2015

Singular Value Computation and Subspace Clustering

Qiao Liang
University of Kentucky, qiao.liang@uky.edu

Recommended Citation

Liang, Qiao, "Singular Value Computation and Subspace Clustering" (2015). *Theses and Dissertations--Mathematics*. 30.
https://uknowledge.uky.edu/math_etds/30

Qiao Liang, Student

Dr. Qiang Ye, Major Professor

Dr. Peter Perry, Director of Graduate Studies

SINGULAR VALUE COMPUTATION AND SUBSPACE CLUSTERING

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Arts and Sciences
at the University of Kentucky

By
Qiao Liang
Lexington, Kentucky

Director: Dr. Qiang Ye, Professor of Mathematics
Lexington, Kentucky 2015

ABSTRACT OF DISSERTATION

SINGULAR VALUE COMPUTATION AND SUBSPACE CLUSTERING

In this dissertation we discuss two problems. In the first part, we consider the problem of computing a few extreme eigenvalues of a symmetric definite generalized eigenvalue problem or a few extreme singular values of a large and sparse matrix. The standard method of choice of computing a few extreme eigenvalues of a large symmetric matrix is the Lanczos or the implicitly restarted Lanczos method. These methods usually employ a shift-and-invert transformation to accelerate the speed of convergence, which is not practical for truly large problems. With this in mind, Golub and Ye proposes an inverse-free preconditioned Krylov subspace method, which uses preconditioning instead of shift-and-invert to accelerate the convergence. To compute several eigenvalues, Wielandt is used in a straightforward manner. However, the Wielandt deflation alters the structure of the problem and may cause some difficulties in certain applications such as the singular value computations. So we first propose to consider a deflation by restriction method for the inverse-free Krylov subspace method. We generalize the original convergence theory for the inverse-free preconditioned Krylov subspace method to justify this deflation scheme. We next extend the inverse-free Krylov subspace method with deflation by restriction to the singular value problem. We consider preconditioning based on robust incomplete factorization to accelerate the convergence. Numerical examples are provided to demonstrate efficiency and robustness of the new algorithm.

In the second part of this thesis, we consider the so-called subspace clustering problem, which aims for extracting a multi-subspace structure from a collection of points lying in a high-dimensional space. Recently, methods based on self expressiveness property (SEP) such as Sparse Subspace Clustering and Low Rank Representations have been shown to enjoy superior performances than other methods. However, methods with SEP may result in representations that are not amenable to clustering through graph partitioning. We propose a method where the points are expressed in terms of an orthonormal basis. The orthonormal basis is optimally chosen in the sense that the representation of all points is sparsest. Numerical results are given to illustrate the effectiveness and efficiency of this method.

KEYWORDS: singular value decomposition, inverse-free preconditioned Krylov subspace method, machine learning, subspace clustering

Author's signature: Qiao Liang

Date: October 8, 2015

SINGULAR VALUE COMPUTATION AND SUBSPACE CLUSTERING

By

Qiao Liang

Director of Dissertation: Qiang Ye

Director of Graduate Studies: Peter Perry

Date: October 8, 2015

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my advisor, Professor Qiang Ye, who has expertly guided me through my doctorate. His immense knowledge and invaluable ideas helped me to complete this dissertation.

Secondly, I am highly indebted to my committee members, Professor Russell Brown, Professor Russell Carden and Professor Yuming Zhang for their insightful suggestions and comments. I would also like to thank many other professors whom I took classes in University of Kentucky.

Finally, and most importantly, I would like to thank my parents, my brothers and my girlfriend for their continuous support through my life. Without their encouragements, I would not have made it this far.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

In this dissertation, we consider three problems, the definite symmetric generalized eigenvalue problem where we find $\lambda \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^n$ such that

$$\mathbf{Ax} = \lambda \mathbf{Bx}, \tag{1.1}$$

(where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ are symmetric and $\mathbf{B}$ is positive definite), the singular value problem for a matrix $\mathbf{C} \in \mathbb{R}^{m \times n}$, where we find $\sigma \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^m, \mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{Cu} = \sigma \mathbf{v}$ and $\mathbf{C}^T \mathbf{v} = \sigma \mathbf{u}$, and a subspace clustering problem which is to cluster a given dataset such that each cluster lies in a single subspace of low dimension.

1.1 Generalized eigenvalue problem

The eigenvalue problem (1.1), also referred to as a pencil eigenvalue problem (A, B), arises in many scientific and engineering applications, such as structural dynamics, quantum mechanics, and machine learning. The matrices involved in these applications are usually large and sparse and only a few of the eigenvalues are desired. Computing a few selected eigenvalues of a large symmetric matrix is a subject that has been well-studied in the last few decades; see [6, 71] for a survey. Iterative methods such as the Lanczos algorithm [19], the implicitly restarted Lanczos method [76] (ARPACK [49]) and the Arnoldi algorithm are some of the most efficient numerical methods developed in the past few decades for computing a few eigenvalues of a large scale eigenvalue problem, see [6, 49, 71]. Their speed of convergence depends on the spectral distribution of (1.1) and they may suffer from slow convergence when the desired eigenvalues are not well separated.

Preconditioning techniques may be used to accelerate the convergence of these iterative methods [19, 25] . One of the most effective techniques is the shift-and-invert transformation. This requires inverting or factorizing a shifted matrix. For sparse matrices, factorization may create excessive fill-ins of the zero entries, which results in significant memory and operation costs. When the factorization of the shifted matrix is inefficient or infeasible, several methods have been developed that employ either inexact shift-and-invert or some preconditioning transformations. The Jacobi-Davidson method [74], the JDCG algorithm [65], the locally preconditioned conjugate gradient method (LOBPCG) [42, 43], and the inverse free preconditioned Krylov subspace method [32, 62] are some of the methods in this class. There is a large body of literature on various aspects of the large symmetric eigenvalue problem; see [3, 4, 6, 26, 31, 63, 71, 77, 90] and the references contained therein for more discussions.

The inverse-free precondioned Krylov subspace method of [32] is a Krylov subspace projection method that computes the smallest (or the largest) eigenvalues of (1.1). The method is called inverse-free, because it is based on an inner-outer iteration that does not require the inversion of $\mathbf{B}$ or any shifted matrix $\mathbf{A} - \lambda\mathbf{B}$. Given an approximate eigenvector $\mathbf{x}_k$ and its Rayleigh quotient ρ_k, it approximates the smallest eigenvalue iteratively through the Rayleigh-Ritz projection on the Krylov subspace

$$\mathcal{K}_m(\mathbf{H}_k, \mathbf{x}_k) := \operatorname{span}\{\mathbf{x}_k, \mathbf{H}_k\mathbf{x}_k, \mathbf{H}_k^2\mathbf{x}_k, \ldots, \mathbf{H}_k^m\mathbf{x}_k\} \tag{1.2}$$

where $\mathbf{H}_k := \mathbf{A} - \rho_k\mathbf{B}$. It is proved in [32] that this method converges at least linearly with a rate determined by the spectral separation of the smallest eigenvalue of H_k. This convergence theory leads to a preconditioning scheme that accelerates the convergence through some equivalent congruent transformation based on incomplete factorizations. This procedure, however, computes one eigenvalue (the smallest) only. To compute additional eigenvalues, a deflation technique needs to be used. Note that a block version developed in [70] can compute several eigenvalues simultaneously, but it is efficient largely for severely clustered eigenvalues.

Deflation processes are standard methods used by iterative eigenvalue algorithms to compute additional eigenvalues after some eigenvalues have converged. Two widely used deflation techniques are the Wielandt deflation (or known as deflation by subtraction) where the matrix is modified with a low rank perturbation to move converged eigenvalue to a different part of spectrum, and deflation by restriction where approximate eigenvectors and relevant subspaces are projected to the orthogonal complement of the converged eigenvectors, see [67, 71, 89]. Both of these deflation schemes can be used in the standard Lanczos and Arnoldi algorithms. There are variations of these schemes that are suitable for some particular methods. For example, the implicitly restarted Arnoldi algorithm [50, 51, 49] employs an elaborate deflation scheme (locking and purging) that is similar to deflation by restriction. The Jacobi-Davidson method [74, 75] incorporates a partial Schur decomposition deflation.

For the inverse-free preconditioned Krylov subspace method [32, 62], a natural deflation scheme is the Wielandt deflation where the method is implicitly applied to a low rank modified problem. The Wielandt deflation alters the structure of the problem which may cause some difficulties in some applications, therefore, it is more desirable to work with the original problem without the low rank modification. The deflation by restriction method projects the search subspace to the subspace that is $\mathbf{B}$-orthogonal to the space spanned by the computed eigenvectors. Although this can be applied in formality to the inverse-free preconditioned Krylov subspace method, its convergence properties are not known.

We formulate a deflation by restriction scheme for the inverse-free precondioned Krylov subspace method for computing a few extreme eigenvalues of the definite symmetric generalized eigenvalue problem $\mathbf{Ax} = \lambda\mathbf{Bx}$. The convergence theory for the inverse-free preconditioned Krylov subspace method is generalized to include this deflation scheme.

1.2 Singular value problem

Computing a few extreme singular values of a large matrix $\mathbf{C} \in \mathbb{R}^{m\times n}(m > n)$ can be addressed by computing a few eigenvalues of either $\mathbf{A} = \mathbf{C}^T\mathbf{C}$ or its augmented matrix $\mathbf{M} = \begin{bmatrix} \mathbf{O} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{O} \end{bmatrix}$. As a special eigenvalue problem, we can apply the Lanczos algorithm or the implicitly restarted Lanczos algorithm to either $\mathbf{M}$ or $\mathbf{A}$ and this can often be done implicitly. Indeed, several methods have been introduced that exploit the special structure and the associated properties of these eigenvalue problems. The Lanczos bidiagonalization based methods discussed in [29, 5, 44, 12, 38, 80] are widely used for the singular value problems that implicitly applies the Lanczos method to $\mathbf{A} = \mathbf{C}^T\mathbf{C}$. These methods work well when the corresponding eigenvalue is reasonably well separated. However, their convergence may be slow if the eigenvalue is clustered, which turns out to be often the case when computing the smallest singular values through $\mathbf{A}$. On the other hand, for formulation $\mathbf{M}$, the smallest singular value is an interior eigenvalue of M, for which a direct application of the Lanczos algorithm does not usually result in convergence.

The shift-and-invert transformation is a standard method to deal with clustering or to compute interior eigenvalues. However, for a non-square matrix $\mathbf{C} \in \mathbb{R}^{m\times n}$ $(m > n)$, a subtle difficulty arises in using the shift-and-invert transformations for $\mathbf{M}$ because $\mathbf{M}$ is singular and, with a shift close to 0, the method often converges to one of the $m - n$ zero eigenvalues of $\mathbf{M}$ rather than to the smallest singular value of $\mathbf{C}$. On the other hand, one can avoid the shift-and-invert by considering the Jacobi-Davidson method on the augmented matrix $\mathbf{M}$ and a method of this type, called JDSVD, has been developed in [35] that efficiently exploits the block structure of $\mathbf{M}$. However, the convergence of JDSVD appears to strongly depend on the quality of preconditioning. This demands a good preconditioner for $\mathbf{M}$ or $\mathbf{M} - \mu\mathbf{I}$, which is unfortunately difficult to construct when $m \neq n$ owing to the singularity of $\mathbf{M}$.

We present an efficient algorithm for computing a few extreme singular values of a large sparse $m \times n$ matrix $\mathbf{C}$. Our algorithm is based on reformulating the singular value problem as an eigenvalue problem for $\mathbf{C}^T\mathbf{C}$. To address the clustering of singular values, we develop an inverse-free preconditioned Krylov subspace method to accelerate convergence. We consider preconditioning that is based on robust incomplete factorizations and we discuss various implementation issues. For deflation, the deflation by restriction can be used here without altering the structure of the eigenvalue problem $\mathbf{C}^T\mathbf{C}$. Indeed, it is the deflation for the singular value problem that motivates us to investigate the deflation for the inverse-free Krylov subspace method. The new algorithm, which we call `svdifp`, overcomes difficulites of computing smallest singular values experienced by some algorithms, such as the Matlab built-in function `svds`, `jdsvd`, `irlba`, etc.

1.3 Subspace clustering problem

Real world data are high-dimensional. It is infeasible to process them directly in their raw forms even though the computational power has grown exponentially in the past few years. Since the intrinsic dimensionality (the minimum number of variables required to capture the structure within the data [88]) is usually much smaller than the ambient dimension, recovering low dimensional structures in data will have a significant reduction in the computational cost and memory requirements of algorithms dealing with data while maintaining the performance. This fact has motivated the development of various techniques of finding low-dimensional representations of high dimensional data. One of the most popular techniques, Principle Component Analysis(PCA), assumes that the high dimensional data lies in a single low dimensional subspace of the ambient space. However, in practice, a given dataset may not be well described by a single subspace. Instead, they may be drawn from multiple subspaces. For instances, motions of different objects in a video sequence [84], face images of different people [34] are generally distributed in a union of multiple linear subspaces. Therefore, a more reasonable model should simultaneously cluster data into multiple groups and fit each group by a linear subspace. That is known as the subspace clustering problem.

Subspace clustering was first proposed as an extension of feature selection in data mining community. For a review of those methods, see [68]. Subspace clustering was also considered in machine learning and computer vision communities where it aims to find the hidden structure which is a union of arbitrary subspaces in the dataset, see [81] for a comprehensive review. A number of subspace clustering algorithms have been proposed based on algebraic, iterative, statistical or spectral clustering approaches (See Section 4.2). Among them, spectral clustering based methods are most effective. These methods try to build a similarity graph whose vertices are data points and weighted edges represent the similarities between data points and then apply spectral clustering methods to cluster the dataset into different clusters such that it has high inter-cluster similarities and low intra-cluster similarities.

The state-of-the-art spectral clustering based methods, such as Sparse Subspace Clustering (SSC) [22] and Low Rank Representation (LRR) [56], build the similarity matrix based on the self-expressiveness property of the dataset, i.e., the data point in a subspace can be well expressed as a linear combination of other points in the same subspace. LRR can recover subspaces if they are independent by finding the lowest rank representation, but fails if the subspaces have large intersections with each other. SSC tries to find the sparest representation. While SSC can successfully separate data points in different subspaces, it may potentially separate out the data points lying in the same subspaces.

We propose a novel subspace clustering algorithm based on dictionary learning. Our algorithm constructs a dictionary of specific type and finds a compact representation of the data with respect to the dictionary simultaneously. Specifically, we describe a method that learns a union of orthonormal bases as the dictionary so that the data have a block structured representation with the learned dictionary. A bipartite graph clustering method is then used to cluster the data into different clusters.

1.4 Outline

The rest of the dissertation is organized as follows.

In Chapter 2, We introduce the inverse-free preconditioned Krylov subspace method and then formulate a deflation by restriction scheme. We provide some global and local convergence results which are extensions of the theory in [32].

In Chapter 3, we develop the inverse-free preconditioned Krylov subspace method for the singular value problem. We presents a robust incomplete factorization(RIF) preconditioner in order to accelerate the convergence. We also briefly describe a MATLAB implementation called `svdifp` that we have developed.

In Chapter 4, we present the motivation of our novel subspace clustering algorithm and its detailed implementation. We demonstrate the effectiveness of our algorithm on several synthetic and real-world datasets.

We conclude our works in Chapter 5.

1.5 Notation

The ij-entry of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}_{ij}$, $\mathbf{A}(i,j)$ or a_{ij} and the j-th column is denoted by $\mathbf{a}_j$. $\mathbf{A}^T$ denotes the transpose of a real matrix $\mathbf{A}$. We use $\lambda_i(\mathbf{A})$ to denote its i-th smallest eigenvalue and $\lambda_{-i}(\mathbf{A})$ to denote its i-th largest eigenvalues. Similarly, $\sigma_i(\mathbf{A})$ and $\sigma_{-i}(\mathbf{A})$ denote the i-th smallest and largest singular values of $\mathbf{A}$ respectively.. Calligraphic and blackboard bold letters are used to denote spaces and sets. In particular, real Euclidean space of n dimension is denoted by $\mathbb{R}^n$. $\mathbb{R}^{m\times n}$ denotes $m \times n$ real matrices. The range space and null space of a matrix $\mathbf{A}$ are denoted by $\mathcal{R}(\mathbf{A})$ or span($\mathbf{A}$) and $\mathcal{N}(\mathbf{A})$ respectively. For a set $\mathcal{S}$, $|\mathcal{S}|$ is the number of elements in $\mathcal{S}$.

We also introduce the notations for some special vectors and matrices. $\mathbf{I}_n$ denotes $n \times n$ identity matrix with $\mathbf{e}_i$ as its i-th column. $\mathbf{O}_{m\times n}$ denotes a zero matrix. For a vector of length n with all its entries be 1, we use $\mathbf{1}_n$. The subscripts could be omitted if they are clear in the context.

Now we review some notations and definitions of frequently used norms. Given a vector $\mathbf{v} = [v_1, \ldots, v_n]^T \in \mathbb{R}^n$, its l_p norm is defined as

$$\|\mathbf{v}\|_p := (\sum_{i=1}^{n} v_i^p)^{1/p}.$$

Throughout this dissertation, we will use $\|\cdot\|_2$ and $\|\cdot\|_1$ extensively. Let $p \to \infty$, we can obtain the infinity norm $\|\mathbf{v}\|_\infty = \max\{v_i, i = 1, \ldots, n\}$.

Another important “norm” called l_0 norm denoted by $\|\mathbf{v}\|_0$ is the number of nonzero entries in $\mathbf{v}$. Technically, l_0 is not a norm as it violates the absolute homogeneity, i.e. $\|\alpha\mathbf{v}\|_0 \neq |\alpha|\|\mathbf{v}\|_0$ for some α and $\mathbf{v}$.

Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, Forbenius norm of $\mathbf{A}$ is

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} a_{ij}^2}$$

Corresponding to the vector l_p norm, the induced l_p norm for $\mathbf{A}$ is defined as

$$\|\mathbf{A}\|_p = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_p}.$$

In particular, the induced 2-norm coincides with spectral norm, i.e.,

$$\|\mathbf{A}\|_2 = \sqrt{\lambda_{-1}(\mathbf{A}^{\mathbf{T}}\mathbf{A})} = \sigma_{-1}(\mathbf{A}).$$

In general, $\|\mathbf{A}\|_1$ is the induced l_1 norm for matrix. However, in this dissertation, we set

$$\|\mathbf{A}\|_1 = \sum_{i,j} |a_{ij}|.$$

We also use $\|\mathbf{A}\|_0$ to denote the number of non-zero entries of $\mathbf{A}$ and $\|\mathbf{A}\|_{2,0}$ to denote the number of non-zero columns of $\mathbf{A}$. The $l_{2,1}$ norm of $\mathbf{A}$ is

$$\|\mathbf{A}\|_{2,1} = \sum_{j=1}^{n} \sqrt{\sum_{i=1}^{m} a_{ij}^2}.$$

Another useful norm is max norm $\|\mathbf{A}\|_{\max} = \max_{ij} |a_{ij}|$.

Chapter 2 Inverse-free preconditioned Krylov subspace method with deflation by restriction

In this chapter, we consider the problem of computing a few eigenvalues of $(\mathbf{A}, \mathbf{B})$. We investigate the convergence behavior of the inverse-free precondioned Krylov subspace method with a different deflation scheme called deflation by restriction comprehensively. First we present relevant materials that will be used in this work. We review the generalized symmetric eigenvalue problem and Krylov subspace methods and also discuss some existing deflation methods which have been applied in solving the generalized symmetric eigenvalue problem in Section 2.1. Then we introduce the inverse-free preconditioned Krylov subspace method and the existing convergence theory of computing the extreme eigenvalues in Section 2.2.1. We show the original deflation method (Wielandt deflation) has some difficulities when we extend the method to the singular value problem. In Section 2.2.2, we propose to use deflation by restriction to obtain the additional eigenvalues. And we also prove that the additional eigenvalues computing by the inverse-free Krylov subspace method with deflation by restriction share the similar convergence behavior as that of the extreme ones. Numerical examples are presented to demonstrate the convergence properties of the algorithm with the deflation scheme in Section 2.3

2.1 Preliminaries

We briefly reviews some basic definitions, theorems and tools needed to study the inverse-free preconditoned Krylov subspace method with deflation by restriction. Most of the materials presented in this section can be found in any standard numerical linear algebra textbook, such as [19, 30, 71, 6, 89, 67].

2.1.1 Eigenvalues and eigenvectors of symmetric matrices

Definition 2.1.1. *[19] The polynomial* $p(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I})$ *is called the characteristic polynomial of* $\mathbf{A} \in \mathbb{R}^{n\times n}$*. The roots of* $p(\lambda) = 0$ *are eigenvalues of* $\mathbf{A}$*. A nonzero vector* $\mathbf{x} \in C^n$ *such that* $\mathbf{Ax} = \lambda\mathbf{x}$ *is a (right) eigenvector for the eigenvalue* λ*. A nonzero vector* $\mathbf{y} \in \mathbb{R}^n$ *with* $\mathbf{y}^T A = \lambda\mathbf{y}^T$ *is a left eigenvector.*

From Definition 2.1.1, we can conclude that the eigenvalues of a triangular matrix are its diagonal entries. It motivates the Schur canonical form.

Theorem 2.1.1. *[19] Given* $\mathbf{A} \in \mathbb{R}^{n\times n}$*, there exists an orthogonal matrix* $\mathbf{Q}$ *and an upper triangular matrix* T *such that* $Q^T AQ = T$*. The eigenvalues of* $\mathbf{A}$ *are the diagonal entries of* T*.*

In particular, T is a diagonal matrix if $\mathbf{A} = \mathbf{A}^T$. We will mainly focus on this kind of matrices in this dissertation.

Definition 2.1.2. $\mathbf{A} \in \mathbb{R}^{n\times n}$ *is a symmetric matrix if* $\mathbf{A} = \mathbf{A}^T$*.*

Symmetric matrices enjoy many simple but beautiful properties, among which is the spectral theorem as a corollary of Theorem 2.1.1.

Theorem 2.1.2. *Suppose* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *is a symmetric matrix, then* $\mathbf{A}$ *is similar to a real diagonal matrix via an orthogonal transformation, i.e.*

$$\mathbf{A} = \mathbf{Z}\boldsymbol{\Lambda}\mathbf{Z}^T \tag{2.1}$$

where $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}$ *and* $\boldsymbol{\Lambda}$ *is a diagonal matrix with real entries.*

From Theorem 2.1.2, we can see that the columns $\mathbf{z}_i$ of $\mathbf{Z}$ are eigenvectors of $\mathbf{A}$ and the diagonal entries λ_i of $\boldsymbol{\Lambda}$ are eigenvalues of $\mathbf{A}$. It is well-known that the eigenvalue problem of a symmetric matrix is well-conditioned, in other words, the absolute change in an eigenvalue is bounded by the absolute change in the matrix. The idea is interpreted by the following theorem.

Theorem 2.1.3. *Suppose* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ *are symmetric matrices, then*

$$\max_i |\lambda_i(\mathbf{A}) - \lambda_i(\mathbf{B})| \leq \|\mathbf{A} - \mathbf{B}\|_2.$$

An important class of symmetric matrices positive definite matrices.

Definition 2.1.3. *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a symmetric matrix. If $\mathbf{z}^T\mathbf{A}\mathbf{z}$ is always positive for any non-zero vector $\mathbf{z} \in \mathbb{R}^n$, we call $\mathbf{A}$ positive definite.*

By the above definition and Theorem 2.1.2, there exists an orthogonal matrix such that $\mathbf{Z}^T\mathbf{A}\mathbf{Z} = \mathbf{\Lambda}$ and $\mathbf{\Lambda}$ is a diagonal matrix with real positive entries if $\mathbf{Z}$ is positive definite. Suppose $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and

$$\mathbf{\Lambda}^{1/2} = \mathrm{diag}(\lambda_1^{1/2}, \lambda_2^{1/2}, \ldots, \lambda_n^{1/2}),$$

then $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ where $\mathbf{L} = \mathbf{Z}\mathbf{\Lambda}^{1/2}$ and $\mathbf{L}$ is invertible. Furthermore, if $\mathbf{L}$ is lower triangular with positive diagonals, then $\mathbf{L}$ is unique. It concludes the following theorem.

Theorem 2.1.4. *If $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a positive definite matrix, then there exists a unique lower triangular matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$ with positive diagonals such that $\mathbf{A} = \mathbf{L}\mathbf{L}^T$. It is the called Cholesky decomposition of $\mathbf{L}$ and $\mathbf{L}$ is the Cholesky factor of $\mathbf{A}$.*

Given a positive definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, we can also obtain a well-defined inner-product in $\mathbb{R}^n$:

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{A}} = \mathbf{x}^T\mathbf{A}\mathbf{y}.$$

We say $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ are $\mathbf{A}$-orthogonal, if $\mathbf{x}^T\mathbf{A}\mathbf{y} = 0$. A matrix $\mathbf{B} \in \mathbb{R}^{n\times m}$ is $\mathbf{A}$-orthonormal if $\mathbf{B}^T\mathbf{A}\mathbf{B} = \mathbf{I}$. The induced norm of the inner-product is $\|\mathbf{x}\|_{\mathbf{A}} = \sqrt{\mathbf{x}^T\mathbf{A}\mathbf{x}}$.

2.1.2 The generalized symmetric eigenvalue problem

Definition 2.1.4. *A scalar $\lambda \in \mathbb{R}$ is called an eigenvalue of a matrix pencil $(\mathbf{A}, \mathbf{B})$ with $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ if there exists a nonzero vector $\mathbf{x} \in \mathbb{R}^n$ such that*

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{B}\mathbf{x}. \tag{2.2}$$

The vector $\mathbf{x}$ is called an eigenvector of $(\mathbf{A}, \mathbf{B})$ associated with λ. $(\lambda, \mathbf{x})$ is called an eigenpair of $(\mathbf{A}, \mathbf{B})$. The set of all eigenvalues of $(\mathbf{A}, \mathbf{B})$ is called the spectrum of $(\mathbf{A}, \mathbf{B})$.

The problem of finding some or all of the eigenpairs of $(\mathbf{A}, \mathbf{B})$ is referred to as the generalized eigenvalue problem. The standard problem corresponds to the case when $\mathbf{B} = \mathbf{I}$. If $\mathbf{B}$ is nonsingular, then (2.2) can be rewritten as

$$\mathbf{B}^{-1}\mathbf{A}\mathbf{x} = \lambda\mathbf{A}\mathbf{x},$$

and the generalized problem is reduced to the standard one. A simpler special case which arises in practice is that $\mathbf{A}, \mathbf{B}$ are symmetric and $\mathbf{B}$ is positive definite. The generalized eigenvalue problem of definite pencil $(\mathbf{A}, \mathbf{B})$ arises in many scientific and engineering applications, such as structural dynamics, quantum mechanics, and machine learning. The matrices involved in these applications are usually large and sparse and only a few of the eigenvalues are desired.

There is a corresponding spectral theorem of definite pencil $(\mathbf{A}, \mathbf{B})$ which is similar to Theorem 2.1.2.

Theorem 2.1.5. *Suppose $\mathbf{A}, \mathbf{B} \in \mathbb{R}^n$ are symmetric and $\mathbf{B}$ is positive definite. Then there exists an invertible matrix $\mathbf{X}$ such that*

$$\mathbf{X}^T\mathbf{A}\mathbf{X} \quad \textit{and} \quad \mathbf{X}^T\mathbf{B}\mathbf{X}$$

are both real and diagonal.

Proof. Since $\mathbf{B}$ is positive definite, we can obtain the following decomposition by Theorem 2.1.4

$$\mathbf{B} = \mathbf{L}\mathbf{L}^T \tag{2.3}$$

where $\mathbf{L}$ is an invertible lower triangular matrix. Consider the matrix $\mathbf{L}^{-1}\mathbf{A}(\mathbf{L}^T)^{-1}$. Since $\mathbf{A}$ is symmetric, i.e., $\mathbf{A} = \mathbf{A}^T$, $\mathbf{L}^{-1}\mathbf{A}(\mathbf{L}^T)^{-1}$ is also symmetric. By Theorem 2.1.2, there exists an orthogonal matrix $\mathbf{Y}$ such that

$$\mathbf{L}^{-1}\mathbf{A}(\mathbf{L}^T)^{-1}\mathbf{Y} = \mathbf{Y}\mathbf{\Lambda}, \tag{2.4}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with real entries. Let

$$\mathbf{X} = (\mathbf{L}^T)^{-1}\mathbf{Y}. \tag{2.5}$$

Then $\mathbf{X}$ is an invertible matrix. Combine (2.5) with (2.3) and (2.4). We can obtain $\mathbf{AX} = \mathbf{BX\Lambda}$. Therefore

$$\mathbf{X}^T\mathbf{AX} = \mathbf{X}^T\mathbf{BX\Lambda}.$$

Since $\mathbf{X}^T\mathbf{BX} = \mathbf{Y}^T\mathbf{L}^{-1}\mathbf{B}(\mathbf{L}^T)^{-1}\mathbf{Y} = \mathbf{Y}^T\mathbf{Y} = \mathbf{I}$, $\mathbf{X}^T\mathbf{AX} = \mathbf{\Lambda}$ is diagonal and real. □

Theorem 2.1.5 is also known as simultaneous diagonalization theorem. By Definition 2.2, we can see that the columns of $\mathbf{X}$ are eigenvectors of $(\mathbf{A}, \mathbf{B})$. Furthermore, we notice that there are infinitely many choices of $\mathbf{X}$. If an $\mathbf{X}$ satisfying Theorem 2.1.5 has been found, then by simply multiplying $\mathbf{X}$ by an arbitrary diagonal matrix we can obtain another matrix satisfying Theorem 2.1.5. In general, we are only interested in $\mathbf{X}$ which satisfies $\mathbf{X}^T\mathbf{BX} = \mathbf{I}$ and $\mathbf{X}^T\mathbf{AX} = \mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, \ldots, \lambda_n\}$, where $\lambda_1 \leq \ldots \leq \lambda_n$. Then $\mathbf{AX} = \mathbf{\Lambda BX}$.

The proof of Theorem 2.1.5 provides a way to reduce the generalized eigenvalue problem of $(\mathbf{A}, \mathbf{B})$ to the standard eigenvalue problem of $\mathbf{L}^{-1}\mathbf{A}(\mathbf{L}^T)^{-1}$ where $\mathbf{L}$ is the Cholesky factor of $\mathbf{B}$.

One of the most important quantities associated with the generalized eigenvalue problem is Rayleigh quotient.

Definition 2.1.5. *The Rayleigh quotient of* $\mathbf{x}$ *for* $(\mathbf{A}, \mathbf{B})$ *is defined by*

$$\rho(\mathbf{x}; \mathbf{A}, \mathbf{B}) = \frac{\mathbf{x}^T\mathbf{Ax}}{\mathbf{x}^T\mathbf{Bx}}. \tag{2.6}$$

We frequently shorten the notation to $\rho(\mathbf{x})$ *when* $\mathbf{A}, \mathbf{B}$ *are clear from context.*

The Rayleigh quotient of $(\mathbf{A}, \mathbf{B})$ enjoys many properties which will be exploited through our work.

Lemma 2.1.1. *[67, Theorem 15.9.2] When* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ *are symmetric and* $\mathbf{B} > 0$, *then*

1. *Homogeneity:* $\rho(\alpha\mathbf{x}) = \rho(\mathbf{x})$, $\alpha \neq 0 \in \mathbb{R}$.

2. *Boundedness:* $\lambda_1 \leq \rho(\mathbf{x}) \leq \lambda_n$ *as* $\mathbf{x}$ *ranges over the unit sphere, where* $\lambda_1 \leq \ldots \leq \lambda_n$ *are eigenvalues of the pencil* $(\mathbf{A}, \mathbf{B})$.

3. *Stationary:* $\nabla\rho(\mathbf{x}) = 2(\mathbf{Ax} - \rho(\mathbf{x})\mathbf{Bx})^T/(\mathbf{x}^T\mathbf{Bx})$. *Thus* $\rho(\mathbf{x})$ *is stationary at and only at the eigenvectors of* $(\mathbf{A}, \mathbf{x})$.

4. *Minimum Residual:* $\|(\mathbf{A}-\sigma\mathbf{B})\mathbf{x}\|^2_{\mathbf{B}^{-1}} \geq \|\mathbf{Ax}\|^2_{\mathbf{B}^{-1}} - |\rho(\mathbf{x})|^2\|\mathbf{Bx}\|^2_{\mathbf{B}^{-1}}$ *with equality when and only when* $\sigma = \rho(\mathbf{x})$ *where* $\sigma \in \mathbb{R}$.

Proof. We give the proofs of part 1, 2, and 4, but not part 3 which is quite straightforward.

1. From the definition of Rayleigh quotient,

$$\rho(\alpha x) = \frac{\alpha^2(\mathbf{x}^T\mathbf{Ax})}{\alpha^2(\mathbf{x}^T\mathbf{Bx})} = \frac{\mathbf{x}^T\mathbf{Ax}}{\mathbf{x}^T\mathbf{Bx}} = \rho(\mathbf{x}).$$

2. By part (1), we only need to consider the bounds of $\rho(\mathbf{x})$ where $\mathbf{x}^T\mathbf{Bx} = 1$. By Theorem 2.1.5, we can find $\mathbf{Z}$ such that $\mathbf{AZ} = \mathbf{BZ\Lambda}$ where $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n]$ is $\mathbf{B}-$orthonormal and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$. Since $\mathbf{Z}$ is invertible, then $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ is a basis of $\mathbb{R}^n$. Hence there exist scalars $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ such that $\mathbf{x} = \alpha_1\mathbf{z}_1 + \ldots + \alpha_n\mathbf{z}_n$ and

$$\rho(\mathbf{x}) = (\sum_{i=1}^{n} \alpha_i\mathbf{z}_i)^T\mathbf{A}(\sum_{i=1}^{n} \alpha_i\mathbf{z}_i) = \sum_{i=1}^{n} \alpha_i\lambda_i.$$

Then $\lambda_1 \leq \rho(\mathbf{x}) \leq \lambda_n$ follows.

4. Since $\mathbf{x}^T\mathbf{Ax} = \rho(\mathbf{x})\mathbf{x}^T\mathbf{Bx}$,

$$\begin{aligned}\|(\mathbf{A} - \sigma\mathbf{B})\mathbf{x}\|^2_{\mathbf{B}^{-1}} &= ((\mathbf{A} - \sigma\mathbf{B})\mathbf{x})^T\mathbf{B}^{-1}(\mathbf{A} - \sigma\mathbf{B})\mathbf{x} \\ &= \mathbf{x}^T\mathbf{A}^T\mathbf{B}^{-1}\mathbf{Ax} - 2\sigma\mathbf{x}^T\mathbf{Ax} + \sigma^2\mathbf{x}^T\mathbf{Bx} \\ &= \|\mathbf{Ax}\|^2_{\mathbf{B}^{-1}} - (2\sigma\rho(\mathbf{x}) - \sigma^2)\|\mathbf{Bx}\|^2_{\mathbf{B}^{-1}} \\ &\geq \|\mathbf{Ax}\|^2_{\mathbf{B}^{-1}} - |\rho(\mathbf{x})|^2\|\mathbf{Bx}\|^2_{\mathbf{B}^{-1}}\end{aligned}$$

□

Part 2 of Lemma 2.1.1 characterizes the relation between Rayleigh quotient and the extreme eigenvalues. The renowned Courant-Fischer Minimax Theorem gives us a more general result.

Theorem 2.1.6 (Courant-Fischer Minimax Theorem). *Suppose* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ *are symmetric and* $\mathbf{B} > 0$ *and* $\lambda_1 \leq \ldots \leq \lambda_n$ *are eigenvalues of* $(\mathbf{A}, \mathbf{B})$*, then*

$$\lambda_k = \min_{\substack{\dim(\mathcal{S})=k \\ \mathcal{S}\subset\mathbb{R}^n}} \max_{\mathbf{x}\in\mathcal{S}} \rho(\mathbf{x}) = \max_{\substack{\dim(\mathcal{S})=n-k+1 \\ \mathcal{S}\subset\mathbb{R}^n}} \min_{\mathbf{x}\in\mathcal{S}} \rho(\mathbf{x}). \tag{2.7}$$

Proof. We will only show the first part. The proof of the second part is quite similar.

Suppose the eigenvectors corresponding to $\lambda_1, \ldots, \lambda_n$ are $\mathbf{z}_1, \ldots, \mathbf{z}_n$ and $\mathbf{z}_i^T \mathbf{B} \mathbf{z}_i = 1, i = 1, \ldots, n$. And let $\mathcal{S}_k$ be the subspace spanned by $\mathbf{z}_1, \ldots, \mathbf{z}_k$. By a similar proof as that of part 2 of Theorem 2.1.1, $\rho(\mathbf{x}) \leq \lambda_i$ for any non-zero vector $\mathbf{x} \in \mathcal{S}_i$ and the equality can only be obtained when $\mathbf{x}$ is a multiple of $\mathbf{z}_k$. Hence

$$\lambda_k \geq \min_{\substack{\dim(\mathcal{S})=k \\ \mathcal{S}\subset\mathbb{R}^n}} \max_{\mathbf{x}\in\mathcal{S}} \rho(\mathbf{x}). \tag{2.8}$$

Let $\mathcal{S}$ be an arbitrary subspace with $\dim(\mathcal{S}) = k$ and $\mathcal{C}_k = \text{span}\{\mathbf{z}_k, \mathbf{z}_{k+1}, \ldots, \mathbf{z}_n\}$. Since $\dim(\mathcal{C}_k) = n - k + 1$, then $\mathcal{C}_k \cap \mathcal{S} \neq \emptyset$. Suppose $\mathbf{x}_0 \in \mathcal{C}_k \cap \mathcal{S}$. Then

$$\max_{\mathbf{x}\in\mathcal{S}} \rho(\mathbf{x}) \geq \rho(\mathbf{x}_0) \geq \min_{\mathbf{x}\in\mathcal{C}_k} \rho(\mathbf{x}) \geq \lambda_k.$$

The last inequality can be proven by a similar proof as that of part 2 of Theorem 2.1.1. Hence

$$\lambda_k \leq \min_{\substack{\dim(\mathcal{S})=k \\ \mathcal{S}\subset\mathbb{R}^n}} \max_{\mathbf{x}\in\mathcal{S}} \rho(\mathbf{x}). \tag{2.9}$$

Combine (2.9) with (2.8), we can obtain the first part of Courant-Fischer min-max Theorem. □

Through Courant-Fischer Minimax Theorem, we can prove the following trace-minimization theorem.

Corollary 2.1.1. *Suppose* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ *are symmetric matrices with* $\mathbf{B} > 0$ *and* $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$ *is the solution to the minimization problem*

$$\begin{aligned} \min_{X} \quad & tr(\mathbf{X}^T \mathbf{A} \mathbf{X}) \\ s.t. \quad & \mathbf{X}^T \mathbf{B} \mathbf{X} = \mathbf{I},\ \mathbf{X} \in \mathbb{R}^{n\times k}. \end{aligned} \tag{2.10}$$

Then the columns of $\mathbf{U}$ *are eigenvectors associated with the* k *smallest eigenvalues of the definite pencil* $(\mathbf{A}, \mathbf{B})$.

In part 4 of Lemma 2.1.1, note that

$$\|\mathbf{Ax}\|^2_{\mathbf{B}^{-1}} - |\rho(\mathbf{x})|^2\|\mathbf{Bx}\|^2_{\mathbf{B}^{-1}} = \|(\mathbf{A}-\rho(\mathbf{x})\mathbf{B})\mathbf{x}\|^2_{\mathbf{B}^{-1}},$$

and $\|(\mathbf{A}-\sigma\mathbf{B})\mathbf{x}\|^2_{\mathbf{B}^{-1}}$ can obtain its minimum if and only if $\sigma = \rho(\mathbf{x})$. In other words, the best estimation to an eigenvalue in a subspace spanned by a single vector $\mathbf{x}$ is given by the Rayleigh quotient of $\mathbf{x}$ and the residual $\mathbf{A}x - \rho(\mathbf{x})\mathbf{Bx}$ is orthogonal to $\mathbf{x}$. This fact inspires a so-called Rayleigh-Ritz method to obtain estimations to the eigenpairs of $(\mathbf{A}, \mathbf{B})$, which is presented in Algorithm 2.1.

Algorithm 2.1 Rayleigh-Ritz method

```
**Input:** Two symmetric matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ with $\mathbf{B} > 0$, a subspace $\mathcal{S}_m$ with dimension $m$.
**Output:** Approximate eigenpairs $(\theta_i, \mathbf{h}_i)$, $i = 1, \ldots, m$.
Find a basis $\mathbf{Z}_m$ of $\mathcal{S}_m$.
Form $\mathbf{A}_m = \mathbf{Z}_m^T\mathbf{A}\mathbf{Z}_m$ and $\mathbf{B}_m = \mathbf{Z}_m^T\mathbf{B}\mathbf{Z}_{\mathbf{m}}$.
Find the eigenpairs $(\theta_i, \mathbf{y}_i)$, $i = 1, \ldots, m$ of $(\mathbf{A}_m, \mathbf{B}_m)$.
$\mathbf{h}_i = \mathbf{Z}_m\mathbf{y}_i$, $i = 1, \ldots, m$.
```

Given a subspace $\mathcal{S}$, Rayleigh-Ritz method finds (θ, h) with $\theta \in \mathbb{R}$ and $\mathbf{h} \in \mathcal{S}$ such that the following Galerkin condition is satisfied:

$$\mathbf{Ah} - \theta\mathbf{Bh} \perp \mathcal{S}. \tag{2.11}$$

Suppose $\mathbf{Z}$ is a $\mathbf{B}$-orthonormal matrix whose columns form a basis of $\mathcal{S}$, then $(\theta, \mathbf{h})$ is an eigenpair of $\mathbf{Z}^T\mathbf{AZ}$. And the approximate eigenpair $(\theta, \mathbf{Zh})$ to the eigenproblem of $(\mathbf{A}, \mathbf{B})$ is called a Ritz pair of $(\mathbf{A}, \mathbf{B})$ with respect to $\mathcal{S}$. The Rayleigh-Ritz approximations are optimal which can be shown by the following theorem.

Theorem 2.1.7. *Let $\mathbf{Z} \in \mathbb{R}^{n\times m}$ whose columns form a basis of a subspace $\mathcal{S} \subset \mathbb{R}^n$ with $\mathbf{Z}^T\mathbf{BZ} = \mathbf{I}$. Let $\mathbf{A}_{\mathcal{S}} = \mathbf{Z}^T\mathbf{AZ}$. Suppose $\theta_i (1 \le i \le m)$ are eigenvalues of $\mathbf{A}_{\mathcal{S}}$ and their corresponding eigenvectors are $\mathbf{y}_1, \ldots, \mathbf{y}_m$. Let $\mathbf{h}_i = \mathbf{Zy}_i$, $i = 1, \ldots, m$ and $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_m]$. Then $(\theta_i, \mathbf{h}_i)$, $i = 1, \ldots, m$ are the best approximations in $\mathcal{S}$ to the eigenpairs of $(\mathbf{A}, \mathbf{B})$ in the sense that they optimize*

$$\sum_{i=1}^{m} \|\mathbf{Ax}_i - \mu_i\mathbf{Bx}_i\|^2_{\mathbf{B}^{-1}}$$

over all $\mathbf{x}_i \in \mathcal{S}$ and $\mathbf{x}_i^T\mathbf{Bx}_j = \delta_{ij}$ where δ_{ij} is the Kronecker delta.

The proof of the above theorem follows the reduction of $(\mathbf{A}, \mathbf{B})$ to the standard eigenvalue problem of $\mathbf{L}^{-1}\mathbf{A}(\mathbf{L}^T)^{-1}$ where $\mathbf{L}$ is the Cholesky factor of $\mathbf{B}$. For complete proof, see [67].

2.1.3 Krylov subspace method

The computation of all the eigenpairs of a symmetric matrix $\mathbf{A}$ has been well-studied in the past few decades. Most of the algorithms first reduce $\mathbf{A}$ to a tridiagonal matrix and compute its eigenvalue decomposition thereafter. All of those algorithms can be applied to the generalized eigenvalue problem of the definite pencil $(\mathbf{A}, \mathbf{B})$ by first reducing it to the standard eigenvalue problem. We refer the interested readers to [67, 71, 89] for a complete review. However, in some cases such as $(\mathbf{A}, \mathbf{B})$ are large and sparse, the computation of all eigenpairs is inefficient and even infeasible. We need to consider an alternative way to compute a few extreme eigenpairs of $(\mathbf{A}, \mathbf{B})$. From the previous discussion, Rayleigh-Ritz method provides us a way to achieve this goal by considering an orthogonal projection of $(\mathbf{A}, \mathbf{B})$ onto a subspace $\mathcal{S}$ to find approximate eigenpairs. Based on Rayleigh-Ritz method, we can iteratively refine the subspace $\mathcal{S}$ to obtain better and better approximations to the eigenpairs of $(\mathbf{A}, \mathbf{B})$.

Krylov subspace methods are a class of methods which extract approximations from a specific subspace of the form

$$\mathcal{K}_m(\mathbf{A}, \mathbf{x}) := \operatorname{span}\{\mathbf{x}, \mathbf{A}\mathbf{x}, \dots, \mathbf{A}^m\mathbf{x}\}.$$

where $m > 0$ is preselected. $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$ is referred as a Krylov subspace.

Krylov subspace methods are popular because they are simple but effective. There are a few properties of the Krylov subspace $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$ which we list as follows:

Proposition 2.1.1. *1. Scaling:* $\mathcal{K}_m(\mathbf{A}, \mathbf{x}) = \mathcal{K}_m(\sigma\mathbf{A}, \tau\mathbf{x})$, $\sigma \neq 0, \tau \neq 0 \in \mathbb{R}$.

2. *Translation:* $\mathcal{K}_m(\mathbf{A}, \mathbf{x}) = \mathcal{K}_m(\mathbf{A} - \sigma\mathbf{I}, \mathbf{x})$, $\sigma \in \mathbb{R}$.

3. *Change of Basis:* $\mathcal{K}_m(\mathbf{P}\mathbf{A}\mathbf{P}^T, \mathbf{P}\mathbf{x}) = \mathbf{P}\mathcal{K}_m(\mathbf{A}, \mathbf{x})$, $\mathbf{P}^T\mathbf{P} = \mathbf{I}$.

4. *For each vector* $\mathbf{v} \in \mathcal{K}_m(A, x)$*, it can be written as* $\mathbf{v} = p(\mathbf{A})\mathbf{x}$*, where* p *is a polynomial of degree not exceeding* m*, i.e.,* $p \in \mathcal{P}^m$.

5. *The Krylov subspace $K_m(\mathbf{A}, \mathbf{x})$ is of dimension $m + 1$ if and only if the degree of the minimal polynomial of $\mathbf{x}$ with respect to $\mathbf{A}$ is larger than m.*

The scaling and translation properties of Krylov subspace $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$ indicate that there is no loss of generality if we assume $|\lambda_1(\mathbf{A})| = |\lambda_{-1}(\mathbf{A})| = 1$. Therefore, as m is increasing, $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$ is getting closer the eigenvectors associated with the extreme eigenvalues of $\mathbf{A}$.

Consider the Rayleigh-Ritz method on a symmetric matrix $\mathbf{A}$ with $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$. Let $(\theta_i, \mathbf{h}_i)$, $i = 1, \ldots, m$ are the Ritz pairs of $\mathbf{A}$ in $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$. It is well known that the bound of $|\theta_i - \lambda_i|$ is closely related to the Chebyshev polynomials. Later we will see that the error bound of inverse-free Krylov subspace method also has a strong relation with the Chebyshev polynomial.

Part 4 of Proposition 2.1.1 shows that the dimension of $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$ is usually $m+1$. Though $\{\mathbf{x}, \mathbf{A}\mathbf{x}, \ldots \mathbf{A}^m\mathbf{x}\}$ is a basis of $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$, we rarely use that in practice since the basis would be ill-conditioned as $\mathbf{A}^k\mathbf{x}$ would converge to the dominant eigenvector of $\mathbf{A}$. Suppose a suitable basis $\mathbf{Z}_m$ has been obtained for $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$, then the Krylov subspace methods will extract the desired Ritz pairs from the projected pencil $(\mathbf{Z}_m^T\mathbf{A}\mathbf{Z}_m, \mathbf{Z}_m^T\mathbf{B}\mathbf{Z}_m)$. There are various methods of forming basis of the Krylov subspace $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$, of which two well-known methods are Lanczos and Arnoldi algorithms. We shall discuss them in details later.

Given an inner product $\langle \cdot, \cdot \rangle$ defined in $\mathbb{R}^n$ and its induced norm $\|\cdot\|$, the Arnoldi algorithm constructs an orthonormal basis for $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$ by the modified Gram-Schmidt process. The Arnoldi process is shown in Algorithm 2.2.

The Arnoldi process breaks down when the computed vector w at step j vanishes. In that case, the subspace $\mathcal{K}_j(\mathbf{A}, \mathbf{x})$ is an invariant subspace of $\mathbf{A}$ and the Ritz pair of $\mathbf{A}$ on $\mathcal{K}_j(\mathbf{A}, \mathbf{x})$ is the exact eigenpair.

Let $\mathbf{H}_m = [h_{j,i}]$ be the Hessenberg matrix obtained in Algorithm 2.2. Then

$$\mathbf{A}\mathbf{Z}_m = \mathbf{Z}_m\mathbf{H}_m + h_{m+1,m}\mathbf{z}_{m+1}\mathbf{e}_{m+1}^T. \tag{2.12}$$

$$\mathbf{Z}_m^T\mathbf{A}\mathbf{Z}_m = \mathbf{H}_m \tag{2.13}$$

where $\mathbf{e}_{m+1}$ is the unit vector with 1 on its $(m + 1)$-th entry.

Algorithm 2.2 Arnoldi Algorithm

```
Input: $\mathbf{A} \in \mathbb{R}^{n\times n}$, a non-zero vector $\mathbf{x} \in \mathbb{R}^n$.
Output: $\mathbf{Z}_m = [\mathbf{z}_0, \ldots, \mathbf{z}_m] \in \mathbb{R}^{n\times(m+1)}$.
$\mathbf{z}_0 = \mathbf{x}/\|\mathbf{x}\|$;
for $i = 0 : (m-1)$ do
    $\mathbf{w} = \mathbf{A}\mathbf{z}_i$;
    for $j = 1 : i$ do
        $h_{j,i} = \langle \mathbf{z}_j, \mathbf{w} \rangle$;
        $\mathbf{w} = \mathbf{w} - h_{j,i}\mathbf{z}_j$;
    end for
    $h_{i+1,i} = \|\mathbf{w}\|$
    if $h_{i+1,i} = 0$ then
        break;
    end if
    $\mathbf{z}_{i+1} = \mathbf{w}/h_{i+1,i}$.
end for
```

It remains to show that $\mathbf{z}_0, \ldots \mathbf{z}_m$ indeed forms a basis of $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$.

Proposition 2.1.2. *$\{\mathbf{z}_0, \ldots \mathbf{z}_m\}$ is a basis of $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$.*

Proof. The Algorithm 2.2 is a form of modified Gram-Schmidt orthogonalization process whereby a new vector $\mathbf{A}\mathbf{z}_i$ is formed and then orthogonalized against all previous $\mathbf{z}_j$. Then the vectors $\mathbf{z}_0, \ldots, \mathbf{z}_m$ are orthonormal by construction. They span $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$ can be proved by induction. Clearly, it is true when $i = 1$. Suppose $\{\mathbf{z}_0, \ldots, \mathbf{z}_i\}$ span $\mathcal{K}_i(\mathbf{A}, \mathbf{x})$. We have

$$h_{i+1,i}\mathbf{z}_{i+1} = \mathbf{A}\mathbf{z}_i - \sum_{j=0}^{j} h_{j,i}\mathbf{z}_j.$$

Therefore $\mathbf{z}_{i+1} \in \text{span}\{\mathbf{z}_0, \ldots, \mathbf{z}_i, \mathbf{A}\mathbf{z}_i\} = \mathcal{K}_{j+1}(\mathbf{A}, \mathbf{x})$. □

At last, we notice that there will be loss of orthogonality in the process of Algorithm 2.2 if m is large. In that case, reorthogonalization is necessary to improve the numerical stability of the algorithm.

In the context of computing a few eigenvalues of the definite pencil $(\mathbf{A}, \mathbf{B})$, we may want to construct a $\mathbf{B}$-orthonormal basis for the Krylov subspace $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$. So we can apply the Arnoldi algorithm with the inner product defined by $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{B}} := \mathbf{x}^T\mathbf{B}\mathbf{y}$.

The Arnoldi process can be further simplified if the matrix $\mathbf{A}$ is symmetric in the sense of the inner product $\langle \cdot, \cdot \rangle$, i.e., $\langle \mathbf{x}, \mathbf{A}\mathbf{y} \rangle = \langle \mathbf{A}^T\mathbf{x}, \mathbf{y} \rangle$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. In that case, $\mathbf{H}_m$ obtained in (2.13) is tri-diagonal and Algorithm 2.2 can be simplified to the so-called Lanczos algorithm. We summarize it as follows:

Algorithm 2.3 Lanczos Algorithm

```
Input: $\mathbf{A} \in \mathbb{R}^{n \times n}$, a non-zero vector $\mathbf{x} \in \mathbb{R}^n$.
Output: $\mathbf{Z}_m = [\mathbf{z}_0, \ldots, \mathbf{z}_m] \in \mathbb{R}^{n \times (m+1)}$.
$\mathbf{z}_0 = x/\|x\|$, $\beta_0 = 0$, $\mathbf{z}_0 = 0$;
for i = 0:(m-1) do
    $\mathbf{w} = \mathbf{A}\mathbf{z}_i$;
    $\mathbf{w} = \mathbf{w} - \beta_i \mathbf{z}_{i-1}$;
    $\alpha_i = \langle \mathbf{w}, \mathbf{z}_i \rangle$;
    $\mathbf{w} = \mathbf{w} - \alpha_i \mathbf{z}_i$;
    $\beta_{i+1} = \|\mathbf{w}\|$;
    if $\beta_{i+1} = 0$ then
        break;
    end if
    $\mathbf{z}_{i+1} = \mathbf{w}/\beta_{i+1}$.
end for
```

For more discussions in Lanczos algorithm, see [67, 19, 71]

2.1.4 Deflation

Suppose the ℓ smallest eigenvalues $\lambda_1 \leq \ldots \leq \lambda_\ell$ of definite pencil $(\mathbf{A}, \mathbf{B})$ and corresponding eigenvectors $\mathbf{v}_1, \ldots \mathbf{v}_\ell$ have been computed, the rest of the eigenvalues and corresponding eigenvectors are $\lambda_{\ell+1}, \ldots, \lambda_n$ and $\mathbf{v}_{\ell+1}, \ldots, \mathbf{v}_n$. Assume $\mathbf{V}_\ell = [\mathbf{v}_1, \ldots, \mathbf{v}_\ell]$ and $\mathbf{\Lambda}_\ell = \mathrm{diag}(\lambda_1, \ldots, \lambda_\ell)$ with $\mathbf{V}_\ell^T \mathbf{B} \mathbf{V}_\ell = \mathbf{I}$ and $\mathbf{A}\mathbf{V}_\ell = \mathbf{B}\mathbf{V}_\ell \mathbf{\Lambda}_\ell$. A technique of computing the next eigenvalue $\lambda_{\ell+1}$ is called deflation. The two most commonly used deflation methods are Wielandt deflation and deflation by restriction [71, 67].

The idea of Wielandt deflation is to shift the spectrum by subtracting a rank ℓ matrix from the $\mathbf{A}$ such that the desired eigenvalue λ_ℓ is shifted to be the smallest eigenvalue of the new matrix pencil. It is shown in the following theorem.

Theorem 2.1.8. *Let* $\mathbf{\Sigma} = \mathrm{diag}\{\lambda_i - \alpha_i\}(1 \leq i \leq \ell)$ *with* α_i *any value chosen to be greater than* $\lambda_{\ell+1}$. *Then the eigenvalues of*

$$(\mathbf{A}_\ell, \mathbf{B}) := (\mathbf{A} - (\mathbf{B}\mathbf{V}_\ell)\mathbf{\Sigma}(\mathbf{B}\mathbf{V}_\ell)^T, \mathbf{B}) \tag{2.14}$$

are the union of $\alpha_1, \ldots, \alpha_\ell$ *and* $\lambda_{\ell+1}, \ldots \lambda_n$.

Proof. For $1 \leq i \leq \ell$ and let $\mathbf{e}_i \in \mathbb{R}^\ell$ be the unit vector with 1 on its i-th entry,

$$\begin{aligned}
\mathbf{A}_\ell \mathbf{v}_i &= (\mathbf{A} - (\mathbf{B}\mathbf{V}_\ell)\mathbf{\Sigma}(\mathbf{B}\mathbf{V}_\ell)^T)\mathbf{v}_i \\
&= \mathbf{A}\mathbf{v}_i - (\mathbf{B}\mathbf{V}_\ell)\mathbf{\Sigma}\mathbf{e}_i \\
&= \mathbf{A}\mathbf{v}_i - (\lambda_i - \alpha_i)\mathbf{B}\mathbf{v}_i \\
&= \alpha_i \mathbf{B}\mathbf{v}_i
\end{aligned}$$

For $\ell + 1 \leq i \leq n$, since $\mathbf{V}_\ell^T \mathbf{B}\mathbf{v}_i = 0$, then $\mathbf{A}_\ell \mathbf{v}_i = \mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{B}\mathbf{v}_i$. Therefore, the eigenvalues of $(\mathbf{A}_\ell, \mathbf{B})$ are $\alpha_1, \ldots, \alpha_\ell$ and $\lambda_{\ell+1}, \ldots, \lambda_n$. Therefore, the smallest eigenvalue is $\lambda_{\ell+1}$ given that α_i are greater than $\lambda_{\ell+1}$. □

The inverse-free preconditioned Krylov subspace method [32, 62] uses Wielandt deflation to compute additional eigenvalues.

The basic idea of deflation by restriction is to orthogonalize the approximate eigenvectors against the computed eigenvectors from time to time, which is equivalent to computing the eigenvalues of the restriction of the pencil $(\mathbf{A}, \mathbf{B})$ on an invariant subspace which is $\mathbf{B}$-orthogonal to the space spanned by the computed eigenvectors. For example, at some step of an iterative method, instead of computing the eigenvalues of $(\mathbf{A}, \mathbf{B})$ in the subspace $\mathcal{S}$, we consider the eigenvalue problem of $(\mathbf{A}, \mathbf{B})$ in the subspace $(\mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B})\mathcal{S}$.

The deflation by restriction is widely used with many iterative methods but they do not share a standard form. The implicitly Arnoldi algorithm [50, 51, 49] employs an elaborate deflation technique with the so-called locking and purging technique. The Jacobi-Davidson method [74, 75] incorporates it with partial Schur decomposition.

2.2 Inverse-free preconditioned Krylov subspace method with deflation by restriction

In this section, we introduce the inverse-free preconditioned Krylov subspace method first. Then we incorporate it with a deflation by restriction method. We prove the additional eigenvalues computed by the method have the same convergence behavior as the extreme ones. We consider real matrices in this section, but all can be generalized to complex matrices in a trivial way

2.2.1 Inverse-free preconditioned Krylov subspace method

The inverse-free precondioned Krylov subspace method of [32] is a Krylov subspace projection method that computes the smallest (or the largest) eigenvalues of (2.2). The method is based on an inner-outer iteration that does not require the inversion of $\mathbf{B}$ or any shifted matrix $\mathbf{A} - \lambda\mathbf{B}$.

Given a vector $\mathbf{x}$, the Rayleigh quotient $\rho(\mathbf{x}) = (\mathbf{x}^T\mathbf{A}\mathbf{x})/(\mathbf{x}^T\mathbf{B}\mathbf{x})$ is the best approximation to an eigenvalue in the sense that $\alpha = \rho(\mathbf{x})$ minimizes the 2-norm of the residual $\mathbf{A}x - \alpha\mathbf{B}\mathbf{x}$. Since $\mathbf{r} = 2(\mathbf{A}\mathbf{x} - \rho(\mathbf{x})\mathbf{B}\mathbf{x})/x^T\mathbf{B}\mathbf{x}$ is the gradient of $\rho(\mathbf{x})$ (Part 3 of Lemma 2.1.1), the well-known steepest descent method aims to minimize the Rayleigh quotient over span$\{\mathbf{x}, \mathbf{r}\}$. Noticing that it can be viewed as a Rayleigh-Ritz orthogonal projection method on the Krylov subspace $\mathcal{K}_1(\mathbf{A} - \rho(\mathbf{x})\mathbf{B}, \mathbf{x}) =$ span$\{\mathbf{x}, (\mathbf{A} - \rho(\mathbf{x})\mathbf{B})\mathbf{x}\}$, the inverse-free Krylov subspace method improves this by considering the Rayleigh-Ritz orthogonal projection on a larger Krylov subspace $\mathcal{K}_m(\mathbf{A} - \rho(\mathbf{x})\mathbf{B}, \mathbf{x}) = \text{span}\{\mathbf{x}, (\mathbf{A} - \rho(\mathbf{x})\mathbf{B})\mathbf{x}, \ldots, (\mathbf{A} - \rho(\mathbf{x})\mathbf{B})^m\mathbf{x}\}$. Namely, assume that $\mathbf{x}_k$ is the approximate eigenvector at step k in an iterative procedure of finding the smallest eigenvalue of the pair $(\mathbf{A}, \mathbf{B})$, [32] obtains a new approximation through the Rayleigh-Ritz orthogonal projection on

$$\mathcal{K}_m(\mathbf{A} - \rho_k\mathbf{B}, \mathbf{x}_k) = \text{span}\{\mathbf{x}_k, (\mathbf{A} - \rho_k\mathbf{B})\mathbf{x}_k, \ldots, (\mathbf{A} - \rho_k\mathbf{B})^m\mathbf{x}_k\}$$

where

$$\rho_k = \rho(\mathbf{x}_k) = \frac{\mathbf{x}_k^T\mathbf{A}\mathbf{x}_k}{\mathbf{x}_k^T\mathbf{B}\mathbf{x_k}} \tag{2.15}$$

and m is a parameter to be chosen. Suppose $\mathbf{Z}_m$ is a matrix whose columns are basis vectors of $\mathcal{K}_m(\mathbf{A} - \rho_k\mathbf{B}, \mathbf{x}_k)$. Let $\mathbf{A}_m = \mathbf{Z}_m^T(\mathbf{A} - \rho_k\mathbf{B})\mathbf{Z}_m$ and $\mathbf{B}_m = \mathbf{Z}_m^T\mathbf{B}\mathbf{Z}_m$. The smallest eigenvalue μ_1 of $(\mathbf{A}_m, \mathbf{B}_m)$ and a corresponding eigenvector $\mathbf{h}$ can be obtained by any state-of-the-art eigensolver. Then the new approximation $\mathbf{x}_{k+1}$ is

$$\mathbf{x}_{k+1} = \mathbf{Z}_m\mathbf{h} \tag{2.16}$$

and, correspondingly, the Rayleigh quotient

$$\rho_{k+1} = \rho_k + \mu_1 \tag{2.17}$$

is a new approximate eigenvalue. The choices of $\mathbf{Z}_m$ are not unique and it can be constructed by either the Lanczos method or the Arnoldi method with the $\mathbf{B}$-inner product; see [32] for a more detailed discussion. Throughout this paper, we will only consider the case when the columns of $\mathbf{Z}_m$ are $\mathbf{B}$-orthonormal, i.e. $\mathbf{Z}_m^T\mathbf{B}\mathbf{Z}_m = \mathbf{I}$. Then the basic procedure of inverse-free Krylov subspace method is given in Algorithm 2.4.

Algorithm 2.4 Inverse-free Krylov subspace method for $(\mathbf{A}, \mathbf{B})$

```
Input: $m \geq 1$ and an initial approximate eigenvector $\mathbf{x}_0$ with $\|\mathbf{x}_0\| = 1$;
Output: $(\rho_k, \mathbf{x}_k)$
$\rho_0 = \rho(\mathbf{x}_0)$;
for $k = 0, 1, 2, \ldots$ until convergence do
    Construct a $\mathbf{B}$-orthonormal basis $\mathbf{Z}_m = [\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_m]$ for $\mathcal{K}_m(\mathbf{A} - \rho_k\mathbf{B}, \mathbf{x}_k)$;
    Form $\mathbf{A}_m = \mathbf{Z}_m^T(\mathbf{A} - \rho_k\mathbf{B})\mathbf{Z}_m$;
    Find the smallest eigenpair $(\mu_1, \mathbf{h})$ of $\mathbf{A}_m$;
    $\rho_{k+1} = \rho_k + \mu_1$ and $\mathbf{x}_{k+1} = \mathbf{Z}_m\mathbf{h}$.
end for
```

The following theorem states that Algorithm 2.4 always converges to an eigenpair of $(\mathbf{A}, \mathbf{B})$.

Theorem 2.2.1. ([32, Proposition 3.1 and Theorems 3.2]) *Let λ_1 be the smallest eigenvalue of $(\mathbf{A}, \mathbf{B})$ and $(\rho_k, \mathbf{x}_k)$ be the eigenpair approximation of Algorithm 2.4 at step k. Then*

1. *$\lambda_1 \leq \rho_{k+1} \leq \rho_k$;*

2. *ρ_k converges to some eigenvalue $\hat{\lambda}$ of $(\mathbf{A}, \mathbf{B})$ and $\|(\mathbf{A} - \hat{\lambda}\mathbf{B})\mathbf{x}_k\| \to 0$.*

Theorem 2.2.1 shows that $\mathbf{x}_k$ in Algorithm 2.4 always converges in direction to an eigenvector of $(\mathbf{A}, \mathbf{B})$. Through a local analysis, we have that Algorithm 2.4 converges linearly under some conditions with a rate bounded below.

Theorem 2.2.2. ([32, Theorems 3.4]) *Let $\lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of $(\mathbf{A}, \mathbf{B})$. Let $(\rho_{k+1}, \mathbf{x}_{k+1})$ be the approximate eigenpair obtained by Algorithm 2.4 at step $k+1$ from $(\rho_k, \mathbf{x}_k)$. Let $\sigma_1 < \sigma_2 \leq \cdots \leq \sigma_n$ be the eigenvalues of $\mathbf{A} - \rho_k \mathbf{B}$ and u_1 be a unit eigenvector corresponding to σ_1. Assume $\lambda_1 < \rho_k < \lambda_2$. Then*

$$\rho_{k+1} - \lambda_1 \leq (\rho_k - \lambda_1)\epsilon_m^2 + 2(\rho_k - \lambda_1)^{3/2}\epsilon_m \left(\frac{\|B\|}{\sigma_2}\right)^{\frac{1}{2}} + \mathcal{O}\left((\rho_k - \lambda_1)^2\right) \qquad (2.18)$$

where

$$\epsilon_m = \min_{p \in \mathcal{P}_m, p(\sigma_1)=1} \max_{i \neq 1} |p(\sigma_i)|$$

and $\mathcal{P}_m$ denote the set of all polynomials of degree not greater than m.

Theorem 2.2.2 shows that ρ_k converges at least at the rate of ϵ_m^2 which is bounded in terms of σ_i as

$$\epsilon_m \leq 2\left(\frac{1-\sqrt{\phi}}{1+\sqrt{\phi}}\right)^m \quad \text{with } \phi = \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1}.$$

It illustrates an interesting fact that the speed of convergence of ρ_k depends on the distribution of eigenvalues of $\mathbf{A} - \rho_k \mathbf{B}$ rather than those of $(\mathbf{A}, \mathbf{B})$. It leads to some equivalent transformations of the problem, called preconditioning, that changes the spectrum of $\mathbf{A} - \rho_k \mathbf{B}$ to accelerate the convergence of Algorithm 2.4. In particular, suppose $\lambda_1 < \rho_k < \lambda_2$ and let $\mathbf{A} - \rho_k \mathbf{B} = \mathbf{L}_k \mathbf{D}_k \mathbf{L}_k^T$ be the LDL^T factorization with $\mathbf{D}_k = \text{diag}\{-1, 1, \ldots, 1\}$. Then the transformed pair $(\hat{\mathbf{A}}_k, \hat{\mathbf{B}}_k) \equiv (\mathbf{L}_k^{-1}\mathbf{A}\mathbf{L}_k^{-T}, \mathbf{L}_k^{-1}\mathbf{B}\mathbf{L}_k^{-T})$ will have the same eigenvalues as $(\mathbf{A}, \mathbf{B})$ and the convergence of Algorithm 2.4 will depend on the spectral gap of $\mathbf{L}_k^{-1}(\mathbf{A} - \rho_k \mathbf{B})\mathbf{L}_k^{-T}$ in which case $\epsilon_m = 0$. Then, by Theorem 2.2.2, the preconditioned Algorithm converges quadratically. However, this is an ideal situation since we assume a complete LDL^T factorization and $\mathbf{L}_k$ is computed for each iteration which is not practical. In practice, we use an approximate LDL^T factorization through an incomplete factorization for example. This usually leads to a small ϵ_m and hence accelerates convergence; see [32] for more discussions.

2.2.2 Deflation by restriction

Algorithm 2.4 computes the smallest eigenvalue of $(\mathbf{A}, \mathbf{B})$ only. When the smallest eigenvalue has been computed, we can use a deflation procedure to compute additional eigenvalues. While both the deflation by restriction method and the Wielandt deflation can be used in most other iterative methods, the Wielandt deflation is the only one that can be directly used for Algorithm 2.4. The process as presented in [62] is given in Theorem 2.1.8. Since $\lambda_{\ell+1}$ is the smallest eigenvalue of $(\mathbf{A}_\ell, \mathbf{B})$ in (2.14), Algorithm 2.4 will converge to $\lambda_{\ell+1}$ under the conditions of Theorem 2.2.2.

As discussed in the introduction, the Wielandt deflation changes the structure of the problem and this may be undesirable in certain applications such as the singular value computations (see Chapter 3 and [55]). In such problems, it is of interest to consider the deflation by restriction, namely, by projecting the subspaces involved to the $\mathbf{B}$-orthogonal complement of $\mathcal{V}_\ell := \operatorname{span}\{\mathbf{v}_1, \cdots, \mathbf{v}_\ell\}$. This can be done by simply using $(\mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B})\mathcal{K}_m(\mathbf{A} - \rho_k \mathbf{B}, \mathbf{x}_k)$, but this does not lead to a convergent algorithm. Also note that, $\mathcal{K}_m(\mathbf{A} - \rho_k \mathbf{B}, (\mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B})\mathbf{x}_k)$ may not be in the $\mathbf{B}$-orthogonal complement of $\mathcal{V}_\ell$. A more appropriate approach is to apply the projection on the matrix or on every step of the basis construction; namely we use $\mathcal{K}_m((\mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B})(\mathbf{A} - \rho_k \mathbf{B}), (\mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B})\mathbf{x}_k)$; This also changes subspaces and the existing convergence theory does not apply. However, it has been observed numerically in Section 3.4 that such a deflation scheme appears to work in practice.

In this section, we formulate a deflation by restriction method for the inverse-free Krylov subspace method (Algorithm 2.4) and present a convergence theory that generalizes the convergence results of Section 2. We first state the deflation by restriction method in the following algorithm.

The difference of this algorithm from the standard one (Algorithm 2.4) is the use of the projected Krylov subspace $\mathcal{K}_m(\mathbf{P}_\mathbf{V}(\mathbf{A} - \rho_k \mathbf{B}), \mathbf{x}_k)$ where $\mathbf{P}_\mathbf{V} = \mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B}$. We can easily show by induction that $\mathbf{P}_\mathbf{V} \mathbf{x}_k = \mathbf{x}_k$ for all k. Then

$$\mathcal{K}_m(\mathbf{P}_\mathbf{V}(\mathbf{A} - \rho_k \mathbf{B}), \mathbf{x}_k) = \mathcal{K}_m(\mathbf{P}_\mathbf{V}(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V}, \mathbf{x}_k).$$

However, since the columns of $\mathbf{V}_\ell$ are generally not eigenvectors of $\mathbf{A} - \rho_k \mathbf{B}$ (when

Algorithm 2.5 Inverse-free Krylov subspace method with deflation by restriction

```
Input: $\mathbf{V}_\ell = [\mathbf{v}_1, \cdots, \mathbf{v}_\ell]$ satisfying $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{B}\mathbf{v}_i$ (for $1 \le i \le l$) and $\mathbf{V}_\ell^T \mathbf{B}\mathbf{V}_\ell = \mathbf{I}$;
    $m$ and $\mathbf{x}_0$ with $\|\mathbf{x}_0\| = 1$ and $\mathbf{V}_\ell^T \mathbf{B}\mathbf{x}_0 = \mathbf{0}$;
$\rho_0 = \rho(\mathbf{x}_0)$;
for $k = 0, 1, 2, \ldots$ until convergence do
    Construct a basis $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m\}$ for $\mathcal{K}_m((\mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B})(\mathbf{A} - \rho_k \mathbf{B}), \mathbf{x}_k)$;
    $\mathbf{A}_m = \mathbf{Z}_m^T (\mathbf{A} - \rho_k \mathbf{B})\mathbf{Z}_m$ and $\mathbf{B}_m = \mathbf{Z}_m^T \mathbf{B}\mathbf{Z}_m$ where $\mathbf{Z}_m = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m]$;
    Find the smallest eigenvalue $\mu_1$ and a unit eigenvector $\mathbf{v}$ for $(\mathbf{A}_m, \mathbf{B}_m)$;
    $\rho_{k+1} = \rho_k + \mu_1$ and $\mathbf{x}_{k+1} = \mathbf{Z}_m \mathbf{v}$.
end for
```

$\mathbf{B} \neq \mathbf{I}$), $\mathbf{P}_\mathbf{V}(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V}$ does not lead to a deflated operator (i.e. a spectral restriction of $\mathbf{A} - \rho_k \mathbf{B}$). Indeed, with $\mathbf{P}_\mathbf{V}$ a $\mathbf{B}$-orthogonal projection, $\mathbf{P}_\mathbf{V}(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V}$ is not even symmetric. However, the following lemma expresses the Krylov subspace as one generated by a symmetric matrix, which is key in our analysis of Algorithm 2.5.

Lemma 2.2.1. *Let* $\mathbf{V}_\ell = [\mathbf{v}_1, \cdots, \mathbf{v}_\ell]$ *be such that* $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{B}\mathbf{v}_i$ *(for* $1 \le i \le l$*) and* $\mathbf{V}_\ell^T \mathbf{B}\mathbf{V}_\ell = \mathbf{I}$. *Let* $\mathbf{P}_\mathbf{V} = \mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B}$. *Then we have*

$$(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V} = \mathbf{P}_\mathbf{V}^T (\mathbf{A} - \rho_k \mathbf{B}) \tag{2.19}$$

and for any $\mathbf{x}_k$ *with* $\mathbf{P}_\mathbf{V}\mathbf{x}_k = \mathbf{x}_k$,

$$\mathcal{K}_m(\mathbf{P}_\mathbf{V}(\mathbf{A} - \rho_k \mathbf{B}), \mathbf{x}_k) = \mathbf{P}_\mathbf{V} \mathcal{K}_m(\mathbf{P}_\mathbf{V}^T (\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V}, \mathbf{x}_k). \tag{2.20}$$

Proof. First we have $\mathbf{A}\mathbf{V}_\ell = \mathbf{B}\mathbf{V}_\ell \mathbf{\Lambda}$, where $\mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, \cdots, \lambda_\ell\}$. Then $\mathbf{V}_\ell^T \mathbf{A} = \mathbf{\Lambda}\mathbf{V}_\ell^T \mathbf{B}$. It follows that

$$\begin{aligned}
\mathbf{P}_\mathbf{V}^T (\mathbf{A} - \rho_k \mathbf{B}) &= (\mathbf{A} - \rho_k \mathbf{B}) - (\mathbf{B}\mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{A} - \rho_k \mathbf{B}\mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B}) \\
&= (\mathbf{A} - \rho_k \mathbf{B}) - (\mathbf{B}\mathbf{V}_\ell \mathbf{\Lambda}\mathbf{V}_\ell^T \mathbf{B} - \rho_k \mathbf{B}\mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B}) \\
&= (\mathbf{A} - \rho_k \mathbf{B}) - (\mathbf{A}\mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B} - \rho_k \mathbf{B}\mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B}) \\
&= (\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V},
\end{aligned}$$

which proves (2.19). From this and $\mathbf{P}_\mathbf{V}^2 = \mathbf{P}_\mathbf{V}$, we have

$$\mathbf{P}_\mathbf{V}^T (\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V} = (\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V}\mathbf{P}_\mathbf{V} = (\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V}.$$

Thus, it follows from $\mathbf{P_V x}_k = \mathbf{x}_k$ that for all $i = 1, \cdots, m-1$,

$$\begin{aligned}(\mathbf{P_V}(\mathbf{A} - \rho_k \mathbf{B}))^i \mathbf{x}_k &= (\mathbf{P_V}(\mathbf{A} - \rho_k \mathbf{B}))^i \mathbf{P_V x}_k \\ &= \mathbf{P_V}((\mathbf{A} - \rho_k \mathbf{B})\mathbf{P_V})^i \mathbf{x}_k \\ &= \mathbf{P_V}(\mathbf{P_V^T}(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P_V})^i \mathbf{x}_k.\end{aligned}$$

Hence $\mathcal{K}_m(\mathbf{P_V}(\mathbf{A} - \rho_k \mathbf{B}), \mathbf{x}_k) = \mathbf{P_V}\mathcal{K}_m(\mathbf{P_V^T}(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P_V}, \mathbf{x}_k)$. □

With the above characterization of the projection subspace used in Algorithm 2.5, the convergence properties described in Section 2 can be generalized following similar lines of proofs in [32, Theorem 3.2] with careful analysis of some subtle effects of the projection that are highly nontrivial. We first present a generalization of the global convergence result (Theorem 2.2.1).

Theorem 2.2.3. *Let* $\mathbf{V}_\ell = [\mathbf{v}_1, \cdots, \mathbf{v}_\ell]$ *be such that* $\mathbf{A v}_i = \lambda_i \mathbf{B v}_i$ *(for* $1 \le i \le l$*) and* $\mathbf{V}_\ell^T \mathbf{B V}_\ell = \mathbf{I}$. *Let* $\lambda_{\ell+1} \le \lambda_{\ell+2} \le \cdots \le \lambda_n$ *together with* $\lambda_1, \cdots, \lambda_\ell$ *be the eigenvalues of* $(\mathbf{A}, \mathbf{B})$. *Let* $(\rho_k, \mathbf{x}_k)$ *be the eigenpair approximation obtained at step* k *of Algorithm 2.5 with* $\mathbf{V}_\ell$. *Then*

$$\lambda_{\ell+1} \le \rho_{k+1} \le \rho_k.$$

Furthermore, ρ_k *converges to some eigenvalue* $\hat{\lambda} \in \{\lambda_{\ell+1}, \cdots, \lambda_n\}$ *of* $(\mathbf{A}, \mathbf{B})$ *and* $\|(\mathbf{A} - \hat{\lambda}\mathbf{B})\mathbf{x}_k\| \to 0$ *(i.e.,* $\mathbf{x}_k$ *converges in direction to a corresponding eigenvector).*

Proof. From Algorithm 2.5, we have

$$\rho_{k+1} = \rho_k + \min_{\mathbf{w} \in \mathcal{W}, \mathbf{w} \neq 0} \frac{\mathbf{w}^T(\mathbf{A} - \rho_k \mathbf{B})\mathbf{w}}{\mathbf{w}^T \mathbf{B w}} = \min_{w \in \mathcal{W}} \frac{\mathbf{w}^T \mathbf{A w}}{\mathbf{w}^T \mathbf{B w}}$$

where $\mathcal{W} = \mathcal{K}_m(\mathbf{P_V}(\mathbf{A} - \rho_\mathbf{k} \mathbf{B}), \mathbf{x_k})$ and $\mathbf{P_V} = \mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B}$. Since $\mathbf{x}_k \in \mathcal{W}$, we have $\rho_{k+1} \le \rho_k$. On the other hand, it follows from Lemma 2.2.1 that $\mathcal{W} = \mathbf{P_V}\mathcal{K}_m(\mathbf{P_V^T}(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P_V}, \mathbf{x}_k) \subset \mathcal{R}(\mathbf{P_V})$. Then

$$\rho_{k+1} \ge \min_{\mathbf{V}_\ell^T \mathbf{B w} = 0, w \neq 0} \frac{\mathbf{w}^T \mathbf{A w}}{\mathbf{w}^T \mathbf{B w}} = \lambda_{\ell+1}.$$

It follows that ρ_k is convergent. Since $\{\mathbf{x}_k\}$ is bounded, there is a convergent subsequence $\{x_{n_k}\}$. Let

$$\lim \rho_k = \hat{\lambda}, \text{ and } \lim \mathbf{x}_{n_k} = \hat{\mathbf{x}}.$$

Write $\hat{\mathbf{r}} = (\mathbf{A} - \hat{\lambda}\mathbf{B})\hat{\mathbf{x}}$. Then it follows from $\mathbf{x}_k^T(\mathbf{A} - \rho_k\mathbf{B})\mathbf{x}_k = 0$ that

$$\hat{\mathbf{x}}^T\hat{\mathbf{r}} = \hat{\mathbf{x}}^T(\mathbf{A} - \hat{\lambda}\mathbf{B})\hat{\mathbf{x}} = 0.$$

Suppose now $\hat{\mathbf{r}} \neq 0$. Using Lemma 2.2.1 and the fact that $\mathbf{P_V}\hat{\mathbf{x}} = \hat{\mathbf{x}}$ which follows from $\mathbf{P_V}\mathbf{x}_k = \mathbf{x}_k$, we obtain

$$\mathbf{P}_\mathbf{V}^T\hat{\mathbf{r}} = \mathbf{P}_\mathbf{V}^T(\mathbf{A} - \hat{\lambda}\mathbf{B})\hat{\mathbf{x}} = (\mathbf{A} - \hat{\lambda}\mathbf{B})\mathbf{P_V}\hat{\mathbf{x}} = (\mathbf{A} - \hat{\lambda}\mathbf{B})\hat{\mathbf{x}} = \hat{\mathbf{r}}. \tag{2.21}$$

We now show that $\hat{\mathbf{x}}$ and $\mathbf{P_V}\hat{\mathbf{r}}$ are linearly independent. If they are linearly dependent, we have $\mathbf{P_V}\hat{\mathbf{r}} = \gamma\hat{\mathbf{x}}$ for some scaler γ. Then by (2.21), $\hat{\mathbf{r}}^T\mathbf{P_V}\mathbf{P}_\mathbf{V}^T\hat{\mathbf{r}} = \hat{\mathbf{r}}^T\mathbf{P_V}\hat{\mathbf{r}} = \gamma\hat{\mathbf{r}}^T\hat{\mathbf{x}} = 0$. Thus $\mathbf{P}_\mathbf{V}^T\hat{\mathbf{r}} = \mathbf{0}$ or by (2.21) again, $\hat{\mathbf{r}} = \mathbf{0}$, which is a contradiction. Therefore, $\hat{\mathbf{x}}$ and $\mathbf{P_V}\hat{\mathbf{r}}$ are linearly independent. We next consider the projection of $(\mathbf{A}, \mathbf{B})$ onto $\text{span}\{\hat{\mathbf{x}}, \mathbf{P_V}\hat{\mathbf{r}}\}$ by defining

$$\hat{\mathbf{A}} = [\hat{\mathbf{x}}, \mathbf{P_V}\hat{\mathbf{r}}]^T\mathbf{A}[\hat{\mathbf{x}}, \mathbf{P_V}\hat{\mathbf{r}}] \text{ and } \hat{\mathbf{B}} = [\hat{\mathbf{x}}, \mathbf{P_V}\hat{\mathbf{r}}]^T\mathbf{B}[\hat{\mathbf{x}}, \mathbf{P_V}\hat{\mathbf{r}}].$$

Clearly, $\hat{\mathbf{B}} > 0$. Furthermore,

$$\hat{\mathbf{A}} - \hat{\lambda}\hat{\mathbf{B}} = \begin{pmatrix} \mathbf{O} & \hat{\mathbf{r}}^T\mathbf{P_V}\hat{\mathbf{r}} \\ \hat{\mathbf{r}}^T\mathbf{P}_\mathbf{V}^T\hat{\mathbf{r}} & \hat{\mathbf{r}}^T\mathbf{P}_\mathbf{V}^T(\mathbf{A} - \hat{\lambda}\mathbf{B})\mathbf{P_V}\hat{\mathbf{r}} \end{pmatrix}$$

is indefinite because, by (2.21), $\hat{\mathbf{r}}^T\mathbf{P_V}\hat{\mathbf{r}} = (\mathbf{P}_\mathbf{V}^T\hat{\mathbf{r}})^T\hat{\mathbf{r}} = \hat{\mathbf{r}}^T\hat{\mathbf{r}} \neq 0$. Thus the smallest eigenvalue of $(\hat{\mathbf{A}}, \hat{\mathbf{B}})$, denoted by $\tilde{\lambda}$, is less than $\hat{\lambda}$, i.e.

$$\tilde{\lambda} < \hat{\lambda}. \tag{2.22}$$

Furthermore, let $\mathbf{r}_k = (\mathbf{A} - \rho_k\mathbf{B})\mathbf{x}_k$,

$$\hat{\mathbf{A}}_k = [\mathbf{x}_k, \mathbf{P_V}\mathbf{r}_k]^T\mathbf{A}[\mathbf{x}_k, \mathbf{P_V}\mathbf{r}_k] \text{ and } \hat{\mathbf{B}}_k = [\mathbf{x}_k, \mathbf{P_V}\mathbf{r}_k]^T\mathbf{B}[\mathbf{x}_k, \mathbf{P_V}\mathbf{r}_k].$$

Let $\tilde{\lambda}_{k+1}$ be the smallest eigenvalue of $(\hat{\mathbf{A}}_k, \hat{\mathbf{B}}_k)$. Clearly, as $n_k \to \infty$, $\hat{\mathbf{A}}_{n_k} \to \hat{\mathbf{A}}$ and $\hat{\mathbf{B}}_{n_k} \to \hat{\mathbf{B}}$. Hence by the continuity property of the eigenvalue, we have

$$\tilde{\lambda}_{n_k+1} \to \tilde{\lambda}.$$

On the other hand, ρ_{k+1} is the smallest eigenvalue of the projection of $(\mathbf{A}, \mathbf{B})$ on $\mathcal{K}_m = \text{span}\{\mathbf{x}_k, \mathbf{P_V}(\mathbf{A} - \rho_k\mathbf{B})\mathbf{x}_k, \cdots, (\mathbf{P_V}(\mathbf{A} - \rho_k\mathbf{B}))^m\mathbf{x}_k\}$, which implies

$$\rho_{k+1} \leq \tilde{\lambda}_{k+1}.$$

Finally, combining the above together, we have obtained

$$\tilde{\lambda} = \lim \tilde{\lambda}_{n_k+1} \geq \lim \rho_{n_k+1} = \hat{\lambda}$$

which is a contradiction to (2.22). Therefore, $\hat{\mathbf{r}} = (\mathbf{A} - \hat{\lambda}\mathbf{B})\hat{\mathbf{x}} = 0$, i.e. $\hat{\lambda}$ is an eigenvalue and $\|(\mathbf{A} - \hat{\lambda}\mathbf{B})\mathbf{x}_{n_k}\| \to 0$.

Now, to show $\|(\mathbf{A} - \hat{\lambda}\mathbf{B})\mathbf{x}_k\| \to 0$, suppose there is a subsequence m_k such that $\|(\mathbf{A} - \hat{\lambda}\mathbf{B})\mathbf{x}_{m_k}\| \geq \alpha > 0$. From the subsequence m_k, there is a subsequence n_k for which $\mathbf{x}_{n_k}$ is convergent. Hence by virtue of the above proof, $\|(\mathbf{A} - \hat{\lambda}\mathbf{B})\mathbf{x}_{n_k}\| \to 0$, which is a contradiction. Therefore $\|(\mathbf{A} - \hat{\lambda}\mathbf{B})\mathbf{x}_k\| \to 0$, i.e. $\mathbf{x}_k$ approaches in direction an eigenvector corresponding to $\hat{\lambda}$. Since $\mathbf{x}_k$ is B-orthogonal to $\{\mathbf{v}_1, \cdots, \mathbf{v}_\ell\}$, we have $\hat{\lambda} \in \{\lambda_{\ell+1}, \cdots, \lambda_n\}$. This completes the proof. □

Next we present a lemma and then our main result concerning local linear convergence of ρ_k that generalizes Theorem 2.2.2.

Lemma 2.2.2. *Let $\mathbf{V}_\ell = [\mathbf{v}_1, \cdots, \mathbf{v}_\ell]$ be such that $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{B}\mathbf{v}_i$ (for $1 \leq i \leq \ell$) and $\mathbf{V}_\ell^T \mathbf{B} \mathbf{V}_\ell = I$ and let $\mathbf{P}_\mathbf{V} = \mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B}$ and $\mathbf{V}_\ell = \operatorname{span}\{\mathbf{v}_1, \cdots, \mathbf{v}_\ell\}$. Let $\lambda_{\ell+1} < \lambda_{\ell+2} \leq \cdots \leq \lambda_n$ together with $\lambda_1, \cdots, \lambda_\ell$ be the eigenvalues of $(\mathbf{A}, \mathbf{B})$. Let $(\rho_k, \mathbf{x}_k)$ be the eigenpair approximation obtained at step k of Algorithm 2.5 with $\mathbf{V}_\ell$ and assume that $\lambda_{\ell+1} \leq \rho_k < \lambda_{\ell+2}$. Let $\mathbf{P}_\mathbf{V}^T(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V} = \mathbf{W}\mathbf{S}\mathbf{W}^T$ be the eigenvalue decomposition of $\mathbf{P}_\mathbf{V}^T(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V}$ where $\mathbf{S} = diag\{0, 0, \cdots, 0, s_{\ell+1}, \cdots, s_n\}$ with $s_{\ell+1} \leq s_{\ell+2} \leq \cdots \leq s_n$, and $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_\ell, \mathbf{w}_{\ell+1}, \cdots, \mathbf{w}_n]$ with $\mathbf{w}_i \in \mathcal{V}_\ell$ (for $i = 1, \cdots, \ell$), and $\mathbf{w}_i \perp \mathbf{V}_\ell$ (for $i = \ell + 1, \cdots, n$). Then we have $s_{\ell+1} \leq 0 < s_{\ell+2}$, $\mathbf{P}_\mathbf{V}\mathbf{w}_{\ell+1} \neq 0$ and*

$$\frac{|s_{\ell+1}|}{\mathbf{w}_{\ell+1}^T \mathbf{P}_\mathbf{V}^T \mathbf{B} \mathbf{P}_\mathbf{V} \mathbf{w}_{\ell+1}} \leq \rho_k - \lambda_{\ell+1}. \tag{2.23}$$

Furthermore, $\rho_k \to \lambda_{\ell+1}$ and

$$\frac{s_{\ell+1}}{\mathbf{w}_{\ell+1}^T \mathbf{P}_\mathbf{V}^T \mathbf{B} \mathbf{P}_\mathbf{V} \mathbf{w}_{\ell+1}} = (\lambda_{\ell+1} - \rho_k) + \mathcal{O}((\lambda_{\ell+1} - \rho_k)^2). \tag{2.24}$$

Proof. First, by Theorem 2.2.3 and the assumption $\lambda_{\ell+1} \leq \rho_k < \lambda_{\ell+2}$, we have the convergence of ρ_k to $\lambda_{\ell+1}$.

Let $\tilde{\mathbf{V}} = [\mathbf{v}_{\ell+1}, \mathbf{v}_{\ell+2}, \cdots, \mathbf{v}_n]$ be such that $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{B}\mathbf{v}_i$ (for $\ell+1 \leq i \leq n$) and $\tilde{\mathbf{V}}^T\mathbf{B}\tilde{\mathbf{V}} = \mathbf{I}$. Let $\mathbf{V} = [\mathbf{V}_\ell, \tilde{\mathbf{V}}]$ and $\mathbf{P}_\mathbf{V}\mathbf{V} = [\mathbf{P}_\mathbf{V}\mathbf{V}_\ell, \mathbf{P}_\mathbf{V}\tilde{\mathbf{V}}] = [\mathbf{O}, \tilde{\mathbf{V}}]$ and hence

$$\mathbf{V}^T\mathbf{P}_\mathbf{V}^T(\mathbf{A} - \rho_k\mathbf{B})\mathbf{P}_\mathbf{V}\mathbf{V} = \begin{pmatrix} \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \tilde{\mathbf{V}}^T(\mathbf{A} - \rho_k\mathbf{B})\tilde{\mathbf{V}} \end{pmatrix} = \begin{pmatrix} \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \tilde{\mathbf{\Lambda}} - \rho_k I \end{pmatrix}$$

where $\tilde{\mathbf{V}}^T(\mathbf{A} - \rho_k\mathbf{B})\tilde{\mathbf{V}} = \tilde{\mathbf{V}}^T\mathbf{B}\tilde{\mathbf{V}}(\tilde{\mathbf{\Lambda}} - \rho_k I) = \tilde{\mathbf{\Lambda}} - \rho_k I$ and $\tilde{\mathbf{\Lambda}} = \operatorname{diag}\{\lambda_{\ell+1}, \cdots, \lambda_n\}$. By Sylvester's law of inertia and $\lambda_{\ell+1} - \rho_k \leq 0 < \lambda_{\ell+2} - \rho_k$, $\mathbf{P}_\mathbf{V}^T(\mathbf{A} - \rho_k\mathbf{B})\mathbf{P}_\mathbf{V}$ has exactly $n - \ell - 1$ negative, ℓ zero, and 1 nonpositive eigenvalues, i.e., $s_{\ell+1} \leq 0 < s_{\ell+2}$.

Let $\tilde{\mathbf{w}}_{\ell+1} = \mathbf{P}_\mathbf{V}\mathbf{w}_{\ell+1}$ and suppose $\tilde{\mathbf{w}}_{\ell+1} = 0$. Then $\mathbf{w}_{\ell+1} = \mathbf{V}_\ell\mathbf{V}_\ell^T\mathbf{B}\mathbf{w}_{\ell+1} \in \mathbf{V}_\ell$. This implies $\mathbf{w}_{\ell+1} = 0$ as $\mathbf{w}_{\ell+1} \perp \mathbf{V}_\ell$. This is a contradiction. Therefore, $\tilde{\mathbf{w}}_{\ell+1} \neq 0$.

Furthermore, $\tilde{\mathbf{w}}_{\ell+1} \perp_\mathbf{B} \mathbf{V}_\ell$, i.e. $\mathbf{V}_\ell^T\mathbf{B}\tilde{\mathbf{w}}_{\ell+1} = 0$. Then

$$\begin{aligned}
\lambda_{\ell+1} &= \min_{\mathbf{V}_\ell^T\mathbf{B}\mathbf{w}=\mathbf{0}, \mathbf{w}\neq\mathbf{0}} \frac{\mathbf{w}^T\mathbf{A}\mathbf{w}}{\mathbf{w}^T\mathbf{B}\mathbf{w}} \\
&\leq \frac{\tilde{\mathbf{w}}_{\ell+1}^T\mathbf{A}\tilde{\mathbf{w}}_{\ell+1}}{\tilde{\mathbf{w}}_{\ell+1}^T\mathbf{B}\tilde{\mathbf{w}}_{\ell+1}} \\
&= \rho_k + \frac{\tilde{\mathbf{w}}_{\ell+1}^T(\mathbf{A} - \rho_k\mathbf{B})\tilde{\mathbf{w}}_{\ell+1}}{\tilde{\mathbf{w}}_{\ell+1}^T\mathbf{B}\tilde{\mathbf{w}}_{\ell+1}} \\
&= \rho_k + \frac{\mathbf{w}_{\ell+1}^T\mathbf{P}_\mathbf{V}^T(\mathbf{A} - \rho_k\mathbf{B})\mathbf{P}_\mathbf{V}\mathbf{w}_{\ell+1}}{\tilde{\mathbf{w}}_{\ell+1}^T\mathbf{B}\tilde{\mathbf{w}}_{\ell+1}} \\
&= \rho_k + \frac{s_{\ell+1}}{\mathbf{w}_{l+1}^T\mathbf{P}_\mathbf{V}^T\mathbf{B}\mathbf{P}_\mathbf{V}\mathbf{w}_{l+1}}.
\end{aligned}$$

where we have used $\mathbf{P}_\mathbf{V}^T(\mathbf{A} - \rho_k\mathbf{B})\mathbf{P}_\mathbf{V}\mathbf{w}_{\ell+1} = s_{\ell+1}\mathbf{w}_{\ell+1}$ in the last equation. This proves (2.23).

Finally, to prove the asymptotic expansion, let $s(t)$ be the smallest eigenvalue of $\mathbf{P}_\mathbf{V}^T(\mathbf{A} - t\mathbf{B})\mathbf{P}_\mathbf{V}$. Then $s(\rho_k) = s_{\ell+1}$. It is easy to check that $s(\lambda_{\ell+1}) = 0$. Using the analytic perturbation theory, we obtain $s'(\rho_k) = -\tilde{\mathbf{w}}_{\ell+1}^T\mathbf{B}\tilde{\mathbf{w}}_{\ell+1}$ and hence

$$\begin{aligned}
s(t) &= s(\rho_k) + s'(\rho_k)(t - \rho_k) + \mathcal{O}((t - \rho_k)^2) \\
&= s_{\ell+1} - \tilde{\mathbf{w}}_{\ell+1}^T\mathbf{B}\tilde{\mathbf{w}}_{\ell+1}(t - \rho_k) + \mathcal{O}((t - \rho_k)^2)
\end{aligned}$$

Choosing $t = \lambda_{\ell+1}$, we have

$$0 = s(\lambda_{\ell+1}) = s_{\ell+1} - \tilde{\mathbf{w}}_{\ell+1}^T\mathbf{B}\tilde{\mathbf{w}}_{\ell+1}(\lambda_{\ell+1} - \rho_k) + \mathcal{O}((\lambda_{\ell+1} - \rho_k)^2)$$

from which the expansion follows. □

Theorem 2.2.4. *Let $\mathbf{V}_\ell = [\mathbf{v}_1, \cdots, \mathbf{v}_\ell]$ be such that $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{B}\mathbf{v}_i$ (for $1 \le i \le l$) and $\mathbf{V}_\ell^T \mathbf{B} \mathbf{V}_\ell = I$ and write $\mathbf{P}_\mathbf{V} = \mathbf{I} - \mathbf{V}_\ell \mathbf{V}_\ell^T \mathbf{B}$ and $\mathbf{V}_\ell = \operatorname{span}\{\mathbf{v}_1, \cdots, \mathbf{v}_\ell\}$. Let $\lambda_{\ell+1} < \lambda_{\ell+2} \le \cdots \le \lambda_n$ together with $\lambda_1, \cdots, \lambda_\ell$ be the eigenvalues of $(\mathbf{A}, \mathbf{B})$. Let $(\rho_k, \mathbf{x}_k)$ be the eigenpair approximation obtained at step k of Algorithm 2.5 with $\mathbf{V}_\ell$ and assume that $\lambda_{\ell+1} \le \rho_k < \lambda_{\ell+2}$. Let $\mathbf{P}_\mathbf{V}^T(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V} = \mathbf{W}\mathbf{S}\mathbf{W}^T$ be the eigenvalue decomposition of $\mathbf{P}_\mathbf{V}^T(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V}$ where $S = diag\{0, 0, \cdots, 0, s_{\ell+1}, \cdots, s_n\}$ with $s_{\ell+1} \le s_{\ell+2} \le \cdots \le s_n$, and $W = [\mathbf{w}_1, \cdots, \mathbf{w}_\ell, \mathbf{w}_{\ell+1}, \cdots, \mathbf{w}_n]$ with $\mathbf{w}_i \in \mathcal{V}_\ell$ (for $i = 1, \cdots, \ell$), and $\mathbf{w}_i \perp \mathbf{V}_\ell$ (for $i = \ell+1, \cdots, n$). Then ρ_k converges to $\lambda_{\ell+1}$ and*

$$\rho_{k+1} - \lambda_{\ell+1} \le (\rho_k - \lambda_{\ell+1})\epsilon_m^2 + 2(\rho_k - \lambda_{\ell+1})^{3/2}\epsilon_m \left(\frac{\|\mathbf{B}\|}{s_{\ell+2}}\right)^{1/2} + \delta_k, \tag{2.25}$$

where

$$0 \le \delta_k := \rho_k - \lambda_{\ell+1} + \frac{s_{\ell+1}}{\mathbf{w}_{\ell+1}^T \mathbf{P}_\mathbf{V}^T \mathbf{B} \mathbf{P}_\mathbf{V} \mathbf{w}_{\ell+1}} = \mathcal{O}((\rho_k - \lambda_{\ell+1})^2)$$

and

$$\epsilon_m = \min_{\substack{p \in \mathcal{P}_m, \\ p(s_{\ell+1}) = 1}} \max_{i \ge 1} |p(s_{\ell+i})|$$

with $\mathcal{P}_m$ denoting the set of all polynomials of degree not greater than m.

Proof. First, it follows from Lemma 2.2.2 that ρ_k converges to $\lambda_{\ell+1}$, $s_{\ell+1} \le 0 < s_{\ell+2}$ and $\mathbf{P}_\mathbf{V}\mathbf{w}_{\ell+1} \ne 0$.

Let $\tilde{\mathbf{H}}_k := \mathbf{P}_\mathbf{V}^T(\mathbf{A} - \rho_k \mathbf{B})\mathbf{P}_\mathbf{V}$. From Lemma 2.2.1, $\mathcal{K}_m(\mathbf{P}_\mathbf{V}(\mathbf{A} - \rho_k \mathbf{B}), \mathbf{x}_k) = \mathbf{P}_\mathbf{V}\mathcal{K}_m(\tilde{\mathbf{H}}_k, \mathbf{x}_k) = \{\mathbf{P}_\mathbf{V} p(\tilde{\mathbf{H}}_k)\mathbf{x}_k, p \in \mathcal{P}_m\}$. At step k of the algorithm, we have

$$\begin{aligned}
\rho_{k+1} &= \min_{\substack{\mathbf{u} \in \mathcal{K}_m(\mathbf{P}_\mathbf{V}(\mathbf{A} - \rho_k \mathbf{B}), \mathbf{x}_k), \\ \mathbf{u} \ne 0}} \frac{\mathbf{u}^T \mathbf{A} \mathbf{u}}{\mathbf{u}^T \mathbf{B} \mathbf{u}} \\
&= \rho_k + \min_{\substack{\mathbf{u} \in \mathcal{K}_m(\mathbf{P}_\mathbf{V}(\mathbf{A} - \rho_k \mathbf{B}), \mathbf{x}_k), \\ \mathbf{u} \ne 0}} \frac{\mathbf{u}^T (\mathbf{A} - \rho_k \mathbf{B}) \mathbf{u}}{\mathbf{u}^T \mathbf{B} \mathbf{u}} \\
&= \rho_k + \min_{\substack{p \in \mathcal{P}_m, \\ \mathbf{P}_\mathbf{V} p(\tilde{\mathbf{H}}_k)\mathbf{x}_k \ne 0}} \frac{\mathbf{x}_k^T p(\tilde{\mathbf{H}}_k) \mathbf{P}_\mathbf{V}^T (\mathbf{A} - \rho_k \mathbf{B}) \mathbf{P}_\mathbf{V} p(\tilde{\mathbf{H}}_k) \mathbf{x}_k}{\mathbf{x}_k^T p(\tilde{\mathbf{H}}_k) \mathbf{P}_\mathbf{V}^T \mathbf{B} \mathbf{P}_\mathbf{V} p(\tilde{\mathbf{H}}_k) \mathbf{x}_k} \\
&= \rho_k + \min_{\substack{p \in \mathcal{P}_m, \\ \mathbf{P}_\mathbf{V} p(\tilde{\mathbf{H}}_k)\mathbf{x}_k \ne 0}} \frac{\mathbf{x}_k^T p(\tilde{\mathbf{H}}_k) \tilde{\mathbf{H}}_k p(\tilde{\mathbf{H}}_k) \mathbf{x}_k}{\mathbf{x}_k^T p(\tilde{\mathbf{H}}_k) (\mathbf{P}_\mathbf{V}^T \mathbf{B} \mathbf{P}_\mathbf{V}) p(\tilde{\mathbf{H}}_k) \mathbf{x}_k}
\end{aligned}$$

Let q be the minimizing polynomial in ϵ_m with $q(s_{\ell+1}) = 1$ and $\max_{i \ge l+2} |q(s_i)| = \epsilon_m < 1$. Let

$$\hat{\mathbf{S}} = \operatorname{diag}[s_{\ell+1}, \cdots, s_n] \quad \text{and} \quad \hat{\mathbf{W}} = [\mathbf{w}_{\ell+1}, \cdots, \mathbf{w}_n].$$

Then $\mathbf{W} = [\mathbf{V}_\ell \mathbf{T}, \hat{\mathbf{W}}]$ for some $\mathbf{T} \in \mathbb{R}^{\ell\times n}$. Since $\mathbf{x}_k^T \tilde{\mathbf{H}}_k \mathbf{x}_k = \mathbf{x}_k^T(\mathbf{A} - \rho_k \mathbf{B})\mathbf{x}_k = 0$ and $\mathbf{x}_k^T \tilde{\mathbf{H}}_k \mathbf{x}_k = \mathbf{x}_k^T \mathbf{W}\mathbf{S}\mathbf{W}^T \mathbf{x}_k = \sum_{i=\ell+1}^n s_i(\mathbf{w}_i^T \mathbf{x}_k)^2$ with $s_i > 0$ for $i \geq \ell+2$, we have $\mathbf{w}_{\ell+1}^T \mathbf{x}_k \neq 0$. Hence $\mathbf{P_V} q(\tilde{\mathbf{H}}_k)\mathbf{x}_k = \mathbf{P_V}\mathbf{W}q(\mathbf{S})\mathbf{W}^T\mathbf{x}_k = [0, \mathbf{P_V}\hat{\mathbf{W}}]q(\mathbf{S})\mathbf{W}^T\mathbf{x}_k \neq 0$ where we note that $\mathbf{V}_\ell^T \hat{\mathbf{W}} = 0$ and hence $\mathbf{P_V}\hat{\mathbf{W}}$ has full column rank. Let $\mathbf{B}_1 = \hat{\mathbf{W}}^T\mathbf{P}_\mathbf{V}^T\mathbf{B}\mathbf{P_V}\hat{\mathbf{W}}$ and $y = \hat{\mathbf{W}}^T\mathbf{x}_k$. Then

$$\begin{aligned}
\rho_{k+1} &\leq \rho_k + \frac{\mathbf{x}_k^T q(\tilde{\mathbf{H}}_k)\tilde{\mathbf{H}}_k q(\tilde{\mathbf{H}}_k)\mathbf{x}_k}{\mathbf{x}_k^T q(\tilde{\mathbf{H}}_k)(\mathbf{P}_\mathbf{V}^T\mathbf{B}\mathbf{P_V})q(\tilde{\mathbf{H}}_k)\mathbf{x}_k} \\
&= \rho_k + \frac{\mathbf{x}_k^T\mathbf{W}q(\mathbf{S})\mathbf{S}q(\mathbf{S})\mathbf{W}^T\mathbf{x}_k}{\mathbf{x}_k^T\mathbf{W}q(\mathbf{S})\mathbf{W}^T\mathbf{P}_\mathbf{V}^T\mathbf{B}\mathbf{P_V}\mathbf{W}q(\mathbf{S})\mathbf{W}^T\mathbf{x}_k} \\
&= \rho_k + \frac{\mathbf{x}_k^T\hat{\mathbf{W}}q^2(\hat{\mathbf{S}})\hat{\mathbf{S}}\hat{\mathbf{W}}^T\mathbf{x}_k}{\mathbf{x}_k^T\hat{\mathbf{W}}q(\hat{\mathbf{S}})\hat{\mathbf{W}}^T\mathbf{P}_\mathbf{V}^T\mathbf{B}\mathbf{P_V}\hat{\mathbf{W}}q(\hat{\mathbf{S}})\hat{\mathbf{W}}^T\mathbf{x}_k} \\
&= \rho_k + \frac{\mathbf{y}^T q^2(\hat{\mathbf{S}})\hat{\mathbf{S}}\mathbf{y}}{\mathbf{y}^T q(\hat{\mathbf{S}})\mathbf{B}_1 q(\hat{\mathbf{S}})\mathbf{y}}. \qquad (2.26)
\end{aligned}$$

where we have used

$$\mathbf{W}q(\mathbf{S})\mathbf{S}q(\mathbf{S})\mathbf{W}^T = \hat{\mathbf{W}}q^2(\hat{\mathbf{S}})\hat{\mathbf{S}}\hat{\mathbf{W}}^T$$

and

$$\mathbf{P_V}\mathbf{W}q(\mathbf{S})\mathbf{W}^T = [\mathbf{O}, \mathbf{P_V}\hat{\mathbf{W}}]q(\mathbf{S})\mathbf{W}^T = \mathbf{P_V}\hat{\mathbf{W}}q(\hat{\mathbf{S}})\hat{\mathbf{W}}^T.$$

Let $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{n-\ell}]^T$, $\hat{\mathbf{y}} = [\mathbf{0}, \mathbf{y_2}, \ldots, \mathbf{y_{n-\ell}}]^\mathbf{T}$, and $\mathbf{e}_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^{n-\ell}$. Then,

$$\begin{aligned}
\mathbf{y}^T q(\hat{\mathbf{S}})B_1 q(\hat{\mathbf{S}})\mathbf{y} &= (\mathbf{y}_1\mathbf{e}_1 + \hat{\mathbf{y}})^T q(\hat{\mathbf{S}})B_1 q(\hat{\mathbf{S}})(\mathbf{y}_1\mathbf{e}_1 + \hat{\mathbf{y}}) \\
&= \mathbf{y}_1^2 q(s_{\ell+1})^2\mathbf{e}_1^T\mathbf{B}_1\mathbf{e}_1 + 2\mathbf{y}_1 q(s_{\ell+1})\mathbf{e}_1^T\mathbf{B}_1 q(\hat{\mathbf{S}})\hat{\mathbf{y}} + \hat{\mathbf{y}}^T q(\hat{\mathbf{S}})B_1 q(\hat{\mathbf{S}})\hat{\mathbf{y}} \\
&= \mathbf{y}_1^2\beta_1^2 + 2\mathbf{y}_1\beta_2 + \beta_3^2,
\end{aligned}$$

where $\beta_1 \geq 0, \beta_2$ and $\beta_3 \geq 0$ are defined such that

$$\begin{aligned}
\beta_1^2 &= \mathbf{e}_1^T\mathbf{B}_1\mathbf{e}_1 = w_{\ell+1}^T\mathbf{P}_\mathbf{V}^T\mathbf{B}\mathbf{P_V}\mathbf{w}_{\ell+1}, \\
\beta_3^2 &= \hat{\mathbf{y}}^T q(\hat{\mathbf{S}})\mathbf{B}_1 q(\hat{\mathbf{S}})\hat{\mathbf{y}} \\
&\leq \max_{l+1\leq i\leq n} q(s_i)^2\|\mathbf{B}_1\|\|\hat{\mathbf{y}}\|^2 \\
&= \epsilon_m^2\|\mathbf{B}\|\|\hat{\mathbf{y}}\|^2
\end{aligned}$$

and

$$|\beta_2| = |\mathbf{e}_1^T\mathbf{B}_1 q(\hat{\mathbf{S}})\hat{w}| \leq \beta_1\beta_3.$$

Since $\mathbf{y}^T\hat{\mathbf{S}}y = \mathbf{x}_k^T\hat{\mathbf{W}}\hat{\mathbf{S}}\hat{\mathbf{W}}^T\mathbf{x}_k = \mathbf{x}_k^T\tilde{\mathbf{H}}_k\mathbf{x}_k = 0$, we have $\sum_{i=1}^{n-\ell} s_{\ell+i}\mathbf{y}_i^2 = 0$. Then

$$|s_{\ell+1}|\mathbf{y}_1^2 = \sum_{i=2}^{n-\ell} s_{\ell+i}\mathbf{y}_i^2 \geq s_{\ell+2}\|\hat{\mathbf{y}}\|^2, \tag{2.27}$$

and hence

$$\beta_3 \leq \epsilon_m\|\mathbf{B}\|^{1/2}\left(\frac{|s_{\ell+1}|}{s_{\ell+2}}\right)^{1/2}|\mathbf{y}_1|. \tag{2.28}$$

On the other hand, we also have

$$\mathbf{y}^T q^2(\hat{\mathbf{S}})\hat{\mathbf{S}}y = \sum_{i=1}^{n-\ell} s_{\ell+i}q^2(s_{\ell+i})\mathbf{y}_i^2 \leq \sum_{i=1}^{n-\ell} s_{\ell+i}\mathbf{y}_i^2 = \mathbf{y}^T\hat{\mathbf{S}}\mathbf{y} = 0$$

and

$$0 \leq \hat{\mathbf{y}}^T q^2(\hat{\mathbf{S}})\hat{\mathbf{S}}\hat{\mathbf{y}} = \sum_{i=2}^{n-\ell} s_{\ell+i}q^2(s_{\ell+i})\mathbf{y}_i^2 \leq \epsilon_m^2\sum_{i=2}^{n-\ell} s_{\ell+i}\mathbf{y}_i^2 = \epsilon_m^2|s_{\ell+1}|\mathbf{y}_1^2, \tag{2.29}$$

where we have used that $q(s_{\ell+1}) = 1$, $|q(s_{\ell+i})| \leq \epsilon_m < 1$ for $i > 1$ and (2.27). Thus

$$\begin{aligned}
\frac{\mathbf{y}^T q^2(\hat{\mathbf{S}})\hat{\mathbf{S}}\mathbf{y}}{\mathbf{y}^T q(\hat{\mathbf{S}})\mathbf{B}_1 q(\hat{\mathbf{S}})\mathbf{y}} &\leq \frac{\mathbf{y}_1^2 s_{\ell+1} + \hat{\mathbf{y}}^T q(\hat{\mathbf{S}})^2\hat{\mathbf{S}}\hat{\mathbf{y}}}{\mathbf{y}_1^2\beta_1^2 + 2|\mathbf{y}_1|\beta_1\beta_3 + \beta_3^2} \\
&= \frac{s_{\ell+1}}{\beta_1^2} - \frac{s_{\ell+1}}{\beta_1^2}\frac{2|\mathbf{y}_1|\beta_1\beta_3 + \beta_3^2}{\mathbf{y}_1^2\beta_1^2 + 2|\mathbf{y}_1|\beta_1\beta_3 + \beta_3^2} \\
&\quad + \frac{\hat{\mathbf{y}}^T q(\hat{\mathbf{S}})^2\hat{\mathbf{S}}\hat{\mathbf{y}}}{\mathbf{y}_1^2\beta_1^2 + 2|\mathbf{y}_1|\beta_1\beta_3 + \beta_3^2} \\
&\leq \frac{s_{\ell+1}}{\beta_1^2} - \frac{s_{\ell+1}}{\beta_1^2}\frac{2|\mathbf{y}_1|\beta_1\beta_3}{\mathbf{y}_1^2\beta_1^2} + \frac{\hat{\mathbf{y}}^T q(\hat{\mathbf{S}})^2\hat{\mathbf{S}}\hat{\mathbf{y}}}{\mathbf{y}_1^2\beta_1^2} \\
&\leq \frac{s_{\ell+1}}{\beta_1^2} + 2\left(\frac{|s_{\ell+1}|}{\beta_1^2}\right)^{3/2}\epsilon_m\left(\frac{\|\mathbf{B}\|}{s_{\ell+2}}\right)^{1/2} + \frac{|s_{\ell+1}|}{\beta_1^2}\epsilon_m^2, \qquad (2.30)
\end{aligned}$$

where we have used (2.28) and (2.29). Finally, combining (2.26), (2.30), and Lemma 2.2.2, we have

$$\begin{aligned}
0 \leq \rho_{k+1} - \lambda_{\ell+1} &\leq \rho_k - \lambda_{\ell+1} + \frac{s_{\ell+1}}{\beta_1^2} + 2(\rho_k - \lambda_{\ell+1})^{3/2}\epsilon_m\left(\frac{\|\mathbf{B}\|}{s_{\ell+2}}\right)^{1/2} + (\rho_k - \lambda_{\ell+1})\epsilon_m^2 \\
&\leq \delta_k + 2(\rho_k - \lambda_{\ell+1})^{3/2}\epsilon_m\left(\frac{\|\mathbf{B}\|}{s_{\ell+2}}\right)^{1/2} + (\rho_k - \lambda_{\ell+1})\epsilon_m^2,
\end{aligned}$$

where $\delta_k = \rho_k - \lambda_{\ell+1} + \frac{s_{\ell+1}}{\beta_1^2} = \mathcal{O}((\rho_k - \lambda_{\ell+1})^2)$ by Lemma 2.2.2. The proof is complete. □

Remark ϵ_m in the theorem can be bounded by the Chebyshev polynomials as

$$\epsilon_m \leq \frac{1}{T_m\left(\frac{1+\psi}{1-\psi}\right)}, \quad \text{where} \quad \psi = \frac{s_{l+2} - s_{l+1}}{s_n - s_{l+1}} \tag{2.31}$$

and T_m is the Chebyshev polynomial of degree m. This bound can be further simplified to

$$\epsilon_m \leq 2\left(\frac{1-\sqrt{\psi}}{1+\sqrt{\psi}}\right)^m \tag{2.32}$$

to show the dependence on the spectral separation ψ. Thus, the speed of convergence of the deflation algorithm depends on the spectral gap of the smallest nonzero eigenvalue of $\mathbf{P}_{\mathbf{V}}^T(\mathbf{A} - \rho_k\mathbf{B})\mathbf{P}_{\mathbf{V}}$, rather than that of $\mathbf{A} - \rho_k\mathbf{B}$ in the original algorithm. In particular, this may have a different convergence characteristic from the Wielandt deflation (2.14).

We also note that for small m, the bound (2.31) may be significantly stronger than (2.32), but when m is sufficiently large, (2.32) is almost as good as (2.31). It is also easy to see that, asymptotically, we can use the eigenvalues of $\mathbf{P}_{\mathbf{V}}^T(\mathbf{A}-\lambda_{\ell+1}\mathbf{B})\mathbf{P}_{\mathbf{V}}$ in the place of $s_{\ell+1} \leq s_{\ell+2} \leq \cdots \leq s_n$ without changing the first order term of the bound; see [32] for more discussions.

As in Section 2.2.1, a congruence transformation can be used in Algorithm 2.5 to reduce ϵ_m to 0 so as to accelerate convergence. Consider the ideal situation that we compute the LDL^T-decomposition of $\mathbf{P}_{\mathbf{V}}^T(\mathbf{A} - \rho_k\mathbf{B})\mathbf{P}_{\mathbf{V}} = \mathbf{L}_k\mathbf{D}_k\mathbf{L}_k^T$ with $\mathbf{D}_k$ being a diagonal matrix of 0 and ±1. Then the congruence transformation

$$(\hat{\mathbf{A}}_k, \hat{\mathbf{B}}_k) := (\mathbf{L}_k^{-1}\mathbf{A}\mathbf{L}_k^{-T}, \mathbf{L}_k^{-1}\mathbf{B}\mathbf{L}_k^{-T})$$

does not change the spectrum of $(\mathbf{A}, \mathbf{B})$. Applying Algorithm 2.5 to the transformed problem, we use

$$\hat{\mathbf{V}}_\ell := \mathbf{L}_k\mathbf{V}_\ell$$

to construct the projection $\mathbf{P}_{\hat{\mathbf{V}}} := \mathbf{I} - \hat{\mathbf{B}}\hat{\mathbf{V}}_\ell\hat{\mathbf{V}}_\ell^T$, as $\hat{\mathbf{A}}_k\hat{\mathbf{v}}_i = \lambda_i\hat{\mathbf{B}}_k\hat{\mathbf{v}}_i(1 \leq i \leq \ell)$ and $\hat{\mathbf{V}}_\ell^T\hat{\mathbf{B}}_k\hat{\mathbf{V}}_\ell = \mathbf{I}$. Then, by Theorem 2.2.4, the convergence rate is determined by the eigenvalues of $\mathbf{P}_{\hat{\mathbf{V}}}^T(\hat{\mathbf{A}} - \rho_k\hat{\mathbf{B}})\mathbf{P}_{\hat{\mathbf{V}}}$. It is easy to see that

$$\mathbf{P}_{\hat{\mathbf{V}}}^T(\hat{\mathbf{A}} - \rho_k\hat{\mathbf{B}})\mathbf{P}_{\hat{\mathbf{V}}} = \mathbf{L}_k^{-1}\mathbf{P}_{\mathbf{V}}^T(\mathbf{A} - \rho_k\mathbf{B})\mathbf{P}_{\mathbf{V}}\mathbf{L}_k^{-T} = \mathbf{D}_k. \tag{2.33}$$

Then at the convergence stage with $\lambda_{l+1} < \rho_k < \lambda_{l+2}$, we have $s_1 = \cdots = s_l = 0$ and $s_{l+1} = -1$, $s_{l+2} = \ldots = s_n = 1$, which implies, for $m \geq 1$, $\epsilon_m = 0$, and hence by Theorem 2.2.4,

$$\rho_k - \lambda_{l+1} \leq \delta_k = \mathcal{O}((\rho_k - \lambda_{l+1})^2).$$

The above is an ideal situation that requires computing the LDL^T-decomposition. In practice, we can use an incomplete LDL^T-decomposition of $\mathbf{P}_{\mathbf{V}}^T(\mathbf{A} - \mu\mathbf{B})\mathbf{P}_{\mathbf{V}} = \mathbf{L}_k\mathbf{D}_k\mathbf{L}_k^T$ with a shift $\mu \approx \rho_k$ (or $\lambda_{\ell+1}$), which would reduce ϵ_m and hence accelerate convergence.

2.3 Numerical examples

In this section, we present two numerical examples to demonstrate the convergence properties of the deflation by restriction for the inverse free Krylov subspace method. All computations were carried out using MATLAB version 8.0.0.783 from MathWorks on a PC with an Intel quad-core i7-2670QM @ 2.20GHz and 12 GB of RAM running Ubuntu Linux 12.04. The machine epsilon is $\mathbf{u} \approx 2.2 \cdot 10^{-16}$.

Our implementation is based on the MATLAB program `eigifp` of [62]. In particular, the basis of the projected Krylov subspace is constructed using the Arnoldi method. In both examples, we compute the three smallest eigenvalues and use the deflation algorithm in computing the second and the third smallest eigenvalues. The initial vectors are generated by `randn(n,3)` and we fix the number of inner iterations as $m = 20$. Note that m can be set to be chosen adaptively in `eigifp`, but here we consider a fixed m for the demonstration of the convergence bound by ϵ_m. The stopping criterion is set as $\|\mathbf{r}_k\| \leq 10^{-8}$, where $\mathbf{r}_k = (\mathbf{A}\mathbf{x}_k - \rho_k\mathbf{B}\mathbf{x}_\mathbf{k})/\|\mathbf{x}_\mathbf{k}\|$.

Example 1. Consider the Laplace eigenvalue problem with the Dirichlet boundary condition on an L-shaped domain. A definite symmetric generalized eigenvalue problem $\mathbf{A}x = \lambda\mathbf{B}\mathbf{x}$ is obtained by a finite element discretization on a mesh with 20,569 interior nodes using PDE toolbox of MATLAB. Three iterations of deflation algorithms are carried out to compute the three smallest eigenvalues and we plot the convergence history of the residuals $\|\mathbf{r}_k\|$ against the number of iterations for the three eigenvalues $\lambda_i (1 \leq i \leq 3)$ together in Figure 2.1. To illustrate Theorem 2.2.4, we also plot in Figure 2.2 the convergence rate $(\rho_{k+1} - \lambda_i)/(\rho_k - \lambda_i)$ and compare it with the upper bound (2.31) of ϵ_m^2. For the purpose of simplicity, the bound (2.31) is computed from the eigenvalues of the projected matrix $\mathbf{P}_\mathbf{V}^T(\mathbf{A} - \lambda_i\mathbf{B})\mathbf{P}_\mathbf{V}$. The top straight lines are the upper bounds of ϵ_m^2 and the bottom three lines are the corresponding actual error ratios $(\rho_{k+1} - \lambda_i)/(\rho_k - \lambda_i)$.

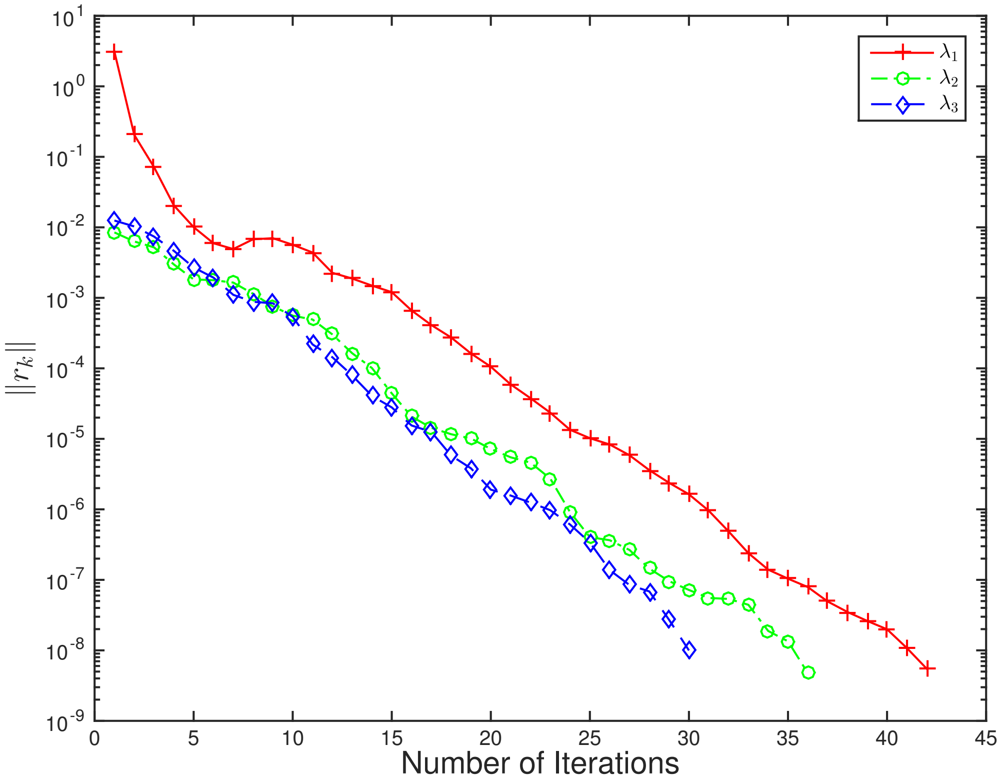

Figure 2.1: Convergence History of Residuals for three eigenvalues $\lambda_1, \lambda_2, \lambda_3$

We observe that the deflation algorithm converges indeed linearly and (2.31) provides a good bound on the rate of convergence. We note that λ_1 takes more iterations overall than the other two eigenvalues. This is due to the use of initial random vector for λ_1, but to compute λ_2 and λ_3 in the `eigifp` implementation, initial approximate eigenvectors are computed from the projection used to compute λ_1. As a result, λ_2 and λ_3 have smaller initial errors, but their overall convergence rates are still com-

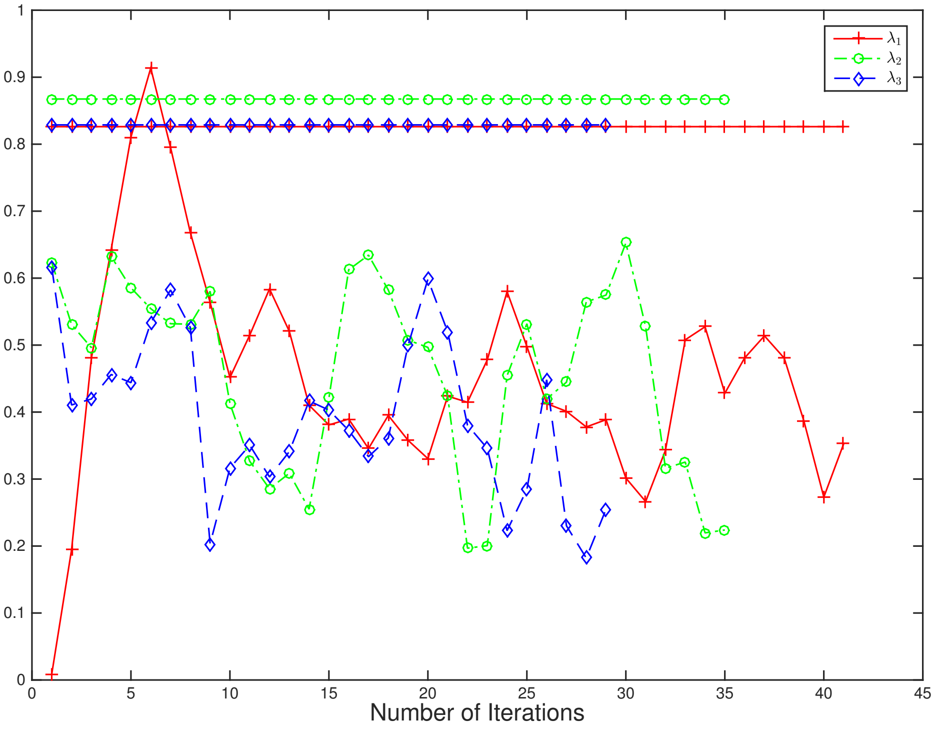

Figure 2.2: Top: bound ϵ_m^2; Bottom: error ratio $(\rho_{k+1} - \lambda_i)/(\rho_k - \lambda_i)$.

parable as suggested by their bounds. Finally, we list all the converged eigenvalues, the number of iterations used to reduce the residuals below the threshold, the CPU time for computing each eigenvalue, and their final residuals in Table 2.1.

Table 2.1: 3 smallest eigenvalues of Laplacian eigenvalue problem on L-shaped domain

λ_l	Number of Iterations	CPU	Residual $\|\mathbf{r}_k\|$
23.3876	42	3.51	5.50e-09
37.9873	36	3.12	4.76e-09
47.4515	30	2.65	9.98e-09

Example 2. In this example, we consider the deflation algorithm when used with preconditioning. **A** and **B** are the same finite element matrices as in Example 1. For preconditioning, we use a constant **L** as obtained by the threshold incomplete LDL^T factorization of $\mathbf{A} - \mu_i B$ with the drop tolerance 10^{-2}, where the shift μ_i is an approximation of the desired eigenvalue λ_i. We use $\mu_1 = 0$ for λ_1 and $\mu_i = \lambda_{i-1}$ for

$i > 1$. Then, the convergence rate is given by ϵ_m as determined by the eigenvalues of $\mathbf{L}^{-1}\mathbf{P}_{\mathbf{V}}^T(\mathbf{A} - \lambda_i\mathbf{B})\mathbf{P}_{\mathbf{V}}\mathbf{L}^{-\mathbf{T}}$ as in (2.31).

As in Example 1, three iterations of deflation algorithms with preconditioning are carried out to compute the three smallest eigenvalues. We plot the convergence history of the residuals $\|\mathbf{r}_k\|$ in Figure 2.3 and the convergence rate $(\rho_{k+1}-\lambda_i)/(\rho_k-\lambda_i)$ as well as its upper bound (2.31) in Figure 2.4. We also list all the converged eigenvalues, the number of iterations used to reduce the residuals below the threshold, the CPU time for computing each eigenvalue (the CPU time for constructing preconditioner is given in parenthesis), and their final residuals in Table 2.2.

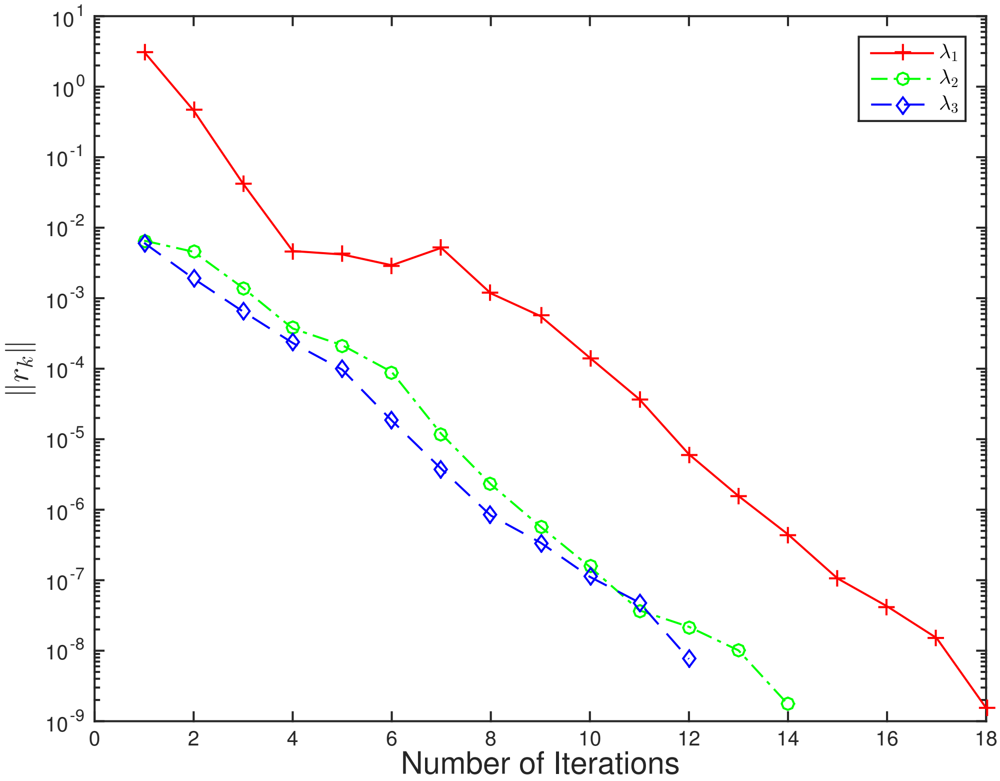

Figure 2.3: Convergence History of Residuals for three eigenvalues $\lambda_1, \lambda_2, \lambda_3$

We observe that the deflation algorithm with preconditioning converges linearly and (2.31) provides a very good bound on the rate of convergence. In particular, with the preconditioning, the convergence bounds are significantly improved and correspondingly, the actual convergence rates are also improved demonstrating the

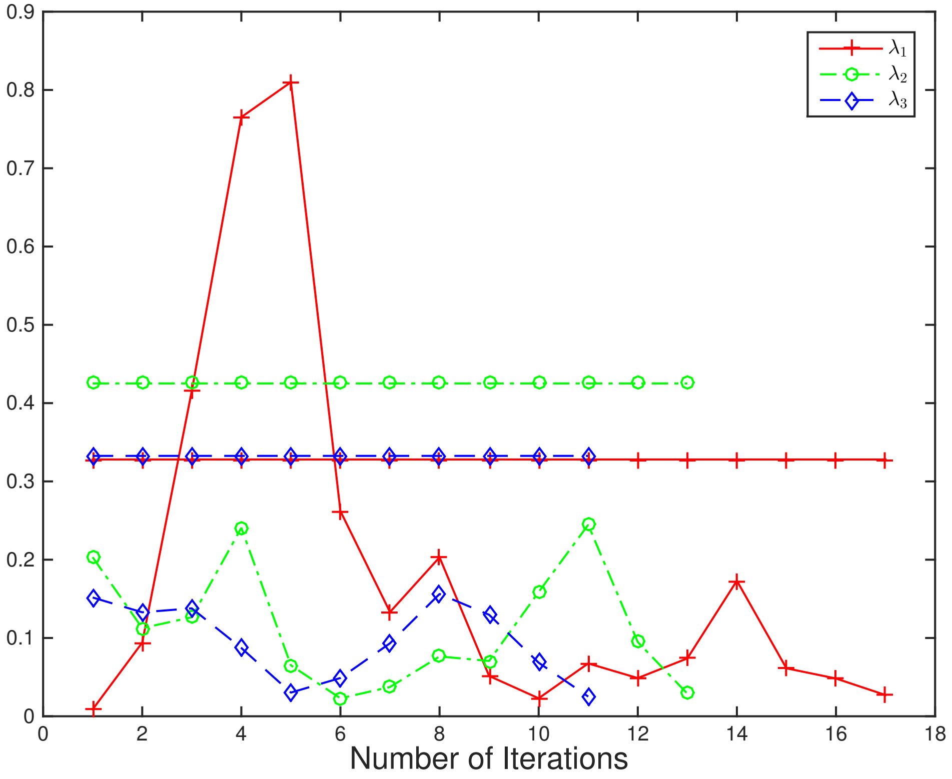

Figure 2.4: Top: bound ϵ_m^2; Bottom: error ratio $(\rho_{k+1} - \lambda_i)/(\rho_k - \lambda_i)$.

Table 2.2: 3 smallest eigenvalues of Laplacian eigenvalue problem on L-shaped domain

λ_l	Number of Iterations	CPU	Residual $\|\mathbf{r}_k\|$
23.3876	18	2.01(0.08)	1.54e-09
37.9873	14	1.65(0.10)	1.73e-09
47.4515	12	1.39(0.10)	7.63e-09

effects of preconditioning.

Chapter 3 An Inverse-free preconditioned Krylov subspace method for singular values problem

In this chapter, we consider the problem of computing a few singular values of large matrices.

The Singular Value Decomposition(SVD) is one of the major matrix decompositions which is used for many different purposes such as total least squares problems, low-rank matrix approximation. Applications of SVD varies from signal processing to machine learning. The matrices involved in these applications are usually large and sparse. Computing a complete SVD of those matrices is very expensive. Fortunately, only a partial SVD is needed in most cases.

In Section 3.1, we introduce the singular value decomposition and its properties. We briefly review the existing algorithms for computing a few extreme singular values of large matrices in Section 3.2. Then we adapt the inverse-free preconditioned algorithm of [32] for the singular value problem in Section 3.3. The new algorithm, which we call it `svdifp`, overcomes diffculites of computing smallest singular values experienced by other algorithms, such as the Matlab built-in function `svds, jdsvd, irlba,` etc. Extensive numerical tests are presented to demonstrate efficiency and robustness of the new algorithm in Section 3.4.

3.1 Singular value decomposition

The Singular value decomposition(SVD) of a matrix $\mathbf{C} \in \mathbb{R}^{m \times n}$ with $m \geq n$. is defined as

$$\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where $\mathbf{U} \in \mathbb{R}^{m \times n}$ with $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, $\mathbf{V} \in \mathbb{R}^{n \times n}$ with $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ and $\mathbf{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_n)$, where $0 \leq \sigma_1 \leq \ldots \leq \sigma_n$. The columns $\mathbf{u}_1, \ldots, \mathbf{u}_n$ of $\mathbf{U}$ are called left singular vectors. The columns $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are called right singular vectors. The σ_i are called singular values. If $m < n$, the SVD is defined by considering $\mathbf{C}^T$. ([19, Theorem 3.2])

The singular value decomposition is closely related to the eigendecomposition.

Theorem 3.1.1. *([19, Theorem 3.3])*

1. *The eigenvalues of the symmetric matrix* $\mathbf{C}^T\mathbf{C}$ *are* σ_i^2*. The right singular vectors* $\mathbf{v}_i$ *are corresponding orthonormal eigenvectors.*

2. *The eigenvalues of the symmetric matrix* $\mathbf{C}\mathbf{C}^T$ *are* σ^2 *and* $m-n$ *zeros. The left singular vectors* $\mathbf{u}_i$ *are corresponding orthonormal eigenvectors for the eigenvalues* σ_i*.*

3. *The augmented matrix* $\mathbf{M} := \begin{bmatrix} \mathbf{O} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{O} \end{bmatrix}$ *has eigenvalues*

$$-\sigma_n \leq \ldots \leq -\sigma_2 \leq -\sigma_1 \leq \underbrace{0 = \ldots = 0}_{m-n} \leq \sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_n$$

Computing the complete SVD of those matrices is very expensive. In this dissertation, our major interest is in partial SVD of large and sparse matrices, so we will not talk about the compuation of complete SVD. For readers who are interested in this topic, see ([19, 30]).

3.2 Computations of singular values of large and sparse matrices

Consider the problem of computing a few of extreme (i.e. largest or smallest) singular values and corresponding singular vectors of $\mathbf{C}$. In this section, we consider real matrices, but all can be generalized to complex matrices in a trivial way. By Theorem 3.1.1, most existing numerical methods are based on reformulating the singular value problem as one of the following two symmetric eigenvalue problems by Theorem 3.1.1:

$$\sigma_1^2 \leq \sigma_2^2 \leq \cdots \leq \sigma_n^2 \text{ are the eigenvalues of } \mathbf{C}^T\mathbf{C} \tag{3.1}$$

and

$$-\sigma_n \leq \cdots \leq -\sigma_2 \leq -\sigma_1 \leq \underbrace{0 = \cdots = 0}_{m-n} \leq \sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_n$$

are the eigenvalues of the *augmented matrix*

$$\mathbf{M} = \begin{bmatrix} \mathbf{O} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{O} \end{bmatrix} \tag{3.2}$$

To compute a few extreme singular values of $\mathbf{C}$, we can apply the Lanczos algorithm (Algorithm 2.3) or the implicitly restarted Lanczos algorithm [76] (ARPACK [49]) to one of the two formulations ((3.1) and (3.2)) and this can often be done implicitly. Indeed, several methods have been introduced that exploit the special structure and the associated properties of these eigenvalue problems.

3.2.1 The Lanczos bidiagonalization method

The Lanczos bidiagonalization method introduced in [29] is a widely used method for the singular value problems that implicitly applies the Lanczos method to formulation (3.1). A robust implementation called `lansvd` is provided in PROPACK [47]. The implicit restart strategy has been developed for the Lanczos bidiagonalization algorithm in [5] and [44], which also include robust MATLAB implementations `irlba` and `irlanb` respectively. Other aspects of the Lanczos bidiagonalization algorithm are discussed in [12, 38, 80].

These methods, based on the Lanczos algorithm for the eigenvalue problem (3.1), work well when the corresponding eigenvalue is reasonably well separated. However, their convergence may be slow if the eigenvalue is clustered, which turns out to be often the case when computing the smallest singular values through (3.1). Specifically, for formulation (3.1), the spectral separation for σ_1^2 as an eigenvalue of $\mathbf{C}^T\mathbf{C}$ may be much smaller than the separation of σ_1 from σ_2 since

$$\frac{\sigma_2^2 - \sigma_1^2}{\sigma_n^2 - \sigma_2^2} = \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_2} \frac{\sigma_1 + \sigma_2}{\sigma_n + \sigma_2} \ll \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_2} \tag{3.3}$$

(assuming $\sigma_2 \ll \sigma_n$). On the other hand, for formulation (3.2), σ_1 is an interior eigenvalue of $\mathbf{M}$, for which a direct application of the Lanczos algorithm does not usually result in convergence.

3.2.2 MATLAB's routine svds

To compute a few of the smallest singular values, MATLAB's routine `svds` applies ARPACK [49, 76] to the augmented matrix $\mathbf{M}$ (3.2) with a shift-and-invert transformation. This works well for square matrices. However, for computing the smallest singular value of a non-square matrix, a subtle difficulty arises in using the shift-and-invert transformation for $\mathbf{M}$ because $\mathbf{M}$ is singular and, with a shift close to 0, the method often converges to one of the $m - n$ zero eigenvalues of $\mathbf{M}$ rather than to σ_1.

3.2.3 JDSVD

On the other hand, one can avoid the shift-and-invert by considering the Jacobi-Davidson method on the augmented matrix (3.2) and a method of this type, called JDSVD, has been developed in [35, 36] that efficiently exploits the block structure of (3.2). The JDSVD method replaces the shift-and-invert by approximately solving so-called correction equations using a preconditioned iterative method. When computing σ_1 as an interior eigenvalue of the augmented matrix (3.2), the convergence of JDSVD appears to strongly depend on the quality of the preconditioner for the correction equation. This demands a good preconditioner for $\mathbf{M}$ or $\mathbf{M} - \mu\mathbf{I}$, which is unfortunately difficult to construct when $m \neq n$ owing to the singularity of $\mathbf{M}$.

3.3 SVDIFP–The proposed algorithm

It appears that the augmented matrix formulation (3.2) has some intrinsic difficulties when it is used for computing a few of the smallest singular values of a non-square matrix because of the existence of the zero eigenvalues of $\mathbf{M}$. For this reason, we propose to reconsider formulation (3.1) in this situation. While formulation (3.1) has the advantage of a smaller dimension in the underlying eigenvalue problem, a clear disadvantage is that there is no efficient method to carry out the shift-and-invert transformation $(\mathbf{C}^T\mathbf{C} - \mu\mathbf{I})^{-1}$ other than explicitly forming $\mathbf{C}^T\mathbf{C}$.

Note that $\mathbf{C}^T\mathbf{C}$ is typically much denser than $\mathbf{C}$ and explicitly computing $\mathbf{C}^T\mathbf{C}$ may result in loss of accuracy with the condition number being squared. In the case of $\mu = 0$, which can be used to compute σ_1 that is sufficiently close to 0, the inverse of $\mathbf{C}^T\mathbf{C}$ can be implicitly obtained by computing the QR factorization of $\mathbf{C}$. This is the approach taken in `lansvd` of PROPACK [47]. However, since a complete QR factorization of a sparse matrix may be expensive owing to possible excessive fill-ins of the zero entries, it will be interesting to study other approaches that use incomplete factorizations instead. Other drawbacks of (3.1) include the need to compute left singular vectors when they are required, and the potential loss of accuracy caused by computing σ_1^2 when σ_1 is tiny (see Section 3.3.1). In particular, the computed left singular vectors may have a low accuracy if the singular values are small (see the discussions in Section 3.3.1).

In this section, we propose to address the small separation of σ_1^2 in formulation (3.1) by considering a preconditioned Krylov subspace method. Specifically, we shall implicitly apply the inverse-free preconditioned Krylov subspace method of [32] (or its block version [70]) to $\mathbf{A} = \mathbf{C}^T\mathbf{C}$. As already discussed, the standard shift-and-invert transformation is not practical for (3.1) as it requires factorization of $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$. The inverse-free preconditioned Krylov subspace method is an effective way to avoid the shift-and-invert transformation for computing a few extreme eigenvalues of the symmetric generalized eigenvalue problem $\mathbf{A}x = \lambda\mathbf{B}\mathbf{x}$ where $\mathbf{A}$ and $\mathbf{B}$ are symmetric with $\mathbf{B}$ positive definite. In this method, an approximate eigenvector $\mathbf{x}_k$ is iteratively improved through the Rayleigh-Ritz projection on the Krylov subspace

$$\mathcal{K}_m(\mathbf{H}_k, \mathbf{x}_k) := \text{span}\{\mathbf{x}_k, \mathbf{H}_k\mathbf{x}_k, \mathbf{H}_k^2\mathbf{x}_k, \ldots, \mathbf{H}_k^m\mathbf{x}_k\} \tag{3.4}$$

where $\mathbf{H}_k := \mathbf{A} - \rho_k\mathbf{B}$ and ρ_k is the Rayleigh quotient of $\mathbf{x}_k$. The projection is carried out by constructing a basis for the Krylov subspace through an inner iteration, where the matrices $\mathbf{A}$ and $\mathbf{B}$ are only used to form matrix-vector products. The method is proved to converge at least linearly and the rate of convergence is determined by the spectral gap of the smallest eigenvalue of $\mathbf{H}_k$ (rather than the original eigenvalue problem as in the Lanczos method). An important implication of this property is that a congruence transformation of $(\mathbf{A}, \mathbf{B})$ derived from an incomplete LDL^T factorization of a shifted matrix $\mathbf{A} - \mu\mathbf{B}$ may be applied to reduce the spectral gap of the smallest eigenvalue of $\mathbf{H}_k$ and hence to accelerate the convergence to the extreme eigenvalue. This is referred to as preconditioning. A block version of this algorithm has also been developed in [70] to address multiple or severely clustered eigenvalues.

In applying the inverse-free preconditioned Krylov subspace method [32, 70] to $\mathbf{A} = \mathbf{C}^T\mathbf{C}$, we shall construct directly the projection of $\mathbf{C}$ rather than the projection of $\mathbf{C}^T\mathbf{C}$ used for the eigenvalue problem. In this way, we compute approximation of σ_1 directly from the singular values of the projection of $\mathbf{C}$ rather than using the theoretically equivalent process of computing approximation of σ_1^2 from the projection of $\mathbf{C}^T\mathbf{C}$. By computing σ_1 directly, we avoid the pitfall of loss of accuracy associated with computing σ_1^2 if σ_1 is tiny. On the other hand, the potential difficulty with the accuracy of the computed left singular vector in this case is intrinsic to the approach of $\mathbf{C}^T\mathbf{C}$. An efficient implementation of the inverse-free preconditioned Krylov subspace method depends on the construction of a preconditioner derived from an incomplete LDL^T factorization of $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$. Constructing a preconditioner for $\mathbf{C}^T\mathbf{C}$ has been discussed extensively in the literature in the context of solving least squares problems (see [7, 10, 11, 28, 64, 66, 86]) and one method well suited for our problem is the robust incomplete factorization (RIF) of [10, 11]. For the shifted matrix $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$, however, there is no known effective method for computing a factorization without forming $\mathbf{C}^T\mathbf{C}$ first. It turns out that the robust incomplete factorization (RIF) can be easily adapted to construct an LDL^T factorization of the shifted matrix $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$ without forming $\mathbf{C}^T\mathbf{C}$. Our numerical testing demonstrates that the RIF preconditioner in combination with the inverse-free preconditioned Krylov subspace method leads to a very efficient preconditioned algorithm for the singular value problem. Numerical tests demonstrate that it is particularly competitive for computing a few of the smallest singular values of non-square matrices.

3.3.1 An inverse-free preconditioned Krylov subspace method

We consider the singular value problem for an $m \times n$ matrix $\mathbf{C}$. We apply Algorithm 2.4 to the eigenvalue problem $\mathbf{A} = \mathbf{C}^T\mathbf{C}$ and $\mathbf{B} = \mathbf{I}$. However, a direct application involves computing the eigenvalue ρ_k of the projection matrix $\mathbf{A}_m$, which converges to σ_1^2. One potential difficulty associated with this approach is that ρ_k computed this way may have a larger error if σ_1 is very small (relative to $\|\mathbf{C}\|$). Specifically, if $\tilde{\rho}_k$ is the computed Ritz value, it follows from the standard backward error analysis [30] that $\tilde{\rho}_k$ is the exact eigenvalue of a perturbed matrix $\mathbf{A}_m + \mathbf{E}_m$ with $\|\mathbf{E}_m\| = \mathcal{O}(\mathbf{u})\|\mathbf{A}_m\|$, where $\mathbf{u}$ is the machine precision. Then

$$|\tilde{\rho}_k - \rho_k| \leq \mathcal{O}(\mathbf{u})\|\mathbf{A}_m\| \leq \mathcal{O}(\mathbf{u})\|\mathbf{A}\| = \mathcal{O}(\mathbf{u})\|\mathbf{C}\|^2 \tag{3.5}$$

and

$$|\sqrt{\tilde{\rho}_k} - \sqrt{\rho_k}| \leq \mathcal{O}(\mathbf{u})\|\mathbf{C}\| \frac{\|\mathbf{C}\|}{\sqrt{\tilde{\rho}_k} + \sqrt{\rho_k}} \approx \mathcal{O}(\mathbf{u})\|\mathbf{C}\|\kappa(\mathbf{C})/2 \tag{3.6}$$

where $\kappa(\mathbf{C}) = \sigma_n/\sigma_1$ is the condition number of $\mathbf{C}$. In particular, the relative error

$$\frac{|\sqrt{\tilde{\rho}_k} - \sqrt{\rho_k}|}{\sqrt{\rho_k}} \leq \mathcal{O}(\mathbf{u}) \frac{\|\mathbf{C}\|^2}{\sqrt{\rho_k}(\sqrt{\tilde{\rho}_k} + \sqrt{\rho_k})} \approx \mathcal{O}(\mathbf{u})\kappa(\mathbf{C})^2/2$$

is proportional to $\kappa(\mathbf{C})^2$. Thus, very little relative accuracy may be expected if $\kappa(\mathbf{C})$ is of order $1/\sqrt{\mathbf{u}}$. In contrast, a backward stable algorithm should produce an approximation of σ_1 with absolute error in the order of $\mathcal{O}(\mathbf{u})\|\mathbf{C}\|$ and the relative error in the order of $\mathcal{O}(\mathbf{u})\kappa(\mathbf{C})$. We note that the above discussion is based on a worst case upper bound. It is likely pessimistic, particularly in the bound of $\|\mathbf{A}_m\|$, but it does highlight the potential loss of accuracy when one computes σ_1 through computing σ_1^2 (see Example 5.1 in Section 5).

To achieve the desired backward stability, we propose to construct a two-sided projection of $\mathbf{C}$, from which we compute approximate singular values directly. This is similar to the Lanczos bidiagonalization algorithm where a bidiagonal projection matrix is constructed whose singular values directly approximate the singular values of $\mathbf{C}$. Algorithmically, we construct an orthonormal basis $\{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_m\}$ for $\mathcal{K}_m(\mathbf{L^{-T}L^{-1}}(\mathbf{A} - \rho_\mathbf{k}\mathbf{I}), \mathbf{x_k})$ and simultaneously an orthonormal basis $\{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m\}$

for span$\{\mathbf{Cz}_0, \mathbf{Cz}_1, \ldots, \mathbf{Cz}_m\}$ as follows. First, $f_{0,0} = \|\mathbf{Cz}_0\|_2$ and $\mathbf{y}_0 = \mathbf{Cz}_0/f_{0,0}$. Then, for $i = 1, \ldots, m$, we generate $\mathbf{z}_i$ and $\mathbf{y}_i$ by

$$f_{i,i}\mathbf{z}_i = \mathbf{L}^{-T}\mathbf{L}^{-1}(\mathbf{C}^T\mathbf{C}\mathbf{z}_{i-1} - \rho_k\mathbf{z}_{i-1}) - f_{0,i}\mathbf{z}_0 - f_{1,i}\mathbf{z}_1 - \cdots - f_{i-1,i}\mathbf{z}_{i-1} \quad (3.7)$$

$$g_{i,i}\mathbf{y}_i = \mathbf{Cz}_i - g_{0,i}\mathbf{y}_0 - g_{1,i}\mathbf{y}_1 - \cdots - g_{i-1,i}\mathbf{y}_{i-1} \quad (3.8)$$

where $f_{j,i} = \mathbf{z}_j^T\mathbf{L}^{-T}\mathbf{L}^{-1}(\mathbf{C}^T\mathbf{C}\mathbf{z}_{i-1} - \rho_k\mathbf{z}_{i-1})$, $g_{j,i} = \mathbf{y}_j^T\mathbf{Cz}_i$, and $f_{i,i}$ and $g_{i,i}$ are chosen so that $\|\mathbf{y}_i\| = \|\mathbf{z}_i\| = 1$. Assuming $\dim(\mathcal{K}_m(\mathbf{L}^{-T}\mathbf{L}^{-1}(\mathbf{A} - \rho_k I), \mathbf{x}_k)) = m + 1$, the recurrence for $\mathbf{z}_i$ does not breakdown and the process leads to an orthonormal basis $\{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_m\}$. It is easy to show that the recurrence for $\mathbf{y}_i$ does not breakdown either and $\{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m\}$ is orthonormal. Let $\mathbf{Y}_m = [\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m]$. Then $\mathbf{CZ}_m = \mathbf{Y}_m\mathbf{G}_m$ where $\mathbf{G}_m = [g_{ij}]_{i,j=0}^m$. It follows that $\mathbf{Z}_m^T(\mathbf{C}^T\mathbf{C})\mathbf{Z}_m = \mathbf{G}_m^T\mathbf{G}_m$. If $\sigma_k^{(1)}$ is the smallest singular value of $\mathbf{G}_m$, then $(\sigma_k^{(1)})^2$ is the smallest eigenvalue of $\mathbf{A}_m = \mathbf{Z}_m^T(\mathbf{C}^T\mathbf{C})\mathbf{Z}_m$, i.e. $(\sigma_k^{(1)})^2$ so constructed is equal to ρ_{k+1} in Algorithm 2.4.

By computing $\sigma_k^{(1)}$ directly, we avoid the possible loss of accuracy. Specifically, if $\tilde{\sigma}_k^{(1)}$ is the computed singular value of $\mathbf{G}_m$ using the standard SVD algorithm such as `svd`, then it follows from the backward stability that $\tilde{\sigma}_k^{(1)}$ is the exact singular value of $\mathbf{G}_m + \mathbf{F}_m$ for some $\mathbf{F}_m$ with $\|\mathbf{F}_m\| = \mathcal{O}(\mathbf{u})\|\mathbf{G}_m\|$. Then

$$|\tilde{\sigma}_k^{(1)} - \sigma_k^{(1)}| \le \mathcal{O}(\mathbf{u})\|\mathbf{G}_m\| \le \mathcal{O}(\mathbf{u})\|\mathbf{C}\|$$

and hence

$$\frac{|\tilde{\sigma}_k^{(1)} - \sigma_k^{(1)}|}{\sigma_k^{(1)}} \le \mathcal{O}(\mathbf{u})\kappa(\mathbf{C}).$$

3.1 Thus, as Algorithm 2.4 converges, i.e. $\sqrt{\rho_k} = \sigma_k^{(1)} \to \sigma_1$, the approximate singular value $\tilde{\sigma}_k^{(1)}$ can approximate σ_1 with a relative accuracy in the order of $\mathcal{O}(\mathbf{u})\kappa(\mathbf{C})$.

To compute additional eigenvalues, we use the deflation by restriction method proposed in Section 2.2.2. We summarize this process as the following algorithm that compute the $(\ell + 1)$st smallest singular value when the first ℓ singular values have already been computed.

We make some remarks concerning Algorithm 3.1. The algorithm presented has deflation included where ℓ singular values and right singular vectors are given as inputs. When none is given, it computes the smallest singular value σ_1 by setting $\ell = 0$

Algorithm 3.1 Inverse free preconditioned Krylov subspace method for SVD

```
1: Input: $m$, $\mathbf{V}_\ell = [\mathbf{v}_1, \cdots, \mathbf{v}_\ell]$ with $\mathbf{C}^T\mathbf{C}\mathbf{v}_i = \sigma_i^2\mathbf{v}_i$ and $\mathbf{V}_\ell^T\mathbf{V}_\ell = \mathbf{I}$, initial right singular vector $\mathbf{x}_0$ s.t. $\|\mathbf{x}_0\| = 1$ and $\mathbf{V}_\ell^T\mathbf{x}_0 = 0$;
2: initialize: $\rho_0 = \|\mathbf{C}\mathbf{x}_0\|$; $\mathbf{G}_m = [g_{ij}] = 0 \in \mathbb{R}^{(m+1)\times(m+1)}$;
3: for $k = 0, 1, 2, \ldots$ until convergence do
4:     construct a preconditioner $\mathbf{L}$;
5:     $\mathbf{z}_0 = \mathbf{x}_k$; $\mathbf{w} = \mathbf{C}\mathbf{z}_0$; $m' = m$;
6:     $g_{0,0} = \|\mathbf{w}\|$ and $\mathbf{y}_0 = \mathbf{w}/g_{0,0}$;
7:     for $i = 1 : m$ do
8:         $\mathbf{z}_i = (\mathbf{I} - \mathbf{V}_\ell\mathbf{V}_\ell^T)\mathbf{L}^{-T}\mathbf{L}^{-1}(\mathbf{C}^T\mathbf{w} - \rho_k\mathbf{z}_{i-1})$;
9:         for $j = 0 : i-1$ do
10:            $\mathbf{z}_i = \mathbf{z}_i - (\mathbf{z}_j^T\mathbf{z}_i)\mathbf{z}_j$;
11:        end for
12:        if $\|\mathbf{z}_i\| \neq 0$ then
13:            $\mathbf{z}_i = \mathbf{z}_i/\|\mathbf{z}_i\|$
14:        else
15:            $m' = i$ and break;
16:        end if
17:        $\mathbf{w} = \mathbf{C}\mathbf{z}_i$; $\mathbf{y}_i = \mathbf{w}$;
18:        for $j = 0 : i-1$ do
19:            $g_{j,i} = \mathbf{y}_j^T\mathbf{y}_i$ and $\mathbf{y}_i = \mathbf{y}_i - g_{j,i}\mathbf{y}_j$;
20:        end for
21:        $g_{i,i} = \|\mathbf{y}_i\|$ and $\mathbf{y}_i = \mathbf{y}_i/g_{i,i}$;
22:    end for
23:    Compute the smallest singular value $\sigma_{k+1}^{(1)}$ of $\mathbf{G}_m = [g_{ij}]_{i,j=0}^{m'}$ and a corresponding unit right singular vector $\mathbf{h}$;
24:    $\rho_{k+1} = (\sigma_{k+1}^{(1)})^2$, $\mathbf{x}_{k+1} = [\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{m'}]h$.
25: end for
```

and $\mathbf{V}_\ell$ to the empty matrix. At line 4, a preconditioner needs to be constructed such that $\mathbf{LDL}^T \approx \mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$ for μ equal to ρ_k or a fixed initial value. An algorithm based on RIF to compute an incomplete factor $\mathbf{L}$ implicitly from $\mathbf{C}$ will be discussed in the next section. As stated, different preconditioners may be used for different iteration steps, but for the efficiency reason, we usually use the same preconditioner. Line 8 implements the deflation and preconditioning techniques implicitly. The *for* loop at lines 7-22 constructs an orthonormal basis $\{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{m'}\}$ for the Krylov subspace and simultaneously an orthonormal basis $\{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{m'}\}$ such that $\mathbf{C}\mathbf{Z}_{m'} = \mathbf{Y}_{m'}\mathbf{G}_{m'}$, where $m' = \dim(\mathcal{K}_m((I - \mathbf{V}_\ell\mathbf{V}_\ell^T)\mathbf{L}^{-T}\mathbf{L}^{-1}(\mathbf{A} - \rho_k\mathbf{B}), \mathbf{x}_k)) - 1$. Then $\mathbf{G}_{m'} = \mathbf{Y}_{m'}^T\mathbf{C}\mathbf{Z}_{m'}$. Its smallest singular value and a corresponding right singular vector $\mathbf{h}$ are computed to construct a new approximate right singular vector at lines 23-24.

The process is theoretically equivalent to Algorithm 2.4 as applied to $\mathbf{A} = \mathbf{C}^T\mathbf{C}$ and $\mathbf{B} = \mathbf{I}$. When no preconditioning is used, i.e. $\mathbf{L} = \mathbf{I}$, the inverse free Krylov subspace method is simply the Lanczos method for $\mathbf{A}$ with restart after m iterations. When the preconditioning is used, we effectively transform the standard eigenvalue problem for $\mathbf{C}^T\mathbf{C}$ to the equivalent generalized eigenvalue problem for $(\hat{\mathbf{A}}, \hat{\mathbf{B}}) = (\mathbf{L}^{-1}\mathbf{C}^T\mathbf{C}\mathbf{L}^{-T}, \mathbf{L}^{-1}\mathbf{L}^{-T})$, to which the inverse free Krylov subspace method is applied.

In the `eigifp` implementation [62] of the inverse free preconditioned Krylov subspace method for the eigenvalue problem, an LOBPCG (locally optimal preconditioned conjugate gradient) [43, 41] type subspace enhancement was also included to further accelerate convergence. Note that, in the LOBPCG method, the steepest descent method is modified by adding the previous approximate eigenvector $\mathbf{x}_{k-1}$ to the space spanned by the current approximation and its residual span$\{\mathbf{x}_k, (\mathbf{A} - \rho_k\mathbf{B})\mathbf{x}_k\}$ to construct a new approximate eigenvector. It results in a conjugate gradient like algorithm that has a significant speedup in convergence over the steepest descent method. This idea has also been used in `eigifp` [62] by adding the previous approximate eigenvector $\mathbf{x}_{k-1}$ to the Krylov subspace $\mathcal{K}_m(\mathbf{A} - \rho_k\mathbf{B}, \mathbf{x}_k)$, which is also found to accelerate the convergence in many problems. As the extra cost of adding this vector is quite moderate, we also use this subspace enhancement in our implementation for the singular value problem. Algorithmically, we just need to add after the *for* loop at lines 7-22 to construct an additional basis vector $\mathbf{z}_{m'+1}$ by orthogonalizing $\mathbf{x}_k - \mathbf{x}_{k-1}$ against $\{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{m'}\}$. Note that we have used $\mathbf{x}_k - \mathbf{x}_{k-1}$ rather than $\mathbf{x}_{k-1}$ for orthogonalization because orthogonalizing $\mathbf{x}_{k-1}$ against $\mathbf{z}_0 = \mathbf{x}_k$ typically leads to cancelation when $\mathbf{x}_k \approx \mathbf{x}_{k-1}$. On the other hand, we can avoid possible cancelation in $\mathbf{x}_k - \mathbf{x}_{k-1}$ by computing its orthogonalization against $\mathbf{z}_0 = \mathbf{x}_k$ implicitly through

$$d = \tilde{\mathbf{Z}}_{m'} \begin{pmatrix} -\frac{\hat{\mathbf{h}}^T\hat{\mathbf{h}}}{h_1} \\ \hat{\mathbf{h}} \end{pmatrix}, \quad \text{where} \quad h = \begin{array}{c} \begin{pmatrix} h_1 \\ \hat{\mathbf{h}} \end{pmatrix} \end{array} \begin{array}{c} 1 \\ \\ m' \end{array}$$

is the unit right singular vector of the projection matrix $\mathbf{G}_m$ and $\tilde{\mathbf{Z}}_{m'}$ is the matrix of the basis vectors at line 19 of previous step (step $k-1$) of Algorithm 3.1, i.e. $\mathbf{x}_k = \tilde{\mathbf{Z}}_{m'}\mathbf{h}$ and $\mathbf{x}_{k-1} = \tilde{\mathbf{Z}}_{m'}\mathbf{e}_1$, where $\mathbf{e}_1 = [1, 0, \cdots, 0]^T$. It is easy to check that

$\mathbf{d} = \tilde{\mathbf{Z}}_{m'}\mathbf{h} - \frac{1}{\mathbf{h}_1}\tilde{\mathbf{Z}}_{m'}\mathbf{e}_1 = \mathbf{x}_k - \frac{1}{h_1}\mathbf{x}_{k-1} = \frac{1}{h_1}(\mathbf{x}_k - \mathbf{x}_{k-1}) - \frac{h_1-1}{h_1}\mathbf{x}_k$ and $\mathbf{x}_k^T\mathbf{d} = \mathbf{0}$. Therefore, a new basis vector that extends the subspace with $\mathbf{x}_k - \mathbf{x}_{k-1}$ can be obtained by orthogonalizing d against $\{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{m'}\}$ as

$$f_{m'+1,m'+1}\mathbf{z}_{m'+1} = \mathbf{d} - f_{0,m'+1}\mathbf{z}_0 - f_{1,m'+1}\mathbf{z}_1 - \cdots - f_{m',m'+1}\mathbf{z}_{m'}$$

Moreover, $\mathbf{Cd} = \mathbf{CZ}_{m'}[-\frac{\hat{\mathbf{h}}^T\hat{\mathbf{h}}}{h_1}, \hat{\mathbf{h}}]^T$ and then $\mathbf{Cz}_{m'+1}$ can be computed without the explicit multiplication by $\mathbf{C}$ from

$$\mathbf{Cz}_{m'+1} = \frac{\mathbf{Cd} - f_{0,m'+1}\mathbf{Cz}_0 - f_{1,m'+1}\mathbf{Cz}_1 - \cdots - f_{m',m'+1}\mathbf{Cz}_{m'}}{f_{m'+1,m'+1}},$$

from which $\mathbf{y}_{m'+1}$ and an additional column of $\mathbf{G}$ are computed as in (3.8). However, with possible cancellations in the last formula, $\mathbf{Cz}_{m'+1}$ may be computed with large errors and we suggest to compute $\mathbf{Cz}_{m'+1}$ explicitly when high accuracy is needed.

The algorithm we have presented computes approximate singular values and simultaneously computes the corresponding right singular vectors only. In applications where singular triplets are required, we can compute approximate left singular vectors from the right singular vectors obtained. This is a limitation of the $\mathbf{C}^T\mathbf{C}$ formulation (3.1) and Algorithm 3.1 where we reduce the eigenvalue residual of the approximate singular value and right singular vector pair $(\sigma_k^{(1)}, \mathbf{x}_k)$ (with $\|\mathbf{x}_k\| = 1$)

$$r_p := \|\mathbf{C}^T\mathbf{C}\mathbf{x}_k - (\sigma_k^{(1)})^2\mathbf{x}_k\|. \tag{3.9}$$

From this residual, the errors of approximate singular value $\sigma_k^{(1)}$ and approximate right singular vector $\mathbf{x}_k$ can be bounded as (see [19, p.205])

$$|\sigma_k^{(1)} - \sigma_1| \leq \frac{r_p^2}{(\sigma_k^{(1)} + \sigma_1)\text{gap}} \quad \text{and} \quad \sin\angle(\mathbf{x}_k, \mathbf{v}_1) \leq \frac{r_p}{\text{gap}}$$

where we assume that σ_1 is the singular value closest to $\sigma_k^{(1)}$, $\mathbf{v}_1$ is a corresponding right singular vector, and $\text{gap} = \min_{i\neq 1}|\sigma_k^{(1)} - \sigma_i|$. When a corresponding left singular vector is needed, it can be obtained as

$$\mathbf{w}_k = \mathbf{Cx}_k/\sigma_k^{(1)} \tag{3.10}$$

provided $\sigma_k^{(1)} \neq 0$. Then the accuracy of the approximate singular triplet $(\sigma_k^{(1)}, \mathbf{w}_k, \mathbf{x}_k)$ can be assessed by

$$\mathbf{r}_t := \left\| \begin{pmatrix} \mathbf{C}\mathbf{x}_k - \sigma_k^{(1)}\mathbf{w}_k \\ \mathbf{C}^T\mathbf{w}_k - \sigma_k^{(1)}\mathbf{x}_k \end{pmatrix} \right\| = \left\| M \begin{pmatrix} \mathbf{w}_k \\ \mathbf{x}_k \end{pmatrix} - \sigma_k^{(1)} \begin{pmatrix} \mathbf{w}_k \\ \mathbf{x}_k \end{pmatrix} \right\|. \tag{3.11}$$

It is easily checked that

$$\mathbf{r}_t = r_p/\sigma_k^{(1)}. \tag{3.12}$$

Therefore, for a tiny singular value, a small residual r_p for the pair $(\sigma_k^{(1)}, \mathbf{x}_k)$ does not imply a small residual $\mathbf{r}_t$ for the singular triplet $(\sigma_k^{(1)}, \mathbf{w}_k, \mathbf{x}_k)$. Indeed, the constructed left singular vector $\mathbf{w}_k$ may not be a good approximation, even when $\mathbf{x}_k$ is a good approximate right singular vector as indicated by r_p. This appears to be an intrinsic difficulty of the $\mathbf{C}^T\mathbf{C}$ formulation (3.1). Specifically, in the extreme case of $\sigma_1 = 0$, a corresponding left singular vector is any vector in the orthogonal complement of $\mathcal{R}(\mathbf{C})$ (the range space of $\mathbf{C}$) and it can not be obtained from multiplying a right singular vector by $\mathbf{C}$ or any vector in the subspace $\mathbf{C}\mathcal{K}_m(\mathbf{C}^T\mathbf{C}, \mathbf{x}_k) = \mathcal{K}_m(\mathbf{C}\mathbf{C}^T, C\mathbf{x}_k) \subset \mathcal{R}(\mathbf{C})$. In this case, we need to consider $\mathbf{C}\mathbf{C}^T$ on a new random initial vector to compute a left singular vector.

An alternative formulation of the left singular vector is by computing the left singular vector $\mathbf{g}$ of the projection matrix $\mathbf{G}_m$ at line 19 of Algorithm 3.1 and then form

$$\mathbf{w}_k = [\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{m'}]\mathbf{g}. \tag{3.13}$$

It is easy to check that this is theoretically equivalent to (3.10) if $\sigma_k^{(1)} \neq 0$. However, an advantage of this formulation is that it is still defined even when $\sigma_k^{(1)} = 0$, although the quality of the approximation is not assured. Our numerical experiments indicate that (3.13) is generally similar to (3.10) but may lead to a slightly better left singular vectors in some problems. In our implementation, we use (3.12) to estimate the residual of singular triplet to avoid the cost of computing the residual of singular triplets, but at termination, we use (3.13) to compute a left singular vector and then recompute its residual.

In the algorithm, we have used r_p for the convergence test unless a left singular vector is required. When a left singular vector is indeed needed, $\mathbf{r}_t$ is used for the convergence test. As discussed above, however, since the $\mathbf{r}_t$ may never converge if $\sigma_k^{(1)}$ is extremely small. Therefore, in order to properly terminate the iteration in such a situation, we propose to monitor the magnitude of the singular value computed and, when an extremely small singular value (i.e in the order of the machine precision) is detected, the stopping criterion should be switched to using r_p. By properly terminating the iteration using r_p, we can still obtain a sufficiently good approximate singular value and a right singular vector. After that, we can then separately apply the algorithm to $\mathbf{C}^T$ to compute a left singular vector.

We present the subspace enhancement steps and termination criteria discussed above as additional steps of Algorithm 3.1. It also includes the additional step needed in computing the left singular vector when it is required.

In Algorithm 3.2, lines 4-16 expand the subspace with $\mathbf{x}_k - \mathbf{x}_{k-1}$ using the method mentioned earlier. Line 18 computes the orthogonalization of $\mathbf{x}_{k+1} - \mathbf{x}_k$ against $\mathbf{x}_{k+1}$ to be used in the next iteration. The algorithm, by default, computes the singular value and the right singular vectors. If singular triplets are desired, Lines 21-23 compute an appropriate residual to be used for testing convergence. This is only for the purpose of terminating the iteration. At convergence, however, we compute the left singular vector $\mathbf{w}_{k+1}$ and the residual of the singular triplets explicitly.

Finally, we mention that the algorithm can be adapted trivially to compute its largest singular value. Namely, to compute the largest singular values of $\mathbf{C}$, we just need to modify line 23 in Algorithm 3.1 and line 17 in Algorithm 3.2 to compute the largest singular value of $\mathbf{G}_m$ and a corresponding right singular vector and the rest of the algorithm remains the same. It is easy to see that the convergence theory of [32] extends to this case. We also note that the above algorithm is based on vector iteration for computing a single singular value. A block matrix iteration version of the inverse free preconditioned Krylov subspace method has been developed in [70] to compute multiple eigenvalues or extremely clustered eigenvalues. It can be adapted as in Algorithm 3.1 to the task of computing multiple or extremely clustered singular

Algorithm 3.2 Inverse free preconditioned Krylov subspace method for SVD with LOBPCG enhancement

```
Same as Line 1-2 in Algorithm 3.1
for $k = 0, 1, 2, \ldots$ until convergence do
    Same as Line 4-22 in Algorithm 3.1
    $\mathbf{z}_{m'+1} = d_k; \mathbf{w} = Cd_k;$
    for $j = 0 : m'$ do
        $\mathbf{z}_{m'+1} = \mathbf{z}_{m'+1} - (\mathbf{z}_j^T \mathbf{z}_{m'+1})\mathbf{z}_j;$
        $\mathbf{w} = \mathbf{w} - (\mathbf{z}_j^T \mathbf{z}_{m'+1})\mathbf{C}\mathbf{z}_j;$
    end for
    if $\|\mathbf{z}_{m'+1}\| \neq 0$ then
        $\mathbf{z}_{m'+1} = \mathbf{z}_{m'+1}/\|\mathbf{z}_{m'+1}\|, \mathbf{w} = \mathbf{w}/\|\mathbf{z}_{m'+1}\|; \mathbf{y}_{m'+1} = \mathbf{w};$
        for $j = 0 : m'$ do
            $g_{j,m'+1} = \mathbf{y}_j^T \mathbf{y}_{m'+1}$ and $\mathbf{y}_{m'+1} = \mathbf{y}_{m'+1} - g_{j,m'+1}\mathbf{y}_j;$
        end for
        $g_{m'+1,m'+1} = \|\mathbf{y}_{m'+1}\|$ and $\mathbf{y}_{m'+1} = \mathbf{y}_{m'+1}/g_{m'+1,m'+1};$
        $m' = m' + 1;$
    end if
    Compute the smallest singular value $\sigma_{k+1}^{(1)}$ of $\mathbf{G}_m = [g_{ij}]_{i,j=0}^{m'}$ and a corresponding unit right singular vector $\mathbf{h}$;
    $\mathbf{d}_{k+1} = \mathbf{Z}_{m'}(\mathbf{h} - \mathbf{e}_1/h_1); Cd_{k+1} = \mathbf{C}\mathbf{Z}_{m'}(\mathbf{h} - \mathbf{e}_1/h_1);$
    $\rho_{k+1} = (\sigma_{k+1}^{(1)})^2$, $\mathbf{x}_{k+1} = \mathbf{Z}_{m'}\mathbf{h};$
    $res = \|\mathbf{C}^T\mathbf{C}\mathbf{x}_{k+1} - \rho_{k+1}\mathbf{x}_{k+1}\|;$
    if *singular triplet* is desired and $\sigma_{k+1}^{(1)} > \mathbf{u}\|\mathbf{C}\|^2$ then
        $res = res/\sigma_{k+1}^{(1)};$
    end if
    Test convergence using $res$;
end for
if *singular triplet* is desired then;
    Compute a left singular vector $\mathbf{g}$ of $\mathbf{G}_m$ and $\mathbf{w}_{k+1} = [\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{m'}]\mathbf{g};$
end if
Output: $(\sigma_{k+1}^{(1)}, \mathbf{x}_{k+1})$ or, if *singular triplet* is required, $(\sigma_{k+1}^{(1)}, \mathbf{w}_{k+1}, \mathbf{x}_{k+1})$.
```

values. Here, we omit a formal statement of the algorithm; see [70].

3.3.2 Preconditioning by robust incomplete factorizations (RIF)

In this section, we discuss how to construct a preconditioner $\mathbf{L}$, i.e. an approximate LDL^T factorization $\mathbf{C}^T\mathbf{C}-\mu\mathbf{I} = LDL^T$ where $\sqrt{\mu}$ is an approximation of the singular value to be computed and D is a diagonal matrix of 0 or ±1. This generally requires forming the matrix $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$, which may be much denser than $\mathbf{C}$ and hence leads to a denser $\mathbf{L}$. In addition, forming the matrix is associated with potential loss of information in very ill-conditioned cases, although this appears not to pose a problem when only an approximate factorization is sought [37].

For computing the smallest singular value, $\mu = 0$ is a natural first choice for the shift. In this case, we need an incomplete factorization of a symmetric positive semidefinite matrix, for which numerous techniques have been developed, see [8] for a survey. Indeed, if $\mu = 0$, the problem is the same as constructing a preconditioner for the linear least squares problem. One method that has been well studied is the incomplete QR factorization; see [7, 28, 66, 64, 86]. The incomplete QR factorization methods, such as incomplete modified Gram-Schmidt method or incomplete Givens rotation method, can be used here to construct a preconditioner for computing smallest singular values that are close to 0. While they are effective and often result in much faster convergence, they tend to have high intermediate storage requirements in our experiences; see [11] as well. Moreover, they can not deal with the cases of $\mu \neq 0$. On the other hand, Benzi and Tuma propose a method for constructing a preconditioner for $\mathbf{C}^T\mathbf{C}$ in [11] called robust incomplete factorization (RIF). This method can be easily adapted to computing an incomplete LDL^T factorization for $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$ and is found to have more moderate fill-ins. We discuss now the RIF preconditioner for the SVD algorithm.

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a sparse symmetric positive definite matrix. The idea of RIF is to obtain the factorization $\mathbf{A} = \mathbf{L}^T\mathbf{D}\mathbf{L}$ by applying an $\mathbf{A}$-orthogonalization process to the unit basis vectors $\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n$ (i.e. $\mathbf{I} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n]$). It will become a Gram-Schmidt process for the unit basis vectors with respect to the inner product

$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{A}} := \mathbf{x}^T \mathbf{A} \mathbf{y}$, i.e., for $i = 1, 2, \cdots, n$,

$$\mathbf{z}_i = \mathbf{e}_i - \sum_{j=1}^{i-1} \frac{\langle \mathbf{e}_i, \mathbf{z}_j \rangle_{\mathbf{A}}}{\langle \mathbf{z}_j, \mathbf{z}_j \rangle_{\mathbf{A}}} \mathbf{z}_j. \tag{3.14}$$

This is the classical Gram-Schmidt (CGS) process. The corresponding modified Gram-Schmidt (MGS) process can be implemented as updating the basis vector $\mathbf{z}_i$ initialized as $\mathbf{z}_i = \mathbf{e}_i$ $(1 \le i \le n)$ by the following nested loop: for $j = 1, 2, \cdots, n$, orthogonalize each $\mathbf{z}_i$ (for $i = j+1, \cdots, n$) against $\mathbf{z}_j$ by

$$\mathbf{z}_i = \mathbf{z}_i - \frac{\langle \mathbf{z}_i, \mathbf{z}_j \rangle_{\mathbf{A}}}{\langle \mathbf{z}_j, \mathbf{z}_j \rangle_{\mathbf{A}}} \mathbf{z}_j. \tag{3.15}$$

This updating process allows discarding $\mathbf{z}_j$ to free the memory once it is orthogonalized against all $\mathbf{z}_i$ (for $i = j+1, \cdots, n$). Let

$$l_{ij} = \frac{\langle \mathbf{z}_i, \mathbf{z}_j \rangle_{\mathbf{A}}}{\langle \mathbf{z}_j, \mathbf{z}_j \rangle_{\mathbf{A}}}, \quad \text{if} \quad i \ge j;$$

and set $l_{ij} = 0$ if $i < j$. Then $\mathbf{L} = [l_{ij}]$ is a unit lower triangular matrix and this process results in an $\mathbf{A}$-orthogonal matrix $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n]$ such that $\mathbf{I} = \mathbf{Z}\mathbf{L}^T$. Then $\mathbf{Z}^T \mathbf{A} \mathbf{Z} = \mathbf{D}$ implies $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ where $\mathbf{D} = \text{diag}[d_1, d_2, \cdots, d_n]$ and $d_j = \langle \mathbf{z}_j, \mathbf{z}_j \rangle_{\mathbf{A}}$.

Clearly, by (3.14), $\mathbf{z}_i \in \text{span}\{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_i\}$ and $\mathbf{Z}$ is upper triangular. Since CGS (3.14) and MGS (3.15) are theoretically equivalent, (3.15) can be formulated as

$$\mathbf{z}_i = \mathbf{z}_i - l_{ij} \mathbf{z}_j, \quad \text{with} \quad l_{ij} = \frac{\langle \mathbf{e}_i, \mathbf{z}_j \rangle_{\mathbf{A}}}{\langle \mathbf{z}_j, \mathbf{z}_j \rangle_{\mathbf{A}}} \tag{3.16}$$

which is computationally more efficient (see [9]) for problem like $\mathbf{A} = \mathbf{C}^T \mathbf{C}$ below. In addition, as $\mathbf{A}$ is sparse, $\langle \mathbf{e}_i, \mathbf{z}_j \rangle_{\mathbf{A}} = \mathbf{e}_i^T \mathbf{A} \mathbf{z}_j$ may be structurally zero for many i, j, resulting a sparse $\mathbf{L}$. The A-orthogonalization process can efficiently exploit the property $l_{ij} = 0$ by skipping the corresponding orthogonalization step. Furthermore, one may also drop the entry l_{ij} and skip the orthogonalization if l_{ij} is sufficiently small. This would result in an incomplete factorization called robust incomplete factorization (RIF).

RIF has also been used in [11] to efficiently construct preconditioners for $\mathbf{C}^T \mathbf{C}$ for a full rank matrix $\mathbf{C} \in \mathbb{R}^{m \times n}$ arising from the normal equation for the least squares problem. An advantage of RIF for $\mathbf{C}^T\mathbf{C}$ is that the $\mathbf{C}^T\mathbf{C}$-orthogonalization process can be carried out using $\mathbf{C}$ only as

$$\mathbf{z}_i = \mathbf{z}_i - l_{ij}\mathbf{z}_j, \quad \text{with} \;\; l_{ij} = \frac{\langle \mathbf{C}\mathbf{z}_i, \mathbf{C}\mathbf{z}_j \rangle}{\langle \mathbf{C}\mathbf{z}_j, \mathbf{C}\mathbf{z}_j \rangle} \tag{3.17}$$

for $j = 1, 2, \cdots, n$ and $i = j + 1, \cdots, n$, where $\langle \cdot, \cdot \rangle$ is the Euclidian inner product. In this setting, the following CGS formulation of l_{ij}

$$\mathbf{z}_i = \mathbf{z}_i - l_{ij}\mathbf{z}_j, \quad \text{with} \;\; l_{ij} = \frac{\langle \mathbf{C}\mathbf{e}_i, \mathbf{C}\mathbf{z}_j \rangle}{\langle \mathbf{C}\mathbf{z}_j, \mathbf{C}\mathbf{z}_j \rangle}$$

is preferred over the MGS formulation because of the need to compute $\mathbf{C}\mathbf{z}_i$ in MGS (3.17) each time $\mathbf{z}_i$ is updated, whereas only $\mathbf{C}\mathbf{e}_i$ (the i-th column of $\mathbf{C}$) is needed in CGS. Since we are only interested in an incomplete factorization by applying dropping threshold in $\mathbf{z}_i$ and l_{ij}, the difference in stability between CGS and MGS is not significant. Also, computation of l_{ij} requires forming $\mathbf{C}\mathbf{z}_j$ once for each $\mathbf{z}_j$, which involves sparse-sparse matrix-vector multiplications and can be efficiently computed as a linear combination of a few columns of $\mathbf{C}$; see [11]. We also observe that the inner products in l_{ij} involve two sparse vectors as well.

If we multiply both sides of (3.17) by $\mathbf{C}$, it is possible to get around computing $\mathbf{w}_i := \mathbf{C}\mathbf{z}_i$ as a matrix-vector multiplication in MGS (3.17) by computing it through the updating formula

$$\mathbf{w}_i = \mathbf{w}_i - l_{ij}\mathbf{w}_j, \quad \text{with} \;\; l_{ij} = \frac{\langle \mathbf{w}_i, \mathbf{w}_j \rangle}{\langle \mathbf{w}_j, \mathbf{w}_j \rangle}, \tag{3.18}$$

which maintains the MGS form. However, since the $\mathbf{L}$ matrix is all we need, it is not necessary in this to compute $\mathbf{z}_i$ anymore. Indeed, since $\mathbf{w}_i$ is initialized as $\mathbf{C}\mathbf{e}_i$, (3.18) is just the modified Gram-Schmidt process in the Euclidian inner product applied to the columns of $\mathbf{C}$ and it becomes the MGS method for the QR factorization of $\mathbf{C}$. However, with $\mathbf{w}_i$ initialized as $\mathbf{C}\mathbf{e}_i$ and $\mathbf{z}_i$ initialized as $\mathbf{e}_i$, the sequence $\mathbf{w}_i$ generated may be expected to be much denser than the corresponding $\mathbf{z}_i$, which appears to be the case in our experiments. This may be the main motivation of using the A-orthogonalization in RIF.

We observe that the same process can be extended to our problem of constructing an LDL^T factorization for $\mathbf{A} := \mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$ with a shift $\mu \approx \sigma_1^2$. The corresponding orthogonalization process is

$$\mathbf{z}_i = \mathbf{z}_i - l_{ij}\mathbf{z}_j, \quad \text{with } \ l_{ij} = \frac{\langle \mathbf{C}\mathbf{e}_i, \mathbf{C}\mathbf{z}_j\rangle - \mu\langle \mathbf{e}_i, \mathbf{z}_j\rangle}{\langle \mathbf{C}\mathbf{z}_j, \mathbf{C}\mathbf{z}_j\rangle - \mu\langle \mathbf{z}_j, \mathbf{z}_j\rangle} \tag{3.19}$$

for $j = 1, 2, \cdots, n$ and $i = j+1, \cdots, n$. Now, if $\mu < \sigma_1^2$, then $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$ is positive definite and, with the divisor in l_{ij} nonzero, the process is well defined.

If $\mu = \sigma_1^2$, then $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$ is positive semidefinite and the process may encounter a zero division if $\langle \mathbf{C}\mathbf{z}_j, \mathbf{C}\mathbf{z}_j\rangle - \mu\langle \mathbf{z}_j, \mathbf{z}_j\rangle = 0$ for some j. However, in this case, $(\mathbf{C}^T\mathbf{C} - \mu\mathbf{I})\mathbf{z}_j = 0$ and then $\langle \mathbf{C}\mathbf{z}_i, \mathbf{C}\mathbf{z}_j\rangle - \mu\langle \mathbf{z}_i, \mathbf{z}_j\rangle = 0$ for any i. Then we do not need to carry out the orthogonalization against $\mathbf{z}_j$. Continuing the process, we still obtain $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$ such that $\langle \mathbf{C}\mathbf{z}_j, \mathbf{C}\mathbf{z}_i\rangle - \mu\langle \mathbf{z}_j, \mathbf{z}_i\rangle = 0$ but $\mathbf{Z}^T\mathbf{A}\mathbf{Z} = \mathbf{D}$ will have zero in diagonals. However, this does not cause any problem as we still have $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ and by using a scaled $\mathbf{L}$, we have $\mathbf{D}$ with 0 and 1 as diagonal elements. This is precisely the factorization needed.

If $\mu > \sigma_1^2$, then $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$ is indefinite and the process may breakdown with the occurrence of $\langle \mathbf{C}\mathbf{z}_j, \mathbf{C}\mathbf{z}_j\rangle - \mu\langle \mathbf{z}_j, \mathbf{z}_j\rangle = 0$ but $(\mathbf{C}^T\mathbf{C} - \mu\mathbf{I})\mathbf{z}_j \neq 0$ for some j. In practice, the exact breakdown is unlikely but we may encounter a near breakdown (with $\langle \mathbf{C}\mathbf{z}_j, \mathbf{C}\mathbf{z}_j\rangle - \mu\langle \mathbf{z}_j, \mathbf{z}_j\rangle \approx 0$), which may cause instability to the process. However, since we are only interested in an incomplete factorization which incur a perturbation through dropping small elements, we propose to modify the pivot by simply setting $\langle \mathbf{C}\mathbf{z}_j, \mathbf{C}\mathbf{z}_j\rangle - \mu\langle \mathbf{z}_j, \mathbf{z}_j\rangle$ to some nonzero scalar such as the dropping threshold and skip the orthogonalization against $\mathbf{z}_j$. This perturbation is consistent with the dropping strategy in the incomplete factorization and would amount to a perturbation to $\mathbf{z}_j$ in the order of magnitude of the dropping threshold. In any case, it only affects the quality of the preconditioner and hence efficiency of the overall algorithm, but it does not reduce the accuracy of the singular value computed by our method. In our experiences, the algorithm handles modest indefiniteness very well, but the quality of preconditioner deteriorates as the matrix indefiniteness increases.

The incomplete LDL^T factorization provided by RIF need to be scaled so that $\mathbf{D}$ has diagonals equal to $0, \pm 1$ for its use as a preconditioner for the singular value problem. This can be achieved by multiplying $\mathbf{L}$ by $\mathbf{D}^{1/2}$ on the right. The following is the RIF algorithm as adapted from [11] with the columns of $\mathbf{L}$ scaled.

Algorithm 3.3 Robust Incomplete Factorization of $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$

```
 1: **Input:** $\eta_1$ (drop threshold for $\mathbf{L}$) and $\eta_2$ (drop threshold for $\mathbf{Z}$);
 2: **initialize:** $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n] = \mathbf{I}$; $\mathbf{L} = [l_{ij}] = \mathbf{I} \in \mathbb{R}^{n \times n}$;
 3: **for** $j = 1$ to $n$ **do**
 4:     $d_j = \langle \mathbf{Cz}_j, \mathbf{Cz}_j \rangle - \mu \langle \mathbf{z}_j, \mathbf{z}_j \rangle$;
 5:     $l_{jj} = \sqrt{|d_j|}$;
 6:     **if** $l_{jj} > \max\{\eta_1 \|\mathbf{Ce}_j\|_1, \mathbf{u}\}$ **then**
 7:         **for** $i = j + 1$ to $n$ **do**
 8:             $p_{ij} = \langle \mathbf{Cz}_j, \mathbf{Ce}_i \rangle - \mu \langle \mathbf{z}_j, \mathbf{e}_i \rangle$;
 9:             **if** $|p_{ij}|/l_{jj} \geq \max\{\eta_1 \|\mathbf{Ce}_j\|_1, \mathbf{u}\}$ **then**
10:                 $\mathbf{z}_i = \mathbf{z}_i - \frac{p_{ij}}{d_j}\mathbf{z}_j$ and $l_{ij} = \text{sgn}(p_{jj}) \cdot p_{ij}/l_{jj}$;
11:                 **if** $|\mathbf{z}_i(\ell)| < \eta_2 \|\mathbf{z}_i\|_1$ for any $\ell$ **then**
12:                     set $\mathbf{z}_i(\ell) = 0$;
13:                 **end if**
14:             **end if**
15:         **end for**
16:     **else**
17:         $l_{jj} = \max\{\eta_1 \|\mathbf{Ce}_j\|, \mathbf{u}\}$;
18:     **end if**
19: **end for**
```

We present some remarks concerning Algorithm 3.3. At line 6, we test the divisor l_{jj} for near-breakdown. If a near-breakdown occurs, we set l_{jj} to the breakdown threshold $\max\{\eta_1 \|\mathbf{Ce}_j\|_1, \mathbf{u}\}$ at line 17 and skip the orthogonalization process. Here, we note that the threshold is chosen to be relative to the norm of $\mathbf{Ce}_j$ as $\mathbf{Cz}_j$ is constructed from it through orthogonalization and $\mathbf{u}$ is added to the definition of the threshold to deal with possible situation of $\mathbf{Ce}_j = 0$. We skip the orthogonalization of $\mathbf{z}_i$ if the l_{ij} is below the given threshold $\max\{\eta_1 \|\mathbf{Ce}_j\|_1, \mathbf{u}\}$. In that case, l_{ij} is set to 0. To further improve efficiency of the algorithm, we also apply a drop rule to $\mathbf{z}_i$ at Line 11 by setting all entries of $\mathbf{z}_i$ that are below the threshold $\eta_2 \|\mathbf{z}_i\|_1$ to 0. This will maintain $\mathbf{Z}$ as sparse as possible and improve the efficiency of the algorithm. In our experiments, the quality of the preconditioner constructed appears to depend more on the magnitude of η_2 than that of η_1. So η_2 is chosen to be much smaller than η_1. In our implementation, we set $\eta_1 = 10^{-3}$ and $\eta_2 = 10^{-8}$ as the default values. Finally, on output, the algorithm produces an approximate factorization $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I} \approx \mathbf{LDL}^T$

with $\mathbf{D}$ having only $0, \pm 1$ as diagonals.

3.3.3 A robust implementation

One advantage of the inverse free preconditioned Krylov subspace method is its relative simplicity for implementation with the number of inner iterations being the only parameter to select. We have implemented Algorithm 3.2 in combination with the RIF preconditioner (Algorithm 3.3) in a black-box MATLAB implementation for the singular value problem. The program called `svdifp` is used in our numerical tests.

Our program `svdifp` is based on the MATLAB program `eigifp` [62] which implements the inverse free preconditioned Krylov subspace method with several algorithmic enhancements for the generalized eigenvalue problem. We have incorporated many features of `eigifp` into our implementation but the core iteration is to construct the projection of $\mathbf{C}$ as outlined in Algorithm 3.2. Noting that for Algorithm 3.2, the only required user input is m (the inner iteration) and a preconditioner, we have adopted the same strategy used in `eigifp` in determining m; see [62].

Namely, m can be either specified by the user or, by default, adaptively determined by the program according to its effect on the rate of convergence. Note that experiments have shown that an optimal value of m is larger if the problem is more difficult while it is smaller if the problem is easier (e.g. with a good preconditioner). On the other hand, to determine a preconditioner, we first need an approximate singular value as a shift for the RIF preconditioner. Here different strategies will be used depending on whether computing the largest or the smallest singular values.

For computing the smallest singular value, we assume 0 is a good initial approximate singular value and, using 0 as the shift, we compute a preconditioner by Algorithm 3.3 and carry out a preconditioned iteration.

For computing the largest singular value, the standard Lanczos bidiagonalization algorithm [29] should work well because the spectral separation is typically increased by 2 times through the $\mathbf{C}^T\mathbf{C}$ formulation (3.1), i.e.

$$\frac{\sigma_n^2 - \sigma_{n-1}^2}{\sigma_{n-1}^2 - \sigma_1^2} = \frac{\sigma_n - \sigma_{n-1}}{\sigma_{n-1} - \sigma_1}\frac{\sigma_n + \sigma_{n-1}}{\sigma_{n-1} + \sigma_1} \approx 2\frac{\sigma_n - \sigma_{n-1}}{\sigma_{n-1} - \sigma_1}.$$

However, for problems with clustered largest singular value, the preconditioning approach can still be very beneficial. One difficulty then is that there is no good approximate singular value readily available initially and no preconditioner can be derived. Following the strategy in `eigifp` [62], we start the iteration with no preconditioning and, when a sufficiently good approximate singular value σ has been found as determined by the residual, we compute a preconditioner for $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$ by Algorithm 3.3 with the shift $\mu = \sigma^2 + r_p$ and then continue the iteration with preconditioning where r_p is the residual and hence μ is an upper bound for the true singular value. This typically leads to accelerated convergence.

In both cases, the program monitors the approximate singular value obtained and the convergence rate and may update the preconditioner using an updated approximate singular value as the shift if significant deviation of the singular value from the shift is detected. The same strategy is followed when computing several singular values with deflation. The program can be run with no required user input. However, it also allows various optional parameters the user may supply to improve the performance. They include the inner iteration m, the RIF thresholds, initial approximate singular value (which can be used to compute a preconditioner) or a preconditioner itself, among others.

3.4 Numerical examples

In this section, we present some numerical examples to demonstrate the capability and efficiency of the preconditioned inverse free Krylov subspace method for singular value problem. We shall compare our MATLAB implementation `svdifp` with several existing programs (i.e. `irlba` of Baglama and Reichel [5], `jdsvd` of Hochstenbach [35, 36], `lansvd` of Larson [47], and `svds` of MATLAB which is based on the ARPACK [49] of Lehoucq, Sorenson and Yang). `irlba` [5] implements an augmented implicitly restarted Lanczos bi-diagonalization algorithm. `jdsvd` [35, 36] implements a Jacobi-

Davidson method on the augmented matrix formulation. (Note that a program based on the Jacobi-Davidson method for $\mathbf{C}^T\mathbf{C}$ has also been developed recently [37].) `lansvd` [47] implements the Lanczos bidiagonalization algorithm for $\mathbf{R}^{-1}$ from the QR factorization of $\mathbf{C} = \mathbf{QR}$ for computing the smallest singular value. `svds` of MATLAB implements ARPACK [49] and uses the inverse of $\mathbf{M}$ (or $\mathbf{M} - \mu\mathbf{I}$) in the formulation (3.2) for computing the smallest singular value. We note that `svdifp` and `jdsvd` compute one singular value at a time, while `irlba`, `lansvd`, and `svds` can compute several singular values simultaneously. On the other hand, `svdifp` and `jdsvd` can use preconditioners to accelerate convergence, while `irlba`, `lansvd`, and `svds` have to use the shift-and-invert approach.

In the first three examples, we test the programs on computing the smallest singular value, while in the fourth example, we demonstrate capability of `svdifp` in computing several largest singular values using deflation. All the executions were carried out using MATLAB version 8.0.0.783 from MathWorks on a PC with an Intel quad-core i7-2670QM @ 2.20GHz and 12 GB of RAM running Ubuntu Linux 12.04. The machine epsilon is $\mathbf{u} \approx 2.2 \cdot 10^{-16}$. The performance parameters we consider for comparisons are the residual of the approximate singular triplet obtained, the number of matrix-vector multiplications where applicable, and the CPU time. The CPU time is gathered with on-screen outputs suppressed. For the methods that require some factorization of the matrix, we also consider the number of non-zeros in the factors, which indicates the memory requirements and their potential limitations.

We first present an example that tests the capability of `svdifp` to compute tiny singular values accurately. We also show that applying `eigifp` directly to the eigenvalue problem for $\mathbf{C}^T\mathbf{C}$ may result in loss of accuracy for the computed singular value. Here, in using `eigifp`, the matrix-vector multiplication $\mathbf{C}^T\mathbf{C}\mathbf{x}$ is obtained by computing $\mathbf{C}\mathbf{x}$ first and then multiplying by $\mathbf{C}^T$. Even though $\mathbf{C}^T\mathbf{C}$ is not explicitly formed, the singular value is obtained from the projection of $\mathbf{C}^T\mathbf{C}$, potentially resulting in loss of accuracy; see the discussion in Section 3.3.1. Both methods are run without preconditioning and with the number of inner iteration set to 20.

Example 1. We consider the following matrix

$$\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \quad \text{with} \ \ \mathbf{\Sigma} = \begin{array}{c} \left(\begin{array}{c} \mathbf{D} \\ 0 \end{array} \right) \end{array} \begin{array}{c} n \\ m-n \end{array}$$

where $\mathbf{D} = \mathrm{diag}(1, 1/2^4, \cdots, 1/n^4)$ and $\mathbf{U}$ and $\mathbf{V}$ are random orthogonal matrices generated by `U=orth(rand(m,m))` and `V=orth(rand(n,n))` in MATLAB. We test and compare the accuracy of the smallest singular value computed by `svdifp` and `eigifp` with $n = 100$ and $m = 100$ or $m = 200$. In either case, the exact smallest singular value of $\mathbf{C}$ is $\sigma_1 = 10^{-8}$ and the second smallest singular value is approximately $1.041 \cdot 10^{-8}$. The convergence is tested using the criterion $\|\mathbf{C}^T\mathbf{C}\mathbf{v}_1 - \sigma_1^2\mathbf{v}_1\| < \eta\|\mathbf{C}\|^2$ and, to achieve the best accuracy possible, we use a very small threshold $\eta = 10^{-19}$ and run the iteration until the residual stagnates.

Table 3.1 lists the best smallest singular values and their residuals obtained. For `svdifp`, with $\kappa(\mathbf{C}) = 10^8$, the residual deceases to about 10^{-18} and σ_1 computed has relative error in the order of $10^{-10} \approx \mathbf{u}\kappa(\mathbf{C})$. This is the best accuracy one may expect from using a backward stable method. On the other hand, for `eigifp`, the residual decreases and then stagnates at around 10^{-16}. The relative error of the computed singular values oscillates around 10^{-4} and no better approximation can be obtained. The singular value computed by applying `eigifp` directly lost about 5 digits of accuracy in this case.

Table 3.1: Example 1: σ_1 - computed smallest singular value by `svdifp` and `eigifp`; Res - $\|\mathbf{C}^T\mathbf{C}\mathbf{v}_1 - \sigma_1^2\mathbf{v}_1\|$.

	$m = 100$		$m = 200$	
	σ_1	Res	σ_1	Res
svdifp	1.0000000008e-08	1e-20	1.00000000001e-08	2e-20
eigifp	1.0001e-08	8e-17	1.00008e-8	8e-17

It is interesting to observe that with a good preconditioning, `eigifp` appears to be able to compute σ_1 accurately. Note that this is a dense matrix and the default preconditioner constructed by `eigifp` is the (complete) LDL^T factorization.

Next, we test and compare `svdifp` with several existing programs for SVD on computing the smallest singular value for a set of test problems. The test matrices consist of both square and non-square matrices taken from the Matrix Market [61] and the University of Florida Sparse Matrix Collection [18]. They are listed in Table 3.2 together with some basic information of the matrices (the smallest singular values are computed by MATLAB's `svd(full(A))`).

Table 3.2: Test Matrices Used for Examples 2 and 3

No.	Matrix	Size	Non-zeros	σ_1	$\kappa(\mathbf{C})$	source
Square Matrix						
1	dw2048	2048×2048	10114	4.68e-4	2.03e3	Matrix Market
2	fidap004	1601×1601	31837	6.57e-4	2.39e3	Matrix Market
3	hor131	434×434	41832	1.53e-5	4.31e4	Matrix Market
4	jagmesh1	936×936	6264	5.63e-3	1.23e3	Matrix Market
5	lshp	3025×3025	20833	1.03e-4	6.78e4	Matrix Market
6	pde2961	2961×2961	14585	1.62e-2	6.42e2	Matrix Market
7	pores3	532×532	3474	2.67e-1	5.61e5	Matrix Market
8	sherman	1000×1000	3750	3.23e-4	1.56e4	Matrix Market
Rectangular Matrix						
9	well1033	1033×320	4372	1.09e-2	1.66e2	Matrix Market
10	well1850	1850×712	8755	1.61e-2	1.11e2	Matrix Market
11	lpi_cplex1	5224×3005	10947	6.39e-2	3.13e3	UFLSMC
12	qiulp	1900×1192	4492	7.57e-1	4.08e1	UFLSMC
13	ge	10099×16369	44825	1.08e-3	1.28e7	UFLSMC
14	p010	10099×19090	118000	1.50e-1	1.18e2	UFLSMC
15	lp_ganges	1309×1706	6937	1.87e-4	2.13e4	UFLSMC
16	cep1	1521×4769	8233	1.00e0	1.49e1	UFLSMC
17	gen2	1121×3264	81855	1.41e0	3.35e1	UFLSMC
18	Maragal_5	3320×4654	93091	7.11e-46	2.30e46	UFLSMC
19	lp_ship12s	1151×2869	8284	0	-	UFLSMC

Since these programs may have very different approaches and have different assumptions on computing resources, we shall carry out the testing in two different settings. We first consider in Example 2 programs that do not use any exact factorization for inverse, i.e. `svdifp`, `jdsvd` and `irlba`. Since `svdifp` and `jdsvd` can be implemented with or without preconditioning, we shall test them first with preconditioning and then test them without preconditioning together with `irlba`. In the second testing (Example 3), we consider `svds` and `lansvd`, where the LU factorization of $\mathbf{M}$ and the QR factorization of $\mathbf{C}$ are respectively computed for the shift-and-invert. To facilitate a comparison, we consider `svdifp` using the $\mathbf{R}$ factor from the QR factorization of $\mathbf{C}$ as a preconditioner. Namely, if a complete factorization is possible, svdifp may also take advantage of it by using a more effective preconditioner, although this is not the best way to use the program.

Table 3.3: Example 2: With preconditioning: CPU - CPU time; MV - # of matrix-vector multiplications; nnz - number of non-zeros of the preconditioner; Res - $\|[\mathbf{C}\mathbf{v}_1-\sigma_1\mathbf{u}_1;\mathbf{C}^T\mathbf{u}_1-\sigma_1\mathbf{v}_1]\|/\|\mathbf{C}\|_1$.

	svdifp				jdsvd			
No.	CPU	MV	nnz	Res	CPU	MV	nnz	Res
Square Matrix								
1	0.6	179	25564	9e-7	0.4	136	49019	2e-11
2	1.5	223	91593	1e-7	0.9	102	179673	2e-8
3	0.6	3545	15719	5e-7	0.1	148	11740	3e-10
4	0.4	289	33065	6e-7	0.7	146	67112	6e-10
5	7.3	1103	170276	8e-7	1.7	100	425650	6e-10
6	1.9	113	69291	3e-8	0.3	126	89000	2e-9
7	0.04	25	4870	3e-13	0.09	96	46461	3e-7
8	0.2	355	13695	3e-7	0.1	84	11630	2e-7
Rectangular Matrix								
9	0.03	91	2235	2e-10	2.8	750	59291	1e-7
10	0.08	69	6325	7e-8	9.6	426	312083	1e-7
11	0.4	69	8995	2e-7	9.0	320	49318	2e-7
12	0.2	91	13620	1e-8	1.2	350	94671	3e-7
13	10.4	91	110017	5e-7	1689	20052	141008	1e-4
14	13.1	157	138793	2e-7	474	438	11276604	1e-7
15	0.3	91	18573	9e-9	10.6	358	421304	2e-13
16	2.0	113	106822	3e-8	1.1	266	41793	6e-7
17	4.3	267	297609	9e-7	36023	36846	8055182	1e-3
18	28.0	24	997991	3e-2[1]	9002	3744	8666363	7e-7
19	0.08	24	6868	7e-2[2]	0.5	136	65642	4e-8

EXAMPLE 2. We consider the performance of svdifp, jdsvd and irlba in computing the smallest singular value of matrices in Table 3.2. For matrices with $m < n$, we consider their transposes instead. We set the initial vector for all three methods to be the same random vector generated by randn(n,1). We also select parameters in the three codes so that each method carries out about the same number of matrix-vector multiplications in each inner iteration. Specifically, for svdifp, we set the number of inner iteration m to 10. In jdsvd, the maximum number of steps of inner linear solver is set to 10, which is also its default value. We use the default settings of jdsvd for all other parameters. In particular, the refined extraction of Ritz vector is used throughout and the dimension of the search subspace varies between 10 and

20. In irlba, we set $k = 1$ (the number of desired singular values) and $adjust = 8$ (the number of initial vectors added to the k restart vectors to form an initial subspace). They are chosen so that the dimension of the initial subspace is consistent with its default choices: $k = 6$, $adjust = 3$. All other parameters in irlba are set to their default values. Then irlba applies 10 bidiagonalization steps after each restart. Based on these settings, all three methods carry out approximately 22 matrix-vector multiplications (by $\mathbf{C}$ or $\mathbf{C}^T$) in each outer iteration. We set the maximum number of outer iterations to 10000 for all and, unless stated otherwise, the stopping criterion is

$$\texttt{Res} := \|[\mathbf{C}\mathbf{v}_1 - \sigma_1\mathbf{u}_1; \mathbf{C}^T\mathbf{u}_1 - \sigma_1\mathbf{v}_1]\|/\|\mathbf{C}\|_1 < 10^{-6} \tag{3.20}$$

where $(\sigma_1, \mathbf{u}_1, \mathbf{v}_1)$ is the approximate singular triplet obtained at step k.

We first compare svdifp and jdsvd, both of which allow using preconditioning to accelerate convergence. In svdifp, the default RIF preconditioner is used, i.e. an incomplete factorization of $\mathbf{C}^T\mathbf{C}$ is constructed by Algorithm 3.3 with the default choices of thresholds $\eta_1 = 10^{-3}$ and $\eta_2 = 10^{-8}$. In jdsvd, a preconditioner is needed for solving a correction equation in the inner iteration and we use the routine create_prec_jdsvd.m that accompanies jdsvd to construct a preconditioner for $\mathbf{M}$. Specifically, for square matrices, we compute the ILU factorization of $\mathbf{C}$, from which a preconditioner for $\mathbf{M}$ is constructed. For non-square matrices, we compute the ILU factorization of $\mathbf{M}$ but because of singularity of $\mathbf{M}$, breakdown often occurs, in which case the ILU factorization of a shifted matrix $\mathbf{M} - \mu\mathbf{I}$ is used where $\mu = 2^p \cdot 10^{-2}\|\mathbf{M}\|_{\max}$ and p is the first non-negative integer that stops the breakdown. The dropping threshold for all ILU factorizations is 10^{-3}. In addition, jdsvd uses BiCGSTAB [20] as the inner linear solver when a preconditioner is present.

[1] For this matrix, $\sigma_1 = 7.11e - 46$ according to MATLAB's svd. Although $\texttt{Res} = 3e - 2$, the residual defined by $\|\mathbf{C}^T\mathbf{C}\mathbf{v}_1 - \sigma_1^2\mathbf{v}_1\|$ is 3e-24 while the computed singular value is 2e-25. The singular values returned by jdsvd for this matrix is 3e-5. Also note that 113 singular values of this matrix are smaller than the machine precision and the second smallest is 1.7e-31.

[2] For this matrix, $\sigma_1 = 0$ according to MATLAB's svd. Although $\texttt{Res} = 6e - 2$, the residual defined by $\|\mathbf{C}^T\mathbf{C}\mathbf{v}_1 - \sigma_1^2\mathbf{v}_1\|$ is 2e-25 while the computed singular value is 4e-27. The singular values returned by jdsvd for this matrix is 6e-7. Also note that 35 singular values are smaller than the machine precision. The second smallest singular value is 0 as well and the third one is 1.3e-18.

Table 3.3 presents the results of this test. In the table, nnz is the number of non-zeros in the preconditioner ($\mathbf{L}$ for svdifp and both $\mathbf{L}$ and $\mathbf{U}$ for jdsvd). In the MV column, we list the number of matrix-vector multiplications by either $\mathbf{C}$ or $\mathbf{C}^T$. Res is the relative residual of the approximate singular triplet (3.20).

Table 3.4: Example 2: without preconditioning. CPU - CPU time; MV - # of matrix-vector multiplications; Res - $\|[\mathbf{C}\mathbf{v}_1 - \sigma_1\mathbf{u}_1; \mathbf{C}^T\mathbf{u}_1 - \sigma_1\mathbf{v}_1]\|/\|\mathbf{C}\|_1$.

	svdifp			jdsvd			irlba		
No.	CPU	MV	Res	CPU	MV	Res	CPU	MV	Res
Square Matrix									
1	2.2	8033	1e-06	2.1	7542	9e-7	2.8	13856	9e-7
2	4.9	18901	1e-06	6.0	21830	1e-6	21.9	104496	8e-7
3	20.1	220002	5e-04	34.8	220038	1e-5	25.9	220018	2e-2
4	9.5	81227	1e-06	4.8	26308	9e-7	13.4	90350	1e-6
5	27.3	69457	1e-06	20.7	62476	1e-6	59.8	220018	3e-2
6	3.5	9023	1e-06	3.0	9280	1e-6	6.1	23668	9e-7
7	21.1	220002	2e-03	35.7	220038	2e-5	27.2	220018	2e-2
8	15.1	127185	9e-07	23.7	127134	1e-6	32.6	220018	3e-2
Rectangular Matrix									
9	1.2	7153	1e-06	0.4	2284	6e-7	0.2	1206	7e-8
10	0.5	2467	8e-07	0.6	2262	1e-6	0.3	1888	8e-8
11	0.7	1697	1e-06	0.4	1074	6e-7	0.2	634	2e-7
12	0.3	1257	1e-06	0.7	2900	1e-6	0.2	1228	1e-7
13	500	220002	1e-03	189	220038	3e-5	167	220018	2e-2
14	18.4	6669	1e-06	2.0	1866	1e-6	2.8	2856	6e-8
15	0.1	553	1e-06	0.3	1008	1e-6	0.09	480	9e-8
16	0.1	245	2e-07	0.08	238	1e-7	0.02	62	9e-9
17	0.9	2269	8e-07	3.8	9918	9e-7	62.0	220018	5e-7
18	122	228135	9e-06[3]	116	220038	2e-6	94	220018	3e-3
19	0.6	2034	7e-16[4]	2.3	8136	6e-7	0.2	942	7e-8

We observe that svdifp achieves satisfactory convergence within 10000 iterations in all problems. For matrices 18 and 19, the singular values are extremely small and therefore the residual of the singular triplet computed by (3.12) is not expected to converge. For these two problems, the termination criterion is switched to using the

[3] For this matrix, the residual defined by $\|\mathbf{C}^T\mathbf{C}\mathbf{v}_1 - \sigma_1^2\mathbf{v}_1\|$ is 2e-15 while the computed singular value is 1e-12. The singular values returned by jdsvd is 5e-10. The singular values returned by irlba is 6e-7.

[4] For this matrix, the residual defined by $\|\mathbf{C}^T\mathbf{C}\mathbf{v}_1 - \sigma_1^2\mathbf{v}_1\|$ is 3e-14 while the computed singular value is 9e-15. The singular values returned by jdsvd is 1e-14. The singular values returned by irlba is 4e-16.

eigenvalue residual $\|\mathbf{C}^T\mathbf{C}\mathbf{v}_1 - \sigma_1^2\mathbf{v}_1\|$ instead when a singular value of order of the machine precision is detected (see the discussion on left singular vectors in Section 2) and then, even though the `Res` is fairly large, the computed singular values, which are given in the footnotes, are actually very good approximations already. Therefore, with the limitation of not returning any good left singular vector in such cases, `svdifp` still produces good approximate singular values and right singular vectors. `jdsvd` also achieve satisfactory convergence within 10000 iterations in all but problems 13 and 17. For those two problems, the preconditioned linear solvers in the inner iterations of `jdsvd` converge early in less than the maximum 10 iterations allowed, which is why the total matrix-vector multiplications are less than the maximum possible. Matrix 17 is also a difficult problem with 138 singular values clustered between 1.41421 and 1.41425. In terms of performance measured by `MV` and `CPU`, `jdsvd` outperforms slightly in square problems, while `svdifp` outperforms in non-square problems. In terms of `nnz`, RIF in `svdifp` has substantially less memory requirement.

We next compare `svdifp` and `jdsvd` without preconditioning. They are also compared with `irlba`. When no preconditioner is present, `jdsvd` uses MINRES as the inner linear solver. For `irlba`, we only report its results with one-sided full reorthogonalization which is the default setting. We list the results of this test in Table 3.4. For Problems 18 and 19, with extremely small singular values, the convergence test is switched to use the eigenvalue residual $\|\mathbf{C}^T\mathbf{C}\mathbf{v}_1 - \sigma_1^2\mathbf{v}_1\|$, but at termination, the residual of the singular triplet with the left singular vector computed by (3.13) has actually converged to a satisfactory level. Nevertheless, we list the computed singular values and the eigenvalue residuals in the footnotes. We note that, without preconditioning, `svdifp` converges much more slowly than the ones with preconditioning, and it appears that the additional iterations have resulted in the substantially more reduction of the singular triplet residual. We do not expect this to be the case in general.

It appears that all three methods are comparable in convergence with each method outperforming in some problems. For non-square matrices, `irlba` has the best results, outperforming in most problems. Note that `svdifp` without preconditioning is simply the restarted Lanczos method with the LOBPCG type subspace enhancement. On the other hand, `irlba` is also essentially the Lanczos method but, with the implicit restart, it uses a larger projection subspace with the same number of matrix-vector multiplications in each restart. Therefore, `irlba` may be expected to outperform `svdifp` without preconditioning in most cases. We also note that the performance of `svdifp` (Table 3.4) is significantly improved by preconditioning (Table 3.3). Several difficult problems with slow convergence are solved fairly easily after preconditioning. With a drop tolerance 10^{-3}, the RIF preconditioner appears to produce a good preconditioner that also has a relatively small number of fill-ins. Indeed, the number of non-zeros in $\mathbf{L}$ (Table 3.3) is typically 2 to 3 times that of $\mathbf{C}$ (Table 3.2).

EXAMPLE 3. In this example, we compare `svdifp` with `svds` and `lansvd`. For computing the smallest singular value, `svds` is based on applying ARPACK [49] to $\mathbf{M}^{-1}$ or the shift-and-invert matrix $(\mathbf{M} - \mu\mathbf{I})^{-1}$. `lansvd` computes the QR factorization by `R = qr(C,0)` in MATLAB and then computes the largest singular value of $\mathbf{R}^{-1}$ by the Lanczos bidiagonalization algorithm. For comparison, we use `R = qr(C,0)` as the preconditioner for `svdifp`. This approach runs into difficulty if $\mathbf{R}$ is singular or nearly singular. Indeed, `lansvd` breaks down in such situations (Problems 18 and 19). An advantage with `svdifp` is that $\mathbf{R}$ is only used as a preconditioner and its accuracy only affects the speed of convergence but not the accuracy of computed singular values. Therefore, we can simply perturb zero or nearly zero diagonals of $\mathbf{R}$ to deal with its singularity. For singular or nearly singular $\mathbf{R}$, it is important to use a column pivoting in the QR factorization but MATLAB's `R = qr(C,0)` employs a column approximate minimum degree permutation to minimize fill-ins. For this test, if the resulting $\mathbf{R}$ is nearly singular, we compute QR factorization by `[~,R,e] = qr(C,0)`, which appears to employ a column pivoting. We then set the diagonals of $\mathbf{R}$ that are less than the threshold $\sqrt{\mathbf{u}}\|\mathbf{R}\|_1$ to the threshold to construct a preconditioner for `svdifp`.

Table 3.5: Example 3: CPU - CPU time; nnz - non-zeros of $\mathbf{R}$ or $\mathbf{L}$ and $\mathbf{U}$; Res - $\|[\mathbf{C}\mathbf{v}_1 - \sigma_1\mathbf{u}_1; \mathbf{C}^T\mathbf{u}_1 - \sigma_1\mathbf{v}_1]\|/\|\mathbf{C}\|_1$.

	svdifp			svds			lansvd		
No.	CPU	nnz	Res	CPU	nnz	Res	CPU	nnz	Res
Square Matrix									
1	0.05	83918	1e-16	0.09	193650	4e-15	0.04	83918	2e-13
2	0.1	249160	6e-17	0.1	259562	5e-16	0.09	249160	7e-14
3	0.01	29165	2e-15	0.04	99351	5e-16	0.01	29165	3e-11
4	0.01	35267	9e-13	0.05	69421	1e-15	0.02	35267	3e-10
5	0.1	196083	4e-16	0.2	439407	4e-15	0.08	196083	3e-12
6	0.06	142050	5e-15	0.1	279930	4e-14	0.06	142050	4e-13
7	0.01	8561	9e-13	0.03	52239	5e-17	0.01	8561	2e-15
8	0.01	32816	2e-16	0.05	49971	3e-16	0.02	32816	2e-13
Rectangular Matrix									
9	0.01	2974	2e-13	-	-	-	0.01	2974	4e-11
10	0.01	9209	1e-12	-	-	-	0.01	9209	2e-10
11	0.8	1514019	1e-14	-	-	-	0.6	1514019	7e-16
12	0.06	48470	2e-12	-	-	-	0.05	48470	2e-13
13	0.4	313320	8e-11	-	-	-	0.3	313320	8e-15
14	0.6	505993	8e-16	-	-	-	0.3	505993	2e-12
15	0.02	30975	1e-17	-	-	-	0.02	30975	4e-14
16	0.4	263226	9e-12	-	-	-	0.2	263226	8e-11
17	54.7	550793	1e-10	-	-	-	15.6	550793	1e-16
18	10.2	2046096	5e-2[5]	-	-	-	-	-	-
19	2.3	7336	5e-17[6]	-	-	-	-	-	-

All three codes require no additional input parameters other than the matrix but we set the initial vector to the same random vector for all of them. We run the

[5] For this matrix, the residual defined by $\|\mathbf{C}^T\mathbf{C}\mathbf{v}_1 - \sigma_1^2\mathbf{v}_1\|$ is 8e-17 while the computed singular value is 2e-17.

[6] For this matrix, the residual defined by $\|\mathbf{C}^T\mathbf{C}\mathbf{v}_1 - \sigma_1^2\mathbf{v}_1\|$ is 3e-15 while the computed singular value is 3e-16.

programs till convergence as determined by themselves. We compare the residual `Res` defined by (3.20), the CPU time, as well as the number of non-zeros used in the factorizations (`nnz`). For `svdifp` and `lansvd`, `nnz` is the number of non-zeros in $\mathbf{R}$, and for `svds`, it is the total non-zeros in $\mathbf{L}$ and $\mathbf{U}$ of the LU-factorization of $\mathbf{M}$.

The results are given in Table 3.5. All three methods perform comparably for square matrices. `svds` with the zero shift fails for all non-square matrices because of singularity of $\mathbf{M}$, which is marked by "-" in the table. Even using a small nonzero shift, `svds` usually converges to the eigenvalue 0 rather than σ_1. `svdifp` and `lansvd` can both solve non-square problems with comparable performances. However, `lansvd` can fail for matrices that are nearly rank deficient (problems 18 and 19, marked by "-") because of inverting a singular or nearly singular $\mathbf{R}$. On the other hand, `svdifp` does not suffer from a similar problem because $\mathbf{L}^{-1}$ is slightly perturbed to be used as a preconditioner. Overall, `svdifp` appears most robust in this setting.

Finally, we consider `svdifp` for computing several largest singular values with deflation. With the shifts chosen inside the spectrum now, RIF constructs an LDL^T factorization for an indefinite matrix $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$. So, this also demonstrates the capability of RIF to work with indefinite matrices.

Table 3.6: Example 4: 5 largest singular values of matrix `lp_ganges`. σ_k- singular value; μ- shift used for preconditioning ; MV - # of matrix-vector multiplications; Res - $\|[\mathbf{C}\mathbf{v}_1 - \sigma_1\mathbf{u}_1; \mathbf{C}^T\mathbf{u}_1 - \sigma_1\mathbf{v}_1]\|/\|\mathbf{C}\|_1$.

	preconditioning			no preconditioning	
σ_k	μ	MV	Res	MV	Res
3.9908	3.9926	91	3e-12	443	5e-11
3.9906	3.9907	91	2e-14	289	9e-11
3.9895	3.9900	91	1e-13	531	7e-11
3.9894	3.9895	91	5e-13	641	4e-11
3.9892	3.9893	91	4e-12	1103	6e-11

Example 4. We consider `svdifp` with and without preconditioning in comput-

ing the 5 largest singular values of Matrix 15 (lp_ganges) in Table 3.2. In both cases, we set the termination threshold to 1e-10 and the number of outer iterations to 10000. To compute the largest singular value, svdifp adaptively chooses a shift for preconditioning (see Section 4). When computing the next largest singular value, the mean of the largest and the second largest singular values of the projection matrix constructed in computing the previous largest singular value is used as the shift to compute an RIF preconditioner. Then, svdifp proceeds with a deflated preconditioned iteration. Note that the second largest singular value of the projection matrix is a lower bound of the singular value to be computed and the mean value should provide a better estimate. The same procedure is used for additional singular values.

We present the results with and without preconditioning for the five largest singular values in Table 3.6. We list the number of matrix-vector multiplications (by $\mathbf{C}$ or $\mathbf{C}^T$) used for each singular value, the residual Res obtained, and in the preconditioned case, the shift μ used. We note that both methods can compute the singular values correctly while preconditioning by RIF significantly accelerates the convergence of svdifp. In particular, the shifted matrix is indefinite now but with the modest indefiniteness in computing a few extreme singular values, RIF results in a very effective preconditioner.

Chapter 4 Subspace clustering via learning a union of orthonormal bases

In this chapter, we consider the problem of clustering data which is assumed to lie in a union of subspaces. We propose a novel algorithm to solve this problem. Our algorithm is based on spectral clustering algorithms. We build the similarity matrix in a dictionary fashion, i.e, learning the dictionary and representation simultaneously.

We review in Section 4.1 some materials about spectral clustering and dictionary learning which will be used in our proposed algorithm. In Section 4.2 we discuss some existing subspace clustering methods. We propose our algorithm and present its detailed implementations in Section 4.3. Finally we demonstrate the effectiveness of our algorithm in Section 4.4.

4.1 Spectral clustering and dictionary learning

Spectral clustering and dictionary learning have been well studied in the last few decades. And they have close relation to the subspace clustering problem. In this section, we briefly review the relevant materials of these two topics.

4.1.1 Spectral clustering

The spectral clustering algorithms are a class of methods for finding clusters in a given dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$. In practice, a cluster is usually a group of similar points. Spectral clustering has been well studied in the last few decades [73]. For a comprehensive tutorial on this topic, see [85].

Let $\mathbf{W} = [w_{ij}]$ be a similarity matrix for the dataset with $w_{ij} \geq 0$ measuring similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, namely the bigger w_{ij} is, the more similar $\mathbf{x}_i$ and $\mathbf{x}_j$ are. If $w_{ij} = 0$, then $\mathbf{x}_i$ and $\mathbf{x}_j$ are not relevant to each other and they should not be classified in the same cluster. The spectral clustering methods construct a undirected graph $G = (\mathcal{X}, E)$ with the adjacency matrix $\mathbf{W}$ and then perform some graph partition methods on the graph to cluster the points.

First we review some materials in graph theory. Let $G = (\mathcal{X}, E)$ be a weighted undirected graph with non-negative weights where $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. The order of G is the number of vertices, $|G| = N$. Let w_{ij} be the weight for the edge between $\mathbf{x}_i$ and $\mathbf{x}_j$. We assume $w_{ij} = 0$ if there is no edge between $\mathbf{x}_i$ and $\mathbf{x}_j$. Since G is undirected, the adjacency matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ is a symmetric non-negative matrix The degree d_i of a vertex $\mathbf{x}_i \in \mathcal{X}$ is defined as

$$d_i := \sum_{j=1}^{N} w_{ij}. \tag{4.1}$$

Let the degree matrix $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_N)$. The Laplacian matrix $\mathbf{L}$ of graph G is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. The volume of G is defined as

$$\mathrm{vol}(G) := \sum_{i=1}^{N} d_i. \tag{4.2}$$

Given $\mathcal{S} \subset \mathcal{X}$, we let $\bar{\mathcal{S}}$ be the complement of $\mathcal{S}$ in $\mathcal{X}$, $|\mathcal{S}|$ be the number of vertices in $\mathcal{S}$ and

$$\mathrm{vol}(\mathcal{S}) = \sum_{i \in \mathcal{S}} d_i. \tag{4.3}$$

where, for convenience, we write $i \in \mathcal{S}$ if $\mathbf{x}_i \in \mathcal{S}$. If $\mathcal{S}_1, \mathcal{S}_2 \subset \mathcal{X}$ are disjoint, we let

$$W(\mathcal{S}_1, \mathcal{S}_2) = \sum_{i \in \mathcal{S}_1, j \in \mathcal{S}_2} w_{ij}. \tag{4.4}$$

$\mathcal{S}$ is called a connected component if there is no edge between $\mathcal{S}$ and $\bar{\mathcal{S}}$. Suppose $\{\mathcal{X}_1, \ldots, \mathcal{X}_K\}$ is a partition of $\mathcal{X}$, i.e., $\mathcal{X} = \mathcal{X}_1 \cup \ldots \cup \mathcal{X}_K$ and $\mathcal{X}_1, \ldots, \mathcal{X}_K$ are disjoint, we define the cut of this partition to be

$$\mathrm{cut}(\mathcal{X}_1, \ldots, \mathcal{X}_K) = \frac{1}{2} \sum_{i=1}^{K} W(\mathcal{X}_i, \bar{\mathcal{X}_i}). \tag{4.5}$$

Given above definitions, we can formulate a so-called min-cut problem.

Definition 4.1.1. *Given a weighted undirected graph $G = (\mathcal{X}, E)$ with non-negative weights, the min-cut problem is to find a partition $\{\mathcal{X}_1, \ldots, \mathcal{X}_K\}$ of $\mathcal{X}$ such that it minimizes cut$(\mathcal{X}_1, \ldots, \mathcal{X}_K)$.*

The min-cut problem is closely related to the clustering problem which is to find a partition such that similar points are in the same cluster. Suppose the w_{ij} characterize the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, then the solution of min-cut problem provides a way to cluster those vertices $\mathbf{x}_1, \ldots, \mathbf{x}_N$. However, in practice, the solution of min-cut problem may simply separate one individual vertex from the rest of the graph and that is what we do not expect in finding a solution of the clustering problem, in which clusters are usually reasonably large groups of points. Therefore, [73] proposes to minimize normalized cut denoted by Ncut instead:

$$\mathrm{Ncut}(\mathcal{X}_1, \ldots, \mathcal{X}_K) = \frac{1}{2}\sum_{i=1}^{K} \frac{W(\mathcal{X}_i, \bar{\mathcal{X}}_i)}{\mathrm{vol}(\mathcal{X}_i)}. \tag{4.6}$$

The minimization of normalized cut problem is known to be NP-complete. But it can be solved approximately in a more efficient way.

For a partition $\mathcal{X}_1, \ldots, \mathcal{X}_K$ of $\mathcal{X}$, we define an indicator matrix $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_K]$ with

$$\mathbf{h}_j = \frac{1}{\sqrt{\mathrm{vol}(\mathcal{X}_j)}} \mathbf{1}_{\mathcal{X}_j}, \quad j = 1, \ldots, K \tag{4.7}$$

where $\mathbf{1}_{\mathcal{X}_j} \in \mathbb{R}^N$ is an indicator vector whose i-th entry is 1 if $i \in \mathcal{X}_j$ and 0 otherwise. From the definition of the Laplacian matrix $\mathbf{L}$, we can obtain $\mathbf{h}_j^T \mathbf{L} \mathbf{h}_j = \mathbf{W}(\mathcal{X}_j, \bar{\mathcal{X}}_j)/\mathrm{vol}(\mathcal{X}_j)$, then the normalized cut of the partition $\mathcal{X}_1, \ldots, \mathcal{X}_K$ can be written as

$$\mathrm{Ncut}(\mathcal{X}_1, \ldots, \mathcal{X}_K) = tr(\mathbf{H}^T \mathbf{L} \mathbf{H}).$$

Observe that $\mathbf{H}^T \mathbf{D} \mathbf{H} = \mathbf{I}$, then the normalized cut minimization problem is equivalent to

$$\min_{\mathbf{H}} tr(\mathbf{H}^T \mathbf{L} \mathbf{H}) \quad s.t. \quad \mathbf{H}^T \mathbf{D} \mathbf{H} = \mathbf{I}. \tag{4.8}$$

where the minimum is taken over all possible $\mathbf{H}$ with the form described in (4.7). Notice that (4.8) is almost the same as (2.10) except that (4.8) has an additional constraint that $\mathbf{H}$ is a discrete-valued matrix with (4.7). By relaxing $\mathbf{H}$ to be a real-valued matrix, Corollay 2.1.1 provides an approximate solution $\hat{\mathbf{H}}$ to (4.8) with K specified and a partition can be constructed according to $\hat{\mathbf{H}}$ thereafter. This procedure is called normalized spectral clustering [73] which is one of the most common used spectral clustering algorithms.

Before we present the spectral clustering algorithm, let us take a look at the generalized eigenvalue problem of the pencil $(\mathbf{L}, \mathbf{D})$. From the proof of Theorem 2.1.5, it is equivalent to the standard eigenvalue problem of $\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. It is easy to verify that $\mathbf{L}_{sym}$ is diagonally dominant and hence, $\mathbf{L}_{sym}$ is positive semi-definite which can be proven by Gershgorin theorem [19]. Hence, the eigenvalues of $(\mathbf{L}, \mathbf{D})$ which are denoted as $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$ are all non-negative. In fact, $\lambda_1 = 0$ because $\mathbf{L}\mathbf{1} = \mathbf{0}$, where $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^N$.

Consider a special case of the normalized cut minimization problem, that is, the graph G has K connected components $\mathcal{X}_1, \ldots, \mathcal{X}_K$ with $|\mathcal{X}_i| = N_i$. Without loss of generality, we assume $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are ordered according to the connected components they belong to. In this special case, $\mathbf{L}$ is a block diagonal matrix, i.e., $\mathbf{L} = \text{diag}(\mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_K)$ where $\mathbf{L}_i$, $i = 1, \ldots, K$, are the Laplacian matrices for the subgraph of those K connected components respectively. Hence,

$$\mathbf{L}\mathbf{1}_{X_i} = \mathbf{L}_i \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{N_i \times N_i} = \mathbf{0}.$$

Therefore, $\mathbf{1}_{\mathcal{X}_1}, \ldots, \mathbf{1}_{\mathcal{X}_K}$ are eigenvectors of $(\mathbf{L}, \mathbf{D})$ associated with the eigenvalue 0.

We can summarize the above discussions as the following theorem.

Theorem 4.1.1. *[85] Let G be an undirected graph with non-negative weights, then the multiplicity K of the eigenvalue 0 of $(\mathbf{L}, \mathbf{D})$ is equal to the number of connected components $\mathcal{X}_1, \ldots, \mathcal{X}_K$ in the graph. And the eigenspace of eigenvalue 0 is spanned by $\mathbf{1}_{\mathcal{X}_1}, \ldots, \mathbf{1}_{\mathcal{X}_K}$.*

Theorem 4.1.1 indicates that we can cluster the rows of $\mathbf{H}$ to find the solution to the normalized cut minimization problem. We present the spectral clustering algorithm in Algorithm 4.1 [85].

Algorithm 4.1 Spectral Clustering Algorithm

```
1: **Input:** The similarity matrix $\mathbf{W}$ of the dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, number $K$ of clusters to construct.
2: **Output:** Clusters $\mathcal{X}_1, \ldots, \mathcal{X}_K$.
3: Compute $K$ eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ of the generalized eigenproblem $\mathbf{Lu} = \lambda\mathbf{Du}$ associated with the $K$ smallest eigenvalues.
4: For $i = 1, \ldots, n$, let $\mathbf{r}_i \in \mathbb{R}^K$ be the vector corresponding to the $i$-th row of $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K]$.
5: Cluster the points $\{\mathbf{r}_i\}_{i=1}^N$ with the $k$-means algorithm into clusters $\mathcal{R}_1, \ldots, \mathcal{R}_K$.
6: Clusters $\mathcal{X}_i = \{\mathbf{x}_j | \mathbf{r}_j \in \mathcal{R}_i\}$, $i = 1, \ldots, K$.
```

4.1.2 Bipartite graph clustering

In some cases, it is hard to find a good similarity matrix for the data set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$. Hence the spectral clustering algorithms can not be applied directly. However, we may be provided with a dictionary set $\mathcal{Q} = \{\mathbf{q}_i\}_{i=1}^d$ and the data set may have a simple representation $\mathbf{C} \in \mathbb{R}^{d \times N}$ in the dictionary set, i.e, $\mathbf{X} = \mathbf{QC}$ with $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ and $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_d]$. The ij-th entry c_{ij} of $\mathbf{C}$ measures how much $\mathbf{x}_j$ is related to the dictionary atom $\mathbf{q}_i$. $c_{ij} = 0$ indicates that $\mathbf{q}_i$ is not related to $\mathbf{x}_j$.

To apply spectral clustering methods, we may consider the above problem as a bipartite graph clustering problem [21, 92]. Let $\mathcal{V} = \{\mathbf{q}_1, \ldots, \mathbf{q}_d, \mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be the vertices and

$$\mathbf{W} = \begin{bmatrix} \mathbf{O} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{O} \end{bmatrix} \tag{4.9}$$

be the adjacency matrix. The we have a bipartite graph. We cluster the dictionary atoms $\mathcal{Q}$ into $\{\mathcal{Q}_i\}_{i=1}^K$ simultaneously while clustering the points $\mathcal{X}$ into $\{\mathcal{X}_i\}_{i=1}^K$ so that $\mathcal{Q}_i$ has weak connections with $\{\mathcal{X}_j | j \neq i\}$ and $\mathcal{X}_i$ has a weak connection to $\{\mathcal{Q}_j | j \neq i\}$. In the words of graph theory, we hope to find clusters that minimize the following quantity

$$\sum_{i=1}^K \frac{W(\mathcal{Q}_i, \sum_{j \neq i} \mathcal{X}_j) + W(\sum_{j \neq i} \mathcal{Q}_j, \mathcal{X}_i)}{\mathrm{vol}(\mathcal{Q}_i) + \mathrm{vol}(\mathcal{X}_i)}$$

which can be represented as $\mathrm{Ncut}((\mathcal{Q}_1, \mathcal{X}_1), \ldots, (\mathcal{X}_K, \mathcal{Q}_K))$ as in (4.6). Then we can apply Algorithm 4.1 to find clusters in $\mathcal{V}$.

Let $\mathbf{D} = \mathrm{diag}(\mathbf{D}_R, \mathbf{D}_C)$ and $\mathbf{D}_R \in \mathbb{R}^{d\times d}$, $\mathbf{D}_C \in \mathbb{R}^{N\times N}$ be diagonal matrices with

$$\mathbf{D}_R(i,i) = \sum_{j=1}^{N} c_{ij}, \quad \mathbf{D}_C(j,j) = \sum_{i=1}^{d} c_{ij}. \tag{4.10}$$

Then the Laplacian matrix of the bipartite graph is

$$\mathbf{L} = \begin{bmatrix} \mathbf{D}_R & -\mathbf{C} \\ -\mathbf{C}^T & \mathbf{D}_C \end{bmatrix}$$

Consider the generalized eigenvalue problem of $(\mathbf{L}, \mathbf{D})$, suppose $\mathbf{z} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}$ with $\mathbf{u} \in \mathbb{R}^d$ and $\mathbf{v} \in \mathbb{R}^N$ is an eigenvector of $(\mathbf{L}, \mathbf{D})$ associated with λ, then

$$\mathbf{C}\mathbf{v} = (1-\lambda)\mathbf{D}_R\mathbf{u}$$
$$\mathbf{C}^T\mathbf{u} = (1-\lambda)\mathbf{D}_C\mathbf{v}$$

which can be rewritten as

$$\begin{aligned} \mathbf{D}_R^{-1/2}\mathbf{C}\mathbf{D}_C^{-1/2}(\mathbf{D}_C^{1/2}\mathbf{v}) &= (1-\lambda)(\mathbf{D}_R^{1/2}\mathbf{u}) \\ \mathbf{D}_C^{-1/2}\mathbf{C}^T\mathbf{D}_R^{-1/2}(\mathbf{D}_R^{1/2}\mathbf{u}) &= (1-\lambda)(\mathbf{D}_C^{1/2}\mathbf{v}) \end{aligned} \tag{4.11}$$

Hence, instead of finding K eigenvectors corresponding to the K smallest eigenvalue of $(\mathbf{L}, \mathbf{D})$ in Algorithm 4.1, we can simply compute the left and right singular vectors $(\hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i)_{i=1}^K$ corresponding to the largest singular value of $\hat{\mathbf{C}} = \mathbf{D}_R^{-1/2}\mathbf{C}\mathbf{D}_C^{-1/2}$. And the eigenvector $\{\mathbf{z}_i\}_{i=1}$ of $(\mathbf{L}, \mathbf{D})$ can be obtained by

$$\mathbf{z}_i = \begin{bmatrix} \mathbf{D}_R^{-1/2}\hat{\mathbf{u}}_i \\ \mathbf{D}_C^{-1/2}\hat{\mathbf{v}}_i \end{bmatrix}, \quad i = 1, \ldots, K. \tag{4.12}$$

The complete bipartite graph clustering algorithm is presented in Algorithm 4.2. Note that in Algorithm 4.2, the number K of clusters is provided. However, we point out that it is not always necessary. One can always determine K by exploring the spectrum of $\hat{C} \in \mathbb{R}^{d\times N}$ (We always assume $d < N$). Namely, let

$$K = d - \arg\max_{i=1,\ldots,d}(\sigma_{i+1}(\hat{C}) - \sigma_i(\hat{C})). \tag{4.13}$$

Algorithm 4.2 Bipartite Graph Clustering

```
Input: The representation matrix $\mathbf{C}$ of $\{\mathbf{x}_i\}_{i=1}^N$ with respect to dictionary $\{\mathbf{q}_i\}_{i=1}^d$, number $K$ of clusters to be clustered in $\{\mathbf{x}_i\}_{i=1}^N$.
Output: Clusters $\mathcal{X}_1, \dots, \mathcal{X}_k$.
$\hat{\mathbf{C}} = \mathbf{D}_R^{-1/2}\mathbf{C}\mathbf{D}_C^{-1/2}$ where $\mathbf{D}_R$ and $\mathbf{D}_C$ are computed by (4.10).
Compute $K$ left singular vectors $\{\hat{\mathbf{u}}_1, \dots, \hat{\mathbf{u}}_K\}$ and right singular vectors $\{\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_K\}$ of $\hat{\mathbf{C}}$ corresponding to the $K$ largest singular values and form $\mathbf{Z}_K = [\mathbf{z}_1, \dots, \mathbf{z}_K]$ where $\{\mathbf{z}_i\}_{i=1}^K$ are computed by (4.12).
For $i = 1, \dots, N + d$, let $\mathbf{r}_i \in \mathbb{R}^K$ be the vector corresponding to the $i$-th row of $\mathbf{Z}$.
Cluster the points $\{\mathbf{r}_i\}_{i=1}^{N+d}$ with the $k$-means algorithm into clusters $\mathcal{R}_1, \dots, \mathcal{R}_K$.
Clusters $\mathcal{X}_i = \{x_j | \mathbf{r}_{j+d} \in \mathcal{R}_i\}, \quad i = 1, \dots, K$.
```

4.1.3 A dictionary learning method

The problem of dictionary learning for sparse representation assumes that a natural signal can be represented by a small number of elementary components, which are called dictionary atoms [45, 53, 54]. It has received extensive interests and a number of sparse representation algorithms have been proposed, such as MOD [24], K-SVD [2] and SimCO [17], etc. In this section, we will introduce one of those algorithms which assumes the dictionary is a union of orthonormal bases [52].

Suppose $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$ is a data matrix with each column representing a data sampled from $\mathbb{R}^D$. The task of dictionary learning is to learn a dictionary $\mathbf{Q} \in \mathbb{R}^{D \times d}$ providing a sparse representation for $\mathbf{X}$, i.e., to find $\mathbf{Q}$ with $\mathbf{X} = \mathbf{QC} + \mathbf{E}$ where $\mathbf{E}$ is a noise matrix and $\mathbf{C}$ is a sparse matrix. In general, when $\mathbf{Q}$ has no special structure, computing the solution to the above problem is very computational intensive. However, when $\mathbf{Q}$ is structured as a union of orthonormal bases, the computational cost is reduced. In additional, it has been found that some real-world datasets, such as audio signals and images, can be modeled as the superimposition of several layers, each of which having sparse representation in its own adapted orthonormal basis. Wavelet decomposition is one such example. [52] proposes to consider the dictionary learning problem with a dictionary set $\mathbf{Q}$ that is locally orthonormal. Namely, we solve

$$\begin{aligned} \arg\min_{\mathbf{Q},\mathbf{C}} \quad & \|\mathbf{X}-\mathbf{QC}\|_F^2+\lambda\|\mathbf{C}\|_1 \\ s.t. \quad & \mathbf{Q}=[\mathbf{Q_1},\ldots,\mathbf{Q_K}] \\ & \mathbf{Q}_i^T\mathbf{Q}_i=\mathbf{I}, \quad i=1,\ldots,K. \end{aligned} \tag{4.14}$$

In (4.14), the number K is known as a prior and the dimensions of each orthonormal basis $\mathbf{Q}_i$ are predetermined. Since the l_1 penalty on $\mathbf{C}$ forces $\mathbf{C}$ to be sparse, the dimensions of $\mathbf{Q}_i$ need not to be accurate as long as they are over-estimations of the true dimensions.

Since joint optimization of $\mathbf{Q}$ and $\mathbf{C}$ is very hard, an alternating optimization strategy is employed which consists of two stages:

1. Sparse coding: given a dictionary $\mathbf{Q}$, find a sparse matrix $\mathbf{C}$:

$$\arg\min_{\mathbf{C}} \|\mathbf{X}-\mathbf{QC}\|_F^2+\lambda\|\mathbf{C}\|_1. \tag{4.15}$$

2. Dictionary update: given $\mathbf{C}$, find a dictionary:

$$\begin{aligned} \arg\min_{\mathbf{Q}} \quad & \|\mathbf{X}-\mathbf{QC}\|_F^2 \\ s.t. \quad & \mathbf{Q}=[\mathbf{Q}_1,\ldots,\mathbf{Q}_K] \\ & \mathbf{Q}_i^T\mathbf{Q}_i=\mathbf{I}, \quad i=1,\ldots,K. \end{aligned} \tag{4.16}$$

[52] suggests to use basis pursit [15] to solve (4.15). Since $\mathbf{Q}$ is a union of orthonormal basis, basis pursuit for (4.15) can be implemented efficiently with a Block Coordinate Relaxation(BCR) algorithm [72].

Let $\mathbf{C}=[\mathbf{C}_1^T,\ldots,\mathbf{C}_K^T]^T$ with $\mathbf{C}_i\in\mathbb{R}^{d_i\times N}$, $i=1,\ldots,K$. The idea of BCR is to iteratively select and update a block $\mathbf{C}_i$ instead of updating $\mathbf{C}$ directly to take advantage of the orthonormality of $\mathbf{Q}_i$. The selection strategy could be a systematic cycle rule or an optimal descent rule [52, 72]. If K is large, it is computationally infeasible to systematically cycle through all blocks $\mathbf{C}_i$ [72]. However, in this work, we assume K is small. Then, the systematic cycle rule is appropriate.

The subproblem of updating $\mathbf{C}_i$ with all other variables fixed is

$$\arg\min_{\mathbf{C}} \|\hat{\mathbf{X}}_i-\mathbf{Q}_i\mathbf{C}_i\|_F^2+\lambda\|\mathbf{C}_i\|_1. \tag{4.17}$$

where $\hat{\mathbf{X}}_i = \mathbf{X} - \sum_{j=i} \mathbf{Q}_i \mathbf{C}_i$. Then the solution of (4.17) is given by

$$\mathbf{C}_i = \mathrm{soft}(\mathbf{Q}_i^T \mathbf{X}, \lambda/2) \tag{4.18}$$

where soft is an entry-wise function on matrices with

$$\mathrm{soft}(z, \tau) = \begin{cases} z - \tau, \text{ if } z > \tau \\ z + \tau, \text{ if } z < -\tau \\ 0, \text{ otherwise} \end{cases}$$

Since BCR might converge very slowly when λ is small, [52] proposes to use a modified BCR algorithm which is summarized in Algorithm 4.3.

Algorithm 4.3 Modified BCR Algorithm

```
Input: $\mathbf{X} \in \mathbb{R}^{D\times N}$, $\{\mathbf{Q}_i : \mathbf{Q}_i^T\mathbf{Q}_i = \mathbf{I}, \mathbf{Q}_i \in \mathbb{R}^{D\times d_i}\}_{i=1}^K$, $\{\mathbf{C}_i : \mathbf{C}_i \in \mathbb{R}^{d_i\times N}\}_{i=1}^K$,
  $\lambda_0 > 0$.
Output: $\{\mathbf{C}_i : \mathbf{C}_i \in \mathbb{R}^{d_i\times N}\}_{i=1}^K$
for $i = 1, \ldots, M$ do
    for $j = 1, \ldots, K$ do
        $\hat{\mathbf{X}}_j = \mathbf{X} - \sum_{l\neq j} \mathbf{Q}_l\mathbf{C}_l$.
        Update $\mathbf{C}_j$ with $\lambda = \lambda_0(1 - (i-1)/M)$ by (4.18).
    end for
end for
```

Similarly, at the stage of dictionary update, $\mathbf{Q}_i$ is also updated one by one. Observe that the subproblem

$$\begin{aligned} \arg\min_{\mathbf{Q_i}} \quad & \|\hat{\mathbf{X}}_\mathbf{i} - \mathbf{Q_i}\mathbf{C_i}\|_F^2 \\ s.t. \quad & \mathbf{Q}_i^T\mathbf{Q}_i = \mathbf{I} \end{aligned} \tag{4.19}$$

is a constrained least squares problem, $\mathbf{Q}_i$ can be solved analytically by the following theorem, which is proved in [52] using an optimization approach. Here we give a proof from the perspective of numerical linear algebra.

Theorem 4.1.2. *Suppose $\mathbf{X} \in \mathbb{R}^{D\times N}$ and $\mathbf{C} \in \mathbb{R}^{d\times N}$ with $d < D$. Let $\mathbf{X}\mathbf{C}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is the singular value decomposition of $\mathbf{X}\mathbf{C}^T$ with $\mathbf{U} \in R^{D\times d}$, $\mathbf{\Sigma} \in \mathbb{R}^{d\times d}$ and $\mathbf{V} \in R^{d\times d}$. Then $\mathbf{Q} = \mathbf{U}\mathbf{V}^T$ solves*

$$\arg\min_{\mathbf{Q}^T\mathbf{Q}=\mathbf{I}} \|\mathbf{X} - \mathbf{Q}\mathbf{C}\|_F^2. \tag{4.20}$$

Proof. We have

$$\begin{aligned}
\arg\min_{\mathbf{Q}^T\mathbf{Q}=I} \|\mathbf{X}-\mathbf{Q}\mathbf{C}\|_F^2 &= \arg\min_{\mathbf{Q}^T\mathbf{Q}=\mathbf{I}} tr((\mathbf{X}-\mathbf{Q}\mathbf{C})^T(\mathbf{X}-\mathbf{Q}\mathbf{C})) \\
&= \arg\min_{\mathbf{Q}^T\mathbf{Q}=I} tr(-\mathbf{C}^T\mathbf{Q}^T\mathbf{X}-\mathbf{X}^T\mathbf{Q}\mathbf{C}+\mathbf{C}^T\mathbf{Q}^T\mathbf{Q}\mathbf{C}) \\
&= \arg\min_{\mathbf{Q}^T\mathbf{Q}=I} -tr(\mathbf{C}^T\mathbf{Q}^T\mathbf{X}) \qquad (4.21) \\
&= \arg\min_{\mathbf{Q}^T\mathbf{Q}=I} -tr(\mathbf{Q}^T\mathbf{X}\mathbf{C}^T) \\
&= \arg\max_{\mathbf{Q}^T\mathbf{Q}=I} tr(\mathbf{Q}^T\mathbf{X}\mathbf{C}^T)
\end{aligned}$$

Since $\mathbf{X}\mathbf{C}^T=\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$,

$$tr(\mathbf{Q}^T\mathbf{X}\mathbf{C}^T)=tr(\mathbf{Q}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)=tr(\mathbf{\Sigma}\mathbf{V}^T\mathbf{Q}^T\mathbf{U})$$

Let $\mathbf{A}=\mathbf{V}^T\mathbf{Q}^T\mathbf{U}$. Then $tr(\mathbf{\Sigma}\mathbf{V}^T\mathbf{Q}^T\mathbf{U})=\sum_{i=1}^{d}\sigma_i a_{ii}$ where σ_i, $i=1,\ldots,d$ are diagonal elements of $\mathbf{\Sigma}$. Let $\tilde{\mathbf{U}}=[\mathbf{U},\hat{\mathbf{U}}]\in\mathbb{R}^{D\times D}$ with $\hat{\mathbf{U}}^T\hat{\mathbf{U}}=\mathbf{I}$ and $\mathbf{U}^T\hat{\mathbf{U}}=\mathbf{O}$. Then, $\mathbf{V}^T\mathbf{Q}^T\tilde{\mathbf{U}}(\mathbf{V}^T\mathbf{Q}^T\tilde{\mathbf{U}})^T=\mathbf{I}$. Since $\mathbf{V}^T\mathbf{Q}^T\tilde{\mathbf{U}}=[\mathbf{A}\ \mathbf{V}^T\mathbf{Q}\hat{\mathbf{U}}]$, then $|a_{ii}|\leq 1$. Therefore

$$tr(\mathbf{\Sigma}\mathbf{V}^T\mathbf{Q}^T\mathbf{U})\leq tr(\mathbf{\Sigma}).$$

The equality can only be obtained with $\mathbf{V}^T\mathbf{Q}^T\mathbf{U}=\mathbf{I}$. Therefore

$$\mathbf{Q}=\mathbf{U}\mathbf{V}^T. \qquad (4.22)$$

□

4.2 Subspace clustering

Let $\mathcal{X}=\{\mathbf{x}_i\}_{i=1}^N$ be a collection of points drawn from a union of K unknown subspaces $\{\mathcal{S}_i\}_{i=1}^K$ of the ambient space $\mathbb{R}^D$ with unknown dimensions $d_i=\dim(\mathcal{S}_i)$, $i=1,\ldots,K$. Let $\mathbf{X}=[x_1,\ldots,\mathbf{x}_N]$. The task of subspace clustering is to find the number K of subspaces and their dimensions $\{d_i\}_{i=1}^K$, the K subspaces $\{\mathcal{S}_i\}_{i=1}^K$ which can be characterized by finding a set of basis matrices $\{\mathbf{B}_i\in\mathbb{R}^{D\times d_i}\}_{i=1}^K$ with $\mathcal{R}(\mathbf{B}_i)=\mathcal{S}_i$, and a segmentation $\{\mathcal{X}_i\}_{i=1}^K$ where $\mathcal{X}_i\subset\mathcal{S}_i\cap\mathcal{X}$ and $|\mathcal{X}_i|=N_i$.

It is worth pointing out that the subspace clustering problem will be reduced to a linear dimensionality reduction problem if $K=1$ which can be solved analytically by Principal Component Analysis(PCA). However, the subspace clustering problem becomes significantly more difficult when $K > 1$ due to a number of challenges which are listed as follows:

1. The unknown parameters K and $\{d_i\}_{i=1}^K$. The ideal algorithm should be capable of estimating the parameters simply from the data matrix $\mathbf{X}$ without any prior knowledge. However, it is not an easy task. In addition, the segmentation of $\mathcal{X}$ may be sensitive to the estimations of K and $\{d_i\}_{i=1}^K$.

2. The separation between subspaces $\{\mathcal{S}_i\}_{i=1}^K$. The subspaces could be linearly independent, disjoint (by disjoint, we mean the subspaces have only trivial intersection) or even have non-trivial intersection (there exists a non zero vector with $\mathbf{x} \in \mathcal{S}_i \cap \mathcal{S}_j$). While it is less difficult to solve the subspace clustering problem for the former two cases, it is much more difficult to solve for the third case.

3. The existence of noises and outliers will distort the true underlying subspace structure, thus making the subspace clustering problem even harder.

Next we will present a number of subspace clustering algorithms from the literature.

4.2.1 Existing subspace clustering algorithms

A number of subspace clustering algorithms have been proposed in the past few decades. Most of them can be divided into four main categories: algebraic methods, iterative methods, statistical methods, and spectral clustering-based methods [81].

4.2.1.1 Algebraic Methods

Algebraic methods obtain the segmentation of the data by exploiting the algebraic structure. For example, [13, 16, 27] are based on a low rank factorization of the data matrix $\mathbf{X}$. Specifically, there exists a permutation matrix $\mathbf{\Gamma} \in \mathbb{R}^{N \times N}$ such that

$\mathbf{X\Gamma} = [\mathbf{X}_1, \ldots, \mathbf{X}_K]$ where the columns of $\mathbf{X}_i$ are sampled from the same subspace $\mathcal{S}_i$ with dimension d_i. Suppose $\mathbf{Q}_i \in \mathbb{R}^{D\times d_i}$ is an orthonormal basis for $\mathcal{S}_i$, then

$$\mathbf{X}_i = \mathbf{Q}_i\mathbf{Z}_i, \quad i = 1, \ldots, K,$$

where $\mathbf{Z}_i \in \mathbb{R}^{d_i\times N_i}$ is the low dimensional representation of the points with respect to $\mathbf{Q}_i$. If we assume the subspaces $\{\mathcal{S}_i\}_{i=1}^K$ are linearly independent, then $d = rank(X) = \sum_{i=1}^n d_i \leq \min\{D, N\}$ and

$$\mathbf{X\Gamma} = [\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_K] \begin{bmatrix} \mathbf{Z}_1 & & & \\ & \mathbf{Z}_2 & & \\ & & \ddots & \\ & & & \mathbf{Z}_K \end{bmatrix} = \mathbf{QZ},$$

where $\mathbf{Q} = [\mathbf{Q}_1, \ldots, \mathbf{Q}_K] \in \mathbb{R}^{D\times d}$ and $\mathbf{Z} \in \mathbb{R}^{d\times N}$. Suppose $\mathbf{X} = \mathbf{U\Sigma V}^T$ is the rank-d SVD of the data matrix, i.e., $\mathbf{U} \in \mathbb{R}^{D\times d}, \mathbf{\Sigma} \in \mathbb{R}^{d\times d}$ and $\mathbf{V} \in \mathbb{R}^{N\times d}$. Let

$$\mathbf{W} = \mathbf{VV}^T \in \mathbb{R}^{N\times N}.$$

Then $w_{ij} = 0$ if points $\mathbf{x}_i$ and $\mathbf{x}_j$ are in different subspaces, see [40] for a proof. This fact can be used to segment the data by thresholding the entries of $\mathbf{W}$. However, the matrix factorization-based methods are provably correct when the subspaces are linearly independent and may fail when the assumption is violated. They are also highly sensitive to the presence of noise and outliers.

Generalized Principal Component Analysis(GPCA) [83] fits the data with a set of polynomials whose gradients at a point give a normal vector to the subspace containing that point. Then an orthonormal basis $\mathbf{Q}_i$ can be obtained for each subspace $\mathcal{S}_i$. A segmentation of the data can be obtained thereafter. GPCA does not require any prior knowledge of the number of subspaces and their dimensions and it can deal with both linearly independent and dependent subspaces. However, it is computationally expensive and its complexity increases exponentially in terms of the number and dimensions of subspaces. It is also sensitive to noises and outliers.

4.2.1.2 Iterative Methods

Iterative methods, such as k-planes [14], K-subspaces [79] and median K-flats [93] are generalizations of K-means algorithm. Basically, given initial subspaces, iterative methods alternate between assigning points to subspaces and re-estimating the subspaces. In particular, K subspace method tries to find a set of centroids $\{\mathbf{z}_i\}_{i=1}^K$, an orthonormal basis $\{\mathbf{Q}_i\}_{i=1}^K$ for subspaces, low rank representations $\{\mathbf{y}_i\}_{i=1}^N$ for $\mathcal{X}$ and a set of scalars $\{w_{ij}\}_{i=1,\ldots K, j=1,\ldots,N}$ with $w_{ij} \in \{0, 1\}$ which solve

$$\min_{\{\mathbf{z}_i\}_{i=1}^K, \{\mathbf{Q}_i\}_{i=1}^K, \{\mathbf{y}_i\}_{i=1}^N, \{w_{ij}\}} \sum_{i=1}^{K} \sum_{j=1}^{N} w_{ij} \|\mathbf{z}_i - \mathbf{Q}_i \mathbf{y}_j\|_2^2$$

with constraint $\sum_{i=1}^K w_{ij} = 1$. Jointly optimizing all these variables is very hard. So K subspace method exploits the same strategy as K means to iteratively cluster those points and solve a PCA problem for each cluster. In general, iterative methods require prior knowledge of the number of subspaces and their dimensions. They are also sensitive to initializations.

4.2.1.3 Statistical Methods

Statistical methods make assumptions about the distribution of data inside the subspaces or the distribution of noise. Mixtures of Probabilistic PCA (MPPCA) [78] and Agglomerative Lossy Compression (ALC) [59] assume that data inside each subspace have (degenerate) Gaussian distribution. MPPCA alternates between clustering and subspace estimation by applying Expectation Maximization method while ALC finds the segmentation of data to minimize the overall coding length subject to a given distortion. The main drawbacks of MPPCA are that it needs to know the number and dimensions of subspaces beforehand and it is sensitive to initialization. Though ALC automatically determines the number and dimensions of subspaces, there is no theoretical proof for its optimality.

4.2.1.4 Spectral Clustering-based Methods

Spectral clustering-based subspace clustering methods construct a similarity matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ for $\{\mathbf{x}_i\}_{i=1}^N$ and then apply spectral clustering algorithms introduced in Section 4.1.1 to segment the data into clusters. Some methods introduced earlier can be extended to form a similarity matrix, such as GPCA.

Local Subspace Affinity(LSA), Spectral Local Best-fit Flats(SLBF) and Locally Linear Manifold Clustering algorithms use local information around each point to build the similarity matrix. They have an intrinsic difficulty in dealing with points near the intersection of two subspaces. In addition, a right choice of the neighborhood size to compute the local information at each point is critical.

The state-of-the-art subspace clustering algorithms, such as Sparse Subspace Clustering [22, 23], Low Rank Representation(LRR) [56] and their variants [87, 69, 57, 82] are also spectral clustering-based methods. Those algorithms build better similarities between data points using global information. The basic observation of SSC and LRR is that data in a union of multiple subspaces have self-expressiveness property(SEP) [23] as defined below.

Definition 4.2.1. *A dataset has a self-expressive property if each data in this dataset can be reconstructed by a combination of other points in this dataset. In other words, there exists a coefficient matrix* $\mathbf{C} \in \mathbb{R}^{N \times N}$ *such that* $\mathbf{X} = \mathbf{X}\mathbf{C}$ *with* $\operatorname{diag}(\mathbf{C}) = 0$.

Since $N > D$ in general, there are infinitely many $\mathbf{C}$. To build a good $\mathbf{C}$, we would like to enforce some penalty function $f(\mathbf{C})$ on $\mathbf{C}$ such that only the representations using points from the same subspace are favored. Namely, we solve the following optimization problem.

$$\arg\min_{\mathbf{C}} f(\mathbf{C}), \quad s.t. \quad \mathbf{X} = \mathbf{X}\mathbf{C},\ \operatorname{diag}(\mathbf{C}) = 0. \tag{4.23}$$

Then the similarity matrix is constructed as $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}^T|$.

The key idea of SSC is that the most efficient representation of the data point $x \in \mathcal{S}_i$ is a combination of at most d_i points in the same subspace. It motivates us to find the sparsest $\mathbf{C}$ in (4.23), i.e., $f(\mathbf{C}) = \|\mathbf{C}\|_0$. Since solving the sparse optimization problem is generally hard, SSC replaces $\|\mathbf{C}\|_0$ by its convex relaxation $\|\mathbf{C}\|_1$. The l_1 optimization step can be solved efficiently. While SSC achieves state-of-the-art performance in subspace clustering problems, it suffers from the graph connectivity problem: it is possible that points in the same subspace form multiple components of the graph and are misclassified as being in different subspaces.

LRR observes that with the assumption that $\{\mathcal{S}_i\}_{i=1}^N$ are linearly independent, the lowest rank solution of $\mathbf{X} = \mathbf{XC}$ implies the desired representation, i.e, the data point $\mathbf{x} \in \mathcal{S}_i$ corresponds to a linear combination of data points in the same subspace. Since rank minimization problem is also NP-hard, LRR uses a convex surrogate $f(\mathbf{C}) = \|\mathbf{C}\|_*$, the nuclear norm of $\|\mathbf{C}\|_*$. In addition, LRR does not require the constraint $\text{diag}(\mathbf{C}) = 0$ to avoid the situation $C = \mathbf{I}$ which might occur in SSC. Though LRR is practically successful, it is provably effective if the subspaces are not independent.

There are also some other choices of $f(\mathbf{C})$ recently, such as $\|\mathbf{C}\|_F^2$, $\|\mathbf{C}\|_* + \lambda\|\mathbf{C}\|_1$, see [58, 87] for more discussions.

4.3 A novel subspace clustering algorithm via learning orthonormal bases

Observe that recently developed subspace clustering algorithms have a close relation with dictionary learning. While the dictionary learning consists of two steps: coding step and dictionary update step, see Section 4.1.3 as an example, SSC and LRR only have the coding step as they already take the dataset $\mathcal{X}$ as a good dictionary because of its self-expressiveness property. But it is still worth finding a better dictionary, with which the representation is more informative about the underlying subspaces in the dataset. [39] developed a dictionary learning based subspace clustering algorithm which assumes the dictionary and sparse coding coefficients are both non-negative. It is effective for clustering in non-negative datasets but not suitable for other more general datasets. Recently, [1] applied a dictionary learning algorithm called k-SVD to generate an over-complete dictionary. With the learned dictionary, a sparse representation is recovered by Orthogonal Matching Pursuit [60]. Clusters are obtained by applying bipartite graph clustering to the representation. While this method is efficient, it lacks of proofs about effectiveness. To the best of our knowledge, there is no direct relation between the dictionary learned by k-SVD and the desirable dictionary which can characterize the subspaces structure of the dataset. Therefore, it is necessary to find a more suitable dictionary to achieve better subspace clustering results.

4.3.1 Motivation

When searching for subspaces that cluster the data points, an orthonormal basis is the most appropriate way to represents a subspace. It also offers better numerical stability in computations. We therefore consider representation of the data points in orthonormal bases. Recall the idea of matrix-factorization based methods in Section 4.2.1.1, the data matrix $\mathbf{X}$ can be represented by a union of orthogonal bases:

$$\mathbf{X} = [\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_K] \begin{bmatrix} \mathbf{Z}_1 & & & \\ & \mathbf{Z}_2 & & \\ & & \ddots & \\ & & & \mathbf{Z}_K \end{bmatrix} \mathbf{\Gamma} = \mathbf{QC}.$$

We write the corresponding blocks of $\mathbf{C}$ as $\mathbf{C} = [\mathbf{C}_1, \ldots, \mathbf{C}_K]$. If $\mathbf{Q}_i$ is an orthogonal basis of $\mathcal{S}_i$, then each block $\mathbf{C}_i$ has only N_i non-zero columns, i.e, $\|\mathbf{C}_i\|_{2,0} = N_i$. For any other orthonormal matrices, $\sum_{i=1}^{K} \|\mathbf{C}_i\|_{2,0} \geq N$. In addition, each column of $\mathbf{C}$ has at most $r = \max\{d_1, \ldots, d_K\}$ elements, which indicates C is a sparse matrix. Those facts inspire us to find $\mathbf{C}$ with minimum $\|\mathbf{C}\|_0$ and $\sum_{i=1}^{K} \|\mathbf{C}_i\|_{2,0}$ to get a desirable block structure of $\mathbf{C}$, i.e., $\mathbf{C}$ is block diagonal up to some column permutation. We find such $\mathbf{C}$ by solving the following problem.

$$\begin{aligned} \arg\min_{\mathbf{Q},\mathbf{C},\mathbf{E}} \quad & \|\mathbf{C}\|_0 + \mu \sum_{i=1}^{K} \|\mathbf{C}_i\|_{2,0} + \lambda \|\mathbf{X} - \mathbf{QC}\|_F^2 \\ s.t \quad & \mathbf{Q} = [\mathbf{Q}_1, \ldots, \mathbf{Q}_K], \mathbf{Q}_i \in \mathbb{R}^{D \times d_i} \\ & \mathbf{Q}_i^T \mathbf{Q}_i = \mathbf{I},\ i = 1, \ldots, K \\ & \mathbf{C} = [\mathbf{C}_1^T, \ldots, \mathbf{C}_K^T]^T, \mathbf{C}_i \in \mathbb{R}^{d_i \times N} \end{aligned} \tag{4.24}$$

where K and $\{d_i\}_{i=1}^K$ are predefined and $\mathbf{E} = \mathbf{X} - \mathbf{QC}$ represents the noise in $\mathbf{X}$. Since the sparse optimization problem is generally NP-hard, we proposes to solve an alternate optimization problem (4.25) by replacing $\|\cdot\|_0$ and $\|\cdot\|_{2,0}$ with their convex relaxations $\|\cdot\|_1$ and $\|\cdot\|_{2,1}$.

$$\begin{aligned} \arg\min_{Q,C} \quad & \|\mathbf{C}\|_1 + \mu \sum_{i=1}^K \|\mathbf{C}_i\|_{2,1} + \lambda \|\mathbf{X} - \mathbf{QC}\|_F^2 \\ s.t \quad & \mathbf{Q} = [\mathbf{Q}_1, \ldots, \mathbf{Q}_K], \mathbf{Q}_i \in \mathbb{R}^{D \times d_i} \\ & \mathbf{Q}_i^T \mathbf{Q}_i = \mathbf{I},\ i = 1, \ldots, K \\ & \mathbf{C} = [\mathbf{C}_1^T, \ldots, \mathbf{C}_K^T]^T, \mathbf{C}_i \in \mathbb{R}^{d_i \times N} \end{aligned} \tag{4.25}$$

The main difference between (4.25) and (4.14) is the extra penalty term $\sum_{i=1}^K \|\mathbf{C}_i\|_{2,1}$, which promotes fewer nonzero columns in each block $\mathbf{C}_i$. At the same time, the l_1 penalty term promotes sparsity. As a result, it forces the data in a subspace to be represented by as few blocks in $\mathcal{Q} = \{\mathbf{Q}_i\}_{i=1}^K$ as possible.

Observe that (4.25) is similar to (4.14) which is briefly discussed in Section 4.1.3. We can use the same alternating optimization strategy to solve (4.25).

1. Assuming $\mathbf{Q}$ is fixed, we can update $\mathbf{C}$ by solving

$$\begin{aligned} \arg\min_{C} \quad & \|\mathbf{C}\|_1 + \mu \sum_{i=1}^K \|\mathbf{C}_i\|_{2,1} + \lambda/2 \|\mathbf{X} - \mathbf{QC}\|_F^2 \\ s.t \quad & \mathbf{C} = [\mathbf{C}_1^T, \ldots, \mathbf{C}_K^T]^T, \mathbf{C}_i \in \mathbb{R}^{d_i \times N} \end{aligned} \tag{4.26}$$

   To take advantage the orthonormality of $\mathbf{Q}_i$, we can use a similar BCR algorithm to update $\mathbf{C}_i$ iteratively. First, let us consider the subproblem of updating $\mathbf{C}_i$:

$$\arg\min_{\mathbf{C}_i} \quad \|\mathbf{C}_i\|_1 + \mu \|\mathbf{C}_i\|_{2,1} + \lambda/2 \|\hat{\mathbf{X}}_i - \mathbf{Q}_i \mathbf{C}_i\|_F^2 \tag{4.27}$$

   where $\hat{\mathbf{X}}_i = \mathbf{X} - \sum_{j \neq i} \mathbf{Q}_j \mathbf{C}_j$. We present an alternating direction method of multipliers (ADMM) method to solve (4.27).

   To start, we introduce an auxiliary matrix $\mathbf{F} \in \mathbb{R}^{d_i \times N}$ and consider the optimization problem

$$\begin{aligned} \arg\min_{\mathbf{C}_i, \mathbf{F}_i} \quad & \|\mathbf{C}_i\|_1 + \mu \|\mathbf{F}_i\|_{2,1} + \lambda/2 \|\hat{\mathbf{X}}_i - \mathbf{Q}_i \mathbf{F}_i\|_F^2 \\ s.t. \quad & \mathbf{C}_i = \mathbf{F}_i \end{aligned} \tag{4.28}$$

The augmented Lagrangian of (4.28) is given by

$$\begin{aligned} L = \|\mathbf{C}_i\|_1 + \mu\|\mathbf{F}_i\|_{2,1} + \lambda/2\|\hat{\mathbf{X}}_i - \mathbf{Q}_i\mathbf{F}_i\|_F^2 \\ + \rho/2\|\mathbf{C}_i - \mathbf{F}_i\|_F^2 + tr(\boldsymbol{\Psi}_i^T(\mathbf{C}_i - \mathbf{F}_i)). \end{aligned} \tag{4.29}$$

where $\boldsymbol{\Psi}_i \in \mathbb{R}^{d_i \times N}$ is a matrix of Lagrange multipliers. The ADMM is an iterative approach of updating $\mathbf{C}_i$, $\mathbf{F}_i$ and $\boldsymbol{\Psi}_i$ alternatively. In the following algorithm, we let $\{\mathbf{C}_i^{(k)}, \mathbf{F}_i^{(k)}\}$ be the optimization variables at step k and $\boldsymbol{\Psi}_i^{(k)}$ be the Lagrange multipliers at step k.

Algorithm 4.4 Update $\mathbf{C}_i$ by an ADMM algorithm

```
Input: $\hat{\mathbf{X}}_i \in \mathbb{R}^{D\times N}$, $\mathbf{Q}_i \in \mathbb{R}^{D\times d_i}$ with $\mathbf{Q}_i^T\mathbf{Q}_i = \mathbf{I}$, $\mu, \lambda, \rho, \epsilon > 0$, maxIter.
Output: $\mathbf{C}_i \in \mathbb{R}^{d_i\times N}$.
Set $\mathbf{C}_i^{(0)}, \mathbf{F}_i^{(0)}$ and $\boldsymbol{\Psi}_i^{(0)}$ be zero. $k = 0$.
while $k <$ maxIter do
    Obtain $\mathbf{F}_i^{k+1)}$ by (4.31).
    $\mathbf{C}_i^{k+1)} = \text{soft}(\mathbf{F}_i^{(k)} - \boldsymbol{\Psi}_i^{(k)}, 1/\rho)$.
    $\boldsymbol{\Psi}_i^{(k+1)} = \boldsymbol{\Psi}_i^{k} + \rho(\mathbf{C}_i^{(k)} - \mathbf{F}_i^{(k)})$.
    if $\|\mathbf{C}_i^{(k)} - \mathbf{F}_i^{(k)}\|_{\text{inf}} \le \epsilon$, $\|\mathbf{C}_i^{k} - \mathbf{C}_i^{k-1}\| \le \epsilon$ and $\|\mathbf{F}_i^{k} - \mathbf{F}_i^{k-1}\| \le \epsilon$ then
        $k =$maxIter.
    end if
end while
```

- Obtain $\mathbf{F}_i^{(k+1)}$ by minimizing L with respect to $\mathbf{F}_i$ while the other variables and Lagrange multipliers are fixed. Compute the partial derivative of L with respect to $\mathbf{F}_i$ and set it to 0, we have

$$\mu\tilde{F}_i^{(k+1)} + (\lambda + \rho)\mathbf{F}_i^{k+1} - (\lambda\mathbf{Q}_i^T\hat{\mathbf{X}}_i + \rho\mathbf{C}_i^{(k)} + \boldsymbol{\Psi}_i^{(k)}) = 0. \tag{4.30}$$

where $\tilde{F}_i^{(k+1)}$ is obtained by normalized column of $\mathbf{F}_i^{k+1}$. Let $\mathbf{V}_i^{(k)} = \lambda\mathbf{Q}_i^T\hat{\mathbf{X}}_i + \rho\mathbf{C}_i^{(k)} + \boldsymbol{\Psi}_i^{(k)}$. Then the j-th column of $\mathbf{F}_i^{k+1}$ is given by

$$(\mathbf{F}_i^{k+1})_j = \begin{cases} \dfrac{(\mathbf{V}_i^{(k)})_j}{(\lambda+\rho)} - \dfrac{\mu}{\lambda+\rho}\dfrac{(\mathbf{V}_i^{(k)})_j}{\|(\mathbf{V}_i^{(k)})_j\|_2}, & \text{if } \|(\mathbf{V}_i^{(k)})_j\|_2 > \mu \\ 0, \text{otherwise.} \end{cases} \tag{4.31}$$

- Obtain $\mathbf{C}_i^{(k+1)}$ by minimizing L with respect to $\mathbf{C}_i$ while the other variables and Lagrange multipliers are fixed. Compute the partial derivative of $\mathbf{R}$

with respect to $\mathbf{C}_i$ and set it be 0, we have

$$\text{sgn}(\mathbf{C}_i^{(k+1)}) + \rho(\mathbf{C}_i^{(k+1)} - \mathbf{F}_i^{(k)}) + \mathbf{\Psi}_i^{(k)} = 0. \tag{4.32}$$

The solution of (4.32) is given by $\text{soft}(\mathbf{F}_i^{(k)} - \mathbf{\Psi}_i^{(k)}/\rho, 1/\rho)$.

- With $\mathbf{F}_i^{(k)}$ and $\mathbf{C}_i^{(k)}$ fixed, perform a gradient ascent update with the step size ρ.

$$\mathbf{\Psi}_i^{(k+1)} = \mathbf{\Psi}_i^{k} + \rho(\mathbf{C}_i^{(k+1)} - \mathbf{F}_i^{(k+1)}). \tag{4.33}$$

These three steps are repeated until convergence is achieved or the number of iterations exceeds a maximum iteration number. Convergence is achieved when $\|\mathbf{C}_i^{(k)} - \mathbf{F}_i^{(k)}\|_{max} \leq \epsilon$, $\|\mathbf{C}_i^{k} - \mathbf{C}_i^{k-1}\|_{max} \leq \epsilon$ and $\|\mathbf{F}_i^{k} - \mathbf{F}_i^{k-1}\|_{max} \leq \epsilon$ where ϵ denotes the error tolerance for the primal and dual residuals. In summary, Algorithm 4.4 shows the updates for the ADMM implementation of the optimization problem (4.28). We also provide a specific modified BCR algorithm to update $\mathbf{C}$, see Algorithm 4.5.

Algorithm 4.5 Update $\mathbf{C}$ by Modified BCR algorithm

```
Input: $X \in \mathbb{R}^{D\times N}$, $\mathbf{Q} = [\mathbf{Q}_1, \dots, \mathbf{Q}_K]$ with $\mathbf{Q}_i^T\mathbf{Q}_i = \mathbf{I}$ and $\mathbf{Q}_i \in \mathbb{R}^{D\times d_i}$, $\mathbf{C} = [\mathbf{C}_1^T, \dots, \mathbf{C}_K^T]^T$ with $\mathbf{C}_i \in \mathbb{R}^{d_i\times D}$, $\mu_0, \lambda_0, \rho, \epsilon > 0$.
Output: $\mathbf{C} \in \mathbb{R}^{d\times D}$.
for $j = 1, \dots, M$ do
    for $i = 1, \dots, K$ do
        $\hat{\mathbf{X}}_i = \mathbf{X} - \sum_{l\neq i} \mathbf{Q}_i\mathbf{C}_i$.
        Obtain $\mathbf{C}_i$ by applying Algorithm 4.4 with $\lambda = \lambda_0 M/(M-j+1)$ and $\mu = \mu_0 M/(M-j+1)$.
    end for
end for
```

2. Assuming $\mathbf{C}$ is fixed, we find $\mathbf{Q}$ by solving

$$\begin{array}{rl} \arg\min\limits_{\mathbf{Q}} & \|\mathbf{X} - \mathbf{Q}\mathbf{C}\|_F^2 \\ s.t & \mathbf{Q} = [\mathbf{Q}_1, \dots, \mathbf{Q}_K] \\ & \mathbf{Q}_i^T\mathbf{Q}_i = \mathbf{I}, i = 1, \dots, K \end{array} \tag{4.34}$$

To update $\mathbf{Q}$, we update $\mathbf{Q}_i$ separately as given by Theorem 4.1.2. We give the details in Algorithm 4.6.

Algorithm 4.6 Update $\mathbf{Q}$

```
Input: $\mathbf{X} \in \mathbb{R}^{D\times N}$ and $\mathbf{C} = [\mathbf{C}_1^T, \ldots, \mathbf{C}_K^T]$ with $\mathbf{C}_i \in \mathbb{R}^{d_i \times N}$, $\mathbf{Q} = [\mathbf{Q}_1, \ldots, \mathbf{Q}_K]$ with $\mathbf{Q}_i \in \mathbb{R}^{D\times d_i}$.
Output: $\mathbf{Q} = [\mathbf{Q}_1, \ldots, \mathbf{Q}_K]$.
for $i = 1, \ldots, K$ do
    $\hat{\mathbf{X}}_i = \mathbf{X} - \sum_{j\neq i} \mathbf{Q}_j \mathbf{C}_j$.
    Compute the SVD of $\hat{\mathbf{X}}_i \mathbf{C}_i^T = \mathbf{U}_i \mathbf{\Sigma}_i \mathbf{V}_i$ with $\mathbf{U}_i \in \mathbb{R}^{D\times d_i}$, $\mathbf{V}_i \in \mathbb{R}^{d_i\times d_i}$.
    $\mathbf{Q}_i = \mathbf{U}_i \mathbf{V}_i^T$.
end for
```

4.3.2 Initialization and estimations of K and $\{d_i\}_{i=1}^K$

Note that the algorithms in Section 4.3.1 require prior knowledge of number K of subspaces and their dimensions $\{d_i\}_{i=1}^K$. However, in practice, we do not know these parameters in advance. Observe that the minimization of $\|\mathbf{C}\|_1$ in (4.24) implies a small number of dictionary atoms will be used. We can conclude that with over-estimations of d_i, we can still obtain a matrix $\mathbf{C}$ with satisfying structure.

[91] suggests an agglomerative clustering procedure to update number K of clusters, as well as regroup the dictionary atoms for each cluster. In particular, assume we have dictionary $\mathcal{Q} = \{\mathbf{q}_i\}_{i=1}^d$ and an coefficient matrix $\mathbf{C}$. A segmentation of $\mathcal{Q}$ will be obtained by grouping rows of C according to their sparsity patterns. In particular, given an initial segmentation of the dictionary $\{\mathcal{Q}_i\}_{i=1}^K$ with each $\mathcal{Q}_i$ corresponding to one cluster, merge $\mathcal{Q}_i$ and $\mathcal{Q}_j$ whose corresponding rows of $\mathbf{C}$ have most similar sparsity patterns.

Algorithm 4.7 Estimations of K and $\{d_i\}_{i=1}^K$

```
Input: $\mathcal{X}$, $\mathcal{Q}$, $\mathbf{C} \in \mathbb{R}^{d\times N}$.
Output: $K$, $\{d_i\}_{i=1}^N$, $\{\mathbf{Q}_i : \mathbf{Q}_i^T\mathbf{Q}_i = \mathbf{I}, \mathbf{Q}_i \in \mathbb{R}^{D\times d_i}\}$, $\{\mathbf{C}_i : \mathbf{C}_i \in \mathbb{R}^{d_i\times N}\}_{i=1}^K$.
Apply Algorithm 4.2 with $\mathbf{C}$ to obtain clusters $\mathcal{X}_1, \ldots, \mathcal{X}_K$ of $\mathcal{X}$, where $K$ is determined by (4.13).
Let $\mathbf{X}_i$ be a matrix which is formed by stacking all vectors in $\mathcal{X}_i$ to its columns. Suppose the index set of elements of $\mathcal{X}_i$ in $\mathcal{X}$ is $\mathcal{I}_i$.
for $i = 1, \ldots, K$ do
    Compute the SVD of $\mathbf{X}_i = \mathbf{U\Sigma}\tilde{\mathbf{V}}$ with $\mathbf{\Sigma} \in \mathbb{R}^{m\times m}$, where $r = \min\{n_i, D\}$.
    $d_i = r - \arg\max_i(\sigma_{i+1}(\mathbf{X}_i) - \sigma_i(\mathbf{X}_i))$, $\mathbf{Q}_i = \mathbf{U}(:, 1 : d_i)$, $\mathbf{C}_i(:, \mathcal{I}_i) = \mathbf{Q}_i^T\mathbf{X}_i$.
end for
```

Observe that the agglomerative clustering procedure in [91] is an iterative bipartite graph method. A more efficient and effective choice to address the bipartite graph partition problem is bipartite graph spectral clustering method which is introduced in Section 4.1.2. In particular, in the setting of subspace clustering, we can also cluster the corresponding data simultaneously and then apply PCA to obtain an orthonormal basis for each cluster.

Suppose we obtain a coefficient matrix $\mathbf{C}$ of $\mathbf{X}$ with respect to a dictionary $\mathbf{Q}$, we compute clusters based on $\mathbf{C}$ with Algorithm 4.2. Then we apply PCA to each cluster to obtain a new orthonormal basis. These orthonormal bases will then be the input to Algorithm 4.5 to obtain a better coefficient matrix $\mathbf{C}$, see Algorithm 4.7.

Remark If N and D are large, line 6 of Algorithm 4.7 will be time consuming as the complete SVD is very expensive. An alternate way is clustering the dictionary set $\mathcal{Q}$ instead of $\mathcal{X}$.

In practice, we do not need to estimate K and d_i at each step, we could update them every a few steps. After we obtain a good representation matrix $\mathbf{C}$, we can apply Algorithm 4.2 to obtain the final clustering results. We summarize the complete procedure of our method in Algorithm 4.8.

Algorithm 4.8 Subspace clustering by learning a union of orthonormal bases

```
Input:  A dataset $\mathcal{X}$ with representation $\mathbf{X}$.
Output:  Clusters $\mathcal{X}_1, \dots, \mathcal{X}_K$.
Initialization:  $\mathbf{C} = \mathbf{X}$
for $i = 1, 2, ...$ until convergence do
    Compute $\{d_i, \mathbf{Q}_i, \mathbf{C}_i\}_{i=1}^K$ by Algorithm 4.7.
    for $j = 1, 2, ..., M$ do
        Solve (4.26) by Algorithm 4.5.
        Solve (4.34) by Algorithm 4.6.
    end for
end for
Compute clusters $\mathcal{X}_1, \dots, \mathcal{X}_K$ by Algorithm 4.2.
```

4.4 Numerical examples

In this section, we present some examples to evaluate the performance of Algorithm 4.8 and compare it with SSC and LRR, two state-of-the-art subspace clustering algorithms. We use the SSC [23] and LRR [56] implementations from the authors' websites. For consistency, we normalize each data point of the dataset to unit norm for all methods.

4.4.1 Synthetic data

Consider three disjoint subspaces $\{\mathcal{S}_i\}_{i=1}^3$ of the same dimension $d = 4$ embedded in the $\mathbb{R}^{20}$. To make the problem hard enough so that every data point in a subspace can also be reconstructed as a linear combination of points from other subspaces, we generate subspace bases $\{\mathbf{U}_i \in \mathbb{R}^{D\times d}\}_{i=1}^3$ such that each subspace lies in the direct sum of the other two subspaces, i.e., $rank([\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3]) = 2d$. Simply, we set

$$\mathbf{U}_1 = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{O} \end{bmatrix} \in \mathbb{R}^{D\times d}, \quad \mathbf{U}_2 = \begin{bmatrix} \mathbf{O} \\ \mathbf{I}_d \end{bmatrix} \in \mathbb{R}^{D\times d}, \quad \mathbf{U}_3 = \begin{bmatrix} \mathbf{D}_1 \\ \mathbf{O} \\ \mathbf{D}_2 \end{bmatrix} \in \mathbb{R}^{D\times d}.$$

where $\mathbf{D}_1 = \mathrm{diag}(\cos(\theta_1), \ldots, \cos(\theta_d))$ and $\mathbf{D}_2 = \mathrm{diag}(\sin(\theta_1), \ldots, \sin(\theta_d))$ with

$$\theta_i = \theta + \frac{i-1}{d-1}(\pi/2 - \theta).$$

Thus, we can verify the effect of the smallest principal angle in the subspace recovery by changing the value of θ. To investigate the effect of the data distribution in the subspace-sparse recovery, we generate the same number of data points, N, in each subspace at random (i.e. random linear combination of the basis vectors) and change the value of N. In our experiment, we let θ vary from 0.1047 (degree 6) to 0.5235 (degree 30) and N vary from 5 to 35. For each (N, θ) group, we compute the subspace clustering error(SCE):

$$\mathrm{SCE} = \frac{\#\text{ of misclassified points}}{\text{total }\#\text{ of points}}.$$

We repeat the process 50 times with different randomly generated points with respect to each (N, θ) group and report only the average SCE for each subspace clustering algorithm, namely, Algorithm 4.8, SSC and LRR. We show the results in Figure 4.1.

Figure 4.1 shows that the performances of Algorithm 4.8 and SSC are much better than LRR, as LRR is developed with the assumption of linearly independent

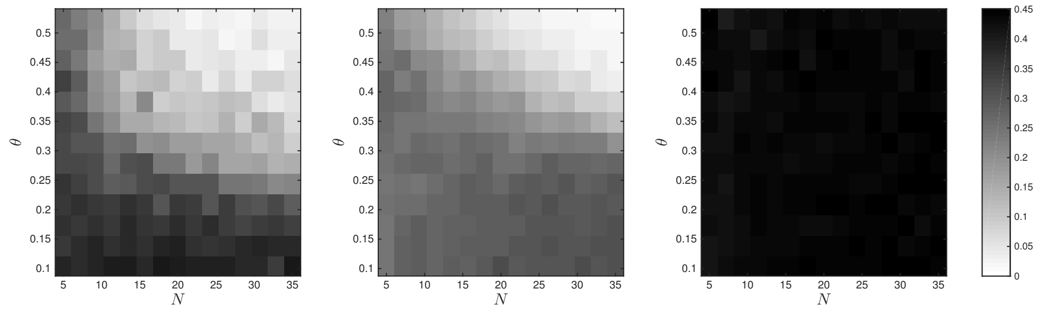

Figure 4.1: Average clustering error of Algorithm 4.8, SSC and LRR on synthetic data for each (N, θ) group. Left: Our method, Middle: SSC, Right: LRR

subspaces. When the subspaces have large intersections with each other, the representation matrix C is not low rank any more and LRR underperforms. To further compare our method and SSC, we plot the differences of SCE measure for the two methods in Fig 4.2 , where Algorithm 4.8 reports smaller SCR in white area and SSC has smaller SCE in black area. We observe that Algorithm 4.8 outperforms SSC when the values of N and θ are larger but underperforms when N and θ are smaller.

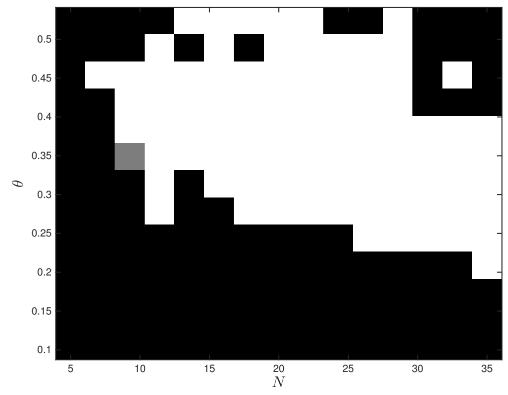

Figure 4.2: Difference of SCR for Algorithm 4.8 and SSC for each (N, θ) group. White: Algorithm 4.8 has smaller SCE. Black: SSC has smaller SCE.

4.4.2 Basic and rotated MNIST dataset

The MNIST dataset [48] is a database of binary images of 10 handwritten digits. The images are all of size 28 × 28 pixels. The database contains 60000 training images and 10000 testing images. We use them to demonstrate the performance of subspace clustering algorithms for different number K of subspaces. We let $K = 3, 6$. We randomly select K digits and randomly select $N = 100$ sample images from the training images for each digit to form a dataset. Then we apply Algorithm 4.8, SSC and LRR to the generated dataset. The process is repeated 100 times with respect to each K and we report their mean and median SCEs in Table 4.1. We notice that LRR almost has perfect clustering. It is because the subspaces of handwritten digits are approximately independent. Thus, the coefficient matrix $\mathbf{C}$ has perfect low-rank property. Our method is better than SSC in the case of $K = 3$ while SSC is slightly better when $K = 6$.

Table 4.1: SCE of Algorithm 4.8, SSC and LRR

K	Algorithm 4.8		SSC		LRR	
	average	median	average	median	average	median
K = 3	0.186	0.137	0.290	0.353	0.005	0.000
K = 6	0.390	0.403	0.356	0.367	0.018	0.000

The rotated MNIST dataset [46] is a variant of the MNIST dataset. It contains gray scale images of hand-written digits of size 28 × 28 pixels, which were originally taken from the MNIST dataset, and transformed in several ways to create more challenging classification problems. Introducing multiple factors of variation leads to four benchmarks: mnist-rot, mnist-back-rand, mnist-back-image, mnist-rot-back-image. We only consider the mnist-rot dataset, in which the images from MNIST dataset were rotated by an angle generated uniformly between 0 and 2π radians. It was shown in [33] that handwritten digits with some variations lie on 12-dimensional subspaces. Hence they can be modeled as data points lying close to a union of 12-dimensional subspaces.

We will evaluate the clustering performances of Algorithm 4.8 as well as SSC and LRR on this challenging dataset. Since the dataset contains a large number of samples, we only use samples from the training and validation sets for clustering. In particular, we randomly select 10 samples per digits and generate a small subset containing 100 samples from 10 digits. We use these samples for clustering and repeat the process 120 times with different randomly select datasets. We report the average and median clustering errors of different methods in Table 4.2. It shows Algorithm 4.8 have comparable performance as SSC and LRR in this case.

Table 4.2: Clustering performance on the rotated MNIST dataset

Dataset	Our method		SSC		LRR	
	average	median	average	median	average	median
mnist-rot	0.746	0.745	0.729	0.730	0.761	0.760

Our experiments have shown that Algorithm 4.8 achieve comparable results as SSC and LRR. In addition, Algorithm 4.8 can deal with more general datasets, in which the data points may not lie in the union of subspaces.

Chapter 5 Concluding remarks

In the dissertation, we have discussed some algorithms and related theories for the generalize eigenvalue problem, singular value problem and subspace clustering problem.

In Chapter 2, we have incorporated the deflation by restriction method into the inverse-free preconditioned Krylov subspace method to find several eigenvalues of the generalized symmetric definite eigenvalue problem. We extend the convergence analysis in [32] to justify the deflation scheme. Numerical examples confirm the convergence properties as revealed by the new theory. This deflation scheme allows implementation of the inverse-free preconditioned Krylov subspace method without using perturbations to the original problems as in the Wielandt deflation. This may be important in applications such as the singular value computation where the structure of the problems needs to be preserved.

In Chapter 3, we have presented an inverse free preconditioned Krylov subspace algorithm for computing a few of the extreme singular values of a rectangular matrix. The robust incomplete factorization (RIF) has been adapted to efficiently construct preconditioners for the shifted matrix $\mathbf{C}^T\mathbf{C} - \mu\mathbf{I}$. A preliminary MATLAB implementation has been developed and is demonstrated to be very competitive compared to other existing programs in both settings of using preconditioners or using shift-and-invert.

In Chapter 4, we have developed a novel subspace clustering algorithm. The main idea of our algorithm is to find the best orthonormal bases for the underlying subspaces as a dictionary and then finding the sparsest block structured representation of the data based on the learned dictionary. Different from some other dictionary learning based algorithms, we find the dictionary and representation of data simultaneously. This idea introduces an extra penalty term in the optimization problem (4.25), which promotes a block structure in the representation of the data with the dictionary as showed in Section 4.4.1. Numerical examples demonstrate the effectiveness of our algorithm in both synthetic and real world datasets.

Bibliography

[1] A. Adler, M. Elad, and Y. Hel-Or. Linear-time subspace clustering via bipartite graph modeling. *IEEE Transactions on Neural Networks and Learning Systems*, 2015.

[2] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54:4311–4322, 2006.

[3] J. Baglama, D. Calvetti, and L. Reichel. Algorithm 827: irbleigs: A MATLAB program for computing a few eigenpairs of a large sparse Hermitian matrix. *ACM Transactions on Mathematical Software*, 29:337–348, September 2003.

[4] J. Baglama, D. Calvetti, and L. Reichel. IRBL: An implicitly restarted block Lanczos method for large-scale Hermitian eigenproblems. *SIAM J. Sci. Comp.*, 24:1650–1677, 2003.

[5] J. Baglama and L. Reichel. Augmented implicitly restarted Lanczos bidiagonalization methods. *SIAM J. Sci. Comp.*, 27:19–42, 2005.

[6] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, PA, 2000.

[7] Z. Bai, I. S. Duff, and A. J. Wathen. A class of incomplete orthogonal factorization methods. I: methods and theories. Report 99/13, Oxford University Computing Laboratory, 1999.

[8] M. Benzi. Preconditioning techniques for large linear systems: A survey. *J. Comp. Phys.*, 182:418–477, 2002.

[9] M. Benzi, J. K. Cullum, and M. Tuma. Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM J. Sci. Comp.*, 22:1318–1332, 2000.

[10] M. Benzi and M. Tuma. A robust incomplete factorization preconditioner for positive definite matrices. *Num. Lin. Alg. Appl.*, 10:385–400, 2003.

[11] M. Benzi and M. Tuma. A robust preconditioner with low memory requirements for large sparse least squares problems. *SIAM J. Sci. Comp.*, 25:499–512, 2003.

[12] M. W. Berry. Large scale sparse singular value computations. *International Journal of Supercomputer Applications*, 6:13–49, 1992.

[13] T. E. Boult and L. G. Brown. Factorization-based segmentation of motions. *in IEEE Workshop on Motion Understanding*, pages 179–186, 1991.

[14] P. S. Bradley and O. L. Mangasarian. k-plane clustering. *Journal of Global Optimization*, 16(1):23–32, 2000.

[15] S. S. Chen, D. L. Donoho, and M. A. Saunder. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[16] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *Int. J. of Computer Vision*, 29(3), 1998.

[17] W. Dai, T. Xu, and W. Wang. Simultaneous codeword optimization (simco) for dictionary update and learning. *IEEE Transactions on Signal Processing*, 60(12):6340–6353, 2012.

[18] T. Davis and Y. Hu. The University of Florida sparse matrix collection. http://www.cise.ufl.edu/research/sparse/matrices/.

[19] J. W. Demmel. *Applied Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.

[20] H. A. Van der Vorst. Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM J. Sci. Comput*, 13:631–644, 1992.

[21] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. *Proceedings of the seventh ACM SIGKDD international conferrence on Knowledge discovery and data mining*, pages 269–274, 2001.

[22] E. Elhamifar and R. Vidal. Sparse subspace clustering. *IEEE International Conference on Computer Vision and Pattern Recogition*, 2009.

[23] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[24] K. Engan, S. O. Aase, and J. H. Husøy. Method of optimal directons for frame design. *Acoustics, Speech, and Signal Processing, 1999. Proceedings. (ICASSP '99). IEEE International Conference on*, 5:2443–2336, 1999.

[25] T. Ericsson and A. Ruhe. The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. *Mathematics of Computation*, 35:1251–1268, 1980.

[26] D. R. Fokkema, G. L. G. Sleijpen, and Henk A. Van Der Vorst. Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM J. Sci. Comput.*, 20:94–125, 1998.

[27] C. W. Gear. Multibody grouping from motion images. *Int. J. of Computer Vision*, 29(2):133–150, 1998.

[28] A. George and J. Liu. Householder reflectors versus Givens rotations in sparse orthogonal decompositions. *Lin. Alg. Appl.*, 88/89:223–238, 1987.

[29] G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis*, 2:205–224, 1965.

[30] G. H. Golub and C. F. VanLoan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[31] G. H. Golub and Q. Ye. Inexact inverse iteration for generalized eigenvalue problems. *BIT*, 40:671–684, 2000.

[32] G. H. Golub and Q. Ye. An inverse free preconditioned Krylov subspace method for symmetric generalized eigenvalue problems. *SIAM J. Sci. Comp.*, 24:312–334, 2002.

[33] T. Hastie and P. Y. Simard. Metrics and models for handwritten character recognition. *Statistical Science*, 13(1):54–65, 1997.

[34] J. Ho, M. Yang, J. Lim, K. Lee, and D. Kriegman. Clustering appearances of objects under varying illumination conditions. *CVPR*, 2003.

[35] M. E. Hochstenbach. A Jacobi-Davidson type SVD method. *SIAM J. Sci. Comput*, 23:606–628, 2001.

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

[37] M. E. Hochstenbach. Private communications, 2014.

[38] Z. Jia and D. Niu. An implicitly restarted refined bidiagonalization Lanczos method for computing a partial singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 25:246–265, 2003.

[39] L. Jing, M. K. Ng, and T. Zeng. Dictionary learning-based subspace structure identification in spectral clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 24(8), 2013.

[40] K. Kanatani. Motion segmentation by subspace separation and model selection. *IEEE International Conference on Computer Vision*, 2:586–591, 2001.

[41] A. V. Knyazev. A preconditioned conjugate gradient method for eigenvalue problems and its implementation in a subspace. In *International Ser. Numerical Mathematics, v. 96, Eigenwertaufgaben in Natur- und Ingenieurwissenschaften und ihre numerische Behandlung, Oberwolfach*, pages 143–154, Basel, 1991. Birkhauser.

[42] A. V. Knyazev. Preconditioned eigensolvers—an oxymoron? *Electron. Trans. Numer. Anal.*, 7:104–123, 1998.

[43] A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient. *SIAM J. Sci. Comp.*, 23:517–541, 2001.

[44] E. Kokiopoulou, C. Bekas, and E. Gallopoulos. Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization. *Appl. Numer. Math*, 49:39–61, 2004.

[45] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, and T. W. Lee. Dictionary learning algorithms for sparse representation. *Neural Computation*, 15(349–396), 2004.

[46] H. Larochelle, D. Erhan, A. Courvill, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. *International Conference on Machine Learning(ICML)*, pages 473–480, 2007.

[47] R. M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. *Technical report, DAIMI PB-357, Department of Computer Science, University of Aarhus, Aarhus, Denmark, also available online from http://sun.stanford.edu/ rmunk/PROPACK/*, 1998.

[48] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[49] R. Lehoucq, D. Sorenson, and C. Yang. *ARPACK Users' Guides, Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Method.* SIAM, Philadelphia, 1998.

[50] R. B. Lehoucq. *Analysis and implementation of an implicitly restarted Arnoldi iteration.* PhD thesis, Rice University, Houston, TX, 1995.

[51] R. B. Lehoucq and D. C. Sorensen. Deflation techniques within an implicitly restarted Arnoldi iteration. *SIAM J. Matrix Anal. Appl.*, 17:789–821, 1996.

[52] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya. Learning unions of orthonormal bases with thresholded singular value decomposition. *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, 5:v/293–v/296, March 2005.

[53] M. S. Lewicki and B. A. Olshausen. Probabilistic framework for the adaption and comparison of image codes. *J. Opt. Soc. Am. A*, 16(7):1587–1601, July 1999.

[54] M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–365, 2000.

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

[56] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. *International Conference on Machine Learning(ICML)*, 2010.

[57] J. Liu, Y. Chen, J. Zhang, and Z. Xu. Enhancing low-rank subspace clustering by manifold regularization. *IEEE Transactions on Image Processing*, pages 4022–4030, 2014.

[58] C. Lu, H. Min, Z. Zhao, L. Zhu, D. Huang, and S. Yan. Robust and efficient subspace segmentation via least squares regression. *12th European Confereence on Computer Vision*, 2012.

[59] Y. Ma, H. Derken, W. Hong, and J. Wright. Segmentation of multivariate mixed data via lossy coding and compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1546–1562, 2007.

[60] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[61] Matrix Market. http://math.nist.gov/MatrixMarket/.

[62] J. H. Money and Q. Ye. Algorithm 845: EIGIFP: A MATLAB program for solving large symmetric generalized eigenvalue problems. *ACM Trans. Math. Softw.*, 31:270–279, 2005.

[63] R. B. Morgan. Computing interior eigenvalues of large matrices. *Lin. Alg. Appl.*, 74:1441–1456, 1991.

[64] L. Na and Y. Saad. MIQR: a multilevel incomplete QR preconditioner for large sparse least-squares problems. *SIAM J. Matrix Anal. Appl.*, 28:524–550, 2006.

[65] Y. Notay. Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem. *Num. Lin. Alg. Appl.*, 9:21–44, 2002.

[66] A. T. Papadopoulos, I. S. Duff, and A. J. Wathen. Incomplete orthogonal factorization methods using Givens rotations II: Implementation and results. Report 02/07, Oxford University Computing Laboratory, 2002.

[67] B. N. Parlett. *The Symmetric Eigenvalue Problem*, volume 20 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. Corrected reprint of the 1980 original.

[68] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, 6(1):90–105, 2004.

[69] V. M. Patel, H. Van Nguyen, and R. Vidal. Latent space sparse subspace clustering. *IEEE Internation Conference on Computer Vision(ICCV)*, 2013.

[70] P. Quillen and Q. Ye. A block inverse-free preconditioned Krylov subspace method for symmetric generalized eigenvalue problems. *J. Comp. Appl. Math*, 233:1298–1313, 2010.

[71] Y. Saad. *Numerical methods for large eigenvalue problems, revised edition.* Classics in Applied Mathematics. SIAM, Philadelphia, 2011.

[72] S. Sardy, A. G. Bruce, and P. Tseng. Block coordinate relaxation methods for nonparametric signal denoising with wavelet dictionaries. *Journal of Computational and Graphical Statistics*, 9:361–379, 2000.

[73] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[74] G. Sleijpen and H. van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 17:401–425, 1996.

[75] G. Sleijpen, H. van der Vorst, and M. van Gijzen. Efficient expansion of subspaces in the Jacobi-Davison method for standard and generalized eigenproblems. *Electron. Trans. Numer. Anal.*, 7:75–89, 1998.

[76] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[77] A. Stathopoulos, Y. Saad, and K. Wu. Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods. *SIAM J. Sci. Comput*, 19:227–245, 1996.

[78] M. Tipping and C. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[79] P. Tseng. Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 105(1):249–252, 2000.

[80] S. Varadhan, M. W. Berry, and G. H. Golub. Approximating dominant singular triplets of large sparse matrices via modified moments. *Numer. Algorithms*, 13:123–152, 1996.

[81] R. Vidal. Subspace clustering. *Signal Processing Magazine*, 2011.

[82] R. Vidal and P. Favaro. Low rank subspace clustering (lrsc). *Pattern Recognition Letters*, 43:47–61, 2014.

[83] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis(gpca). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1–15, 2005.

[84] R. Vidal, R. Tron, and R. Hartley. Multiframe motion segmentation with missing data using powerfactorization and gpca. *Internation Journal of Computer Vision*, 79(1):85–105, 2008.

[85] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[86] X. Wang, K. A. Gallivan, and R. Bramley. CIMGS: An incomplete orthogonal factorization preconditioner. *SIAM J. Sci. Comput.*, 18:516–536, 1997.

[87] Y. Wang, H. Xu, and C. Leng. Provable subspace clustering: When lrr meets ssc. *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 2013.

[88] Andrew R. Webb. *Statistical Pattern Recognition, Second Edition.* John Wiley & Sons, Ltd., 2002.

[89] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford University Press, New York, 1965.

[90] Q. Ye. An adaptive block Lanczos algorithm. *Numerical Algorithms*, 12:97–110, 1996.

[91] L. Zelnik-Manor, K. Rosenblum, and Y. C. Eldar. Dictionary optimization for block sparse representation. *IEEE Transactions on Signal Processing*, pages 2386–2395, 2012.

[92] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. *Proceedings of the tenth international conference on Information and knowledge management*, pages 25–32, 2001.

[93] T. Zhang, A. Szlam, and G. Lerman. Median k-flats for hybrid linear modeling with many outliers. *IEEE 12th International Conference on Computer Vision Workshops*, pages 234–241, 2009.

Vita

Education

- University of Kentucky, Lexington, Kentucky
  M. A. in Mathematics, May, 2013
- University of Science and Technology of China, Hefei, Anhui, China
  B. S., Mathematics and Applied Mathematics

Experience

- Research Assistant under Dr. Qiang Ye, University of Kentucky, Fall 2012, Fall 2013, Spring 2014, Fall 2014, Summer 2015
- Teaching Assistant, University of Kentucky, August 2011 - May 2015
- Software Engineering Intern, Google, Pittsburgh, PA, May 2014 - August 2014

Publications

- (with Qi. Ye) Deflation by restriction for the inverse-free precondtioned Krylov subspace method, *submitted*.
- (with Q. Ye) Computing singular values of large matrices with inverse free preconditioned Krylov subspace method, *Electronic Transactions on Numerical Analysis*, 42:197-221, 2014.