDQMR enjoys the benefits of both eigensolvers and iterative refinement.

Using the eigenvector approximations $\tilde{v}_i, \tilde{u}_i = A\tilde{v}_i/\tilde{\sigma}_i$ from the first stage, we form initial guesses to insert in the search space of the JDQMR on B. If the guesses are less than the minimum restart size, we fill the rest of the positions with a Lanczos space from the first targeted eigenvector. In addition, we provide the eigenvalue approximations from the first stage, which are typically very accurate because of Hermiticity, as shifts

to JDQMR. However, the problems with seeking interior eigenvalues are accentuated in the maximally indefinite case of SVD problems. Spurious Ritz values can cause Ritz vectors to fail to converge [122] or to have a detrimental effect during restart when major eigenvector components may be discarded and need to be recovered [91, 92, 112]. We have addressed these problems as follows.

First, we observed that sometimes eigenvector approximations that are introduced initially in the search space but have not been targeted yet may degrade in quality or even be displaced. Thus, when an eigenvector converges and is locked out of the search space, we re-introduce the initial guess of the next vector to be targeted. This resulted in significant improvement in robustness and often in convergence speed.

Second, we introduced an efficient implementation of the refined projection that minimizes the residual $\|BVy-\tilde{\sigma}Vy\|$ for a given $\tilde{\sigma}$ over the search space V [65, 122]. Because the shifts $\tilde{\sigma}$ are accurate, a harmonic Ritz procedure is not necessary, and the refined one is expected to give the best approximation for the targeted eigenpair. Our refined projection is similar to the one in [60, 91] which produces refined Ritz vectors for all required eigenvectors (not just the closest to $\tilde{\sigma}$). Since $\tilde{\sigma}$ remains constant, there is no need to perform a QR factorization of $BV - \tilde{\sigma}V$ at every step. Instead, as part of Gram-Schmidt, we update the factorization matrices Q and R with a new column. A full QR factorization is only needed at restart. Then, following [122], we compute the refined Ritz vectors by solving the small SVD problem with R, and replace the targeted Ritz value with the Rayleigh quotient of the first refined Ritz vector.

Solving the small SVD problem in each iteration for only the targeted shift reduces the cost of the refined procedure considerably, making it similar to the cost of computing the Ritz vectors. However, the quality of non-targeted refined Ritz vectors reduces with the distance of their Rayleigh quotient from $\tilde{\sigma}$, so they may not be as effective in a block algorithm. Yet, these approximations have the desirable property of monotonic convergence as claimed in [60, 91] and also observed in our experiments. This added robustness for JDQMR more than justifies the small additional cost. Algorithm 2 summarizes the second

Algorithm 2 PHSVDS second stage: necessary enhancements in Jacobi-Davidson

Initial shifts $\tilde{\sigma}_i$, initial vectors $[\tilde{v}_i\ ; A\tilde{v}_i/\tilde{\sigma}_i], i = 1, ..., k,\ qr\_full = 1,\ j = s + 1$
Build an orthonormal basis V of $[K_{s-k}(B, \tilde{v}_1), \tilde{v}_i]$. Set t as the Lanczos residual
while all k eigenvalues have not converged **do**
    Orthonormalize t against V. Update $v_j = t, w_j = Bv_j, H_{:,j} = V^T w_j$
    if $qr\_full == 1$ **then**
        $W - \tilde{\sigma}_1 V = QR,\ qr\_full = 0$
    else
        During orthogonalization update QR by one-column
    end if
    Compute eigendecomposition $H = S\Theta S^T$ with θ_j ordered by closeness to $\tilde{\sigma}_1$
    Compute SVD decomposition of $R = U\Sigma S^T$, Rayleigh quotient $\theta_1 = s_1^T H s_1$
    If $(\sigma_i, [v_i; u_i])$ converged, lock, and re-introduce $[\tilde{v}_{i+1}\ ; A\tilde{v}_{i+1}/\tilde{\sigma}_{i+1}]$ into V
    If restarting, set $qr\_full = 1$
    Obtain the next vector $t = Prec(r)$ (e.g., by solving the correction equation)
end while

stage modifications in the context of JD.

2.3.3 Outline of the implementation

We have implemented the PHSVDS meta-method as a MATLAB function on top of PRIMME. This allowed us flexibility for algorithmically tuning the two stages and experimenting with various eigensolvers. We first developed a MATLAB MEX interface for PRIMME, which exposes its full functionality to a broader class of users, who can now take advantage of MATLAB's built-in matrix times block-of-vectors operators and preconditioners. Its user interface is similar to MATLAB `eigs` and it is fully tunable. Many of the enhancements, such as the refined projection method or a user provided stopping criterion, were implemented directly in PRIMME and will be part of its next release. We are currently working on a native C implementation of PHSVDS in PRIMME.

PHSVDS expects an input matrix A, or a matrix function that performs matrix-vector operations with A and A^T, or directly with B and/or C. Then, it sets up the matrix-vector functions and calls Algorithm 1. The returned approximations are provided as initial guesses to Algorithm 2. One exception is when the singular value is extremely small or

zero, so the first stage yields no accuracy for $\tilde{u}_i$. In this extreme case, it is better to choose $\tilde{u}_i$ as a random vector. For tiny and highly clustered singular values, eigensolvers with locking and block features are preferable.

2.4 Preconditioning in PHSVDS

The shift-invert technique is sometimes thought of as a form of preconditioning. If a factorization of A, C or B is possible, this is often the method of choice for highly clustered or indefinite eigenproblems. For smallest singular values, MATLAB `svds` relies solely on shift-invert ARPACK [79] using the LU factorization of B. PROPACK [77, 78] uses a QR factorization of C. However, for rectangular matrices `svds` often converges to the zero eigenvalues of B rather than the smallest singular value (see [80]). Our method can also be used in shift-invert mode, assuming the user provides the inverted operator as a matrix-vector. For large matrices, however, preconditioners become a necessary alternative.

JDSVD accepts a preconditioner for a square matrix A or, if A is rectangular, leverages a preconditioner for $B - \tau I$ [59]. SVDIFP includes by default the robust incomplete factorization (RIF) method [18] to provide a preconditioner directly for $C - \tau I$ without forming C [80]. The advantage is that it works seamlessly for both square and rectangular matrices, but RIF may not be the best choice of preconditioner.

PHSVDS accepts any user-provided preconditioning operator that the underlying eigensolver allows. In the most general form, any preconditioner directly for C or B can be used. When $M \approx A$ is available (e.g., the incomplete LU factorization of a square matrix), PHSVDS forms $M^{-1}M^{-T}$ and $[0\ M^{-1}; M^{-T}\ 0]$ as the preconditioning operators for the different stages. Moreover, if a preconditioner such as RIF is given, $M \approx C^{-1}$, we can build preconditioners for the second stage as $[0\ AM; MA^T\ 0]$. It is not clear in general how to form a preconditioner for C from a preconditioner of B.

2.4.1 A dynamic two stage method with preconditioning

The analysis in Sections 2.2.2 and 2.2.3 holds for Krylov methods but it is less meaningful with preconditioners. Clearly, if two different preconditioners are provided for C and B their relative strengths are not known by PHSVDS. But we have also noticed cases where the first stage benefits less than the second stage when a less powerful preconditioner M for A is used to form preconditioners for both C and B. If M is ill-conditioned but its near-kernel space does not correspond well to that of A, it may work for B, but taking $M^T M$ produces an unstable preconditioner for C [106]. On the other hand, with a sufficiently good preconditioner, both methods enjoy similar benefits on convergence. If the relative strengths of the provided preconditioner are known, users can choose the two-stage approach or only one of the stages (e.g., the second one). For the general case, we present a method that, based on runtime measurements, switches dynamically between the normal equations and the augmented approach to identify the most effective one for the given preconditioning. This is shown in Algorithm 3.

To estimate the convergence of the two approaches, we run a set of initial tests alternating between running on C and on B. Because JDQMR relies on good initial guesses which are not available initially, the dynamic algorithm uses only the GD+k method. Once the algorithm decides on the approach, any eigenmethod that allows for preconditioning can be used. Without loss of generality, we only consider GD+k for our dynamic PHSVDS experiments. The approximations obtained from one run are passed as initial guesses to the next run.

We estimate the convergence rate by measuring the average reduction per iteration of the residual norm. To capture the convergence at different phases of the iterative method, we must switch between the two approaches several times. However, switching too frequently incurs a lot of overhead (rebuilding the initial basis, performing extra Rayleigh-Ritz procedures, and possibly convergence loss from restarting the search space). Switching too infrequently may be wasteful when the preconditioner for C does not work well. Thus,

we control the maximum number of iterations for the next GD+k run, $maxIter$. This number is always larger than $initIter$ which is a reasonably small number, i.e., 50. If the same approach is chosen in two successive runs, $maxIter$ doubles. If the approach should be switched, $maxIter$ is reduced more aggressively for the next run (Step 12) to avoid wasting too much time on the wrong approach. If one eigenvalue converges in the initial tests or at least two eigenvalues converge later, we stop the dynamic switching and choose the currently faster approach. If the faster approach is the normal equations, a two-stage method might be necessary to get to full accuracy. Although two or three switches typically suffice to distinguish between approaches, we also limit the number of switches.

Algorithm 3 Dynamic switching between stages for preconditioned PHSVDS

1: Set $initIter$, $maxSwitch$
2: $numSwitch = numConverged = j = 0, maxIter = initIter, Undecided = true$
3: Run $initIter$ iterations of an eigensolver (such as GD+k) on C and on B and collect initial average convergence rate of both approaches.
4: **while** ($numSwitch < maxSwitch$ and $Undecided$) **do**
5: Choose estimated faster approach (C or B) for next call
6: **if** ($numSwitch == 0$ **and** $numConverged > 0$) **or** $numConverged > 1$ **then**
7: undecided = false (Choose faster approach and no more switching)
8: **else**
9: **if** Same approach is chosen again **then**
10: $j = j + 1;\ \ maxIter = initIter * 2^j$
11: **else if** Different approach is chosen **then**
12: $j =$ floor$(j/2);\ \ maxIter = initIter * 2^j$
13: **end if**
14: **end if**
15: $numSwitch = numSwitch + 1$
16: Call the eigensolver with $maxIter$ and current chosen approach
17: **end while**
18: **if** All desired singular triplets are found on B **then**
19: Return final singular triplets to users
20: **else if** All desired singular triplets are found on C **then**
21: Return resulting singular triplets to augmented approach
22: **else if** Faster approach is on C **then**
23: Proceed with the two-stage approach
24: **else if** Faster approach is on B **then**
25: Continue only with the augmented approach
26: **end if**

2.5 Numerical Experiments

Our first two experiments use diagonal matrices to demonstrate the principle of the two stage method and that the method can compute artificially clustered tiny singular values to full accuracy. Then, we conduct an extensive set of experiments for finding the smallest singular values of several matrices. Large singular values are also computed under the shift-invert setting. The matrix set is chosen to overlap with those in other researchs in the literature. We compare against several state-of-the-art SVD methods: JDSVD [59, 60], SVDIFP [80], IRRHLB [69], IRLBA [8], lansvd [78], and MATLAB's `svds`. All methods are implemented in MATLAB. First, we compute a few of the smallest singular triplets on both square and rectangular matrices without a preconditioner. Then we show show the effect of the dynamic PHSVDS for different quality of preconditioners, and demonstrate that PHSVDS provides faster convergence over other methods on these test matrices as well as on some large scale problems.

All computations are carried out on a DELL dual socket with Intel Xeon processors at 2.93GHz for a total of 16 cores and 50 GB of memory running the SUSE Linux operating system. We use MATLAB 2013a with machine precision $\epsilon = 2.2 \times 10^{-16}$ and PRIMME is linked to the BLAS and LAPACK libraries available in MATLAB. Our stopping criterion requires that the left and right residuals satisfy,

$$\sqrt{\|r_u\|^2 + \|r_v\|^2} < \|A\|\delta_{user}. \tag{2.13}$$

For JDSVD we use the refined projection method as it performed best in our experiments, which is also consistent with [59]. We choose the default for all parameters except for setting 'krylov = 0' to avoid occasional convergence problems for smallest singular values. For SVDIFP the maximum number of inner iterations can be chosen as fixed or adaptive. We run with both choices and report the best result. Also, we modify slightly the code to use singular triplet residuals as the stopping criterion for SVDIFP instead of the

default residual of the normal equations. For IRLBA and IRRHLB, we choose all default parameters as suggested in the code.

All methods start with the same initial guess, `ones(min(m,n),1)`, except for matrix lshp3025 for which a random guess is necessary. We set the maximum number of restarts to 5000 for IRRHLB and IRLBA and to 10000 for JDSVD and PHSVDS. Since SVDIFP can only set a maximum number of iterations for each targeted singular triplet, we report that SVDIFP cannot converge to all desired singular values if its overall number of matrix-vector operations is larger than $(maxBasisSize - k) * 5000$. For PHSVDS, we set maxBasisSize=35, minRestartSize=21 and experiment with two δ tolerances, 1E-8 and 1E-14. For $\delta =$1E-8, PHSVDS does not need to enter the second stage for any of our tests. For the first stage of PHSVDS we use the GD_Olsen_PlusK method. For the second stage, we run experiments with both GD_PlusK and JDQMR. Since our implementation is mainly in C, we compare the number of matrix-vector operations as the primary measurement of the performance. However, we also report execution times which is relevant since matrix-vector, preconditioner, and all BLAS/LAPACK operations are performed by the MATLAB libraries.

Since the numbers of matrix-vector products with A and A^T are the same, the tables report as "MV" the number of products with A only. "Sec" is the run time in seconds, and "–" means the method cannot converge to all desired singular values or that the code breaks down. Bargraphs report the ratio of matvecs and time of each method over the PHSVDS method that uses the first stage only or JDQMR in stage two. Ratio values are truncated to less than 10, and empty bar means the same as "–" in the tables.

2.5.1 PHSVDS for clustered tiny singular values

We illustrate first how our two-stage method works in a seamless manner. We consider a diagonal matrix A = diag([1:10,1000:100:1E6]) and the preconditioner M = A + diag(rand(1,10000)*1E4). In Figure 2.3, the green and black lines show the convergence

behaviors of GD+k on B and C respectively. Indeed, the convergence on B is very slow due to a highly indefinite problem while the accuracy on C stagnates when reaching its limit. The two stage PHSVDS combines the benefits of the two methods, and determines the smallest singular value efficiently and accurately as the magenta line shows.

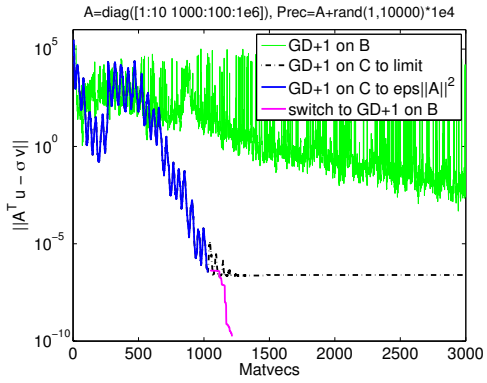


Figure 2.3: Two stages of PHSVDS working seamlessly to find smallest singular values accurately.

Next we show how PHSVDS can determine several clustered tiny singular values. Consider the matrix A = diag([1E-14, 1E-12, 1E-8:1E-8:4E-8, 1E-3:1E-3:1]) with identity matrix as a preconditioner. Although locking may be able to determine clustered or multiple singular values, we increase robustness by using a block size of two for the first stage only. We set the user tolerance δ_{user} = 1E-15 to examine the ultimate accuracy of PHSVDS. As shown in Table 2.1, PHSVDS is capable of computing all the desired clustered, tiny singular values accurately.

Table 2.1: Computation of 10 clustered smallest singular values by PHSVDS with block size 2.

σ_i	PHSVDS $\tilde{\sigma}_i$	$\|r_B\|$	σ_i	PHSVDS $\tilde{\sigma}_i$	$\|r_B\|$
1E-14	9.952750930151887E-15	4.9E-16	4E-8	4.000000001929591E-08	9.2E-16
1E-12	1.000014102726476E-12	8.9E-16	1E-3	1.000000000000025E-03	9.7E-16
1E-8	1.000000003582006E-08	6.5E-16	2E-3	1.999999999999956E-03	8.1E-16
2E-8	2.000000002073312E-08	9.2E-16	3E-3	2.999999999999997E-03	9.1E-16
3E-8	3.000000000893043E-08	9.0E-16	4E-3	3.999999999999958E-03	9.8E-16

2.5.2 Without preconditioning

We compare two variants of PHSVDS with four methods, JDSVD, SVDIFP, IRRHLB, and IRLBA on both square and rectangular matrices without preconditioning. Since a good preconditioner is usually not easy to obtain for SVD problems, it is important to examine the effectiveness of a method in this case. We compute $k = 1, 10$ smallest singular triplets (see [136] for a more detailed and extensive set of results). In order to speed up the convergence of `svds`, IRRHLB and IRLBA, we compute $k + 3$ eigenvalues when k eigenvalues are required. For PHSVDS, we found this is not necessary.

Table 2.2: Properties of the test matrices. $\gamma_m(k) = \min_{i=1}^{k}(gap(\sigma_i))$, and $gap(\sigma_i) = min_{j \neq i}|\sigma_i - \sigma_j|$.

Matrix	pde2961	dw2048	fidap4	jagmesh8	wang3	lshp3025
order	2961	2048	1601	1141	26064	3025
nnz(A)	14585	10114	31837	7465	77168	120833
$\kappa(A)$	9.5E+2	5.3E+3	5.2E+3	5.9E+4	1.1E+4	2.2E+5
$\|A\|_2$	1.0E+1	1.0E+0	1.6E+0	6.8E+0	2.7E-1	7.0E+0
$\gamma_m(1)$	8.2E-3	2.6E-3	1.5E-3	1.7E-3	7.4E-5	1.8E-3
$\gamma_m(5)$	2.4E-3	2.9E-4	2.5E-4	4.8E-5	1.9E-5	1.8E-4
$\gamma_m(10)$	7.0E-4	1.6E-4	2.5E-4	4.8E-5	6.6E-6	2.2E-5

Matrix	well1850	lp_ganges	deter4	plddb	ch	lp_bnl2
rows m:	1850	1309	3235	3049	3700	2324
cols n:	712	1706	9133	5069	8291	4486
nnz(A)	8755	6937	19231	10839	24102	14996
$\kappa(A)$	1.1E+2	2.1E+4	3.7E+2	1.2E+4	2.8E+3	7.8E+3
$\|A\|_2$	1.8E+0	4.0E+0	1.0E+1	1.4E+2	7.6E+2	2.1E+2
$\gamma_m(1)$	3.0E-3	1.1E-1	1.1E-1	4.2E-3	1.6E-3	7.1E-3
$\gamma_m(5)$	3.0E-3	2.4E-3	8.9E-5	5.1E-5	3.6E-4	1.1E-3
$\gamma_m(10)$	2.6E-3	8.0E-5	8.9E-5	2.0E-5	4.0E-5	1.1E-3

We select six square and six rectangular matrices from other research researchs [69, 80] and the University of Florida Sparse Matrix Collections [28]. Table 2.2 lists these matrices along with some of their basic properties. Among them, the matrices pde2961, dw2048, well1850 and lp_ganges have relative larger gap ratios and smaller condition number, and thereby are easy ones. Matrices fidap4, jagmesh8, wang3, deter4, and

plddb are hard cases, and matrices lshp3025, ch, and lp_bnl2 are very hard ones. We expect all methods to perform well for solving easy problems. Harder problems tend to magnify the difference between methods.

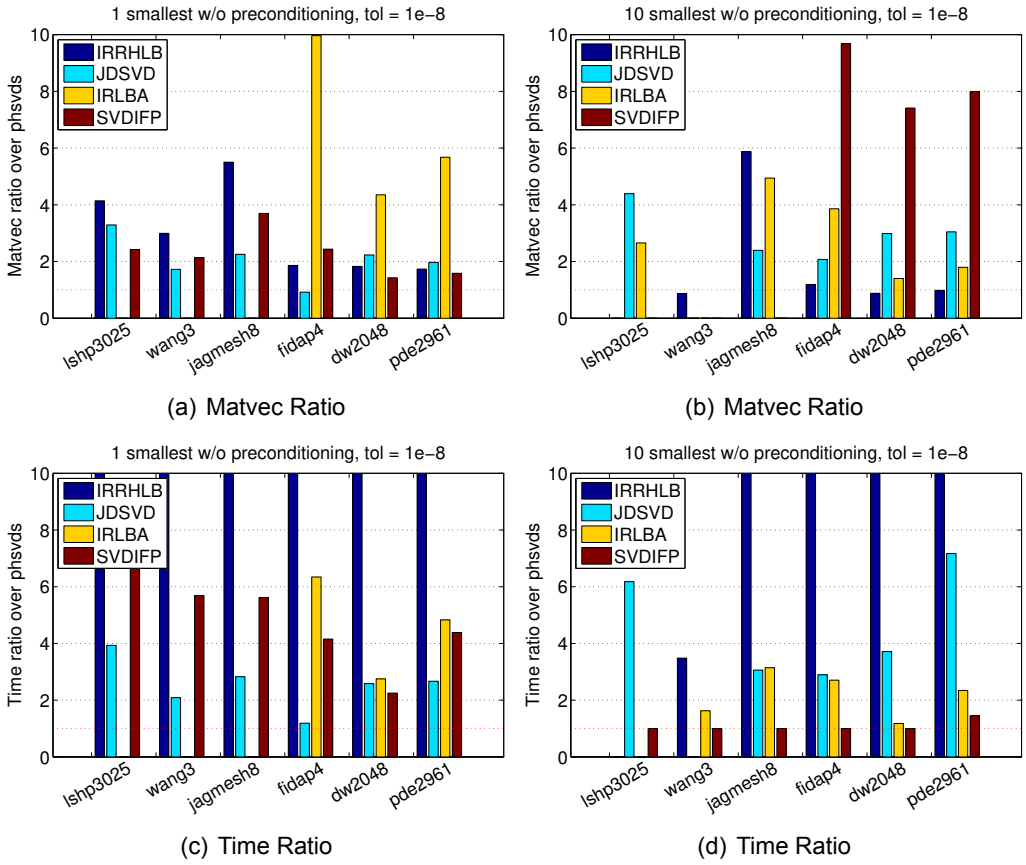


Figure 2.4: Matvec and time ratios over PHSVDS(1st stage only) when seeking 1 and 10 smallest singular triplets of square matrices with user tolerance 1E-8.

Figures 2.4, 2.5, 2.6 and 2.7 show that PHSVDS variants converge faster and more robustly than all other methods on both square and rectangular matrices. Specifically, Figure 2.4 shows that for moderate accuracy the normal equations solved with GD+k are significantly faster. For instance, PHSVDS is at least two or three times faster than other methods when solving hard problems for any number of smallest singular values. In fact, we have noticed that even for moderate accuracy, all other methods are challenged by hard problems, where they are often inefficient or even fail to converge to all desired

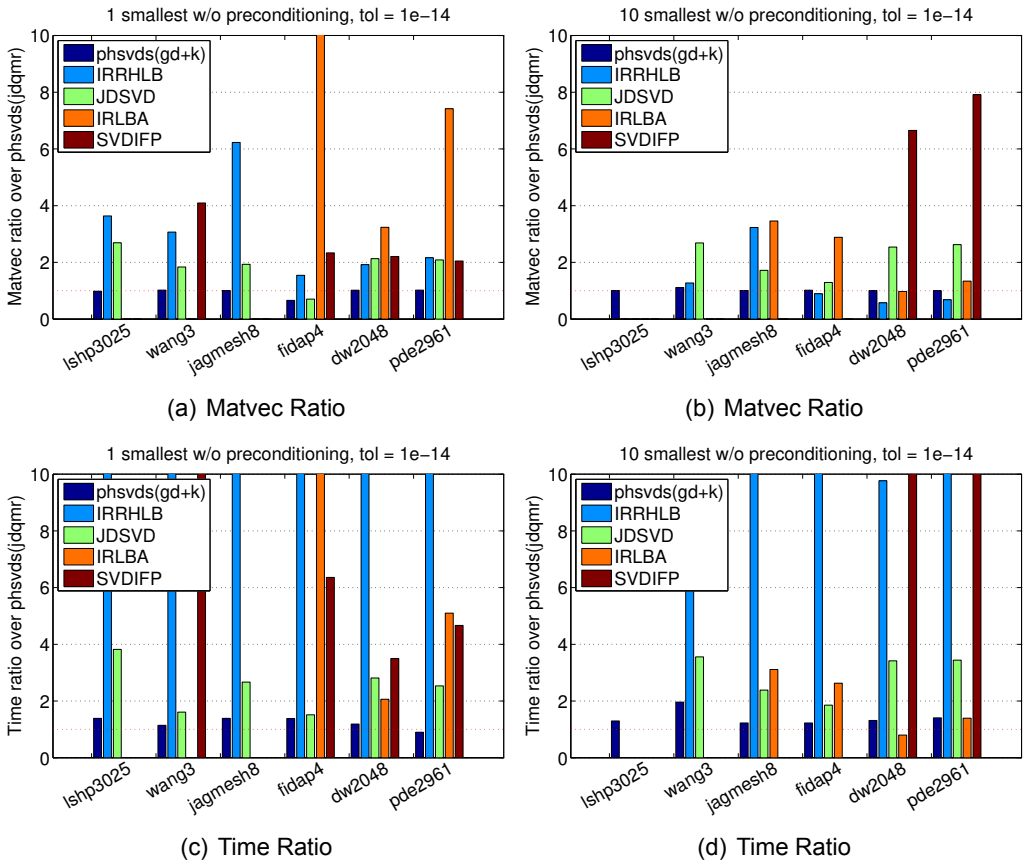


Figure 2.5: Matvec and time ratios over PHSVDS(JDQMR) when seeking 1 and 10 smallest singular triplets of square matrices with user tolerance 1E-14. The PHSVDS(GD+k) variant uses the GD+k eigenmethod in the second stage.

singular values. When solving easy problems, still PHSVDS is faster than other methods and only IRRHLB can be competitive when seeking 10 singular values. This better global convergence for many eigenvalues is typical of the Lanczos method. The superiority of PHSVDS is a result of using a near-optimal eigenmethod.

Figure 2.5 shows that the PHSVDS(GD+k) variant is comparable in terms of matvecs to PHSVDS(JDQMR), which is the base of the ratios, but the JDQMR typically requires less time if the matrix is sparse enough. It also shows that despite the slower convergence on the augmented matrix in stage two, the higher accuracy requirement does not help the rest of the methods. For computing 10 eigenpairs, IRRHLB shows a small edge in the number of iterations for two easy cases. However, PHSVDS method never misses

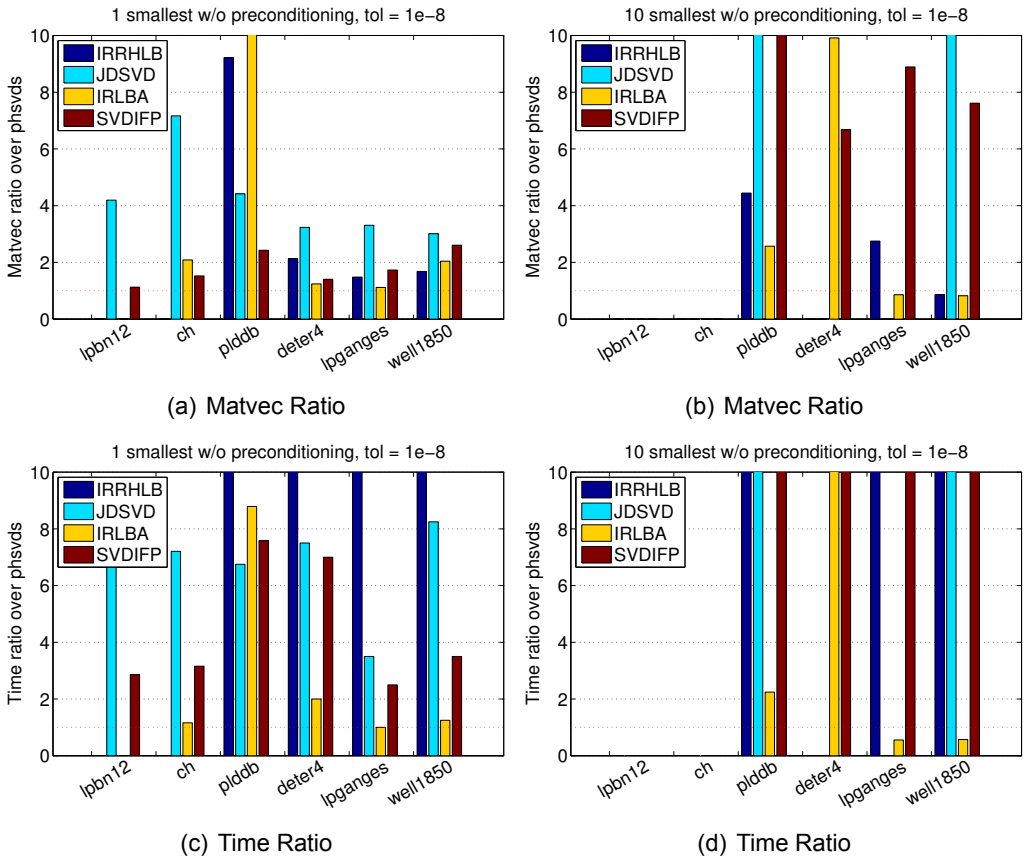


Figure 2.6: Matvec and time ratios over PHSVDS(1st stage only) when seeking 1 and 10 smallest singular triplets of rectangular matrices with user tolerance 1E-8.

eigenvalues, is consistently much faster than all other methods, and significantly faster than IRRHLB in hard cases. SVDIFP is also not competitive, partly due to its inefficient restarting strategy. Interestingly, not only does PHSVDS enjoy better robustness but also its execution time is ten times faster than IRRHLB for the cases where IRRHLB takes fewer matvecs.

Figures 2.6 and 2.7 show that the advantage of PHSVDS is even more significant on rectangular matrices. For example, except for the two easy problems well1850 and lp_ganges, PHSVDS is often five or ten times faster than the other methods. The reason is the beneficial use of the first stage, but also because PHSVDS works on C with dimension min(m, n), which saves memory and computational costs. SVDIFP also shares

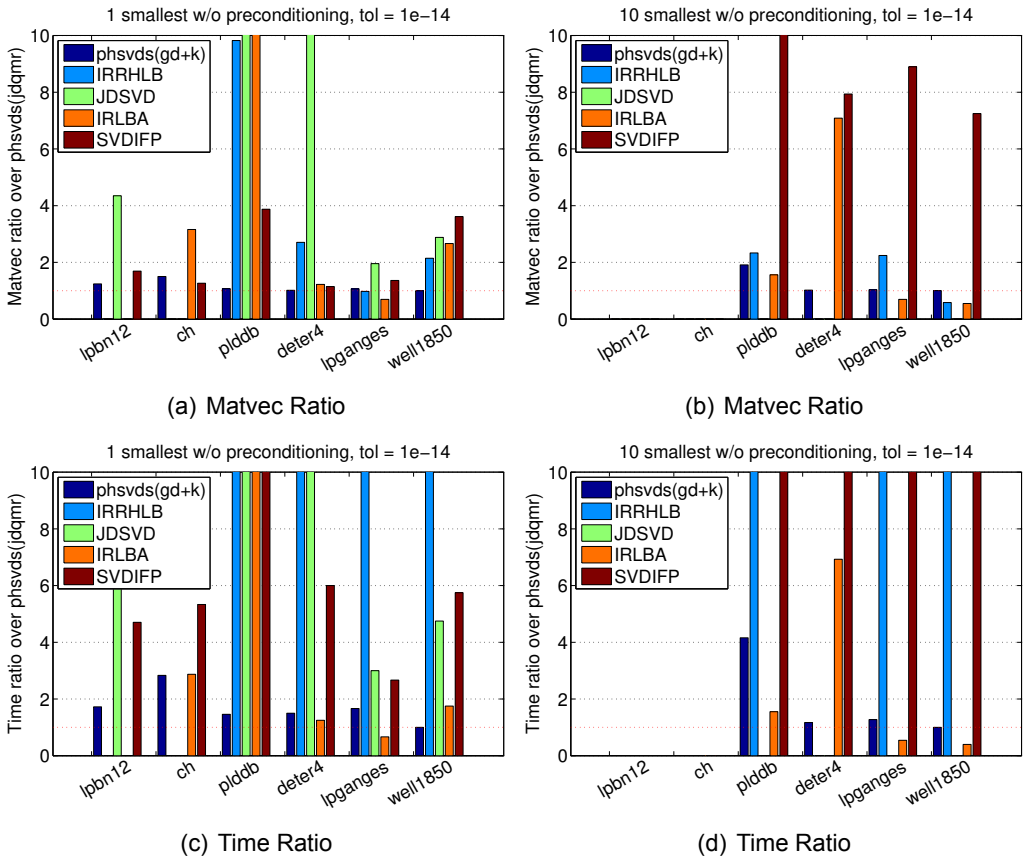


Figure 2.7: Matvec and time ratios over PHSVDS(JDQMR) when seeking 1 and 10 smallest singular triplets of rectangular matrices with user tolerance 1E-14. The PHSVDS(GD+k) variant uses the GD+k eigenmethod in the second stage.

this advantage. Interestingly, JDQMR converges much faster than GD+k on some hard problems such as plddb, ch and lp_bnl2 in Figure 2.7. The reason is the availability of excellent shifts from the first stage. We conclude that PHSVDS is the fastest method and sometimes the only method that converges for hard problems without preconditioning.

2.5.3 With preconditioning

The previous figures show the remarkable difficulty of solving for the smallest singular values. Preconditioning is a prerequisite for practical problems, which limits our choice to PHSVDS, JDSVD and SVDIFP.

We first compare our two stage method and our dynamic two-stage method for two dif-

ferent quality preconditioners. We choose $M = LU$, the factorization obtained from MATLAB's ILU function on a square matrix A with parameters `'type=ilutp'`, `'thresh=1.0'`, and varying `'droptol=1E-2'` or `'droptol=1E-3'`. Given these two M, we form the preconditioners for PHSVDS as $M^{-1}M^{-T}$ and $[0\ M^{-1}; M^{-T}\ 0]$. Without loss of generality, PHSVDS chooses GD+k for both stages. We seek ten smallest singular values with tolerance 1E-14.

As shown in Table 2.3, both variants of PHSVDS can solve the problems effectively with a good preconditioner (`'droptol=1E-3'`). In this case, the static two stage method is always better than the dynamic one because of the overhead incurred by switching between the two methods. On the other hand, when using the preconditioner with `'droptol=1E-2'`, the two stage PHSVDS is slower than the dynamic in some cases, and in the case of lshp3025, much slower. The reason is the inefficiency of the preconditioner in the normal equations. Our dynamic PHSVDS can detect the convergence rate difference and choose the faster method to accomplish the remaining computations. Of course, if this issue is known beforehand, users can bypass the dynamic heuristic and call directly the second stage.

Table 2.3: Dynamic stage switching PHSVDS (D) vs two stage PHSVDS (P) with two preconditioners: (H)igh quality ILU(1e-3) and (L)ow quality ILU(1E-2). We seek 10 smallest triplets with $\delta =$1E-14.

		pde2961		dw2048		fidap4		jagmesh8		wang3		lshp3025	
		MV	Sec	MV	Sec	MV	Sec	MV	Sec	MV	Sec	MV	Sec
H	P	166	0.4	211	0.7	210	1.3	163	0.5	306	5.5	209	3.0
H	D	242	0.4	283	0.7	286	1.4	223	0.6	396	5.5	273	3.6
L	P	258	0.5	673	1.6	813	3.2	990	3.1	736	8.9	7631	132
L	D	307	0.5	668	1.5	1043	3.7	547	1.6	1038	9.6	696	10

Next, we compare the two stage PHSVDS with JDSVD and SVDIFP with a good quality preconditioner. Except for preconditioning, all other parameters remain as before. For the first preconditioner we use MATLAB's ILU on a square matrix A. For the second preconditioner we use the RIF MEX function provided in [80] on a rectangular matrix with

'droptol=1E-3'. The resulting RIF factors $LDL^T \approx A^T A$, where D is diagonal matrix with 0 and 1 elements, are used to construct the pseudoinverses $M^{-1} = L^{-T} L^{-1} A^T$ and $M^{-T} = A L^{-T} L^{-1}$ for preconditioning the second stage of PHSVDS and JDSVD. To obtain uniform behavior across methods, we disable in SVDIFP the parameter 'COLAMD', which computes an approximate minimum degree column permutation to obtain sparser LU factors. For JDSVD, we try both enabling and disabling the initial Krylov subspace and report the best result.

Table 2.4: Seeking 1 and 10 smallest singular triplets with ILU, droptol = 1E-3. We report results from both PHSVDS variants: PHSVDS(GD+k) and PHSVDS(JDQMR). We report separately the time for generating the preconditioner and the time for running each method.

	$\delta =$ 1E-8 Matrix:	fidap4		jagmesh8		wang3		lshp3025	
	ILU Time:	0.1		0.1		2.8		0.1	
k	Method	MV	Sec	MV	Sec	MV	Sec	MV	Sec
1	PHSVDS(1st stage only)	15	0.1	13	0.1	46	2.5	19	0.3
1	JDSVD	67	0.7	34	0.3	45	2.0	56	1.5
1	SVDIFP	58	0.4	51	0.2	132	5.7	82	1.7
10	PHSVDS(1st stage only)	117	0.6	91	0.2	185	2.5	122	1.7
10	JDSVD	342	3.1	287	1.4	320	15.7	364	10.5
10	SVDIFP	691	3.0	561	1.2	1179	29.1	1187	21.9

	$\delta =$ 1E-14 Matrix:	fidap4		jagmesh8		wang3		lshp3025	
	ILU Time:	0.1		0.1		2.8		0.1	
1	PHSVDS(GD+k)	62	0.6	52	0.1	102	1.8	66	0.5
1	PHSVDS(JDQMR)	64	0.3	55	0.1	106	1.1	68	0.5
1	JDSVD	78	1.5	45	0.3	67	3.0	79	1.2
1	SVDIFP	98	0.6	100	0.3	235	8.3	159	2.3
10	PHSVDS(GD+k)	210	1.3	163	0.5	306	5.5	209	3.0
10	PHSVDS(JDQMR)	251	1.2	215	0.5	402	5.0	265	3.5
10	JDSVD	573	5.5	408	1.9	518	14.6	606	26.9
10	SVDIFP	1152	5.4	981	1.9	1991	51.4	1897	29.1

Tables 2.4 and 2.5 show that a good preconditioner makes the problems tractable, with all three methods solving the problems effectively. Still, in most cases PHSVDS provides much faster convergence and execution time on both square and rectangular matrices. We see that when seeking one smallest singular value with high accuracy, JDSVD takes less iterations for one square matrix (wang3), and SVDIFP is competitive

Table 2.5: Seeking 1 and 10 smallest singular triplets with RIF, droptol = 1E-3. We report results from both PHSVDS variants: PHSVDS(GD+k) and PHSVDS(JDQMR). We report separately the time for generating the preconditioner and the time for running each method.

$\delta =$ 1E-8 Matrix:		fidap4		jagmesh8		deter4		plddb		lp_bnl2	
	RIF Time:	1.5		0.5		11.0		0.4		1.6	
k	Method	MV	Sec	MV	Sec	MV	Sec	MV	Sec	MV	Sec
1	PHSVDS	291	0.4	119	0.1	27	0.2	10	0.1	15	0.1
1	JDSVD	1729	8.9	1311	3.6	122	3.8	67	0.3	89	0.5
1	SVDIFP	513	1.4	622	0.9	142	2.1	29	0.1	49	0.1
10	PHSVDS	1224	1.8	307	0.5	405	2.4	52	0.1	74	0.1
10	JDSVD	8131	39.5	2356	5.7	–	–	–	–	–	–
10	SVDIFP	4359	10.3	3118	4.2	2278	45.4	390	1.0	453	1.0

$\delta =$ 1E-14 Matrix:		fidap4		jagmesh8		deter4		plddb		lp_bnl2	
	RIF Time:	1.5		0.5		11.0		0.4		1.6	
k	Method	MV	Sec	MV	Sec	MV	Sec	MV	Sec	MV	Sec
1	p(GD+k)	521	1.0	207	0.3	82	0.5	45	0.1	66	0.1
1	p(JDQMR)	544	1.0	229	0.2	87	0.4	45	0.1	69	0.1
1	JDSVD	2037	10.5	1410	3.6	188	5.5	122	0.5	134	0.6
1	SVDIFP	843	2.1	990	1.3	221	4.3	48	0.1	63	0.1
10	p(GD+k)	2074	3.2	562	0.8	769	2.5	152	0.5	192	0.5
10	p(JDQMR)	2604	4.9	641	0.9	877	5.3	247	0.5	242	0.5
10	JDSVD	–	–	12057	28.2	–	–	–	–	–	–
10	SVDIFP	7470	15.1	5024	6.1	3705	64.5	748	1.0	862	1.0

in two rectangular cases (plddb and lp_bnl2). This is because these cases require very few iterations, and the first stage of PHSVDS forces a Rayleigh-Ritz with 21 extra matrix-vector operations. This robust step is not necessary for this quality of preconditioning. If we are allowed to tune some of its parameters (as we did with JDSVD and SVDIFP) PHSVDS does require fewer iterations even in these cases.

2.5.4 With the shift-invert technique

We report results on seeking 10 smallest eigenvalues with PHSVDS, SVDIFP, `svds`, and `lansvd`. Because shift-invert turns an interior to a largest eigenvalue problem, PHSVDS does not need the second stage. `svds` uses the augmented matrix B for the shift-invert operator, while `lansvd` computes a QR factorization of A. For PHSVDS and SVDIFP we use two different factorizations, an LU and a QR factorization of A. All methods get the same basis size of 40, except for `lansvd` which is an unrestarted LBD code. Thus, `lansvd` represents an optimal method in terms of convergence, albeit expensive in terms of memory and computation per step. We disable the 'COLAMD' option in SVDIFP and `lansvd`, set the SVDIFP shifts to zero, and give a shift 1E-8 to `svds`. We have instrumented the `svds` code to return the number of iterations. To facilitate comparisons, we include the LU and QR factorization times in the running times of all methods, but also report them separately. The tolerance is δ = 1E-10.

Table 2.6 shows that PHSVDS is faster than `svds` both in convergence and execution time, partly because it works on C which is smaller in size and allows for faster convergence. Note that `svds` does not work well on rectangular matrices because B becomes singular and cannot be inverted, and if instead a small shift is used, it finds the zero eigenvalues of B first. SVDIFP's strategy to use an inverted operator as preconditioner does not seem to be as effective. PHSVDS seems to follow closely the optimal convergence of `lansvd`, although there is high variability in execution times. We believe this is a function not only of the cost of the iterative method but also of the different factorizations used by

Table 2.6: Seeking 10 smallest singular triplets using shift-invert. LU(A) and QR(A) are the times for LU and QR factorizations of A. The time of each method includes the associated factorization time.

δ = 1E-10	`fidap4`		`jagmesh8`		`deter4`		`plddb`		`lp_bnl2`	
Method	MV	Sec	MV	Sec	MV	Sec	MV	Sec	MV	Sec
LU(A) time	–	0.02	–	0.01	–	0.01	–	0.01	–	0.01
PHSVDS	31	0.10	26	0.07	167	14.4	47	0.28	35	1.01
SVDIFP	380	0.90	316	0.31	1177	168.4	418	1.92	432	9.3
QR(A) time	–	0.02	–	0.01	–	0.53	–	0.01	–	0.10
PHSVDS	31	0.29	26	0.08	166	9.1	27	0.08	36	0.48
SVDIFP	383	2.24	316	0.37	1177	55.4	418	0.55	432	3.13
`svds`	73	0.33	61	0.23	–	–	–	–	–	–
`lansvd`	31	0.37	26	0.24	133	3.3	28	0.23	34	0.43

the two algorithms.

2.5.5 On large scale problems

We use PHSVDS, SVDIFP, and JDSVD to compute the smallest singular triplet of matrices of order larger than 1 million. Information on these matrices appears in Table 2.7. We apply the two-stage PHSVDS on all test matrices except thermal2, which is solved with dynamic PHSVDS. The preconditioners are applied similar to our previous experiments with the exception that ILU uses `'thresh=0.1'`, and `'udiag=1'`. The tolerance is $\delta = 1E-12$. The symbol "*" means the method returns results that either did not satisfy the desired accuracy or did not converge to the smallest singular triplet.

Table 2.7: Basic information of some large scale matrices

Matrix	debr	cage14	thermal2	sls	Rucci1
rows m:	1048576	1505785	1228045	1748122	1977885
cols n:	1048576	1505785	1228045	62729	109900
nnz(A)	4194298	27130349	8580313	6804304	7791168
σ_1	1.11E-20	9.52E-2	1.61E-6	9.99E-1	1.04E-3
$\kappa(A)$	3.60E+20	1.01E+1	7.48E+6	1.30E+3	6.74E+3
Application	undirected graph	directed graph	thermal	Least Squares	Least Squares
Preconditioner	No	ILU(0)	ILU(1E-3)	RIF(1E-3)	RIF(1E-3)

Table 2.8 shows the results without or with various preconditioners. Debr is a numerically singular square matrix. PHSVDS is capable of resolving this more efficiently than JDSVD, while SVDIFP returns early when it detects that it is not likely to converge to the desired accuracy for left singular vector [80]. All methods easily solve problem cage14 with ILU(0), but PHSVDS is much faster. Thermal2 is an ill-conditioned matrix, whose preconditioner turns out to be less effective for C than for B. Therefore, SVDIFP has much slower convergence than JDSVD. Thanks to the dynamic scheme, PHSVDS recognizes this deficiency and converges without too many additional iterations, and with the same execution time as JDSVD. However, if we had prior knowledge about the preconditioner's performance, running only at the second stage gives almost exactly the same matrix-vectors as JDSVD and much lower time. Reducing further the overhead of the dynamic heuristic is part of our current research. JDSVD often fails to converge to the smallest singular value for rectangular matrices since it has difficulty to distinguish them from zero eigenvalues of B, as shown in the cases sls and Rucci1. For matrix sls, SVDIFP misconverges to the wrong singular triplet while PHSVDS is successful in finding the correct one. SVDIFP and PHSVDS have similar performance for solving problem Rucci1. In summary, PHSVDS is far more robust and more efficient than either of the other two methods for large problems.

Table 2.8: Seeking the smallest singular triplet for large scale problems. We report the time of each method including their running time and associated factorization time (PRtime) separately.

δ = 1E-12		PHSVDS			SVDIFP			JDSVD		
Matrix	PRtime	MV	Sec	RES	MV	Sec	RES	MV	Sec	RES
debr	—	539	84	3E-12	403*	246*	2E-1	1971	474.6	2E-12
cage14	2E+0	19	11	4E-13	33	28	6E-13	111	185	7E-14
thermal2	3E+3	419	506	7E-12	–	–	4E-9	309	535	4E-12
sls	3E+3	1779	170	1E-09	408*	328*	1E-9	–	–	2E-0
Rucci1	6E+4	4728	1087	7E-12	4649	6464	6E-12	–	–	5E-3

2.6 PRIMME_SVDS: A High-Performance Preconditioned SVD Software in PRIMME

Our goal is to provide a high quality, state-of-the-art SVD solver software that enables practitioners to solve a variety of large, sparse singular value problems with *unprecedented efficiency, robustness, and accuracy*. In this section we firstly describe the current state-of-the-art SVD software and then discuss our preconditioned two-stage meta-method proposed in [138] for effectively and accurately computing both largest and smallest singular values. Next we illustrate in detail the parallel implementation of PRIMME_SVDS as well as various characteristics of our high-performance SVD software.

2.6.1 Current SVD Software

Given the availability of several SVD algorithms, it is surprising that there is a lack of corresponding good quality software, especially with preconditioning. Table 2.9 summarizes the state-of-the-art dedicated SVD software. For each software, we show the methods it implements, its programming language, its parallel computing and preconditioning capabilities, whether it can obtain a fully accurate approximation efficiently, and the main interfaces to these libraries. The top four are high performance libraries, while the rest are MATLAB research codes.

SVDPACK [19] and PROPACK [77] implement variants of Lanczos or LBD methods. In addition, PROPACK implements an implicitly restarted LBD method. Both methods work well for computing a few largest, well separated singular values, which in most cases is an easy problem. The computation of smallest singular triplets is not supported in SVDPACK and it is not efficient in PROPACK which only implements the Rayleigh-Ritz projection [69]. In addition, neither library can use preconditioning or support message passing parallelism, although PROPACK does support shared memory multithreading. These are severe limitations for large-scale problems that need to run on supercomputers and that

often converge too slowly without preconditioning.

SLEPc offers an LBD method with thick restarting [57], which has similar algorithmic limitations to PROPACK for computing smallest singular values. In addition, this particular SVD solver cannot be directly used with preconditioning. However, the SLEPc LBD has an efficient parallel implementation.

Despite accuracy limitations, eigenvalue iterative methods based on C are widely used for computing the largest eigenpairs where the loss of accuracy is limited and even for low accuracy computations of the smallest singular values. For example, two popular packages in machine learning, scikit-learn[1] and Spark's library MLib[2], use a wrapper for the popular software ARPACK (implicit restarting Arnoldi method) [79]. However other solvers for standard Hermitian eigenvalue problems can be used. Table 2.10 lists the most widely used eigensolver libraries with high-performance computing implementations.

Our hybrid PHSVDS method can leverage these eigensolver libraries to solve the partial SVD problem in full accuracy. However, to optimize for efficiency and robustness several modifications and code additions are required that were not all available in previous eigensolvers. For example, Anasazi features a robust, high performance computing implementation but does not provide the near-optimal eigenmethods that are critical for fast convergence. SLEPc relies on PETSc for basic linear algebra kernels with support for various high-performance standards for shared and distributed memory machines and GPU. It also provides the appropriate preconditioned eigensolvers [58]. However, these are not tuned to deal with the high accuracy requirements of the first stage of PHSVDS or the need for refined projection methods for the highly interior problem in the second stage. PRIMME is designed to take advantage of all special properties of the Hermitian eigenvalue problem and therefore is a natural candidate that only required minor extensions, as described later in this paper. The goal is to produce a high quality, general purpose SVD software that can solve large-scale problems with high accuracy, using parallelism

[1] `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html`

[2] `https://spark.apache.org/docs/1.2.1/mllib-dimensionality-reduction.html`

Table 2.9: Dedicated SVD solver software for computing the partial SVD. The first four libraries have high performance implementations. The rest are MATLAB research codes. M, S, G stand for MPI, SMP and GPU, respectively. Fort, Mat, Py, and R stand for Fortran, Matlab, Python, and R programming languages, respectively.

HPC library	Method	Lang	Parallel	Precon.	Fast Full Acc.	Main Bindings
PRIMME	PHSVDS	C	M S	Y	Y	Fort, Mat, Py, R
PROPACK	IRLBD	Fort	S	N	N	Mat
SLEPc	TRLBD/KS	C	M S G	N	N	Fort, Mat, Py
SLEPc	JD/GD+k	C	M S G	Y	N	Fort, Mat, Py
SVDPACK	Lanczos	Fort	–	N	N	–
–	IRRHLB	Mat	S	N	Y	–
–	IRLBA	Mat	S	N	Y	–
–	JDSVD	Mat	S	Y	Y	–
–	SVDIFP	Mat	S	Y	Y	–

Table 2.10: Eigenvalue solver software available for computing partial SVD by solving an equivalent Hermitian eigenvalue problems on B and C. M, S, G stand for MPI, SMP and GPU, respectively. Fort, Mat, Py, R, and Jul stand for Fortran, Matlab, Python, R and Julia programming languages, respectively.

Software	Method	Lang	Parallel	Precon.	Main Bindings
Anasazi	KS/GD/LOBPCG	C++	M S G	Y	Py
(P)ARPACK	Arnoldi	Fort	M S	N	Mat, Py, R, Jul
BLOPEX	LOBPCG	C	M S G	Y	Mat
FEAST	CIRR	Fort	M S	Y	–
MAGMA	LOBPCG	C++	S G	Y	–
PRIMME	JD(QMR)/GD+k/LOBPCG	C	M S	Y	Fort, Mat, Py, R
Pysparse	JD	Py	S	Y	–
SciPy	LOBPCG	Py	S	Y	–
SLEPc	KD/JD/GD+k	C	M S G	Y	Fort, Mat, Py
SPRAL	Block	C	S G	N	Fort

and preconditioning, and with as close to a black-box interface as possible.

We mention that in some applications (for example in data mining) the required accuracy is so low or the rank of the matrix is so small that the use of the power method [81] or the block power method (see Randomized PCA in [52]) is sufficient. These methods cannot be considered general purpose SVD software and thus they are beyond the scope of this paper.

2.6.2 Method for Efficient and Accurate Computation

The PHSVDS approach [138] relies on eigensolvers that work on the equivalent eigenvalue formulations C and B, switching from one to the other to obtain the best performance. The PHSVDS method starts on C because without preconditioning the convergence in terms of iterations is much faster than that on B. Furthermore, the cost per iteration (computation of residual vectors, orthogonalization, etc.) on C is up to two times cheaper than on B (because of dimension n versus $n+m$). We refer to the computations on C as the *first stage* of the method. If further accuracy is required, the method switches to a *second stage* where the eigensolver is reconfigured to work on B, but with initial guesses from the first stage. We do not consider eigensolvers such as Lanczos that accept only one initial guess. Such methods would have to work for each needed singular triplet independently which is usually not as efficient as using all the good quality initial guesses from the first stage in one search space.

The singular value solver is configured to stop when the residual norm of the required singular triplets is less than the requested tolerance $\|A\|_2\delta_{\text{user}}$, or

$$\|\tilde{r}_i\|_2 = \sqrt{\|A\tilde{v}_i - \tilde{\sigma}_i\tilde{u}_i\|_2^2 + \|A^T\tilde{u}_i - \tilde{\sigma}_i\tilde{v}_i\|_2^2} < \|A\|_2\delta_{\text{user}}. \tag{2.14}$$

Let $(\tilde{\lambda}_i^C, \tilde{x}_i^C)$ be the eigenpair approximation from the eigensolver on C. Considering that $\tilde{\sigma}_i$ will be set as $\|A\tilde{x}_i^C\|_2$, $\tilde{v}_i$ as $\tilde{x}_i^C$ and $\tilde{u}_i$ as $A\tilde{x}_i^C\tilde{\sigma}_i^{-1}$, the above is translated into a convergence criterion for the eigensolver on C as

$$\|\tilde{r}_i^C\| = \|C\tilde{x}_i^C - \tilde{\lambda}_i^C\tilde{x}_i^C\|_2 < \sqrt{|\tilde{\lambda}_i^C|\|C\|_2}\delta_{\text{user}}.$$

The eigensolver returns when all requested triplets satisfy the convergence criterion. However the eigensolver may reach its maximum achievable accuracy before the residual norm reduces below the above convergence tolerance. Setting this limit properly is a critical point in the robustness and efficiency of the method. If the tolerance is set below this

limit the eigensolver may stagnate. If the limit is overestimated, then the iterations of the second stage will increase, making the whole solver more expensive. Selecting this limit depends on the numerical properties of the eigensolver, so we discuss it in Section 2.6.3.

In the second stage, the vectors $[\tilde{v}_i; \tilde{u}_i]$ are set as initial guesses of the eigensolver. The convergence criterion directly checks (2.14) with $\tilde{\sigma}_i$ set as $|\tilde{\lambda}_i^B|$ and $\tilde{v}_i$ and $\tilde{u}_i$ set as the normalized subvectors $\tilde{x}_i^B(1:n)$ and $\tilde{x}_i^B(n+1:n+m)$. Because this computation requires extra reduction operations, it is only checked after an eigenvalue residual condition is satisfied,

$$\|\tilde{r}_i^B\| = \|B\tilde{x}_i^B - \tilde{\lambda}_i^B \tilde{x}_i^B\|_2 \approx \sqrt{2}\|\tilde{r}_i\| < \sqrt{2}\|B\|_2 \delta_{\text{user}}. \tag{2.15}$$

The above is derived by assuming that $\|\tilde{x}_i^B(1:n)\|_2 \approx \|\tilde{x}_i^B(n+1:n+m)\|_2$. If the eigenvector $\tilde{x}_i^B$ corresponds to a singular triplet, then this assumption is satisfied near convergence. It is possible, however, that after (2.15) is satisfied to have $\|\tilde{x}_i^B(1:n)\|_2 \ll \|\tilde{x}_i^B(n+1:n+m)\|_2$. This is the case when $\tilde{x}_i^B$ has large components in the $(m-n)$-dimensional null space of B, and therefore it does not correspond to a singular triplet. Checking our second level criterion (2.14) avoids this problem. We have observed this situation when computing the smallest singular values in problems with condition number larger than 10^8.

If the smallest singular values are wanted, the eigensolver on B must compute interior eigenvalues. Then, it is important that we specify where the eigensolver should look for them. First, we want only the non-negative eigenvalues of B. Second, if $m \neq n$ and the required singular values are very small, asking for the eigenvalues closest to zero will make the eigensolver to keep trying to converge on the unwanted null space. Therefore, we should only try to find eigenvalues on the right of some positive number. Third, because this number should be a lower bound to the singular value we seek, we can use the perturbation bounds of the approximations of the first stage. Specifically, we know that $\sigma_i \in [\tilde{\sigma}_i - \|\tilde{r}_i^C\|\tilde{\sigma}_i^{-1}\sqrt{2}, \tilde{\sigma}_i]$, where $\tilde{\sigma}_i = (\tilde{\lambda}_i^C)^{\frac{1}{2}}$ [138]. Because the augmented approach cannot distinguish eigenvalues smaller than $\|A\|$mach, we configure the eigensolver to

find the smallest eigenvalue σ_i that is greater than max$(\tilde{\sigma}_i - \|\tilde{r}_i^C\|\tilde{\sigma}_i^{-1}\sqrt{2}, \|A\|\text{mach})$. This heuristic is also used in [60].

For interior eigenproblems, alternatives to the Rayleigh-Ritz extraction are recommended, such as the harmonic or the refined variants. As described in the next section, we employ a variant of the refined extraction. For each eigenvalue, the shift for the refined extraction is the corresponding singular value lower bound. In PRIMME, these lower bounds are also used as shifts in the Jacobi-Davidson correction equation. If the shifts are close enough to the exact eigenvalues, they accelerate the convergence of Jacobi-Davidson.

Algorithm 4 shows the specific functionality needed for the two stages of PHSVDS.

2.6.3 Descriptions of changes in PRIMME

To support PRIMME_SVDS, we have implemented many enhancements to PRIMME, including a user defined convergence criterion, improved numerical quality of converged eigenvectors, improved robustness to achieve convergence near machine precision, a simplified refined projection method, a different locking scheme for interior eigenvalues, a new scheme for initializing the search space, and finally a new two-stage meta-method interface.

To achieve the required accuracy at the first stage, we have to adjust the convergence tolerance at every step based on the value of the current eigenvalue. This was not possible in the original PRIMME implementation in which δ_{user} was set as a user input parameter. In the new version, we have added a function pointer into PRIMME main data structure to allow the user to provide their own convergence test function. Our top level SVD interface provides line 2 of Algorithm 4 as the default convergence test function.

When PRIMME uses soft locking (i.e., no locking), before exiting it now performs an additional Rayleigh-Ritz on the converged Ritz vectors to adjust the angles of the desired Ritz vectors. The resulting Ritz vectors have improved quality that has proved helpful in

Algorithm 4 PHSVDS: a preconditioned hybrid two-stage method for SVD

Input: matrix-vector products $A\,x$ and $A^T\,x$, preconditioner function, global summation reduction, number of singular triplets seeking k, tolerance δ_{user}
Output: Converged desired singular triplets $\{\tilde{\sigma}_i, \tilde{u}_i, \tilde{v}_i\},\ i = 1, \ldots, k$

First-stage on C***:***

Set eigensolver matrix-vector as $C = A^T A$ or AA^T
Set eigensolver convergence criterion as $\|\tilde{r}_i\|_2 \leq \max(\sqrt{|\tilde{\lambda}_i^C|}\|C\|_2\delta_{\text{user}}, \text{mach}\|C\|_2)$
Run eigensolver seeking largest/smallest eigenvalues of C
Perform Rayleigh-Ritz on the returned vector basis
Set $\tilde{\sigma}_i = |\tilde{\lambda}_i^C|^{\frac{1}{2}}$, $\tilde{v}_i = \tilde{x}_i^C$ and $\tilde{u}_i = A\tilde{v}_i\tilde{\sigma}_i^{-1}$
if all triplets converged with tolerance $\|A\|_2\delta_{\text{user}}$ **then**
    Return $\{\tilde{\sigma}_i, \tilde{u}_i, \tilde{v}_i\}$, for $i = 1, \ldots, k$
end if

Second-stage on B***:***

Set eigensolver initial guesses as $\sqrt{2}[V; U]$
if finding the largest singular values **then**
    Set eigensolver extraction method as *standard Rayleigh-Ritz*
    Set eigensolver to find the largest eigenvalues
else
    Set eigensolver extraction method as *simplified refined projection*
    Set eigensolver to find the eigenvalues closest to but greater than
$$\max(\tilde{\sigma}_i - \|\tilde{r}_i^C\|_2\tilde{\sigma}_i^{-1}\sqrt{2}, \|A\|_2\text{mach}).$$
end if
Set eigensolver convergence criterion as $\|\tilde{r}_i\|_2 \leq \sqrt{2}\|B\|_2\delta_{\text{user}}$
Run eigensolver on B
Set $\tilde{\sigma}_i = |\tilde{\lambda}_i^B|$, $\tilde{x}_i^B = [\tilde{v}_i; \tilde{u}_i]$, normalize $\tilde{u}_i$ and $\tilde{v}_i$
Return $\{\tilde{\sigma}_i, \tilde{u}_i, \tilde{v}_i\}$, for $i = 1, \ldots, k$

the second stage [138]. This is mentioned in Line 4 of the algorithm.

As an iterative method based on matrix-vector multiplications by C, the maximum accuracy that the eigensolver can obtain should be close to $\|C\|_2$mach. We observed that in slow converging cases PRIMME eigensolvers may stagnate when the residual norm is still 10 or 100 times above that limit. A brief analysis reveals that the major propagation of error occurs when restarting the search space V and the precomputed $W = AV$ as Vy and Wy respectively. Despite $\|V\|_2 = 1 = \|y\|_2$, the operation occurs a number of times equal to the number of restarts, and thus the expected accumulated error increases by a factor of $\sqrt{\text{restarts}}$. This factor is more problematic for W where the accumulated error becomes $\sqrt{\text{restarts}}\|C\|_2$mach, thus preventing the residual to converge to full accuracy. This was also confirmed experimantally. Our solution was to reset both matrices, by fully reorthogonalizing V and computing $W = AV$ directly, when $\|\tilde{r}_i^C\| < \sqrt{\text{restarts}}\|C\|_2$mach, where restarts is the number of restarts since last resetting. This change has returned the stagnation level to less than $10\|C\|_2$mach facilitating a very accurate solution at the first stage.

To address the interior eigenproblem of the second stage, we have implemented a refined extraction procedure in PRIMME. The refined procedure computes an eigenvector approximation $\tilde{x}_i$ in the span of V that minimizes the norm of the residual $\|(B - \tau I)\tilde{x}_i\|/\|\tilde{x}_i\|$. In general, τ should be as close as possible to the eigenvalue so most implementations set it as the Ritz or harmonic Ritz value from the current search space at every iteration [65, 60, 91]. The minimization requires the QR factorization of the tall skinny matrix $BV - \tau V$, for a step cost of $O((m+n)k^2)$ flops and $O(k)$ global reductions per iteration, where $k = \dim(V)$. In our case, the $\tilde{\sigma}_i$ from the first stage are very good eigenvalue approximations (if $\kappa_i < 10^8$) so there is little gain to updating the shift at every iteration. This leads to a simplified and much more efficient implementation of the refined procedure. With constant τ, the cost of updating the QR factorization at every iteration is $O((m+n)k)$, the same as the cost of orthogonalization. Also, Q and R can be restarted without communications; if V is restarted as VY, then we compute the QR factorization

$RY = \tilde{Q}\tilde{R}$ and restart Q as $Q\tilde{Q}$ and R as $\tilde{R}$. The factorization of R involves a matrix of small dimension and can be replicated in every process.

In the second stage we force the PRIMME eigensolver to use locking. The earlier version of PRIMME locked converged eigenvectors only at restart, allowing them to improve for a few more steps in the basis. The new version of PRIMME changes this for interior problems only. When an eigenvector converges we force a restart and lock it out. This improves robustness with the Rayleigh-Ritz method since converged interior eigenvalues may become unconverged causing the method to misconverge. The refined extraction avoids this problem but it still benefits from the new change. When an eigenvector is locked, the QR factorization for the refined extraction is recomputed with the new target shift.

In PRIMME the search space is initialized as a block Krylov subspace starting from any available initial guesses or random vectors. We have extended the library's setup options to allow for finer user control on the initialization step. Among other options, the user can now deactivate the Krylov subspace. We have found this to be helpful because of the good quality initial guesses in the second stage.

2.6.4 High performance characteristics of PRIMME_SVDS

Our library extension inherits the design philosophy of PRIMME with respect to performance. This is summarized below and its effects on performance in Table 2.11.

- The user must provide as function pointers the matrix-vector product and, optionally, the preconditioner application.
- PRIMME's implementation works for both parallel and sequential runs. It follows the SPMD parallelization model, so if the user has distributed the matrix by rows onto processes, each process in PRIMME will hold the corresponding local rows of the vectors. Small objects are replicated across processes. If the code is used in paral-

Table 2.11: The parallel characteristics of PRIMME_SVDS operations in PRIMME.

Operations	Kernels or Libs	Cost per Iteration	Scalability
Dense algebra: MV, MM, Inner Prods, Scale	BLAS (eg, MKL, ESSL, OpenBlas, ACML)	O((m+n)*k)	Good
Sparse algebra: SpMV, SpMM, Preconditioner	User defined (eg, PETSc, Trilinos, HYPRE, librsb)	O(1) calls	Application dependent
Global reduction	User defined (eg, MPI_AllReduce)	O(1) calls of size O(k)	Machine dependent

lel, in addition to parallel implementations of the matrix-vector and preconditioning operators, the user must also provide an operator for the global sum reduction.

- PRIMME relies on third-party BLAS and LAPACK libraries to achieve single and multi-threaded performance of operations with dense matrices and vectors within each SPMD process.

- The required workspace is allocated internally or may be provided by the user as a block of memory.

Some libraries, such as SLEPc and Anasazi, use an object oriented abstraction for matrices and vectors, thus externalizing the control over the actual memory and the operations on it. SLEPc is based on the structures of PETSc and Anasazi defines its own structures with templates. The goal is to facilitate optimizations on different memory hierarchies and heterogeneous architectures such as accelerators. However, this design may increase overhead and induce unnecessary memory copies to conform with the given abstraction.

PRIMME handles the vectors directly as a block of memory, which may allow for fewer memory copies when the matrix-vector product and preconditioning operators receive and return the corresponding local part of the vectors, but also in other places in the code. Moreover, it displays better locality for cache performance as long as PRIMME orders its functionality appropriately. All numerical operations are then performed through calls to optimized BLAS and LAPACK which are compatible with this memory model. The disadvantage is that with vectors residing in main memory, calls to accelerator enhanced

BLAS/LAPACK libraries have to transfer their arguments during every call. PRIMME tries to use the highest level BLAS when this is beneficial, e.g., the use of level 3 BLAS when the block size is greater than one.

Following the SPMD model for parallel programming, only the largest data structures in PRIMME are distributed; specifically the eigenvectors to be returned, the vectors in the search space V, the auxiliary vectors $W = AV$, and, when refined extraction is used, the array Q that holds the orthogonal matrix of the QR factorization of $W - \tau V$. The cost of small matrix operations (such as solving the small projected eigenvalue problem) is negligible and the operation is duplicated across all processes. Long vector updates are performed locally with no communication. Inner products involve a global sum reduction which is the only communication primitive required and is provided by the user. To reduce latency, PRIMME blocks as many reductions together as possible without compromising numerical stability.

Most, but not all, applications use PRIMME with the MPI framework that nowadays can be used effectively even on shared memory machines. This avoids the need to store the matrix or the preconditioner on each core. Similarly, pure shared memory parallelism or optimized sequential execution is obtained by simply linking to the appropriate libraries.

The user-provided matrix-vector and preconditioning operators must be able to perform operations with both the matrix and its transpose and implement the desired parallel distribution in each case. Note that the parallel behavior of the two stages might be very different for rectangular matrices where $n \ll m$. The PRIMME_SVDS interface also allows the user to pass functions for performing matrix vector and preconditioning operations with B and C directly. This is useful as there are many optimizations that a user can perform to optimize these operations (especially on B, see [114]). This feature is not available in other software.

2.6.5 Interfaces of PRIMME_SVDS

PRIMME_SVDS is part of PRIMME's distribution with a native C interface. We provide several driver programs with calling examples as part of the documentation, ranging from a simple sequential version with a basic matrix-vector multiplication, to a fully parallel and preconditioned version with such functionality provided by the PETSc library. In addition, we offer interfaces to Matlab, Python, and R that can be used easily by both ordinary and advanced users to integrate with domain specific codes or simply experiment in an interactive environment. These interfaces expose the full functionality of PRIMME_SVDS which, depending on the supporting libraries, can include parallelism and preconditioning.

Next, we describe the most important functionality and features of PRIMME_SVDS using the C interface. Other interfaces are wrappers that call the C interface. The problem parameters and the specific method configuration are set in a C structure called `primme_svds_params`. The most important parameters are the following.

- `m` and `n`: the number of rows and columns of the problem matrix
- `matrixMatvec(x, ldx, y, ldy, blockSize, transpose, primme_svds)`: function pointer to the matrix-vector product with A. The result of the product is stored in y. If `transpose` is zero, the function should compute Ax, otherwise A^*y. x and y are matrices with `blockSize` columns and leading dimensions `ldx` and `ldy` respectively. `primme_svds` is included to provide access to all `primme_svds_params` parameters.
- `matrix`: (optional) pointer to user data for `matrixMatvec`.
- `numSVals`: the number of desired singular triplets to find.
- `target`: select which singular values to find: the smallest, the largest or the closest to some value (not discussed in this paper).
- `eps`: the desired accuracy for wanted singular triplets, see (2.14).

To run in parallel SPMD mode, the matrix-vector operator must be parallel, every process should have the same values for `eps`, `numProcs`, and `numSVals`, and the following parameters much be set.

- `numProcs`: the number of MPI processes (must be greater than 1).
- `procID`: the rank of the local process (e.g., the MPI rank).
- `mLocal` and `nLocal`: the number of rows and columns local to this process. In the parallel `matrixMatvec`, x and y address the corresponding local parts, with x having `nLocal` rows and y having `mLocal` rows.
- `globalSumDouble`: function pointer to the global sum reduction.

These parallel environment parameters are also required in other frameworks such as PETSc (see Sec 3.1 in PETSc user manual [14]) and Tpetra (see class CsrMatrix [13]). In shared memory environments, the user may choose to run a single process and link to a threaded BLAS library such as OpenBLAS [146].

The following optional parameters may be used to accelerate convergence.

- `applyPreconditioner(x, ldx, y, ldy, blockSize, mode, primme_svds)`: function pointer to the preconditioning; the function applies the preconditioner to x and stores it into y. `mode` indicates which operator is the preconditioner for: A^TA (`primme_svds_op_AtA`), AA^T (`primme_svds_op_AAt`) or $[0\ A^T; A\ 0]$ (`primme_svds_op_augmented`).
- `preconditioner`: pointer to user data for `applyPreconditioner`.
- `maxBasisSize`: the maximum number of columns in the search space basis.
- `maxBlockSize`: the maximum number of approximate eigenpairs to be corrected at every iteration. Larger block size may be helpful for multiple or highly clustered singular values and usually improves cache and communication performance.

- `primme`: PRIMME parameter structure for first stage.
- `primmeStage2`: PRIMME parameter structure for second stage.

The default maximum basis dimension (`maxBasisSize`) for the eigensolvers is 15 and the dimension after restarting (`minRestartSize`) is 6 if finding less than 10 largest singular values. Otherwise restarting parameters are set to 35 and 14 respectively. The default block size is 1.

All `primme_svds_params` parameters can be modified by the user. Furthermore the user can tune individual eigensolver parameters in `primme` and `primmeStage2` for each stage respectively. Currently, the default method for the first stage is the DYNAMIC method of PRIMME which switches dynamically between GD+k and JDQMR attempting to minimize time. The second stage defaults to the JDQMR method. Users can change the default eigensolver methods by calling,

```
primme_svds_set_method(svds_method, method_stage1, method_stage2,
                       primme_svds);
```

`method_stage1` and `method_stage2` can be any PRIMME preset method. With this function the user can also change the PHSVDS to a different SVD method, for example to perform only a single stage with the normal equations (`primme_svds_normalequations`) or the augmented approach (`primme_svds_augmented`). Future versions may include other methods such as LBD and JDSVD.

Other advanced features include the following.

- `initSize`: the number of singular vectors provided as initial guesses.
- `targetShifts`: contains the values closest to which we should find singular values. Only accessed if `target` has been set to find interior singular values.
- `numTargetShifts`: the number of values in `targetShifts`.
- `numOrthoConst`: number of singular vectors provided as external orthogonalization constraints (see explanation below).

- `printLevel`: specifies level for printing out information (0–5).
- `outputFile`: the output file descriptor.
- `stats`: the performance report of this run.

After specifying the required and optional fields in the structure, we can call the main function:

```
primme_svds(svals,svecs,rnorms,primme_svds_params)
```

The argument `svals` and `rnorms` are arrays at least of size `numSvals` to store the computed singular values and the residual norms, computed as in (2.14). Both arrays are filled by all processes. The argument `svecs` is a dense matrix at least of dimension $(\texttt{mLocal} + \texttt{nLocal}) \times (\texttt{numSVals} + \texttt{numOrthoConst})$. On input the first `numOrthoConst` columns of size `mLocal` are the left constrain vectors, followed by the next `initSize` initial left singular vector guesses. The following vectors are of size `nLocal` with the same configuration corresponding to the constraints and the right singular vectors. On output, the `numOrthoConst` left and right constrain vectors remain unchanged in their `svecs` position. They are followed respectively by the `initSize` left and right converged singular vectors. `initSize` has been updated with the number of converged singular triplets.

Figure 2.8 illustrates the main parts of a simple example in C that computes the four smallest singular values of a rectangular matrix.

Finally, we briefly discuss the MATLAB interface which is a MEX wrapper of PRIMME_SVDS. It is similar to MATLAB's `svds`, allowing it to be called by non-expert users but also by experts that can adjust over 30 parameters.

```
primme_svds(A)
primme_svds(A, numSvds)
primme_svds(A, numSvds, target)
primme_svds(A, numSvds, target, opts)
primme_svds(A, numSvds, target, opts, precond)
primme_svds(Afun, M, N,...)
```

```
#include "primme.h" /* header file for PRIMME SVDS too */
double *svals; /* Array with the computed singular values */
double *rnorms; /* Array with the computed residual norms */
double *svecs; /* Array with the computed singular vectors;
                  i-th left (u) vector starts in svecs[primme_svd.m*i],
                  i-th right (v) vector starts in
                  svecs[primme_svd.m*primme_svds.initSize +
                  primme_svds.n*i] */
primme_svds_params primme_svds;
                  /* PRIMME SVDS configuration struct */

/* Set default values in PRIMME SVDS configuration struct */
primme_svds_initialize(&primme_svds);

/* Set the function that implements A*x and A^t*x */
primme_svds.matrixMatvec = MatrixMatvecSVD;

/* Set problem parameters */
primme_svds.m = 1000;
primme_svds.n = 100;         /* set problem dimension */
primme_svds.numSvals = 4;    /* Number of singular values wanted */
primme_svds.eps = 1e-12;     /* ||r|| <= eps * ||matrix|| */
primme_svds.target = primme_svds_smallest;
                 /* Seeking for the smallest singular values */

/* Allocate space for converged Ritz values and residual norms */
svals = (double *)malloc(primme_svds.numSvals*sizeof(double));
svecs = (double *)malloc((primme_svds.n+primme_svds.m)
                              *primme_svds.numSvals*sizeof(double));
rnorms = (double *)malloc(primme_svds.numSvals*sizeof(double));

dprimme_svds(svals, svecs, rnorms, &primme_svds);
```

Figure 2.8: Simple sequential example code that computes the four smallest singular values of a rectangle matrix of dimensions 1000×100 with PRIMME. The matrix-vector multiplication code and some details have been omitted. The full version can be found at `exsvds_dseq.c` under the folder `TEST`.

Like `svds`, users only need to provide the matrix A while PRIMME_SVDS sets a list of expert defaults underneath. Users can tackle more advanced tasks incrementally by specifying more parameters. For example, they can pass their own matrix-vector and preconditioning operations, or they can simply take advantage of MATLAB's built-in matrix times block-of-vectors operators and preconditioners. Interfaces to the other scripting languages are developed similarly.

2.7 Numerical Experiments

In this section we report numerical results in order to access the performance of PRIMME_SVDS software. To conduct a comprehensive performance analysis, we have performed extensive experiments by evaluating different functionality of our SVD software such as largest or smallest singular triplets, square or rectangular matrices, with preconditioning or not, different application problems, and different supercomputer or clusters.

For our performance test we consider seeking small number of smallest and largest singular triplets from various real applications including ill-conditioned least-squares problem, DNA electrophoresis, linear programming, and graph clustering. Among them, three of matrices are rectangular while others are square. The size of test matrices ranges from the order of 1E+6 to the order of 1E+7. To the best of our knowledge, we report for the first time the experimental results on these large-scale size problems using high-performance SVD software. The condition number of these matrices also varies from 1E+1 to 1E+16 which indicates the achievable accuracy and the hardness of the problems for seeking smallest singular values. Basic information of matrices and parameter settings are listed in Table 2.12.

Table 2.12: Properties of some real-world test matrices. L/S stands for seeking largest/smallest singular values.

					gaps	
Matrix	rows m	cols n	nnz(A)	$\kappa(A)$	larg.	small.
atmosmodl	1,489,752	1,489,752	10,319,760	1.14E+3	5E-5	5E-5
Rucci1	1,977,885	109,900	7,791,168	6.74E+3	3E-3	5E-5
LargeRegFile	2,111,154	801,374	4,944,201	1.1E+4	1.2	3E-7
cont1_l	1,918,399	1,921,596	7,031,999	∞	6E-6	–
cage15	5,154,859	5,154,859	99,199,551	1.19E+1	6E-4	1E-3
sls	1,748,122	62,729	6,804,304	1.3E+3	4E-2	8E-7
relat9	12,360,060	549,336	7,791,168	∞	3E-3	–
delaunay_n24	16,777,216	16,777,216	50,331,601	∞		–
Laplacian	8,000p	8,000p	55,760p	$\approx$8.5E+2		

We compare against several SVD methods that are implemented in SLEPc [58] and PROPACK [77, 78]: Jacobi-Davidson (JD) on C, Generalized Davidson (GD) on C, Krylov Schur (KS) on C, and thick-restart LBD in SLEPc, and implicit restarted LBD in PROPACK.

First, we compared PRIMME_SVDS with SLEPc that leverages an eigensolver on C for solving SVD problems with low accuracy when increasing number of MPI processes. We demonstrate the advantages of our PHSVDS method over simple eigensolver on C. Then we compute a few of the largest or smallest singular triplets on both square and rectangular matrices with low and high user accuracy without a preconditioner using fixed 48 MPI processes. We show consistently better performance of PHSVDS implemented in PRIMME_SVDS compared with LBD implemented in SLEPc when the hardness of problems increases. Second, we compare PRIMME_SVDS with PROPACK and SLEPc when seeking a few of largest and smallest singular triplets on a shared memory system (using OpenMP). Finally, we present some numerical results to demonstrate good parallel scalability of our SVD software on some very large-scale real-world problem under strong and weak scaling.

All computations are carried out on the NERSC's Edison and a cluster SciClone at college of William and Mary. In Edison each compute node has two Intel "Ivy Bridge" processors at 2.4 GHz for a total of 12 cores and 64 GB of memory, interconnected with high-speed Cray Aries in Dragonfly topology. In SciClone, we use 36 Dell PowerEdge R415 servers with AMD Opteron processors at 3.1 GHz for a total of 12 cores and 32 GB of memory running the Red Hat Enterprise Linux operating system, interconnected by a FDR InfiniBand communication network. All tests are conducted with machine precision $\epsilon = 2.2 \times 10^{-16}$. PRIMME is linked to the BLAS and LAPACK libraries and PESTc library [14] in a distributed memory system while to the OpenBLAS library [146] and the LIBRSB library [89] in a shared memory system. Our stopping criterion is for the left and right residuals to satisfy,

$$\sqrt{\|A\tilde{v}_i - \tilde{\sigma}_i\tilde{u}_i\|_2^2 + \|A^T\tilde{u}_i - \tilde{\sigma}_i\tilde{v}_i\|_2^2} < \|A\|_2\delta_{user}. \tag{2.16}$$

All methods start with the same initial guess, `rand(min(m,n),1)` with fixed random seed. For PRIMME_SVDS, we set `maxBasisSize=15` and most of experiments with two

δ tolerances, 1e-6 and 1e-12. For δ =1e-6, PRIMME_SVDS does not need to enter the second stage for any of our tests. For all methods in SLEPc and PROPACK, the same maximum basis size 15 is set. PROPACK often has difficulty to seek the smallest singular values without using shift and invert techniques so we report the best obtained results. The tables report as "MV" the number of products with A and A^T. "Sec" is the run time in seconds, and "–" means the method cannot converge to all desired singular values or that the code breaks down. For the first stage of PRIMME_SVDS we use either GD_Olsen_PlusK or JDQMR methods. For the second stage, we run experiments with default JDQMR method for interior eigenvalue problems. Although all software implementation are written in different programming languages, these differences are subtle since they are all implemented in efficient scientific computing languages (such as C, C++ and FORTRAN). Therefore, we compare not only execution times but also the number of matrix-vector operations as the primary measurement of the performance.

2.7.1 Comparison with SLEPc LBD on a distributed memory system

We investigate the performance of various methods implemented in different SVD software in a distributed memory system. The sparse matrix-vector operations are performed using PETSc in both PRIMME and SLEPc.

We first compare two variants of PHSVDS with thick-restart LBD in SLEPc on both square and rectangular matrices without preconditioning. It is important to examine the effectiveness of a method in this scenario since a good preconditioner is usually not easy to obtain for SVD problems. We compute 5 largest and smallest singular triplets using 48 MPI processes on three selected square and two rectangular matrices as shown in Table 2.12. Among them, the matrices cage15 and atmosmodl have relative larger gap ratios and smaller condition number, and thereby are easy ones. Matrices Rucci1 and LargeRegFile are hard cases, and matrix cont1_l is a very hard one. We expect all software to perform well for solving easy problems. But harder problems tend to magnify the

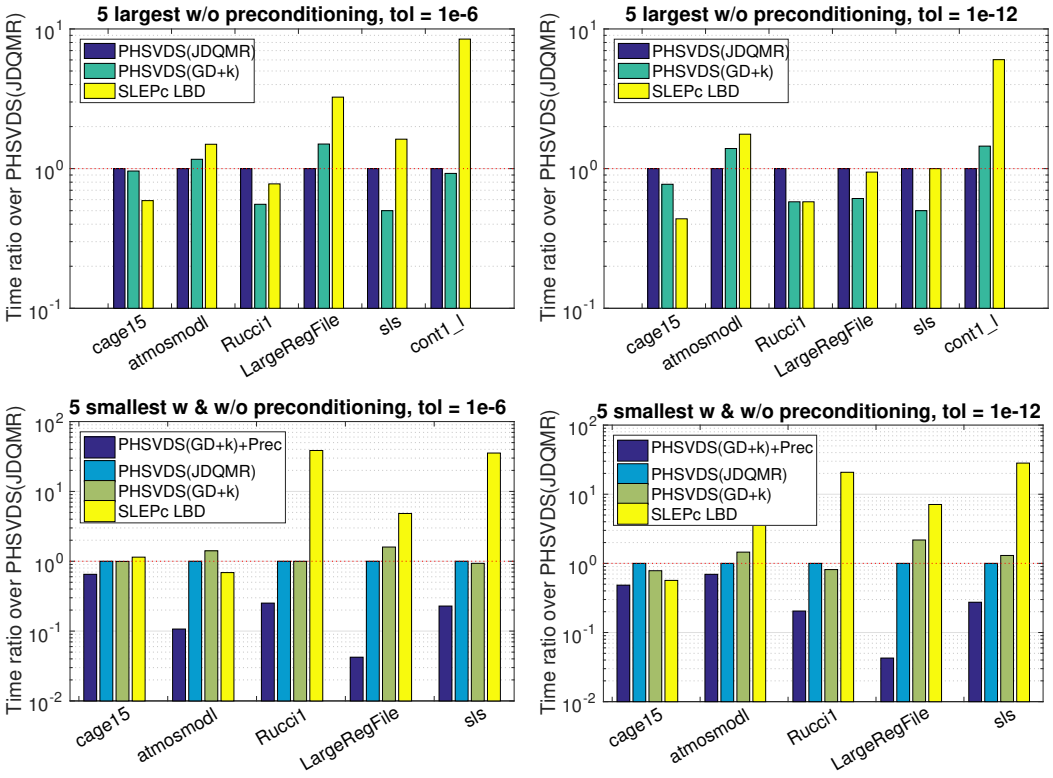


Figure 2.9: Time ratio over PHSVDS(JDMQR) when computing 5 largest and smallest singular values with user tolerance 10^{-6} and 10^{-12} without and with preconditioning using 48 MPI processes in distributed memory. The sparse matrix-vector operations are performed using PETSc. For seeking smallest singular values on matrix cont1_l, all methods fail to converge except PHSVDS(JDMQR).

difference between methods.

Figures 2.9 and Table 2.13 show that PHSVDS variants converge faster and more robustly than all other methods on both square and rectangular matrices. For instance, even for largest singular problems, LBD performs as well as PHSVDS for relatively easy problems but starts to converge slowly for hard problem. This is because significant more iterations are needed to achieve the desired accuracy of singular triplets when the condition number of the target singular values become larger. In addition, we illustrate that PRIMME_SVDS can fully take advantage of preconditioning, which become more and more important especially for hard smallest singular value problems as we have seen in Table 2.13.

Specifically, Figure 2.9 shows that for seeking smallest singular values PHSVDS are significantly faster than LBD when the problems become harder. For instance, PHSVDS is often one order of magnitude faster than LBD when solving hard problems with relative low accuracy. It indicates that LBD is often inefficient or even fail to converge to the desired singular values of hard problems. When solving easy problems, still PHSVDS is competitive with LBD when seeking 5 singular values. The superiority of PHSVDS is a result of using a near-optimal eigenmethod and a hybrid two-stage SVD method in PRIMME.

Figure 2.9 also shows the performance comparison when computing 5 smallest singular values using a preconditioner. For cage15 and atmosmodl, a multigrid preconditioner is constructed using HYPRE library while for LargRegFile, Rucci1and sls a simple yet easily constructed block-Jacobi preconditioner is built using the block-diagonal of C matrix. It is not suprised that it is not easy to obtain a good preconditioner that can significantly accelerate the convergence and runtime of a SVD problem. However, it is important that a state-of-the-art SVD solver can fully take advantage of a good preconditioner if it is available. For instance, for Rucci1 matrix, the convergence is significantly improved and thereby the runtime performance. In practice, PRIMME_SVDS provides full flexibility for users to operate on their preconditioning function that could be designed and constructed for A and A^T or directly for C and B.

2.7.2 Comparison with PROPACK on a shared memory system

We compare the performance of PRIMME_SVDS with PROPACK on a shared memory system. Both solvers are provided with the multithreaded sparse matrix-vector operation in librsb [89] and linked with a threaded version of BLAS, the AMD Core Math Library (ACML). We use a single node of SciClone with a total of 12 threads.

Table 2.14 shows the results in terms of matrix-vector products and time. For PRIMME_SVDS the results are qualitatively similar to the previous numerical experi-

Table 2.13: Seeking 5 largest and smallest singular triplets with user tolerance 1e-6 and 1e-12 without preconditioning. We report both runtime and number of matrix-vector operations from both PHSVDS variants: PHSVDS(GD+k) and PHSVDS(JDQMR) in PRIMME_SVDS and LBD in SLEPc.

	P(GD+k)		P(JDQMR)		LBD		Prec	
Matrix	MV	Sec	MV	Sec	MV	Sec	MV	Sec
the 5 largest singular values with tolerance 1E-6								
cage15	1069	81.7	1379	85.9	**744**	**48.8**		
atmosmodl	**2077**	25.3	2161	**17.7**	2264	26.0		
Rucci1	285	**1.3**	463	2.0	**184**	1.4		
LargeRegFile	109	**0.9**	167	1.2	**84**	1.3		
sls	105	**0.8**	189	1.3	**78**	1.3		
cont1_l	**1521**	**21.1**	2595	25.6	12408	201.4		
the 5 largest singular values with tolerance 1E-12								
cage15	2393	183.4	3559	219.0	**1272**	**94.2**		
atmosmodl	**6453**	78.8	7711	**61.8**	8824	107.8		
Rucci1	577	2.6	945	3.9	**264**	**2.2**		
LargeRegFile	157	**1.3**	301	2.0	**98**	1.7		
sls	177	**1.3**	357	2.4	**88**	1.4		
cont1_l	15371	220.9	**12203**	**111.4**	35864	654.6		
the 5 smallest singular values with tolerance 1E-6								
cage15	**1247**	**70.8**	1411	87.8	874	92.4	*375*	*52.5*
atmosmodl	88459	1081.4	70873	569.2	**21104**	**392.2**	*187*	*61.0*
Rucci1	125067	520.7	**91123**	**380.9**	1118532	14914.2	*11947*	*97.4*
LargeRegFile	19391	160.5	**12837**	**83.2**	15056	411.2	*363*	*3.6*
sls	31551	211.2	**22043**	**165.0**	100956	3568.6	*2755*	*23.0*
the 5 smallest singular values with tolerance 1E-12								
cage15	2691	203.4	3419	210.7	**1124**	**119.1**	*711*	*101.7*
atmosmodl	276305	3396.9	**211471**	**1691.8**	753886	14176.2	*3205*	*1190.6*
Rucci1	295439	1239.2	**259681**	**1087.5**	1669040	22554.3	*27267*	*222.2*
LargeRegFile	61149	522.9	**29027**	**188.9**	49906	1363.8	*793*	*8.2*
sls	198193	1610.2	**48769**	**313.2**	218134	7807.1	*10969*	*75.9*

ments, demonstrating a clear advantage in robustness and performance. PROPACK, even with fine-tuned settings, has trouble converging to the largest singular values of the most difficult case and to the smallest singular values of almost all cases. And when it converges it is significantly slower than PRIMME_SVDS. It is also slower than the mathematically equivalent LBD method in SLEPc, probably because of the use of partial instead of full reorthogonalization.

Table 2.14: Seeking 5 largest and smallest singular triplets with user tolerance 10^{-6} and 10^{-12} without preconditioning. We report both runtime and number of matrix-vector operations from both PHSVDS variants: PHSVDS(GD+k) and PHSVDS(JDQMR) in PRIMME_SVDS and PROPACK.

	P(GD+k)		P(JDQMR)		PROPACK	
Matrix	MV	Sec	MV	Sec	MV	Sec
the 5 largest singular values with tolerance 10^{-6}						
cage15	**872**	532.9	1238	**499.9**	1640	741.7
atmosmodl	**1824**	233.3	2514	**184.6**	15308	1429.2
Rucci1	**206**	**8.2**	426	11.2	348	28.0
LargeRegFile	**52**	**6.6**	108	7.5	144	24.8
sls	**50**	**3.3**	154	5.4	144	11.9
cont1_l	**1292**	217.8	2990	**210.8**	–	–
the 5 smallest singular values with tolerance 10^{-6}						
cage15	**1054**	652.7	1428	**600.3**	1368	659.3
atmosmodl	**64082**	8603.8	69292	**5548.4**	–	–
Rucci1	**86072**	**2290.5**	103762	2394.1	–	–
LargeRegFile	16464	1168.4	**14434**	**530.8**	–	–
sls	20134	500.9	**18122**	**390.5**	–	–

2.7.3 Strong and Weak Scalability

We investigate the parallel performance of various methods in PRIMME_SVDS and in SLEPc on a distributed memory system. All methods use the parallel sparse matrix-vector operation in PETSc.

In the first comparison, we report speedups in time of various methods over SLEPc's LBD method as we vary the number of processes from one to 100. The other methods are the GD+k method in the first stage of PRIMME_SVDS, and GD+k, Jacobi-Davidson (JD), and Krylov-Schur (KS) as implemented in SLEPc, all operating on C. The runs were made on SciClone and the results are shown in Figure 2.10. While PHSVDS clearly outperforms the rest of the solvers in terms of time, what is more relevant is that the ratio for PHSVDS keeps almost constant with the number of processes. This is an indicator that PRIMME_SVDS has similar parallel scalability with LBD, and better scalability than other SLEPc solvers (including the similar GD+k). Notice that the matrix of the left plot, delaunay_n24, is extremely sparse with only 3 elements per row, so this scalability study reflects better the parallel efficiency of the solver and much less of the PETSc matvec.

To further study the parallel scalability of the code, we again use the delaunay_n24

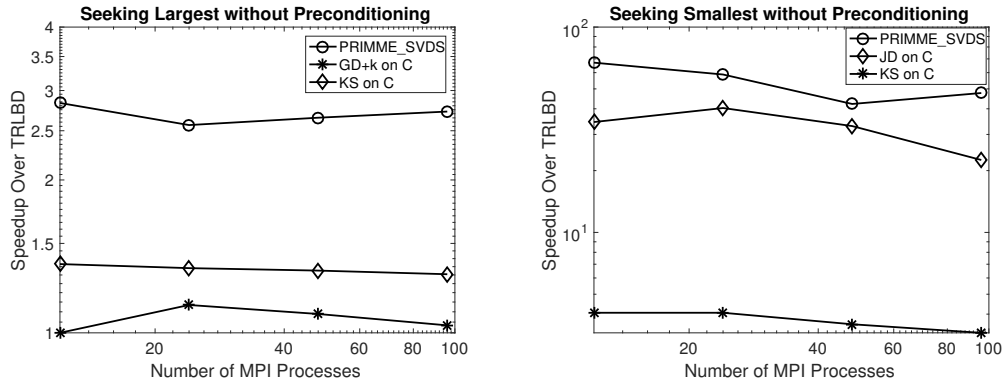


Figure 2.10: Speedup over SLEPc LBD computing 10 largest and smallest singular values on delaunay_n24 (left) and relat9 (right) with user tolerance 10^{-6} without preconditioning when increasing number of MPI processes on SciClone. The sparse matrix-vector operations are performed using PETSc.

sparse matrix and also the two somewhat denser matrices, cage15 and relat9. The relat9 is rectangular with a small dimension of about half a million which is used as a stress test for strong scalability. We also test the weak parallel scalability of the code using a series of 3D Laplacian matrices, making one of its dimensions proportional to the number of processes; each process maintains 8,000 rows when the number of the MPI processes increases from 64 to 1000. The plots in Figures 2.11 show the scalability performance of PRIMME_SVDS on Edison when seeking 10 extreme singular triplets with and without preconditioning.

In Figure 2.11(a), PRIMME_SVDS can achieve near-ideal speedup until 256 processes on relat9, despite the small size. With 512 processes, the speedup starts to level off as each process has only about 1,000 rows of a very sparse matrix. In Figure 2.11(b), we use the HYPRE boomerang multigrid preconditioner so the parallel efficiency is dominated by this library function. Still, the speedup is good up to 512 processes implying that good preconditioners should be used when available. Figure 2.11(c) illustrates the same good scalability performance when seeking largest singular triplets without preconditioning. In Figure 2.11(d) the code demonstrates good performance under weak scaling of the 3D Laplacian when adjusting the number of nodes from 80 to 200. The number of

MPI processes increases accordingly so that the local matrix size in each MPI process is kept fixed to 8000.

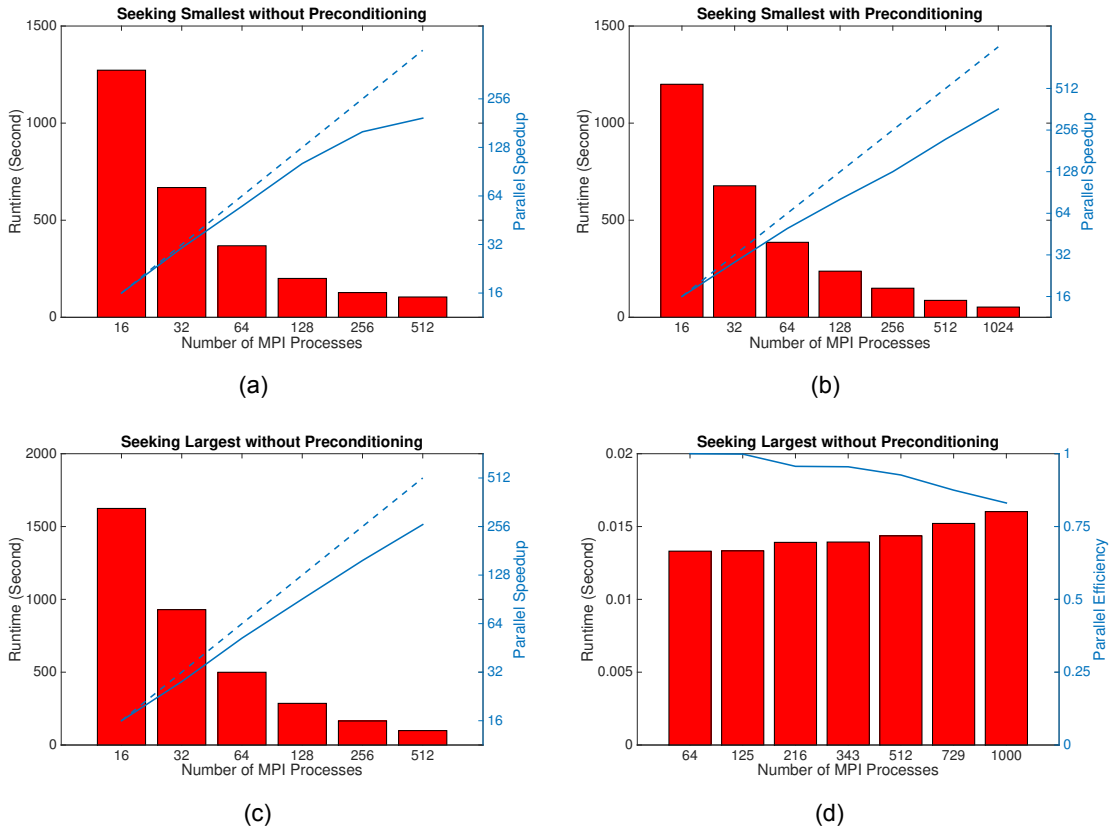


Figure 2.11: Speedup and runtime seeking the 10 smallest singular triplets in relat9 without preconditioning (a) and in cage15 with HYPRE boomeramg as preconditioner (b), and the 10 largest in delaunay_n24 (c); parallel efficiency and runtime seeking the 50 largest singular values in 3D Laplacian with dimension 8,000 times the number of precesses (d).

2.8 Conclusion and Future Work

In this research, we present a two stage meta-method, PHSVDS, that computes smallest or largest singular triplets of large matrices. In the first stage PHSVDS solves the eigenvalue problem on the normal equations as a fast way to get sufficiently accurate approximations, and if further accuracy is needed, solves an interior eigenvalue problem from the augmented matrix. We have presented an algorithm and several techniques re-

quired both at the meta-method and at the eigenvalue solver level to allow for an efficient solution of the problem. We have motivated the merit of this approach theoretically, and confirmed its performance through an extensive set of experiments.

We also implemented PHSVDS on top of PRIMME to provide a state-of-the-art robust, high performance SVD solver supporting accurate computation of extreme singular triplets for both square and rectangular matrices, with or without preconditioning. In particular, we show that our SVD solver can take advantage of a preconditioner, if present, in order to effectively tackle very large-scale real-world problems. In addition, we demonstrate the good scalability of our SVD software, PRIMME_SVDS, on parallel machines under different parameter settings.

Chapter 3

Efficient Computation of Interior Eigenvalues

3.1 Introduction

We examine the problem of seeking a few interior eigenvalues of a real symmetric matrix A. We assume that the user provides at least one shift τ or multiple shifts $\tau_1, \tau_2, \ldots, \tau_k$ near the eigenvalues of interest. We denote by $\|.\|$ the 2-norm of a vector or a matrix, by A^T the transpose of A.

Iterative methods are often the only means for computing eigenpairs (λ, x) when A is a large, sparse matrix. Well-known examples include the Lanczos method [99], the Arnoldi method [79] and the (Jacobi-)Davidson type methods [27, 112]. Typically, most of iterative methods approximate eigenvalues of a large sparse matrix A by constructing a sequence of subspaces V_m that contain increasingly accurate approximations to the desired eigenvectors. After building a subspace V_m, the second step is the use of a projection technique to find an eigenvector approximation from that subspace that is best in some way. This step is critical for the convergence of the method because an approximate eigenpair from the subspace is used to restart the method and often even in the computation of a vector for expanding the subspace.

The most commonly used method for extracting an approximation from a subspace is the Rayleigh-Ritz method. For a symmetric matrix, the Rayleigh-Ritz method has some optimality properties for extreme eigenvalues [99]. Unfortunately, for interior eigenvalues, the Rayleigh-Ritz method could give poor eigenvector approximations even in the symmetric case [91, 122]. This is the result of the interlacing property of the Ritz values [99], which makes Rayleigh-Ritz robust for eigenvalues at the boundary of the spectrum but not in the interior of spectrum [91, 65, 113].

To cope with interior eigenvalue problems, the best method is to apply the Rayleigh-Ritz method to a shift and invert operator [108], $(A - \tau I)^{-1}$. However, inverting the matrix becomes intractable due to large size of the problem. The harmonic Rayleigh-Ritz method, mainly due to the early work by Morgan [91] and formally introduced by Paige, Parlett and Van der Vorst [98], performs Rayleigh-Ritz using the subspace $(A - \tau I)V$ as search space to avoid explicitly inverting the matrix $(A - \tau I)$. The approximate eigenpairs near τ can be computed more accurately from V once the approximate eigenvectors from $(A - \tau I)V$ are determined. In addition, when approximating interior eigenvalues, one can achieve monotonic convergence of harmonic Ritz values compared to the irregular convergence of Ritz values, which is certainly beneficial for restarting [92]. Therefore, it has received considerable attention both for interior eigenvalue problems and singular value problems [91, 92, 59, 60, 8, 69, 74]. Several efficient implementations are also proposed in the context of the Krylov-Shur method [103] and the Jacobi-Davidson method in [11].

Another remedy for interior eigenvalues is the Refined Rayleigh-Ritz method proposed by Jia [65]. The refined projection method seeks a unit length refined eigenvector approximation from the subspace V that minimizes the corresponding residual norm with respect to the shift τ and the subspace V. In the refined projection method, it is proposed to pick τ as the currently best known approximation for the desired eigenvalue, such as the Ritz value in the Rayleigh-Ritz method [70] or the Harmonic Ritz value in the Harmonic Projection method [67]. A more general choice has also been discussed in [111]. A number of research efforts have been proposed to apply refined projection method to extract bet-

ter eigenvector or singular vector approximations [68, 69, 66, 74, 60]. Although efficient implementations of the refined Rayleigh-Ritz methods have been presented for Krylov methods [65, 8], few studies [66, 39] have discussed how to efficiently carry out refined projection for non-Krylov subspace methods such as the Generalized Davidson (GD) or Jacobi-Davidson (JD) type methods.

In this paper, we provide insights and develop practical algorithms to accomplish efficient and accurate computation of interior eigenpairs using refined projection techniques for non-Krylov methods. We firstly compare different implementations of the refined projection and analyze their numerical accuracy and computational costs. Based on the advantages of different approaches, we propose a new hybrid method to more efficiently find interior eigenpairs without compromising accuracy. We also provide more insights by investigating the effects of single and multiple user shifts for robustly seeking more than one eigenvalues. Our numerical experiments illustrates the efficiency and robustness of the proposed method.

The rest of this chapter is organized as follows: we briefly introduce the Rayleigh-Ritz procedure and discuss why it may fail to converge in Section 3.2. In Section 3.3, we describe the refined Rayleigh-Ritz method, discuss the numerical issues and computational costs of different implementations and present an accurate and more efficient approach. In Section 3.4, we conclude this research and discuss future work.

3.2 The Rayleigh-Ritz Method

Let $V \in \Re^{n\times m}$ be an orthonormal matrix, whose columns span an m dimensional subspace $\mathcal{V}$. We are interested in iterative methods that only leverage the information about $\mathcal{V}$ and $A\mathcal{V}$ to compute approximations for interior eigenvalues. Among them, the Rayleigh-Ritz projection method is the most popular method.

The Rayleigh-Ritz approach extracts the approximate eigenpairs (θ, u) where $u \in \mathcal{V}$

by imposing the Galerkin condition

$$Au - \theta u \perp V \tag{3.1}$$

where V are both a search space and a test space. However, a test space could be flexible which generates different projection methods (such as Harmonic projection method [91]). Algorithm 5 describes the Rayleigh-Ritz procedure in the simplest form.

Algorithm 5 The Rayleigh-Ritz Method

1: Given a subspace $\mathcal{V}$ of dimension m. Compute an orthonormal basis V, an n by m orthonormal matrix for $\mathcal{V}$.
2: Compute the m by m projected matrix H_m = $V^T AV$ based on (3.1).
3: Compute the eigenpairs (θ_i, g_i) of H_m, $i = 1, \ldots, m$.
4: Order θ_i according to the user shift τ, and take (θ_i, u_i) = (θ_i, Vg_i) as the approximate eigenpairs.

Approximate eigenvectors $u_i = Vg_i$ are called Ritz vectors, and $\|u_i\| = 1$. The eigenvalue approximations $\theta_i = \rho(u_i) = g_i^T H_m g_i$ are called Ritz values, the Rayleigh quotient of the corresponding vectors u_i. To monitor if the current Ritz pairs (θ_i, u_i) approximating the eigenpairs converged, the most common convergence criterion is shown below based on a user defined tolerance δ_{user} [105],

$$\frac{\|Au_i - \theta_i u_i\|}{\|A\|} \leq \delta_{user}. \tag{3.2}$$

It is expected that if $\mathcal{V}$ contains an approximate eigenspace of A, there should be an eigenpair (θ_i, g_i) of H such that (θ_i, Vg_i) is an approximate eigenpair of A [122]. Jia and Stewart [70] have shown that if the Ritz value θ converges to the simple eigenvalue λ and the residual approaches to zero, then the convergence of the Ritz vector u to the eigenvector x is guaranteed. However, the assumption that the desired eigenvalue is well separated from the remaining eigenvalues may not hold, especially for interior eigenvalues.

As observed in [91, 92, 122, 70], there are two difficulties to ensure that the Ritz vector

converges to the eigenvector when seeking interior eigenpairs. First, even though current subspace contains good approximations for the desired eigenvector, Rayleigh-Ritz may not extract good Ritz vectors from the subspace, if there are other Ritz values that are close to the desired Ritz value [91, 92, 70]. Second, spurious Ritz values close to the desired Ritz value may be produced for interior eigenvalues due to cancellations from combinations of nearby eigenvectors, which have little to do with the desired eigenvector x . More importantly, it is difficult to distinguish the right Ritz value from spurious ones unless the residual norms were computed first, especially when the user shifts are not close to the desired eigenvalues. In order to overcome the difficulties of the Rayleigh-Ritz method when targeting interior eigenpairs, the refined projection method could be leveraged.

3.3 The Refined Rayleigh-Ritz Method

As the subspace $\mathcal{V}$ contains an increasingly accurate approximation u to the eigenvector x, the Ritz value θ can converge to the simple eigenvalue λ [122]. Therefore the problem can be attacked by retaining the Ritz value and replacing the Ritz vector with a vector $\hat{u} \in V$ such that $\|A\hat{u} - \theta\hat{u}\|$ is minimized over the subspace. Then the vector $\hat{u}$ is ensured to converge to the eigenvector x. This suggests the following procedure proposed by Jia [65] when we seek a number of k interior eigenvalues:

$$\begin{aligned} &\text{minimize} \quad \|A\hat{u}_i - \theta_i\hat{u}_i\| \\ &\text{subject to} \quad \hat{u}_i \in V, \|\hat{u}_i\| = 1, \end{aligned} \qquad i = 1, 2, \ldots, k. \tag{3.3}$$

where $\hat{u}_i$ is called refined Ritz vector with respect to some approximate eigenvalue θ_i. Since the refined Ritz vector is a better approximation than the Ritz vector in the sense of smaller residual norm, the Rayleigh quotient $\hat{\theta}_i = \hat{u}_i^T A\hat{u}_i$ can be used to replace the Ritz value θ_i. The reason is that $\hat{\theta}_i$ minimizes the $\|A\hat{u}_i - \hat{\theta}_i\hat{u}_i\|$ over all Ritz values and is likely to yield a more accurate eigenvalue approximation [122].

A closely related method is proposed by Morgan in [91], where a spectrum transformation is used so that the target interior eigenvalues are mapped to the boundary by applying Rayleigh-Ritz to the operator $(A-\tau I)^2$, which leads to

$$(A-\tau I)^2\tilde{u}-\tilde{\theta}\tilde{u}\perp V, \tag{3.4}$$

where the smallest eigenpairs $(\tilde{\theta}_i,\tilde{g}_i)$ of $V^T(A-\tau I)^2V$ are computed. The desired interior eigenvalues $\tilde{\theta}$ can be computed as Rayleigh quotients $\rho(\tilde{u})$, where $\tilde{u}=V\tilde{g}_i$. However, solving (3.4) directly is not numerically stable since the rounding-off errors could be as large as $\|A\|^2$mach. A better way is to solve an optimization problem just like the refined projection in (3.3),

$$\begin{aligned} &\text{minimize} \quad \|A\tilde{u}-\tau\tilde{u}\| \\ &\text{subject to} \quad \tilde{u}\in V, \|\tilde{u}\|=1. \end{aligned} \tag{3.5}$$

Compared (3.3) and (3.5), the only difference is that Jia uses updated shifts at each iteration while Morgan keeps the original user shifts. Due to this close relationship with the Refined projection method, we call Morgan's method, simplified Refined projection method. Since τ is a fixed shift, an accurate computation using the stable QR factorization and the singular value decomposition can be efficiently carried out in non-Krylov methods as we will discuss later. In addition, a set of refined Ritz vectors are computed once when solving (3.5). A similar strategy has also been discussed in [138, 60].

3.3.1 Analysis of computation and accuracy

Let the refined Ritz vector $\hat{u}_i$ associated with θ_i be represented by $V\hat{g}_i$, where $\|\hat{g}_i\|=1$. It is easy to see that:

$$\begin{aligned} \min\|(A-\theta_iI)\hat{u}_i\| &= \min\|(A-\theta_iI)V\hat{g}_i\| \\ &= \sigma_{min}((A-\theta_iI)V) \end{aligned} \tag{3.6}$$

To compute each refined Ritz vector, we have to solve the smallest singular value problem for each different θ_i. However, if the refined projection method is applied to a Krylov subspace such as the ones produced by the Lanczos or Arnoldi methods, the computation of the refined Ritz vectors can be reduced to solving a set of small SVD problems with the projected matrix H_m [65, 68, 69]. But in this work, we focus on addressing the general case of a non-Krylov subspace such as in the GD/JD type methods.

In the first approach (we call it approach I), we compute the QR decomposition of $(A - \theta_i I)V = Q_i R_i, i = 1, 2, \ldots, k$, where Q_i are $n \times m$ orthonormal matrices, and R_i are $m \times m$ matrices. Then we solve a set of small SVD problems on each R_i and take the corresponding right singular vector $\hat{g}_i$. The operation cost for these computations requires $O(knm^2)$ per iteration [122]. Therefore, the total operations of the refined projection method are $O(knm^3)$ per restart in the GD/JD type methods. Compared to the Rayleigh-Ritz method that required only $O(nm^2)$ operations to obtain k number of Ritz vectors per restart [122], it becomes tremendously expensive taking longer time by a factor of at least km and needs additional memory for maintaining Q_i. However, the merit of this approach is that it can achieve the same accuracy of $\|A\|$mach as the Rayleigh-Ritz method, where mach is the machine precision. In addition, the QR factorization can take advantage of BLAS level 3 kernels thereby it could be quite efficiently performed. Overall, the first approach is a computationally expensive but numerically stable method when the subspace V is not a Krylov subspace.

To reduce the cost of computing the refined Ritz vectors, the second approach (we call it approach II) finds the smallest eigenpair for each eigenvalue approximation θ_i by solving k small standard eigenvalue problems. The same strategy was also suggested in

[66]. Specifically, from (3.6) it is easy to derive that:

$$\begin{aligned}
\min\|(A-\theta_i I)\hat{u}_i\| &= \min\|(A-\theta_i I)V\hat{g}_i\| \\
&= \min((A-\theta_i I)V\hat{g}_i)^T((A-\theta_i I)V\hat{g}_i) \\
&= \min(\hat{g}_i^T(V^TA^TAV - 2\theta_i V^TAV + \theta_i^2 I)\hat{g}_i) \\
&= \sigma_{min}(V^TA^TAV - 2\theta_i V^TAV + \theta_i^2 I) \\
&= \sigma_{min}(W^TW - 2\theta_i V^TW + \theta_i^2 I) \\
&= \sigma_{min}(G - 2\theta_i H + \theta_i^2 I)
\end{aligned} \tag{3.7}$$

where $W = AV$, $H = V^TAV$ and $G = W^TW$. If we store the G and H matrices and keep updating them every iteration, the computation of k small eigenvalue problems is $O(km^3)$ which is negligible compared to the overall cost per outer iteration. Compared to computing the Rayleigh-Ritz vectors, it needs additional memory for storing G and an additional $O(nm^2)$ operations to update it per restart. However, the serious problem with this approach is its numerical accuracy issue. Since the norm of G is the square of the norm of W, any operation involving G could be inaccurate since the error bound for the matrix multiplication is $\|G\|$mach. In addition, since the numerical error for forming G could spread to the formation of $(G - 2\theta_i H + \theta_i^2 I)$, therefore the eigenpairs obtained by solving this eigenvalue problem are either not numerically accurate either. As a result, the computation of V and W could also accumulate numerical errors during each restart.

We design an approach (we call it approach III) to inherit the merit of the first approach, good numerical stability, but compute the refined Ritz vectors inexpensively. We compute a set of the smallest singular values and the right singular vectors of the $(A-\theta I)V$ instead of computing only one singular triplet every iteration like the simplified Refined projection method in (3.5). However, we update the shift θ by the latest desired Ritz value every iteration. Therefore, the computation of the refined Ritz vectors is reduced at the cost of the effectiveness of the refined Ritz vectors $\hat{u}_i, i = 2, \ldots, k$. The similar strategy is also suggested in [60] where the author only considers the special case when the shift $\theta = 0$.

We can see that the computation of the refined Ritz vectors requires $O(nm^2)$ operations per iteration and $O(nm^3)$ operations per restart, which is still much higher than that of computing the Ritz vectors. Although the effectiveness of other refined Ritz vectors is reduced, these approximations have the desirable property of monotonic convergence to targeted eigenvalues as claimed in [60, 138]. In addition, it may provide a better subspace than the original refined projection method at the early stage of the convergence. The reason is that the Ritz values θ_i could be very bad approximations of the actual eigenvalues therefore the resulting refined Ritz vectors obtained by solving each SVD problem with shift θ_i could be much worse than these obtained by solving only one SVD problem with the target shift.

Another approach we propose (we call it approach IV) is similar to the second approach that solves the smallest eigenvalue problem rather than the smallest singular value problem. In contrast, it follows the spirit of the third approach that solves only one smallest eigenvalue problem to obtain all refined Ritz vectors. This approach may suffer from the same numerical problems as the second approach but has the least computational cost of only $O(m^3)$ per iteration. Also, it needs an additional $O(nm^2)$ operations to update G per restart, which is similar to the Rayleigh-Ritz method.

The last approach (we call it approach V) is the reformulation of Morgan's method [91], the simplified Refined projection method as described in (3.5). It is similar to the third approach but uses fixed user shifts. Therefore, it still inherits the features of good numerical stability and overall convergence since it produces a set of refined Ritz vectors with one shift in each iteration. In particular, its computational costs could be substantially reduced to $O(nm^2)$ per restart, just like the fourth approach and the Rayleigh-Ritz method. This is because there is no need to perform a full QR factorization of (3.5) at every iteration since τ remains constant. Instead, as part of Gram-Schmidt, we add one more column to the orthonormal matrix Q and to the matrix R and only a full QR factorization of R is needed at restart.

However, there are two serious disadvantages for this approach. First, even though

this approach can correctly select target interior eigenvalues and achieve monotonic convergence in terms of residual norm, it may converge much slower than the other approaches. The reason is that solving (3.3) with updated shifts could generate the smallest possible Lehmann interval [111] and therefore produce a better approximation to the target eigenvalue λ than solving (3.5) with a fixed shift. Second, solving (3.5) with a fixed shift tends to converge to the interior eigenvalues nearest to user shift first while solving (3.3) with updated shift approximating target eigenvalue can be configured to converge from one direction nearest to user shift. For instance, in some large-scale applications, the target shift τ is placed slightly to the right of the origin in order to seek a few positive interior eigenvalues [149]. Although one can search all nearby eigenvalues to seek desired direction of interior eigenvalues, it can be less efficient especially when a lot of unwanted eigenvalues have to be computed firstly.

3.3.2 Comparisons between various approaches

In order to investigate the performance of these five approaches, we incorporate them into the GD algorithm, a basic version of GD in [117], as shown in Algorithms 6 and 7. We search for nev interior eigenpairs of the matrix NOS3 using these refined projection approaches.

Figure 3.1 compares the convergence of five different approaches when seeking four interior eigenvalues with single shift and multiple shifts. In order to make a fair comparison of convergence, we set the tolerance to be achievable for all approaches. Also, we have to mention that these user shifts are not used for computing refined Ritz vectors but used to select the desired Ritz values in first four approaches. If a single shift is used, then it will be used for targeting all desired number of interior eigenvalues; If multiple shifts are used, then each shift is used for determining the interior eigenvalues after one eigenvalue is found. For approach V, these shifts are used as τ in the computation of refined Ritz vectors in (3.5).

Algorithm 6 The Generalized Davidson(m_{min},m_{max}) algorithm

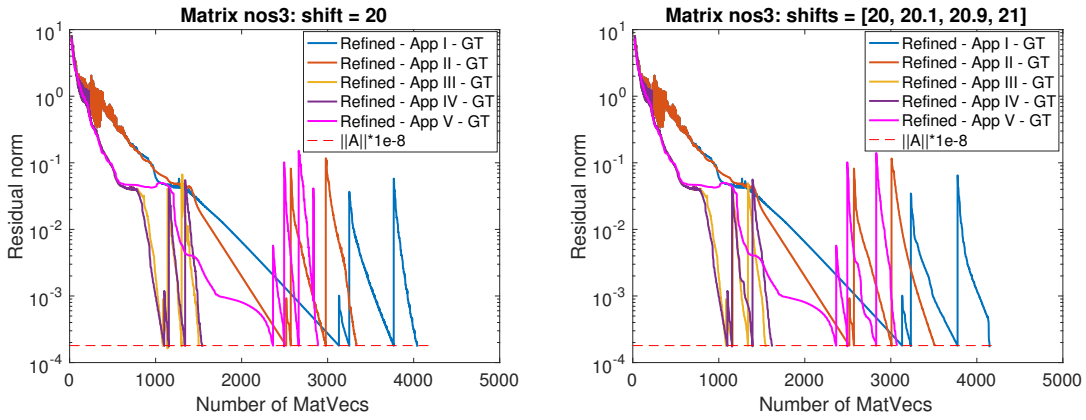
start with starting vector v_0 and user shifts $\tau_i, i = 0, \ldots, nmv$
$t^{(0)} = v_0, l = m = 0, X = [\ ]$
while $l < nev$ and $nmv < max\_num\_matvecs$ **do**
  Orthonormalize $t^{(m)}$ against $v_i, i = 1, \ldots, m$, $m = m + 1, v_m = t^{(m-1)}, w_m = Av_m$
  $H_{i,m} = v_i^T w_m$ for $i = 1, \ldots, m$
  if $(App == 2 \parallel App == 4)$ **then**
    $G_{i,m} = w_i^T w_m$ for $i = 1, \ldots, m$
  end if
  ${\hat{g}_i}^{old} = \hat{g}_i, i = 1, \ldots, m$
  Compute eigendecomposition $H = S\Theta S^T$
  Call algorithm 10 to obtain τ and sort Θ closest to τ w.r.t. target eigenvalues
  if $(App == Hybrid)$ **then**
    Call Algorithm 9 with Θ for computing refined Ritz vectors $\hat{g}_i$
  else
    Call Algorithm 7 with Θ for computing refined Ritz vectors $\hat{g}_i$
  end if
  $u^{(m)} = V\hat{g}_1, \theta^{(m)} = \hat{\theta}_1 = {\hat{g}_i}^T H\hat{g}_i, w^{(m)} = W\hat{g}_1$, $r^{(m)} = w^{(m)} - \theta^{(m)}u^{(m)}$
  while $\|r^{(m)}\| \leq tol$ **do**
    $\lambda_{l+1} = \theta^{(m)}, X = [X, u^{(m)}], l = l + 1$, $m = m - 1, H = 0$
    if $l == nev$ **then**
      Return all converged eigenpairs with desired accuracy
    end if
    if $(App == 1 \parallel App == 2)$ **then**
      Compute QR decomposition of $\hat{g}$ and set $\hat{g} = Q$
    end if
    $v_i = V\hat{g}_{i+1}, w_i = W\hat{g}_{i+1}, \theta_i = \theta_{i+1}, H = \hat{g}_{i+1}^T H\hat{g}_{i+1}$ for $i = 1, \ldots, m$
    if $(App == 2 \parallel App == 4)$ **then**
      $G = \hat{g}_{i+1}^T G\hat{g}_{i+1}$ for $i = 1, \ldots, m$
    end if
    $u^{(m)} = v_1, \theta^{(m)} = \theta_1, r^{(m)} = w_1 - \theta^{(m)}u^{(m)}$
  end while
  if $m \geq m_{max}$ **then**
    if $(App == 1 \parallel App == 2)$ **then**
      Compute QR decomposition of $\hat{g}$ and set $\hat{g} = Q$
    end if
    Call algorithm 8 for a locally optimal restarting technique (+k)
    $m = m_{min} + k, v_i = V\hat{g}_i, w_i = W\hat{g}_i, H = \hat{g}_i^T H\hat{g}_i$ for $i = 1, \ldots, m_{min}$
    if $(App == 2 \parallel App == 4)$ **then**
      $G = \hat{g}_i^T G\hat{g}_i$ for $i = 1, \ldots, m_{min}$
    end if
  end if
  Precondition the residual $t^m = Prec(r^m)$
end while

Algorithm 7 Check different approaches for computing refined Ritz vectors

if $App == 1$ **then**
  for $i = 1 : k$ **do**
    $W - \theta_i V = Q_i R_i$
    compute smallest singular value σ_1 and its right singular vector $\hat{g}_1$ of R_i
  end for
else if $App == 2$ **then**
  for $i = 1 : k$ **do**
    $G - 2\theta_i H + \theta_i^2 I = Q_i R_i$
    compute the smallest eigenpair (σ_1, $\hat{g}_1$) of R_i
  end for
else if $App == 3$ **then**
  $W - \theta_1 V = Q_1 R_1$
  compute singular values $R_1 = U \Sigma S^T$ with $\sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_m$
  $\hat{g} = S$
else if $App == 4$ **then**
  $G - 2\theta_i H + \theta_i^2 I = Q_1 R_1$
  compute eigendecomposition $R_1 = S \Sigma S^T$ with $\sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_m$
  $\hat{g} = S$
else if $App == 5$ **then**
  if $m == m_{min}$ **then**
    Perform full QR, $W - \tau V = Q_1 R_1$
  else
    Perform one-column updating, $W - \tau V = Q_1 R_1$
  end if
  compute singular values $R_1 = U \Sigma S^T$ with $\sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_m$
  $\hat{g} = S$
end if

Algorithm 8 Locally optimal restarting technique (+k) for GD

Orthogonalize $s_i^{old}, i = 1, \ldots, k$ against s
set $s = [s_1, \ldots, s_{min}, s_1^{old}, \ldots, s_k^{old}]$



(a) Single User Shift (b) Multiple User Shifts

Figure 3.1: Compare the convergence of five different approaches on matrix NOS3 from the Florida Sparse Matrix Collection [28] with single shift and multiple shifts, respectively. The largest and smallest eigenvalues of matrix NOS3 are 689.9 and 0.018. The targeted interior eigenvalues are 20, 20.1, 20.9, 21, respectively.

In both cases, the approaches III and IV converge faster than the approaches I and II at the beginning and keep their advantages to the end despite being less expensive. The reason is that these two approaches build better subspace than the approaches I and II in the early stage. When the Ritz values are not good, the quality of refined Ritz vectors computed with each Ritz value (approach I and II) may be worse than those with only one target Ritz value (approach III and IV). These are indeed observed in Figure 3.1. Theoretically, one may expect when the Ritz values are good approximations of the actual eigenvalues, the subspace built by these refined Ritz vectors computed individually should be better than these computed with only one target Ritz value. However, as revealed in Figure 3.1, this strategy that computes a set of refined Ritz vectors with only one target Ritz value seems also to improve the global convergence. On the other hand, we observe that the approach V converges as fast as the approaches III and IV but much worse when the Ritz value starts approximating the target eigenvalue well. It confirms our previous discussion that as the Ritz value approximates the target eigenvalue better and better, the benefits with more accurate shifts in (3.3) magnify in terms of the improvement of refined Ritz vectors compared to the ones with a fixed shift (3.5) when the problem gets harder.

This is also observed in our subsequent experiments.

3.3.3 An efficient and accurate hybrid method

Our preliminary experimental results reveal that approaches III and IV are not only computationally less expensive than their counterparts but also display better convergence. However, we have to make two comments. First, when the shift τ is a highly accurate approximation to the desired eigenvalue, the approach V is expected to deliver the same quality of refined Ritz vectors as other approaches with updated shifts since the difference between τ and θ is negligible. See also discussions of the refined projection in [138, 149]. It implies that if the shift τ is very close to the desired eigenvalue, the approach V should be the method of choice among these approaches because it is a numerically stable and the most efficient one. In addition, approach II may not seem a good choice for the computation of the refined Ritz vectors in general, but one significant advantage of this approach is that it enables inexpensive computation of refined Ritz vectors with multiple different shifts. This could be used to select the right Ritz value to efficiently compute the refined Ritz vectors when the spurious Ritz values appearing close to the target eigenvalue is detected. Therefore, to fully take advantage of the merits of these approaches, we propose an efficient and accurate hybrid method for implementing the refined projection in non-Krylov methods, as shown in Algorithm 9.

First, our hybrid approach takes advantage of the cheaper computation of approach IV while the numerical errors that it introduces are below the residual norm of the approximations of the iterative solver. The main goal of this stage is to leverage its faster convergence and inexpensive computation of the refined Ritz vectors. In addition, as long as the condition number of the target interior eigenvalues is not too large ($\leq \sqrt{\text{mach}}$), the refined Ritz value $\hat{\theta}$ will converge to the target eigenvalue and could be much more accurate than the corresponding refined Ritz vectors to the target eigenvector [66, 149]. If low accuracy is required, this stage is expected to be sufficient. However, if high accuracy

is demanded, a second stage is needed to continue refining the accuracy of the desired eigenpairs, especially for improving the refined Ritz vectors.

Second, note that as the angle between the search space V and the deisred eigenvectors approaches to zero, it is known theoretically [70, 122] and empirically [91, 67] that spurious Ritz values could frequently appear close to the desired eigenvalue, which cannot be distinguished from the right Ritz pair without computing its residual norm. This case would happen more frequently if the user shift τ is not close to the desired eigenvalues. If such a spurious Ritz value is picked up, the corresponding computed eigenvector approximation may be poor [70], which not only leads to the irregular convergence but also damages the convergence if the bad eigenvector approximations are chosen to restart with [103]. Therefore, how to appropriately select the right Ritz value to compute refined Ritz vectors is very important. We develop a new inexpensive scheme to dynamically detect spurious eigenvalues in order to avoid misconvergence and reduce irregular convergence by taking into account the merits of the approach II.

Third, when the residual norm approaches $\|A\|^2$mach (which can be monitored), we switch to approach V which allows convergence to $\|A\|$mach. In the same spirit as in the development of our two stage SVD method [138], we perform one-column updating of the QR factorization since the Ritz value is good enough and almost remains constant in Rayleigh-Ritz. Since τ is set to be $\hat{\theta}$ computed from the last interation in the first stage and remains constant, there is no need to perform a QR factorization of $AV - \tau V$ at every step. Instead, as part of Gram-Schmidt, we update the factorization matrices Q and R with a new column. A full QR factorization is only needed at restart. Then, following [122], we compute the refined Ritz vectors by solving the small SVD problem with R, and replace the targeted Ritz value with the Rayleigh quotient of the first refined Ritz vector $\hat{u}$.

Approaches II and IV have similar cost to computing the Ritz vectors. Solving approach V with a fixed shift requires the same computational expense as the Rayleigh-Ritz procedure. Therefore, the total computation of the proposed hybrid approach is as efficient as the Rayleigh-Ritz method but with added robustness due to the monotonic con-

Algorithm 9 A hybrid method for refined projection in line 14 of Algorithm 6

First-stage with Approaches IV:

if $(\|(A - \hat{\theta}_i I)\hat{u}_i\| \leq \|A\|^2 \text{mach})$ **then**
  while Criteria are not meet **do**
    $\tau = \theta_i$, $i = i + 1$
    Set $app = 4$ and call Algorithm 7 with shift τ to compute refined Ritz vectors
  end while
else

Second-stage with Approach V:

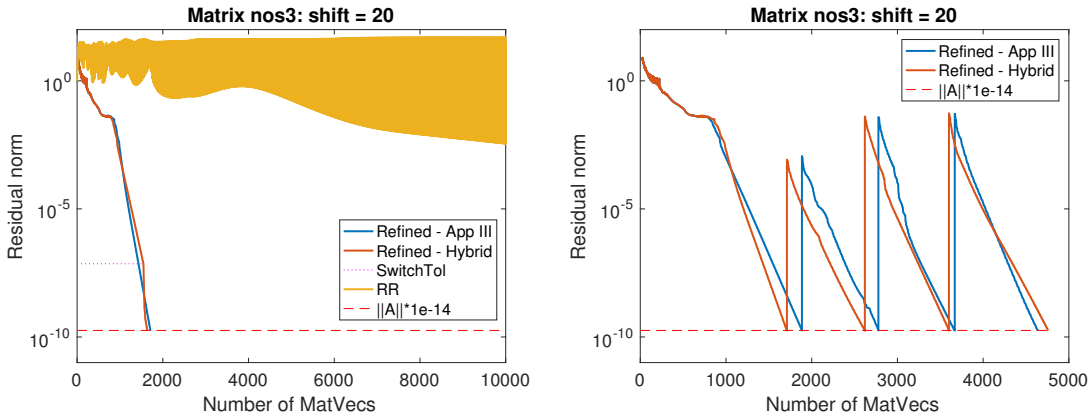
  Set $\tau = \hat{\theta}$ obtained in the last iteration of the first stage
  Set $app = 5$ and call Algorithm 7 with shift τ to compute refined Ritz vectors
end if

vergence of the refined projection method in the context of interior eigenvalue problems.

In Figure 3.2, we compare our hybrid approach with approach III as well as Rayleigh-Ritz for seeking a few interior eigenvalue greater than $\tau = 20$. When seeking one interior eigenvalue, our hybrid approach (denoted as red line) can dynamically switch from approach IV with automatically filtering out spurious Ritz values to approach V with efficient QR updating. The switching tolerance is denoted as pink dashed line. Compared to approach III (denoted as blue line), it achieves very close performance but with much smaller computational costs. Rayleigh-Ritz converges very irregularly and slowly in this case largely because of the effect of spurious Ritz value and restarting with much worse eigenvector approximations. In addition, we see that when seeking several interior eigenvalues, the hybrid approach can still converge as efficiently as the approach III.

3.3.4 Effect of single user shift and multiple user shifts on seeking many eigenvalues

In this section, we study a subtle issue involving the shift selection for determining the order of the desired interior eigenvalues in the refined Rayleigh-Ritz procedure. We have observed that the convergence of the interior eigenvalues that are farther away from the given user shift in the refined Rayleigh-Ritz procedure is sometimes irregular, with long plateaus, and sometimes may stagnate. It often happens when seeking several interior



(a) Single User Shift (b) Multiple User Shifts

Figure 3.2: Sample example to show the convergence of hybrid approach for seeking one and several interior eigenvalues respectively.

eigenvalues with only one user shift. When the user provides only one shift, the first interior eigenvalue usually has the desired property of monotonic convergence since the user shift is closest to that interior eigenvalue. However, after the converged interior eigenvalue is locked out, the distance between the user shift and the next interior eigenvalue becomes increasingly larger, which makes it easier for spurious eigenvalues to appear and be selected as the targeted Ritz value. The refined Ritz vectors computed based on these spurious Ritz values as shifts, build a less meaningful subspace thereby the refined projection method often displays irregular convergence or even may not converge.

One possible solution to overcome this difficulty is to ask the user to provide many shifts, each closest to a required eigenvalue. However, it may not be a practical solution because the user may not have complete knowledge of where these desired eigenvalues are located in the spectrum. We propose to use the previous converged interior eigenvalues to derive more tight shifts. In addition, based on the user target eigenvalues (GT: greater than τ, or LT: smaller than τ, or closest to τ), we can use the perturbation bounds of the approximations of the eigenvalue value we seek. For instance, specifically for GT, we know that $\lambda_l \in [\hat{\theta}_l, \hat{\theta}_l + r_l]$, where $r_l = \|A\hat{u} - \hat{\theta}_l\hat{u}\|$. It is easy to see the next target eigenvalue must have $\lambda_{l+1} > \hat{\theta}_l + r_l$ based on the interlacing Property theory [99]. There-

fore, we can compare $\lambda_l + r_l$ with the next available shift τ_{l+1} and choose the one that is closer to the next desired interior eigenvalues. We show this strategy in Algorithm 10.

Algorithm 10 A new user shifts updating strategy

if (target == GT) and ($l \geq 1$) and (X is not empty) **then**
  $\tau = \texttt{max}(\tau_{l+1}, \lambda_l + r_l)$
else if (target == LT) and ($l \geq 1$) and (X is not empty) **then**
  $\tau = \texttt{min}(\tau_{l+1}, \lambda_l - r_l)$
else
  $\tau = \tau_{l+1}$
end if

Figure 3.3 shows the comparison results between the new and old shifting strategy for selecting the desired Ritz value given one user shift and multiple user shifts. First, we see that the new shifting strategy and multiple shifts can deliver the convergence for all four interior eigenvalues while the old shifting strategy fails to converge to the fourth one. In addition, as the distance between the user shift and the interior eigenvalues increases, the convergence of the old shifting strategy with one shift becomes more and more irregular than that of the new shifting strategy also with one shift. The convergence of the method with multiple shifts displays the least irregular convergence. This is expected since tighter shifts can significantly reduce the chance of spurious eigenvalues appearing. Furthermore, the new shifting strategy offers faster convergence speed than other two strategies.

3.4 Conclusion and Future Work

The computation of interior eigenvalues of large sparse matrices remains a challenging problem. Compared to the Rayleigh-Ritz method, the refined and harmonic Rayleigh-Ritz methods are more effective ways to extract Ritz pairs and achieve monotonic convergence of the eigenvalues, but with a much higher computational cost in Davidson type methods. In this work, we firstly analyze various different implementations of the refined Rayleigh-Ritz, and present a new accurate and efficient approach when computing a few

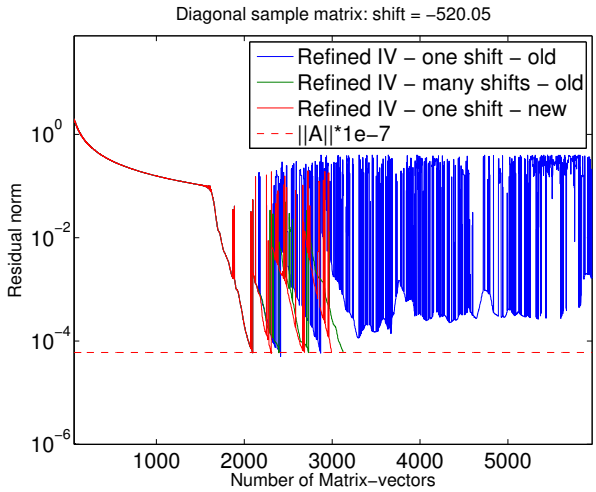


Figure 3.3: Sample example to show the difference between new and old shifting strategy with one user shift and multiple user shifts.

interior eigenvalues for non-Krylov methods. We also propose a new scheme for picking up the appropriate shift when only one user shift is available and compare its performance with that has multiple user shifts for robustly seeking more than one eigenvalues. Our numerical experiments demonstrate the effectiveness and accuracy of the presented method.

Chapter 4

A Novel Trace Estimator of Large Matrix Inverse

4.1 Introduction

The need to estimate the trace of a matrix which is implicitly available through a function arises in many applications. The main purpose of this research is to develop practical numerical techniques to address the computation of the trace of the inverse of a large, sparse matrix. But our technique can also be adapted to other functions such as the trace of the logarithm (yielding the determinant) or the trace of the matrix exponential.

For small size problems, computing A^{-1} through a dense or sparse LDU decomposition is the most efficient and accurate approach [33]. This works well for discretizations of differential operators in low dimensions but becomes intractable in high dimensional discretizations. For larger size problems, domain decomposition and divide and conquer strategies are more tractable but still expensive [125]. In many cases, however, a low accuracy approximation is sufficient. Numerous methods have been presented to address this need for estimating the trace of the inverse of symmetric positive definite matrices through Gaussian quadratures [12, 50], modified moments [90, 21], and MC techniques [12, 50, 129, 17, 90, 6].

MC methods for computing the trace of a matrix are based on the structure of the Hutchinson method [64], which iteratively computes an average of matrix quadratures with random vectors. Variants of MC estimators are mainly analyzed and compared based on the variance of one sample [6, 104], which depends on the quality of the selected random vectors. For real matrices, choosing random vectors having each element ± 1 with equal probability is known to minimize variance over all other choices of random vectors [64, 6] and therefore has been widely used in many applications. For complex matrices, the same result holds for vectors with $\pm 1, \pm i$ elements. In [6], Avron and Toledo analyze the quality of trace estimators through three different metrics such as trace variance, (ϵ, δ)-approximation of the trace, and the number of random bits for different choices of random vectors. In [104], Khorasani and Ascher improve the bounds of (ϵ, δ)-approximation for the Hutchinson, Gaussian and unit vector estimators. However, the structure of the matrix can also be exploited to accelerate the convergence of the Hutchinson method.

There has been a number of efforts to combine the Hutchinson method with well-designed vectors based on the structure of the matrix [129, 17, 126, 116]. In [17], they use columns of the Hadamard matrix, rather than random vectors, to systematically capture certain diagonals of the matrix. Then, the MC iteration achieves the required accuracy by continuously annihilating more diagonals with more Hadamard vectors. However, the location of the nonzeros, or of the large elements of A^{-1}, often does not coincide with the diagonals annihilated by the Hadamard vectors. In [126], graph coloring and probing vectors are used to identify and exploit special structures, such as bandedness or decaying properties in the elements of A^{-1}, to annihilate the error contribution from the largest elements. However, if the error for the chosen number of colors is large, all work has to be discarded and the probing procedure repeated until the accuracy is satisfied. In [116], the authors introduce hierarchical probing on lattices to avoid the previous problems and achieve the required accuracy in an incremental way. For all these approaches, the approximation error comes from non-zero, off-diagonal elements that have not been annihilated yet. Instead, this research looks only at the main diagonal of A^{-1}.

Our motivation for focusing only on the main diagonal is that the trace of A^{-1} is simply a summation of a discrete, 1-D signal of either the eigenvalues or the diagonal elements of A^{-1}. Although we cannot compute all the diagonal elements, we may have an approximation to the whole signal from the diagonal of an approximation of A^{-1} (e.g., of a preconditioner). If the two diagonals have sufficiently correlated patterns, fitting methods can be used to refine the approximation both for the diagonal and the trace. Therefore, the proposed method may serve as a standalone kernel for providing a good trace estimate with a small number of samples. But it can also be viewed as a preprocessing method for stochastic variance reduction for MC in cases where the variance reduces sufficiently. This can be monitored dynamically by our method.

We present several techniques that improve the robustness of our method and implement dynamic error monitoring capabilities. Our extensive experiments show that we typically obtain trace estimates with much better accuracy than other competing methods, and in some cases the variance is sufficiently reduced to allow for further improvements through an MC.

The remainder of this work is structured as follows. In Section 4.2, we review the Hutchinson method and the unit vector estimator, discuss different means of computing an approximation, describe the deflation techniques for reducing stochastic variance and provide a brief comparison of different MC methods. Section 4.3 presents a number of sampling strategies and illustrates two fitting models to exploit the pattern correlation between the diagonal of the approximation and the diagonal of the matrix inverse. In Section 4.4, we propose a dynamic evaluation framework for estimating the variance of different MC methods and monitor the trace error within the fitting stage. In Section 4.5, we give some numerical experiments to demonstrate the effectiveness of the proposed method. Conclusions and further thoughts are gathered in Section 4.6.

4.2 Preliminaries

We denote by $\|.\|$ the 2-norm of a vector or a matrix, by N the order of A, by Z an approximation of A, by D the diagonal elements of A^{-1}, by M the diagonal elements of Z^{-1}, by $Tr(f(A))$ the trace of the matrix $f(A)$, and by extension, $Tr(D)$ the sum of the elements of the vector D, by $T_{e_i}(f(A))$ the MC trace estimator of $f(A)$ using unit (orthocanonical) vectors, by $T_{Z_2}(f(A))$ the MC trace estimator of $f(A)$ using Rademacher vectors, by $diag(.)$ the diagonal operator of a matrix, and by $Var(.)$ the variance operator of a random variable or a vector.

4.2.1 Hutchinson trace estimator and unit vector estimator

The standard MC method to estimate the trace of the matrix inverse is due to Hutchinson [64]. It estimates the $Tr(A^{-1})$ by averaging s quadratures with random vectors $z_j \in Z_2^N = \{z(i) = \pm 1 \text{ with probability } 0.5\}$,

$$T_{Z_2}(A^{-1}) = \frac{1}{s}\sum_{j=1}^{s} z_j^T A^{-1} z_j. \tag{4.1}$$

The variance of this method is given by

$$Var(T_{Z_2}(A^{-1})) = \frac{2}{s}\|A^{-1}\|_F^2 - \frac{2}{s}\sum_{i=1}^{N}\|D_i\|^2. \tag{4.2}$$

The variance of this trace estimator is proven to be minimum over all vectors with real entries [64]. The confidence interval of a MC method reduces as $O(\sqrt{Var(T_{Z_2}(A^{-1}))})$ for the given matrix.

The unit vector estimator uniformly samples s vectors from the orthocanonical basis $\{e_1, \ldots, e_N\}$ [6],

$$T_{e_i}(A^{-1}) = \frac{N}{s}\sum_{j=1}^{s} e_{i_j}^T A^{-1} e_{i_j}, \tag{4.3}$$

where i_j are the random indices. The variance of the unit vector estimator is given by

$$Var(T_{e_i}(A^{-1})) = \frac{N^2}{s} Var(D). \tag{4.4}$$

The variance of the Hutchinson method depends on the magnitude of the off-diagonal elements. It converges in one step for diagonal matrices and rapidly if A^{-1} is highly diagonal dominant. On the other hand, the variance of the unit vector estimator depends only on the variance of the diagonal elements. It converges in one step if the diagonal elements are all the same and rapidly if the diagonal elements are similar. Thus, the method of choice depends on the particular matrix.

4.2.2 Reducing stochastic variance through matrix approximations

Given an approximation $Z \approx A$, for which Z^{-1} and $Tr(Z^{-1})$ are easily computable, we can decompose

$$Tr(A^{-1}) = Tr(Z^{-1}) + Tr(E), \tag{4.5}$$

where $E = A^{-1} - Z^{-1}$. We hope that by applying the MC methods on E instead on A^{-1}, the variance of the underlying trace estimator, in (4.2) or (4.4), can be reduced, thereby accelerating the convergence of MC. Among many ways to obtain a Z, we focus on the following two.

The first approach is when $Z^{-1} = (LU)^{-1}$, where the $L,\ U$ matrices stem from an incomplete LU (ILU) factorization of A, one of the most commonly used preconditioners. If the ILU is sufficiently accurate, then $M = diag(Z^{-1})$ may be a good approximation to D. To obtain the vector M without computing the entire Z^{-1}, we can use an algorithm described in [37]. This algorithm requires the computation of only those entries Z^{-1}_{ij} for which L_{ij} or $U_{ij} \neq 0$. If the L, U factors are sufficiently sparse or structured, this computation can be performed efficiently (see [82] for an example in the symmetric case).

The second approach is a low rank approximation $Z^{-1} = V\Lambda^{-1}U^T$, where Λ is a diagonal matrix with the $k \ll N$ smallest singular values of A, and U and V are the cor-

responding left and right singular vectors. Eigenvalues and eigenvectors can be used instead but we do not consider it in this research. This subspace can be obtained directly by an iterative eigensolver [135, 136] as a preprocessing step. The cost of this procedure is relatively small since the singular space is not needed in high accuracy. Alternatively, the space can be approximated by methods such as eigCG [119] or eigBiCG [1] while solving linear systems of equations during the MC method. This incremental approach adds only minimal overhead to the iterative linear solver but it cannot compute as many and as good quality singular vectors as the first approach. The quality of the approximation of A^{-1} by Z^{-1} depends on the separation of the computed singular space. Therefore, depending on the matrix, more singular triplets may be needed for a good low rank approximation. On the other hand, this space can also be used to deflate and thus accelerate subsequent linear systems. Once the singular space is computed, each diagonal element can be obtained with a single inner product of short vectors, $M_i = V(i,:)^T U(i,:)/\lambda_i$, and thus it is computationally inexpensive. Finally, the incremental SVD approach requires some special algorithmic attention during our algorithm, which will be pointed out later.

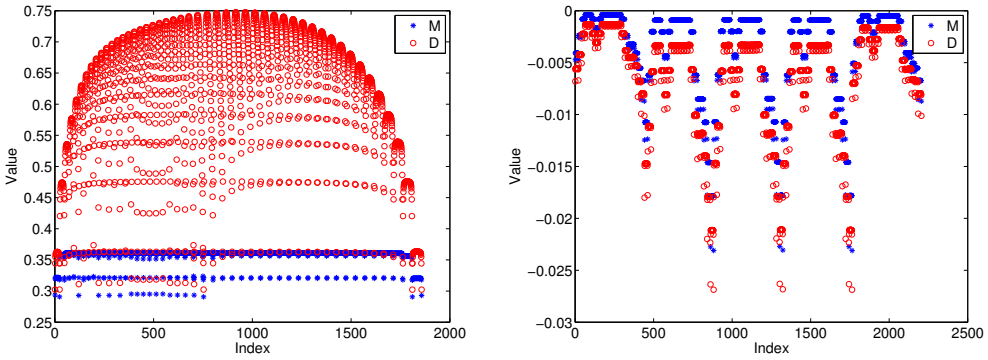
A computationally inexpensive, albeit less accurate approach for computing an approximation M is based on variational bounds on the entries of A^{-1} [12, 102]. Upper and lower bounds on the i-th diagonal entry A_{ii}^{-1} are derived inexpensively since they only depend on estimates of the smallest and largest algebraic eigenvalues, λ_1, λ_N, and the entries of A. The bounds apply to both symmetric and unsymmetric matrices. For the case of a real symmetric A, we have [102],

$$\frac{1}{\lambda_N} + \frac{(\lambda_N - A_{ii})^2}{\lambda_N(\lambda_N A_{ii} - s_{ii})} \leq (A^{-1})_{ii} \leq \frac{1}{\lambda_1} - \frac{(A_{ii} - \lambda_1)^2}{\lambda_1(s_{ii} - \lambda_1 A_{ii})}, \tag{4.6}$$

where $s_{ij} = \sum_{k=1}^{n} A_{ik}A_{kj}$. However the bounds in (4.6) will not be sharp especially the upper bound [90] and the error in the approximation can be large.

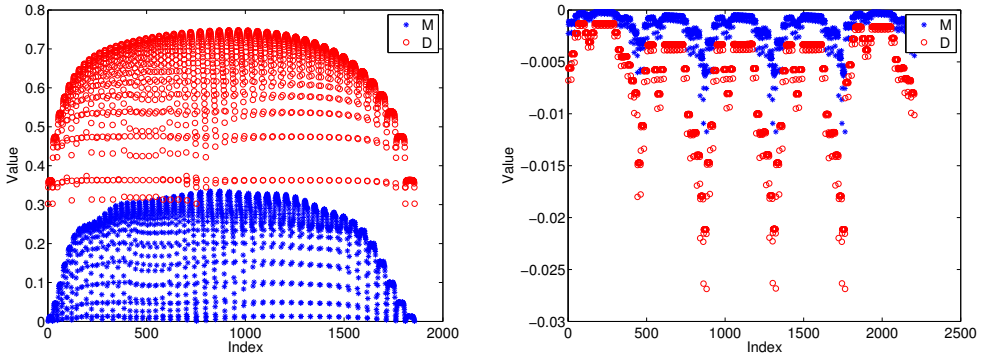
In general, unless Z^{-1} is highly accurate, we do not expect $Tr(M)$ to be close to $Tr(A^{-1})$. However, the patterns of M and D often show some correlation. We demon-

strate this for two example matrices, delsq50 and orsreg2205, in Figures 4.1 and 4.2 using ILU and SVD respectively. For matrix orsreg2205, both the ILU and SVD approaches return an approximate diagonal M which captures the pattern of D very well, with the M returned by ILU being slightly better than the one from SVD. For matrix delsq50, SVD clearly captures the pattern of D better than what ILU does. As in preconditioning for linear systems of equations, the appropriate approximation technique depends on the given matrix.



(a) M does not captures pattern of D (b) M quite close to D and captures pattern

Figure 4.1: The pattern correlation between the diagonals of A^{-1} and its approximation Z^{-1} computed by ILU(0) on matrices (a) delsq50 and (b) orsreg2205. delsq50 is created in MATLAB by `delsq(numgrid('S',50))`.



(a) M not close to D but captures pattern (b) M not close to D but captures pattern

Figure 4.2: The pattern correlation between the diagonals of A^{-1} and its SVD approximation Z^{-1} computed from the 20 smallest singular triplets of A on matrices (a) delsq50 and (b) orsreg2205. delsq50 is created in MATLAB by `delsq(numgrid('S',50))`.

4.2.3 Comparison of different MC methods and discussion on importance sampling

Based on (4.2–4.4), we express the variance of the trace estimators $T_{Z_2}(E)$ and $T_{e_i}(E)$ as follows:

$$Var(T_{Z_2}(E)) = \frac{2}{s}\|E\|_F^2 - \frac{2}{s}\sum_{i=1}^{N}\|diag(E)\|^2, \tag{4.7}$$

$$Var(T_{e_i}(E)) = \frac{N^2}{s}Var(diag(E)). \tag{4.8}$$

Figures 4.1 and 4.2 show there is potential for the variances of $T_{Z_2}(E)$ or $T_{e_i}(E)$ to be smaller than those of $T_{Z_2}(A^{-1})$ and $T_{e_i}(A^{-1})$. However, for a given matrix, we must gauge which MC method would be better, and whether the variances need further improvement.

The estimator $T_{e_i}(E)$ has the interesting property that if $M = D + c$, where c is a constant, then its variance in (4.8) is zero and we obtain the correct trace in one step. Although we cannot expect this in practice, it means that the shift observed between M and D in Figure 4.2(a) should not affect the effectiveness of $T_{e_i}(E)$.

On the other hand, $T_{e_i}(E)$ fails to identify correlations of the form $M = cD$. For such cases, importance sampling is preferred, where M plays the role of a new distribution simulating the distribution of D. Assume that both D and M have been shifted by the same shift so that $M_i > 0$ if $D_i > 0,\ i = 1, \ldots, N$. To transform M into a probability mass function, let $G = \frac{1}{Tr(M)}M$. To obtain an estimator of the trace of D with importance sampling, we replace the uniform sampling of D_i values with sampling with probability G_i [88]. Then, instead of (4.3), the importance sampling estimator is:

$$T_{IS}(D) = \frac{N}{s}\sum_{j=1}^{s} D_{i_j}\frac{\frac{1}{N}}{G_{i_j}} = \frac{Tr(M)}{s}\sum_{j=1}^{s}\frac{D_{i_j}}{M_{i_j}}. \tag{4.9}$$

When $M = cD$, the variance of $T_{IS}(D)$ is zero and it finds the trace in one step. However, it completely fails to identify shift correlations. In general D and M have a more complex relationship that neither $T_{IS}(D)$ or $T_{e_i}(E)$ can capture. This motivates our idea to explore general fitting models to approximate D.

4.3 Approximating The Trace of A Matrix Inverse

We seek to construct a function f, such that $D \approx f(M)$. Then we can decompose

$$Tr(A^{-1}) = Tr(D - f(M)) + Tr(f(M)). \tag{4.10}$$

$Tr(f(M))$ is trivially computed for a given f. A key difference between the approaches in (4.10) and (4.5) is that it is easier to find a fitting of the two vectors M and D if a strong pattern correlation exists between them than to fit the entire matrices A^{-1} and Z^{-1}. If $Tr(f(M))$ is a good approximation to $Tr(A^{-1})$ and its accuracy can be evaluated easily, we can directly use this quantity; Otherwise, we can apply the unit vector MC estimator to compute $Tr(E_{fit}) = Tr(D - f(M))$, provided that its variance

$$Var(T_{e_i}(E_{fit})) = \frac{N^2}{s} Var(E_{fit}) \tag{4.11}$$

is smaller than the variances in (4.2), (4.3), (4.7) and (4.8).

Algorithm 11 Basic algorithm for approximating $Tr(A^{-1})$

Input : $A \in \mathbb{R}^{n \times n}$
Output : $Tr(A^{-1})$ estimation and $Z \in \mathbb{R}^{n \times n}$

1: Compute $M = diag(Z^{-1})$, where Z^{-1} is an approximation to A^{-1} (Section 4.2.2)
2: Compute fitting sample S_{fit}, a set of k indices (Section 4.3.1)
3: Solve linear systems $D_i = e_i^T A^{-1} e_i, \ \forall i \in S_{fit}$ (Section 4.2.2)
4: Obtain a fitting model $f(M) \approx D$ by fitting $f(M(S_{fit}))$ to $D(S_{fit})$ (Section 4.3.2)
5: Compute refined trace approximation $T_{e_i}(E_{fit})$ using (4.3)
6: Estimate the relative trace error and, if needed, the variances for different MC methods (Section 4.4)

The basic description of the proposed estimator is outlined in Algorithm 11. First, our method computes an approximation M of D using one of the methods discussed in the previous section. If that method is based on a preconditioner or a low rank approximation, that preconditioner Z^{-1} could also be used to speed up the solution of linear systems in step 3. Second, it finds a fitting sample S_{fit}, a set of indices that should capture the

important distribution characteristics of D. Since we have no information about D, Section 4.3.1 discusses how to tackle this task by considering the distribution of M. Third, it computes the values of $D(S_{fit})$ by solving the corresponding linear systems using a preconditioned iterative solver. Since this is the computational bottleneck, the goal is to obtain good accuracy with far fewer fitting points than the number of vectors needed in MC. Fourth, it computes a fitting model that has sufficient predictive power to improve the diagonal approximation. This critical task is discussed in Section 4.3.2. Finally, since there are no a-posteriori bounds on the relative error of the trace, we use a combination of statistical approaches and heuristics to estimate it incrementally at every step. If the estimated error is not sufficiently small, our method can be followed by an MC method. For this reason, the algorithm also estimates dynamically the variances of the two different MC estimators (4.7) and (4.8) so that the one with the smallest variance is picked. The dynamic evaluation of error and variances is discussed in Section 4.4.

4.3.1 Point Identification Algorithm

We need to identify a set of indices S_{fit} based on which we can compute a function f that fits $f(M(S_{fit}))$ to $D(S_{fit})$ so that $|Tr(f(M)) - Tr(D)|$ is minimized. It is helpful to view $Tr(D)$ as an integral of some hypothetical one dimensional function which takes the values $D(i) = D_i$ on the discrete points $i = 1, \ldots, n$. Our goal is to approximate $\int_0^n D(x)dx$ with far fewer points than n. Because it is one dimensional, it may be surprising that Monte Carlo is the standard approach and not numerical integration. However, the effectiveness of any numerical quadrature rule relies on the smoothness of the function to be integrated, specifically on the magnitude of its higher order derivatives. In our case, our hypothetical function may have no smoothness or bounded derivatives. More practically, we deal with a set of discrete data points in D for which smoothness must be defined carefully. A typical definition of smoothness for discrete data is based on the Lipschitz continuity, i.e., $|D_i - D_j| < c|i - j|$ [121]. The lower the constant c, the smoother the set of data.

Hence, for a continuous function, the larger the magnitude of its first derivative, the less smooth its discretized points are. In the context of integrating an arbitrary D, the first order divided difference (i.e., the discretized first derivative) $D_i - D_{i+1}$ may be arbitrarily large in magnitude and therefore numerical integration may not work any better than random averaging. Even for matrices that model physically smooth phenomena (e.g., in PDEs), the matrix may be given in an ordering that does not preserve physical locality.

Consider a sorted permutation of the diagonal, $\hat{D} = sort(D)$. Obviously $Tr(D) = Tr(\hat{D})$, but $\hat{D}$ is monotonic and, based on the definition of Lipschitz discrete smoothness, it is maximally smooth, i.e.,

$$|\hat{D}(i) - \hat{D}(j)| \leq \Delta |i - j|, \tag{4.12}$$

for the smallest possible $\Delta \in \mathbb{R}_+$ among all permutations of D. Monotonicity implies that, in the absence of any additional information about the data, a simple trapezoidal rule minimizes the worst case integration error [72]. In addition, if we are allowed to choose the integration points sequentially, based on the points computed so far, then a much better average case error can be obtained [95, 124]. On the other hand, if bounds are known on the discrete smoothness of $\hat{D}$, better worst case error bounds can be established. Since D, however, is not available, we turn to its approximation M.

A close pattern correlation between M and D means that the elements of M should have a similar distribution as those of D, or that $\hat{M} = sort(M)$ should be similar to $\hat{D}$. Let us assume for the moment, that the index that sorts D to $\hat{D}$ sorts also M to $\hat{M}$. In other words, we assume a complete correlation between the monotonic orderings of M and D even though their values may differ. Then, we can work on the surrogate model $\hat{M}$ for which we can afford to identify the best quadrature points that yield the smallest error in $Tr(\hat{M})$. These will be the ideal points for computing the fitting function f.

Specifically, we need to select indices that capture the important distribution changes in $\hat{M}$. For example, identifying minimum and maximum elements of M sets the range of approximation for f and avoids extrapolation. We also look for entries in $\hat{M}$ that deviate

highly from their neighbors (where the first order divided difference of $\hat{M}$ and hopefully of $\hat{D}$ has a large value). This strategy has three advantages. First, between such indices the data is smooth so the integral $Tr(\hat{M})$ should be captured well by the trapezoidal rule. Second, and more important, we can obtain a more accurate fitting function f in a piecewise manner in intervals where $\hat{D}$ has similar behavior. Third, in this way we avoid picking points with the same value $\hat{M}_i = \hat{M}_j$ which can create problems in the fitting function.

The proposed index selection method is shown in Algorithm 12. Initially the set of sampled indices, $\hat{S}_{fit}$, includes the indices of the extrema of $\hat{M}$, 1 and N. Then, for every interval (i, j), with i, j consecutive indices in $\hat{S}_{fit}$, we find the index t from $i + 1$ to $j - 1$ that minimizes the trapezoidal rule error for computing $Tr(\hat{M}(i : j))$:

$$\underset{i<t<j,\ t\in\mathbb{Z}}{\text{argmin}} \left(|(\hat{M}(i)-\hat{M}(j))*(i-j)-(\hat{M}(i)-\hat{M}(t))*(i-t)-(\hat{M}(t)-\hat{M}(j))*(t-j)|\right). \quad (4.13)$$

The process continues until it reaches a maximum number of sampling points or until the maximum error over all intervals decreases by a fixed value, say 0.001. The value of this threshold depends on how close the patterns of M and D are correlated, a question we had postponed and we discuss next.

The points selected by the previous algorithm minimize the error for computing $Tr(M)$ with only $|\hat{S}_{fit}|$ points. How good are these points for fitting $\hat{M}$ to $\hat{D}$ and thus obtaining a small error in $Tr(D)$? Consider the M and D as discrete functions from $\{1, \ldots, n\} \to \mathbb{R}$ and assume they are bijective in their ranges. Let $G, J : \{1, \ldots, n\} \to \{1, \ldots, n\}$ be the two permutation functions that sort M to $\hat{M}$ and D to $\hat{D}$, i.e., $\hat{M}(i) = M(J(i))$ and $\hat{D}(i) = D(G(i))$. Our method relies on the assumption that the distribution of $\hat{M}$ captures the distribution of $\hat{D}$, or that there exists a smooth function f that can be approximated well with a low degree polynomial that fits $\hat{D}(i) = f(\hat{M}(i))$. For any i, there are two indices

$m = G(i), k = J(i)$ with $m \neq k$ in general that satisfy,

$$D(m) = D(G(i)) = \hat{D}(i) = f(\hat{M}(i)) = f(M(J(i))) = f(M(k)).$$

Our algorithm first picks an index i for $\hat{M}$ and derives the original index $k = J(i)$ in M. Since the permutation g is unknown, we do not know the $D(m)$ that should be matched with $M(k)$. Thus, the algorithm makes the simplifying assumption that $g = s$ and computes $D(k)$ instead. However, this corresponds to a different $\hat{D}(j)$, where $G(j) = k$. Therefore, using (4.12), we can bound the error for this mismatch by

$$|\hat{D}(i) - \hat{D}(j)| \leq \Delta|i - j| = \Delta|J^{-1}(k) - G^{-1}(k)|. \tag{4.14}$$

This implies that as long as the permutations J and G are locally similar, or in other words they do not shuffle the same index of M and D too far from each other, the error is small and therefore the fitting should work well. On the other hand, if M is a random permutation of D, there is no good fitting, even though $\hat{M} = \hat{D}$. This implies that the traces of $f(M)$ and D may be very close but not their individual elements. It also means that its variance of E_{fit} may not always be small (see the examples in the following section as well as in Section 4.4.2).

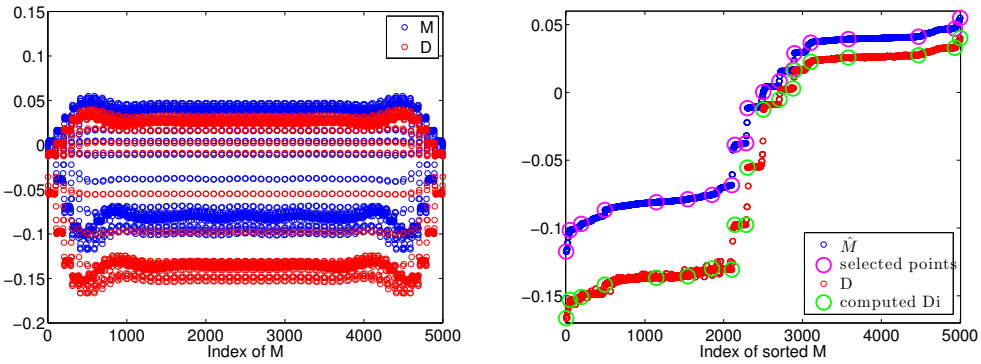
In practice, since we do not know the permutation G, it is possible that some flat area of $\hat{M}$ is associated with important changes in $\hat{D}$. To alleviate the effect of a possible local pattern mismatch, we instead empirically insert the midpoint of the current largest interval every k samples (we choose $k = 5$ in lines 11–14). Returning to the choice of threshold in the algorithm, we see that going below a threshold helps the accuracy of $Tr(M)$ but not necessarily of $Tr(D)$. The reason is that the greedy point selection strategy becomes less effective in determining the local mismatching patterns. Therefore, we terminate searching for a new index based on (4.13) if the maximum error is less than 0.001. We found this threshold to be sufficient in our experiments. If the maximum

required samples have not been generated, we continue by simply bisecting the largest intervals until $maxPts$ is reached (in lines 16–18). On exit, we compute S_{fit} which maps $\hat{S}_{fit}$ to the original unsorted ordering.

Algorithm 12 Point identification algorithm based on the trapezoidal rule

Input : $M \in \mathbb{R}^n$ and $maxPts$
Output : $\hat{S}_{fit}$: desired sampling index set
1: [$\hat{M}$, J] = sort(M)
2: $numSamples = 1$, $initErr = tempErr = |Tr(\hat{M}) - (\hat{M}(1) + \hat{M}(N))\frac{N-1}{2}|$
3: Add $1, N$ into $\hat{S}_{fit}$ and push interval $(1, N)$ and its $tempErr$ in the queue Q including interval and error
4: **while** $numSamples < maxPts$ and $tempErr > 0.001 * initErr$ **do**
5: Pop interval (L, R) with largest error from Q
6: **for** $k = L + 1 : R - 1$ **do**
7: Use (4.13) to find a bisecting index t
8: **end for**
9: Add t into $\hat{S}_{fit}$, $numSamples = numSamples + 1$
10: Push intervals (L, t) and (t, R) each with their corresponding $tempErr$ in Q
11: **if** $numSamples$ is a multiple of 5 **then**
12: Insert one midpoint index into largest interval in $\hat{S}_{fit}$, $numSamples = numSamples + 1$
13: Insert its corresponding left and right intervals in Q
14: **end if**
15: **end while**
16: **while** $numSamples < maxPts$ **do**
17: Insert middle index of the largest interval into S, $numSamples = numSamples + 1$
18: **end while**
19: Return S_{fit} indices in original ordering, such that $\hat{S}_{fit} = J(S_{fit})$

A typical sampling result produced by our method is shown in Figure 4.3. The graph on the left shows the diagonals M and D plotted in their original order. The pattern correlation between them is clear, but it is less straightforward how to pick the fitting points. Figure 4.3(b) shows both diagonals plotted with the order of indices that sorts $\hat{M}$, i.e., $M(J), D(J)$. The points picked by our algorithm based on $\hat{M}$ correspond to an almost monotonic sequence of points in $D(J)$. The associated indices can then be used to identify a suitable set of entries in D to perform the numerical integration.



(a) Distribution of elements of M and D (b) Sample points for fitting

Figure 4.3: A typical example to show our sampling strategy based on the pattern correlation of M and D. M is computed by 20 singular vectors of matrix RDB5000. In the right figure, the magenta and green circles denote the sample points associated with the sampling indices S_{fit} in $\hat{M}$ and D.

4.3.2 Two Fitting Models

Next we construct a fitting model that minimizes $\|f(M(S_{fit})) - D(S_{fit})\|$. The fitting model must have sufficient predictive power with only a small number of points and it should avoid oscillating behavior since we work on the monotonic sequence $\hat{M}$ which we assume correlates well with $\hat{D}$. Therefore, we consider a linear model and a piecewise polynomial model.

The MC methods in (4.5) and (4.10) can resolve the trace when $D = M + c$ while importance sampling can resolve the trace when $D = cM$. To combine these, we first use a linear model, $y = bM + c$. We determine the parameters b, c by a least squares fitting, $\text{argmin}_{b,c\in\mathbb{R}}\|D(S_{fit}) - (bM(S_{fit}) + c)\|_2$. The linear model may be simple but avoids the large oscillations of higher degree polynomials, and in many cases it is quite effective in improving the accuracy of the trace estimation and reducing the variance of the diagonal elements of E_{fit}. The linear fitting algorithm is described in Algorithm 13. Figure 4.4(a) shows the fitting result on the example matrix of the previous section, in the original order of D.

The linear model preserves the shape of M, and therefore relies exclusively on the quality of M. To take advantage of our premise that the distribution $\hat{M}$ approximates well

Algorithm 13 Linear least squares fitting model for approximating $Tr(A^{-1})$.

Input : $A \in \mathbb{R}^{n \times n}$
Output : $Tr(A^{-1})$ estimation: T_f
1: Compute M using ILU or Eigendecompostion or SVD on A
2: Call Algorithm 12 to compute sample set S_{fit}.
3: Find $[b, c] = \text{argmin}\|D(S_{fit}) - (bM(S_{fit}) + c)\|_2$.
4: Compute trace approximation $T_f = \sum_{i=1}^{N}(b * M + c)$

the distribution of $\hat{D}$, our next fitting model is the Piecewise Cubic Hermite Spline Interpolation (PCHIP). It was proposed in [40] to construct a visually pleasing monotone piecewise cubic interpolant to monotone data. The PCHIP interpolant is only affected locally by changes in the data and, most importantly, it preserves the shape of the data and respects monotonicity. Therefore, we work on $\hat{M}$ and the indices $\hat{S}_{fit} = [1 = s_1, s_2, \ldots, s_{k-1}, s_k = N]$ which are given in an order such that $\alpha = \hat{M}(s_1) \leq \hat{M}(s_2) \leq \cdots \leq \hat{M}(s_k) = \beta$ is a partition of the interval $[\alpha, \beta]$. An index s_i corresponds to the index $J^{-1}(s_i)$ in the original ordering of M, where J is from Algorithm 12 and J^{-1} denotes the inverse mapping of the sorted list. Thus, for each s_i we compute $D(J^{-1}(s_i)), i = 1, \ldots, k$, and we use PCHIP to construct a piecewise cubic function such that,

$$p(\hat{M}(s_i)) = D(J^{-1}(s_i)), \quad i = 1, 2, \ldots, k. \tag{4.15}$$

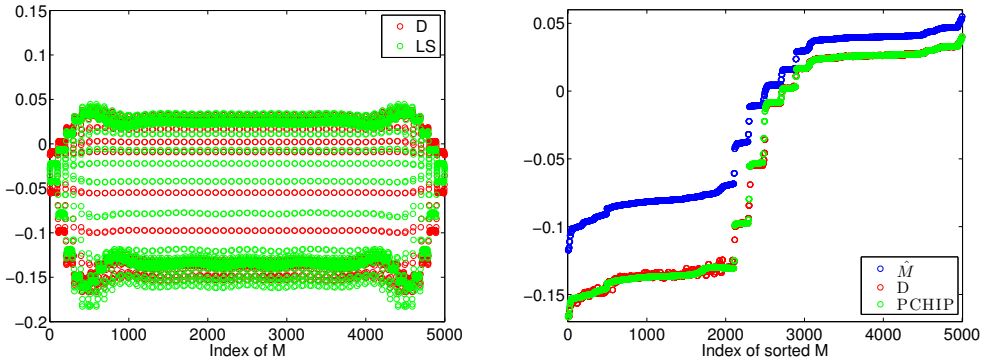
Notice that $p(x)$ will be monotone in the subintervals where the fitting points $D(J^{-1}(s_i))$ are also monotone. Therefore, as long as M is close to D in the sense of (4.14), integration of $p(x)$ will be very accurate.

The PCHIP model is given in Algorithm 14. The first two steps are the same as in Algorithm 13. In step 3, we apply the function **unique** to remove the duplicate elements of $\hat{M}(\hat{S}_{fit}))$ to produce a sequence of unique values as required by PCHIP. This yields a subset of the indices, $\hat{S}'_{fit}$, which is mapped to original indices as $I = J^{-1}(\hat{S}'_{fit})$ to be used in PCHIP.

Figure 4.4(b) shows the result of fitting on the RDB5000 example of the previous fig-

Algorithm 14 PCHIP fitting model for approximating $Tr(A^{-1})$.

Input : $A \in \mathbb{R}^{n \times n}$
Output : $Tr(A^{-1})$ estimation: T_f
% Steps 1 and 2 are the same as in Algorithm 13
3: Remove duplicates: $\hat{S}'_{fit} = \textbf{unique}(\hat{M}(\hat{S}_{fit}))$, $I = J^{-1}(\hat{S}'_{fit})$
4: Apply PCHIP to fit $p(\hat{M}(I)) = D(I)$ and obtain a polynomial $p(M) \approx D$
5: Compute trace approximation $T_f = \sum_{i=1}^{N} p(M)$



(a) Linear LS model in original order (b) PCHIP model in sorted order

Figure 4.4: Fitting results of the matrix RDB5000 in original order and sorted order with linear LS model and PCHIP model.

ures. Table 4.1 compares the relative error for the trace, as well as the variances of D, $D - M$ and $D - p(M)$ when using the linear LS and PCHIP models on two test matrices, OLM5000 and KUU. In both cases, the two models can provide a trace estimate of surprising relative accuracy $O(1e-2)$ with only 20 fitting points. In addition, for matrix OLM5000, the standard deviation of MC on $D - p(M)$ is reduced by a factor of 10 compared to that of MC on D (a speedup of 100 in terms of samples). However, for matrix KUU, the standard deviation of MC on $D - p(M)$ does not improve much over MC on D. This reflects the discussion in Section 4.3.1, and suggests that our method can serve as a standalone kernel for giving a fast trace estimate. In addition, in some cases the method can reduce the variance significantly to accelerate a second stage MC.

Table 4.1: Comparing the trace relative error and variances in (4.4), (4.8) and (4.11) for matrices OLM5000 and KUU, using the linear LS and PCHIP models with 20 fitting points each. The traces of matrix OLM5000 and KUU are $Tr(A^{-1}) = -5.0848e+02$ and $3.6187e+03$, respectively.

Matrix	OLM5000			KUU		
Model	M	LS	PCHIP	M	LS	PCHIP
$Relative\ error$	9.6e-01	2.2e-02	1.4e-02	5.8e-01	1.2e-02	1.4e-02
$Std(T_{e_i}(A^{-1}))$	1.1e+02	1.1e+02	1.1e+02	2.8e+02	2.8e+02	2.8e+02
$Std(T_{e_i}(D-M))$	1.1e+02	1.1e+02	1.1e+02	1.4e+02	1.4e+02	1.4e+02
$Std(T_{e_i}(D-p(M)))$	–	2.0e+01	1.7e+01	–	1.6e+02	1.6e+02

4.4 Dynamic Evaluation of Variance and Relative Trace Error

Since there are no a-posteriori bounds for the accuracy of our results, we develop methods that use the information from the solution of the linear systems to incrementally estimate the trace error and the variances of the resulting approximations. This approach is also useful when M is updated with more left and right eigenvectors or singular vectors obtained from the solution of additional linear systems.

4.4.1 Dynamic Variance Evaluation

To decide which MC method we should use after the fitting stage or even whether it is beneficial to use the fitting process for variance reduction, we monitor incrementally the variances $Var(T_{e_i}(A^{-1}))$, $Var(T_{e_i}(E_{fit}))$, $Var(T_{Z_2}(A^{-1}))$, and $Var(T_{Z_2}(E))$, with the aid of the cross-validation technique [54].

Our training set is the fitting sample set $D(S_{fit})$, while our test set $D(S_{mc})$ is a small random set which is independent of the fitting sample set. If we want to combine our method with MC, eventually more samples need to be computed, and thus we can precompute a certain number of them as the test set $D(S_{mc})$. We have used the holdout method [5], a single train-and-test experiment for some data splitting strategy since the fitting sample set is fixed.

To compute $D(S_{fit})$ or $D(S_{mc})$, a linear system with multiple right hand sides is solved

as follows:

$$A_{ii}^{-1} = e_i^T x_i, \quad Ax_i = e_i, \ \forall i \in S_{fit} \cup S_{mc}. \tag{4.16}$$

The computed column vectors x_i can be used to estimate the Frobenius norm of both A^{-1} and $E = A^{-1} - Z^{-1}$ [48, 71]. Then $Var(T_{Z_2}(A^{-1}))$ and $Var(T_{Z_2}(E))$ can be estimated as follows:

$$Var(T_{Z_2}(A^{-1})) \approx \frac{2N}{s^2} \sum_i (\|x_i\|^2 - |D_i|^2), \ \forall i \in S_{fit} \cup S_{mc}, \tag{4.17}$$

$$Var(T_{Z_2}(E)) \approx \frac{2N}{s^2} \sum_i (\|E(:,i)\|^2 - |E(i,i)|^2), \ \forall i \in S_{fit} \cup S_{mc}, \tag{4.18}$$

where $E(:,i) = Ee_i$. Simultaneously, based on the sampled diagonal elements A_{ii}^{-1}, we can also update the evaluation of $Var(T_{e_i}(E_{fit}))$, $Var(T_{e_i}(E))$ and $Var(T_{e_i}(A^{-1}))$. Here we only show the computation of the unbiased variance estimation for $Var(T_{e_i}(E_{fit}))$ by:

$$Var(T_{e_i}(E_{fit})) \approx \frac{N^2}{s-1} Var(E_{fit}(S_{mc})). \tag{4.19}$$

Note that S_{fit} should not be used for estimating the variance of the unit vector MC estimator since these sample points are exact roots of the PCHIP function.

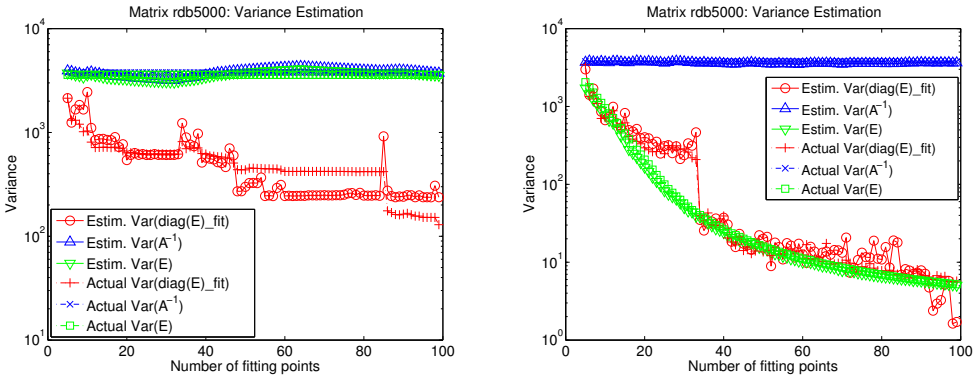
Algorithm 15 Dynamic variance evaluation algorithm for estimating variances of different MC methods

Input : $A \in \mathbb{R}^{n \times n}$
Output : $Tr(A^{-1})$ estimation and variances estimations of various MC methods

1: Initialize $maxPts$, S_{fit}, S_{mc}
2: **if** M is computed using ILU on A **then**
3: Compute the approximation M
4: **end if**
5: Generate random index set S_{mc} without replacement and compute MC samples $D(S_{mc})$
6: **for** $i = 5 : 1 : maxPts$ **do**
7: **if** M is computed using Eigendecompostion or SVD on A **then**
8: Update M with $2 * i$ number of left and right eigenpairs or singular triplets
9: **end if**
10: Call Algorithm 12 to find more indices so that S_{fit} has i fitting points
11: Call Algorithms 13 or 14 to update approximation of $Tr(A^{-1})$
12: Estimate variances of different MC methods based on (4.17), (4.18) and (4.19)
13: **end for**

We implement the dynamic variance evaluation scheme in Algorithm 15. In lines 2-4 and 7-9, the approximation M can be computed using an ILU factorization at the beginning of the procedure or be updated with increasing number of singular triplets or eigenpairs. Note that if M is obtained by a partial eigendecomposition or SVD, the updated M is different in two consecutive steps, and thus Algorithm 12 will return a slightly different index set S_{fit} which may not be incremental. This may not provide a consistent improvement of the relative trace error during the fitting progress. In line 10 of the algorithm, we force the points to be incremental between steps i and $i+1$ as follows; we generate the entire set $S_{fit}^{(i+1)}$ and remove the indices that lie the closest to the previous index set $S_{fit}^{(i)}$. The remaining index set is incorporated into $S_{fit}^{(i)}$. This simple scheme works quite well experimentally.

Figure 4.5 shows how the actual and the estimated variances for three MC methods match for the test matrix RDB5000. In addition, the relative difference between different MC methods becomes clear after only a few points which facilitates not only the proper choice of MC method but also an early decision to stop if further fitting is not beneficial. In the numerical experiments section we show that these results are typical for matrices from a wide variety of applications.

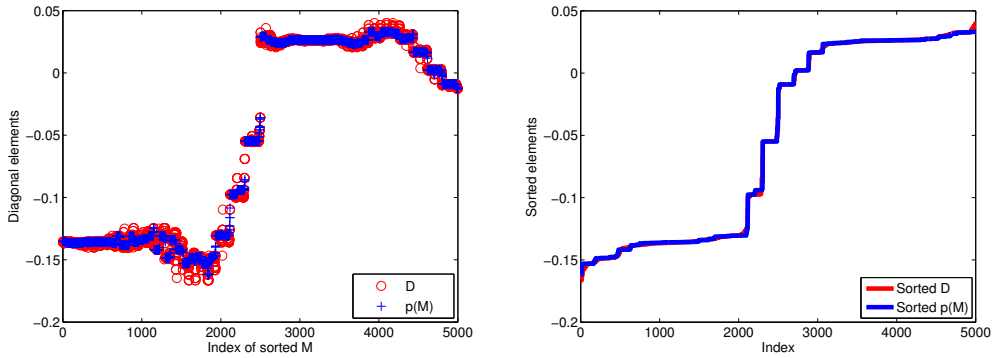


(a) Dynamic variance estimation with ILU (b) Dynamic variance estimation with SVD

Figure 4.5: Comparing estimated variances and actual variances of unit vector on E_{fit} (denoted as $Var(diag(E)\_fit)$) and Rademacher vector on A^{-1} (denoted as $Var(A^{-1})$) and E (denoted as $Var(E)$) of the matrix RDB5000 with ILU and SVD respectively.

4.4.2 Monitoring Relative Trace Error

As discussed in Section 4.3.1 and showed in Table 4.1, the variance of MC on the vector E_{fit} may not be reduced if the sorting permutations of $\hat{M}$ and $\hat{D}$ are dissimilar. Even in such cases, however, our fitting method can provide a good trace estimation with only a small number of samples. We further investigate this by comparing the elements of D and $p(M)$ with different orderings. Figure 4.6(a) shows the elements of $p(M(J))$ and $D(J)$, i.e., with respect to the order of $\hat{M}$. It illustrates that although $p(M)$ captures the pattern of D, the order of its elements does not correspond exactly to that of D; hence the small reduction in $Var(T_{e_i}(E_{fit}))$. However, Figure 4.6(b) reveals that the distributions of the sorted $p(M)$ and the sorted D almost coincide; hence, the two integrals $Tr(p(M))$ and $Tr(D)$ are very close.



(a) Comparing D and $p(M)$ in order of $\hat{M}$ (b) Comparing sorted D and sorted $p(M)$

Figure 4.6: Compare D and $p(M)$ of the matrix RDB5000 in different order where M is computed by ILU.

The main obstacle for using our method as a standalone kernel for trace estimation is that there is no known way to measure or bound the relative trace error. Resorting to the confidence interval computed by the variance of a MC estimator is pessimistic as our results show (see also [6, 104]). We note the similarity to the much smaller error obtained in the average case of integrating monotonic functions with adaptive quadratures versus the worst case known bounds [72, 95].

Motivated by these previous research results, we show how to develop practical criteria to monitor the relative trace error in our Algorithms. Suppose at each step of the fitting process in Algorithm 15 we collect a sequence of trace estimates $T_i, i \in [1, maxPts]$. Consider the trace estimations in two successive steps,

$$\frac{|T_i - T_{i+1}|}{|T_{i+1}|} = \frac{|(T_i - Tr(D)) - (T_{i+1} - Tr(D))|}{|T_{i+1}|} = \frac{|E_i - E_{i+1}|}{|T_{i+1}|} \leq \frac{2max(|E_i|, |E_{i+1}|)}{|T_{i+1}|} \approx \frac{2max(|E_i|, |E_{i+1}|)}{|Tr(D)|}. \quad (4.20)$$

As long as T_i converges with more fitting points, the relative difference of two successive trace estimations can serve as an approximation to the relative error. However, when the global pattern provided by M and $p(M)$ is not fully matched to that of D, convergence of T_i stagnates until enough points have been added to resolve the various local patterns. To determine whether the current relative trace error estimation can be trusted, we present our second heuristic by considering the error bound of our fitting models.

When approximating $f(\hat{M})$ with $p(\hat{M})$, the PCHIP Hermite cubic splines with k points on the interval $[\alpha, \beta]$, the bound on the error $E(\hat{M}) = f(\hat{M}) - p(\hat{M})$ is given by [87],

$$|E(\hat{M})| \leq \frac{1}{384} h^4 \|f^{(4)}\|_{\infty,[\alpha,\beta]} , \quad (4.21)$$

where $h = \frac{\beta - \alpha}{k}$, and $\|f^{(4)}\|_{\infty,[\alpha,\beta]}$ denotes the maximum value of the fourth derivative of f in the entire interval $[\alpha, \beta]$. Since $\|f^{(4)}\|_{\infty,[\alpha,\beta]}/384$ is a constant, in two successive fitting steps we have,

$$\frac{|E_i(\hat{M})|}{|E_{i+1}(\hat{M})|} \approx \frac{h_i^4}{h_{i+1}^4} = \frac{((\beta - \alpha)/k_i)^4}{((\beta - \alpha)/k_{i+1})^4} = (\frac{k_{i+1}}{k_i})^4, \quad (4.22)$$

where k_i and k_{i+1} are the number of fitting points in two consecutive steps. We can use (4.22) to estimate the maximum possible improvement between two consecutive trace errors. If the $i+1$ trace error estimate reduces over the i-th estimate by a factor of more than $(k_i/k_{i+1})^4$, we do not trust it.

One caveat is that (4.21) may not be tight since the same bound holds for each subin-

Algorithm 16 Dynamic relative trace error evaluation algorithm during the fitting process

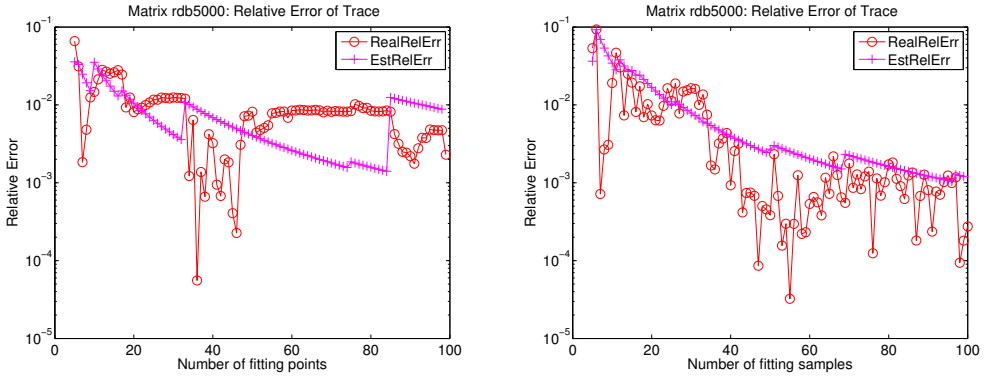
Input : $TraceFit$ from Alg. 15
Output : $TraceErr$ estimation
% $TraceErr(i)$ is defined only for $i > 5$ and we assume TraceErr(6) is well defined
1: **if** $i == 6$ **then**
2: $TraceErr(6) = |TraceFit(6) - TraceFit(5)|/|TraceFit(5)|$
3: **end if**
4: **if** $i > 6$ **then**
5: $TempTraceErr = |TraceFit(i) - TraceFit(i-1)|/|TraceFit(i)|$
6: **if** $(TempTraceErr/TraceErr(i-1) \geq ((i-1)/i)^4)$ **then**
7: $TraceErr(i) = TempTraceErr$
8: **else**
9: $TraceErr(i) = TraceErr(i-1) * ((i-1)/i)^{9/4}$
10: **end if**
11: **end if**

terval $[\hat{M}(s_j), \hat{M}(s_{j+1})]$. This means that a high derivative in one subinterval might dominate the bound in (4.21) but should not affect the error in other intervals. Therefore, convergence might not be fully dictated by (4.22). In practice, we found that the improvement ratio is between $O(k_i/k_{i+1})$ and $O((k_i/k_{i+1})^4)$. Therefore, if the current relative trace error estimate is determined not to be trusted, we may instead use $|E_{i+1}(\hat{M})| = (k_i/k_{i+1})^k |E_1(\hat{M})|,\ k \in [1, 4]$. The choice of k depends on the quality of M. In our experiments, we use the geometric mean of the four rates, yielding $k = 9/4$. Recall that the corresponding ratio in MC is $O(\sqrt{k_i/k_{i+1}})$, which is much slower than the proposed trace estimation method.

Algorithm 16 combines the two heuristics in (4.20) and (4.22) to dynamically monitor the relative trace error. It is called after step 12 of Algorithm 15. Figure 4.7 shows two examples of how our dynamic method provides reasonable estimates of the relative trace error.

4.5 Numerical Experiments

We run experiments on matrices that are sufficiently large to avoid problems of sampling from small spaces but can still be inverted to obtain the exact trace error. We select



(a) Monitoring relative trace error with ILU (b) Monitoring relative trace error with SVD

Figure 4.7: Two examples of monitoring relative trace error of the matrix RDB5000 with ILU and SVD respectively.

matrices RDB5000 and cfd1 from the University of Florida sparse matrix collection [28] and generate three test matrices from applications that appear in [12]. The Heatflow160 matrix is from the discretization of the linear heat flow problem using the simplest implicit finite difference method. The matrix Poisson150 is from 5-point central difference discretization of the 2D Poisson's equation on a square mesh. The VFH6 matrix is from the transverse vibration of a Vicsek fractal that is constructed self-similarly. We also use matrix matb5 which is a discretization of the Wilson Dirac operator on a 8^4 uniform lattice with 12 degrees of freedom at each node, using a mass near to critical.

Table 4.2 lists these matrices along with some of their basic properties. All experiments are conducted using MATLAB 2013a. The number of fitting points increases as $s = 5 : 100$. The approximation M is computed by ILU with parameters `type = ilutp` and `droptol = 1E-2`, or as a low rank approximation of $2s$ smallest singular vectors with accuracy `1E-6` (twice the number of fitting points at each step), or by the bounds on the diagonal.

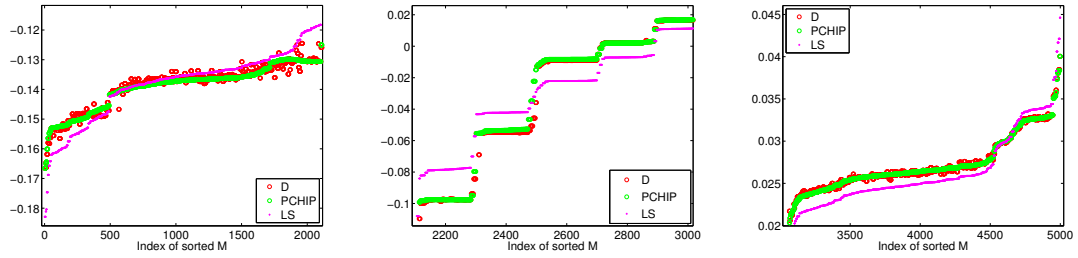
4.5.1 Effectiveness of the fitting models

In Figure 4.8, we divide the diagonal elements of the matrix RDB5000 into three contiguous sets and zoom in the details. We see that despite a good M, the linear LS model

Table 4.2: Basic information of the test matrices

Matrix	Order	nnz(A)	$\kappa(A)$	Application
RDB5000	5000	29600	1.7E3	computational fluid
cfd1	70656	1825580	1.8E7	computational fluid
Heatflow160	25600	127360	2.6E0	linear heat flow
Poisson150	22500	111900	1.3E4	computational fluid
VFH6	15625	46873	7.2E1	vicsek fractal
matb5	49152	2359296	8.2E4	lattice QCD

cannot scale the entire M onto D. The more flexible piecewise approach of PCHIP results in a much better fit.



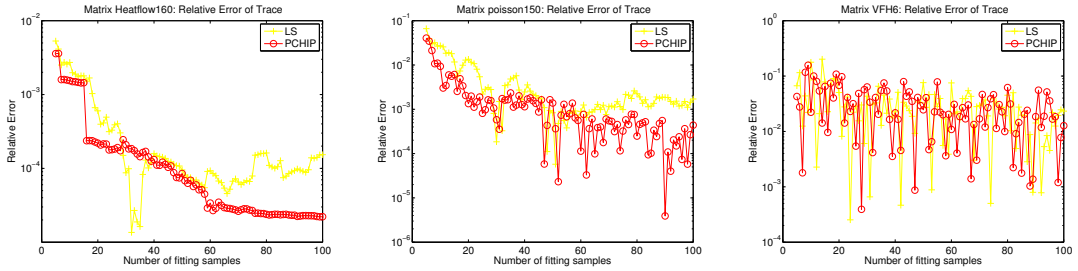
(a) Diagonal elements in (1,2000) (b) Diagonal elements in (2000,3000) (c) Diagonal elements in (3000,5000)

Figure 4.8: Comparing the linear LS model with the PCHIP model on RDB5000 matrix with M from SVD.

In Figure 4.9, we look at three matrices with M generated using the SVD. The PCHIP model typically has smaller relative trace error than the LS model. We also see that as more fitting points are sampled, the relative trace error of both models decreases significantly at early stages and slowly after a certain point. This relates to the quality of M, not of the model. Typically M will approximate the global pattern of D and the two can be matched well with only a few fitting points. But if the local patterns of M and D differ, a large number of fitting points will be required. This can be seen in Figure 4.10. Using the permutation of $\hat{M}$ for each case, we plot $\hat{M}$, D, and $p(\hat{M})$ using 100 fitting points. For the Heatflow160 matrix, $p(M)$ approximates D well everywhere except for the small leftmost part of the plot, which allows the relative error to reach below 10^{-4} before convergence

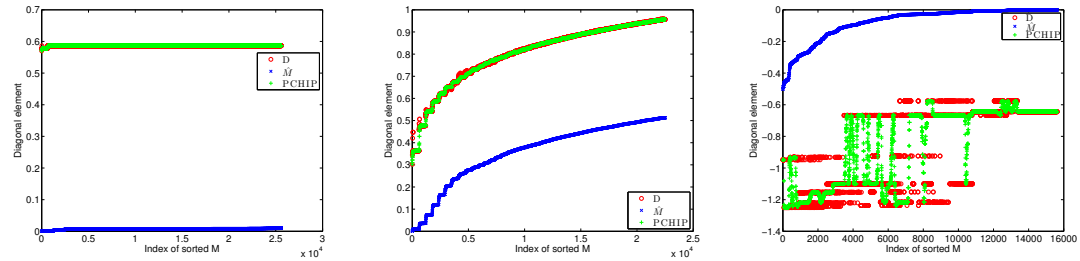
slows down (Figure 4.9). The behavior is similar for the Poisson150. The issue is more pronounced on matrix VFH6, where M and $p(M)$ capture the average location of D but completely miss the local pattern, which is reflected by a very slowly improving error in Figure 4.9.

We mention that the irregularity of the relative trace errors in Figure 4.9 relates to the variability of successive updates of M and of the sampling indices, especially when M is of lower quality.



(a) Matrix Heatflow160 (b) Matrix Poisson150 (c) Matrix VFH6

Figure 4.9: Comparing relative trace error between the LS model and the PCHIP model in three typical cases with SVD.



(a) Matrix Heatflow160 (b) Matrix Poisson150 (c) Matrix VFH6

Figure 4.10: Fitting results of three typical cases with the PCHIP model and 100 fitting points using SVD.

Figure 4.11 demonstrates that the PCHIP model has smaller actual variance for MC on E_{fit} than the LS model. Therefore, we only consider the PCHIP model in the rest of experiments.

4.5.2 Comparison between the fitting model and different MC methods

We address the question of whether the number of matrix inversions we spend on computing the fitting could have been used more efficiently in an MC method, specifically the Hutchinson method on A^{-1} and the Hutchinson method on E. In Table 4.3 we compare the relative trace error of the PCHIP model with 20 fitting points against the relative errors of the two MC methods as computed explicitly from their respective standard deviations in (4.2) and (4.7), with $s = 20$, divided by the actual trace of D.

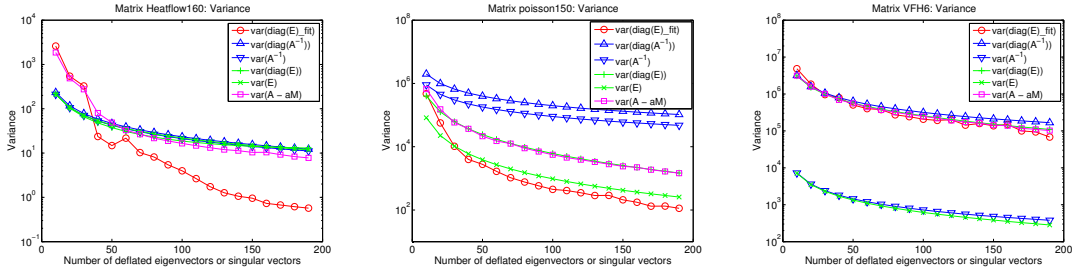
When M approximates D sufficiently well, the trace from the fitted diagonal is better, and for the ILU approximations far better, than if we just use the Hutchinson method on A^{-1} (the first column of results). Although MC on E exploits the ILU or SVD approximation of the entire matrix (not just the diagonal that our method uses), we see that it does not always improve on MC on A^{-1}, and in some cases (cfd1 with ILU) it is far worse. In contrast, our diagonal fitting typically improves on MC on E. The last column shows that even with an inexpensive diagonal approximation we obtain a similar or better error than MC on A^{-1}. The only exception is the matrix VFH6 where, as we saw earlier, M cannot capture the pattern of D. Even then, its error is close to the errors from the MC methods and, as we show next, the best method can be identified dynamically with only a small number of samples.

Table 4.3: Relative trace error from our PCHIP model and from the MC method on A^{-1} and on E (computed explicitly as the standard deviation with $s = 20$ from (4.2) and (4.7) divided by the actual trace).

		ILU		SVD		Bounds
Matrix	$T_{Z_2}(A^{-1})$	PCHIP	$T_{Z_2}(E)$	PCHIP	$T_{Z_2}(E)$	PCHIP
RDB5000	5.2E-2	8.1E-3	4.8E-2	4.1E-3	1.2E-2	5.3E-2
cfd1	1.3E-1	2.8E-2	8.2E+2	8.8E-3	1.8E-2	2.6E-2
Heatflow160	4.9E-4	1.6E-7	4.0E-5	2.0E-4	4.9E-4	3.5E-4
Poisson150	2.6E-2	2.3E-3	2.5E-2	1.4E-3	4.3E-3	8.3E-3
VFH6	3.2E-3	6.8E-5	3.1E-5	1.0E-2	3.2E-3	6.0E-2

If the user requires better trace accuracy than our fitting technique provides, we ex-

plore the performance of the diagonal fitting as a variance reduction for MC with unit vectors. We compute the actual values of $Var(T_{e_i}(E_{fit}))$, $Var(T_{e_i}(A^{-1}))$, $Var(T_{e_i}(E))$, $Var(T_{Z_2}(A^{-1}))$ and $Var(T_{Z_2}(E))$ for every step $s = 5 : 100$ and show results for three matrices in Figure 4.11. Note that the low rank approximation uses $2s$ singular vectors. As before, for Heatflow160, MC on E_{fit} performs much better than other MC methods, achieving about two orders reduction in variance. For Poisson150, MC on E_{fit} is slightly better compared to the Hutchinson method on E. In contrast, for VFH6, MC with unit vectors do not perform well regardless of the diagonal.



(a) Matrix Heatflow160 (b) Matrix Poisson150 (c) Matrix VFH6

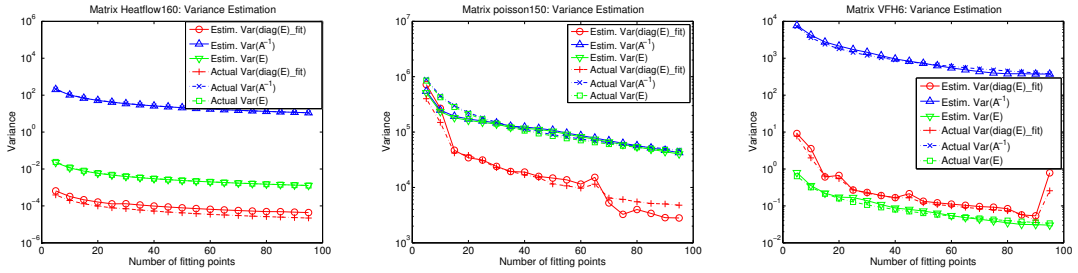
Figure 4.11: Comparing actual variances of different MC methods in three typical cases with SVD.

4.5.3 Dynamic evaluation of variance and relative trace error

The above results emphasize the importance of being able to assess quickly and accurately the relative differences between the variances of different methods as well as the trace error, so that we can decide whether to continue with fitting or which MC method to switch to. First we show the effectiveness of the dynamic variance evaluation algorithm for our fitting MC method on $(D - p(M))$ with unit vectors, and on A^{-1} and E with Rademacher vectors. Then, we evaluate our algorithm for estimating the relative trace error during the fitting process.

Figure 4.12 compares the estimated variances with the actual variances of the three MC methods when increasing the number of fitting points from 5 to 100. The approximation M is computed by using ILU. We can see that the estimated values of $Var(T_{Z_2}(A^{-1}))$

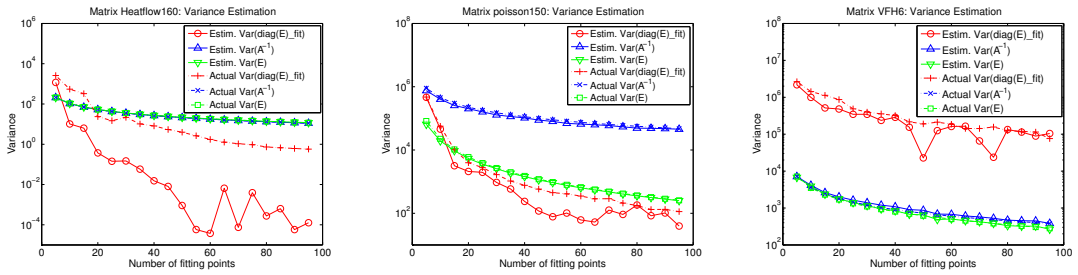
and $Var(T_{Z_2}(E))$ converge to the actual variances after only a few sample points. The estimated value of $Var(T_{e_i}(E_{fit}))$ gets close to and captures the trend of the the actual variance as the fitting samples increase. Nevertheless, the relative differences between the variances of the various MC methods are apparent almost immediately.



(a) Matrix Heatflow160 (b) Matrix Poisson150 (c) Matrix VFH6

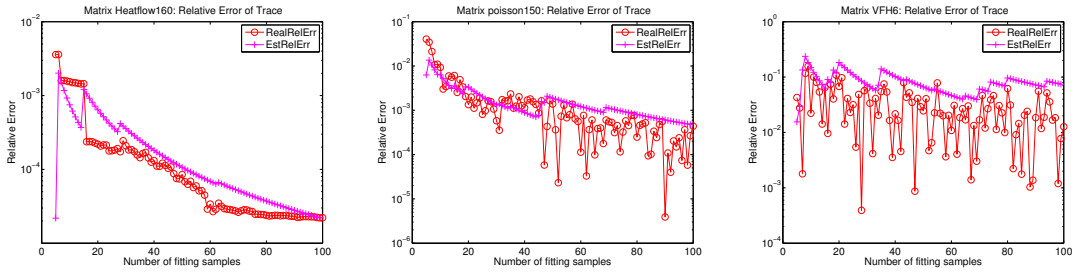
Figure 4.12: Comparing estimated variances and actual variances of three MC methods with ILU.

Figure 4.13 shows the same experiments when the approximation M is computed by SVD. Since M is updated each step, $Var(T_{Z_2}(E))$ and $Var(T_{e_i}(E_{fit}))$ change accordingly. As with ILU, $Var(T_{Z_2}(A^{-1}))$ and $Var(T_{Z_2}(E)$ can be estimated very well in a few steps. $Var(T_{e_i}(E_{fit}))$ could be underestimated but the relative variance difference between these MC methods becomes clear when the fitting points increase beyond 20. Thus we are able to determine whether the fitting process is beneficial as a variance reduction preprocessing and which is the best MC method to proceed with for the trace estimation.



(a) Matrix Heatflow160 (b) Matrix Poisson150 (c) Matrix VFH6

Figure 4.13: Comparing estimated variances and actual variances of three MC methods with SVD.



(a) Matrix Heatflow160 (b) Matrix Poisson150 (c) Matrix VFH6

Figure 4.14: Comparing estimated relative trace error with actual relative trace error with SVD.

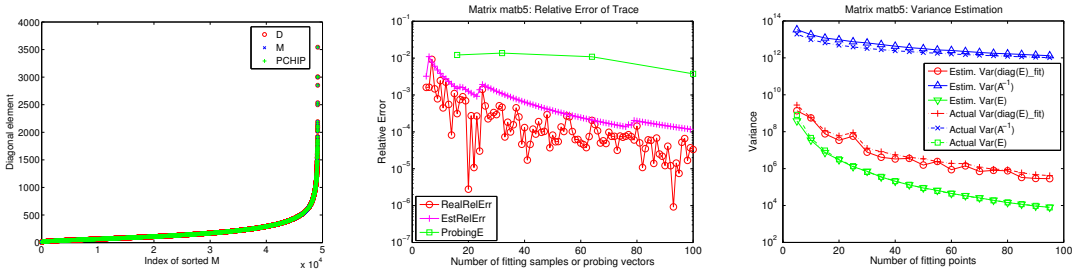
Figure 4.14 compares the estimated relative trace error with the actual one in the cases of Figure 4.13. We observe that the estimation is accurate as the fitting samples increase, even for cases such as VFH6 where the fitting process is not as successful. Moreover, because our algorithm is based on upper bounds on the error of a piecewise cubic polynomial, the actual relative trace error could be lower than predicted.

4.5.4 A large QCD problem

The trace estimator presented in this research has the potential of improving a number of LQCD calculations, where the trace of the Dirac matrix is related to an important property of QCD called spontaneous chiral symmetry breaking [119]. In our previous work [116], we presented the method of hierarchical probing that achieves almost optimal variance reduction incrementally and inexpensively.

As shown in Figure 4.15(a), a low rank approximation with 200 singular vectors yields a good approximation M and an excellent fit $p(M)$. In Figure 4.15(b), we see that the actual relative trace error decreases very fast to $O(10^{-4})$ with increasing number of fitting points and singular vectors, and can be monitored well by our dynamic trace estimation algorithm. These singular vectors can be approximated while solving the linear systems with eigCG. Interestingly, we can improve the relative trace error of hierarchical probing by two orders of magnitude. In addition, the variances of different MC methods can be estimated dynamically to allow us to continue with the best estimator if needed (Figure

4.15(c)).



(a) PCHIP results (b) Relative trace error (c) Variances

Figure 4.15: Fitting results, dynamic evaluation of relative trace error and variances with SVD on a large QCD matrix. In Figure 4.15(b), the green square denotes the relative trace error by applying hierarchical probing technique on deflated matrix E in [116].

4.6 Conclusion and Future Work

A novel method has been presented to estimate the trace of the matrix inverse by exploiting the pattern correlation between the diagonal of the inverse of the matrix and some approximation. The key idea is to construct a good approximation $M \approx D$ through eigenvectors or some preconditioner, sample important patterns of D by using the distribution of the elements of M, and use fitting techniques to obtain a better approximation $p(M) \approx D$ from where we obtain a trace estimate. The proposed method can provide a fast trace estimate with 2-3 digits relative accuracy given only a few samples while may or may not improve the variance of MC. When the variance is reduced sufficiently, our method can be also used as a diagonal estimator. We also propose an effective dynamic variance evaluation algorithm to determine the MC method with the smallest variance and a dynamic relative trace error estimation algorithm without any additional costs. We demonstrated the effectiveness of these methods through a set of experiments in some real applications.

Chapter 5

High-Performance Outlier Detection Method in Plasma

5.1 Introduction

During the last decade, there has been a significantly increasing need for knowledge discovery in spatial-temporal databases. Classical multi-dimensional outlier detection techniques are designed to detect global outliers. However, these techniques do not distinguish between non-spatial attributes and spatial attributes and do not consider apriori information about the statistical distribution of the data [109]. Since spatial-temporal data types have unique characteristics and their relations are more complicated than ordinary data, dedicated outlier detection techniques are typically required to examine anomalies in data across space and time [51]. In this study, we propose for the first time to formulate blob-filaments detection problem as region outlier detection problem in the domain of fusion plasma. Then we consider outliers with respect to their spatial neighbors and track them over time. A spatial outlier is a spatial object whose non-spatial attribute values are sigfinicantly different from those of other spatial objects in its spatial neighborhood [109]. In fusion plasma data, we relate spatial outliers with blob-filaments, which do not happen at a collection of scattered points but usually several groups of adjoining spatial points

or regions. Detection and tracking of these multiple regions from a continuously arriving stream is a challenging task due to various spatial scales and shapes of region outliers, which could change signiciantly over time [150, 86].

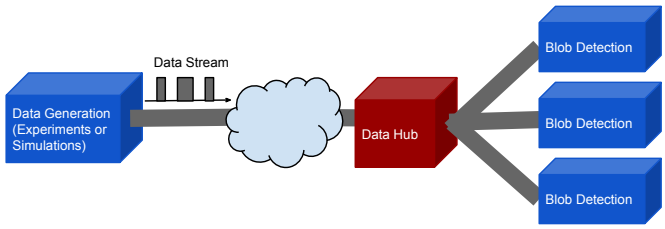


Figure 5.1: A real-time data analysis frame for finding blob-filaments in fusion plasma data streams

This work is motivated by several considerations responding to extreme scale computing and big data challenges in fusion energy. Fusion experiments and numerical simulations can easily generate massive amounts of data per run. During a magnetic fusion device experiment (or ”shot”), terabytes of data are generated over short time periods (on the order of hundreds of seconds). In the XGC1 fusion simulation [24, 75], a few tens of terabytes can be generated per second. Timely access to this amount of data can already be a challenge [30, 31], but analyzing all this data in real time is impractical. Currently, there are three types of analyses in most of fusion experiments: in-shot-analysis, between-shot-analysis, and post-run-analysis. All existing blob detection methods address post-run-analysis, but in this work, we focus on the more challenging first two cases to provide a real-time analysis so that scientists can monitor the progress of fusion experiments. Figure 5.1 presents a real-time analysis frame for finding blob-filaments in fusion plasma data streams. To perform this data analysis in real time, we utilize effectively modern supercomputers to address the high volume and velocity challenges arising from fusion plasma big data.

To this end, this work has been integrated into the International Collaboration Framework for Extreme Scale Experiments (ICEE), a wide-area in-transit data analysis framework for near real-time scientific applications [25]. ICEE takes advantage of an efficient IO solution ADIOS [83], and a cutting-edge indexing solution FastBit [130], to design and

construct a real-time remote data processing framework over wide-area networks for international collaborations such as ITER. In this system, a blob detection algorithm is used to monitor the health of fusion experiments at the Korea Superconducting Tokamak Advanced Research (KSTAR). However, existing data analysis approaches are often single-threaded, only for post-run analysis, and take a long time to produce results. Also, compared to the simulation data, the resolution of the raw camera data may be coarse, but interesting features can still be identified after normalization. In order to meet real-time feedback requirement, we develop a real-time blob detection method, which can leverage in-situ raw data in the ICEE server and find blob-filaments efficiently during fusion experiments. Our blob detection algorithm is not limited to KSTAR only, and can be applied to other fusion experiments and numerical simulations.

In this research, we exploit outlier detection techniques to effectively tackle the fusion blob detection problem on extremely large parallel machines. To the best of our knowledge, this is the first time the region outlier detection method has been used to detect blob-filaments in fusion plasma. The blob-filaments are detected as outliers by constantly monitoring specific features of the experimental or simulation data and comparing the real-time data with these features.

We first propose a two-phase region outlier detection method for finding blob-filaments. In the first phase, we apply a distribution-based outlier detection scheme to identify blob candidate points. In the second phase, we adopt a fast two-pass *connected component labeling* (CCL) algorithm from [131] to find different region outliers on an irregular mesh. Then we develop a high-performance blob detection approach to meet real-time feedback requirements by exploiting many-core architectures in a large cluster system. This is the first work to achieve real-time blob detection in *only a few milliseconds*. In addition, we propose a scheme to efficiently track the movement of region outliers by linking the centers of the region outlier over consecutive frames. We have implemented our blob detection algorithm with hybrid MPI/OpenMP, and demonstrated the effectiveness and efficiency of our implementation with a set of data from the XGC1 fusion simulations. Our

tests show that we can achieve *linear time speedup* and complete blob detection in *two or three milliseconds* using a cluster at NERSC. In addition, we demonstrate that our method is more robust than recently developed state-of-the-art blob detection methods in [29, 94].

The rest of research is organized as follows. In Section 5.2, we give the problem formulation of the blob detection and discuss related work. In Section 5.3 we describe a two-phase region outlier detection algorithm and a tracking scheme for identifying and tracking blobs. We then present a real-time blob detection approach by leveraging MPI/OpenMP parallelization in a large cluster in Section 5.4. The blob detection and tracking results and its real time evaluation are shown in Section 5.5. We conclude the research, and give our future plans in Section 5.6.

5.2 Problem Definition and Related Work

In this section, we introduce our problem definitions and discuss previous work related to our study. For related work, we first discuss existing research work on outlier detection, and then review previous work on blob detection in fusion plasma domain.

5.2.1 Problem Definition

In fusion plasma, the definition of a blob is varied in the literature depending on fusion experiments or simulations as well as available diagnostic information for measurements [35]. This makes blob detection a challenging task. Figure 5.2 plots local normalized density distribution in the regions of interest in one time frame. We can observe that there are two reddish spots located at the left portion of the figure, which are associated with blob-filaments and are significantly different from their surrounding neighbors. It is clear that a reddish spot is not a single point but a group of connected points or a region. Therefore, we formulate the blob detection problem as a region outlier detection problem. Similar to the spatial outlier [109], a region outlier is a group of spatial connected objects whose non-spatial attribute values are significantly different from those of other spatial

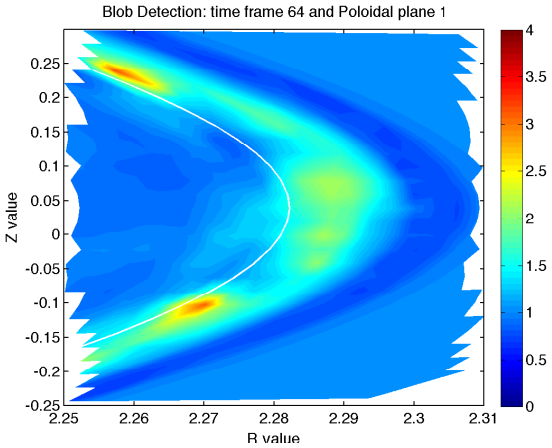


Figure 5.2: A contour plot of the local normalized density in the region of interests in one time frame in fusion experiments or numerical simulations. A cross-section of the torus is called a poloidal plane. R and Z are cylindrical coordinates and the major radius of the torus is denoted by R.

surrounding objects in its spatial neighborhood. As shown in Figure 5.2, blobs are region outliers. The number of region outliers detected is determined by pre-defined criteria provided by domain experts.

The problem is to design an efficient and effective approach to detect and track different shapes of region outliers simultaneously in fusion plasma data streams. By identifying and monitoring these blob-filaments (region outliers), scientists can gain a better understanding about this phenomena. In addition, a data stream is an ordered sequence of data that arrives continuously and has to be processed online. Due to the high arrival rate of data, the blob detection must finish processing before the next data chunk arrives [107]. Therefore, another critical problem is to develop a high-performance blob detection approach in order to meet the real-time requirements.

5.2.2 Outlier Detection

The problem of outlier detection has been extensively studied and can be generally classified into four categories: distance-based, density-based, clustering-based, and distribution-based approaches [61, 23].

Distance-based methods [73] use a distance metric to measure the distances among data points. If the number of data points within a certain distance from the given point is less than pre-defined threshold, then this point is determined as an outlier. This approach could be very useful with accurate pre-defined threshold. However, it may not be proper to use a simple threshold if different densities in various regions of the data exhibit across space or time.

Density-based methods [20] assign a local outlier factor (LOF) to each sample based on their local density. The LOF determines the degree of outlierness, where samples with high LOF value are identified as outliers. This approach does not require any prior knowledge of underlying distribution of the data. However, it has a high computational complexity since pair-wise distances have to be computed to obtain each local density value.

Clustering-based methods [49, 55] conduct clustering-based techniques on the sample points of the data to characterize the local data behavior. Since this method does not focus on outlier detection, the outliers are produced as by-products and it is not optimized for outlier detection.

Distribution-based methods [38, 109] applies machine learning techniques to estimate a probability distribution over the data and develop a statistical test to detect outliers. These methods use all dimensions to define a neighborhood for comparison and typically do not distinguish non-spatial attributes from spatial attributes.

In the context of data streams, a line of research has been devoted to develop efficient outlier detection techniques [123, 101, 36, 3, 2, 107]. But their main focus is to solve the problem of event detection in sensor network [123], query processing [3, 101], clustering [36], and graph outliers [2]. Therefore, these methods cannot be easily generalized to region outlier detection problems. In addition, the problem of blob detection presents a special challenge, because the spatial-temporal attributes of the blob-filaments has to be considered together to study their various characteristics including speed, direction, movement, and size. More importantly, these methods are mostly single-threaded which

cannot cope with real-time requirements in fusion plasma.

A number of distributed outlier detection methods have also been studied in [123, 34, 85, 97, 63]. Most of these methods are seeking an efficient way to parallelize classical outlier detection methods such as distance-based outliers [63, 85], distribution-based outliers [123], density-based outliers [85], density-based outliers [97], and PCA-based techniques [34]. However, there methods are not generally applicable to region outlier detection and tracking. In particular, in order to tackle high volume and velocity challenges arising from fusion plasma big data, specialized outlier detection scheme and suitable high performance computing technique are demanded to complete blob detection in the order of milliseconds.

In this work, we first apply distribution-based outlier detection to detect outlier points by considering only non-spatial attributes and then leverage fast CCL to construct the region outliers by taking into account spatial-attributes. We choose distribution-based outlier detection since it can solve the problem of finding outliers efficiently if an accurate approximation of a data distribution can be properly found [109, 123]. Normally the distribution of the stream data may change over time [51]. However, this assumption may not hold in fusion experiments since a fusion experiment lasts very short time period from a few seconds to hundreds of seconds. Therefore, we can perform exploratory data analysis to compute best fitted distribution parameters offline and then build an accurate online distribution model.

5.2.3 Blob Detection in Fusion Plasma

Independently, fusion blob detection problems have been researched by the physics community in the context of coherent structures in fusion plasma [35]. Various post-run blob detection methods have been proposed to identify and track these structures, to study the impact of the size, movement and dynamics of blobs. A plasma blob is most commonly determined by some threshold, computed statistically in the local plasma density signal

[148, 41, 151, 93]. However, the exact criteria have varied from one experiment to another, which reflects the intrinsic variability and complexity of the blob structures. In [148], a conditional averaging approach is applied to analyze spatiotemporal fluctuation data obtained from a two-dimensional probe array inside the last closed flux surface (LCFS) of the HL-2A tokamak. When the vorticity is larger than one standard deviation at some time frame, a blob is considered to be detected by the probe. In [41], the conditional averaging technique is also used to study the evolution of the blob-filaments using Langmuir probes and a fast camera. If a reference signal, with a certain sampling interval, has large fluctuation amplitude greater than a specified trigger condition, a blob structure is declared at that time frame.

Without using a conditional averaging technique, [151] searches for blob structures can be done using local measurements of the 2D density data obtained from a 2D probe array. Identification of a blob is based on the choices of several constraints such as the threshold intensity level, the minimum distance of blob movement, and the maximum allowed blob movement between successive frames. The trajectories of the different blobs can be computed with the blob centers based on identification results in each time frame. The seminal work by Zweben, et. al.[151] was the first attempt to take only individual time frame data into account to detect blobs and track their movements, although the process of identification of a blob was somewhat arbitrary and oversimplified. In [93], an analysis method was presented in terms of object-related observables to allow a sound probabilistic analysis. After preprocessing the signals from 2D imaging data to form signal matrix, a threshold-segmentation approach is used to identify blob structures when the local density is greater than an appropriately chosen threshold. Bounding polygons are also employed to track blob movements and compute their trajectories.

Due to the emergence of fast cameras and beam emission spectroscopy in the last decade, the situations of insufficient diagnostic access and limited spatial and temporal resolution have been greatly improved. In [84], an image analysis for the identification of blobs has been presented based on gas puff imaging (GPI) diagnostic images from

an ultra-high speed, high resolution camera. The raw images are first processed to remove the noise spikes, followed by further smoothing using a Gaussian filter. The blobs are identified by various image segmentation techniques after further processing which removes the background intensity from the images. However, due to noise and lack of a ground truth image, this approach can be sensitive to the setting of parameters, and it is hard to use generic method for all images. In addition, the output from visualization is not convenient to feed into other analysis [133]. The regions of interest computed from this work can be more conveniently fed into other analyses. For instance, one can compute blobs in the regions of interest very quickly and transmit these compact meta information over internet to remote domain scientists for real-time analysis.

Recently, several researchers [29, 76, 94] have developed a blob-tracking algorithm that uses raw fast camera data directly with GPI technique. In [29, 94], they leverage a contouring method, database techniques and image analysis software to track the blob motion and changes in the structure of blobs. After normalizing each frame by an average frame created from roughly one thousand frames around the target time frame, the resulting images are contoured and the closed contours satisfying certain size constraints are determined as blobs. Then, an ellipse is fitted to the contour midway between the smallest level contours and the peak. All information about blobs are added into a SQL database for more data analysis. This method is close to our approach but it can not be used for real-time blob detection since they compute time-averaged intensity to normalize the local intensity. Additionally, only closed contours are treated as blobs, which may miss blobs at the edges of the regions of interest. Finally, these methods are still post-run-analysis, which cannot provide real-time feedback in fusion experiments.

5.3 Our Proposed Approach

In this section, we provide a detailed description of our proposed approach to region outlier detection for finding blobs. Given a fusion data stream, which consists of a time

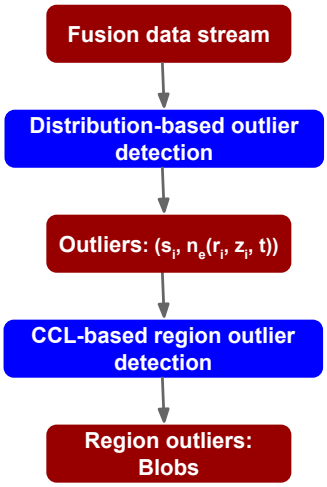
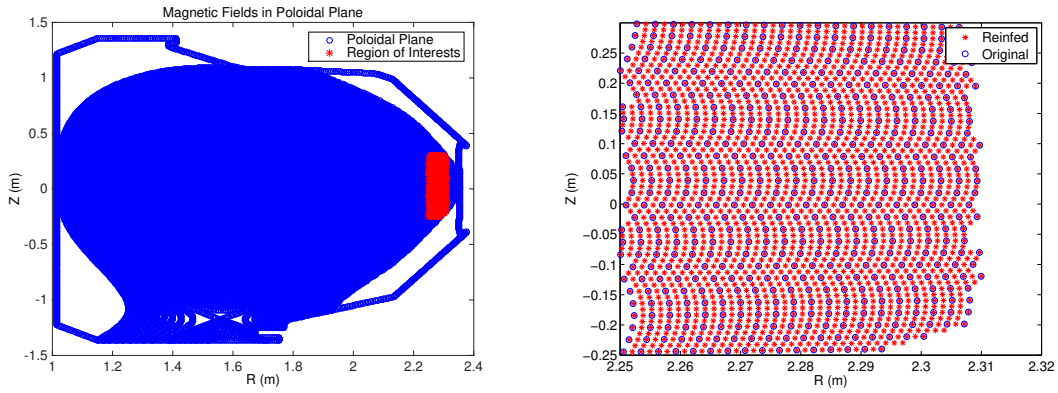


Figure 5.3: Two-phase region outlier detection for finding blobs

ordered sequence of sample frames that arrive continuously from fusion experiments or numerical simulations through remote direct memory access protocols. Our data sets are simulated electron density from the fusion simulation code XGC1 [24, 75]. In the present data sets, simulation data is captured every 2.5 microseconds for a total time window of 2.5 milliseconds. Each point $s_i \in S$ in a time frame t has a spatial attribute (r, z, t) which defines its location in a triangulated measurement grid, and some non-spatial attributes including all important plasma quantities such as electron density $n_e(r, z, t)$ as well as connectivity information in a poloidal plane. The spatial neighborhoods are defined for each point from the connectivity database in a triangulated grid. Formally, an region outlier responding to a blob is defined as a spatial area in the regions of interest where a subset $B_i \subseteq S$ is a group of connected outlier points s_i.

Our overall goal is to develop an algorithm to detect and track spatial region outliers (blobs) using a stream of fusion data. To achieve this, we propose a two-phase approach, as shown in Figure 5.3. In the first phase, we apply a distribution-based outlier detection algorithm to the fusion data stream in order to detect outlier points which have significantly higher non-spatial attributes than other points. The outputs of this step are tuples $(s_i, n_e(r_i, z_i, t))$, the 2D spatial attributes, and non-spatial attributes such as electron den-



(a) Regions of interest

(b) Refined mesh

Figure 5.4: An example of the regions of interest and the comparison between refined and original triangular mesh vertices in the R (radial) direction and the Z (poloidal) direction.

sity. These tuples, as well as connectivity information, are used as input for the second phase, where region outlier are detected by applying a fast CCL [131] to efficiently find different connected components on the triangular mesh. The outputs of the CCL-based region outlier detection algorithm are a set of connected components with outlier points inside, which are associated with blobs if some criteria are satisfied.

Note that our approach consists of two orthogonal steps, therefore each of the two phases can be replaced by other outlier detection methods. For example, one can leverage density-based outlier detection to find outlier points in the first phase. In addition, edge detection with fuzzy classifier can be used to detect the boundary of region outlier in the second phase [86].

In the following section, we describe the proposed two-phase region outlier detection in detail.

5.3.1 Distribution-Based Outlier detection

The main task of this phase is to perform efficient outlier detection to determine outlier points which form the region outliers associated with blobs. In this work, we propose a two-step, distribution-based outlier detection algorithm based on the electron density

with various criteria for fusion plasma data streams. We separate spatial attributes from non-spatial attributes and consider the statistical distribution of the non-spatial attributes to develop a test based on distribution properties, since it is more suitable for detecting spatial outliers [109]. As claimed in [123], it is very efficient to find outliers by using a data distribution approximation if we estimate the underlying distribution of data accurately. Values for various criteria are determined by domain experts or subjectively by examining the resulting plotting and adjusting them until satisfied.

The first step of the proposed outlier detection is to preprocess the sample frame to compute needed quantities in the region of interests, as shown in Figure 5.4(a). Then it is analyzed by normalizing the total electron density $n_e(r, z, t)$ (which includes fluctuations) with respect to the initial background electron density, $n_e(r, z, 1)$ (if using real diagnostic data from, e.g. GPI, actual emission intensity $I(r, z, t)$ would be used instead of electron density). Note that using the initial time frame as the benchmark is an important factor to achieve real-time blob detection. The normalized electron density in the subsequent time frames can be easily computed, especially compared to the time-average electron density with a long time interval [94].

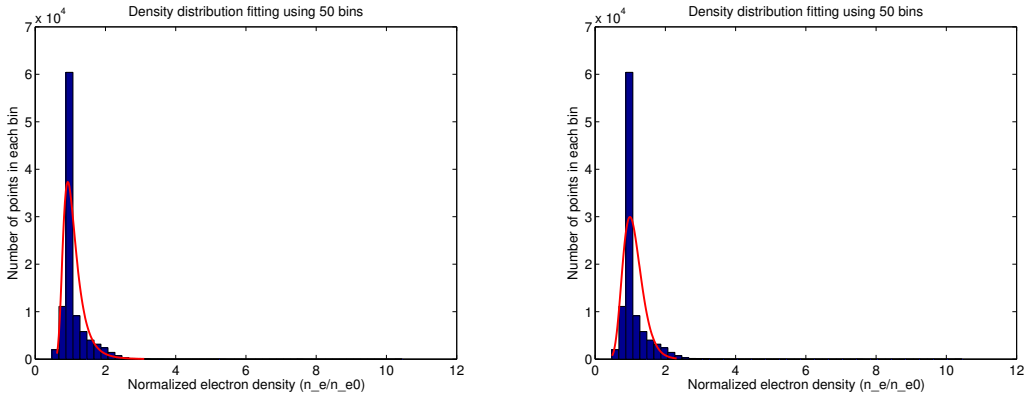
Algorithm 17 Triangular mesh refinement algorithm

Input/output:

$triGrid$: connectivity array of the triangular mesh
(r, z): spatial coordinate of each point
n_e: normalized electron density of each point

1: Compute unique edges E and indices vector I_E by sorting and removing duplicates based on $triGrid$
2: Compute spatial coordinate of each new vertices in the middle of E based on (r, z)
3: Compute electron density of each new vertices on E by performing linear interpolation based on n_e
4: Compute indices for each new vertices by adding vector index I_E with the number of original points
5: Compute a new triangular mesh by assigning appropriate indices from each new and old vertices

To obtain meaningful region outliers using the CCL method, it is necessary to have fine grained connectivity information. This particular simulation mesh has coarse verti-



(a) Extreme Value (b) Log Normal

Figure 5.5: An example of exploratory data analysis to analyze the underlying distribution of the local normalized density over all poloidal planes and time frames.

cal resolution, so resolution enhancement techniques are applied to generate a higher resolution triangular mesh based on the original triangulated mesh. As shown in Algorithm 17, the resulting triangular mesh is refined to achieve four times better granularity. We split each original triangle into four smaller ones by linking three middle points of the original mesh edges in each triangle. The corresponding density of generated vertices can be obtained using linear interpolation of the original triangular mesh. This step can be applied recursively until the satisfactory resolution of the triangular mesh is computed. Figure 5.4(b) shows the resulting triangular mesh vertices after applying the triangular mesh refinement algorithm once.

In order to apply an appropriate predefined quantile in two-step distribution-based outlier detection, it is advised to perform exploratory data analysis to exploit main characteristics of the data sets. Figure 5.5 reveals that extreme value distribution and log normal distribution are fitted best with one of our sample data sets (after comparing over sixteen different common distributions). After analyzing the underlying distribution, a two-step outlier detection is performed to determine outlier points in the regions of interest. The basic idea of the proposed two-step outlier detection is motivated from the observations that there are relatively high density areas (a half banded ellipse area with cyan color)

in the edge and several significantly high density small regions (a few small areas with reddish yellow color) in these relatively high density areas, as shown in Figure 5.2. The proposed outlier detection method extends the previous approach that applies statistical detection with conditional averaging intensity value [148, 41], and applies more intelligent two-step outlier detection with only considering individual time frame data. Compared to traditional single threshold segmentation approach, our approach is more generic, flexible and easier to tune a satisfactory result.

In the first step, the standard deviation σ and the expected value μ are computed over all sixteen poloidal planes in one time frame. Using the best fitted distribution, we apply first step outlier detection to identify the relative high density areas with a specified predefined quantile:

$$N(r_i, z_i, t) - \mu > \alpha * \sigma, \forall (r_i, z_i) \in \Gamma \tag{5.1}$$

where N is the normalized electron density, α is the multiple of σ associated to the specified predefined quantile and Γ is the domain in the region of interests. Once the relative high density regions are determined, we compute another standard deviation σ_2 and the expected value μ_2 in these areas. Then we employ second step outlier detection to identify the outlier points in the relative high density areas with an appropriately chosen predefined quantile:

$$N(r_i, z_i, t) - \mu_2 > \beta * \sigma_2, \forall (r_i, z_i) \in \Gamma_2 \tag{5.2}$$

where β is the multiple of σ_2 associated to the judiciously chosen confidence level and Γ_2 is the domain of blob candidates. In practice, α and β can be chosen to be same or different, depending on the characteristics of blob-filaments. In our experience, the α value is generally greater than β since the standard deviation σ over the region of interests is much smaller than the standard deviation σ_2 from the relative high density areas.

However, two-step outlier detection alone cannot be used to distinguish the blob candidates since identified blob candidates may actually have small density, which does not satisfy traditional definition of blobs. Therefore, the density of the mesh points in the out-

lier points smaller than a certain minimum absolute value criterion need to be filtered out. On the other hand, it is also possible that the middle areas between surrounding plasmas and outlier points have density higher than the given minimum absolute value criterion. Thus, we also apply a minimum relative value criterion to remove these unwanted points. To combine these two rules together, we have a more robust and flexible criterion:

$$N(r_i, z_i, t) > \textbf{max}(d_{ma}, (d_{mr} * \mu_2)), \forall (r_i, z_i) \in \Gamma_3 \tag{5.3}$$

where d_{ma} and d_{mr} are minimum absolute value and minimum relative value respectively, and Γ_3 is the domain of good blob candidates.

5.3.2 CCL-Based Region Outlier Detection

The main task of the second phase is to apply an efficient connected component labeling algorithm adopted from [131] on a refined triangular mesh to find different blob candidate components. A connected component labeling algorithm generally considers the problem of labeling binary 2D images with either 4-connectedness or 8-connectedness. It performs an efficient scanning technique, and fills the label array labels so that the neighboring object pixels have the same label. Wu [131] presents an efficient two-pass labeling algorithm that is much faster than other state-of-the-art methods and theoretically optimal. However, since we process a refined triangular mesh rather than the traditional 2D images, we have modified their CCL algorithm to take the special features of a triangular mesh into account. As shown in Algorithm 18, each triangle is scanned first rather than a point. Since we know the three vertices in a triangle are connected, we can reduce unnecessary memory accesses once any vertex in a triangle is found to be connected with another vertex in a different triangle. Then we compute the current minimum parent label in this triangle, and assign each vertex a parent label if its label has already filled or a label if its label has not initialized yet. If all three vertices in a triangle are scanned for the first time, then a new label number is issued and assigned to their labels and the as-

sociated parent label. After the label array is filled full, we need flatten the union and find tree. Finally, a second pass is performed to correct labels in the label array, and all blob candidates components are found. Note that to perform efficient union-find operations, the union-find data structure is implemented with a single array as suggested in [131].

Algorithm 18 Connected component labeling algorithm on triangular mesh to find various blob candidates components

Input:
$triGrid$: connectivity array of the triangular mesh
Output:
B_c: Region structure of each blob candidate
Initialize $label$, $parentLabel$, and $labnum$
for Scanning each triangle until the end of $triGrid$ **do**
  if $label$ of three vertices are all zero **then**
    Assign a new $labnum$ to all three vertices
    Update $label$ and $parentLabel$ with $labnum$
  else
    Find the minimum $parentLabel$ of all three vertices
    Update their $label$ and $parentLabel$ with this value
  end if
end for
for Scanning until the end of $parentLabel$ **do**
  Update $parentLabel$ by flattening union-find tree
end for
for Scanning until the end of $Label$ **do**
  Update $Label$ with latest $parentLabel$
end for
Find each B_c of points with same $parentLabel$

After all blob candidates are determined, a blob is claimed to be found if the median of a blob candidate component satisfies a certain minimum absolute median value criterion. The reason we are setting this constraint to select the blobs is that the minimum value criterion has to be a reasonably small value in order to produce more blob candidate components. It is possible that if the minimum absolute median value criterion is too large, it may also remove the blobs. On the other hand, it is also possible if this value is too small, it does not have effect on filtering out unwanted components. Therefore, with the same philosophy of measurement, a minimum relative median value criterion is

also applied to determine the blobs. However, in order to protect the blobs from being removed due to the extremely large mean value μ_2, we also set the maximum absolute median value criterion to limit the power of minimum relative median value criterion. We unify these three rules to be one:

$$N(r_i, z_i, t) > \textbf{max}(\hat{d}_{ma}, min((\hat{d}_{mr} * \mu_2), \hat{d}_{xa})), \forall (r_i, z_i) \in \Gamma_4 \tag{5.4}$$

where $\hat{d}_{ma}$, $\hat{d}_{mr}$ and $\hat{d}_{xa}$ are minimum absolute and relative median values and maximum absolute median value respectively and Γ_4 is the domain of blobs.

5.3.3 Tracking Region Outliers

Another objective of this work is to track the direction and speed of the detected blobs over time. The blob tracking algorithm has to cope with the problem of tracking multiple region outliers simultaneously even when the blobs merge together or split into separated ones. On the other hand, the blob tracking method should be simple and efficient to meet real-time requirements. To achieve this goal, we propose an efficient blob tracking algorithm by leveraging cues from changes of blobs area and distance of center of blobs. We compute the correspondence between previously tracked blobs and currently detected blobs, and then recover the trajectories of the tracked blobs.

To identify the location center of detected blob, we compute the density-weighted average of the spatial coordinates of each point inside a blob.

$$(r_c, z_c) = \frac{1}{M} \sum_{i=1}^{n} (r, z) n_e \tag{5.5}$$

where M is summation of n_e of all points in a blob. The density-weighted average is used to better capture the center of density of a blob. We track the movement of these detected blobs by linking the centers in consecutive time frames. In order to obtain the boundary of region outliers (blobs), we compute the convex hull [22] of a set of points in a blob. The

area of a blob is computed by counting the number of points in a blob.

Algorithm 19 Efficient blob tracking algorithm

Input:
B: Current detected blobs
T: Previous blob tracks
Output:
T: Updated blob tracks with B appended

1: Initialize $hull$, cen, and $area$
2: $hull$ = getBoundary(B)
3: cen = getCenter(B)
4: $area$ = getArea(B)
5: **for** Scanning until the end of B **do**
6: $cenDis$ = getCenterDis(B,T)
7: $areaDif$ = getAreaDif(B,T)
8: **if** $cenDis \leq$ maxJump $\wedge$ $areaDif \leq$ maxDif **then**
9: Find a blob track T with smallest $cenDis$
10: Append current blob into this blob track T
11: **end if**
12: **end for**
13: Update T with $hull$, cen, $area$, and computed $speed$

As shown in Algorithm 19, the input parameters are current detected blobs and the previous blob tracks. The data structure of a blob track is composed of the track ID, the length of track, the area of previous blob, the time-stamps, the center points, the boundary points, and the velocity. There are two heuristics to verify whether a blob is associated with an existing blob track. The first heuristic is based on the fact that the area of a blob between consecutive time frames cannot decrease or increase significantly. The second heuristic takes into account the distance of the centers of a blob does not change dramatically over very short time period (microseconds). The proper thresholds for these two heuristics are provided by domain experts. Since blobs can appear, disappear, merge together or split, a greedy scheme is applied to find the best matching pair of blob and track based on closest distance of the centers of current detected blob and the latest blob in a blob track. Based on computed correspondence between a blob track and the currently detected blobs, existing blob tracks are automatically processed through corresponding operations such as adding a blob into a track, creating a new track, and a track ending.

If the length of a track is smaller than 3 consecutive time frames, the track will be treated an anomaly and deleted due to errors in data or inappropriate blob detection thresholds. The speed and direction of the blobs can thus be computed from two consecutive center points. Finally, we can recover the trajectories of the tracked blobs by monitoring the movement of blob centers.

5.4 A Real-Time Blob Detection Approach

Existing blob detection approaches cannot tackle the two challenges of the large amount of data produced in a shot and the real-time requirement. In addition, existing data analysis approaches are often operated in a single thread, only for post-run analysis and often take a few hours to generate the results [93]. In order to meet the real-time feedback requirement, we address these challenges by developing a high performance blob detection approach, which can leverage in situ raw data and find blob-filaments efficiently in fusion experiments or numerical simulations.

5.4.1 A hybrid MPI/OpenMP parallelization

The key idea is to exploit many cores in a large cluster system by running MPI to allocate n processes to process the data in one or several time frames at the high level, and by leveraging OpenMP to accelerate the computations using m threads at the low level. Figure 5.6 shows our hybrid MPI/OpenMP parallelization for blob detection. Using this approach, we can complete our blob detection in a few milliseconds with in situ evaluation.

In order to achieve blob detection in real time, the goal is to minimize data movements in the memory and speed up computation. Ideally, the performance is optimal without any communication if we can perform the job correctly. The proposed blob detection algorithm in the previous section supports embarrassed parallel since we only need the initial time frame and the target time frame to do the computation. This is an important difference between our blob detection method and recently developed methods [29, 94] in terms

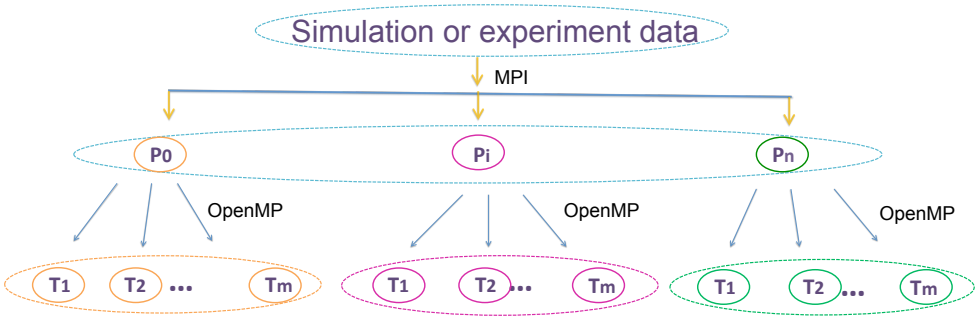


Figure 5.6: Hybrid MPI/OpenMP parallelization

of real-time requirement. Furthermore, we explore many-core processor architectures to speed up the computation of each MPI task by taking full advantage of multithreading in the shared memory. Therefore, our real-time blob detection approach based on hybrid MPI/OpenMP parallelization is a natural choice and is expected to provide the optimal performance for fusion plasma data streams.

A practical interesting issue is how to tune the number of MPI processes and OpenMP threads for the best performance by taking both analysis speed and memory size into account. As shown in Figure 5.7, we vary the number of MPI processes and OpenMP threads but fix the total number to be 24 for investigating the performance when processing the same amount of time frames data. A faster analysis speed is achieved when increasing the number of MPI processes since more data frames can be processed simultaneously. On the other hand, the analysis speed remains constant with a few OpenMP threads and degrades with more OpenMP threads due to lack of enough computation in one time frame. However, more OpenMP threads could significantly reduce the memory demands. Therefore, in this study, we choose the number of OpenMP threads to be four for each MPI task, to achieve a good trade off between analysis speed and memory savings.

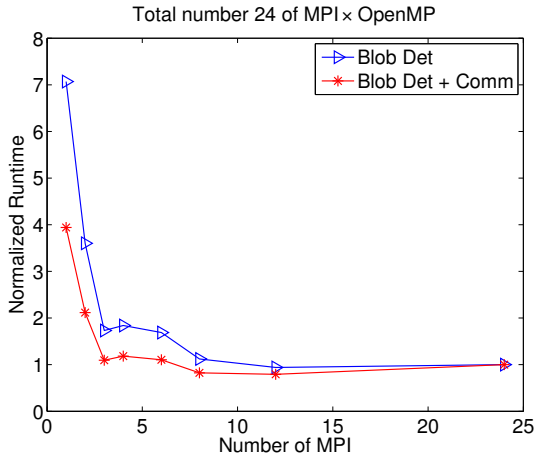


Figure 5.7: Investigate the performance of hybrid MPI/OpenMP parallelization when varying number of MPI processes and OpenMP threads. The blue triangle denotes only normalized blob detection time. The red star denotes the normalized total time including both blob detection time and initial communication time for broadcasting the first time frame to all analysis nodes for normalization.

5.4.2 Outline of the implementation

We implement our blob detection algorithm in C with a hybrid MPI/OpenMP parallelization. Algorithm 20 summarizes the proposed blob detection algorithm without considering OpenMP. Users can specify the regions of interest by (Rmin, Rmax, Zmin, Zmax), the range of target time frames by (t_start, t_end), and the location of the data sets. However, with in situ evaluation, there is no need to specify the file location since all data are already in memory. We use static scheduling to evenly divide the number of time frames for each MPI task for efficiency. The n MPI processes are allocated to process one or several time frames and m OpenMP threads are launched to accelerate the computation in one time frame. Note that the MPI process is also the master thread in the runtime environment. At the beginning, the initial time frame data is broadcasted to all MPI processes so that normalization can be performed with new coming time frames. Then each MPI process embarrassingly process the data in each time frame with multithreading in the shared memory. Only detected blobs information are maintained and added into local database. Since these local blobs information are very compact, they can be efficiently

transmitted over internet to remote servers for real-time analysis by domain scientists.

Algorithm 20 A real-time blob detection approach

Input:
Rmin, Rmax, Zmin, Zmax: specify region of interest
t_start, t_end : start and end time frames
FileDir: location where data sets locate
Output:
B: Detected region outliers (blobs)

1: Apply static scheduling to assign equal amount of n time frames data to each MPI process
2: Broadcast the initial time frame to all MPI processes
3: **for** $t = 1 : n$ **do**
4: Process i loads raw data in one frame and computes normalized density $n_e(r, z, t)$ in region of interest
5: Refine the triangular mesh. See Algorithm 17
6: Apply two-step distribution-based outlier detection to identify outliers with various criteria
7: Apply CCL-based region outlier detection on a triangular mesh to find blob components. See Algorithm 18
8: A blob is added into B if certain criteria is satisfied
9: **end for**

5.5 Experiments and Results

In this section we present experimental evaluations of our blob detection and tracking algorithms, and report the performance of the real-time blob detection under both strong and weak scaling. Before showing experimental results in the next section, we briefly introduce our experimental environment, data sets, and parameters setting in our blob detection and tracking algorithms. We have tested our implementation on the NERSC's newest supercomputer Edison, where each compute node has two Intel "Ivy Bridge" processors (2.4GHz with 12 cores) and 64 GB of memory. Our data sets are small simulation data sets (30GB) with 1024 time frames based on the XGC1 simulation [24][75] from the Princeton Plasma Physics Laboratory, which last around 2.5 milliseconds. One of our main goals is that we can control analysis speed by varying the number of processes to

complete the blob detection on the entire data set in a time close to 2.5 milliseconds. It would indicate that our algorithm could monitor fusion experiments in real time (neglecting data transfer latency). If we consider internet transfer latency in real experiments or numerical simulation, the system needs at least 1 to 25 milliseconds to transfer one time frame data depending on size of data, which may give us more time for data analysis.

Another goal is to validate the effectiveness of the proposed algorithms. In Algorithm 20, we apply various criteria to identify the blobs. The parameters for blob detection and tracking in our experiments are given in Table 5.1. One criterion we have not mentioned in the previous section is parameter "minArea". This parameter is used to decide how many points a blob should have, which is used to remove impossibly small blobs. In our experiment, this parameter is set to three since there are at least three vertices connected as a 2D component in a triangular mesh. Another criteria are parameters "maxFrames" and "minFrames", which are used to control the length of a blob track and remove noisy tracks. It is important to note that these parameters need to be tuned in order to achieve optimal performance in different fusion experiments or numerical simulations. The reasons for this uncertainty in the context of blob detection are from the intrinsic variability and complexity of the blob structures observed in different experiments [35].

Table 5.1: Parameters setting for the proposed blob detection and tracking algorithms on XGC1 simulation data sets

detection criteria		tracking criteria	
minArea	3	maxAreaChange	25
minRden (d_{ma})	1.2	maxJump	0.04
minAbsden (d_{mr})	2.05	maxFrames	100
maxAbsMden ($\hat{d}_{xa}$)	2.75	minFrames	3
minMden ($\hat{d}_{ma}$)	1.3		
minAbsMden ($\hat{d}_{mr}$)	2.15		

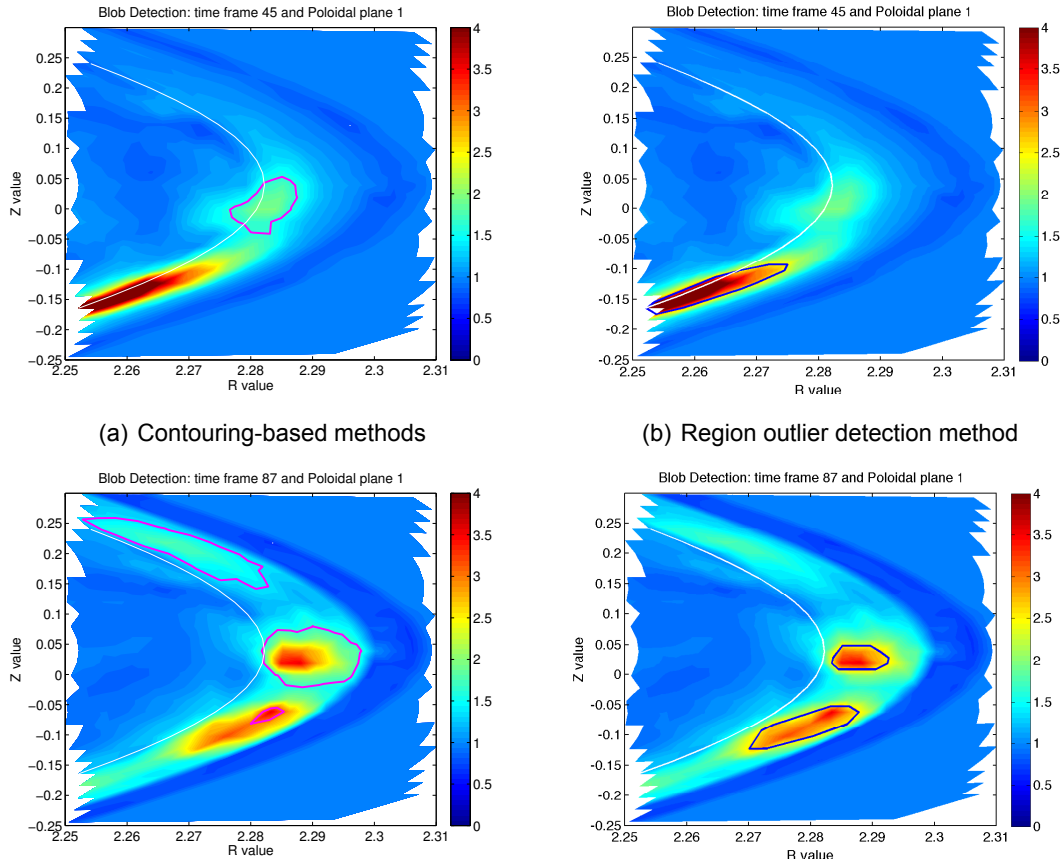
5.5.1 Performance comparison

We first conduct experiments to compare our method with recently developed state-of-the-art blob detection methods in [29, 94]. Since their methods are based on the contouring methods and thresholding, we call their methods the contouring-based methods. We have to point out that strictly quantitative comparisons are not possible since the blob itself is not well-defined [35]. Due to this reason, there are rarely direct comparisons between any new proposed method and existing ones in the literature in the domain of fusion plasma [148, 41, 151, 93, 84, 29, 76, 94]. However, in order to demonstrate that our methods is more robust than the contouring-based methods, we compare these two methods in two typical cases to shed light on their performance in terms of the detection accuracy.

Figure 5.8 shows the comparison of the blob detection results between our region outlier detection method and the contouring-based methods in two different time frames. As shown in Figures 5.8(a) and 5.8(b), we can see that our region outlier detection method does not miss detecting the blob at the edge of the regions of interest while the contouring-based methods fail the detection. The reason is that the contouring-based methods require the computed contours are closed, which do not exist at the edge of the regions of interest. In Figures 5.8(c) and 5.8(d), we notice that our region outlier detection method can accurately detect all blobs. However, the contouring-based methods either yield the blobs with incorrect areas (much larger or smaller), or misdetect the wrong area as a blob. This is because that it is hard to use one single threshold to identify the blobs for various time frames even in the same experimental data. Our region outlier detection method does not have such problem since we use more generic two-step distribution-based outlier detection.

5.5.2 More blob detection results

We perform more experiments to comprehensively examine the blob detection results in five continuous time frames and four different poloidal planes as shown in Figure 5.9. As



(a) Contouring-based methods

(b) Region outlier detection method

(c) Contouring-based methods

(d) Region outlier detection method

Figure 5.8: Two examples of comparing our region outlier detection method with the Contouring-based methods in the R (radial) direction and the Z (poloidal) direction. The separatrix position is shown by a white line and the different pink and blue circles denote blobs.

we can see from the figure, our region outlier detection method can provide consistently good results in different situations. In addition, our method does not miss any blobs at the edge of the regions of interest, as shown in subfigures 5.9(b), 5.9(g), 5.9(c) and 5.9(h). It is interesting to see that large-scale blob structures are often generated, which could cause substantial plasma transport [151]. As pointed out in [147], these large-scale structures are mainly contributed by the low-frequency and long-wavelength fluctuating components, which may be responsible for the observations of long-range correlations. We also noticed that different poloidal planes may display significant diversity in edge turbulence, even in the same time frame. We have shown that we are able to effectively detect the blobs

and reveal some interesting results to help physicists improve their understanding of the characteristic of blobs and their correlation with other plasma properties.

5.5.3 Blob tracking results

We investigate the blob tracking results in two different situations. Figure 5.10(a) exhibits a 2D trajectory of a blob. Again, the trajectory is generated by plotting the location of the density peak of the detected blobs over five consecutive time frames. We can see that our blob tracking algorithm can track two separate blobs simultaneously. The blob size can grow when they move towards confined plasma in the right region of separatrix. Figure 5.10(b) shows a 3D trajectory for a detected blob over fifteen consecutive time frames. In this case, the blob seems to maintain its size for a few time frames, then gradually decreases, and eventually disappears. Through these interesting results, physicists may be able to understand the characteristics of blobs better.

5.5.4 Real-time blob detection under strong scaling

We have illustrated the effectiveness and robustness of the proposed blob detection and tracking methods. Next, we perform a set of experiments to demonstrate the performance of our real-time blob detection approach under strong scaling and weak scaling.

Our most encouraging results are that we can complete blob detection on the simulation data set described above in around 2 milliseconds with MPI/OpenMP using 4096 cores and in 3 milliseconds with MPI using 1024 cores. In Figure 5.11, we can achieve linear time speedup in blob detection time under strong scaling. The MPI and the MPI/OpenMP implementations accomplish 800 and 1200 times speedup respectively, when the number of processes is scaled to 1024. Also, we can see that the hybrid MPI/OpenMP implementation is about two times faster than the MPI implementation when varying the number of processes from 1 to 512. With 1024 processes, both of them achieve similar perfor-

mance, but the MPI/OpenMP one is slightly better. This demonstrates that we are able to control analysis speed by varying the number of processes to achieve real-time analysis.

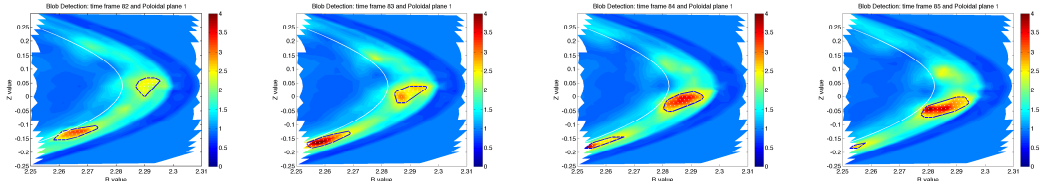
5.5.5 Real-time blob detection under weak scaling

In this experiment, we evaluate the performance of our real-time blob detection under weak scaling. We replicate existing data sets (30GB) in order to obtain adequate experimental data sets (4.3TB). The basic unit data contains 128 time frames and the size of data increases linearly with the number of processes. In Figure 5.12, the blob detection time remains almost constant under weak scaling, which indicates that our implementations scale very well to solve much larger problems. Also, both MPI and MPI/OpenMP implementations achieve high parallel efficiency as the number of processes increases from 1 to 1024.

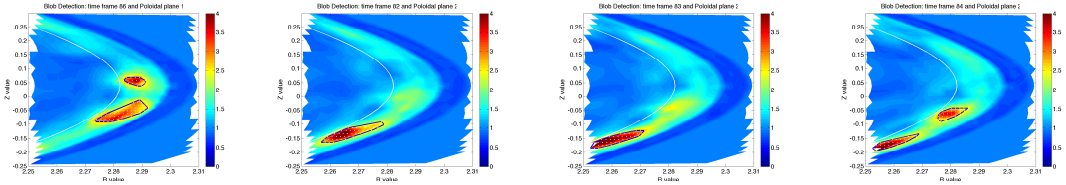
5.6 Conclusion and Future Work

Near real-time data analysis of the long-pulse fusion plasma experiments presents both opportunities and challenges responding to extreme scale computing and big data in fusion energy. In this research, we propose, for the first time, a real-time blob detection approach for finding blob-filaments in fusion experiments or numerical simulations. The key idea of the proposed two-phase region outlier detection scheme is based on distribution-based outlier detection with various criteria and a fast CCL method to find blob components. In addition, an efficient blob tracking scheme is presented to recover the trajectories of the motions of blobs. We have implemented our blob detection algorithm with hybrid MPI/OpenMP and demonstrated the effectiveness and efficiency of our implementation with a set of fusion plasma simulation data. Our tests show that we can achieve linear time speedup and complete blob detection in two or three milliseconds using a cluster at NERSC.

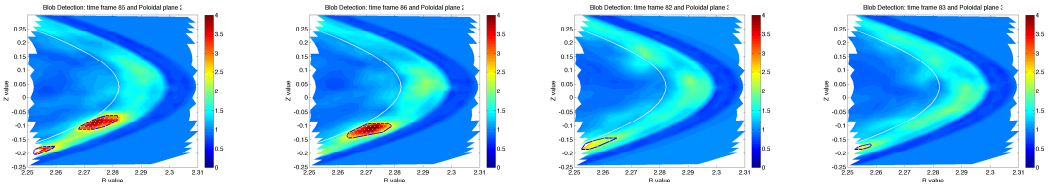
We are currently working on integrating our blob detection algorithm into the ICEE system for consuming fusion plasma data streams where the blob detection function is used in a central data analysis component and the resulting detection results are monitored and controlled from portable devices, such as an iPad. We plan to test the proposed method in both numerical simulations and real fusion experiments.



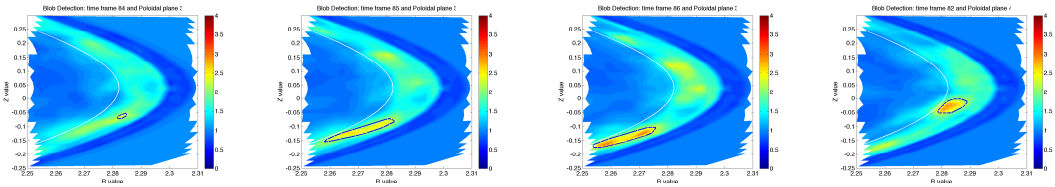
(a) Frame 82 and plane 1 (b) Frame 83 and plane 1 (c) Frame 84 and plane 1 (d) Frame 85 and plane 1



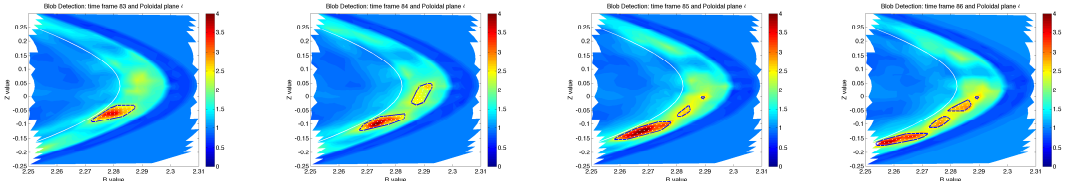
(e) Frame 86 and plane 1 (f) Frame 82 and plane 2 (g) Frame 83 and plane 2 (h) Frame 84 and plane 2



(i) Frame 85 and plane 2 (j) Frame 86 and plane 2 (k) Frame 82 and plane 3 (l) Frame 83 and plane 3

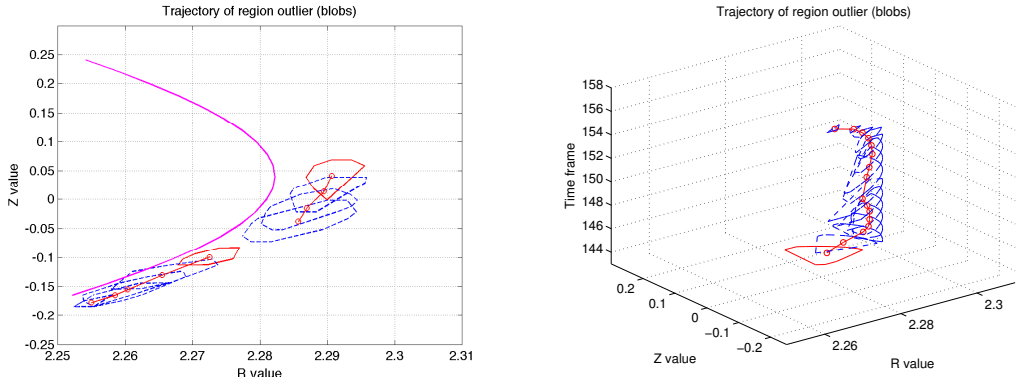


(m) Frame 84 and plane 3 (n) Frame 85 and plane 3 (o) Frame 86 and plane 3 (p) Frame 82 and plane 4



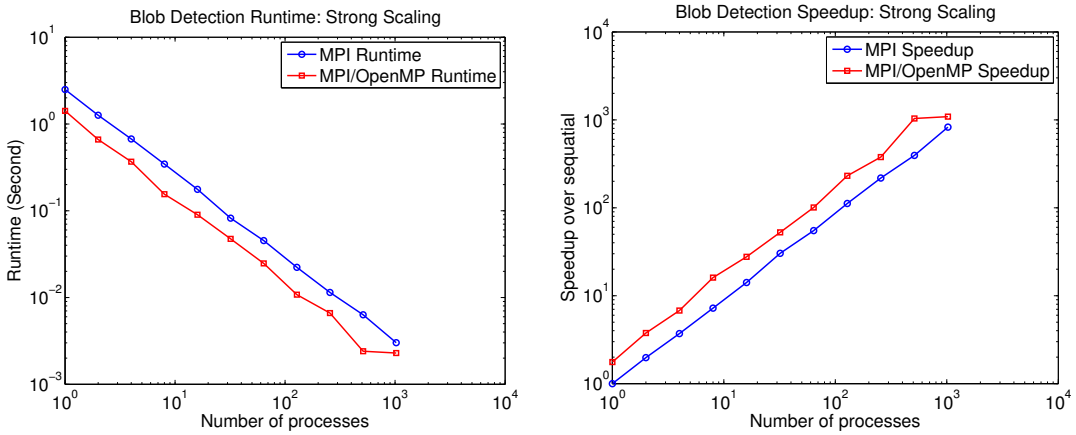
(q) Frame 83 and plane 4 (r) Frame 84 and plane 4 (s) Frame 85 and plane 4 (t) Frame 86 and plane 4

Figure 5.9: An example of the blob detection in five continuous time frames and four different poloidal planes in the R (radial) direction and the Z (poloidal) direction. The separatrix position is shown by a white line and the different blue circles denote blobs.



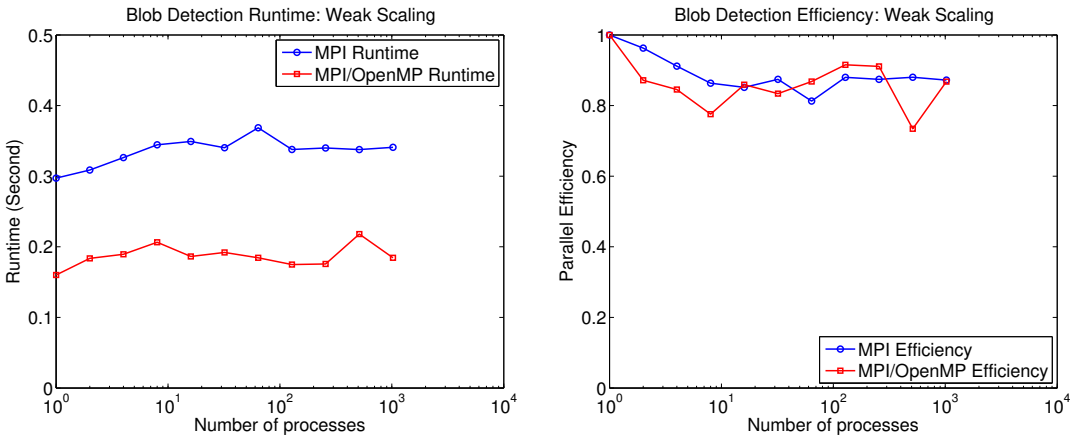
(a) 2D trajectory for detected blobs (b) 3D trajectory for detected blobs

Figure 5.10: 2D and 3D center trajectories for detected blobs over consecutive time frames. The red solid polygon indicates the starting times of the blobs tracked while the blue broken polygons indicate subsequent times of the same blobs tracked. The centers of the moving blobs are linked to show their trajectories of the blob motion. The pink line represents the separatrix.



(a) Time (b) Speedup

Figure 5.11: Blob detection time and speedup with MPI and MPI/OpenMP varying number of processes under strong scaling



(a) Time (b) Speedup

Figure 5.12: Blob detection time and speedup with MPI and MPI/OpenMP varying number of processes under weak scaling

Chapter 6

Summary and Future Work

The goal of this thesis is to design new algorithms and develop high quality software for solving difficult large-scale eigenvalue problems, singular value problems, and some big data analysis applications. Specifically, we studied the computation of the smallest singular triplets and interior eigenpairs, the trace estimation of an implicit matrix, and the detection of blob-filaments in fusion plasma. We summarize the main results we have achieved and the future work in each chapter:

- In Chapter 2, we have developed a full functionality, high quality SVD solver for a few smallest or largest singular values of a large matrix. The key idea is a two stage meta-method, PHSVDS, which firstly solves the normal equations efficiently to get sufficiently accurate approximations, and continues solving an interior eigenvalue problem on the augmented matrix if further accuracy is needed. Furthermore, an efficient computation of the interior eigenproblem is allowed based on a simplified refined projection due to the availability of good initial shifts and guesses from the first stage. PHSVDS significantly advances current the state-of-the-art in singular value methods.

  We further develop a high-performance preconditioned SVD software to implement the PHSVDS method on top of PRIMME. PRIMME_SVDS is an attempt to fill a gap in high quality SVD software for efficiently and accurately solving both largest

and smallest SVD problems. We have demonstrated its good parallel performance up to 1024 MPI processes on the largest problems we have ever seen in the SVD literature. More importantly, we have provided multiple popular user interfaces such as Python, Matlab, R, and C to serve a broader class of users, especially these who are not experts in SVD computations.

- In Chapter 3, we discuss different approaches to compute refined Ritz vectors for interior eigenvalue problems and analyze their merits and drawbacks. Based on these investigations, we propose a new hybrid approach for the efficient and accurate computation of the refined Rayleigh-Ritz procedure in non-Krylov iterative methods such as GD/JD type methods. We have shown that seeking even a small number of interior eigenvalues is a really hard problem. With increasing size of problems, a practical and efficient method for refined projection is highly demanded for solving interior eigenvalue problems. We hope our method has shed some lights towards developing a high-quality eigensolver software in this direction.

- In Chapter 4, we present a novel trace estimator for the trace of the matrix inverse by exploiting the pattern correlation between the diagonal of the inverse of the matrix and that of some approximation. We investigate a number of sparse matrix decomposition techniques to compute a good approximation of A^{-1}. In addition, we study a sophisticated sampling method, a linear LS fitting and a PCHIP fitting technique to build an effective piecewise polynomial model for estimating the trace. Furthermore, we propose a dynamical variance evaluation algorithm to estimate the variances of different MC methods, which can be used to choose appropriate MC whose variance has been reduced sufficiently by our fitting model. Also, we present a dynamical evaluation scheme to monitor the relative error of the estimated trace. We implement a framework with various options of different techniques and demonstrate the effectiveness of the proposed methods through a set of experiments.

  An interesting future direction is to take all available information such as index num-

bers and values of diagonal elements as input to an appropriate machine learning technique such as kernel ridge regression [145, 134] to improve the accuracy of predicted diagonal elements. Some new approaches, can be exploited to compute a better approximation in order to further facilitate the improvement with these machine learning techniques.

- In Chapter 5, we present the first work to achieve a real-time blob-filaments detection in fusion simulations and experiments. In order to tackle high volume and velocity challenges arising from fusion plasma big data, we have designed a specialized outlier detection scheme and suitable high performance computing techniques. The key components of the proposed algorithm are a two-step outlier detection with various criteria and a fast connected component labeling method to find blob components. We have implemented our blob detection algorithm in C with hybrid MPI/OpenMP and demonstrated parallel efficiency of our implementation with a set of fusion plasma simulation data. Our tests show that we can achieve linear time speedup up to 1024 MPI processes and complete blob detection in two or three milliseconds using supercomputer Edison at NERSC. We plan to conduct more experiments on the data from a real Tokamak device.

Because most projects in this research are closely related, we expect a strong cross fertilization of results. For example, our two stage SVD method depends on many aspects of the proposed hybrid approach for the computation of the refined Ritz vectors. Also, our SVD solver and eigensolver can be applied to construct low rank approximations for the trace estimator.

Bibliography

[1] Abdou M Abdel-Rehim, Andreas Stathopoulos, and Kostas Orginos. Extending the eigCG algorithm to nonsymmetric lanczos for linear systems with multiple right-hand sides. *Numerical Linear Algebra with Applications*, 21(4):473–493, 2014.

[2] Charu C Aggarwal, Yuchen Zhao, and Philip S Yu. Outlier detection in graph streams. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 399–409. IEEE, 2011.

[3] Fabrizio Angiulli and Fabio Fassetti. Detecting distance-based outliers in streams of data. In *Proceedings of the sixteenth ACM Conference on information and knowledge management*, pages 811–820. ACM, 2007.

[4] M. Arioli and J. Scott. Chebyshev acceleration of iterative refinement. *Numerical Algorithms*, 66(3):591–608, 2014.

[5] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010.

[6] Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM*, 58(2):8, 2011.

[7] R Aymar, P Barabaschi, and Y Shimomura. The ITER design. *Plasma Physics and Controlled Fusion*, 44(5):519, 2002.

[8] James Baglama and Lothar Reichel. Augmented implicitly restarted Lanczos bidiagonalization methods. *SIAM J. Sci. Comput.*, 27(1):19–42, 2005.

[9] James Baglama and Lothar Reichel. Restarted block Lanczos bidiagonalization methods. *Numerical Algorithms*, 43(3):251–272, 2006.

[10] James Baglama and Lothar Reichel. An implicitly restarted block Lanczos bidiagonalization method using Leja shifts. *BIT Numerical Mathematics*, 53(2):285–310, 2013.

[11] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*, volume 11. SIAM, 2000.

[12] Zhaojun Bai, Gark Fahey, and Gene Golub. Some large-scale matrix computation problems. *Journal of Computational and Applied Mathematics*, 74(1):71–89, 1996.

[13] Christopher G Baker and Michael A Heroux. Tpetra, and the use of generic programming in scientific computing. *Scientific Programming*, 20(2):115–128, 2012.

[14] S Balay, S Abhyankar, M Adams, J Brown, P Brune, K Buschelman, V Eijkhout, W Gropp, D Kaushik, M Knepley, et al. PESTc users manual revision 3.5. Technical report, Technical report, Argonne National Laboratory (ANL), 2014.

[15] Jesse L. Barlow. Reorthogonalization for the Golub–Kahan–Lanczos bidiagonal reduction. *Numerische Mathematik*, 124(2):237–278, 2013.

[16] Constantine Bekas, Alessandro Curioni, and I Fedulova. Low cost high performance uncertainty quantification. In *Proceedings of the 2nd Workshop on High Performance Computational Finance*, page 8. ACM, 2009.

[17] Costas Bekas, Effrosyni Kokiopoulou, and Yousef Saad. An estimator for the diagonal of a matrix. *Applied Numerical Mathematics*, 57(11):1214–1229, 2007.

[18] Michele Benzi and Miroslav Tuma. A robust preconditioner with low memory requirements for large sparse least squares problems. *SIAM J. Sci. Comput.*, 25(2):499–512, 2003.

[19] Michael W. Berry. Large-scale sparse singular value computations. *Int. J. Supercomputer. Appl.*, 6(1):13–49, 1992.

[20] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM SIGMOD Record*, volume 29, pages 93–104. ACM, 2000.

[21] Claude Brezinski, Paraskevi Fika, and Marilena Mitrouli. Moments of a linear operator, with applications to the trace of the inverse of matrices and the solution of equations. *Numerical Linear Algebra with Applications*, 19(6):937–953, 2012.

[22] Timothy M Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16(4):361–368, 1996.

[23] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[24] CS Chang, S Ku, PH Diamond, Z Lin, S Parker, TS Hahm, and N Samatova. Compressed ion temperature gradient turbulence in diverted tokamak edge). *Physics of Plasmas (1994-present)*, 16(5):056108, 2009.

[25] Jong Y Choi, Kesheng Wu, Jacky C Wu, Alex Sim, Qing G Liu, Matthew Wolf, CS Chang, and Scott Klasky. ICEE: Wide-area in transit data processing framework for near real-time scientific applications. In *4th SC Workshop on Petascale (Big) Data Analytics: Challenges and Opportunities in conjunction with SC13*, 2013.

[26] Jane Cullum, Ralph A. Willoughby, and Mark Lake. A Lanczos algorithm for computing singular values and vectors of large matrices. *SIAM J. Sci. Stat. Comput.*, 4(2):197–215, 1983.

[27] 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.

[28] Timothy A Davis and Yifan Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1, 2011.

[29] WM Davis, MK Ko, RJ Maqueda, AL Roquemore, F Scotti, and SJ Zweben. Fast 2-d camera control, data acquisition, and database techniques for edge studies on NSTX. *Fusion Engineering and Design*, 89(5):717–720, 2014.

[30] Bin Dong, S. Byna, and Kesheng Wu. Expediting scientific data analysis with reorganization of data. In *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, pages 1–8. IEEE, Sept 2013.

[31] Bin Dong, Surendra Byna, and Kesheng Wu. SDS: A framework for scientific data services. In *Proceedings of the 8th Parallel Data Storage Workshop*, PDSW '13, pages 27–32. ACM, 2013.

[32] Jack J. Dongarra. Improving the accuracy of computed singular values. *SIAM J. Sci. Stat. Comput.*, 4(4):712–719, 1983.

[33] Iain S Duff, Albert Maurice Erisman, and John Ker Reid. *Direct methods for sparse matrices*. Clarendon Press Oxford, 1986.

[34] Haimonti Dutta, Chris Giannella, Kirk D Borne, and Hillol Kargupta. Distributed top-k outlier detection from astronomy catalogs using the DEMAC system. In *SDM*, pages 473–478. SIAM, 2007.

[35] DA D'Ippolito, JR Myra, and SJ Zweben. Convective transport by intermittent blob-filaments: Comparison of theory and experiment. *Physics of Plasmas (1994-present)*, 18(6):060501, 2011.

[36] Manzoor Elahi, Kun Li, Wasif Nisar, Xinjie Lv, and Hongan Wang. Efficient clustering-based outlier detection algorithm for dynamic data stream. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on*, volume 5, pages 298–304. IEEE, 2008.

[37] AM Erisman and WF Tinney. On computing certain elements of the inverse of a sparse matrix. *Communications of the ACM*, 18(3):177–179, 1975.

[38] Eleazar Eskin. Anomaly detection over noisy data using learned probability distributions. In *In Proceedings of the International Conference on Machine Learning*, pages 255–262. Morgan Kaufmann, 2000.

[39] Shaoqiang Feng and Zhongxiao Jia. A refined Jacobi–Davidson method and its correction equation. *Computers & Mathematics with Applications*, 49(2):417–427, 2005.

[40] Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[41] G Fuchert, G Birkenmeier, B Nold, M Ramisch, and U Stroth. The influence of plasma edge dynamics on blob properties in the stellarator TJ-K. *Plasma Physics and Controlled Fusion*, 55(12):125002, 2013.

[42] Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, 2(2):205–224, 1965.

[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, 1981.

[44] Gene H Golub and Charles F Van Loan. An analysis of the total least squares problem. *SIAM Journal on Numerical Analysis*, 17(6):883–893, 1980.

[45] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[46] Gene H. Golub and Qiang Ye. An inverse free preconditioned Krylov subspace method for symmetric generalized eigenvalue problems. *SIAM J. Sci. Comput.*, 24(1):312–334, 2002.

[47] Justin Grimmer. We are all social scientists now: how big data, machine learning, and causal inference work together. *PS: Political Science & Politics*, 48(01):80–83, 2015.

[48] Thorkell Gudmundsson, Charles S Kenney, and Alan J Laub. Small-sample statistical estimates for matrix norms. *SIAM Journal on Matrix Analysis and Applications*, 16(3):776–792, 1995.

[49] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, pages 73–84. ACM, 1998.

[50] H. Guo. Computing traces of functions of matrices. *A Journal of Chinese Universities (English series)*, 2:204–215, 2000.

[51] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):1–1, 2014.

[52] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, May 2011.

[53] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[54] Frank E Harrell. *Regression modeling strategies*. Springer Science & Business Media, 2001.

[55] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003.

[56] Matthew Herland, Taghi M Khoshgoftaar, and Randall Wald. A review of data mining using big data in health informatics. *Journal of Big Data*, 1(1):1–35, 2014.

[57] Vicente Hernandez and Jose E. Roman. A robust and efficient parallel SVD solver based on restarted Lanczos bidiagonalization. *Electronic Transactions on Numerical Analysis*, 31:68–85, 2008.

[58] Vicente Hernandez, Jose E. Roman, and Vicente Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Softw.*, 31(3):351–362, 2005.

[59] Michiel E. Hochstenbach. A Jacobi–Davidson type SVD method. *SIAM J. Sci. Comput.*, 23(2):606–628, 2001.

[60] Michiel E. Hochstenbach. Harmonic and refined extraction methods for the singular value problem, with applications in least squares problems. *BIT Numerical Mathematics*, 44(4):721–754, 2004.

[61] Victoria J Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[62] Cho-Jui Hsieh and Peder Olsen. Nuclear norm minimization via active subspace selection. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 575–583, 2014.

[63] Edward Hung and David W Cheung. Parallel mining of outliers in large database. *Distributed and Parallel Databases*, 12(1):5–26, 2002.

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

[65] Zhongxiao Jia. Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems. *Linear algebra and its applications*, 259:1–23, 1997.

[66] Zhongxiao Jia. A refined subspace iteration algorithm for large sparse eigenproblems. *Applied Numerical Mathematics*, 32(1):35–52, 2000.

[67] Zhongxiao Jia. The refined harmonic Arnoldi method and an implicitly restarted refined algorithm for computing interior eigenpairs of large matrices. *Applied Numerical Mathematics*, 42(4):489–512, 2002.

[68] Zhongxiao Jia and Datian Niu. An implicitly restarted refined bidiagonalization Lanczos method for computing a partial singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 25(1):246–265, 2003.

[69] Zhongxiao Jia and Datian Niu. A refined harmonic Lanczos bidiagonalization method and an implicitly restarted algorithm for computing the smallest singular triplets of large matrices. *SIAM J. Sci. Comput.*, 32(2):714–744, 2010.

[70] Zhongxiao Jia and GW Stewart. An analysis of the Rayleigh–Ritz method for approximating eigenspaces. *Mathematics of Computation*, 70(234):637–647, 2001.

[71] CS Kenney, Alan J Laub, and MS Reese. Statistical condition estimation for linear systems. *SIAM Journal on Scientific Computing*, 19(2):566–583, 1998.

[72] J. Kiefer. Optimum sequential search and approximation methods under minimum regularity assumptions. *J. Soc. Indust. Appl. Math.*, 5:105–136, 1957.

[73] Edwin M Knox and Raymond T Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the International Conference on Very Large Data Bases*, pages 392–403. Citeseer, 1998.

[74] Effrosyni Kokiopoulou, C. Bekas, and E. Gallopoulos. Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization. *Applied Numerical Mathematics*, 49(1):39–61, 2004.

[75] S Ku, CS Chang, and PH Diamond. Full-f gyrokinetic particle simulation of centrally heated global itg turbulence from magnetic axis to edge pedestal top in a realistic tokamak geometry. *Nuclear Fusion*, 49(11):115021, 2009.

[76] Ralph Kube, Odd Erik Garcia, Brian LaBombard, JL Terry, and SJ Zweben. Blob sizes and velocities in the Alcator C-Mod scrape-off layer. *Journal of Nuclear Materials*, 438:S505–S508, 2013.

[77] Rasmus Munk Larsen. Lanczos bidiagonalization with partial reorthogonalization. *DAIMI Report Series*, 27(537), 1998.

[78] Rasmus Munk Larsen. Combining implicit restarts and partial reorthogonalization in Lanczos bidiagonalization. *SCCM, Stanford University*, 2001.

[79] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, volume 6. Siam, 1998.

[80] Qiao Liang and Qiang Ye. Computing singular values of large matrices with an inverse-free preconditioned Krylov subspace method. *Electronic Transactions on Numerical Analysis*, 42:197–221, 2014.

[81] Frank Lin and William W Cohen. Power iteration clustering. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 655–662, 2010.

[82] 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. *ACM Trans. Math. Softw.*, 37(4):40, 2011.

[83] Jay F Lofstead, Scott Klasky, Karsten Schwan, Norbert Podhorszki, and Chen Jin. Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS). In *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*, pages 15–24. ACM, 2008.

[84] Nicole S Love and Chandrika Kamath. Image analysis for the identification of coherent structures in plasma. *Optical Engineering+ Applications*, pages 66960D–66960D, 2007.

[85] Elio Lozano and E Acufia. Parallel algorithms for distance-based and density-based outliers. In *Data Mining, Fifth IEEE International Conference on*, pages 4–pp. IEEE, 2005.

[86] Chang-Tien Lu and Lily R Liang. Wavelet fuzzy classification for detecting and tracking region outliers in meteorological data. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 258–265. ACM, 2004.

[87] Thomas R Lucas. Error bounds for interpolating cubic splines under various end conditions. *SIAM Journal on Numerical Analysis*, 11(3):569–584, 1974.

[88] David JC MacKay. Introduction to monte carlo methods. In *Learning in graphical models*, pages 175–204. Springer, 1998.

[89] Michele Martone. Efficient multithreaded untransposed, transposed or symmetric sparse matrix–vector multiplication with the recursive sparse blocks format. *Parallel Computing*, 40(7):251–270, 2014.

[90] Gérard Meurant. Estimates of the trace of the inverse of a symmetric matrix using the modified Chebyshev algorithm. *Numerical Algorithms*, 51(3):309–318, 2009.

[91] Ronald B. Morgan. Computing interior eigenvalues of large matrices. *Linear Algebra and its Applications*, 154:289–309, 1991.

[92] 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.

[93] SH Müller, A Diallo, A Fasoli, I Furno, B Labit, G Plyushchev, M Podestà, and FM Poli. Probabilistic analysis of turbulent structures from two-dimensional plasma imaging. *Physics of Plasmas (1994-present)*, 13(10):100701, 2006.

[94] JR Myra, WM Davis, DA D'Ippolito, B LaBombard, DA Russell, JL Terry, and SJ Zweben. Edge sheared flows and the dynamics of blob-filaments. *Nuclear Fusion*, 53(7):073013, 2013.

[95] Erich Novak. Quadrature formulas for monotone functions. *Proceedings of the American Mathematical Society*, 115(1):59–68, 1992.

[96] Jeppe Olsen, Poul Jørgensen, and Jack Simons. Passing the one-billion limit in full configuration-interaction (FCI) calculations. *Chemical Physics Letters*, 169(6):463–472, 1990.

[97] Matthew Eric Otey, Amol Ghoting, and Srinivasan Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery*, 12(2-3):203–228, 2006.

[98] Chris C Paige, Beresford N Parlett, and Henk A Van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numerical linear algebra with applications*, 2(2):115–133, 1995.

[99] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, 1980.

[100] B. Philippe and M. Sadkane. Computation of the fundamental singular subspace of a large matrix. *Linear algebra and its applications*, 257:77–104, 1997.

[101] Dragoljub Pokrajac, Aleksandar Lazarevic, and Longin Jan Latecki. Incremental local outlier detection for data streams. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pages 504–515. IEEE, 2007.

[102] Peter D Robinson and Andrew J Wathen. Variational bounds on the entries of the inverse of a matrix. *IMA journal of numerical analysis*, 12(4):463–486, 1992.

[103] JE Roman. Practical implementation of harmonic Krylov–Schur. Technical report, Technical Report STR-9, Universidad Politécnica de Valencia, available at< http://www. grycap. upv. es/slepc, 2009.

[104] Farbod Roosta-Khorasani and Uri Ascher. Improved bounds on sample size for implicit matrix trace estimators. *Foundations of Computational Mathematics*, pages 1–26, 2014.

[105] Youcef Saad. *Numerical methods for large eigenvalue problems*, volume 158. SIAM, 1992.

[106] Yousef Saad and Maria Sosonkina. Enhanced preconditioners for large sparse least squares problems. Technical Report UMSI-2001-1, Minnesota Supercomputer Institute, University of Minnesota, 2001.

[107] Shiblee Sadik and Le Gruenwald. Research issues in outlier detection for data streams. *ACM SIGKDD Explorations Newsletter*, 15(1):33–40, 2014.

[108] DS Scott. The advantages of inverted operators in Rayleigh–Ritz approximations. *SIAM Journal on Scientific and Statistical Computing*, 3(1):68–75, 1982.

[109] Shashi Shekhar, Chang-Tien Lu, and Pusheng Zhang. A unified approach to detecting spatial outliers. *GeoInformatica*, 7(2):139–166, 2003.

[110] J. Shen, G. Strang, and A. J. Wathen. The potential theory of several intervals and its applications. *Applied Mathematics and Optimization*, 44(1):67–85, 2001.

[111] Gerard LG Sleijpen and Jasper van den Eshof. On the use of harmonic Ritz pairs in approximating internal eigenpairs. *Linear algebra and its applications*, 358(1):115–137, 2003.

[112] Gerard LG Sleijpen and Henk A Van der Vorst. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM Review*, 42(2):267–293, 2000.

[113] Gerard LG Sleijpen, Henk A van der Vorst, and Ellen Meijerink. Efficient expansion of subspaces in the Jacobi–Davidson method for standard and generalized eigenproblems. *Electron. Trans. Numer. Anal*, 7(75-89):8, 1998.

[114] A. Stathopoulos. A case for a biorthogonal Jacobi-Davidson method: restarting and correction equation. *SIAM J. Matrix Anal. Appl.*, 24(1):238–259, 2002.

[115] Andreas Stathopoulos. Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. Part I: Seeking one eigenvalue. *SIAM J. Sci. Comput.*, 29(2):481–514, 2007.

[116] 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.

[117] Andreas Stathopoulos and James R. McCombs. Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. Part II: Seeking many eigenvalues. *SIAM J. Sci. Comput.*, 29(5):2162–2188, 2007.

[118] Andreas Stathopoulos and James R. McCombs. PRIMME: Preconditioned iterative multimethod eigensolver: Methods and software description. *ACM Trans. Math. Softw.*, 37(2):21:1–21:30, April 2010.

[119] Andreas Stathopoulos and Konstantinos Orginos. Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to

quantum chromodynamics. *SIAM Journal on Scientific Computing*, 32(1):439–462, 2010.

[120] Andreas Stathopoulos, Yousef Saad, and Kesheng Wu. Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods. *SIAM J. Sci. Comput.*, 19(1):227–245, 1998.

[121] J. M. Steele. Certifying smoothness of discrete functions and measuring legitimacy of images. *Journal of Complexity*, (5):261–270, 1989.

[122] Gilbert W. Stewart. *Matrix Algorithms Volume 2: Eigensystems*. Society for Industrial and Applied Mathematics, 2001.

[123] Sharmila Subramaniam, Themis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the International Conference on Very Large Data Bases*, pages 187–198. VLDB Endowment, 2006.

[124] Aleksei G. Sukharev. The concept of sequential optimality for problems in numerical analysis. *Journal of Complexity*, 3(3):347–357, 1987.

[125] Jok M Tang and Yousef Saad. Domain-decomposition-type methods for computing the diagonal of a matrix inverse. *SIAM Journal on Scientific Computing*, 33(5):2823–2847, 2011.

[126] 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, 2012.

[127] Lloyd N. Trefethen. *Computation of Pseudospectra*, volume 8. Cambridge University Press, 1999.

[128] Henk A van der Vorst. *Computational Methods For Large Eigenvalue Problems*, volume 8. Elsevier, 2002.

[129] Mei Ning Wong, Fred J Hickernell, and Kwong Ip Liu. Computing the trace of a function of a sparse matrix via Hadamard-like sampling. Technical report, Department of Mathematics, Hong Kong Baptist University, 2004.

[130] Kesheng Wu, Sean Ahern, E Wes Bethel, Jacqueline Chen, Hank Childs, Estelle Cormier-Michel, Cameron Geddes, Junmin Gu, Hans Hagen, Bernd Hamann, et al. FastBit: interactively searching massive data. In *Journal of Physics: Conference Series*, volume 180, page 012053. IOP Publishing, 2009.

[131] Kesheng Wu, Ekow Otoo, and Kenji Suzuki. Optimizing two-pass connected-component labeling algorithms. *Pattern Analysis and Applications*, 12(2):117–135, 2009.

[132] Kesheng Wu and Horst Simon. Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 22(2):602–616, 2000.

[133] Kesheng Wu, Rishi R Sinha, Chad Jones, Stephane Ethier, Scott Klasky, Kwan-Liu Ma, Arie Shoshani, and Marianne Winslett. Finding regions of interest on toroidal meshes. *Computational Science & Discovery*, 4(1):015003, 2011.

[134] Lingfei Wu*, Jie Chen*, Kartik Audhkhasi, Brian Kingsbury, and Bhuvana Ramabhadran (*equal contribution). Efficient one-vs-one kernel ridge regression for speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2454 – 2458, 2016.

[135] Lingfei Wu and Andreas Stathopoulos. Enhancing the PRIMME eigensolver for computing accurately singular triplets of large matrices. Technical Report WM-CS-2014-03, Department of Computer Science, College of William and Mary, 2014.

[136] Lingfei Wu and Andreas Stathopoulos. PRIMME_SVDS: A preconditioned SVD solver for computing accurately singular triplets of large matrices based on the

PRIMME eigensolver. Technical Report WM-CS-2014-06, Department of Computer Science, College of William and Mary, 2014.

[137] Lingfei Wu and Andreas Stathopoulos. High-performance algorithms for large-scale singular value problems and big data applications. *ACM/IEEE SC15 Doctoral Showcase*, 2015.

[138] Lingfei Wu and Andreas Stathopoulos. A preconditioned hybrid SVD method for accurately computing singular triplets of large matrices. *SIAM Journal on Scientific Computing*, 37(5):S365–S388, 2015.

[139] Lingfei Wu, Andreas Stathopoulos, Jesse Laeuchli, Vassilis Kalantzis, and Efstratios Gallopoulos. Estimating the trace of the matrix inverse by interpolating from the diagonal of an approximate inverse. Technical Report WM-CS-2015-02, Department of Computer Science, College of William and Mary, 2015.

[140] Lingfei Wu, Andreas Stathopoulos, Jesse Laeuchli, Vassilis Kalantzis, and Efstratios Gallopoulos. Estimating the trace of the matrix inverse by interpolating from the diagonal of an approximate inverse. *The Journal of Computational Physics, (second-round review)*, 2015.

[141] Lingfei Wu, Andreas Stathopoulos, and Eloy Romero. A high-performance preconditioned SVD solver for accurately computing large-scale singular value problems in primme. *ACM/IEEE SC15 ACM SRC Poster*, 2015.

[142] Lingfei Wu, Kesheng Wu, Alex Sim, Michael Churchill, Jong Y Choi, Andreas Stathopoulos, CS Chang, and Scott Klasky. High-performance outlier detection algorithm for finding blob-filaments in plasma. *Proc. of 5rd International Workshop on Big Data Analytics: Challenges and Opportunites (BDAC-14), held in conjunction with ACM/IEEE SC14*, 2014.

[143] Lingfei Wu, Kesheng Wu, Alex Sim, Michael Churchill, Jong Y Choi, Andreas Stathopoulos, Cs Chang, and Scott Klasky. Towards real-time detection and tracking of blob-filaments in fusion plasma big data. *The IEEE Transaction on Big Data, (second-round review)*, 2015.

[144] Lingfei Wu, Kesheng Wu, Alex Sim, and Andreas Stathopoulos. Real-time outlier detection algorithm for finding blob-filaments in plasma. *ACM/IEEE SC14 ACM SRC Poster*, 2014.

[145] Lingfei Wu*, Ian E.H. Yen*, Jie Chen, and Rui Yan (*equal contribution). Revisiting random binning feature: Fast convergence and strong parallelizability. In *22st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[146] Zhang Xianyi, Wang Qian, and Zhang Yunquan. Model-driven level 3 blas performance optimization on Loongson 3A processor. In *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*, pages 684–691. IEEE, 2012.

[147] GS Xu, BN Wan, W Zhang, QW Yang, L Wang, and YZ Wen. Multiscale coherent structures in tokamak plasma turbulence. *Physics of Plasmas (1994-present)*, 13(10):102509, 2006.

[148] M Xu, PH Diamond, GR Tynan, C Holland, P Manz, N Fedorczak, S Chakraborty Thakur, JH Yu, KJ Zhao, JQ Dong, et al. Turbulent eddy-mediated particle, momentum, and vorticity transport in the edge of HL-2A tokamak plasma. In *24th IAEA Fusion Energy Conference, San Diego*, 2012.

[149] Chao Yang. Solving large-scale eigenvalue problems in SciDAC applications. In *Journal of Physics: Conference Series*, volume 16, page 425. IOP Publishing, 2005.

[150] Jiang Zhao, Chang-Tien Lu, and Yufeng Kou. Detecting region outliers in meteorological data. In *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*, pages 49–55. ACM, 2003.

[151] SJ Zweben. Search for coherent structure within tokamak plasma turbulence. *Physics of Fluids*, 28(3):974, 1985.