A numerical algorithm for the *k*-th eigenvalue problem of large matrices and its applications

LEE Dongjin

Doctoral Thesis

Submitted to
Department of Computational Science and Engineering,
Graduate School of Engineering,
Nagoya University

January 2019

Abstract

For a given index k, this thesis presents a numerical algorithm for computing the k-th eigenvalue and its associated eigenvector of a Hermitian definite generalized eigenproblem of large sparse matrices, or the k-th eigenvalue problem, and its applications.

Eigenvalue problems are a fundamental problem in numerical linear algebra, and their numerical solution is essential to various areas of computational science and engineering. Of practical interest is often a subset of eigenvalues and eigenvectors, and a variety of algorithms have been proposed for or have been found to be effective for computing a specific type of subsets, such as a small number of eigenvalues closest to a given point and their associated eigenvectors.

The k-th eigenvalue problem has originated from electronic structure calculations of materials, where the eigenvalue and eigenvector of a material-specific index k play a fundamental role in the research of various material properties. The k-th eigenvalue and its associated eigenvector do not fall into a typical subset at which the existing algorithms are aimed, and all or a substantial number of eigenvalues and eigenvectors have been computed to obtain the k-th eigenvalue and eigenvector.

This thesis proposes an efficient algorithm for computing the k-th eigenvalue and eigenvector with validation of their index, based on Sylvester's law of inertia and the bisection algorithm and by utilizing the Lanczos method and a sparse direct linear solver. Closely related to eigenvalue problems are singular value problems, and the k-th singular value and its associated left and right singular vectors of a large sparse matrix are computed based on the proposed algorithm. Contributions of the thesis are discussed from the perspective of Real-World Data Circulation.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Shao-Liang Zhang for his enduring guidance and encouragement. He has provided me with countless insights and opportunities throughout my years of undergraduate and graduate studies at Nagoya.

I sincerely appreciate Prof. Takeshi Furuhashi and Prof. Takahiro Katagiri for being a member of the thesis committee. Their valuable suggestions and constructive advice led to improvements in the thesis.

I would like to thank Prof. Takeo Hoshi for introducing me to the k-th eigenvalue problem, the main focus of the thesis, and for providing insightful advice from the perspectives of materials science and applied mathematics. I have been able to learn much from discussions with Prof. Tomohiro Sogabe and am deeply indebted to him for his tireless guidance, more than helpful advice, and constant encouragement. I am very grateful to Prof. Yuto Miyatake and Prof. Tomoya Kemmochi for their advice on mathematical thinking and technical writing. I would like to acknowledge fruitful discussions with Prof. Takafumi Miyata from which the underlying ideas of this work have come. I wish to thank Dr. Hiroto Imachi and the members of Prof. Hoshi's research group for providing me with results of large-scale computations and for their hospitality during my stay in Tottori. I appreciate Dr. Kanoko Yoshizawa and Mr. Gilles Gouaillardet for their advice on parallel computing. I thank Mr. Fuminori Tatsuoka and all other members of Prof. Zhang's research group for their friendship and appreciate the current and former secretaries of the research group, Ms. Kaori Iwata and Ms. Yumi Kobayashi, for their support.

I sincerely appreciate Prof. Kazuya Takeda, Prof. Mutsunori Yagiura, and Prof. Naoki Nishida for reviewing the thesis from the perspective of Real-World Data Circulation (RWDC) and for providing helpful advice to improve the presentation of the thesis.

I am very grateful to Prof. Mehrdad Teratani for being my mentor of the Graduate Program for RWDC Leaders, one of the Nagoya University Leading Graduate School Programs by MEXT, Japan, and thank the RWDC students for their friendship. Last but not least, I would like to express my deep appreciation to the RWDC Program for providing me with generous support and valuable opportunities during my years of graduate study.

Contents

List of Symbols and Acronyms

Symbols

$\alpha,\ldots,\omega$	scalars
$a,\ldots,z$	integers
$\boldsymbol{a},\ldots,\boldsymbol{z}$	vectors
$\mathbf{0}$	zero vector
$A,\ldots,Z$	matrices
O	zero matrix
I	identity matrix
A^{-1}	inverse of A
A^{T}	transpose of A
$\bar{A}$	complex conjugate of A
A^{H}	conjugate transpose of A
$A^{-\mathrm{H}}$	inverse of A^{H}, or equivalently conjugate transpose of A^{-1}
$A > B$	$A - B$ is positive definite, provided that A and B are Hermitian
A_{ij}	matrix element in the i-th row and the j-th column of A
$\mathrm{diag}(\alpha_1,\alpha_2,\ldots)$	diagonal matrix whose i-th diagonal element is α_i
$\mathbb{N}$, $\mathbb{R}$, $\mathbb{C}$	sets of natural, real, and complex numbers
$\mathbb{R}^n$, $\mathbb{C}^n$	n–dimensional real and complex coordinate spaces
$\lVert\boldsymbol{x}\rVert_p$, $\lVert A\rVert_p$	vector and matrix p–norms
$\lVert A\rVert_{\mathrm{F}}$	Frobenius norm
$(\boldsymbol{x},\boldsymbol{y})_A$	A–inner product $\boldsymbol{x}^{\mathrm{H}}A\boldsymbol{y}$, provided that A is Hermitian positive definite
$\lVert\boldsymbol{x}\rVert_A$	A–norm $\sqrt{(\boldsymbol{x},\boldsymbol{x})_A}$
$\Lambda(A)$	set of eigenvalues of A
$O(\cdot)$	asymptotic upper bound

Acronyms

FLOP	floating point operation
GEP	generalized eigenvalue problem
HDGEP	Hermitian definite generalized eigenvalue problem
HEP	Hermitian eigenvalue problem
k-EP	k-th eigenvalue problem
k-SVP	k-th singular value problem
SEP	standard eigenvalue problem
SVD	singular value decomposition
SVP	singular value problem

Chapter 1

Introduction

1.1 Eigenvalue problems

Eigenvalue problems are a fundamental problem in numerical linear algebra. In their general form, eigenvalue problems can be written as follows:

$$A(\lambda)\boldsymbol{x} = \boldsymbol{0}. \tag{1.1}$$

Here, $A : \mathbb{C} \to \mathbb{C}^{n\times n}$ is a matrix-valued function. Scalars $\lambda \in \mathbb{C}$ and nonzero vectors $\boldsymbol{x} \in \mathbb{C}^n$ satisfying (1.1) are referred to as eigenvalues and eigenvectors, respectively. The pairs $(\lambda, \boldsymbol{x})$ are referred to as eigenpairs. The goal of solving eigenvalue problems is to find all or a subset of eigenvalues, eigenvectors, or eigenpairs.

Eigenvalues problems are classified into several classes by their structure and properties. A *standard eigenvalue problem* (SEP), also referred to as simply an *eigenvalue problem*, is of the simplest structure:

$$A\boldsymbol{x} = \lambda\boldsymbol{x}.$$

If matrix $A \in \mathbb{C}^{n\times n}$ has special properties, eigenvalues and eigenvectors enjoy attractive features that make the problem more tractable than the case of general A. For example, if A is Hermitian, the matrix has n real eigenvalues and n eigenvectors that can be selected to be orthogonal to each other, which has made possible to develop numerous effective algorithms to solve the problem. This class of SEP with Hermitian A is referred to as a *Hermitian eigenvalue problem* (HEP).

Closely related to HEP is a *singular value problem* (SVP):

$$\begin{cases} A\boldsymbol{v} = \sigma\boldsymbol{u}, \\ \boldsymbol{u}^{\mathrm{H}}A = \sigma\boldsymbol{v}. \end{cases} \tag{1.2}$$

Here, $A \in \mathbb{C}^{m\times n}$. Non-negative scalars $\sigma \in \mathbb{R}$ and nonzero vectors $\boldsymbol{u} \in \mathbb{C}^m, \boldsymbol{v} \in \mathbb{C}^n$ satisfying (1.2) are referred to as singular values and left and right singular vectors, respectively. The triplets $(\sigma, \boldsymbol{u}, \boldsymbol{v})$ are referred to as singular triplets. SVP of a general matrix A leads to HEP. Gram matrices $A^{\mathrm{H}}A$ and AA^{H} have σ as their eigenvalues and have $\boldsymbol{v}$ and $\boldsymbol{u}$ as their eigenvectors, respectively. The extended matrix $\left[\begin{smallmatrix} O & A \\ A^{\mathrm{H}} & O \end{smallmatrix}\right]$ has $\pm\sigma$ as its eigenvalues and $\left[\begin{smallmatrix} \pm\boldsymbol{u} \\ \boldsymbol{v} \end{smallmatrix}\right]$ as its eigenvectors. Because of this relation between SVP and HEP, algorithms for HEP have been applied to solution of SVP, and vice versa.

A straightforward generalization of SEP is a *generalized eigenvalue problem* (GEP):

$$A\boldsymbol{x} = \lambda B\boldsymbol{x}.$$

Here, $A, B \in \mathbb{C}^{n\times n}$. GEP of A and B is also referred to as the eigenvalue problem of linear *matrix pencil* (A, B), which is a family of matrices $A - zB$, $z \in \mathbb{C}$. If B equals the identity matrix I, GEP is nothing but SEP of A. If B is of full rank, GEP can be reduced to SEP of $B^{-1}A$. In the case of rank-deficient B, GEP has notable differences with SEP [5, Chapter 8.7], such as possible appearance of infinite eigenvalues. Similar to SEP, GEP becomes more tractable if A and B have special properties. For example, if A is Hermitian and B is Hermitian positive definite, the pencil has n real eigenvalues and n eigenvectors that can be selected to be orthogonal to each other with respect to *B-inner product* $(\boldsymbol{a}, \boldsymbol{b})_B = \boldsymbol{a}^{\mathrm{H}} B \boldsymbol{b}$. This class of GEP is referred to as a *Hermitian definite generalized eigenvalue problem* (HDGEP). HDGEP can be reduced to HEP of, e.g., $\tilde{A} = L^{-1}AL^{-\mathrm{H}}$ in which L is the Cholesky factor of positive definite $B = LL^{\mathrm{H}}$. Eigenpairs of Hermitian $\tilde{A}$ are $(\lambda, L^{\mathrm{H}}\boldsymbol{x})$. Because of this relation between HDGEP and HEP, algorithms for HDGEP have been developed on the basis of those for HEP in general, and algorithms for HEP have been applied to solution of HDGEP.

Further generalizations of SEP and GEP include *quadratic eigenvalue problems* with $A(\lambda) = A + \lambda B + \lambda^2 C \in \mathbb{C}^{n\times n}$, *polynomial eigenvalue problems* with $A(\lambda) = \sum_{i=0}^{d} \lambda^i A_i \in \mathbb{C}^{n\times n}$, other classes of *nonlinear eigenvalue problem* [6, 63], and *multiparameter eigenvalue problems* [25].

1.2 Existing algorithms

This thesis is about HDGEP of large sparse A and B of size $n \gtrsim 10^6$. Studies of algorithms for solving eigenvalue problems date back at least to work of Jacobi in 1846, and a variety of numerical algorithms have been proposed for or can be applied to computing all or a specific subset of eigenpairs of HDGEP [5, 57].

A standard way to compute all eigenpairs is to reduce HDGEP to an equivalent HEP of $\tilde{A} = L^{-1}AL^{-\mathrm{H}}$ by using the Cholesky factorization of $B = LL^{\mathrm{H}}$ (Section 1.1) and compute eigenpairs of the equivalent HEP. The reduction of HDGEP to the equivalent HEP does not preserve sparsity structure of pencil (A, B) of HDGEP in general, and the resulting Hermitian matrix $\tilde{A}$ can be much denser than the pencil and may become fully dense even if A and B are sparse. Eigenpairs of dense matrix $\tilde{A}$ are usually computed in a three-stage manner. The first stage reduces $\tilde{A}$ to tridiagonal matrix T in a finite number of algebraic operations by, e.g., Householder transformations, which requires $O(n^3)$ times of floating point operations (FLOPs). The second stage computes eigenpairs of T iteratively by, e.g., the QR iteration, the divide-and-conquer method, or MRRR, which requires $O(n^2)$ to $O(n^3)$ FLOPs depending on algorithms [15]. The third stage is for the back-transformation of eigenvectors of T to those of A, requiring $O(n^3)$ FLOPs.

A subset of eigenpairs of large sparse pencil (A, B) are generally computed by algorithms based on subspace projection, often referred to as subspace methods. Subspace methods generate a sequence of subspaces and project the pencil to the subspaces in order to find a specific subset of eigenvectors from the subspaces and their associated eigenvalues. Subspace projection involves two subspaces in general. One is the right subspace, or the search subspace, from which approximate eigenvectors are constructed. In other words, an approximate eigenvector is a linear combination of a basis of the search subspace. The other is the left subspace, or the test subspace, to which a residual of approximate eigenpairs is imposed to be orthogonal. An approximate eigenvalue and the coefficients for the linear combination are determined from this orthogonality constraint. The two subspaces for the projection can be either identical (Ritz–Galerkin approach) or different (Petrov–Galerkin approach). By the subspace projection, the original pencil (A, B) is reduced to a small pencil whose size is equal to the dimension of the subspaces, and all eigenpairs of the small pencil are computed by the aforementioned dense procedures and algorithms.

Subspace methods differ from one another mainly in their generation of the search subspace. For example, a Krylov subspace and its B-orthonormal basis are generated in Lanczos method [43] and the shift-and-invert (SI) Lanczos method [18] in order to find extreme (smallest or largest) eigenvalues or eigenvalues closest to a given point and their associated eigenvectors. Jacobi–Davidson method [59] solves its correction equation to find a new search vector and to expand a subspace and is capable of computing extreme eigenvalues and eigenvalues closest to a given point and their associated eigenvectors. Sakurai–Sugiura method [58] utilizes a contour integral to extract the eigenspace (span of the eigenvectors) associated with the eigenvalues in the region bounded by the contour.

1.3 The k-th eigenvalue problem

Among all eigenpairs of large sparse HDGEP, this thesis is aimed at the k-th eigenpair $(\lambda_k, \boldsymbol{x}_k)$,

$$A\boldsymbol{x}_k = \lambda_k B\boldsymbol{x}_k, \tag{1.3}$$

where $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. It is assumed here that given index k satisfies $1 \ll k \ll n$ such that λ_k is close to neither λ_1 nor λ_n. This type of HDGEP is referred to as *the k-th eigenvalue problem* (k-EP) in this thesis.

The study of k-EP has been originally motivated by computational materials science. Specifically, k-EP has arisen in large-scale electronic structure calculations [33], where eigenvalues correspond to the energy of an electron and eigenvectors represent an electronic wave function. In such applications, eigenpairs are associated with either occupied or unoccupied states that have a different physical meaning. The occupied states include $(\lambda_1, \boldsymbol{x}_1), \dots, (\lambda_k, \boldsymbol{x}_k)$, while $(\lambda_{k+1}, \boldsymbol{x}_{k+1}), \dots, (\lambda_n, \boldsymbol{x}_n)$ belong to the unoccupied states. The target eigenpair $(\lambda_k, \boldsymbol{x}_k)$ is associated with the highest occupied state whose index k is a material-specific value and is approximately 10–50% of the matrix size [49, 66], e.g., $(n, k) = (2\,040\,000, 1\,020\,000)$, thereby satisfying $1 \ll k \ll n$. The index of the eigenpair must be validated because several of the physical properties of materials, such as those of optoelectronic device materials, are governed by the eigenpair with the material-specific index k. In particular, mistakenly computing another eigenvector, e.g., $\boldsymbol{x}_{k-1}$ or $\boldsymbol{x}_{k+1}$, rather than $\boldsymbol{x}_k$ leads to a completely unreliable result because eigenvectors are B-orthogonal to each other. A further explanation of eigenvalue problems in electronic structure calculations is provided in Chapter 5.

This thesis also considers computing the k singular triplet $(\sigma_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$,

$$A\boldsymbol{v}_k = \sigma_k \boldsymbol{u}_k, \tag{1.4}$$

of large sparse $A \in \mathbb{C}^{m\times n}$ ($m \geq n$). Here, the singular values are index in *decreasing order*, or $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$, and a given index k satisfies $1 \ll k \ll n$ such that σ_k is not at either end of $[\sigma_n, \sigma_1]$. This type of SVP is referred to as *the k-th singular value problem* (k-SVP) in this thesis and is studied as a special case of k-EP, based on relation between HDGEP, HEP, and SVP (Section 1.1).

1.4 Contributions

The k-th eigenpair of large sparse HDGEP is different from typical subsets of eigenpairs that can be computed by existing subspace methods. Some eigenvalues at the ends of $[\lambda_1, \lambda_n]$ and their associated eigenvectors can be computed by the Lanczos [43] and LOBPCG [39] methods, and some eigenvalues closest to a given target point and their eigenvectors are computed by the SI Lanczos [18] and Jacobi–Davidson [59] methods. In addition, eigenvalues in a given target interval and their eigenvectors are computed by the Sakurai–Sugiura method [58], FEAST method [56], and filtering methods [48]. However, none of these methods aim at computing the eigenpair of a given target index k with $1 \ll k \ll n$.

One possibility to compute the k-th eigenpair with validation of its index is computing all or substantial portion of eigenpairs based on dense linear algebra of $O(n^3)$ FLOPs in massively parallel environment. Recently, a one-million-dimensional generalized eigenvalue problem, the world's largest, was solved by a dense eigensolver [17, 35] using the full K computer. The elapsed time was

5516 seconds to compute all eigenpairs [30], indicating the practical size limit of eigenpair computation by a dense eigensolver. This thesis proposes a numerical algorithm for k-EP which requires significantly less execution time and computational resources than dense eigensolvers and is able to cope with problems of size $n \gtrsim 10^6$.

1.5 Relation to data circulation

Contributions of this thesis are discussed in terms of their role in the context of Real-World Data Circulation (RWDC)[1]. The concept of RWDC can be briefly explained as follows. To begin with, a set of data is acquired by measurement and observation of natural phenomena and social activities, or by experiments in short, using specific devices and systems. Next, the acquired data set is analysed to figure out useful information for scientific discoveries and social decision makings, or to figure out underlying principles of the data, improving our or computers' understanding of the real-world. Then, the improved understanding of the nature and society is utilized to predict and control the real-world by means of products and services implemented in the world, completing a circulation of real-world data. All three processes of data acquisition, data analysis, and data implementation are indispensible to the circulation.

The thesis is focused on a new type of matrix eigenvalue problems, which has emerged from large-scale simulations (numerical experiments) of materials, and proposes a numerical algorithm for solving the problems. By the proposed algorithm, physical quantities of industrial importance are able to be computed for more large-scale materials while requiring significantly less execution time and computational resources than currently available algorithms. In materials simulations, being larger in scale means being more accurate and realistic, and the proposed algorithm enables generation of more accurate and realistic materials data in a fast and efficient manner, which can lead to the acceleration and scaling-up of computational and data-driven studies of materials.

1.6 Outline

The rest of this thesis is organized as follows. Chapter 2 provides preliminaries for the subsequent chapters, including a priori and a posteriori eigenvalue bounds, Sylvester's law of inertia, the bisection algorithm, and the Lanczos method. Chapter 3 brings together theoretical background and algorithms explained in the previous chapter in order to propose a numerical algorithm for k-EP. Chapter 3 is based on the following papers and is the main contribution of the thesis.

[44] Lee, D., Hoshi, T., Sogabe, T., Miyatake, Y. and Zhang, S.-L., Solution of the k-th eigenvalue problem in large-scale electronic structure calculations, J. Comput. Phys., **371** (2018), 618–632.

[46] Lee, D., Miyata, T., Sogabe, T., Hoshi, T. and Zhang, S.-L., An interior eigenvalue problem from electronic structure calculations, Jpn. J. Ind. Appl. Math., **30** (2013), 625–633.

Chapter 4 is about solution of k-SVP, which is an application of the proposed algorithm in the previous chapter and is based on the following paper.

[47] Lee, D., Sogabe, T., Miyatake, Y. and Zhang, S.-L., On computing the k-th singular triplet (in Japanese), Trans. Jpn. Soc. Ind. Appl. Math., accepted.

Chapter 5 discusses contributions of the thesis from the perspective of RWDC. Chapter 6 summarizes the thesis with remarks on future work.

[1]RWDC is a systematic study of integrating data acquisition, data analysis, and data implementation processes to generate circulation of real-world data for creation of novel social values, initiated by the Graduate Program for Real-World Data Circulation Leaders [20], one of the Nagoya University Leading Graduate School Programs.

Chapter 2

Preliminaries

This chapter provides preliminaries for the subsequent chapters. Section 2.1 considers HEP and explains some classical eigenvalue bounds, the inertia of a Hermitian matrix, and inertia computation. Section 2.2 considers HDGEP and explains the bisection algorithm and the Lanczos method.

2.1 Hermitian eigenvalue problems

This section considers HEP of matrix $A \in \mathbb{C}^{n\times n}$:

$$A\boldsymbol{x} = \lambda \boldsymbol{x}.$$

The eigenvalues of A are the roots of the *characteristic equation* $\det(A-\lambda I) = 0$. An eigenvalue λ has *algebraic multiplicity* m if λ is a root of multiplicity m of $\det(A-\lambda I) = 0$. Eigenvalues are *simple* if their algebraic multiplicity equals one. Otherwise, eigenvalues are *multiple*. The set of all eigenvalues is the *spectrum* of A, denoted $\Lambda(A)$. Nontrivial solution $\boldsymbol{x}$ of linear system $(A-\lambda I)\boldsymbol{x} = \boldsymbol{0}$ is an eigenvector associated with λ. The null space of $A-\lambda I$ is the *eigenspace* associated with λ, and its dimension is the *geometric multiplicity* of λ. All eigenvalues of a Hermitian matrix have the same geometric and algebraic multiplicities, or are *semi-simple*, and thus Hermitian matrices are *diagonalizable* by the eigenvectors.

All eigenvalues of Hermitian A are real and are indexed in the increasing order, or $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$, in the thesis unless otherwise stated. Eigenvectors are indexed accordingly such that $A\boldsymbol{x}_i = \lambda_i \boldsymbol{x}_i$. Eigenvectors associated with distinct eigenvalues, or $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with $\lambda_i \ne \lambda_j$, are orthogonal. Eigenvectors associated with the same eigenvalue, or $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with $\lambda_i = \lambda_j,\ i \ne j$, can be selected to be orthogonal, leading to *unitary diagonalization* $X^{\mathrm{H}}AX = \Lambda$ and *spectral decomposition* $A = X\Lambda X^{\mathrm{H}}$. Here, $\Lambda = \mathrm{diag}(\lambda_1,\ldots,\lambda_n)$, and $X = [\boldsymbol{x}_1 \ldots \boldsymbol{x}_n]$ is unitary.

The eigenvalues of a Hermitian matrix have the following variational characterization from which the index of eigenvalues can be defined.

Definition 2.1 (Rayleigh quotient). Let $A \in \mathbb{C}^{n\times n}$ be Hermitian and $\boldsymbol{x} \in \mathbb{C}^n$ be nonzero. The Rayleigh quotient $\rho(A,\boldsymbol{x})$ of A and $\boldsymbol{x}$ is $\rho(A,\boldsymbol{x}) = \frac{\boldsymbol{x}^{\mathrm{H}}A\boldsymbol{x}}{\boldsymbol{x}^{\mathrm{H}}\boldsymbol{x}}$.

Theorem 2.2 (Courant–Fischer [27, Theorem 4.2.6]). *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian and $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ be its eigenvalues. Let $i \in \{1,\ldots,n\}$ and S_i denote any i-dimensional subspace of $\mathbb{C}^n$. Then,*

$$\lambda_i = \min_{S_i} \max_{\boldsymbol{x}\in S_i\setminus\{\boldsymbol{0}\}} \rho(A,\boldsymbol{x}) = \max_{S_{n-i+1}} \min_{\boldsymbol{x}\in S_{n-i+1}\setminus\{\boldsymbol{0}\}} \rho(A,\boldsymbol{x}).$$

2.1.1 Eigenvalue bounds

This subsection explains some classical results on eigenvalue location. The first result to introduce is by Gershgorin, which provides an a priori bound of the spectrum of a matrix.

Definition 2.3 (row sum and column sum)**.** Let $A \in \mathbb{C}^{m\times n}$. Let $i \in \{1,\dots,m\}$ and $j \in \{1,\dots,n\}$. Row sums $r_i(A) = \sum_j \left|A_{ij}\right|$. Deleted row sums $r_i'(A) = r_i(A) - |A_{ii}|$. Column sums $c_j(A) = \sum_i \left|A_{ij}\right|$. Deleted column sums $c_j'(A) = c_j(A) - \left|A_{jj}\right|$.

Theorem 2.4 (Gershgorin [27, Theorem 6.1.1])**.** *Let $A \in \mathbb{C}^{n\times n}$ and $i \in \{1,\dots,n\}$. Consider discs $G_i(A) = \left\{x \in \mathbb{C} \;\middle|\; |x - A_{ii}| \le r_i'(A)\right\}$. Then, spectrum $\Lambda(A)$ is included in the union of the n discs, $G(A) = \bigcup_i G_i(A)$.*

Because $A \in \mathbb{C}^{n\times n}$ and A^{T} have the same eigenvalue, by applying Theorem 2.4 to A^{T}, it follows that $\Lambda(A) \subset G(A) \bigcap G(A^{\mathrm{T}})$. Here, $G(A^{\mathrm{T}}) = \bigcup_i G_i(A^{\mathrm{T}})$, where $G_i(A^{\mathrm{T}}) = \left\{x \in \mathbb{C} \;\middle|\; |x - A_{ii}| \le c_i'(A)\right\}$. If A is Hermitian, since its diagonal elements A_{ii} are real and $r_i'(A) = c_i'(\bar{A})$, it follows that $G(A) = G(A^{\mathrm{T}})$. In addition, since its eigenvalues are real, spectrum $\Lambda(A) \subset \mathbb{R} \bigcap G(A)$.

The next result provides an a posteriori eigenvalue error bound in terms of the residual $\boldsymbol{r} = A\tilde{\boldsymbol{x}} - \tilde{\lambda}\tilde{\boldsymbol{x}}$ of approximate eigenpairs $(\tilde{\lambda}, \tilde{\boldsymbol{x}})$.

Theorem 2.5 (Bauer–Fike [27, Theorem 6.3.14])**.** *Let $A \in \mathbb{C}^{n\times n}$ be diagonalizable and suppose that $A = X\Lambda X^{-1}$ with diagonal Λ. Let $\tilde{\lambda} \in \mathbb{C}$ and $\tilde{\boldsymbol{x}} \in \mathbb{C}^n$ be nonzero. Then, there exists an eigenvalue λ of A such that*

$$\left|\lambda - \tilde{\lambda}\right| \le \frac{\|X\|_2}{\|X^{-1}\|_2} \cdot \frac{\|A\tilde{\boldsymbol{x}} - \tilde{\lambda}\tilde{\boldsymbol{x}}\|_2}{\|\tilde{\boldsymbol{x}}\|_2}.$$

For Hermitian A, since it is diagonalizable by unitary X and $\|X\|_2 = 1$, it follows from Theorem 2.5 that there exist an eigenvalue λ of A such that

$$\left|\lambda - \tilde{\lambda}\right| \le \frac{\|A\tilde{\boldsymbol{x}} - \tilde{\lambda}\tilde{\boldsymbol{x}}\|_2}{\|\tilde{\boldsymbol{x}}\|_2} \tag{2.1}$$

for any $\tilde{\lambda} \in \mathbb{C}$ and nonzero $\tilde{\boldsymbol{x}} \in \mathbb{C}^n$. Therefore, the small residual norm implies that an approximate eigenvalue is close to an eigenvalue. Note that the small residual norm does not imply that an approximate eigenvector is close to an eigenvector [27, p. 411].

The last result is about interlacing of eigenvalues by adding or deleting a row and column of a Hermitian matrix.

Theorem 2.6 (Cauchy [27, Theorem 4.3.17])**.** *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian and $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ be its eigenvalues. Let $\boldsymbol{y} \in \mathbb{C}^n$ and $a \in \mathbb{R}$. Let $\tilde{A} = \left[\begin{smallmatrix} A & \boldsymbol{y} \\ \boldsymbol{y}^{\mathrm{H}} & a \end{smallmatrix}\right]$ and $\tilde{\lambda}_1 \le \tilde{\lambda}_2 \le \cdots \le \tilde{\lambda}_{n+1}$ be its eigenvalues. Then, $\tilde{\lambda}_i \le \lambda_i \le \tilde{\lambda}_{i+1}$ for $i = 1,\dots,n$.*

2.1.2 Congruence and inertia

This subsection explains a congruence, which is an equivalence relation on the set of Hermitian matrices of the same size, and the inertia of a Hermitian matrix.

Definition 2.7 (congruence)**.** Let $A, B \in \mathbb{C}^{n\times n}$ be Hermitian and $C \in \mathbb{C}^{n\times n}$ be invertible. A congruence is the linear transformation $A \mapsto C^{\mathrm{H}}AC$. Two matrices A, B are congruent if there exists C such that $B = C^{\mathrm{H}}AC$.

Definition 2.8 (inertia)**.** The inertia of Hermitian A is the triple

$$(n(A), z(A), p(A)),$$

where $n(A)$, $z(A)$, and $p(A)$ are the number of negative, zero, and positive eigenvalues of A, respectively.

For Hermitian A, its rank and *signature* are $n(A)+p(A)$ and $p(A)-n(A)$, respectively. The inertia uniquely determines the rank and signature, and vice versa.

An important result about the inertia is the following theorem by Sylvester, which is generally referred to as Sylvester's law of inertia.

Theorem 2.9 (Sylvester [27, Theorem 4.5.8])**.** *The inertia of a Hermitian matrix is invariant under a congruence. Namely, for Hermitian A and invertible C,*

$$(n(CAC^{\mathrm{H}}), z(CAC^{\mathrm{H}}), p(CAC^{\mathrm{H}})) = (n(A), z(A), p(A)).$$

In other words, Hermitian matrices are congruent if and only if they have the same inertia.

A standard way to compute the inertia of a Hermitian matrix is to perform Gauss elimination to generate an LDL^{H} factorization. In principle, any Hermitian A can be factored into the form of $PAP^{\mathrm{T}} = LDL^{\mathrm{H}}$ by using an appropriate permutation P, where L is a lower triangular matrix with its diagonal elements equal to one (and thus is invertible) and D is a block diagonal matrix with block size one or two. An LDL^{H} factorization of possibly indefinite matrices, unlike the Cholesky factorization of positive definite matrices, is not unique and depends mainly on the choice of permutation P. Once an LDL^{H} factorization of A is computed, because of Sylvester's law of inertia, it follows that $(n(A), z(A), p(A)) = (n(D), z(D), p(D))$, and the inertia of A is obtained by counting negative, zero, and positive eigenvalues of block diagonal D.

2.2 Hermitian definite generalized eigenvalue problems

This section considers HDGEP of Hermitian $A, B \in \mathbb{C}^{n\times n}$,

$$Ax = \lambda Bx, \tag{2.2}$$

or equivalently the eigenvalue problem of definite pencil (A,B). Pencil (A,B) *definite* if there exist $\alpha, \beta \in \mathbb{R}$ such that $\alpha A + \beta B > O$ (positive definite). When considering HDGEP, it can be assumed without loss of generality that $B > O$. This is because the eigenvalue problem of (A,B) can be reduced to the eigenvalue problem of $(A, \alpha A + \beta B)$, or $A\boldsymbol{x} = \tilde{\lambda}(\alpha A + \beta B)\boldsymbol{x}$ with $\tilde{\lambda} = \frac{\lambda}{\alpha\lambda+\beta}$, provided that β is nonzero. If $A > O$ and thus β can be zero, the problem of (A,B) is reduced to that of (B,A). Therefore, unless otherwise stated, it is assumed that $B > O$ in this thesis.

The eigenvalues of definite pencil (A,B) are the roots of the characteristic equation $\det(A-\lambda B) = 0$. Nontrivial solution $\boldsymbol{x}$ of linear system $(A-\lambda B)\boldsymbol{x} = \boldsymbol{0}$ is an eigenvector associated with λ, and the null space of $A - \lambda B$ is the eigenspace associated with λ. All eigenvalue of (A,B) are real and are indexed in the increasing order, or $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$, in the thesis. Eigenvectors are indexed accordingly such that $A\boldsymbol{x}_i = \lambda_i B\boldsymbol{x}_i$. Eigenvectors associated with distinct eigenvalues, or $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with $\lambda_i \neq \lambda_j$, are *B-orthogonal*, or $(\boldsymbol{x}_i, \boldsymbol{x}_j)_B = 0$. Eigenvectors associated with the same eigenvalue, or $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with $\lambda_i = \lambda_j$, $i \neq j$, can be selected to be B-orthogonal, leading to *simultaneous diagonalization* $X^{\mathrm{H}}AX = \Lambda$, $X^{\mathrm{H}}BX = I$. Here, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, and $X = [\boldsymbol{x}_1 \ldots \boldsymbol{x}_n]$.

HDGEP can be reduced to HEP. For example, let C be either the Cholesky factor L or the principal square root S of $B = LL^{\mathrm{H}} = (S)^2$. Then, HDGEP (2.2) can be reduced to HEP $\tilde{A}\tilde{\boldsymbol{x}} = \lambda\tilde{\boldsymbol{x}}$ in which $\tilde{A} = C^{-1}AC^{-\mathrm{H}}$ and $\tilde{\boldsymbol{x}} = C^{\mathrm{H}}\boldsymbol{x}$. Note that eigenvalues λ do not change by the reduction. Because of this relation between HDGEP and HEP, theoretical results on HEP and numerical algorithms for HEP can often be extended and applied to HDGEP in a straightforward manner. For example, Theorem 2.9 for HEP carries over to HDGEP as follows.

Definition 2.10. The inertia of definite pencil (A,B), $B > O$ is the triple

$$(n(A,B), z(A,B), p(A,B)),$$

where $n(A,B)$, $z(A,B)$, and $p(A,B)$ are the number of negative, zero, and positive eigenvalues of (A,B), respectively.

Corollary 2.11. *Let (A,B) be a definite pencil with $B > O$. Let X, Y be invertible. Then,*

$$(n(XAX^{\mathrm{H}}, YBY^{\mathrm{H}}), z(XAX^{\mathrm{H}}, YBY^{\mathrm{H}}), p(XAX^{\mathrm{H}}, YBY^{\mathrm{H}})) = (n(A,B), z(A,B), p(A,B)).$$

In particular, $(n(A), z(A), p(A)) = (n(A,B), z(A,B), p(A,B))$.

HDGEP of (A,B) is equivalent to a *real symmetric* definite GEP. Let $A = A_1 + iA_2$ with real symmetric A_1 and real skew symmetric A_2. In the same way, let $B = B_1 + iB_2$ with $B_1 = B_1^{\mathrm{T}}$ and $B_2 = -B_2^{\mathrm{T}}$. Assume that $(\lambda, \boldsymbol{x})$ is an eigenpair of (A,B), and let $\boldsymbol{x} = \boldsymbol{x}_1 + i\boldsymbol{x}_2$ with real vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Then, $\left(\lambda, \left[\begin{smallmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{smallmatrix}\right]\right)$ and $\left(\lambda, \left[\begin{smallmatrix} -\boldsymbol{x}_2 \\ \boldsymbol{x}_1 \end{smallmatrix}\right]\right)$ are eigenpairs of *symmetric* definite pencil $\left(\left[\begin{smallmatrix} A_1 & A_2^{\mathrm{T}} \\ A_2 & A_1 \end{smallmatrix}\right], \left[\begin{smallmatrix} B_1 & B_2^{\mathrm{T}} \\ B_2 & B_1 \end{smallmatrix}\right]\right)$. Because of this equivalence, HDGEP can be solved by algorithms using only real arithmetic.

2.2.1 Bisection algorithm

This subsection explains the bisection algorithm for computing the k-th eigenvalue of HDGEP. Given interval $[\alpha, \beta) \subset \mathbb{R}$ containing λ_k, the bisection algorithm computes the number of eigenvalues smaller than $\mu = \frac{\alpha+\beta}{2}$, denoted $n_\mu(A,B)$. If $n_\mu(A,B) \geq k$, the current interval $[\alpha, \beta)$ is narrowed down to the left half-interval $[\alpha, \mu)$. Otherwise (if $n_\mu(A,B) < k$), interval $[\alpha, \beta)$ is narrowed down to the right half-interval $[\mu, \beta)$. Because of Corollary 2.11, $n_\mu(A,B) = n(A - \mu B, B) = n(A - \mu B)$ and thus number $n_\mu(A,B)$ can be computed by LDL^{H} factorization of shifted matrix $A - \mu B$.

Algorithm 2.1 shows the bisection algorithm. When an interval $[\mu_{\text{lower}}, \mu_{\text{upper}})$ containing λ_k is given, the midpoint μ of the interval is calculated in line 2, and then $n_\mu(A,B)$ is computed by utilizing LDL^{H} factorization of $A - \mu B$ in lines 3–4. Here, $A - \mu B$ is indefinite; thus, permutation P is necessary for the existence and numerical stability of factorization and is generated based on pivoting strategies by, e.g, Bunch–Kaufman [9] and Duff–Reid [16]. L and D are a unit lower triangular matrix and a block diagonal matrix of block size one or two, respectively. Depending on $n(D) = n_\mu(A,B)$, either the left or right half-interval is selected as the next interval in line 5. Approximately $\lceil \log_2[(\mu_{\text{upper}} - \mu_{\text{lower}})/\tau] \rceil$ iterations are required until the interval becomes narrower than a given tolerance τ. Thus, a narrower initial interval will result in fewer required iterations to locate λ_k.

Algorithm 2.1: Bisection for computing the k-th eigenvalue of HDGEP

```
Input  : matrices $A, B$ of HDGEP (2.2), target index $k \in \mathbb{N}$,
         interval $[\mu_{\text{lower}}, \mu_{\text{upper}}) \subset \mathbb{R}$ containing the $k$-th eigenvalue, tolerance $\tau \in \mathbb{R}$.
Output: approximate eigenvalue $\hat{\lambda}_k := (\mu_{\text{lower}} + \mu_{\text{upper}})/2$, where $|\hat{\lambda}_k - \lambda_k| < \tau/2$.
1 repeat until $\mu_{\text{upper}} - \mu_{\text{lower}} < \tau$ do
2     $\mu := (\mu_{\text{lower}} + \mu_{\text{upper}})/2$,
3     $LDL^{\mathrm{H}} \leftarrow P(A - \mu B)P^{\mathrm{T}}$,   ▷ $P$ : permutation for numerical stability
4     $\nu := n(D)$,                ▷ $n(D)$ : number of negative eigenvalues of block diagonal $D$
5     if $k \leq \nu$ then $\mu_{\text{upper}} := \mu$ else $\mu_{\text{lower}} := \mu$.
6 end
```

2.2.2 Lanczos method

The Lanczos method [43] is a subspace projection method in which approximate solutions are constructed within a Krylov subspace and are determined to be optimal in the sense of the Galerkin condition. The subspace and its orthonormal basis are generated by the Lanczos process, which can be expressed in the matrix form:

$$AV_j = BV_jT_j + B\boldsymbol{v}_{j+1}\beta_j\boldsymbol{e}_j^{\mathrm{T}}. \tag{2.3}$$

Here, $V_j = [\boldsymbol{v}_1 \cdots \boldsymbol{v}_j]$ is an $n \times j$ matrix whose columns span a Krylov subspace and are B-orthonormal. Matrix $T_j = V_j^{\mathrm{H}} A V_j$ is real symmetric tridiagonal and has non-zero off-diagonal elements (irreducible). Vector $\boldsymbol{v}_{j+1}$ is B-orthogonal to the columns of V_j and is normalized with respect to the B-norm by scale factor β_j. Vector $\boldsymbol{e}_j$ is the last column of the identity matrix of size j. In this thesis, (2.3) is referred to as j-step Lanczos decomposition whose algorithm is shown in Algorithm 2.2.

Algorithm 2.2: j-step Lanczos decomposition in B-inner product [5, Chapter 5.5]

```
Input  : matrices $A, B$ of HDGEP (2.2), parameter $j \in N$.
Output: matrix $V_j \in \mathbb{C}^{n\times j}$, vector $\boldsymbol{v}_{j+1}$, real symmetric tridiagonal $T_j \in \mathbb{R}^{j\times j}$, scalar $\beta_j \in \mathbb{R}$.
1 $V_0 := [\,]$, $\boldsymbol{w}_0 := \mathbf{0}$, set random starting vector $\boldsymbol{v}_1$.
2 $\boldsymbol{w}_1 := B\boldsymbol{v}_1$, $\beta_0 := (\boldsymbol{w}_1^{\mathrm{H}}\boldsymbol{v}_1)^{\frac{1}{2}}$, $\boldsymbol{v}_1 := \boldsymbol{v}_1/\beta_0$, $\boldsymbol{w}_1 := \boldsymbol{w}_1/\beta_0$,
3 for $i = 1, 2, \ldots, j$ do
4     $V_j := [V_{j-1}\boldsymbol{v}_j]$, $\boldsymbol{w}_{i+1} := A\boldsymbol{v}_i$, $\boldsymbol{w}_{i+1} := \boldsymbol{w}_{i+1} - \beta_{i-1}\boldsymbol{w}_{i-1}$,
5     $\alpha_i := \boldsymbol{w}_{i+1}^{\mathrm{H}}\boldsymbol{v}_i$, $\boldsymbol{w}_{i+1} := \boldsymbol{w}_{i+1} - \alpha_i\boldsymbol{w}_i$,
6     solve linear system $B\boldsymbol{v}_{i+1} = \boldsymbol{w}_{i+1}$, $\beta_i := (\boldsymbol{w}_{i+1}^{\mathrm{H}}\boldsymbol{v}_{i+1})^{1/2}$,
7     $\boldsymbol{v}_{i+1} := \boldsymbol{v}_{i+1}/\beta_i$, $\boldsymbol{w}_{i+1} := \boldsymbol{w}_{i+1}/\beta_i$,
8 end
9 $T_j := \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{j-1} \\ & & \beta_{j-1} & \alpha_j \end{bmatrix}$.
```

From the Galerkin condition, the standard eigenvalue problem of T_j is derived:

$$T_j \boldsymbol{y}_i^{(j)} = \theta_i^{(j)} \boldsymbol{y}_i^{(j)}, \quad \boldsymbol{y}_i^{(j)} \neq \mathbf{0}. \tag{2.4}$$

Since T_j is irreducible, eigenvalues $\theta_i^{(j)}$ are distinct from each other [52, Chapter 1.3] and can be indexed in increasing order, i.e., $\theta_1^{(j)} < \theta_2^{(j)} < \cdots < \theta_j^{(j)}$. The Lanczos method can be considered a Rayleigh–Ritz procedure, and eigenvalues $\theta_i^{(j)}$ are referred to as *Ritz values*.

Shift-and-invert Lanczos method

A small number of eigenvalues closest to a target point $\mu \in \mathbb{R}$ and their associated eigenvectors can be computed by the shift-and-invert (SI) Lanczos method [18]. The SI Lanczos method applied to the original HDGEP (2.2) can be considered as applying the original Lanczos method to the shift-and-inverted (SI) problem:

$$(A - \mu B)^{-1}\tilde{\boldsymbol{x}} = \tilde{\lambda} B^{-1}\tilde{\boldsymbol{x}}, \quad \tilde{\boldsymbol{x}} \neq \mathbf{0}. \tag{2.5}$$

Here, it is assumed that shift μ does not coincide with an eigenvalue of (2.2), i.e., $\mu \neq \lambda_i$. The eigenvalues of the original and SI problems have the relationship $\tilde{\lambda} = (\lambda - \mu)^{-1}$, while eigenvectors satisfy both $\tilde{\boldsymbol{x}} = B\boldsymbol{x}$ and $\tilde{\boldsymbol{x}} = (A - \mu B)\boldsymbol{x}$ because $B\boldsymbol{x}$ and $(A - \mu B)\boldsymbol{x}$ are collinear. Based on these relationships, approximate eigenpairs for the SI problem (2.5) are transformed to those for the original HDGEP (2.2).

The matrix form of j-step SI Lanczos decomposition is given as follows:

$$(A - \mu B)^{-1}\tilde{V}_j = B^{-1}\tilde{V}_j\tilde{T}_j + B^{-1}\tilde{\boldsymbol{v}}_{j+1}\tilde{\beta}_j\boldsymbol{e}_j^{\mathrm{T}}. \tag{2.6}$$

Here, $\tilde{V}_j$ is an $n \times j$ matrix whose columns span a Krylov subspace and are B^{-1}-orthonormal, and $\tilde{T}_j$ is real symmetric tridiagonal and irreducible. $\tilde{\boldsymbol{v}}_{j+1}$ is B^{-1}-orthogonal to the columns of $\tilde{V}_j$ and is normalized with respect to the B^{-1}-norm by scale factor $\tilde{\beta}_j$.

From the Galerkin condition, the standard eigenvalue problem of $\tilde{T}_j$ is derived:

$$\tilde{T}_j \tilde{\boldsymbol{y}}_i^{(j)} = \tilde{\theta}_i^{(j)} \tilde{\boldsymbol{y}}_i^{(j)}, \quad \tilde{\boldsymbol{y}}_i^{(j)} \neq \boldsymbol{0}. \tag{2.7}$$

Here, eigenvalues $\tilde{\theta}_i^{(j)}$ are indexed in increasing order, and eigenvectors $\tilde{\boldsymbol{y}}_i^{(j)}$ are assumed to be normalized with respect to the 2-norm in the thesis. Using the i-th eigenpair $(\tilde{\theta}_i^{(j)}, \tilde{\boldsymbol{y}}_i^{(j)})$ of $\tilde{T}_j$, the approximate eigenvalues for (2.2) are expressed as follows:

$$\lambda_i^{(j)} = \mu + 1/\tilde{\theta}_i^{(j)}. \tag{2.8}$$

Approximate eigenvectors can be expressed in two different ways.

1. Using the first relationship $\tilde{\boldsymbol{x}} = B\boldsymbol{x}$, approximate eigenvectors are given as:

$$\boldsymbol{x}_{i,1}^{(j)} = B^{-1} \tilde{V}_j \tilde{\boldsymbol{y}}_i^{(j)}. \tag{2.9}$$

2. From the second relationship $\tilde{\boldsymbol{x}} = (A - \mu B)\boldsymbol{x}$,

$$\boldsymbol{x}_{i,2}^{(j)} = (A - \mu B)^{-1} \tilde{V}_j \tilde{\boldsymbol{y}}_i^{(j)}. \tag{2.10}$$

In this thesis, $\boldsymbol{x}_{i,2}^{(j)}$ in (2.10) is utilized as an approximate eigenvector, although it is common to use $\boldsymbol{x}_{i,1}^{(j)}$ in (2.9). This is because, as will be shown in Proposition 3.1, $\boldsymbol{x}_{i,2}^{(j)}$ allows economical evaluation of an eigenvalue error bound that can be utilized to validate the index of approximate eigenpairs. Other perspectives on approximate eigenvectors and their further treatment can be found in the literature [18] and [5, Chapter 7.6.8], in which $\boldsymbol{x}_{i,2}^{(j)}$ is considered a modification of $\boldsymbol{x}_{i,1}^{(j)}$.

The rest of this subsection provides theoretical results that support the utilization of $\boldsymbol{x}_{i,2}^{(j)}$ as an approximate eigenvector. Proposition 2.12 shows that $\boldsymbol{x}_{i,2}^{(j)}$ converges to the same eigenvector of (2.2) at the same iteration of the SI Lanczos method as $\boldsymbol{x}_{i,1}^{(j)}$. Proposition 2.13 shows that $\boldsymbol{x}_{i,2}^{(j)}$ with $1 \le i \le j$ become B-orthogonal to each other as the SI Lanczos method proceeds.

Proposition 2.12. *Approximate eigenvectors* $\boldsymbol{x}_{i,1}^{(j)}$ *in* (2.9) *and* $\boldsymbol{x}_{i,2}^{(j)}$ *in* (2.10) *are collinear if and only if* $\tilde{\boldsymbol{v}}_{j+1} = \boldsymbol{0}$ *in* (2.6).

Proof. We first prove the sufficiency. By post-multiplying (2.6) by $\tilde{\boldsymbol{y}}_i^{(j)}$ and from the sufficient condition $\tilde{\boldsymbol{v}}_{j+1} = \boldsymbol{0}$, we have:

$$\boldsymbol{x}_{i,2}^{(j)} = (A - \mu B)^{-1} \tilde{V}_j \tilde{\boldsymbol{y}}_i^{(j)} = B^{-1} \tilde{V}_j \tilde{T}_j \tilde{\boldsymbol{y}}_i^{(j)} = \tilde{\theta}_i^{(j)} \boldsymbol{x}_{i,1}^{(j)}.$$

The third equality follows from (2.7). This proves the sufficiency. Now, we prove the necessity. By post-multiplying (2.6) by $\tilde{\boldsymbol{y}}_i^{(j)}$, we have:

$$\boldsymbol{x}_{i,2}^{(j)} = \tilde{\theta}_i^{(j)} \boldsymbol{x}_{i,1}^{(j)} + B^{-1} \tilde{\boldsymbol{v}}_{j+1} (\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)}).$$

Here, $\boldsymbol{x}_{i,1}^{(j)}$ and $\boldsymbol{x}_{i,2}^{(j)}$ are collinear from the necessary condition; thus, the last term $B^{-1} \tilde{\boldsymbol{v}}_{j+1} (\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)})$ must be collinear with $\boldsymbol{x}_{i,1}^{(j)}$ and $\boldsymbol{x}_{i,2}^{(j)}$. In addition, the last term is B-orthogonal to $\boldsymbol{x}_{i,1}^{(j)}$ because $\tilde{\boldsymbol{v}}_{j+1}$ is B^{-1}-orthogonal to the columns of $\tilde{V}_j$:

$$(\boldsymbol{x}_{i,1}^{(j)})^{\mathrm{H}} B \left[B^{-1} \tilde{\boldsymbol{v}}_{j+1} (\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)}) \right] = (\tilde{\boldsymbol{y}}_i^{(j)})^{\mathrm{H}} \tilde{V}_j^{\mathrm{H}} B^{-1} \tilde{\boldsymbol{v}}_{j+1} (\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)}) = 0.$$

Due to this B-orthogonality and the positive-definiteness of B, the last term can never be collinear with $\boldsymbol{x}_{i,1}^{(j)}$ unless it is the zero vector. Therefore, $\tilde{\boldsymbol{v}}_{j+1} = \boldsymbol{0}$ (thus, $\tilde{\beta}_j = 0$) or $\boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)} = 0$. However, $\boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)}$ is non-zero because it is the last element of an eigenvector of an irreducible tridiagonal matrix [55, Theorem 7.9.3]. This proves the necessity. □

Note that $\boldsymbol{x}_{i,2}^{(j)}$ with $1 \le i \le j$ do not have exact B-orthogonality. Their B-orthogonality can be measured by (2.11) in Proposition 2.13, which is the cosine similarity of two approximate eigenvectors in the B-inner product.

Proposition 2.13. *The following holds for* $1 \le l < m \le j$*:*

$$\frac{|(\boldsymbol{x}_{l,2}^{(j)})^{\mathrm{H}} B \boldsymbol{x}_{m,2}^{(j)}|}{\|\boldsymbol{x}_{l,2}^{(j)}\|_B \cdot \|\boldsymbol{x}_{m,2}^{(j)}\|_B} = \frac{|\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_l^{(j)} / \tilde{\theta}_l^{(j)}|}{\sqrt{1 + |\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_l^{(j)} / \tilde{\theta}_l^{(j)}|^2}} \cdot \frac{|\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_m^{(j)} / \tilde{\theta}_m^{(j)}|}{\sqrt{1 + |\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_m^{(j)} / \tilde{\theta}_m^{(j)}|^2}}. \tag{2.11}$$

Proof. For $1 \le l \le m \le j$,

$$\begin{aligned}
(\boldsymbol{x}_{l,2}^{(j)})^{\mathrm{H}} B \boldsymbol{x}_{m,2}^{(j)} &= \left[(A-\mu B)^{-1} \tilde{V}_j \tilde{\boldsymbol{y}}_l^{(j)}\right]^{\mathrm{H}} B \left[(A-\mu B)^{-1} \tilde{V}_j \tilde{\boldsymbol{y}}_m^{(j)}\right] \\
&= \left(\tilde{V}_j \tilde{T}_j \tilde{\boldsymbol{y}}_l^{(j)} + \tilde{\boldsymbol{v}}_{j+1} \tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_l^{(j)}\right)^{\mathrm{H}} B^{-1} B B^{-1} \left(\tilde{V}_j \tilde{T}_j \tilde{\boldsymbol{y}}_m^{(j)} + \tilde{\boldsymbol{v}}_{j+1} \tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_m^{(j)}\right) \\
&= \left(\tilde{V}_j \tilde{T}_j \tilde{\boldsymbol{y}}_l^{(j)}\right)^{\mathrm{H}} B^{-1} \left(\tilde{V}_j \tilde{T}_j \tilde{\boldsymbol{y}}_m^{(j)}\right) + \left(\tilde{\boldsymbol{v}}_{j+1} \tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_l^{(j)}\right)^{\mathrm{H}} B^{-1} \left(\tilde{\boldsymbol{v}}_{j+1} \tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_m^{(j)}\right) \\
&= \tilde{\theta}_l^{(j)} \tilde{\theta}_m^{(j)} \left[\delta_{lm} + (\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_l^{(j)} / \tilde{\theta}_l^{(j)})(\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_m^{(j)} / \tilde{\theta}_m^{(j)})\right]. \qquad (2.12)
\end{aligned}$$

Here, δ_{lm} denotes the Kronecker delta. The second equality follows from (2.6), and the third and fourth are due to the B^{-1}-orthonormality of the columns of $\tilde{V}_j$ and $\tilde{\boldsymbol{v}}_{j+1}$. (2.11) is an immediate result of (2.12) with $l \ne m$. □

Scalars $|\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)} / \tilde{\theta}_i^{(j)}|$ in (2.11) are simply the B^{-1}-norm of the residual vectors:

$$\boldsymbol{r}_{i,2}^{(j)} \equiv (A - \lambda_i^{(j)} B) \boldsymbol{x}_{i,2}^{(j)} = \left[A - (\mu + 1/\tilde{\theta}_i^{(j)}) B\right] (A-\mu B)^{-1} \tilde{V}_j \tilde{\boldsymbol{y}}_i^{(j)} = -\tilde{\boldsymbol{v}}_{j+1} (\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)} / \tilde{\theta}_i^{(j)}). \tag{2.13}$$

Since the residual norm $|\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)} / \tilde{\theta}_i^{(j)}| \ll 1$ as the SI Lanczos method proceeds, (2.11) converges to zero and approximate eigenvectors $\boldsymbol{x}_{i,2}^{(j)}$ become B-orthogonal to each other. Therefore, approximate eigenvectors $\boldsymbol{x}_{i,2}^{(j)}$ have B-orthogonality in a practical sense.

Chapter 3

An algorithm for the k-th eigenvalue problem

This chapter presents a three-stage algorithm with the following features for solving k-EP (1.3), which is based on theoretical background and algorithms in the previous chapter.

In the first stage of the algorithm, we propose an efficient way of finding an interval containing the k-th eigenvalue ($1 \ll k \ll n$) with a non-standard application of the Lanczos method. A standard approach is to utilize some Gershgorin-type theorems so that the interval includes the entire spectrum and thus contains λ_k. The proposed approach iteratively generates a sequence of disjoint intervals until an interval validated as containing λ_k is obtained. For our problem with the target index $1 \ll k \ll n$, the numerical results show that information conventionally considered useless in the Lanczos method, i.e., Ritz values of the first few steps of the method, is of paramount importance to generate a narrow initial interval.

In the second stage, bisection for large sparse problems is realized using a sparse direct linear solver to narrow down the interval of the k-th eigenvalue. Each bisection iteration requires computation of $n_\mu(A,B)$, or the number of eigenvalues smaller than μ, which is based on LDL^{H} factorization of a shifted matrix $A-\mu B$ and thus requires $O(n^2)$ memory in general. In the proposed approach, number $n_\mu(A,B)$ is computed by utilizing a sparse direct linear solver to save memory to realize bisection for large sparse problems. Once a fill-reducing ordering and symbolic factorization are obtained for some $A-\mu B$, they can be recycled for other shifted matrices because they depend on only the sparsity structure of a matrix.

In the third stage, we switch to a modified shift-and-invert Lanczos method to reduce bisection iterations and compute the k-th eigenpair with validation. As will be shown in Proposition 3.1, the index is validated by utilizing an eigenvalue error bound that can be evaluated from the residual vector at negligible cost.

The remainder of this chapter is organized as follows. In Section 3.1, after explaining our approach to set an initial interval and compute the k-th eigenvector, we present the three-stage algorithm for k-EP. Section 3.2 reports numerical results of several real research problems and a comparison of the three-stage algorithm and dense eigensolvers in order to examine the accuracy and efficiency of the proposed algorithm. Concluding remarks are given in Section 3.3.

3.1 A three-stage algorithm

3.1.1 An efficient initial interval for bisection

To utilize bisection, it is necessary to set an initial interval that contains the k-th eigenvalue. A common approach is to set an interval that includes the entire spectrum, which necessarily contains λ_k. For SEP, one of the most economical ways to set such an interval is to use the Gershgorin theorem (Theorem 2.4). Several Gershgorin-type theorems have been proposed for GEP [42, 54, 60]. These theorems provide an inclusion set of the spectrum that is guaranteed to be bounded only when at least one of A and B is diagonally dominant for each row [54]. Unfortunately, such diagonal dominance of the matrices is not always assumed to be the case. The remainder of this subsection presents a systematic way to set an initial interval based on an application of the Lanczos method (Section 2.2.2).

Ritz values by the Lanczos method become more accurate by expanding the subspace, exhibiting the interlacing property (Theorem 2.6). By denoting the i-th Ritz value at the j-th iteration of the method as $\theta_i^{(j)}$ as in (2.4), the interlacing property can be formally expressed as follows: $\theta_i^{(j+1)} < \theta_i^{(j)} < \theta_{i+1}^{(j+1)}$ for $i \leq j < n$. The monotonic convergence follows from this property, which states that the i-th smallest (resp. largest) Ritz value decreases (resp. increases) monotonically and converges to the i-th smallest (resp. largest) eigenvalue. We utilize this monotonicity of Ritz values to set an initial interval.

Algorithm 3.1 shows how we set an interval containing λ_k by utilizing Ritz values, and this process is illustrated in Figure 3.1. Here, we utilize either the smallest $\theta_1^{(j)}$ or largest Ritz values $\theta_j^{(j)}$ with $j \geq 1$. We begin by computing $\theta_1^{(1)}$, which is equal to the generalized Rayleigh quotient $\boldsymbol{v}_1^{\mathrm{H}} A \boldsymbol{v}_1 / \boldsymbol{v}_1^{\mathrm{H}} B \boldsymbol{v}_1$ of a random starting vector $\boldsymbol{v}_1$. The quotient is expected to be the average of the eigenvalues of (1.3). It is advantageous to begin from approximately the middle of the eigenvalue distribution when the target index satisfies $1 \ll k \ll n$. We then compute $\nu^{(1)} = n_{\theta_1^{(1)}}(A,B)$. If $k \leq \nu^{(1)}$, it follows that $\lambda_k < \theta_1^{(1)} < \theta_j^{(j)}$ for $j > 1$. Therefore, in the subsequent iterations, the smallest Ritz values $\theta_1^{(j)}$ are utilized as the points for setting an interval containing λ_k. On the other hand, if $k > \nu^{(1)}$, the largest Ritz values are selected as the endpoints of an interval. Accordingly, a sequence of disjoint intervals, i.e., either $[\theta_1^{(j)}, \theta_1^{(j-1)})$ or $[\theta_{j-1}^{(j-1)}, \theta_j^{(j)})$ with $j > 1$, is generated until an interval validated as containing λ_k is obtained.

```
Algorithm 3.1: Setting an interval containing the $k$-th eigenvalue
  Input  : matrices $A, B$ of HDGEP (2.2), target index $k \in \mathbb{N}$.
  Output: interval $[\mu_{\mathrm{lower}}, \mu_{\mathrm{upper}}) \subset \mathbb{R}$ containing the $k$-th eigenvalue,
          $\nu_{\mathrm{lower}} = n_{\mu_{\mathrm{lower}}}(A,B),\ \nu_{\mathrm{upper}} = n_{\mu_{\mathrm{upper}}}(A,B)$.
1  set a random starting vector $\boldsymbol{v}_1,\ \boldsymbol{v}_1 := \boldsymbol{v}_1/\|\boldsymbol{v}_1\|_B,\ \nu^{(0)} := 0$,
2  for $j = 1, 2, \ldots$ do
3      compute $j$-step Lanczos decomposition (2.3),
4      solve standard eigenvalue problem (2.4),
5      if $k \leq \nu^{(j-1)}$ then $\mu^{(j)} := \theta_1^{(j)}$ else $\mu^{(j)} := \theta_j^{(j)}$,
6      $LDL^{\mathrm{H}} \leftarrow P(A - \mu^{(j)} B)P^{\mathrm{T}}$,   ▷ $P$ : permutation for numerical stability
7      $\nu^{(j)} := n(D)$,                 ▷ $n(D)$ : number of negative eigenvalues of block diagonal $D$
8      if $j \neq 1$ and ($\nu^{(j)} < k \leq \nu^{(j-1)}$ or $\nu^{(j-1)} < k \leq \nu^{(j)}$) then break,
9  end for
10 $\mu_{\mathrm{lower}} := \min\{\mu^{(j-1)}, \mu^{(j)}\},\ \mu_{\mathrm{upper}} := \max\{\mu^{(j-1)}, \mu^{(j)}\}$,
11 $\nu_{\mathrm{lower}} := \min\{\nu^{(j-1)}, \nu^{(j)}\},\ \nu_{\mathrm{upper}} := \max\{\nu^{(j-1)}, \nu^{(j)}\}$.
```

An advantage of utilizing Ritz values is that an initial interval is necessarily included in and can be much narrower than $[\lambda_1, \lambda_n]$, which leads to a reduction of the number of bisection iterations. However, we must consider the cost of setting the interval, i.e., j iterations of the Lanczos method and j LDL^{H} factorizations. The interlacing property provides a theoretical upper bound of the iteration count required to set an interval. Here, since $\theta_1^{(n-k+1)} < \lambda_k < \theta_k^{(k)}$, at most $n-k+1$ iterations are required for the $k \leq \nu^{(1)}$ case, and k iterations are required for the $k > \nu^{(1)}$ case. In practice, j can be very small because eigenvalues at the ends of $[\lambda_1, \lambda_n]$ are rapidly approximated by the Lanczos

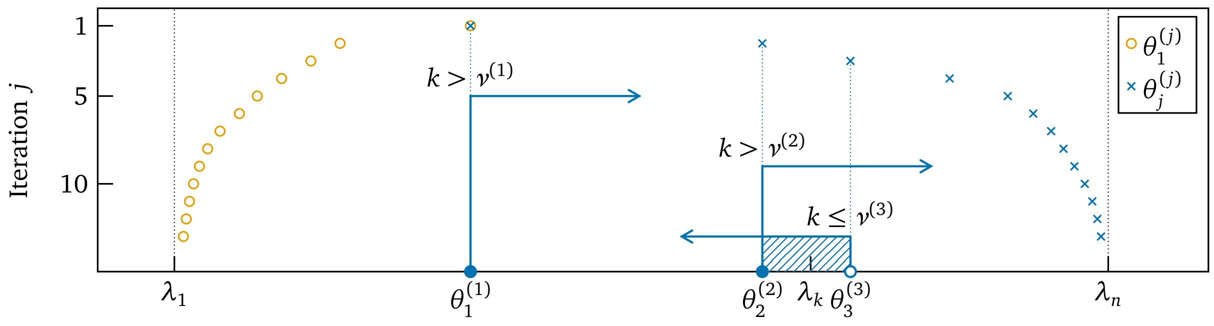


Figure 3.1: Illustration of Algorithm 3.1. $[\theta_1^{(1)}, \theta_2^{(2)})$ does not contain λ_k but $[\theta_2^{(2)}, \theta_3^{(3)})$ does (hatched).

method and the target index k is approximately 10–50% of the matrix size n, thereby satisfying $1 \ll k \ll n$. In the numerical results presented in Section 3.2.3, the iteration count j is two for all experiments.

3.1.2 Bisection for large sparse problems

We utilize a sparse direct linear solver [13] for computing $n_\mu(A,B)$ of large sparse A,B in order to realize bisection for narrowing down an initial interval. Algorithm 3.2 shows a modification of Algorithm 2.1 (bisection). The main difference can be found in line 3, where a fill-reducing ordering Q based, e.g., on minimum degree or nested dissection [1, 38], is utilized to handle large sparse matrices, in addition to permutation P for numerical stability. In contrast to P, ordering Q is independent from shift μ because it depends on only the sparsity structure of $A - \mu B$. Thus, once ordering and symbolic factorization are obtained for some shifted matrix, they can be recycled for other shifted matrices with varying shifts in the subsequent iterations. Computation of $n_\mu(A,B)$ in Algorithm 3.1 can be performed in the same manner by utilizing a sparse direct linear solver.

Algorithm 3.2: Bisection for narrowing down an interval of the k-th eigenvalue

```
Input  : matrices $A, B$ of HDGEP (2.2), target index $k$,
         interval $[\mu_{\text{lower}}, \mu_{\text{upper}}) \subset \mathbb{R}$ containing the $k$-th eigenvalue,
         $\nu_{\text{lower}} = n_{\mu_{\text{lower}}}(A,B)$, $\nu_{\text{upper}} = n_{\mu_{\text{upper}}}(A,B)$, stopping criterion $m_{\max} \in \mathbb{N}$.
Output: interval $[\mu_{\text{lower}}, \mu_{\text{upper}})$, $\nu_{\text{lower}}$, $\nu_{\text{upper}}$.
1 repeat until $\nu_{\text{upper}} - \nu_{\text{lower}} \le m_{\max}$   ▷ $\nu_{\text{upper}} - \nu_{\text{lower}}$: number of eigenvalues in the interval
2     $\mu := (\mu_{\text{lower}} + \mu_{\text{upper}})/2$,
3     $LDL^{\text{H}} \leftarrow PQ(A - \mu B)Q^{\text{T}}P^{\text{T}}$,   ▷ $Q$ : fill-reducing ordering for sparse matrices
4     $\nu := n(D)$,
5     if $k \le \nu$ then $\mu_{\text{upper}} := \mu$, $\nu_{\text{upper}} := \nu$ else $\mu_{\text{lower}} := \mu$, $\nu_{\text{lower}} := \nu$.
6 end
```

3.1.3 Computation of the k-th eigenpair with validation

Once an initial interval is set for bisection, it is narrowed down until it contains the k-th eigenvalue and some nearby eigenvalues. We then compute the k-th eigenpair along with the other eigenpairs of the interval while validating their index. The eigenpairs of the interval can be computed by a variety of methods (Section 1.2). We utilize a modified SI Lanczos method (Section 2.2.2), where approximate eigenvectors $\boldsymbol{x}_{i,2}^{(j)}$ are computed based on (2.10) and have B-orthogonality, as shown in Proposition 2.13.

In Proposition 3.1, we explain that an eigenvalue error bound for validating the index of the eigenpairs can be evaluated from the residual vector of the SI Lanczos method at negligible cost.

Proposition 3.1. *With the same notation as* (2.6)–(2.8) *in Section 2.2.2, there is an eigenvalue* $\lambda_{l_i^{(j)}}$ *of problem* (2.2) *such that:*

$$\lambda_{l_i^{(j)}} \in \Gamma_i^{(j)} \equiv [\lambda_i^{(j)} - \eta_i^{(j)}, \lambda_i^{(j)} + \eta_i^{(j)}], \quad \eta_i^{(j)} \equiv \frac{1}{|\tilde{\theta}_i^{(j)}|} \cdot \frac{|\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)} / \tilde{\theta}_i^{(j)}|}{\sqrt{1 + |\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)} / \tilde{\theta}_i^{(j)}|^2}}. \tag{3.1}$$

Proof. According to a classical result of the perturbation theory [55, Theorem 15.9.1], which is a direct extension of (2.1), there is an eigenvalue λ_l of problem (2.2) for a scalar μ and non-zero vector $\boldsymbol{u}$ such that:

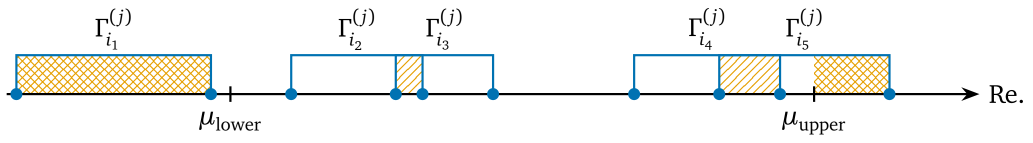
$$|\lambda_l - \mu| \le \frac{\|(A - \mu B)\boldsymbol{u}\|_{B^{-1}}}{\|B\boldsymbol{u}\|_{B^{-1}}}. \tag{3.2}$$

By substituting $(\mu, \boldsymbol{u})$ in (3.2) with an approximate eigenpair $(\lambda_i^{(j)}, \boldsymbol{x}_{i,2}^{(j)})$ obtained by the SI Lanczos method, we have from (2.12) and residual (2.13) that:

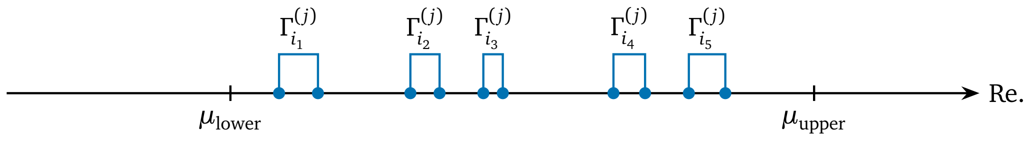
$$|\lambda_{l_i^{(j)}} - \lambda_i^{(j)}| \le \frac{\|(A - \lambda_i^{(j)} B)\boldsymbol{x}_{i,2}^{(j)}\|_{B^{-1}}}{\|B\boldsymbol{x}_{i,2}^{(j)}\|_{B^{-1}}} = \frac{\|\boldsymbol{r}_{i,2}^{(j)}\|_{B^{-1}}}{\|\boldsymbol{x}_{i,2}^{(j)}\|_{B}} = \frac{1}{|\tilde{\theta}_i^{(j)}|} \cdot \frac{|\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)} / \tilde{\theta}_i^{(j)}|}{\sqrt{1 + |\tilde{\beta}_j \boldsymbol{e}_j^{\mathrm{T}} \tilde{\boldsymbol{y}}_i^{(j)} / \tilde{\theta}_i^{(j)}|^2}}. \tag{3.3}$$

By rewriting inequality (3.3), we have an eigenvalue error bound $\Gamma_i^{(j)}$ for $\lambda_i^{(j)}$ that includes an eigenvalue $\lambda_{l_i^{(j)}}$ of problem (2.2). □

When bound (3.1) becomes sufficiently narrow, we can associate $\lambda_i^{(j)}$ with an eigenvalue of problem (2.2) and thus validate the index of the approximate eigenpairs. Here, assume that an interval $[\mu_{\mathrm{lower}}, \mu_{\mathrm{upper}})$ contains m eigenvalues. If there are m error bounds in the interval that are mutually disjoint, as illustrated in Figure 3.2, then each bound contains only one of the m eigenvalues of the interval. When the approximate eigenpairs have one-to-one correspondence with the eigenvalues of the interval, the index of each approximate eigenpair can be validated readily because we already know the index range of the eigenpairs of the interval from the bisection.



(a) $\Gamma_{i_1}^{(j)}$ and $\Gamma_{i_5}^{(j)}$ are not included in the interval (crosshatched). $\Gamma_{i_2}^{(j)}$ and $\Gamma_{i_3}^{(j)}$ overlap, so do $\Gamma_{i_4}^{(j)}$ and $\Gamma_{i_5}^{(j)}$ (hatched).



(b) $m = 5$ mutually disjoint bounds in the interval

Figure 3.2: Error bounds (3.1) for $m = 5$

Algorithm 3.3 shows an implementation example, where the midpoint $\mu = (\mu_{\mathrm{lower}} + \mu_{\mathrm{upper}})/2$ of the interval is selected as the shift for the SI Lanczos method. In line 8, we sort the approximate eigenpairs $(\lambda_i^{(j)}, \boldsymbol{x}_{i,2}^{(j)})$ of the method and re-index them for simplicity:

$$|\lambda_1^{(j)} - \mu| \le |\lambda_2^{(j)} - \mu| \le \cdots \le |\lambda_j^{(j)} - \mu|, \text{ or equivalently } |\tilde{\theta}_1^{(j)}| \ge |\tilde{\theta}_2^{(j)}| \ge \cdots \ge |\tilde{\theta}_j^{(j)}|. \tag{3.4}$$

Then, if the following two conditions hold, each $\lambda_i^{(j)}$ best approximates a distinct eigenvalue of the interval:

$$\text{inclusion: } \Gamma_i^{(j)} \subset [\mu_{\text{lower}}, \mu_{\text{upper}}) \text{ for } 1 \le i \le m. \tag{3.5}$$

$$\text{disjointness: } \Gamma_{i_1}^{(j)} \cap \Gamma_{i_2}^{(j)} = \emptyset \text{ for } 1 \le i_1 < i_2 \le m. \tag{3.6}$$

When (3.5) and (3.6) hold, we test for convergence in line 10. The algorithm is assumed to reach convergence when the relative residual 2-norm of each approximate eigenpair of the interval becomes less than a given tolerance τ_{res}. We also utilize the following criterion for the convergence test, which we refer to as the relative difference 2-norm between the approximate eigenvectors at the $(j-1)$-st and j-th iterations:

$$\|\boldsymbol{x}_{i,2}^{(j)} - \boldsymbol{x}_{i,2}^{(j-1)}\|_2 / \|\boldsymbol{x}_{i,2}^{(j-1)}\|_2 < \tau_{\text{diff}} \text{ for } 1 \le i \le m. \tag{3.7}$$

Here, $\boldsymbol{x}_{i,2}^{(j-1)}$ and $\boldsymbol{x}_{i,2}^{(j)}$ are normalized to satisfy $\|\boldsymbol{x}_{i,2}^{(j-1)}\|_2 = \|\boldsymbol{x}_{i,2}^{(j)}\|_2$. Criterion (3.7) is required because, as mentioned in Section 2.1.1 and will be discussed in Section 3.2.3, a small relative residual 2-norm does not necessarily imply that an approximate eigenvector is close to convergence. After the convergence test, we sort the approximate eigenpairs of the interval to the original order in line 13 to obtain the k-th eigenpair in line 14.

```
Algorithm 3.3: Computing the $k$-th eigenpair in the interval
  Input  : matrices $A, B$ of HDGEP (2.2), target index $k$,
           interval $[\mu_{\text{lower}}, \mu_{\text{upper}}) \subset \mathbb{R}$ containing the $k$-th eigenvalue,
           $\nu_{\text{lower}} = n_{\mu_{\text{lower}}}(A,B)$, $\nu_{\text{upper}} = n_{\mu_{\text{upper}}}(A,B)$,
           tolerances $\tau_{\text{res}}, \tau_{\text{diff}} \in \mathbb{R}$ for relative residual and difference 2-norms.
  Output: approximate eigenpair $(\hat{\lambda}_k, \hat{\boldsymbol{x}}_k)$, where $\|(A - \hat{\lambda}_k B)\hat{\boldsymbol{x}}_k\|_2 / \|\hat{\boldsymbol{x}}_k\|_2 < \tau_{\text{res}}$.
1  $l := k - \nu_{\text{lower}}$, $m := \nu_{\text{upper}} - \nu_{\text{lower}}$, $\mu := (\mu_{\text{lower}} + \mu_{\text{upper}})/2$,
2  set a random starting vector $\tilde{\boldsymbol{v}}_1$, $\tilde{\boldsymbol{v}}_1 := \tilde{\boldsymbol{v}}_1 / \|\tilde{\boldsymbol{v}}_1\|_{B^{-1}}$,
3  for $j = 1, 2, \dots$ do
4      compute $j$-step SI Lanczos decomposition (2.6) with reorthogonalization,
5      if $j \ge m$ then
6          solve standard eigenvalue problem (2.7),
7          for $i = 1$ to $j$ do compute approximate eigenpairs $(\lambda_i^{(j)}, \boldsymbol{x}_{i,2}^{(j)})$ as (2.8) and (2.10),
8          sort $(\lambda_i^{(j)}, \boldsymbol{x}_{i,2}^{(j)})$ with $1 \le i \le j$ to index as (3.4),
9          for $i = 1$ to $m$ do compute the $i$-th eigenvalue error bound (3.1),
10         if (3.5) and (3.6) and ($\|(A - \lambda_i^{(j)} B)\boldsymbol{x}_{i,2}^{(j)}\|_2 / \|\boldsymbol{x}_{i,2}^{(j)}\|_2 < \tau_{\text{res}}$ for $1 \le i \le m$) and (3.7)
             then break,
11     end if
12 end for
13 sort $(\lambda_i^{(j)}, \boldsymbol{x}_{i,2}^{(j)})$ with $1 \le i \le m$ to index in increasing order $\lambda_1^{(j)} < \lambda_2^{(j)} < \cdots < \lambda_m^{(j)}$,
14 $(\hat{\lambda}_k, \hat{\boldsymbol{x}}_k) := (\lambda_l^{(j)}, \boldsymbol{x}_{l,2}^{(j)})$.
```

3.1.4 Overview of the three-stage algorithm

Algorithm 3.4 presents a three-stage algorithm for solving k-EP. We first run Algorithm 3.1 to set an interval $[\mu_{\text{lower}}, \mu_{\text{upper}})$ containing the k-th eigenvalue. We then run Algorithm 3.2 to narrow down the interval until it contains less than or equal to $m_{\max}$ eigenvalues that include λ_k. Compared with further narrowing down the interval to isolate λ_k from the other eigenvalues, approximately $\log_2 m_{\max}$ bisection iterations can be reduced. Finally, we run Algorithm 3.3 to compute the k-th eigenpair along with the other eigenpairs of the interval while validating their index.

Algorithm 3.4: Three-stage algorithm for solving the k-th eigenvalue problem

```
Input  : matrices $A, B$ of HDGEP (2.2), target index $k \in \mathbb{N}$, stopping criterion $m_{\max} \in \mathbb{N}$,
         tolerances $\tau_{\text{res}}, \tau_{\text{diff}} \in \mathbb{R}$ for relative residual and difference 2-norms.
Output: approximate eigenpair $(\hat{\lambda}_k, \hat{\boldsymbol{x}}_k)$.
1 run Algorithm 3.1 to set an interval $[\mu_{\text{lower}}, \mu_{\text{upper}})$ containing $\lambda_k$ and obtain
    $\nu_{\text{lower}} = n_{\mu_{\text{lower}}}(A, B)$ and $\nu_{\text{upper}} = n_{\mu_{\text{upper}}}(A, B)$,
2 run Algorithm 3.2 to narrow down the interval $[\mu_{\text{lower}}, \mu_{\text{upper}})$ until it contains less than or
    equal to $m_{\max}$ eigenvalues that include $\lambda_k$,
3 run Algorithm 3.3 to obtain the $k$-th eigenpair $(\hat{\lambda}_k, \hat{\boldsymbol{x}}_k)$ with its relative residual and
    difference 2-norms less than $\tau_{\text{res}}$ and $\tau_{\text{diff}}$, respectively.
```

Here, $m_{\max}$ is an important parameter that influences the overall performance of the three-stage algorithm because there is a trade-off between the second and third stages (Algorithms 3.2 and 3.3), i.e., greater $m_{\max}$ results in fewer required bisection iterations, while more iterations are required for the SI Lanczos method. $m_{\max}$ was set the same for each problem in the numerical experiments. Tuning this parameter will be the focus of future work.

In the presence of multiple eigenvalues or a cluster of eigenvalues around λ_k, modification to Algorithm 3.4 is necessary because the algorithm is ineffective in detecting them and may end up in misconvergence; the stopping criterion (line 1 of Algorithm 3.2) for bisection and a convergence criterion (3.6) for the SI Lanczos method may not be satisfied. In addition, from the SI Lanczos method, only a one dimensional representation is obtained for the eigenspace associated to a multiple eigenvalue. To detect multiple or a cluster of eigenvalues during bisection, the length of the interval can be used along with the current stopping criterion. If detected, they and their associated eigenspace can be computed by, e.g., a block SI Lanczos method [21] whose block size can be determined from the number of eigenvalues in the interval. When a block eigensolver is used, the convergence criteria (line 10 of Algorithm 3.3) need to be modified accordingly to take account of the multiplicity or the cluster. Further investigation of the modification will be future work.

3.2 Numerical experiments

This section reports the numerical results of several real research problems from electronic structure calculations and a comparison of the proposed three-stage algorithm (Algorithm 3.4) and dense eigensolvers. In Section 3.2.1, we describe the matrix data used in the numerical experiments. Section 3.2.2 provides implementation details of dense eigensolvers and the three-stage algorithm. The numerical results are reported in Section 3.2.3.

3.2.1 Matrix data

Table 3.1 shows the matrix data used in the numerical experiments, which were generated by the ELSES quantum mechanical nanomaterial simulator [33] and obtained from the ELSES Matrix Library (`http://www.elses.jp/matrix/`). Here, the two $n \times n$ matrices A and B of each matrix data are real symmetric and real symmetric positive definite, respectively. A and B have the same sparsity structure shown in Figure 3.3 with #nz non-zero elements (in their lower triangular part) in Table 3.1. The target index k is associated with the HO state. The entire spectrum is included in the interval $[\lambda_1, \lambda_n]$. Footnotes below Table 3.1 describe the origin of the matrix data.

Table 3.1: Matrix data

Data	n	#nz	k	$[\lambda_1, \lambda_n]$
APF4686[1]	4686	53 950	2343	$[-1.157, 5.581]$
AUNW9180[2]	9180	1 783 313	5610	$[-0.210, 0.883]$
CPPE32346[3]	32 346	861 764	16 173	$[-1.169, 7.953]$
NCCS430080[4]	430 080	10 696 416	215 040	$[-1.195, 13.602]$
VCNT1512000[5]	1 512 000	294 953 351	336 000	$[-1.098, 0.475]$

[1] amorphous-like conjugated polymer, poly(9,9-dioctyl-fluorene) [33]
[2] helical multishell gold nanowire with defects [29]
[3] condensed polymer systems, poly(phenylene-ethynylene) [30, 36]
[4] sp^2–sp^3 nano-composite carbon solid [28]
[5] vibrating carbon nanotube within a supercell with spd orbitals [10]

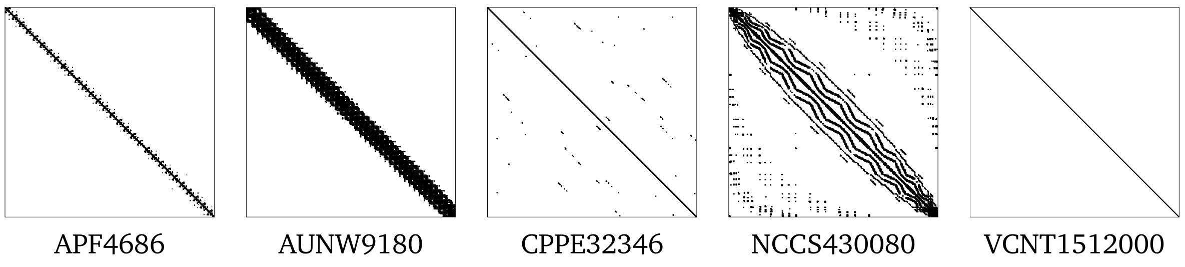


Figure 3.3: Sparsity structures of matrix data

3.2.2 Implementation details

As dense eigensolvers, we used the LAPACK [4] and ScaLAPACK [7] routines. Specifically, the LAPACK `dsygvd` routine was used to solve APF4686 and AUNW9180. In `dsygvd`, a generalized eigenvalue problem is transformed to a standard eigenvalue problem of a tridiagonal matrix, and then eigenpairs of the tridiagonal matrix are computed by the divide and conquer method [11, 22, 62]. To date, there is no single ScaLAPACK routine to perform the same task as `dsygvd` in parallel. Therefore, to solve CPPE32346 and NCCS430080, the ScaLAPACK `pdpotrf`, `pdsygst`, `pdsytrd`, `pdstedc`, and `pdormtr` routines were utilized through EigenKernel [17, 35]. Note that the results for VCNT1512000 are not provided because the problem size prevents it from being solved by a dense eigensolver in practical time.

In the three-stage algorithm (Algorithm 3.4), $n_\mu(A, B)$ and solution of linear systems in the Lanczos and SI Lanczos methods were computed based on LDL^{H} factorization by the MUMPS sparse direct linear solver [2, 3] with the METIS fill-reducing ordering [38]. The bisection narrowed down the initial interval until the number of eigenvalues in the interval became less than or equal to $m_{\max} = 20$. Tolerances for the relative residual and difference 2-norms in the SI Lanczos method were set to $\tau_{\mathrm{res}} = 10^{-10}$ and $\tau_{\mathrm{diff}} = 10^{-10}$, respectively. All codes were written in Fortran 90, and the numerical experiments were performed in double-precision.

3.2.3 Results

In this subsection, we first compare the k-th eigenpair computed by Algorithm 3.4 with that obtained by the dense eigensolvers described in Section 3.2.2 and report the computation time with some details about the computational environment and implementation. Then, in Sections 3.2.3 to 3.2.3, we present the detailed results of each algorithm (i.e., Algorithms 3.1, 3.2, and 3.3) of the three-stage algorithm.

Table 3.2 compares the k-th eigenpair of the three-stage algorithm $(\hat{\lambda}_k, \hat{\boldsymbol{x}}_k)$ and that of the dense eigensolvers $(\lambda_k^{(\mathrm{d})}, \boldsymbol{x}_k^{(\mathrm{d})})$. As can be seen, at least 15 digits were the same for $\hat{\lambda}_k$ and $\lambda_k^{(\mathrm{d})}$. The last column shows the relative error 2-norm, where $\hat{\boldsymbol{x}}_k$ and $\boldsymbol{x}_k^{(\mathrm{d})}$ were normalized to satisfy $\|\hat{\boldsymbol{x}}_k\|_2 = \|\boldsymbol{x}_k^{(\mathrm{d})}\|_2$. The error norm had an order of magnitude less than -10, indicating that the k-th eigenvector of the three-stage algorithm agrees well with that of the dense eigensolvers.

Table 3.2: k-th eigenpair

Data	k	$\hat{\lambda}_k$	$\lambda_k^{(\mathrm{d})}$	$\frac{\lvert\hat{\lambda}_k-\lambda_k^{(\mathrm{d})}\rvert}{\lvert\lambda_k^{(\mathrm{d})}\rvert}$	$\frac{\|\hat{\boldsymbol{x}}_k-\boldsymbol{x}_k^{(\mathrm{d})}\|_2}{\|\boldsymbol{x}_k^{(\mathrm{d})}\|_2}$
APF4686	2343	−0.4258775547956963	−0.4258775547956963	0	4×10^{-14}
AUNW9180	5610	0.1305388835941175	0.1305388835941177	2×10^{-15}	1×10^{-12}
CPPE32346	16 173	−0.4332412034185730	−0.4332412034185731	2×10^{-16}	7×10^{-14}
NCCS430080	215 040	−0.3689638375042860	−0.3689638375042869	2×10^{-15}	4×10^{-11}
VCNT1512000	336 000	−0.5517499297808635	n/a	n/a	n/a

Table 3.3 shows the total computation time and computational resources consumed by Algorithm 3.4 (Alg. 4), its variant (Ger.), and the dense eigensolvers (Dense). In the variant, line 1 of Algorithm 3.4 was changed to set an interval including the entire spectrum based on the Gershgorin circle theorem[1]. Here, #Core is the number of cores used in the experiments, and superscripts $^{(\mathrm{w})}$ and $^{(\mathrm{K})}$ represent a workstation and the K computer, respectively. Memory indicates peak memory usage. Actual measurement of the memory usage was performed using the GNU `time` command. Estimation (in italics) shows the memory required to store $4n^2$ double-precision numbers, which is based on the memory requirement of the LAPACK `dsygvd` routine.

Details about the computation time and implementation of the three-stage algorithm are shown in Figure 3.4, where the total time is scaled to one. As described in the figure legend, our implementation can be divided into the following seven major computational tasks. (i) B is pre-processed in a symbolic manner to produce a fill-reducing ordering and an elimination tree for its LDL^{H} factorization. The ordering is recycled for the LDL^{H} factorization of shifted matrices $A-\mu B$ because matrices A and B have the same sparsity structure in our numerical experiments. (ii) Based on the symbolic factorization, the numerical factorization of B is computed to solve linear systems in the Lanczos method. (iii) Ritz values are computed to set an initial interval. (iv–vi) Numerical factorization of shifted matrices is computed to set an initial interval, bisect the interval, and solve the linear systems in the SI Lanczos method. (vii) The k-th eigenpair is computed.

As can be seen in Figure 3.4, the second stage dominated computation time as the problem size increases. This is because, as the problem size increases, more bisection iterations are expected to be required to narrow down an initial interval in order to make the number of eigenvalues in the

[1]Instead of some Gershgorin-type theorem, an inclusion set of the spectrum of $B^{-1}A$ was computed based on the original theorem because the diagonal dominance of A and B (described in Section 3.1.1) does not hold for all matrix data. To compute the inclusion set, columns of $B^{-1}A$ were obtained by solving linear systems with MUMPS, and then Gershgorin disks were calculated from the columns. In the VCNT1512000 case, the linear systems were solved in single-precision to reduce the time to solution.

interval less than or equal to $m_{\max} = 20$, which was the same value regardless of the problem size.

Table 3.3: Computation time and computational resources consumed by Algorithm 3.4, its variant, and dense eigensolvers

Data	Time (s)			#Core			Memory (MB)		
	Alg. 4	Ger.	Dense	Alg. 4	Ger.	Dense	Alg. 4	Ger.	Dense
APF4686	0.3	0.8	82.1			$1^{(w)}$	13	12	629
AUNW9180	19.4	69.0	655.2			$1^{(w)}$	267	245	2836
CPPE32346	4.4	194.0	1366.8	$1^{(w)}$	$1^{(w)}$	$32^{(K)}$	121	116	*33 480*
NCCS430080	2024	109 597	10 586			$180\,000^{(K)}$	5069	5055	*5 919 002*
VCNT1512000	5132	602 525	n/a			n/a	38 242	31 618	n/a

$^{(w)}$ workstation with Xeon E5-2690 (2.90 GHz)
$^{(K)}$ K computer with SPARC64 VIIIfx (2.00 GHz) and Tofu interconnect

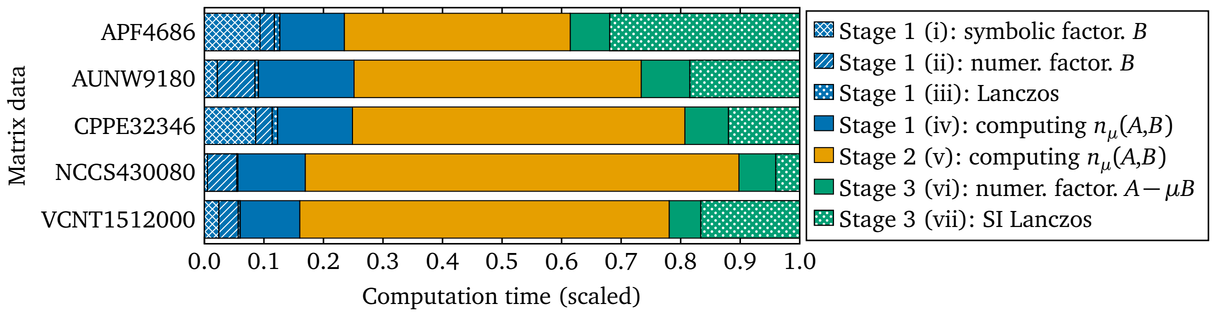


Figure 3.4: Computation time of stages and computational tasks

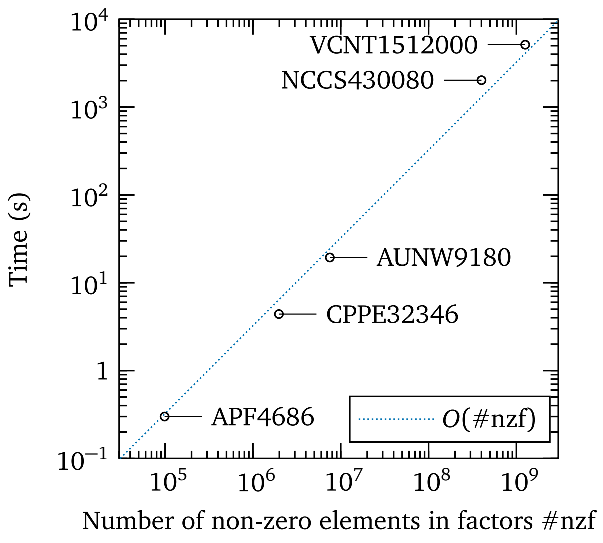


Figure 3.5: Computation time vs. number of non-zero elements in factors L and D of LDL^{H} factorization

Figure 3.5 shows the relationship between total computation time and the number of non-zero elements in factors L and D of LDL^{H} factorization, denoted #nzf, in log–log scale. Here, #nzf is the average of the factorizations of B and shifted matrices $A-\mu B$ with varying μ. The dotted line in the figure is of slope one, which corresponds to the linear scaling $O(\#\text{nzf})$. As can be seen, computation time is proportional to #nzf (the slope for linear least squares fitting of the data points is 1.06). This is because the most time-consuming tasks in the three-stage algorithm, i.e., computation of $n_\mu(A,B)$ and solving linear systems in the Lanczos and SI Lanczos methods, are performed based on factorizations by a sparse direct linear solver. Generally, the estimation of #nzf can be obtained in the symbolic factorization stage, which can be utilized to predict total computation time.

Initial interval

Table 3.4 shows the initial interval $[\mu_{\text{lower}}, \mu_{\text{upper}})$ obtained by Algorithm 3.1. The fourth column shows the length $\mu_{\text{upper}}-\mu_{\text{lower}}$ of the interval, which contains m eigenvalues with their index ranging from i_{lower} to i_{upper}. The seventh column shows the average gap between the eigenvalues in the interval defined as Length$/m$. In the last column, we compare the length of the initial interval with that of $[\lambda_1, \lambda_n]$ in Table 3.1 by calculating the ratio of $(\lambda_n-\lambda_1)/$Length. The initial intervals were 3.2 to 29.5 times narrower than $[\lambda_1, \lambda_n]$, i.e., the tightest interval that can be obtained from some Gershgorin-type theorem in general. Note that all problems required two iterations of Algorithm 3.1 (thus, two LDL^{H} factorizations) to set the interval, which is the minimum required iterations to obtain an interval validated as containing λ_k.

Bisection

The initial interval in Table 3.4 was narrowed down to the interval $[\mu_{\text{lower}}, \mu_{\text{upper}})$ in Table 3.5 based on Algorithm 3.2. Figure 3.6 shows the number of eigenvalues in the interval after each bisection iteration in log scale. The horizontal dotted line in the figure indicates the stopping criterion $m_{\max}=20$ for the bisection. In most cases, the number of eigenvalues was approximately halved after each iteration. However, the number remained unchanged after the fifth iteration of APF4686 and the sixth iteration of CPPE32346. In addition, there was a sharp decrease in the number of eigenvalues at the final iteration of CPPE32346, in which the number decreased by more than an order of magnitude. This convergence behavior implies that eigenvalues are distributed in a highly non-uniform manner and that there are clusters of eigenvalues or large gaps between eigenvalues. Indeed, in the CPPE32346 case, Gap in Table 3.5 is approximately 40 times greater than that shown in Table 3.4.

Here, we note that the straightforward bisection[2] required 47 to 49 iterations, which were roughly 4 to 8 times greater than those required for the three-stage algorithm shown in Figure 3.6 (6 to 12 iterations), to compute the k-th eigenvalue to the accuracy very similar to that in Table 3.2.

Computation of the k-th eigenpair

In Table 3.6, we show the iteration counts of Algorithm 3.3 for computing m eigenvalues in the interval $[\mu_{\text{lower}}, \mu_{\text{upper}})$ of Table 3.5. The third column is the iteration count required for (3.5) and (3.6) to be satisfied such that the index of each approximate eigenpair of the interval is validated. The fourth and fifth columns represent the iteration counts required for the relative 2-norm of the residual and difference (3.7) of each approximate eigenpair of the interval to become less than $\tau_{\text{res}}=10^{-10}$ and $\tau_{\text{diff}}=10^{-10}$, respectively.

Figure 3.7 shows the convergence history of the k-th eigenpair ($k=215\,040$) of NCCS430080. $(\hat{\lambda}_k^{(j)}, \hat{\boldsymbol{x}}_k^{(j)})$ in the figure legend denotes the k-th eigenpair computed at the j-th iteration of Algorithm

[2]In the straightforward bisection, line 1 of Algorithm 3.2 was changed to use the length of the interval as a stopping criterion. Specifically, the stopping criterion was set to $(\mu_{\text{upper}}-\mu_{\text{lower}})/\max\{|\mu_{\text{lower}}|, |\mu_{\text{upper}}|\} < 10^{-14}$. The k-th eigenvalue was computed using only bisection and from the equation $\hat{\lambda}_k = (\mu_{\text{lower}}+\mu_{\text{upper}})/2$.

Table 3.4: Initial interval

Data	k	$[\mu_{\text{lower}}, \mu_{\text{upper}})$	Length	$[i_{\text{lower}}, i_{\text{upper}}]$	m	Gap	Ratio
APF4686	2343	[−0.777, 0.475)	1.252	[751,3458]	2708	5×10^{-4}	5.4
AUNW9180	5610	[−0.079, 0.186)	0.265	[877,5853]	4977	5×10^{-5}	4.1
CPPE32346	16 173	[−0.731, 1.023)	1.754	[5586,26 409]	20 824	8×10^{-5}	5.2
NCCS430080	215 040	[−0.777,−0.275)	0.502	[64 252,224 635]	160 384	3×10^{-6}	29.5
VCNT1512000	336 000	[−0.920,−0.429)	0.491	[84 320,422 420]	338 101	1×10^{-6}	3.2

Table 3.5: Interval narrowed down by bisection

Data	k	$[\mu_{\text{lower}}, \mu_{\text{upper}})$	Length	$[i_{\text{lower}}, i_{\text{upper}}]$	m	Gap
APF4686	2343	[−0.44450,−0.42494)	0.01956	[2334,2343]	10	2×10^{-3}
AUNW9180	5610	[0.12826, 0.13240)	0.00414	[5601,5615]	15	3×10^{-4}
CPPE32346	16 173	[−0.43657,−0.42971)	0.00685	[16 172,16 173]	2	3×10^{-3}
NCCS430080	215 040	[−0.36897,−0.36884)	0.00013	[215 040,215 049]	10	1×10^{-5}
VCNT1512000	336 000	[−0.55178,−0.55166)	0.00012	[335 995,336 010]	16	8×10^{-6}

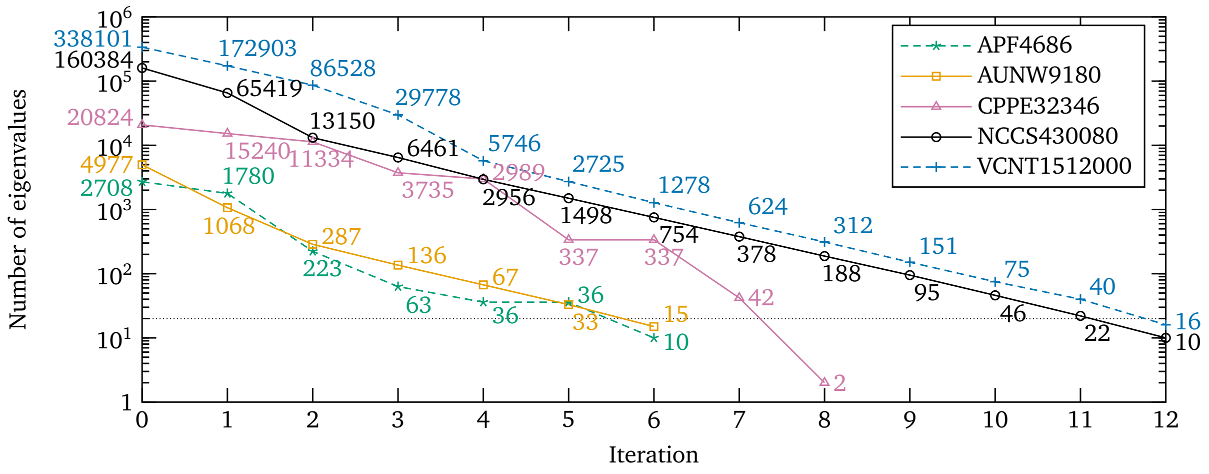


Figure 3.6: Number of eigenvalues in an interval after each bisection iteration

3.3. $\boldsymbol{x}_k^{(\mathrm{d})}$ represents the k-th eigenvector computed by the dense eigensolver. As described in the legend, the figure shows the relative 2-norm history of (i) the residual, (ii) the difference between the $(j-1)$-st and j-th iterations defined in (3.7), and (iii) the error compared with the dense eigensolver. The figure also shows (iv) the pairwise B-orthogonality (2.11) between the k-th eigenvector and the other $m-1$ eigenvectors of the interval in Table 3.5. Here, the three vertical dotted lines indicate the iteration counts of Bound, Residual, and Difference in Table 3.6. The horizontal dotted line indicates the convergence criteria $\tau_{\mathrm{res}} = 10^{-10}$ and $\tau_{\mathrm{diff}} = 10^{-10}$.

As can be seen in Figure 3.7, a small residual norm does not necessarily imply that eigenvector $\hat{\boldsymbol{x}}_k^{(j)}$ is close to convergence. Indeed, the residual norm converged first, and convergence of the error and difference norms followed. Since the error norm cannot be measured in general, the difference norm (3.7) is utilized in Algorithm 3.3 to test for convergence.

Table 3.6: Iteration counts of the SI Lanczos method

Data	m	Iteration		
		Bound	Residual	Difference
APF4686	10	24	33	37
AUNW9180	15	26	42	48
CPPE32346	2	7	19	23
NCCS430080	10	23	31	41
VCNT1512000	16	27	39	50

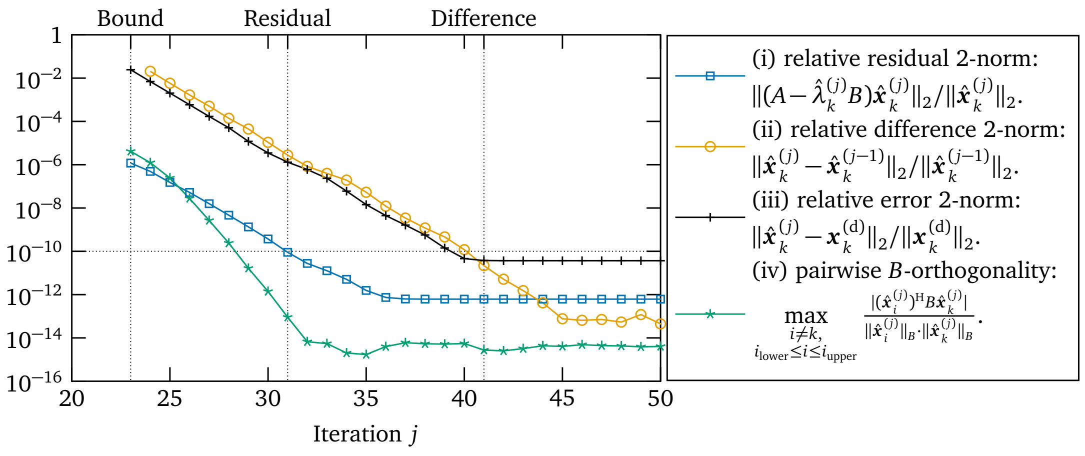


Figure 3.7: Convergence history of the k-th eigenpair ($k = 215\,040$) of NCCS430080

3.3 Concluding remarks

The proposed three-stage algorithm obtained the validated k-th eigenpair $(\lambda_k, \boldsymbol{x}_k)$ for large sparse HDGEP with accuracy comparable with dense eigensolvers under limited computational resources. The three-stage algorithm (Algorithm 3.4) consists of Algorithms 3.1, 3.2, and 3.3, each of which has been found to be effective for computation of the eigenpair and validation of its index. In particular, from the numerical experiments, we have learned the following.

1. Algorithm 3.1 can set a narrow interval containing λ_k. The resulting intervals were 3 to 29 times narrower than $[\lambda_1, \lambda_n]$, i.e., the tightest interval that can be obtained from some Gershgorin-

type theorem in general. In all experiments, only two iterations of Algorithm 3.1 were required, i.e., the minimum required iterations to obtain an interval validated as containing λ_k.

2. Algorithm 3.3 can compute the k-th eigenpair with high accuracy. The eigenpairs computed by Algorithm 3.3 agreed well with the results of dense eigensolvers, including a result obtained by a massively parallel eigenpair computation on the K computer. Specifically, the eigenvalues were the same to at least 15 digits, and the relative error norms of the eigenvectors were less than 10^{-10}.

3. By utilizing a sparse direct linear solver, large sparse matrices can be handled with efficiency. For example, a nano-composite carbon solid problem of size $n = 430\,080$ was solved in 0.6 hours using one core and 5.1 GB of memory on a workstation (a dense eigensolver required 2.9 hours using 180 000 cores and an estimated 5.9 TB of memory on the K computer).

Fortran codes for the three-stage algorithm are available online [37] as free and open source software under the MIT license.

In future, we plan to examine parameter $m_{\max}$. As explained in Section 3.1.4, this parameter influences the overall performance of the three-stage algorithm because there is a trade-off between the second and third stages (Algorithms 3.2 and 3.3). In addition, we plan to compare bisection with its variants. As long as algorithms are derivative-free and a root is bracketed in the algorithms, they can be readily applied to the second stage and can locate λ_k. Brent's method [8] is one such algorithm. Finally, we plan to modify the three-stage algorithm to deal with the presence of multiple eigenvalues or a cluster of eigenvalues. Some possible modifications are described in Section 3.1.4.

Chapter 4

An algorithm for the k-th singular value problem

This chapter is about solution of k-SVP (1.4) of a large sparse $A \in \mathbb{C}^{m\times n}$ ($m \geq n$), which is based on the three-stage algorithm proposed in the previous chapter. If $m < n$, we consider k-SVP of A^{H}.

Several subspace methods have been proposed for computing a specific subset of singular triplets and are in close relationship with methods for HEP. The Golub–Kahan bidiagonalization method [19], which is equivalent to applying the Lanczos method to the extended matrix $\left[\begin{smallmatrix} O & A \\ A^{\mathrm{H}} & O \end{smallmatrix}\right]$ with a special initial vector, can compute the smallest and largest singular values and their associated left and right singular vectors. JDSVD [23,24], which is a Jacobi–Davidson [59] type method exploiting the block structure of the extended matrix, can compute singular values closest to a target point and their associated left and right singular vectors. PHSVDS [65], which solves HEP of a Gram matrix $A^{\mathrm{H}}A$ or the extended matrix depending on the required accuracy, can compute the smallest and largest singular values and their associated left and right singular vectors.

The k-th singular triplet is different from typical singular triplets that can be computed by existing subspace methods. In addition, because of the required memory, it is infeasible to obtain the k-th triplet of large matrices by computing their singular value decomposition (SVD).

Based on the relation between HEP and SVP and utilizing the three-stage algorithm of the previous chapter, this chapter presents an algorithm for computing the k-th singular triplet. The remainder of this chapter is organized as follows. Section 4.1 recaps the three-stage algorithm (Algorithm 3.4) for HDGEP with $B = I$ (which means HEP). Section 4.2 explains relation between HEP and SVP and presents a three-stage algorithm for k-SVP. Section 4.3 reports results of numerical experiments and examines the accuracy and efficiency of the proposed algorithm for k-SVP. Concluding remarks are given in Section 4.4.

4.1 A three-stage algorithm for Hermitian eigenvalue problems

Let $H \in \mathbb{C}^{n\times n}$ be a large sparse Hermitian matrix with the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.[1] For $1 \ll k \ll n$, Algorithm 4.0 solves the problem of computing the k-th eigenpair $(\lambda_k, \boldsymbol{x}_k)$ of H by dividing the problem into three stages of independent purposes, as Algorithm 3.4.

```
Algorithm 4.0: Three-stage algorithm for computing the $k$-th eigenpair (HEP)
  Input  : Hermitian $H \in \mathbb{C}^{n\times n}$, target index $k \in \mathbb{N}$, parameters $l \in \mathbb{N}$ and $\tau \in \mathbb{R}$.
  Output: approximate eigenpair $(\hat{\lambda}_k, \hat{\boldsymbol{x}}_k)$.
1 run Algorithm 4.1 to set an interval $(\alpha, \beta]$ containing $\lambda_k$,
2 run Algorithm 4.2 to narrow down $(\alpha, \beta]$ until it contains less than or equal to $l$ eigenvalues,
3 run Algorithm 4.3 to compute $(\lambda_k, \boldsymbol{x}_k)$ in $(\alpha, \beta]$ with its relative residual 2-norm less than $\tau$.
```

[1] In contrast to the previous chapter in which eigenvalues of HDGEP are index in increasing order, eigenvalues of HEP in this chapter are indexed in decreasing order, as singular values.

Each line of Algorithm 4.0 corresponds to one of the three stages. Algorithm 4.1 in line 1 is for finding a narrow interval containing λ_k in an efficient way. Algorithm 4.2 in line 2 narrows down the interval containing λ_k until the number of eigenvalues in the interval becomes small enough. Algorithm 4.3 in line 3 computes a small number of eigenvalues in the interval and their associated eigenvectors. Here, parameter l is the stopping criterion for Algorithms 4.2, and τ is a parameter for the convergence test in Algorithm 4.3.

The algorithm for each stage can be developed independently. The rest of the section shows Algorithms 4.1, 4.2, and 4.3, which correspond to Algorithms 3.1, 3.2 and 3.3 of the previous chapter, respectively.

The only notable difference between the three algorithms in the previous chapter and those in the current section is that we now compute the number of eigenvalues of H greater than μ, denoted $p_\mu(H)$, instead of $n_\mu(A,B)$. The tools of the trade are the same as Section 3.1.2. Based on Sylvester's law of inertia and by using a sparse direct linear solver, $p_\mu(H)$ is computed from LDL^{H} factorization of shifted matrix $H-\mu I$,

$$PQ(H-\mu I)Q^{\mathrm{T}}P^{\mathrm{T}} = LDL^{\mathrm{H}}, \tag{4.1}$$

from which we have $p_\mu(H) = p(H-\mu I) = p(D)$. Here, P and Q are permutation for numerical stability and fill-reducing, respectively. Matrix L is unit lower triangular, and D is a block diagonal matrix with block size one or two. Once $p_\mu(H)$ is computed, we can locate λ_k based on the following relations:

$$p_\mu(H) \geq k \iff \mu < \lambda_k, \quad p_\mu(H) < k \iff \mu \geq \lambda_k. \tag{4.2}$$

Stage 1

As Algorithm 3.1, Algorithm 4.1 finds an initial interval containing λ_k by using the smallest or largest Ritz values of the Lanczos method. We first recap the Lanczos method and the interlacing property of the Ritz values. Let j-step Lanczos decomposition be

$$HW_j = W_jT_j + \eta_j \boldsymbol{w}_{j+1}\boldsymbol{e}_j^{\mathrm{T}}, \tag{4.3}$$

where $W_j = [\boldsymbol{w}_1 \ \cdots \ \boldsymbol{w}_j]$ is an $n \times j$ matrix whose columns are orthonormal. Matrix $T_j = W_j^{\mathrm{H}}AW_j$ is real symmetric tridiagonal and irreducible. Vector $\boldsymbol{w}_{j+1}$ is orthogonal to the columns of W_j and is normalized by scale factor η_j. Eigenvalues $\theta_i^{(j)}$ of T_j (Ritz values) are indexed in decreasing order, or $\theta_1^{(j)} > \theta_2^{(j)} > \cdots > \theta_j^{(j)}$. Eigenvalues of T_j and T_{j+1} have the following interlacing property:

$$\theta_i^{(j+1)} > \theta_i^{(j)} > \theta_{i+1}^{(j+1)} \quad (1 \leq i \leq j). \tag{4.4}$$

Algorithm 4.1 utilizes the interlacing property (4.4) for selecting the endpoints of an initial interval. At its first iteration ($j=1$), Algorithm 4.1 computes $\mu^{(1)} = \theta_1^{(1)}$ and $p^{(1)} = p_{\mu^{(1)}}(H)$. For the rest of the iterations ($j \geq 2$), by comparing a given index k with $p^{(j-1)}$, either the smallest $\theta_j^{(j)}$ or largest $\theta_1^{(j)}$ eigenvalue is selected as $\mu^{(j)}$ in line 5, and it is checked in line 8 whether the interval whose endpoints are $\mu^{(j)}$ and $\mu^{(j-1)}$ contains λ_k. Sequences of intervals $\{(\theta_1^{(i-1)}, \theta_1^{(i)}]\}_{i=2}^{j}$ and $\{(\theta_i^{(i)}, \theta_{i-1}^{(i-1)}]\}_{i=2}^{j}$ are disjoint, and $(\theta_j^{(j)}, \theta_1^{(j)}]$ converges to $(\lambda_n, \lambda_1]$ as j becomes large. Therefore, for $k \neq n$, Algorithm 4.1 is able to find an interval containing λ_k.

Algorithm 4.1: Setting an interval containing the k-th eigenvalue (HEP)

```
Input  : Hermitian $H \in \mathbb{C}^{n\times n}$, target index $k \in \mathbb{N}$.
Output: interval $(\alpha,\beta]$ containing $\lambda_k$, $p_\alpha(H)$ and $p_\beta(H)$.
set a random unit vector $\boldsymbol{w}_1 \in \mathbb{C}^n$, $p^{(0)} := 0$,
for $j = 1,2,\dots$ do
    compute Lanczos decomposition (4.3),
    compute eigenvalues $\theta_i^{(j)}$ $(1 \le i \le j)$ of $T_j$,
    if $p^{(j-1)} \ge k$ then $\mu^{(j)} := \theta_1^{(j)}$ else $\mu^{(j)} := \theta_j^{(j)}$,
    $\mu := \mu^{(j)}$, compute $LDL^{\mathrm{H}}$ factorization (4.1),
    compute eigenvalue count $p^{(j)} := p_0(D)$,
    if $j \ne 1$ and $(p^{(j-1)} \ge k > p^{(j)}$ or $p^{(j)} \ge k > p^{(j-1)})$ then exit from the for-loop,
end for
$\alpha := \min\{\mu^{(j-1)}, \mu^{(j)}\}$, $\beta := \max\{\mu^{(j-1)}, \mu^{(j)}\}$,
$p_\alpha(H) := \max\{p^{(j-1)}, p^{(j)}\}$, $p_\beta(H) := \min\{p^{(j-1)}, p^{(j)}\}$.
```

Algorithm 4.2: Bisection for narrowing down an interval of the k-th eigenvalue (HEP)

```
Input  : Hermitian $H \in \mathbb{C}^{n\times n}$, target index $k \in \mathbb{N}$, parameter $l \in \mathbb{N}$,
         interval $(\alpha,\beta]$ containing $\lambda_k$, $p_\alpha(H)$ and $p_\beta(H)$.
Output: $(\alpha,\beta]$, $p_\alpha(H)$, $p_\beta(H)$.
while $p_\alpha(H) - p_\beta(H) > l$          ▷ $p_\alpha(H) - p_\beta(H)$: number of eigenvalues in $(\alpha,\beta]$.
    compute midpoint $\mu := (\alpha+\beta)/2$,
    compute $LDL^{\mathrm{H}}$ factorization (4.1),
    compute eigenvalue count $p := p_0(D)$,
    if $p \ge k$ then $\alpha := \mu$, $p_\alpha(H) := p$ else $\beta := \mu$, $p_\beta(H) := p$.
end while
```

Algorithm 4.3: Computing the k-th eigenpair in the interval (HEP)

```
Input  : Hermitian $H \in \mathbb{C}^{n\times n}$, target index $k$, parameter $\tau \in \mathbb{R}$,
         interval $(\alpha,\beta]$ containing $\lambda_k$, $p_\alpha(H)$ and $p_\beta(H)$.
Output: approximate eigenpair $(\hat{\lambda}_k, \hat{\boldsymbol{x}}_k)$.
$\tilde{l} := p_\alpha(H) - p_\beta(H)$,          ▷ $p_\alpha(H) - p_\beta(H)$: number of eigenvalues in $(\alpha,\beta]$.
compute midpoint $\mu := (\alpha+\beta)/2$, compute $LDL^{\mathrm{H}}$ factorization (4.1),
set a random unit vector $\tilde{\boldsymbol{w}}_1 \in \mathbb{C}^n$,
for $j = 1,2,\dots$ do
    compute Lanczos decomposition (4.5) using the $LDL^{\mathrm{H}}$ factorization in line 2,
    if $j \ge \tilde{l}$ then
        compute eigenpairs $(\tilde{\theta}_i^{(j)}, \tilde{\boldsymbol{y}}_i^{(j)})$ $(1 \le i \le j)$ of $\tilde{T}_j$, sort and index $(\tilde{\theta}_i^{(j)}, \tilde{\boldsymbol{y}}_i^{(j)})$ as (4.9),
        for $1 \le i \le \tilde{l}$ do compute approximate eigenpair $(\lambda_i^{(j)}, \boldsymbol{x}_i^{(j)})$ as (4.6),
                    ▷ $(\lambda_i^{(j)}, \boldsymbol{x}_i^{(j)})$ is indexed in increasing order (4.9) because of line 7.
        for $1 \le i \le \tilde{l}$ do compute interval $\Gamma_i^{(j)}$ as (4.7) and (4.8),
        if (4.10) and (4.11) and $(\|H\boldsymbol{x}_i^{(j)} - \lambda_i^{(j)}\boldsymbol{x}_i^{(j)}\|_2 / \|\boldsymbol{x}_i^{(j)}\|_2 < \tau$ for $1 \le i \le \tilde{l})$ then exit
          from the for-loop,
    end if
end for
sort and index $(\lambda_i^{(j)}, \boldsymbol{x}_i^{(j)})$ $(1 \le i \le \tilde{l})$ in decreasing order $\lambda_1^{(j)} > \lambda_2^{(j)} > \cdots > \lambda_{\tilde{l}}^{(j)}$,
$\tilde{k} := k - p_\beta(H)$, $(\hat{\lambda}_k, \hat{\boldsymbol{x}}_k) := (\lambda_{\tilde{k}}^{(j)}, \boldsymbol{x}_{\tilde{k}}^{(j)})$.
```

Stage 2

As Algorithm 3.2, Algorithm 4.2 narrows down an initial interval containing λ_k by bisecting the interval based on the relations (4.2) and runs until the number of eigenvalues in the interval becomes smaller than or equal to a given parameter l.

Stage 3

As Algorithm 3.3, Algorithm 4.3 computes eigenpairs of an interval based on a modified SI Lanczos method (Section 2.2.2). We first consider j-step SI Lanczos decomposition,

$$(H-\mu I)^{-1}\tilde{W}_j = \tilde{W}_j\tilde{T}_j + \tilde{\eta}_j\tilde{\boldsymbol{w}}_{j+1}\boldsymbol{e}_j^{\mathrm{T}}, \tag{4.5}$$

where $\mu \neq \lambda_i$ $(1 \le i \le n)$ and $\tilde{W}_j = [\tilde{\boldsymbol{w}}_1 \ \cdots \ \tilde{\boldsymbol{w}}_j]$ is an $n \times j$ matrix whose columns are orthonormal. Matrix $\tilde{T}_j = \tilde{W}_j^{\mathrm{H}}(H-\mu I)^{-1}\tilde{W}_j$ is real symmetric tridiagonal and irreducible. Vector $\tilde{\boldsymbol{w}}_{j+1}$ is orthogonal to the columns of $\tilde{W}_j$ and is normalized by scale factor $\tilde{\eta}_j$. Eigenvalues $\theta_i^{(j)}$ of $\tilde{T}_j$ are indexed in decreasing order, or $\tilde{\theta}_1^{(j)} > \tilde{\theta}_2^{(j)} > \cdots > \tilde{\theta}_j^{(j)}$, and their associated eigenvectors $\tilde{\boldsymbol{y}}_i^{(j)}$ are of unit norm, or $\|\tilde{\boldsymbol{y}}_i^{(j)}\|_2 = 1$. Approximate eigenpairs of H are set as follows:

$$(\lambda_i^{(j)}, \boldsymbol{x}_i^{(j)}) \equiv (\mu + 1/\tilde{\theta}_i^{(j)}, (H-\mu I)^{-1}\tilde{W}_j\tilde{\boldsymbol{y}}_i^{(j)}) = (\mu + 1/\tilde{\theta}_i^{(j)}, \tilde{W}_j\tilde{\boldsymbol{y}}_i^{(j)}\tilde{\theta}_i^{(j)} + \tilde{\boldsymbol{w}}_{j+1}\tilde{\eta}_j\boldsymbol{e}_j^{\mathrm{T}}\tilde{\boldsymbol{y}}_i^{(j)}). \tag{4.6}$$

As Proposition 3.1, we now consider an a posteriori error bound (2.1) of approximate eigenvalues, which leads to an interval

$$\Gamma_i^{(j)} \equiv [\lambda_i^{(j)} - \varepsilon_i^{(j)}, \lambda_i^{(j)} + \varepsilon_i^{(j)}], \tag{4.7}$$

$$\varepsilon_i^{(j)} \equiv \frac{\|H\boldsymbol{x}_i^{(j)} - \lambda_i^{(j)}\boldsymbol{x}_i^{(j)}\|_2}{\|\boldsymbol{x}_i^{(j)}\|_2} = \frac{1}{|\tilde{\theta}_i^{(j)}|}\cdot\frac{|\tilde{\eta}_j\boldsymbol{e}_j^{\mathrm{T}}\tilde{\boldsymbol{y}}_i^{(j)}/\tilde{\theta}_i^{(j)}|}{\sqrt{1+|\tilde{\eta}_j\boldsymbol{e}_j^{\mathrm{T}}\tilde{\boldsymbol{y}}_i^{(j)}/\tilde{\theta}_i^{(j)}|^2}} \tag{4.8}$$

containing at least one eigenvalue of H.

Here, let interval $(\alpha, \beta]$ contain $\tilde{l}$ eigenvalues and the shift μ of the modified SI Lanczos method be the midpoint of the interval, or $\mu = (\alpha+\beta)/2$. For the sake of simplicity of exposition, we sort approximate eigenvalues $(\tilde{\theta}_i^{(j)}, \tilde{\boldsymbol{y}}_i^{(j)})$ and re-index them as follows:

$$|\tilde{\theta}_1^{(j)}| \ge |\tilde{\theta}_2^{(j)}| \ge \cdots \ge |\tilde{\theta}_j^{(j)}| \iff |\lambda_1^{(j)} - \mu| \le |\lambda_2^{(j)} - \mu| \le \cdots \le |\lambda_j^{(j)} - \mu|. \tag{4.9}$$

Then, if intervals $\Gamma_i^{(j)}$ $(1 \le i \le \tilde{l} \le j)$ satisfy the following two conditions,

$$\Gamma_i^{(j)} \subset (\alpha, \beta] \quad \text{for } 1 \le i \le \tilde{l}, \tag{4.10}$$

$$\Gamma_{i_1}^{(j)} \cap \Gamma_{i_2}^{(j)} = \varnothing \quad \text{for } 1 \le i_1 < i_2 \le \tilde{l}, \tag{4.11}$$

each $\Gamma_i^{(j)}$ contains only one of $\tilde{l}$ eigenvalues in $(\alpha, \beta]$. In other words, each eigenvalue in $(\alpha, \beta]$ is isolated by interval $\Gamma_i^{(j)}$ and thus its index can be readily validated provided that we know the index range of the eigenvalues in interval $(\alpha, \beta]$.

Algorithm 4.3 computes eigenpairs of an interval by utilizing conditions (4.10) and (4.11) for its convergence test in line 10, along with the relative residual of approximate eigenpairs.

4.2 A three-stage algorithm for singular value problems

Recalling Section 1.1, we know that SVP of general matrix A leads to HEP of the following Hermitian matrices:

$$H_1 \equiv A^{\mathrm{H}}A, \quad H_2 \equiv AA^{\mathrm{H}}, \quad H_3 \equiv \left[\begin{smallmatrix} O & A \\ A^{\mathrm{H}} & O \end{smallmatrix}\right]$$

whose k-th ($1 \le k \le n$) eigenpair is $(\sigma_k^2, \boldsymbol{v}_k)$, $(\sigma_k^2, \boldsymbol{u}_k)$, and $(\sigma_k, \left[\begin{smallmatrix} \boldsymbol{u}_k \\ \boldsymbol{v}_k \end{smallmatrix}\right])$, respectively. Because of this relation between SVP and HEP, the three-stage algorithm (Algorithm 4.1) for HEP can been applied to solution of k-SVP by plugging H_1, H_2, or H_3 into H of the algorithm. Specifically, by selecting H of Algorithms 4.1 to 4.3 as H_1, H_2, or H_3, we may find an interval containing σ_k, narrow down the interval, and compute the singular triplets of the interval.

In Section 4.2.1, we discuss which Hermitian matrix (H_1, H_2, or H_3) to plug into each algorithm of the previous section. Based on the discussion, we show a three-stage algorithm for k-SVP in Section 4.2.2.

4.2.1 Selection of Hermitian matrices

In principle, it is possible to select H_1, H_2, or H_3 as H of Algorithms 4.1 to 4.3, producing 3^3 different algorithms from Algorithm 4.0. Table 4.1 shows the major computation tasks involving Hermitian H in Algorithms 4.1 to 4.3, which we take into account for the selection of the Hermitian matrix for each algorithm.

Table 4.1: Major computational tasks involving matrix H in Algorithms 4.1 to 4.3.

Algorithm	Line	Task
4.1	3	Matrix–vector multiplication $H\boldsymbol{w}_j$ in Lanczos decomposition (4.3)
	6	LDL^{H} factorization (4.1) for computing eigenvalue count $p_\mu(H)$
4.2	3	
4.3	2	LDL^{H} factorization (4.1) for computing $(H-\mu I)^{-1}\tilde{\boldsymbol{w}}_j$ in Lanczos decomposition (4.5)

Algorithm 4.1 utilizes approximate eigenvalues of H by the Lanczos method (Ritz values) as the endpoints of an initial interval. If H_1 is selected as H in line 3 of Algorithm 4.1, approximate eigenvalues are located in interval $[\sigma_n^2, \sigma_1^2]$, with the interlacing property (4.4). If H_2 is selected, approximate eigenvalues are located in $[0, \sigma_1^2]$. Therefore, it is possible to set an initial interval containing σ_k by utilizing the square root of approximate eigenvalues of H_1 or H_2 as the endpoints of the interval. If H_3 is selected, approximate eigenvalues are located in $[-\sigma_1, \sigma_1]$ and can be used for setting an initial interval. Algorithms for setting an initial interval that utilize H_1, H_2, or H_3 will be explained in Section 4.2.2.

Algorithm 4.1 and Algorithm 4.2 set an initial interval and narrow down the interval, respectively, based on $p_\mu(H)$. For non-negative μ, let $\pi_\mu(A)$ be the number singular values of A greater than μ. Then, we have the following relationship.

$$\pi_\mu(A) = p_{\mu^2}(H_1) = p_{\mu^2}(H_2) = p_\mu(H_3). \tag{4.12}$$

Because of the relationship (4.12), it is possible to compute $\pi_\mu(A)$ by computing LDL^{H} factorization of either $H_1 - \mu^2 I, H_2 - \mu^2 I$, or $H_3 - \mu I$ in line 6 of Algorithm 4.1 and line 3 of Algorithm 4.2, and thus it is possible to set an initial interval and narrow down the interval. In this chapter, we compute $\pi_\mu(A)$ by computing LDL^{H} factorization of $H_3 - \mu I$. This is because although the size of

$H_3 - \mu I$ is larger than that of $H_1 - \mu^2 I$ and $H_2 - \mu^2 I$, it inherits the sparsity of A and does not require matrix–matrix multiplication to explicitly form $A^{\mathrm{H}}A$ or AA^{H}.

Algorithm 4.3 computes all eigenvalues in interval $(\alpha, \beta]$ and their associated eigenvectors by a modified SI Lanczos method with the shift $\mu = (\alpha+\beta)/2$. In line 2 of the algorithm, LDL^{H} factorization of $(H-\mu I)^{-1}$ is computed for the computational task involving $(H-\mu I)^{-1}$. If either H_1 or H_2 is selected as H and the shift is set as $\mu = ((\alpha+\beta)/2)^2$, it is possible to compute all singular values in $(\alpha, \beta]$ and either right or left singular vectors associated with the singular values in the interval. If H_3 is selected, it is possible to compute all singular values in $(\alpha, \beta]$ and their associated left and right singular vectors. In this chapter, we compute eigenpairs of the interval by selecting H_3 as H of Algorithm 4.3 because it inherits the sparsity of A.

To summarize this subsection, we select H_3 as H of the second and third tasks in Table 4.1. For H of the first task, H is selected from H_1, H_2, and H_3, producing three out of 3^3 algorithms from Algorithm 4.0.

4.2.2 Algorithms for the k-th singular value problem

Based on the discussion of the previous subsection, we show a three-stage algorithm for k-SVP and four different algorithms for finding an initial interval.

Algorithm 4.4 shows a three-stage algorithm for computing $(\sigma_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$. In the first stage, depending on parameter $i \in \{1,2,3,4\}$, Algorithm 4.1$^{(i)}$ is selected to set an initial interval containing σ_k. In the following two stages, Algorithm 4.2 and Algorithm 4.3 narrow down the interval and compute eigenpairs of the interval, respectively, by selecting H_3 as H in both algorithms. At the end of the third stage, approximate left and right singular vectors $\hat{\boldsymbol{u}}_k$ and $\hat{\boldsymbol{v}}_k$ are obtained from approximate eigenvector $\hat{\boldsymbol{x}}_k$ by Algorithm 4.3.

```
Algorithm 4.4: Three-stage algorithm for solving the $k$-th singular value problem
  Input  : $A \in \mathbb{C}^{m\times n}$, target index $k \in \mathbb{N}$, parameters $i \in \{1,2,3,4\}$, $l \in \mathbb{N}$, and $\tau \in \mathbb{R}$.
  Output: approximate singular triplet $(\hat{\sigma}_k, \hat{\boldsymbol{u}}_k, \hat{\boldsymbol{v}}_k)$.
1 run Algorithm 4.1$^{(i)}$ to set an interval $(\alpha, \beta]$ containing $\sigma_k$,
2 $H := \left[\begin{smallmatrix} O & A \\ A^{\mathrm{H}} & O \end{smallmatrix}\right]$, $p_\alpha(H) := \pi_\alpha(A)$, $p_\beta(H) := \pi_\beta(A)$, run Algorithm 4.2 to narrow down $(\alpha, \beta]$
    until it contains less than or equal to $l$ eigenvalues of $H$,
3 run Algorithm 4.3 to compute $(\lambda_k, \boldsymbol{x}_k)$ of $H$ in $(\alpha, \beta]$ with its relative residual 2-norm less
    than $\tau$, $\hat{\sigma}_k := \hat{\lambda}_k$, $\left[\begin{smallmatrix} \hat{\boldsymbol{u}}_k \\ \hat{\boldsymbol{v}}_k \end{smallmatrix}\right] := \hat{\boldsymbol{x}}_k$ for $\hat{\boldsymbol{u}}_k \in \mathbb{C}^m$ and $\hat{\boldsymbol{v}}_k \in \mathbb{C}^n$, $\hat{\boldsymbol{u}}_k := \hat{\boldsymbol{u}}_k/\|\hat{\boldsymbol{u}}_k\|_2$, $\hat{\boldsymbol{v}}_k := \hat{\boldsymbol{v}}_k/\|\hat{\boldsymbol{v}}_k\|_2$.
```

In the rest of the subsection, we explain Algorithms 4.1$^{(i)}$, $i \in \{1,2,3,4\}$, for setting an initial interval. Algorithms 4.1$^{(1)}$ to 4.1$^{(3)}$ are achieved by selecting H_1, H_2, and H_3 as H in Algorithm 4.1. Initial intervals by the three algorithms are included in $(0, \sigma_1]$ because of the interlacing property of approximate eigenvalues (4.4). Algorithm 4.1$^{(4)}$ sets an initial interval by utilizing a norm of A and a Gershgorin-type theorem[2] applied to H_3. The initial interval by the algorithm does not depend on the value of the target index k and includes $(0, \sigma_1]$.

The essential difference between Algorithm 4.1 and Algorithms 4.1$^{(1)}$ and 4.1$^{(2)}$ is in line 5. In Algorithms 4.1$^{(1)}$ and 4.1$^{(2)}$, Gram matrices $A^{\mathrm{H}}A$ and AA^{H} appear for matrix–vector multiplication in line 3, respectively, and approximate eigenvalues of H_1 and H_2 are computed in line 4, respectively. Because of this, we take the square root of approximate eigenvalues in line 5 and utilize it as the

[2]In contrast to the case of HDGEP (Section 3.1.1), an a priori bound of the spectrum of HEP by Gershgorin-type theorems is guaranteed to be bounded.

endpoints of an initial interval. For computing $\pi_\mu(A)$, LDL^{H} factorization of $H_3 - \mu I$ is utilized.

Algorithm 4.1$^{(1)}$: Setting an interval containing the k-th singular value (Gram matrix $A^{\mathrm{H}}A$)

```
Input  : $A \in \mathbb{C}^{m\times n}$, target index $k \in \mathbb{N}$.
Output: interval $(\alpha, \beta]$ containing $\sigma_k$, $\pi_\alpha(A)$ and $\pi_\beta(A)$.
1  set a random unit vector $\boldsymbol{w}_1 \in \mathbb{C}^n$, $W_1 := [\boldsymbol{w}_1]$, $\pi^{(0)} := 0$, $H := \left[\begin{smallmatrix} O & A \\ A^{\mathrm{H}} & O \end{smallmatrix}\right]$,
                                   ▷ $H$ is utilized for the $LDL^{\mathrm{H}}$ factorization in line 6.
2  for $j = 1, 2, \dots$
3      $\boxed{\text{compute Lanczos decomposition } A^{\mathrm{H}}AW_j = W_jT_j + \eta_j\boldsymbol{w}_{j+1}\boldsymbol{e}_j^{\mathrm{T}},}$ $W_{j+1} := [W_j\ \boldsymbol{w}_{j+1}]$,
4      compute eigenvalues $\theta_i^{(j)}$ $(1 \le i \le j)$ of $T_j$,
5      if $\pi^{(j-1)} \ge k$ then $\boxed{\mu^{(j)} := \sqrt{\theta_1^{(j)}}}$ else $\boxed{\mu^{(j)} := \sqrt{\theta_j^{(j)}},}$
6      $\mu := \mu^{(j)}$, compute $LDL^{\mathrm{H}}$ factorization (4.1),
7      compute singular value count $\pi^{(j)} := p_0(D)$,
8      if $j \ne 1$ and $(\pi^{(j-1)} \ge k > \pi^{(j)}$ or $\pi^{(j)} \ge k > \pi^{(j-1)})$ then exit from the for-loop,
9  end for
10 $\alpha := \min\{\mu^{(j-1)}, \mu^{(j)}\}$, $\beta := \max\{\mu^{(j-1)}, \mu^{(j)}\}$,
11 $\pi_\alpha(A) := \max\{\pi^{(j-1)}, \pi^{(j)}\}$, $\pi_\beta(A) := \min\{\pi^{(j-1)}, \pi^{(j)}\}$.
```

Algorithm 4.1$^{(2)}$: Setting an interval containing the k-th singular value (Gram matrix AA^{H})

```
Input  : $A \in \mathbb{C}^{m\times n}$, target index $k \in \mathbb{N}$.
Output: interval $(\alpha, \beta]$ containing $\sigma_k$, $\pi_\alpha(A)$ and $\pi_\beta(A)$.
1  set a random unit vector $\boldsymbol{w}_1 \in \mathbb{C}^m$, $W_1 := [\boldsymbol{w}_1]$, $\pi^{(0)} := 0$, $H := \left[\begin{smallmatrix} O & A \\ A^{\mathrm{H}} & O \end{smallmatrix}\right]$,
                                   ▷ $H$ is utilized for the $LDL^{\mathrm{H}}$ factorization in line 6.
2  for $j = 1, 2, \dots$
3      $\boxed{\text{compute Lanczos decomposition } AA^{\mathrm{H}}W_j = W_jT_j + \eta_j\boldsymbol{w}_{j+1}\boldsymbol{e}_j^{\mathrm{T}},}$ $W_{j+1} := [W_j\ \boldsymbol{w}_{j+1}]$,
4      compute eigenvalues $\theta_i^{(j)}$ $(1 \le i \le j)$ of $T_j$,
5      if $\pi^{(j-1)} \ge k$ then $\boxed{\mu^{(j)} := \sqrt{\theta_1^{(j)}}}$ else $\boxed{\mu^{(j)} := \sqrt{\theta_j^{(j)}},}$
6      $\mu := \mu^{(j)}$, compute $LDL^{\mathrm{H}}$ factorization (4.1),
7      compute singular value count $\pi^{(j)} := p_0(D)$,
8      if $j \ne 1$ and $(\pi^{(j-1)} \ge k > \pi^{(j)}$ or $\pi^{(j)} \ge k > \pi^{(j-1)})$ then exit from the for-loop,
9  end for
10 $\alpha := \min\{\mu^{(j-1)}, \mu^{(j)}\}$, $\beta := \max\{\mu^{(j-1)}, \mu^{(j)}\}$,
11 $\pi_\alpha(A) := \max\{\pi^{(j-1)}, \pi^{(j)}\}$, $\pi_\beta(A) := \min\{\pi^{(j-1)}, \pi^{(j)}\}$.
```

The essential difference of Algorithm 4.1 and Algorithm 4.1$^{(3)}$ is in lines 8 and 11 of Algorithm 4.1$^{(3)}$. In Algorithm 4.1$^{(3)}$, the extended matrix $\left[\begin{smallmatrix} O & A \\ A^{\mathrm{H}} & O \end{smallmatrix}\right]$ appears in line 3, and approximate eigenvalue of H_3 are computed in 4. Approximate eigenvalues of H_3 are located in $[-\sigma_1, \sigma_1]$ and can be negative. Because of this, if approximate eigenvalues are negative, the sign of the approximate eigenvalues are flipped (and the smallest and largest approximate eigenvalues are interchanged) to make the endpoints of an initial interval non-negative. In addition, if an initial interval happens to contain the origin, the left endpoint of the interval is set as zero in line 11. For computing $\pi_\mu(A)$, as in Algorithms 4.1$^{(1)}$ and 4.1$^{(2)}$, LDL^{H} factorization of $H_3 - \mu I$ is utilized.

```
Algorithm 4.1^(3): Setting an interval containing the k-th singular value (Extended matrix)
  Input  : $A \in \mathbb{C}^{m\times n}$, target index $k \in \mathbb{N}$.
  Output: interval $(\alpha, \beta]$ containing $\sigma_k$, $\pi_\alpha(A)$ and $\pi_\beta(A)$.
1 set a random unit vector $\boldsymbol{w}_1 \in \mathbb{C}^{m+n}$, $\pi^{(0)} := 0$, $H := \left[\begin{smallmatrix} O & A \\ A^{\mathrm{H}} & O \end{smallmatrix}\right]$,
2 for $j = 1, 2, \ldots$
3     $\boxed{\text{compute Lanczos decomposition (4.3),}}$
4     compute eigenvalues $\theta_i^{(j)}$ $(1 \le i \le j)$ of $T_j$,
5     if $\theta_1^{(1)} \ge 0$ then
6         if $\pi^{(j-1)} \ge k$ then $\mu^{(j)} := \theta_1^{(j)}$ else $\mu^{(j)} := \theta_j^{(j)}$,
7     else
8         $\boxed{\textbf{if } \pi^{(j-1)} \ge k \textbf{ then } \mu^{(j)} := -\theta_j^{(j)} \textbf{ else } \mu^{(j)} := -\theta_1^{(j)},}$
9     end if
10    if $\mu^{(j)} < 0$ then
11        $\boxed{\mu^{(j)} := 0,\ \pi^{(j)} := n,}$
12    else
13        $\mu := \mu^{(j)}$, compute $LDL^{\mathrm{H}}$ factorization (4.1),
14        compute singular value count $\pi^{(j)} := p_0(D)$,
15    end if
16    if $j \ne 1$ and $(\pi^{(j-1)} \ge k > \pi^{(j)}$ or $\pi^{(j)} \ge k > \pi^{(j-1)})$ then exit from the for-loop,
17 end for
18 $\alpha := \min\{\mu^{(j-1)}, \mu^{(j)}\}$, $\beta := \max\{\mu^{(j-1)}, \mu^{(j)}\}$,
19 $\pi_\alpha(A) := \max\{\pi^{(j-1)}, \pi^{(j)}\}$, $\pi_\beta(A) := \min\{\pi^{(j-1)}, \pi^{(j)}\}$.
```

In Algorithm 4.1$^{(4)}$, the left endpoint of an initial interval is fixed to zero, and the right endpoint is set by utilizing upper bounds of σ_1. Therefore, both endpoints are independent from the value of the target index k.

Let nnz(A) be the number of non-zero elements of A. Algorithm 4.1$^{(4)}$ utilizes upper bounds of σ_1 that can be computed in $O(\mathrm{nnz}(A))$ FLOPs. Specifically, either Frobenius norm $\|A\|_{\mathrm{F}}$ or an upper bound of the largest eigenvalue of H_3 by a Gershgorin-type theorem [53],

$$\lambda_{\mathrm{N}} = \max\left\{ \max_{1\le i\le m} \sqrt{\sum_{1\le j\le n} |A_{ij}| c_j(A)},\ \max_{1\le j\le n} \sqrt{\sum_{1\le i\le m} |A_{ij}| r_i(A)} \right\}, \tag{4.13}$$

is utilized. Here, as Definition 2.3, column sums $c_j(A) = \sum_{1\le i\le m} |A_{ij}|$, and row sums $r_i(A) = \sum_{1\le j\le n} |A_{ij}|$. The equation (4.13) is derived in Appendix A. Because $\|A\|_{\mathrm{F}}$ can be a sharper upper bound of σ_1 than λ_{N}, and vice versa, both $\|A\|_{\mathrm{F}}$ and λ_{N} are computed in line 1, and the sharper one is selected as the right endpoint of an initial interval in line 2. Note that Appendix A also discusses the sharpness of some upper bounds of σ_1 that can be computed in $O(\mathrm{nnz}(A))$ FLOPs.

```
Algorithm 4.1^(4): Setting an interval containing all non-zero singular values
  Input  : $A \in \mathbb{C}^{m\times n}$.
  Output: interval $(\alpha, \beta]$ containing $\sigma_i \ne 0$ $(1 \le i \le n)$, $\pi_\alpha(A)$ and $\pi_\beta(A)$.
1 compute Frobenius norm $\|A\|_{\mathrm{F}}$, compute Nakatsukasa bound (4.13),
2 $\alpha := 0$, $\beta := \min\{\|A\|_{\mathrm{F}}, \lambda_{\mathrm{N}}\}$, $\pi_\alpha(A) := n$, $\pi_\beta(A) := 0$.
```

4.3 Numerical experiments

This section reports numerical results of Algorithm 4.4 and examines its accuracy and efficiency for computing $(\sigma_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$. In Section 4.3.1, to examine the accuracy of Algorithm 4.4, we compare the result of Algorithm 4.4 with that of SVD for matrices of size $m+n \leq 30\,000$. In Section 4.3.2, to examine the efficiency of Algorithm 4.4 for large-scale problems, we compute $(\sigma_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$ of matrices of size $m+n > 30\,000$.

Test matrices were achieved from The University of Florida Sparse Matrix Collection [12]. Codes were written in Fortran and compiled by GNU Fortran compiler 4.8.2. Numerical experiments were performed on Intel Xeon E5-2690 (2.90 GHz, using its one core) and memory 128 GB in double precision. The stopping criterion for the second stage was set to $l = 20$, and the convergence criterion for the third stage was $\tau = 10^{-10}$ for Algorithm 4.4. LDL^{H} factorization (4.1) was computed by using MUMPS 5.0.1 [2,3] with fill-reducing ordering by METIS 5.1.0 [38]. For computing SVD, the `dgesvd` routine of Netlib's LAPACK 3.5.0 [4] was utilized. The `dgesvd` routine computes SVD by reducing a matrix into a bidiagonal matrix and applying QR method [14] to the bidiagonal matrix. For BLAS, ATLAS 3.10.1 [64] was utilized.

4.3.1 Experiment 1

This section examines the accuracy of Algorithm 4.4. Real non-symmetric matrices of size $m+n \leq 30\,000$ were used as test matrices. The target index is set to $k = \lfloor(\min\{m,n\}+1)/2\rfloor$. The k-th singular triplet was computed by Algorithm 4.4 and the LAPACK routine `dgesvd`.

Table 4.2 shows the test matrices. Figure 4.1 shows the comparison of the k-th singular triplet by Algorithm 4.4 with that by SVD. The horizontal axis of Figure 4.1 shows No. of the test matrices in Table 4.2. From Figure 4.1, the relative errors of singular values were about 10^{-15}, and the error 2-norms of singular vectors were around 10^{-10}. For each test matrix, the error 2-norms for left and right singular vectors are almost the same.

Table 4.2: Test matrices and target indices. nnz denotes the number of non-zero elements.

No.	Name	m	n	nnz	k
(1)	dw2048	2048	2048	10 114	1024
(2)	lp_bnl2	2324	4486	14 996	1162
(3)	pde2961	2961	2961	14 585	1481
(4)	deter4	3235	9133	19 231	1618
(5)	ch	3700	8291	24 102	1850
(6)	large	4282	8617	20 635	2141
(7)	Maragal_5	4654	3320	93 091	1660
(8)	gemat1	4929	10 595	46 591	2465
(9)	add32	4960	4960	19 848	2480
(10)	ge	10 099	16 369	44 825	5050

For each test matrix, the difference in setting an initial interval (Algorithm 4.1$^{(i)}$, $i \in \{1,2,3,4\}$) results in moderate difference of the relative error and error 2-norm. The largest difference of the relative error was 4.4×10^{-1} for the test matrix No. (7), and the largest difference of the error 2-norm was 2.4×10^{-2} for the test matrix No. (3).

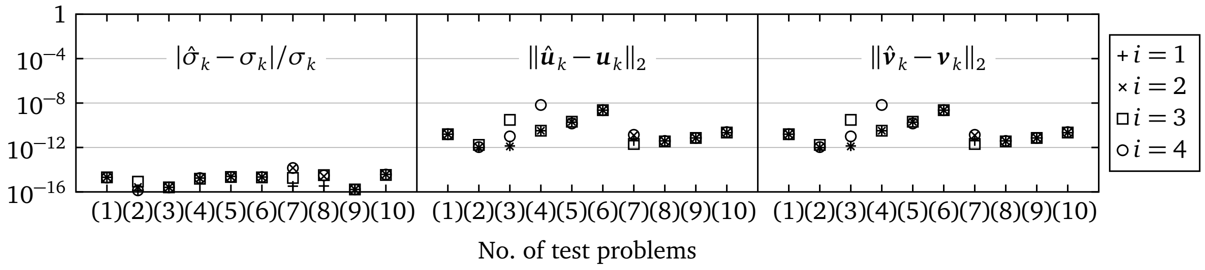


Figure 4.1: Relative error $|\hat{\sigma}_k - \sigma_k|/\sigma_k$ of singular values and error 2-norms $\|\hat{\boldsymbol{u}}_k - \boldsymbol{u}_k\|_2$ and $\|\hat{\boldsymbol{v}}_k - \boldsymbol{v}_k\|_2$ of singular vectors. $(\hat{\sigma}_k, \hat{\boldsymbol{u}}_k, \hat{\boldsymbol{v}}_k)$ and $(\sigma_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$ denote the k-th singular triplet computed by Algorithm 4.4 and the LAPACK `dgesvd` routine, respectively. The parameter $i \in \{1, 2, 3, 4\}$ indicates that Algorithm 4.1$^{(i)}$ was selected for setting an initial interval.

4.3.2 Experiment 2

This sections examines the efficiency of Algorithm 4.4 for large-scale matrices. Real non-symmetric matrices of size $m+n > 30\,000$ were used as test matrices. The target index is set to $k = \lfloor(\min\{m, n\}+1)/2\rfloor$.

Table 4.3 shows test matrices. For the test matrices No. (11), (12), and (13), computation of SVD by the LAPACK `dgesvd` routine was not converged (INFO > 0). For the test matrices No. (14) and (15), computation of SVD by the `dgesvd` routine was impossible because of insufficient memory[3].

Table 4.4 shows computational results of Algorithm 4.4. For each test matrix, the k-th singular values $\hat{\sigma}_k$ computed by Algorithm 4.4 with different algorithms (Algorithm 4.1$^{(i)}$, $i \in \{1, 2, 3, 4\}$) for setting an initial interval were the same at least 14 digits[3]. The residual 2-norms $\|\hat{\boldsymbol{r}}_k\|_2$ were around 10^{-11}. Peak memory consumption for test matrices (14) and (15) was 2.0 GB and 4.1 GB, respectively.

In the fifth column of Table 4.4 is computation time of Algorithm 4.4, with its detail provided in Figure 4.2. Overall, computation time for different $i \in \{1, 2, 3, 4\}$ was similar to one another. When compared to computation time of $i = 4$ (computation time of $i = 4$ is scaled to one in Figure 4.2), difference in computation time was about 10% for each test matrix, and the largest difference was 32% in the case of the test matrix (14). As a whole, computation time of $i = 1$ was relatively short, while that of $i = 4$ was relatively long. In addition, difference in computation time of $i = 1$ and $i = 2$ was notable especially for the case of non-square test matrices (13), (14), and (15).

In the sixth column of Table 4.4 were the numbers of iterations of each algorithm consisting of Algorithm 4.4. In general, for each test matrix, the larger the iteration number of Algorithm 4.1$^{(i)}$ is, the smaller the iteration number of Algorithm 4.2 is. As a whole, on the one hand, the iteration number of Algorithm 4.1$^{(i)}$ was relatively small for $i = 3$, except the case of $i = 4$ in which the initial interval was set regardless of the value of k. On the other hand, the iteration number of Algorithm 4.2 was relatively small for $i = 1$ and large for $i = 4$. In addition, for $i = 1$ and $i = 2$, the iteration number of both Algorithms 4.1$^{(i)}$ and 4.2 were identical for the case of square test matrices (11) and (12). For Algorithm 4.3, in general, it took about 40 iterations, accounting for about 10% of computation time as shown in Figure 4.2.

[3]The LAPACK `dgesdd` routine (divide and conquer method [22] for SVD) was able to compute SVD of test matrices No. (11) and (12). Computed k-th singular values were 4.415844009694552E+01 and 5.589031396650464E–02, respectively, and were the same as those by Algorithm 4.4 at least 14 digits. For test matrices No. (13), (14), and (15), computation of SVD by the `dgesdd` routine was impossible because of insufficient memory.

Table 4.3: Test matrices and target indices. nnz denotes the number of non-zero elements.

No.	Name	m	n	nnz	k
(11)	af23560	23 560	23 560	460 598	11 780
(12)	wang3	26 064	26 064	177 168	13 032
(13)	tomographic1	73 159	59 498	647 495	29 749
(14)	degme	185 501	659 415	8 127 528	92 751
(15)	Rucci1	1 977 885	109 900	7 791 168	54 950

Table 4.4: Numerical results of Algorithm 4.4. The parameter $i \in \{1,2,3,4\}$ indicates that Algorithm 4.1$^{(i)}$ was selected for setting an initial interval. $(\hat{\sigma}_k, \hat{\boldsymbol{u}}_k, \hat{\boldsymbol{v}}_k)$ is the computed value of the k-th singular triplet. $\|\hat{\boldsymbol{r}}_k\|_2$ denotes the residual 2-norm with $\hat{\boldsymbol{r}}_k = \frac{1}{\sqrt{2}}\begin{bmatrix} A\hat{\boldsymbol{v}}_k - \hat{\sigma}_k\hat{\boldsymbol{u}}_k \\ A^{\mathrm{H}}\hat{\boldsymbol{u}}_k - \hat{\sigma}_k\hat{\boldsymbol{v}}_k \end{bmatrix}$.

No.	i	$\hat{\sigma}_k$	$\|\hat{\boldsymbol{r}}_k\|_2$	Time (s)	Iterations of Algorithm		
					$2^{(i)}$	3	4
(11)	1	4.415844009694546E+01	1E−11	45.2	7	6	47
	2	4.415844009694555E+01	2E−11	45.2	7	6	46
	3	4.415844009694559E+01	2E−11	44.6	2	11	35
	4	4.415844009694549E+01	1E−11	49.7	1	15	36
(12)	1	5.589031396650460E−02	4E−14	48.8	3	8	35
	2	5.589031396650459E−02	1E−13	49.4	3	8	44
	3	5.589031396650448E−02	5E−11	56.9	2	11	25
	4	5.589031396650465E−02	1E−11	56.0	1	13	34
(13)	1	3.409163952571004E−01	1E−12	133.9	8	6	40
	2	3.409163952571005E−01	2E−12	128.8	7	6	48
	3	3.409163952571011E−01	1E−12	128.1	2	11	52
	4	3.409163952571015E−01	8E−13	140.4	1	15	31
(14)	1	1.706200039955296E+01	7E−13	642.6	3	12	40
	2	1.706200039955296E+01	1E−12	938.8	2	21	42
	3	1.706200039955296E+01	2E−12	680.8	2	14	38
	4	1.706200039955296E+01	1E−12	926.7	1	23	31
(15)	1	2.065089627276347E+00	1E−12	1529.2	2	11	31
	2	2.065089627276345E+00	1E−12	1774.2	2	13	44
	3	2.065089627276347E+00	2E−12	1829.8	3	13	32
	4	2.065089627276348E+00	2E−12	1766.5	1	15	42

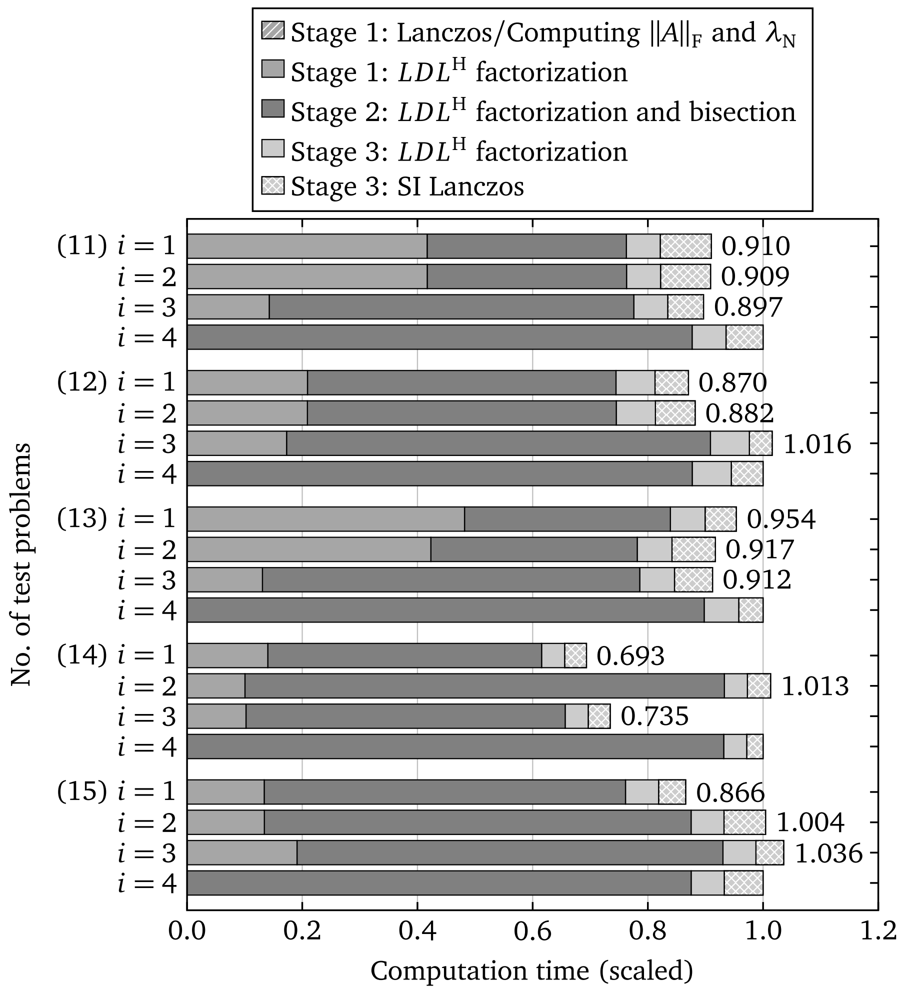


Figure 4.2: Computation time of stages and computational tasks

4.4 Concluding remarks

In this chapter, based on relation between HEP and SVP (Section 1.1), we presented a three-stage algorithm for computing the k-th singular triplet of large sparse matrices.

In Experiment 1, we showed that Algorithm 4.4 can compute the k-th singular triplet with accuracy comparable to the results of SVD. In Experiment 2, Algorithm 4.4 was able to compute the k-th singular triplet of a large-scale matrix. For example, for a two million by one hundred thousand matrix, the k-th singular triplet was computed by Algorithm 4.4 using 4.1 GB of memory, while SVD could not be executed because of an estimated several TB of memory requirement.

Among at least 36 algorithms that can be derived from Algorithm 4.4, we have explored 4 algorithms in this chapter. Other possible algorithms will be investigated in future.

Chapter 5

Relation to data circulation

This chapter discusses contributions of the thesis in the context of RWDC. In particular, contributions to data circulation in computational and data-driven studies of materials are discussed.

First-principles simulations of materials are based on quantum mechanics and enable studies of materials and their properties at the nanoscopic level that are expensive or even impossible to be conducted by theoretical and experimental approaches. As illustrated in Figure 5.1, fundamental to the simulations are matrix eigenvalue problems and their numerical solution. Several properties of materials, such as the electrical conductivity of flexible device materials [31], are analysed by using eigenvalues and eigenvectors of the problems. In addition, as will be explained in Section 5.1, eigenvectors are processed to be a *descriptor*, or a unique identifier of a material, for data-driven studies of materials [32, 34].

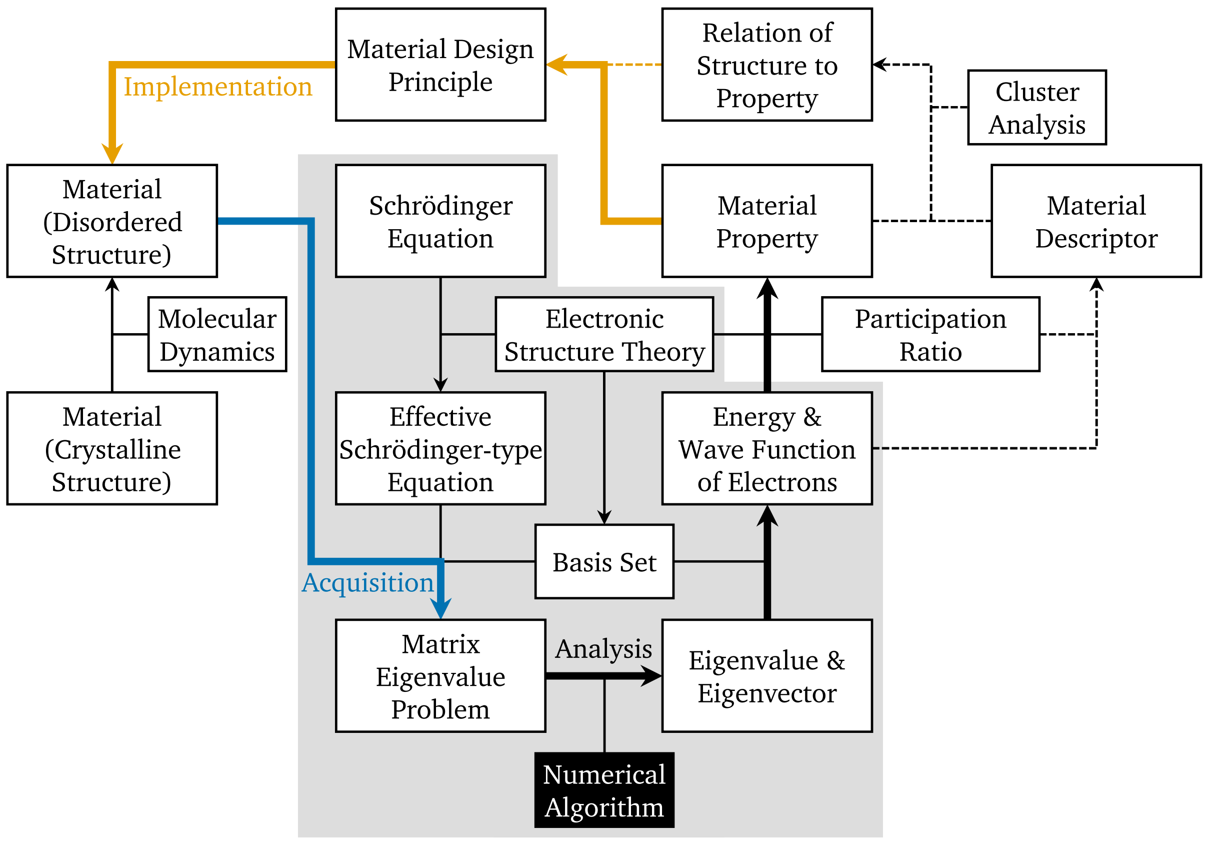

Material Design Principle
Relation of Structure to Property
Cluster Analysis
Implementation
Material (Disordered Structure)
Schrödinger Equation
Material Property
Material Descriptor
Molecular Dynamics
Electronic Structure Theory
Participation Ratio
Material (Crystalline Structure)
Effective Schrödinger-type Equation
Energy & Wave Function of Electrons
Basis Set
Acquisition
Matrix Eigenvalue Problem
Analysis
Eigenvalue & Eigenvector
Numerical Algorithm

Figure 5.1: Data circulation in computational (thick lines) and data-driven (dashed lines) studies of materials based on first-principles simulations (shaded in gray). Numerical algorithms for eigenvalue problems, which are the focus of this thesis, are fundamental to the simulations and circulation.

This thesis proposes a numerical algorithm for solving eigenvalue problems. The proposed algorithm is able to compute eigenpairs of industrial importance in materials simulations with efficiency and to cope with more large-scale eigenvalue problems than currently avaiable algorithms, which can lead to the acceleration and scaling-up of computational and data-driven studies of materials. Further discussion on contributions of the thesis to the studies of materials is provided in Section 5.2.

5.1 Data circulation

This section explains data circulation in computational and data-driven studies of materials that are related to this thesis. In particular, studies of flexible device materials are considered based largely on the recent literature [31] and references therein.

This section is divided into three parts. The first part provides background on eigenvalue problems in electronic structure calculations of materials. The second part illustrates the analysis of material properties by using *participation ratio* (PR), which is calculated from eigenvectors. The third part describes data-driven studies of materials in which PR is utilized as a material descriptor to conduct cluster analysis of organic materials.

Eigenvalue problems in electronic structure calculations

Practical electronic structure calculations use effective independent-electron theories, such as density functional theory [26, 40, 41], that are formulated with an effective Schrödinger-type equation for electronic wave functions $\{\psi_i(\boldsymbol{r})\}_{i\geq 1}$,

$$\hat{H}_{\text{eff}}\psi_i(\boldsymbol{r}) = \varepsilon_i\psi_i(\boldsymbol{r}) \tag{5.1}$$

with the Hamiltonian operator $\hat{H}_{\text{eff}} \equiv -\frac{\hbar^2}{2m_{\text{e}}}\Delta + V_{\text{eff}}(\boldsymbol{r})$. Here, m_{e} is the electron mass, and $\hbar$ denotes the Planck constant. The scalar function $V_{\text{eff}}(\boldsymbol{r})$ is the potential function for electrons at coordinate $\boldsymbol{r}$ and varies among materials. The solution of (5.1) gives the eigenpairs $(\varepsilon_i, \psi_i(\boldsymbol{r}))$. The eigenvalues ε_i are the energy of electron and are real. The eigenvalues are indexed in increasing order in this chapter. The square $|\psi_i(\boldsymbol{r})|^2$ of a wave function is the probability density of an electron, and the wave functions are normalized with respect to the L^2-norm, $\int |\psi_i(\boldsymbol{r})|^2\, d\boldsymbol{r} = 1$.

Independent-electron systems can be discretized by the Galerkin method and be formulated as a matrix eigenvalue problem. When an electronic wave function $\psi_i(\boldsymbol{r})$ is expanded (more accurately, approximated) by the linear combination of n non-orthogonal basis functions $\{\chi_j(\boldsymbol{r})\}_{j=1}^n$,

$$\psi_i(\boldsymbol{r}) = \sum_{j=1}^{n} x_j^{(i)}\chi_j(\boldsymbol{r}), \tag{5.2}$$

a generalized eigenvalue problem,

$$A\boldsymbol{x}_i = \lambda_i B\boldsymbol{x}_i,$$

appears with the $n \times n$ Hermitian matrices A and B whose i, j elements are defined as follows:

$$A_{ij} \equiv \int \bar{\chi}_i(\boldsymbol{r})\hat{H}_{\text{eff}}\chi_j(\boldsymbol{r})d\boldsymbol{r}, \quad B_{ij} \equiv \int \bar{\chi}_i(\boldsymbol{r})\chi_j(\boldsymbol{r})d\boldsymbol{r}. \tag{5.3}$$

Here, $\bar{\chi}$ denotes the complex conjugate of function χ. Matrix B is positive definite from the definition, and its diagonal elements all equal one provided that the basis functions are normalized with respect to the L^2-norm. Matrices A and B are referred to as Hamiltonian and overlap matrices, respectively.

Eigenvalues λ_i correspond to the energy ε_i of an electron, and eigenvectors $\boldsymbol{x}_i = [x_1^{(i)} \cdots x_n^{(i)}]^{\mathrm{T}}$ represent an electronic wave function $\psi_i(\boldsymbol{r})$. From the normalization condition of $\psi_i(\boldsymbol{r})$, eigenvectors are normalized with respect to the B-norm, $\sqrt{\boldsymbol{x}_i^{\mathrm{H}} B \boldsymbol{x}_i} = 1$.

The size and structure of matrices A, B depend on the construction of the Hamiltonian operator $\hat{H}_{\mathrm{eff}}$ and the choice of the basis set $\{\chi_j(\boldsymbol{r})\}$. This chapter considers a first-principle-based modeled (transferable tight-binding) theory [33] in which basis functions, referred to as atomic orbitals, are localized in real space with their localization center being the position of an atom in a material. The index j of the basis functions χ_j can then be expressed as a composite of two indices l and m that represent a localization center, or an atom, and the shape of an orbital, respectively. Using indices l and m, an alternative expression for the expansion of wave functions (5.2) is obtained as follows: $\psi_i(\boldsymbol{r}) = \sum_{l=1}^{n_{\mathrm{atom}}} \sum_{m=1}^{n_l} x_{lm}^{(i)} \chi_{lm}(\boldsymbol{r})$. Here, n_{atom} denotes the number of atoms in a material. n_l is the number of orbitals centered at the l-th atom and differs depending on the given atomic species. For example, one s-type orbital is prepared for each hydrogen atom, while one s-type and three p-type orbitals are utilized for a carbon atom. Therefore, the matrix size n is roughly proportional to the number n_{atom} of atoms in a material. Since orbitals are localized in real space, the matrix elements (5.3) decay rapidly as the distance between the localization centers of the orbitals increases. Thus, the matrices become sparse. Further details about the physical origin of the problem can be found in the literature [30].

Analysis of flexible device materials

Organic semiconductor materials form the foundation for flexible devices such as flexible displays and wearable electronics. The organic materials have large degrees of freedom in their physical structure because they consist of molecules that form a solid based on weak physical interactions [31]. In other words, the structure of the materials is disordered in general.

The structural disorder affects the spatial extension of a quantum mechanical wave of an electron in real space, which governs electrical and optical properties of the materials and is important for the device performance. Specifically, electronic wave functions extended over a wide region of the space (or a large number of molecules) are prefereable for the electrical conductivity, while those localized in a narrow region of the space (or a small number of molecules) are prefereable for optical properties [31]. Investigating the relation of the structure disorder to material properties by the spatial extension of wave functions is crucial especially for the design and production of optoelectronic devices that need to be electroconductive and to interact with light at the same time.

The spatial extension of wave function ψ_i is measured by *molecular PR* [31] that is calculated from the associated eigenvector $\boldsymbol{x}_i$ as follows: $\mathrm{PR}^{(\mathrm{mol})}(\boldsymbol{x}_i) \equiv \left(\sum_{j=1}^{n_{\mathrm{mol}}} \left| \sum_{l \in \mathrm{M}_j} \left(\sum_{m=1}^{n} x_l^{(i)} B_{lm} x_m^{(i)} \right) \right|^2 \right)^{-1}$. Here, n_{mol} denotes the number of molecules in a material, and M_j is a set of indices associated with the orbitals belonging to the j-th molecule in a material. A wave function with $\mathrm{PR}^{(\mathrm{mol})} = p$ can be considered to be extended over (or localized in) roughly p molecules in a material.

Large-scale simulations are required to invesitage the spatial extension of wave functions in device materials with a complicated disordered structure. Specifically, in order to investigate a wave function extended over a few dozens of molecules, or $\mathrm{PR}^{(\mathrm{mol})} = O(10)$, it is necessary to simulate $n_{\mathrm{mol}} \gg O(10)$ molecules. A recent study [31] conducted simulations of a thin organic film consisting of $n_{\mathrm{mol}} = 1800$ pentacene molecules ($n_{\mathrm{atom}} = 64\,800$, $n = 183\,600$), which is considered to be the semiconducting layer in an organic field-effect transistor (OFET). A wave function with $\mathrm{PR}^{(\mathrm{mol})} \approx 35$ was observed in the simulations, which is in accordance with an experimental observation [51] of wave functions with $\mathrm{PR}^{(\mathrm{mol})} = O(10)$.

Data-driven studies of flexible device materials

Sample materials with various disordered structures are required to be analysed in order to investigate the device performance. Many eigenvalue problems have been solved to conduct data-driven studies [31, 32, 34] of poly(phenylene-ethynylene) (PPE), a typical electroconductive polymer, with disordered structures. Specifically, for the j-th sample polymer with $j \in \{1, \dots, n_{\mathrm{sample}}\}$, a generalized eigenvalue problem of size n,

$$A^{(j)}\boldsymbol{x}_i^{(j)} = \lambda_i^{(j)} B^{(j)} \boldsymbol{x}_i^{(j)},$$

is solved to compute all its eigenpairs. Each n-dimensional eigenvector $\boldsymbol{x}_i^{(j)}$ is processed into PR, which is calculated as $\mathrm{PR}(\boldsymbol{x}_i^{(j)}) \equiv (\|\boldsymbol{x}_i^{(j)}\|_4^4)^{-1}$. Based on PR calculated from eigenvectors, the j-th sample polymer is represented by n-dimensional vector $\boldsymbol{d}^{(j)} \equiv [\mathrm{PR}(\boldsymbol{x}_1^{(j)}) \cdots \mathrm{PR}(\boldsymbol{x}_n^{(j)})]^{\mathrm{T}}$, which is utilized as a descriptor for data-driven studies.

The set $\{\boldsymbol{d}^{(j)}\}_{j=1,\dots,n_{\mathrm{sample}}}$ of material descriptors can be analysed by common methods for machine learning and data mining. A recent study [34] solved $n_{\mathrm{sample}} = 200$ eigenvalue problems of PPE. Each sample polymer consists of 20 monomers ($n_{\mathrm{atom}} = 240$, $n = 712$) and has a rotational degree of freedom between adjacent monomers. Based on the k-means clustering, the sample materials were successfully clustered into two disordered structures, polymers with a small or large dihedral rotation angle. Another recent study [32] solved $n_{\mathrm{sample}} = 40\,000$ eigenvalue problems of PPE consisting of 100 monomers ($n_{\mathrm{atom}} = 1200$, $n = 3594$). In addition to the rotational degree of freedom, adjacent monomers of the sample polymers are now junctioned in two different ways, in a linear or zigzag manner (para-junctioned or meta-junctioned [61]). The sample polymers of the study were clustered into four different disordered structures based on the principal component analysis. Specifically, the difference in the rotation angle and the way of junction in the monomers of the sample polymers was able to be captured by the first and second principal components, and the material descriptors $\boldsymbol{d}^{(j)}$ projected to the linear span of the first and second principal components were separated into four clusters of different rotation angles and ways of junction.

5.2 Contributions to data circulation

This section discusses how the proposed algorithm contributes to the data circulation in the computational and data-driven studies of materials (Section 5.1).

Various material properties are governed by a small number of eigenpairs near the eigenpair associated with the highest occupied (HO) state [50, Chapter 2], denoted by $(\lambda_k, \boldsymbol{x}_k)$ with a material-specific index k. The value of k is determined by the number of electrons n_{electron} in a material. Typically, in a para-spin material, the index k is defined as one-half the number of electrons, or $k \equiv \lceil n_{\mathrm{electron}}/2 \rceil$, because each wave function ψ_i with $i = 1, 2, \dots, k$ is occupied by two electrons in the ground (most stable) state. The next index $(k+1)$ is for the lowest unoccupied state. The difference between the k-th and $(k+1)$-st eigenvalues, or $\lambda_{k+1} - \lambda_k$, corresponds to the energy gap, which is crucial for electrical properties because the value is zero in metallic materials and non-zero in semiconducting or insulating materials.

This thesis proposes a purpose-specific algorithm for computing only eigenpairs of practical interest for large-scale electronic structure calculations. In particular, in contrast to currently available algorithms, the proposed algorithm is able to compute the material-specific k-th eigenpair in a selective manner without requiring computation of all or a substantial portion of eigenpairs, realizing efficient and large-scale simulations of materials. For the study of a pentacene thin film ($n_{\mathrm{atom}} = 64\,800$, $n = 183\,600$) in Section 5.1, for example, only a small number of eigenvectors whose associated eigenvalue is close to and smaller than or equal to λ_k, $k = 91\,800$ are necessary [31]. A few dozens of necessary eigenpairs of the film have been able to be computed by the proposed algorithm in about 0.2 hours using 1 central processing unit [45], while all eigenpairs were computed by a currently available algorithm in about 6 hours using 36 computational nodes [31]. In addition, the proposed algorithm has been able to solve an eigenvalue problem of $n_{\mathrm{mol}} = 20\,000$ pentacene molecules ($n_{\mathrm{atom}} = 720\,000,\ n = 2\,040\,000$) [45], which is beyond the pracitial size limit ($n \approx 10^6$) of the currently available algorithms.

The proposed algorithm can contribute to the acceleration and scaling-up of computational and data-driven studies of materials. With the proposed algorithm, it is expected that more large-scale simulations can be conducted to investigate complicated device materials and their properties, such as the interface system of the semiconducting and insulating layers in OFET [31]. Also expected is data-driven studies of polymer systems in which solution of more large-scale and larger number of eigenvalue problems is considered necessary, compared to the studies of a single polymer in Section 5.1.

Chapter 6

Conclusion

6.1 Summary

This thesis proposed a three-stage algorithm for computing the k-th eigenpair of a large sparse HDGEP. Numerical experiments showed that the proposed algorithm is able to solve real-world problems with efficiency and to cope with problems whose size is beyond the practical limit of current available dense eigensolvers. Singular value problems can be considered to be a special case of HDGEP, and the proposed algorithm was applied to computation of the k-th singular triplet of a large sparse matrix. A possible role of the proposed algorithm in studies of materials was discussed in the context of Real-World Data Circulation.

6.2 Future work

- A first stage algorithm for finding an initial interval
  The proposed Lanczos-based algorithm was able to find a narrow interval containing the k-th eigenvalue in two iterations for k-EP from real-world applications. When applied to k-SVP, the proposed algorithm required up to eight iterations to find an interval containing the k-th singular value, which were almost comparable to and sometimes more than the number of iterations required for the bisection algorithm in the second stage. As a remedy for a possible case in which the proposed algorithm requires many iterations for k-EP, a first stage algorithm that can find an initial interval at a fixed and moderate computational cost is necessary.

- Application of the three-stage algorithm
  Performance evaluation on various and extensive real-world problems is necessary to clarify capabilities and limitations of the proposed three-stage algorithm for its future improvement. Ongoing motivation for the study of k-EP is electronic structure calculations of materials. It is also important to discover k-EP and k-SVP from other areas of computational science and engineering in order to broaden application areas of the proposed algorithm.

Appendix A

Upper bounds of the largest singular value

This chapter compares the sharpness of several upper bounds of the largest singular value σ_1 of $A \in \mathbb{C}^{m\times n}$. In particular, two types of upper bounds are considered. The first type includes matrix norms $\|A\|_1$, $\|A\|_\infty$, and $\|A\|_\mathrm{F}$. The second type includes upper bounds $\lambda_\mathrm{G}, \lambda_\mathrm{O}, \lambda_\mathrm{B}$, and λ_N of the largest eigenvalue λ_1 of the extended matrix $H \equiv \left[\begin{smallmatrix} O & A \\ A^\mathrm{H} & O \end{smallmatrix}\right]$, which are computed from Gershgorin-type theorems by Gershgorin, Ostrowski, Brauer, and Nakatsukasa, respectively.

The following is the main result on the sharpness, which will be proved in the rest of the chapter:

$$\|A\|_2 = \sigma_1 = \lambda_1 \le \lambda_\mathrm{N} \le \sqrt{\|A\|_1 \cdot \|A\|_\infty} \le \lambda_\mathrm{B} \le \max\{\|A\|_1, \|A\|_\infty\} = \lambda_\mathrm{O} = \lambda_\mathrm{G}. \tag{A.1}$$

Also proved is that no inequality holds in general between $\|A\|_\mathrm{F}$ and λ_N; Depending on matrices, $\|A\|_\mathrm{F}$ can be a sharper upper bound of σ_1 than λ_N, and vice versa.

As Definition 2.3, let $r_i(A)$ ($r'_i(A)$) be the (deleted) row sums and $c_j(A)$ ($c'_j(A)$) be the (deleted) column sums. The following theorems are extensions of Theorem 2.4 by Gerghgorin.

Theorem A.1 (Ostrowski [27, Theorem 6.4.1]). *For $\alpha \in [0,1]$, all eigenvalues of $A \in \mathbb{C}^{n\times n}$ are included in*

$$O(A) \equiv \bigcup_{1\le i\le n} O_i(A), \quad O_i(A) \equiv \left\{ z \in \mathbb{C} \;\middle|\; |z - A_{ii}| \le r'_i(A)^\alpha \cdot c'_i(A)^{1-\alpha} \right\}.$$

Theorem A.2 (Brauer [27, Theorem 6.4.7]). *All eigenvalues of $A \in \mathbb{C}^{n\times n}$ ($n \ge 2$) are included in*

$$B(A) \equiv \bigcup_{1\le i<j\le n} B_{ij}(A), \quad B_{ij}(A) \equiv \left\{ z \in \mathbb{C} \;\middle|\; |z - A_{ii}| \cdot |z - A_{jj}| \le r'_i(A) \cdot r'_j(A) \right\}.$$

Theorem A.3 (Nakatsukasa [53, Theorem 11.1]). *All eigenvalues of $A \in \mathbb{C}^{n\times n}$ are included in*

$$N(A) \equiv \bigcup_{1\le i\le n} N_i(A), \quad N_i(A) \equiv \left\{ z \in \mathbb{C} \setminus D(A) \;\middle|\; |z - A_{ii}| \le \sum_{\substack{1\le j\le n,\\ j\ne i}} |A_{ij}| \frac{r'_j(A)}{|z - A_{jj}|} \right\} \cup D(A),$$

where $D(A) \equiv \{A_{11}, \ldots, A_{nn}\}$.

Proof of (A.1) Because every diagonal element of H is zero, it follows from Theorem 2.4 and

Theorems A.1 to A.3 that

$$\begin{aligned}
G(H) &= \{z \in \mathbb{C} \mid |z| \le \lambda_{\mathrm{G}}\}, \quad \lambda_{\mathrm{G}} \equiv \max_{1\le i\le m+n} r_i'(H),\\
O(H) &= \{z \in \mathbb{C} \mid |z| \le \lambda_{\mathrm{O}}\}, \quad \lambda_{\mathrm{O}} \equiv \max_{1\le i\le m+n} r_i'(H)^{\alpha} c_i'(H)^{1-\alpha},\\
B(H) &= \{z \in \mathbb{C} \mid |z| \le \lambda_{\mathrm{B}}\}, \quad \lambda_{\mathrm{B}} \equiv \max_{1\le i<j\le m+n} \sqrt{r_i'(H) r_j'(H)},\\
N(H) &= \{z \in \mathbb{C} \mid |z| \le \lambda_{\mathrm{N}}\}, \quad \lambda_{\mathrm{N}} \equiv \max_{1\le i\le m+n} \sqrt{\sum_{\substack{1\le j\le m+n,\\ j\ne i}} |H_{ij}| r_j'(H)}.
\end{aligned}$$

From the 2×2 block structure of H,

$$\begin{aligned}
\lambda_{\mathrm{G}} &= \max\left\{\max_{1\le i\le m} r_i'(H), \max_{m+1\le i\le m+n} r_i'(H)\right\}\\
&= \max\left\{\max_{1\le i\le m} r_i(A), \max_{1\le i\le n} c_i(A)\right\} = \max\left\{\|A\|_\infty, \|A\|_1\right\}.
\end{aligned} \tag{A.2}$$

Because H is Hermitian,

$$\lambda_{\mathrm{O}} = \max_{1\le i\le m+n} r_i'(H)^{\alpha} r_i'(H)^{1-\alpha} = \max_{1\le i\le m+n} r_i'(H) = \lambda_{\mathrm{G}}. \tag{A.3}$$

From the 2×2 block structure of H,

$$\begin{aligned}
\lambda_{\mathrm{B}} &= \max\left\{\max_{1\le i<j\le m} \sqrt{r_i'(H) r_j'(H)}, \max_{m+1\le i<j\le m+n} \sqrt{r_i'(H) r_j'(H)}, \max_{\substack{1\le i\le m,\\ m+1\le j\le m+n}} \sqrt{r_i'(H) r_j'(H)}\right\}\\
&= \max\left\{\max_{1\le i<j\le m} \sqrt{r_i(A) r_j(A)}, \max_{1\le i<j\le n} \sqrt{c_i(A) c_j(A)}, \max_{\substack{1\le i\le m,\\ 1\le j\le n}} \sqrt{r_i(A) c_j(A)}\right\}\\
&= \max\left\{\max_{1\le i<j\le m} \sqrt{r_i(A) r_j(A)}, \max_{1\le i<j\le n} \sqrt{c_i(A) c_j(A)}, \sqrt{\|A\|_\infty \cdot \|A\|_1}\right\}.
\end{aligned} \tag{A.4}$$

For the first and second terms in the parenthesis of equation (A.4),

$$\begin{aligned}
\max_{1\le i<j\le m} \sqrt{r_i(A) r_j(A)} &\le \max_{\substack{1\le i\le m,\\ 1\le j\le m}} \sqrt{r_i(A) r_j(A)} = \max_{1\le i\le m} r_i(A) = \|A\|_\infty,\\
\max_{1\le i<j\le n} \sqrt{c_i(A) c_j(A)} &\le \max_{\substack{1\le i\le n,\\ 1\le j\le n}} \sqrt{c_i(A) c_j(A)} = \max_{1\le i\le n} c_i(A) = \|A\|_1,
\end{aligned}$$

which leads to

$$\sqrt{\|A\|_1 \cdot \|A\|_\infty} \le \lambda_{\mathrm{B}} \le \max\{\|A\|_1, \|A\|_\infty\}. \tag{A.5}$$

From the 2×2 block structure of H,

$$\begin{aligned}
\lambda_{\mathrm{N}} &= \max\left\{\max_{1\le i\le m} \sqrt{\sum_{m+1\le j\le m+n} |H_{ij}| r_j'(H)}, \max_{m+1\le i\le m+n} \sqrt{\sum_{1\le j\le m} |H_{ij}| r_j'(H)}\right\}\\
&= \max\left\{\max_{1\le i\le m} \sqrt{\sum_{1\le j\le n} |A_{ij}| c_j(A)}, \max_{1\le i\le n} \sqrt{\sum_{1\le j\le m} |A_{ji}| r_j(A)}\right\}.
\end{aligned} \tag{A.6}$$

For the first and second terms in the parenthesis of equation (A.6),

$$\begin{aligned}
\max_{1\le i\le m}\sqrt{\sum_{1\le j\le n}|A_{ij}|c_j(A)} &\le \max_{1\le i\le m}\sqrt{\sum_{1\le j\le n}|A_{ij}|\cdot \max_{1\le j\le n} c_j(A)}\\
&= \max_{1\le i\le m}\sqrt{r_i(A)}\cdot \max_{1\le j\le n}\sqrt{c_j(A)} = \sqrt{\|A\|_\infty\cdot\|A\|_1},\\
\max_{1\le i\le n}\sqrt{\sum_{1\le j\le m}|A_{ji}|r_j(A)} &\le \max_{1\le i\le n}\sqrt{\sum_{1\le j\le m}|A_{ji}|\cdot \max_{1\le j\le m} r_j(A)}\\
&= \max_{1\le i\le n}\sqrt{c_i(A)}\cdot \max_{1\le j\le m}\sqrt{r_j(A)} = \sqrt{\|A\|_1\cdot\|A\|_\infty},
\end{aligned}$$

which leads to

$$\lambda_{\mathrm{N}} \le \sqrt{\|A\|_1\cdot\|A\|_\infty}. \tag{A.7}$$

The inequality (A.1) follows from equations (A.2), (A.3), (A.5), and (A.7). □

Finally, it is shown below that no inequality holds in general between $\|A\|_{\mathrm{F}}$ and λ_{N}.

- For rank–1 matrix A, $\|A\|_{\mathrm{F}} = \sigma_1$, while it is not necessarily true that $\lambda_{\mathrm{N}} = \sigma_1$.
- For diagonal matrix A, $\lambda_{\mathrm{N}} = \sigma_1$, while it is not necessarily true that $\|A\|_{\mathrm{F}} = \sigma_1$.

Bibliography

[1] Amestoy, P. R., Davis, T. A. and Duff, I. S., An approximate minimum degree ordering algorithm, SIAM J. Matrix Anal. Appl., **17** (1996), 886–905.

[2] Amestoy, P. R., Duff, I. S., L'Excellent, J.-Y. and Koster, J., A fully asynchronous multifrontal solver using distributed dynamic scheduling, SIAM J. Matrix Anal. Appl., **23** (2001), 15–41.

[3] Amestoy, P. R., Guermouche, A., L'Excellent, J.-Y. and Pralet, S., Hybrid scheduling for the parallel solution of linear systems, Parallel Comput., **32** (2006), 136–156.

[4] Anderson, E., Bai, Z., Bischof, C., Blackford, L., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A. and Sorensen, D., LAPACK Users' Guide, Third edition, SIAM, Philadelphia, 1999.

[5] Bai, Z., Demmel, J., Dongarra, J., Ruhe, A. and van der Vorst, H., Templates for the solution of algebraic eigenvalue problems: A practical guide, SIAM, Philadelphia, 2000.

[6] Betcke, T., Higham, N. J., Mehrmann, V., Schröder, C. and Tisseur, F., NLEVP: A collection of nonlinear eigenvalue problems, ACM Trans. Math. Softw., **39** (2013), 7:1–7:28.

[7] Blackford, L. S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D. and Whaley, R. C., ScaLAPACK Users' Guide, SIAM, Philadelphia, 1997.

[8] Brent, R. P., An algorithm with guaranteed convergence for finding a zero of a function, Comput. J., **14** (1971), 422–425.

[9] Bunch, J. R. and Kaufman, L., Some stable methods for calculating inertia and solving symmetric linear systems, Math. Comput., **31** (1977), 163–179.

[10] Cerdá, J. and Soria, F., Accurate and transferable extended Hückel-type tight-binding parameters, Phys. Rev. B, **61** (2000), 7965–7971.

[11] Cuppen, J. J. M., A divide and conquer method for the symmetric tridiagonal eigenproblem, Numer. Math., **36** (1981), 177–195.

[12] Davis, T. A. and Hu, Y., The University of Florida Sparse Matrix Collection, ACM Trans. Math. Softw., **38** (2011), 1:1–1:25.

[13] Davis, T. A., Rajamanickam, S. and Sid-Lakhdar, W. M., A survey of direct methods for sparse linear systems, Acta Numer., **25** (2016), 383–566.

[14] Demmel, J. and Kahan, W., Accurate singular values of bidiagonal matrices, SIAM J. Sci. Stat. Comput., **11** (1990), 873–912.

[15] Demmel, J. W., Marques, O. A., Parlett, B. N. and Vömel, C., Performance and accuracy of LAPACK's symmetric tridiagonal eigensolvers, SIAM J. Sci. Comput., **30** (2008), 1508–1526.

[16] Duff, I. S. and Reid, J. K., The multifrontal solution of indefinite sparse symmetric linear equations, ACM Trans. Math. Softw., **9** (1983), 302–325.

[17] EigenKernel - a package of hybrid parallel solvers for eigenvalue problems, `https://github.com/eigenkernel` and `https://ma.issp.u-tokyo.ac.jp/en/app/254`.

[18] Ericsson, T. and Ruhe, A., The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems, Math. Comput., **35** (1980), 1251–1268.

[19] Golub, G. and Kahan, W., Calculating the singular values and pseudo-inverse of a matrix, J. Soc. Ind. Appl. Math. Ser. B Numer. Anal., **2** (1965), 205–224.

[20] Graduate Program for Real-World Data Circulation Leaders, Nagoya University, `http://www.rwdc.is.nagoya-u.ac.jp`.

[21] Grimes, R. G., Lewis, J. G. and Simon, H. D., A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems, SIAM J. Matrix Anal. Appl., **15** (1994), 228–272.

[22] Gu, M. and Eisenstat, S. C., A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem, SIAM J. Matrix Anal. Appl., **16** (1995), 172–191.

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

[24] Hochstenbach, M. E., Harmonic and refined extraction methods for the singular value problem, with applications in least squares problems, BIT Numer. Math., **44** (2004), 721–754.

[25] Hochstenbach, M. E. and Plestenjak, B., Backward error, condition numbers, and pseudospectra for the multiparameter eigenvalue problem, Linear Algebra Appl., **375** (2003), 63–81.

[26] Hohenberg, P. and Kohn, W., Inhomogeneous electron gas, Phys. Rev., **136** (1964), B864–B871.

[27] Horn, R. A. and Johnson, C. R., Matrix Analysis, Second edition, Cambridge University Press, New York, 2013.

[28] Hoshi, T., Akiyama, Y., Tanaka, T. and Ohno, T., Ten-million-atom electronic structure calculations on the K computer with a massively parallel order-N theory, J. Phys. Soc. Jpn., **82** (2013), 023710.

[29] Hoshi, T. and Fujiwara, T., Domain boundary formation in helical multishell gold nanowires, J. Phys.: Condens. Matter, **21** (2009), 272201.

[30] Hoshi, T., Imachi, H., Kumahata, K., Terai, M., Miyamoto, K., Minami, K. and Shoji, F., Extremely scalable algorithm for 10^8-atom quantum material simulation on the full system of the K computer, in Proceedings of the 7th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, 33–40 (2016).

[31] Hoshi, T., Imachi, H., Kuwata, A., Kakuda, K., Fujita, T. and Matsui, H., Numerical aspect of large-scale electronic state calculation for flexible device material, arXiv:1808.02027, 2018.

[32] Hoshi, T., Imachi, H., Oohira, K., Abe, Y. and Hukushima, K., Principal component analysis with electronic wavefunctions for exploration of organic polymer device materials, International Meeting on High-Dimensional Data-Driven Science (HD^3-2017), Kyoto, Japan, September 2017.

[33] Hoshi, T., Yamamoto, S., Fujiwara, T., Sogabe, T. and Zhang, S.-L., An order-N electronic structure theory with generalized eigenvalue equations and its application to a ten-million-atom system, J. Phys.: Condens. Matter, **24** (2012), 165502.

[34] Imachi, H., Numerical Methods for Large-scale Quantum Material Simulations, Ph.D. thesis, Tottori University, 2017.

[35] Imachi, H. and Hoshi, T., Hybrid numerical solvers for massively parallel eigenvalue computations and their benchmark with electronic structure calculations, J. Inf. Process., **24** (2016), 164–172.

[36] Imachi, H., Yokoyama, S., Kaji, T., Abe, Y., Tada, T. and Hoshi, T., One-hundred-nm-scale electronic structure and transport calculations of organic polymers on the K computer, AIP Conf. Proc., **1790** (2016), 020010.

[37] k-ep - Solution of the k-th eigenvalue problem, `https://github.com/lee-djl/k-ep` and `https://ma.issp.u-tokyo.ac.jp/en/app/1412`.

[38] Karypis, G. and Kumar, V., A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput., **20** (1998), 359–392.

[39] Knyazev, A. V., Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method, SIAM J. Sci. Comput., **23** (2001), 517–541.

[40] Kohn, W., Nobel Lecture: Electronic structure of matter–wave functions and density functionals, Rev. Mod. Phys., **71** (1999), 1253–1266.

[41] Kohn, W. and Sham, L. J., Self-consistent equations including exchange and correlation effects, Phys. Rev., **140** (1965), A1133–A1138.

[42] Kostić, V., Cvetković, L. J. and Varga, R. S., Geršgorin-type localizations of generalized eigenvalues, Numer. Linear Algebra Appl., **16** (2009), 883–898.

[43] Lanczos, C., An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Natl. Bur. Stand., **45** (1950), 255–282.

[44] Lee, D., Hoshi, T., Sogabe, T., Miyatake, Y. and Zhang, S.-L., Solution of the k-th eigenvalue problem in large-scale electronic structure calculations, J. Comput. Phys., **371** (2018), 618–632.

[45] Lee, D., Hoshi, T., Sogabe, T. and Zhang, S.-L., Computing the k-th eigenpair of large-scale generalized eigenvalue problems (in Japanese), The 3rd Annual Meeting–Creation of New Functional Devices and High-Performance Materials to Support Next-Generation Industries (CDMSI), Tokyo, Japan, July 2018.

[46] Lee, D., Miyata, T., Sogabe, T., Hoshi, T. and Zhang, S.-L., An interior eigenvalue problem from electronic structure calculations, Jpn. J. Ind. Appl. Math., **30** (2013), 625–633.

[47] Lee, D., Sogabe, T., Miyatake, Y. and Zhang, S.-L., On computing the k-th singular triplet (in Japanese), Trans. Jpn. Soc. Ind. Appl. Math., accepted.

[48] Li, R., Xi, Y., Vecharynski, E., Yang, C. and Saad, Y., A thick-restart Lanczos algorithm with polynomial filtering for Hermitian eigenvalue problems, SIAM J. Sci. Comput., **38** (2016), A2512–A2534.

[49] Marek, A., Blum, V., Johanni, R., Havu, V., Lang, B., Auckenthaler, T., Heinecke, A., Bungartz, H.-J. and Lederer, H., The ELPA library: scalable parallel eigenvalue solutions for electronic structure theory and computational science, J. Phys.: Condens. Matter, **26** (2014), 213201.

[50] Martin, R. M., Electronic structure: Basic theory and practical methods, Cambridge University Press, Cambridge, 2004.

[51] Matsui, H., Mishchenko, A. S. and Hasegawa, T., Distribution of localized states from fine analysis of electron spin resonance spectra in organic transistors, Phys. Rev. Lett., **104** (2010), 056602.

[52] Meurant, G., The Lanczos and conjugate gradient algorithms: From theory to finite precision computations, SIAM, Philadelphia, 2006.

[53] Nakatsukasa, Y., Algorithms and Perturbation Theory for Matrix Eigenvalue Problems and the Singular Value Decomposition, Ph.D. thesis, University of California, Davis, 2011.

[54] Nakatsukasa, Y., Gerschgorin's theorem for generalized eigenvalue problems in the Euclidean metric, Math. Comput., **80** (2011), 2127–2142.

[55] Parlett, B. N., The symmetric eigenvalue problem, SIAM, Philadelphia, 1998, First published by Prentice-Hall, Englewood Cliffs, 1980.

[56] Polizzi, E., Density-matrix-based algorithm for solving eigenvalue problems, Phys. Rev. B, **79** (2009), 115112.

[57] Sakurai, T., Matsuo, T. and Katagiri, T., editors, Numerical Linear Algebra: Theory and HPC (in Japanese), Kyoritsu Shuppan, Tokyo, 2018.

[58] Sakurai, T. and Sugiura, H., A projection method for generalized eigenvalue problems using numerical integration, J. Comput. Appl. Math., **159** (2003), 119–128.

[59] Sleijpen, G. L. G. and van der Vorst, H. A., A Jacobi–Davidson iteration method for linear eigenvalue problems, SIAM J. Matrix Anal. Appl., **17** (1996), 401–425.

[60] Stewart, G. W., Gershgorin theory for the generalized eigenvalue problem $A\boldsymbol{x} = \lambda B\boldsymbol{x}$, Math. Comput., **29** (1975), 600–606.

[61] Terao, J., Wadahama, A., Matono, A., Tada, T., Watanabe, S., Seki, S., Fujihara, T. and Tsuji, Y., Design principle for increasing charge mobility of π-conjugated polymers using regularly localized molecular orbitals, Nat. Commun., **4** (2013), 1691.

[62] Tisseur, F. and Dongarra, J., A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures, SIAM J. Sci. Comput., **20** (1999), 2223–2236.

[63] Voss, H., Nonlinear eigenvalue problems, in Hogben, L., editor, Handbook of Linear Algebra, Second edition, chapter 60, CRC Press, Boca Raton (2014).

[64] Whaley, R. C., Petitet, A. and Dongarra, J. J., Automated empirical optimizations of software and the ATLAS project, Parallel Comput., **27** (2001), 3–35.

[65] Wu, L. and Stathopoulos, A., A preconditioned hybrid SVD method for accurately computing singular triplets of large matrices, SIAM J. Sci. Comput., **37** (2015), S365–S388.

[66] Zhang, H., Smith, B., Sternberg, M. and Zapol, P., SIPs: Shift-and-invert parallel spectral transformations, ACM Trans. Math. Softw., **33** (2007), 9.