A study on efficient eigenvalue computation using a contour integral based solver

Graduate School of Systems and Information Engineering

University of Tsukuba

March 2014

Yasunori FUTAMURA

Abstract

Matrix eigenvalue problems with large sparse matrices arise in a variety of scientific computations. The solutions of eigenvalue problems tend to be the most time-consuming part of the computations. In this study we consider to solve generalized eigenvalue problems with large sparse matrices.

Numerical methods for solving generalized eigenvalue problems are roughly categorized into two groups: methods based on unitary transformation and projection methods. A method based on unitary transformation solves an eigenproblem by transforming matrices to a simple form with an unitary transformation. Then an iterative procedure such as the QZ iteration is efficiently utilized to obtain eigenvalues. However, to store the data representing the transformations, the methods require an amount of memory proportional to the square of the matrix size. Thus, it is difficult to use a method based on unitary transformations for large sparse matrices.

In such case, one consider a projection method such as the Arnold method and the Jacobi-Davidson method. A projection method is a method which extracts approximate eigenvalues from a low dimensional subspace and is basically designed so that it accesses the matrices only in the form of matrix-vector multiplications to utilize the sparsity. However, the algorithms of such conventional projection methods mainly consist of iterative procedures. Since the iterative procedures demand frequent global synchronizations, it is difficult to perform highly scalable parallel computation. In such a situation, contour integral based methods are received attentions since their inherent hierarchical parallelism is suitable for modern highly parallel supercomputers.

The goal of the study of this thesis is to develop efficient methods and techniques for utilizing a contour integral based method. This study consists of three main topics: the development of techniques improving the performance the contour integration method itself and analyses of the techniques, the derivation of a stochastic estimator of eigenvalue distribution which can be used to set the parameters efficiently for the contour integral based eigensolver, and development of methods for solving linear systems with special

forms that arise in the contour integral based eigensolver.

A number of numerical experiments are performed to show how presented methods and techniques works for problems arising in practical applications.

Acknowledgements

First and foremost I would like to express the deepest appreciation to my advisor Professor Tetsuya Sakurai. He has devoted a great deal of his time and effort to me from the undergraduate research to the Ph.D study. His guidance helped me in all the time of research and writing of this thesis. I would not have been completed my Ph.D study without his support.

Besides my advisor, I would like to express my sincere gratitude to the rest of my thesis committee: Professor Kazuhiro Yabana, Professor Mitsuhisa Sato, Professor Daisuke Takahashi, and Associate Professor Shinichi Yamagiwa, for their encouragement, insightful comments, and hard questions.

I am deeply grateful to current and past member of my laboratory. I have had the support and encouragement of Assistant Professor Hiroto Tadano, Assistant Professor Akira Imakura, and Dr. Lei Du of University of Tsukuba. Discussions with them were quite exciting and invaluable for me.

I would like to offer my special thanks to Lecturer Jun-Ichi Iwata of the University of Tokyo and Dr. Shinnosuke Furuya of ARGO GRAPHICS Inc. They gave me insight comments and suggestions from a standpoint of computational material physics.

I owe my appreciation to the financial support from the Japan Society for the Promotion of Science Research Fellowship for Young Scientists.

Last, but not least, I would like to thank my parents and grandparents for giving birth to me at the first place and supporting me spiritually throughout my life.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background

Matrix eigenvalue problems arise in a variety of scientific or engineering computations. The solutions of eigenvalue problems tend to be the most time-consuming part of the computations. Eigenvalue problems arising in scientific computations have different mathematical characteristics. For instance, the symmetry of the matrix, the sparsity of the matrix, the number of required eigenvalues, and their locations. Due to this variation, a number of numerical methods that have different features have been proposed.

Projection type methods are known as methods designed to find some selected eigenvalues and corresponding eigenvectors of large sparse matrices. The development of efficient algorithms of the projection type methods is important for scientific computations that require solutions of eigenvalue problems of large sparse matrices.

On the other hand, modern highly parallel super-computers that have a large number of nodes are now commonly used for extra large scale scientific computations. The number of cores in a node is increasing since installations of many core coprocessors become common. This trend leads to an increase in super-computers that have hierarchical structure. However, the algorithms of conventional projection type eigensolvers such as the Arnoldi method mainly consist of iterative procedures that demand frequent global synchronizations. Thus it is difficult to perform highly scalable parallel computation on hierarchical parallel computational environments with such methods.

In such a situation, contour integral based methods are receiving attentions. A contour integral based method is a method that compute eigenvalues located in a specified contour path and corresponding eigenvectors, and they are categorized into projection type methods. The contour integration is discretized by numerical integration, and the complexity of a solution for an eigenvalue problem is transformed to that of solutions for independent systems of linear equation (linear systems) with respect to each quadrature point. Since not only the solutions of the linear systems but also the each solution of linear system can be done in parallel, the methods allow us to naturally implement a hierarchal parallel code. The contour integral based methods are relatively new compared to conventional methods. A number of possibilities for specializations, generalizations and further analysis still remains.

1.2 Aim of this thesis

The goal of this thesis is to develop efficient techniques and implementation of a contour integral based method, specifically, the Sakurai-Sugiura (SS) method [39]. In this thesis we consider the generalized eigenvalue problem

$$A\boldsymbol{u} = \lambda B\boldsymbol{u}, \tag{1.1}$$

where A and B are square matrices with complex values, λ is a scalar, $\boldsymbol{u}$ is a non-zero vector and $A - \lambda B$ is a regular matrix pencil. The generalized eigenvalue problem is a problem to find non-trivial pairs of λ and $\boldsymbol{u}$. λ is called eigenvalue, $\boldsymbol{u}$ is called eigenvector and a pair of them is called eigenpair. In some part of this thesis, the standard eigenvalue problem, the special case of (1.1) when B is the identity matrix, are also considered.

The SS method consists of numerical integration for discretization of a contour integration. The effect of numerical integration with general integration points and weights for the accuracy of the solutions have not been studied, whereas an analysis for a trapezoidal rule on a circle contour path have been done [38]. Thus we clarify the effect of the general numerical integration. In addition to this, we develop techniques to obtain better accuracy by considering the numerical integration as an operation of a filter for the input subspace.

In the SS method, there are several parameters such as the block size and the moment size. The setting of parameters affects the accuracy of the solutions and the computational complexity of the method. The parameters can be efficiently set by using a knowledge of the distribution of the eigenvalues. The development of a method for computing an estimation of the eigenvalue distribution is also aimed.

As mentioned earlier, solutions of linear systems with respect to integration points are required in the SS method. This leads to demand for decision about the algorithm for solving the linear systems. Unfortunately, the fastest and the most efficient method depends on the non-zero pattern and the values of given coefficient matrix, and it cannot be preliminarily known in general. In this thesis, we also intend to develop an efficient algorithms for the linear systems arise in the SS method by focusing on special forms of them.

1.3 Organization of this thesis

In this section, we describe the organization of this thesis.

In Chapter 2, we overview numerical methods for solving standard and generalized eigenvalue problem. The position of the contour integral based method in the set of methods is described.

In Chapter 3, some numerical properties of the SS method are presented by regarding the numerical contour integration as a filter for a subspace. The effect of the numerical contour integration is analyzed under a certain condition of the quadrature points and the weights. This condition is more general than the condition that the trapezoidal rule is used on a circle contour path. In addition, effect of iterative application of the SS method is clarified with the notion of the subspace filtering.

In Chapter 4, we propose a stochastic estimation method of eigenvalue counting within a given closed curve. The method is feasible for large sparse matrices or matrices that are only referenced in the form of matrix-vector multiplication. A stochastic estimation method for the eigenvalue distribution is defined by separating the given domain to several sub-domains and estimating the eigenvalue count in each sub-domain. The proposed method can be used for a preprocess of the SS method to set efficient parameters. Some numerical experiments are shown to see the performance of the proposed method.

In Chapter 5, we show the derivations of several block Krylov type methods for the approximation of $C^{\mathrm{H}}A^{-1}B$, where A is a square matrix, B and C are tall-skinny rectangular matrices. This problem is arises in the special case of the SS method and also the method described in Chapter 4. Several numerical examples are shown to compare the derived methods with other existing block Krylov type methods.

In Chapter 6, we propose a CG type method for linear systems with multiple shifts and multiple right hand sides and efficient implementation techniques of the proposed method. The proposed method can be used for linear systems that arise in the SS method when the SS method applied to Hermitian standard eigenproblems. We compare the proposed method with a conventional method by several numerical experiments.

1.4 Basic notations

Throughout this thesis we use following notations:

- We denote a vector as bold lower case character
- We denote the set of real number as $\mathbb{R}$
- We denote the set of n dimensional real vectors as $\mathbb{R}^n$
- We denote the set of $n \times m$ real matrices as $\mathbb{R}^{n \times m}$
- We denote the set of complex number as $\mathbb{C}$
- We denote the set of n dimensional complex vectors as $\mathbb{C}^n$
- We denote the set of $n \times m$ dimensional complex matrices as $\mathbb{C}^{n \times m}$
- We denote the transpose of a matrix A as A^{T}
- We denote the conjugate transpose of a matrix A as A^{H}
- We denote the Frobenius norm of a matrix A as $||A||_F$
- We denote the n dimensional identity matrix as I_n, or we simply denote I if there is no confusion
- We denote the $n \times m$ zero matrix as $O^{n \times m}$
- We denote the zero vector as $\mathbf{0}$
- We denote 2-norm of a vector $\boldsymbol{a}$ as $||\boldsymbol{a}||_2$.
- We denote the subspace spanned by the column vectors of a matrix V as $\mathrm{Span}\{V\}$
- We use Fortran or MATLAB notations $A(i : j, k : m)$ to denote the submatrix of A represents rows i through j and columns k through m.

Chapter 2

Numerical methods for Eigenvalue Problems

In this chapter we describe overview of the numerical methods for eigenvalue problems. Numerical methods for eigenvalue problems are roughly categorized into two broad types :

- Method based on unitary transformations,
- Projection method.

In Section 2.1, we introduce methods based on unitary transformations. Projection methods are described in Section 2.2.

2.1 Methods based on unitary transformations

We first describe the methods for solving standard eigenvalue problem. Then we introduce the methods for generalized eigenvalue problems afterward.

2.1.1 Methods for standard eigenvalue problems

Let $A \in \mathbb{C}^{n\times n}$ be and $P \in \mathbb{C}^{n\times n}$ be a non-singular matrix. The matrix

$$C = PAP^{-1}$$

has same eigenvalues of A. This transformation $A \to C$ is called the similarity transformation.

The QR method is known as the most common practical algorithm for computing eigenpairs of standard eigenproblems when one needs to compute all eigenpair of non-symmetric dense matrix. In the QR method, one first computes a decomposition $STS^{\mathrm{H}} = A$ so called a Schur decomposition, where S is an unitary matrix and T is an upper triangular matrix whose diagonal elements are the eigenvalues of A. Then eigenvectors of A are computed by solving triangular systems with respect to T.

In the QR method, one iteratively computes the QR decomposition of a matrix to obtain a schur form T. This iteration is called the QR iteration. Algorithm 2.1 shows the pseudocode of the QR iteration. It is known that A_i

Algorithm 2.1 QR iteration.

```
Let A_0 = A
for i=0,1,... do
    Compute QR decomposition: Q_i R_i = A_i
    Compute A_{i+1} = R_i Q_i
end for
```

converges to a Schur form T, if all eigenvalues of A are distinct in absolute values. More specifically, the diagonal elements of $\lim_{i\to\infty} A_i$ (the eigenvalues of A) line up in descending order. Note that, in a QR iteration,

$$A_{i+1} = Q_i^{-1} A_i Q_i = Q_i^{\mathrm{H}} A_i Q_i \tag{2.1}$$

holds. Thus the QR iteration can be considered an iteration of an unitary similarity transformation. The total computational cost of this naive QR iteration is too expensive since the cost of single QR iteration is $O(n^3)$.

In practice, one first reduces the matrix A to an upper Hessenberg form by the Householder transformation. This Hessenberg reduction is also an unitary similarity transformation. Thus it preserves the eigenvalues. Since the transformation (2.1) preserves the Hessenberg structure of A_i, the QR decomposition can be cheaply done by the Givens rotation. The computational cost for single QR iteration become $O(n^2)$ by using the Givens rotation. It is faster than full matrix QR iteration by an order of magnitude.

In order to improve the rate of the convergence, one may introduce a *shift* for the QR iteration. Algorithm 2.2 illustrates the algorithm of the QR iteration with a shift. The shift is a scalar σ_i which can be changed from iteration to iteration. A transformation $A_i \to A_{i+1}$ is also an unitary similarity transformation even if a shift is introduced. In practice, for complex problems, the element of lower right corner of A_i is usually taken for σ_i. A local quadratic convergence is obtained by this shift. Additionally, a transformation $A_i \to A_{i+1}$ also preserves the Hessenberg form. The Householder transformation can be applied at the first step to reduce the computational cost.

Algorithm 2.2 QR iteration with a shift.

1: Let $A_0 = A$
2: **for** i=0,1,... **do**
3: Compute QR decomposition: $Q_i R_i = A_i - \sigma_i I$
4: Compute $A_{i+1} = R_i Q_i + \sigma_i I$
5: **end for**

Hermitian case

For a Hermitian matrix $A = A^{\mathrm{H}}$, several practical methods have been proposed. In what follows, we briefly introduce methods for Hermitian problems.

The tridiagonal QR method is one of widely used algorithm. In the QR method, the Hessenberg reduction at the first step leads to a tridiagonal form if A is Hermitian since the Householder transformation is an unitary transformation. The tridiagonal form allow us to perform a QR iteration with $O(n)$ operations.

The divide-and-conquer method is a method solves a (tridiagonal) Hermitian eigenproblem by recursively dividing the original problem to smaller problems. This method also requires the tridiagonal reduction of the original matrix A. This method is known as the fastest algorithm if one needs to compute all eigenvalues and eigenvectors of a tridiagonal Hermitian matrix sized more than 25 [8]. When the divided matrix become sufficiently small, the QR method is applied. At the conquer phase, one needs to solve a rational equation. This problem is usually solved by a Newton's method.

The bisection method is prefered for computing only k eigenvalues located in some interval or indexed with some index range. Suppose that A is decomposed as $A = LDL^{\mathrm{H}}$, where D is a diagonal matrix and L is a lower triangular matrix (LDLH decomposition). Let $\pi(A)$, $\zeta(A)$, and $\mu(A)$ be eigenvalue count of positive numbers, zeros, and negative numbers, respectively. According to the Sylvester's low of inertia, it is said that

$$\pi(A - \sigma I) = \pi(D),\ \zeta(A - \sigma I) = \zeta(D),\ \text{and}\ \mu(A - \sigma I) = \mu(D),$$

where σ is a real scalar. Thus one can obtain the eigenvalue counts less than σ, equal to σ, and more than σ by counting elements of D. Once an index of eigenvalue is given, One can compute the eigenvalue which has the index with arbitral precision by the bisection search. As a preprocess, one reduces A to a triangular form by Householder transformations. This allows one to perform the LDLH decomposition with $O(n)$ operations.

The Jacobi's method is a classical method for Hermitian eigenproblems. In contrast to the above methods, this method does not require the tridiagonal form. In the Jacobi's method one performs the Givens rotation iteratively to let the matrix converge to the diagonal matrix whose entries are the eigenvalues of A. Although the Jacobi's method is often slower then the above methods, this method and its variants have received attention in recent years since they are expected to provide high scalability on highly parallel computational environments due to their inherent parallelism.

2.1.2 Methods for generalized eigenvalue problems

Now we consider a generalized eigenvalue problem (1.1). Supposing that $A, B \in \mathbb{C}^{n\times n}$, there exist unitary matrices U, V such that $U^{\mathrm{H}}AV = R$ and $U^{\mathrm{H}}BV = S$, where R, S are upper triangular matrices. This pair of decompositions is called the generalized Schur decomposition or the QZ decomposition [15]. If $A - \lambda B$ is regular matrix pencil, finite eigenvalues of (1.1) is $\lambda_i = r_{i,i}/s_{i,i}$ ($s_{i,i} \neq 0$). Here, $r_{i,i}$ and $s_{i,i}$ are the i-th diagonal elements of R and S, respectively.

There is an analogue of the QR method for the generalized eigenproblem, which is called as the QZ method [30]. In the QZ method, one first transforms A to the upper Hessenberg form H_A and transform B to the upper triangular form T_B by using the Householder transformation and the Givens rotations.

It is worth mentioning here that when the $(k + 1, k)$ element of H_A is 0, the matrix pencil $H_A - \lambda T_B$ of the generalized eigenproblem can be split into to two smaller pencils

$$H_A(1 : k, 1 : k) - \lambda T_B(1 : k, 1 : k)$$

and

$$H_A(k + 1 : n, k + 1 : n) - \lambda T_B(k + 1 : n, k + 1 : n).$$

Additionally, if the (k, k) element of T_B is zero, one can zero the $(n - 1, n)$ element of H_A and the (n, n) element of T_B with Givens rotations. Since zero diagonal elements of T_B can be cut out from the problem in this way, we can assume that T_B is non-singular without loss of generality.

Once the pair of A and B is transformed to upper-Hessenberg-triangular form, the generalized Schur decomposition is computed by the QZ iteration. The QZ iteration is equivalent to the QR iteration which is applied to $H_A T_B^{-1}$. For more details see [15, 30].

Hermitian definite case

If matrix A and B Hermitian and $\alpha A + \beta B$ is positive definite with some scalars α and β, the matrix pencil $A - \lambda B$ is called Hermitian definite pencil. In this case, a generalized eigenvalue problem (1.1) is reduced to

$$A\boldsymbol{u} = \theta(\alpha A + \beta B)\boldsymbol{u}. \tag{2.2}$$

The new right hand side matrix $\tilde{B} \equiv \alpha A + \beta B$ can be decomposed as $\tilde{B} = LL^{\mathrm{H}}$ with lower triangular matrix L (the Cholesky decomposition). Using L, (2.2) can be reduced to a standard eigenvalue problem

$$\tilde{A}\tilde{\boldsymbol{u}} - \theta\tilde{\boldsymbol{u}}, \tag{2.3}$$

where $\tilde{A} \equiv L^{-1}AL^{-\mathrm{H}}$ and $\tilde{\boldsymbol{u}} \equiv L^{\mathrm{H}}\boldsymbol{u}$. Since $\tilde{A}$ is Hermitian, (2.3) can be solved by the algorithms for Hermitian standard eigenproblems described in the previous subsection. Once (2.3) is solved, an eigenvalue λ of the original problem can be computed as

$$\lambda = \frac{\beta\theta}{1 - \alpha\theta}.$$

For more details about methods based on unitary transformations, consult [8, 15, 48].

2.2 Projection methods

In this section we discuss about projection methods.

2.2.1 Methods for standard eigenvalue problems

In this subsection we describe projection methods for solving standard eigenproblems. Methods for generalized eigenproblems are described in the next subsection.

Projection method is known as a type of method finds an approximate eigenvector $\tilde{\boldsymbol{u}}$ from a m dimensional subspace $\mathcal{M}$ i.e.

$$\tilde{\boldsymbol{u}} \in \mathcal{M} \tag{2.4}$$

and imposes a condition to the residual with k-th dimensional subspace $\mathcal{L}$ such that

$$A\tilde{\boldsymbol{u}} - \lambda\tilde{\boldsymbol{u}} \perp \mathcal{L}. \tag{2.5}$$

Usually, $m \ll n$. The subspaces $\mathcal{M}$ and $\mathcal{L}$ are produced by some procedure. Suppose that $V, W \in \mathbb{C}^{n\times m}$ are given such that $\mathcal{M} = \mathrm{Span}\{V\}$ and $\mathcal{L} = \mathrm{Span}\{W\}$. (2.4) and (2.5) can be written as

$$\tilde{\boldsymbol{u}} = V\boldsymbol{y}$$

and

$$W^{\mathrm{H}}(A\boldsymbol{u} - \lambda\boldsymbol{u}) = \boldsymbol{0}.$$

Therefore, one can obtain an approximate eigenpairs by computing a small m dimensional generalized eigenvalue problem:

$$W^{\mathrm{H}}AV\boldsymbol{y} = \theta W^{\mathrm{H}}V\boldsymbol{y}. \tag{2.6}$$

The small generalized eigenproblem (2.6) is usually solved by a method based on unitary transformations. In some cases the condition

$$V^{\mathrm{H}}W = I_m \tag{2.7}$$

maintained. The reduced problem become a standard eigenvalue

$$W^{\mathrm{H}}AV\boldsymbol{y} = \theta\boldsymbol{y}$$

in such cases. Note that, to accomplish (2.7), for any basis V and W of $\mathcal{M}$ and $\mathcal{L}$, respectively, $\det(V^{\mathrm{H}}W) \neq 0$ must hold.

If $\mathcal{M} = \mathcal{L}$ in a projection method, the method is called orthogonal projection method. Otherwise, it is called oblique projection method. The condition imposed in orthogonal projection method is called the Ritz-Galerkin condition. And the condition for the oblique projection method is called the Petrov-Galerkin condition.

The procedure to obtain eigenpairs in projection methods is known as the Rayleigh-Ritz procedure. The algorithm of the procedure for orthogonal projection method is shown in Algorithm 2.3

Algorithm 2.3 Rayleigh-Ritz procedure. (for orthogonal projection)

1: Compute orthogonal basis $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k$, let $V \equiv [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k]$
2: Compute $\tilde{A} = V^{\mathrm{H}}AV$
3: Solve eigenvalue problem $\tilde{A}\boldsymbol{y} = \theta\boldsymbol{y}$
4: Compute $\tilde{\boldsymbol{u}} = V\boldsymbol{y}$
5: Let $(\theta, \tilde{\boldsymbol{u}})$ be an approximate eigenpairs

The most common choice of $\mathcal{M}$ is the Krylov subspace

$$\mathcal{K}_m(A; \boldsymbol{v}) \equiv \mathrm{Span}\{\boldsymbol{v}, A\boldsymbol{v}, A^2\boldsymbol{v}, \ldots, A^{m-1}\boldsymbol{v}\},$$

where $\boldsymbol{v}$ is an arbitrary non-zero vector. Using the Krylov subspace we can proceed the computation with only matrix-vector multiplications. To compute orthogonal basis, the Arnoldi procedure is used. Algorithm 2.4 shows the Arnoldi process. By using the Arnoldi process, we can obtain the orthogonal basis of the Krylov subspace $\mathcal{K}_m(A; \tilde{\boldsymbol{v}}_1)$, where $\tilde{\boldsymbol{v}}_1$ is the starting

Algorithm 2.4 Arnoldi process.

1: Choose the initial vector $\tilde{\boldsymbol{v}}_1$
2: $\boldsymbol{v}_1 = \dfrac{\tilde{\boldsymbol{v}}_1}{||\tilde{\boldsymbol{v}}_1||_2}$
3: **for** $k = 1, 2, \ldots, m$ **do**
4: $\quad h_{i,k} = \boldsymbol{v}_i^{\mathrm{H}} A \boldsymbol{v}_k$, $(i = 1, 2, \ldots, k)$
5: $\quad \tilde{\boldsymbol{v}}_{k+1} = A\boldsymbol{v}_k - \sum_{i=1}^{k} h_{i,k}\boldsymbol{v}_i$
6: $\quad h_{k+1,k} = ||\tilde{\boldsymbol{v}}_{k+1}||_2$
7: $\quad \boldsymbol{v}_{k+1} = \dfrac{\tilde{\boldsymbol{v}}_{k+1}}{h_{k+1,k}}$
8: **end for**

vector. Let here $V_m \equiv [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m]$. In addition, the Arnoldi process gives us the equation

$$AV_m = V_m H_m + h_{m+1}\boldsymbol{v}_{m+1}\boldsymbol{e}_m^{\mathrm{T}}$$

with a Hessenberg matrix

$$H_m = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,m-1} & h_{1,m} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,m-1} & h_{2,m} \\ 0 & \ddots & \ddots & \vdots & h_{2,m} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{m,m-1} & h_{m,m} \end{pmatrix}.$$

Due to the orthonormality of $\{\boldsymbol{v}_i\}_{i=1,2,\ldots,m+1}$, we have

$$H_m = V_m^{\mathrm{H}} A V_m.$$

Thus one can obtain approximate eigenpairs by solving a small eigenproblem

$$H_m \boldsymbol{y} = \theta \boldsymbol{y}.$$

Since H_m is already a Hessenberg matrix, the QR method can be cheaply utilized. It is known that the eigenvalues located in the outermost part of the spectrum tend to be well approximated in the Arnoldi method.

Since the computational complexity and the memory requirement for the Arnoldi process is $O(nm^2)$ and $O(nm)$, respectively, one needs to terminate the process at some m. Then one restarts the process with new $\tilde{\boldsymbol{v}}_1$ which includes an information from the previous cycle (e.g. $\tilde{\boldsymbol{v}}_1$ set to be an approximate eigenvector of the largest eigenvalue). In order to effectively use the information in the previous cycles of the Arnoldi process, more sophisticated (implicitly) restarting techniques such as the implicitly restarted Arnoldi method [43], and the Krylov-Schur method [44] have been proposed.

If the interested eigenvalues are located inside the spectrum and are clustered, the spectrum transformation is used for transform the location of eigenvalues of interest to the exterior position. A major instance of the spectrum transformation is the shift-and-invert spectrum transformation. In the shift-and-invert transformation, the original eigenvalue problem is transformed to

$$(A - \sigma I)^{-1}\boldsymbol{u} = \tau \boldsymbol{u}, \tag{2.8}$$

where

$$\tau \equiv \frac{1}{\lambda - \sigma}$$

and σ is a scalar such that $\det(A - \sigma I) \neq 0$. The eigenvalue which is the closest to σ become the largest eigenvalue in the absolute value in (2.8). Thus the Arnoldi method applied to (2.8) easily obtain eigenvalues that are close to σ. Unfortunately, if the shift-and-invert transformation is used, one needs to solve the linear system whose coefficient matrix is $(A - \sigma I)$. This often increase the computational complexity of an iteration by an order of magnitude.

There is the type of projection method which is based on a contour integration

$$S_k \equiv \frac{1}{2\pi \mathrm{i}} \int_\Gamma z^k (zI - A)^{-1} Y \, \mathrm{d}z, \quad k = 0, 1, \ldots, M-1$$

which seeks eigenvalues inside of closed curve Γ and corresponding eigenvectors. Here $Y \in \mathbb{C}^{n \times s}$ is a basis which contains wanted eigenvectors as its components. The column vectors of matrix $[S_0, S_1, \ldots, S_{M-1}]$ is used as basis for the Rayleigh-Ritz procedure. In practice, one approximates S_k by a numerical quadrature. The contour integral based eigensolver is a main topic of this thesis. More detailed discussions are shown in Chapter 3.

When interior eigenvalues are required and also one wants to avoid solutions of linear systems for a spectrum transformation, an alternative choice is to use the Jacobi-Davidson method. A Jacobi-Davidson method consists of two main factors: the construction of orthogonal basis and the solutions of linear systems so called the correction equation. The correction equation is roughly solved with an iterative method. Thus computational effort for one iteration of the Jacobi-Davidson method tend to be smaller than that of the Arnoldi method with the spectral transformation which demands accurate solutions of linear systems.

Hermitian case

When A is a Hermitian matrix, the Hessenberg matrix H_m of the Arnoldi process becomes a Hermitian tridiagonal matrix since $H_m = V_m^{\mathrm{H}} A V_m$. Thus the Arnoldi process is simplified by this property. This simplified process is called the Lanczos process. Moreover, $h_{k+1,k}$ is real since it is defined by a norm and $h_{k,k}$ is also real since A is Hermitian. Therefore H_m must be a real symmetric tridiagonal matrix. Algorithm 2.5 shows the Lanczos process. In Algorithm 2.5, we set $\alpha_k \equiv h_{k,k}$, $\beta_k \equiv h_{k-1,k}$.

Algorithm 2.5 Lanczos process.

1: Choose the initial vector $\tilde{\boldsymbol{v}}_1$
2: $\boldsymbol{v}_0 = \boldsymbol{0}, \beta_1 = 0, \boldsymbol{v}_1 = \dfrac{\tilde{\boldsymbol{v}}_1}{||\tilde{\boldsymbol{v}}_1||_2}$
3: **for** $k = 1, 2, \dots, m$ **do**
4: $\quad \hat{\boldsymbol{v}}_{k+1} = A\boldsymbol{v}_k - \beta_j \boldsymbol{v}_{k-1}$
5: $\quad \alpha_k = \hat{\boldsymbol{v}}_{k+1}^{\mathrm{H}} \boldsymbol{v}_k$
6: $\quad \tilde{\boldsymbol{v}}_{k+1} = \hat{\boldsymbol{v}}_{k+1} - \alpha_k \boldsymbol{v}_k$
7: $\quad \beta_{k+1} = ||\tilde{\boldsymbol{v}}_{k+1}||_2$
8: $\quad \boldsymbol{v}_{k+1} = \dfrac{\tilde{\boldsymbol{v}}_{k+1}}{\beta_{k+1}}$
9: **end for**

Unfortunately, in practice, global orthogonality of $\{\boldsymbol{v}_k\}_{k=1}^{m+1}$ of the Lanczos process is usually lost. Thus reorthogonalization is performed to improve numerical stability of the Lanczos process. Several strategies for reorthogonalization have been proposed.

2.2.2 Methods for generalized eigenvalue problems

In this subsection, we describe several approaches for solving generalized eigenproblem (1.1) by projection methods.

If B is non-singular, one reduce the original problem to a standard eigenvalue problem

$$B^{-1} A \boldsymbol{u} = \lambda \boldsymbol{u}.$$

by the inverse of B. Then one can apply projection methods in the previous subsection such as the Arnoldi method to obtain approximate solutions for $(\lambda, \boldsymbol{u})$. In this approach, $B^{-1}A$ is not computed explicitly, instead, one calculates matrix-vector multiplication $\boldsymbol{y} = B^{-1} A \boldsymbol{x}$ as follows:

1. Compute $\boldsymbol{z} = A\boldsymbol{x}$,

2. Solve linear system $B\boldsymbol{y} = \boldsymbol{z}$ for $\boldsymbol{y}$.

If B is singular, one may consider spectral transformation to generalized eigenvalue problems. The analogue of the shift-and-invert transformation (2.8) for generalized eigenproblems

$$(A - \sigma B)^{-1} B\boldsymbol{u} = \tau\boldsymbol{u}$$

is a commonly used spectral transformation. Here,

$$\tau \equiv \frac{1}{\lambda - \sigma}.$$

Using this transformation, one can apply the Arnoldi method to $(A-\sigma B)^{-1}B$. In this case, one needs to solve a linear system whose coefficient matrix is $(A - \sigma B)$ at each iteration of the Arnoldi method. Another possibility is to use the Jacobi-Davidson algorithm. This algorithm only requires rough solution of linear systems related to $(A - \theta B)$ with some scalar θ.

A method based on contour integration is also applicable for generalized eigenvalue problem. For generalized eigenvalue problem, basis for the Rayleigh-Ritz procedure is given by

$$S_k \equiv \frac{1}{2\pi \mathrm{i}} \int_\Gamma z^k (zB - A)^{-1} BY \,\mathrm{d}z, \quad k = 0, 1, \ldots, M-1 \quad .$$

Hermitian definite case

As seen in Section 2.1.2, the original generalized eigenvalue problem is reduced to a standard eigenproblem

$$L^{-1}AL^{-H}(L^{\mathrm{H}}\boldsymbol{u}) = \lambda(L^{\mathrm{H}}\boldsymbol{u}).$$

with cholesky factorization of $B = LL^{\mathrm{H}}$ if B is Hermitian and positive definite. Simular to the above case, in stead of forming $L^{-\mathrm{H}}AL^{\mathrm{H}}$ explicitly, one can use compute $\boldsymbol{y} = L^{-\mathrm{H}}AL^{\mathrm{H}}\boldsymbol{x}$ as follows:

1. Solve linear system $L^{\mathrm{H}}\boldsymbol{w} = \boldsymbol{x}$ for $\boldsymbol{w}$,
2. Compute $\boldsymbol{z} = A\boldsymbol{w}$,
3. Solve linear system $L\boldsymbol{y} = \boldsymbol{z}$ for $\boldsymbol{y}$.

In addition, as seen in Section 2.1.2, $L^{-\mathrm{H}}AL^{\mathrm{H}}$ is also Hermitian if A is Hermitian. Thus one can use the Lanczos method for this problem.

In this thesis, further details about the projection methods described in this subsection are not discussed and other existing projection methods are not treated. For a broader and more detailed view of projection method, see [3, 8, 36].

Chapter 3

Efficient parameter estimation and implementation of a contour integaral-based eigensolver

3.1 Introduction

A contour integral based eigensolver was proposed by Sakurai and Sugiura in 2003 [39]. This method is called the Sakurai-Sugiura (SS) method. In the original SS method in [39], a contour integral with a source vector $\boldsymbol{v}$ are used to generate a subspace spanned by a set of eigenvectors with respect to the eigenvalues in a target domain. A large-scale eigenvalue problem is reduced to a small eigenvalue problem with Hankel matrices constructed from complex moments. In [38], an interpretation for filtering of spectrum is used to discuss numerical properties of a contour integral approximated by numerical quadrature. An influence of approximation by numerical quadrature is considered as a contamination of eigencomponents, and the choice of an appropriate subspace size provides accurate eigenpairs in a target domain.

A variant of the SS method that improves numerical accuracy by using the Rayleigh-Ritz procedure is presented in [37]. Ikegami, et al. [24, 23] presented a block version of the SS method that uses multiple source vectors instead of the single source vector for the contour integrals. The block SS method improves numerical stability when the target domain contains many eigenvalues. Moreover, this method can treat multiple eigenvalues. In [1, 2], the SS method is extended to nonlinear eigenvalue problems. As related works of eigensolvers using contour integrals, Polizzi [35] proposed an iterative refinement of a contour integral method for symmetric or Hermitian positive definite eigenvalue problems. Beyn [5] proposed a method for nonlinear eigenvalue problems using contour integrals with a singular value decomposition of a matrix with a Hankel type structure. Yokota, et al. [54] proposed a Rayleigh-Ritz type method using contour integrals for nonlinear eigenvalue problems. In this method, a subspace that includes target eigenvectors are generated by contour integrals, and a large-scale nonlinear eigenvalue problem is projected to a small nonlinear eigenvalue problem, and the projected problem is solved by Hankel type nonlinear eigensolver using contour integrals.

The SS method computes a set of eigenvalues by computing the solutions to systems of linear equations

$$(z_j B - A)Y_j = BV, \quad j = 1, \ldots, N, \tag{3.1}$$

where V is a matrix with L column vectors and z_j is a shift point on the complex plane. The method computes the desired eigenvalues inside of a border defined by the set of shifts $\{z_j\}$. The first step of the SS method is the construction of a subspace that includes the eigenvectors corresponding to the eigenvalues located inside the given domain. In this step, solutions of linear systems at several shift points are used. The second step is to solve the projected problem in the subspace and to extract the approximate eigenvalues and the corresponding eigenvectors for the original problem. Since the size of the projected subspace is assumed to be small compared with the original matrix size, the computational costs of the first step is dominant.

Krylov subspace methods for multiple right-hand sides are efficient for solving the linear systems (3.1). In [31, 49], methods to improve numerical stability and convergence for block Krylov subspace methods are presented. In the case of standard eigenvalue problems, the linear systems (3.1) are shifted linear systems, and a shift invariance of the Krylov subspace reduces computational costs to obtain solutions of linear systems at several shift points [33]. The application of the SS method with the shifted CG method for shell model calculations is reported in [29]. Yamazaki, et al. [53] implemented a nonlinear version of the SS method, and evaluated parallel performances of the method.

Each of the linear systems is independent with respect to the other shifts, so each can be solved without any consideration of the nodes assigned to different shifts in distributed computing. Therefore, the method provides coarse-grained parallelism of computation. By employing a parallel linear solver for each shift point, the total number of nodes is the product of the number of nodes assigned for each linear system and the number of shift points.

The SS method has several parameters, and the choice of these parameters is crucial for achieving high accuracy and good parallel performance. In this chapter, we show some numerical properties of the method. The contour integral for a matrix inverse is regarded as a filter for an eigensubspace. When the contour integral is approximated by numerical quadrature, the quadrature error causes contamination of the eigencomponents corresponding to the eigenvalues located outside of the contour path. Based on these properties, we propose efficient parameter estimation techniques for the SS method.

In Chapter 4, a method for stochastic estimation of number of eigenvalues in a given domain is proposed. This estimation can be used for predicting appropriate parameters. Maeda, et al. [27] extended this eigenvalue count method to nonlinear eigenvalue problems.

The rest of this chapter is organized as follows. In Section 3.2, we briefly introduce the SS method. In Section 3.3, the properties of numerical quadrature applied for a matrix inverse are discussed. In Section 3.4, efficient parameter estimation methods are presented. Some numerical experiments are shown in Section 3.5. The last section concludes the chapter.

3.2 A contour integral based eigensolver

In this section, we briefly introduce the SS method. For matrices $A, B \in \mathbb{C}^{n\times n}$, let $\lambda_1, \dots, \lambda_n$ be eigenvalues of the matrix pencil $A - \lambda B$, and let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ be corresponding eigenvectors. Let Γ be a positively oriented closed Jordan curve in the complex plane, and let G be a domain for which the border is given by Γ. We will find the eigenvalues inside Γ and the corresponding eigenvectors by using contour integrals.

3.2.1 Eigensubspace obtained by contour integrals

Suppose that m eigenvalues $\lambda_1, \dots, \lambda_m$ are located inside Γ, and other eigenvalues are located outside Γ. Define a sequence of matrices $F_0, F_1, \dots$ as

$$F_k = \frac{1}{2\pi \mathrm{i}} \int_\Gamma z^k (zB - A)^{-1} B \, \mathrm{d}z, \quad k = 0, 1, \dots. \tag{3.2}$$

For a matrix $V \in \mathbb{R}^{n\times L}$ with a positive integer L, let

$$S_k = F_k V = \frac{1}{2\pi \mathrm{i}} \int_\Gamma z^k (zB - A)^{-1} B \, \mathrm{d}z \, V, \quad k = 0, \dots, M-1, \tag{3.3}$$

where M is chosen such that $LM \geq m$, and set

$$F = [F_0, F_1, \ldots, F_{M-1}]$$

and

$$S = [S_0, S_1, \ldots, S_{M-1}].$$

According to [39], the column vectors of S are given by linear combinations of the eigenvectors with respect to the eigenvalues located inside Γ, and thus

$$\mathrm{span}(S) = \mathrm{span}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m),$$

if the column space of V includes $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$. V is called a source matrix for the contour integral. In practice, the elements of V are set by a random number generator. The eigenvectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ are obtained from S when the maximum multiplicity of the eigenvalues in Γ is less than or equal to L.

Using the Rayleigh-Ritz procedure with S, we can extract the eigenpairs. Let the singular value decomposition of S be

$$S = U\Sigma W^{\mathrm{H}},$$

where $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_{LM})$, $U \in \mathbb{C}^{n\times LM}$ and $W \in \mathbb{C}^{LM\times LM}$. Since the rank of S is m, $\sigma_m \neq 0$ and $\sigma_{m+1} = \cdots = \sigma_{LM} = 0$. Setting $U_m = U(:, 1 : m)$, we calculate the projected matrices as

$$A_m = U_m^{\mathrm{H}} A U_m, \quad B_m = U_m^{\mathrm{H}} B U_m. \tag{3.4}$$

Let $\omega_1, \ldots, \omega_m$ be the eigenvalues of the matrix pencil $A_m - \lambda B_m$, and let $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m$ be the corresponding eigenvectors. Then the eigenvalues inside Γ of the matrix pencil $A - \lambda B$ are given by

$$\lambda_i = \omega_i, \quad i = 1, \ldots, m,$$

and the corresponding eigenvectors are given by

$$\boldsymbol{x}_j = U_m \boldsymbol{r}_j, \quad j = 1, \ldots, m. \tag{3.5}$$

When the matrices are large, storage of S and computation of the singular value decomposition restrict the application size of the method. The use of Hankel matrices reduces the memory requirement and computational costs. Let $\mathcal{M}_k \in \mathbb{C}^{L\times L}$ be

$$\mathcal{M}_k = \frac{1}{2\pi \mathrm{i}} \int_\Gamma z^k V^{\mathrm{T}} (zB - A)^{-1} B V \,\mathrm{d}z. \tag{3.6}$$

Let the Hankel matrices $H_{LM}, H_{LM}^{<} \in \mathbb{C}^{LM \times LM}$ be

$$H_{LM} = \begin{bmatrix} \mathcal{M}_0 & \mathcal{M}_1 & \cdots & \mathcal{M}_{M-1} \\ \mathcal{M}_1 & \mathcal{M}_2 & \cdots & \mathcal{M}_M \\ \vdots & \vdots & & \vdots \\ \mathcal{M}_{M-1} & \mathcal{M}_M & \cdots & \mathcal{M}_{2M-2} \end{bmatrix}$$

and

$$H_{LM}^{<} = \begin{bmatrix} \mathcal{M}_1 & \mathcal{M}_2 & \cdots & \mathcal{M}_M \\ \mathcal{M}_2 & \mathcal{M}_3 & \cdots & \mathcal{M}_{M+1} \\ \vdots & \vdots & & \vdots \\ \mathcal{M}_M & \mathcal{M}_{M+1} & \cdots & \mathcal{M}_{2M-1} \end{bmatrix}.$$

Let the singular value decomposition of $\tilde{H}_m$ be

$$\tilde{H}_m = \tilde{U}\tilde{\Sigma}\tilde{W}^{\mathrm{H}}, \tag{3.7}$$

where $\tilde{H}_m = H_{LM}(1:m, 1:m)$ and $\tilde{H}_m^{<} = H_{LM}^{<}(1:m, 1:m)$. Let $\tilde{\omega}_1, \ldots, \tilde{\omega}_m$ and $\tilde{\boldsymbol{q}}_1, \ldots, \tilde{\boldsymbol{q}}_m$ be the eigenvalues and the corresponding eigenvectors such that

$$(\tilde{\Sigma}^{-1}\tilde{U}^{\mathrm{H}}\tilde{H}_m^{<}\tilde{W})\tilde{\boldsymbol{q}}_i = \tilde{\omega}_i\tilde{\boldsymbol{q}}_i, \quad i = 1, \ldots, m.$$

Then the eigenvalues of the matrix pencil $A - \lambda B$ in Γ are given by

$$\lambda_i = \tilde{\omega}_i.$$

The eigenvectors are given by

$$\boldsymbol{x}_i = S(:, 1:m)\tilde{W}\boldsymbol{q}_i, \quad i = 1, \ldots, m.$$

In this computation, the singular value decomposition of S is not required. A disadvantage using the Hankel matrices with the moment matrices $\mathcal{M}_k$ is numerical instability comparing with the Rayleigh-Ritz procedure in the case of numerical computation with large m.

In the case of the nonlinear eigenvalue problem $T(\lambda)\boldsymbol{x} = \boldsymbol{0}$ with a matrix valued function $T(\lambda)$, the integrand $V^{\mathrm{T}}(zB - A)^{-1}BV$ in (3.6) is replaced by $V^{\mathrm{T}}T(z)^{-1}V$ [1, 2]. Note that the derived eigenvalue problem with Hankel matrices are linear even if the original problem is nonlinear. In [5], the integrand in the contour integral (3.6) is replaced by $T(z)^{-1}V$ instead of $V^{\mathrm{T}}T(z)^{-1}V$.

3.2.2 Approximation by a numerical quadrature

The contour integral in (3.2) is approximated by an N-point numerical quadrature. Suppose that a Jordan curve Γ is represented by scaling and shifting from a Jordan curve Γ_0 with a scaling factor ρ and a shift γ. Without any loss of generality, we assume that Γ_0 encloses the origin. Let $\zeta(\theta)$ be a point on Γ_0 with a parameter θ, $0 \leq \theta \leq 2\pi$, and let z on Γ be given by

$$z(\theta) = \gamma + \rho\zeta(\theta).$$

Then the contour integral of a function $f(z)$ is given by

$$\frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z) \mathrm{d}z = \frac{1}{2\pi} \int_0^{2\pi} f(z)(-\mathrm{i}\rho\zeta'(\theta)) \mathrm{d}\theta = \frac{1}{2\pi} \int_0^{2\pi} \rho f(z) w(\theta) \mathrm{d}\theta, \quad (3.8)$$

where $w(\theta) = -\mathrm{i}\zeta'(\theta)$. The integral (3.8) is approximated by the N-point quadrature rule

$$\frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z) \mathrm{d}z \approx \sum_{j=1}^{N} \rho w_j f(z_j), \quad (3.9)$$

where $w_j = w(\theta_j)\Delta_j/(2\pi)$, $\zeta_j = \zeta(\theta_j)$ and $z_j = \gamma + \rho\zeta_j$ with appropriate θ_j and Δ_j, $j = 1, \ldots, N$.

Since

$$\frac{1}{2\pi \mathrm{i}} \int_{\Gamma_0} \zeta^k \mathrm{d}\zeta = \begin{cases} 1, & k = -1 \\ 0, & \text{otherwise} \end{cases}$$

for integer k, the quadrature points $\zeta_1, \ldots, \zeta_N$ on Γ_0 and the corresponding weights $w_1, \ldots, w_N$ are set to satisfy

$$\sum_{j=1}^{N} w_j \zeta_j^k = \begin{cases} \nu \neq 0, & k = -1 \\ 0, & k = 0, \ldots, N-2 \end{cases}, \quad (3.10)$$

where ν is a nonzero constant.

In particular, when Γ is a circle with center γ and radius ρ, and the quadrature points are set as

$$z_j = \gamma + \rho(\cos\theta_j + \mathrm{i}\sin\theta_j), \quad j = 1, \ldots, N,$$

where $\theta_j = (2\pi/N) \times (j - 1/2)$, $j = 1, \ldots, N$, then Γ_0 is the unit circle and the quadrature weights are given by

$$w_j = \cos\theta_j + \mathrm{i}\sin\theta_j, \quad j = 1, \ldots, N.$$

In the case that all the eigenvalues are located on the real axis, it might be better to put the quadrature points closer to the real axis as follows:

$$z_j = \gamma + \rho(\cos\theta_j + \mathrm{i}\alpha\sin\theta_j), \quad j = 1, \ldots, N \tag{3.11}$$

with a vertical scaling factor $0 < \alpha < 1$. The corresponding quadrature weights are given by

$$w_j = \alpha\cos\theta_j + \mathrm{i}\sin\theta_j, \quad j = 1, \ldots, N. \tag{3.12}$$

In [33], quadrature points are set on straight lines to reuse solutions of linear systems. The Gauss-Legendre quadrature rule on a circle is used for the numerical quadrature in [35].

Using the quadrature rule (3.9), F_k and S_k are approximated by

$$F_k \approx \hat{F}_k = \sum_{j=1}^{N} \rho w_j \zeta_j^k (z_j B - A)^{-1} B \tag{3.13}$$

and

$$\hat{S}_k = \hat{F}_k V = \sum_{j=1}^{N} \rho w_j \zeta_j^k (z_j B - A)^{-1} BV. \tag{3.14}$$

Matrices F and S are approximated by $\hat{F} = [\hat{F}_0, \ldots, \hat{F}_{M-1}]$ and $\hat{S} = [\hat{S}_0, \ldots, \hat{S}_{M-1}]$.

The Rayleigh-Ritz procedure for $\hat{S}$ gives the approximate eigenvalues $\hat{\lambda}_i$ and the eigenvectors $\hat{\boldsymbol{x}}_i$. Let the singular value decomposition of $\hat{S}$ be

$$\hat{S} = \hat{U}\hat{\Sigma}\hat{W}^{\mathrm{H}},$$

where $\hat{\Sigma} = \mathrm{diag}(\hat{\sigma}_1, \ldots, \hat{\sigma}_{LM})$. Let K be the number of singular values of $\hat{S}$ that satisfy $\hat{\sigma}_i \geq \delta$, $1 \leq i \leq K$ with small $\delta > 0$. We calculate the projected matrices as

$$\hat{A} = \hat{U}(:, 1:K)^{\mathrm{H}}(A - \gamma B)\hat{U}(:, 1:K), \quad \hat{B} = \hat{U}(:, 1:K)^{\mathrm{H}} B\hat{U}(:, 1:K). \tag{3.15}$$

Let $\hat{\omega}_1, \ldots, \hat{\omega}_K$ be the eigenvalues of the matrix pencil $\hat{A} - \lambda\hat{B}$, and let $\hat{\boldsymbol{r}}_1, \ldots, \hat{\boldsymbol{r}}_K$ be the corresponding eigenvectors. Then the approximate eigenvalues inside Γ are given by

$$\hat{\lambda}_i = \gamma + \hat{\omega}_i, \quad i = 1, \ldots, K,$$

and the corresponding eigenvectors are given by

$$\hat{\boldsymbol{x}}_j = \hat{U}(:, 1:K)\hat{\boldsymbol{r}}_j, \quad j = 1, \ldots, K. \tag{3.16}$$

3.3 Filtering for a subspace

In this section, we discuss the properties of the subspace obtained by the numerical quadrature (3.14) from the view-point of a filter for a subspace.

Here, for simplicity, we consider the case that all the eigenvalues inside Γ are simple, and the inverse of the matrix $zB - A$ is expanded as

$$(zB - A)^{-1} = \sum_{i=1}^{n} \frac{\boldsymbol{x}_i \boldsymbol{y}_i^{\mathrm{H}}}{z - \lambda_i}, \tag{3.17}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are the right and left eigenvectors corresponding to the eigenvalue λ_i. This expansion can be generalized to the case of multiple eigenvalues and nonlinear problems ([2, 5, 39]).

Let $P_i = \boldsymbol{x}_i \boldsymbol{y}_i^{\mathrm{H}} B$, $1 \le i \le n$. With the expansion (3.17), from the residue theorem, we have

$$\begin{aligned} F_k &= \frac{1}{2\pi \mathrm{i}} \int_\Gamma z^k (zB - A)^{-1} B \mathrm{d}z \\ &= \sum_{i=1}^{n} \left(\frac{1}{2\pi \mathrm{i}} \int_\Gamma \frac{z^k P_i}{z - \lambda_i} \mathrm{d}z \right) \\ &= \sum_{i=1}^{m} \lambda_i^k P_i, \end{aligned}$$

and

$$S_k = F_k V = \sum_{i=1}^{m} \lambda_i^k P_i V.$$

Define a function $\mathcal{F}_k(\lambda)$ as

$$\mathcal{F}_k(\lambda) = \frac{1}{2\pi \mathrm{i}} \int_\Gamma \frac{z^k}{z - \lambda} \mathrm{d}z.$$

Then

$$\mathcal{F}_k(\lambda_i) = \begin{cases} \lambda_i^k & \lambda_i \in G \\ 0, & \text{otherwise} \end{cases},$$

and S_k is represented as

$$S_k = \sum_{i=1}^{n} \mathcal{F}_k(\lambda_i) P_i V.$$

This equation shows that a projected component associated with P_i in V is filtered with the factor $\mathcal{F}_k(\lambda_i)$. Therefore the function $\mathcal{F}_k(\lambda)$ is regarded to give the factor of filtering with respect to λ.

For the case that the contour integral is approximated by the numerical quadrature, we define the corresponding filter function by

$$\hat{\mathcal{F}}_k(\lambda) = \sum_{j=1}^{N} \frac{\rho w_j \zeta_j^k}{z_j - \lambda}.$$

The following result is obtained.

Theorem 3.1. *Let* λ *be a complex number that is located outside* Γ. *Then the following holds:*

$$\hat{\mathcal{F}}_k(\lambda) = -\nu_{N-1}\eta^{-N+k}\left(1 + \eta^{-1}\sum_{p=0}^{\infty}\frac{\nu_{N+p}}{\nu_{N-1}}\eta^{-p}\right), \tag{3.18}$$

where $\eta = (\lambda - \gamma)/\rho$ *and* $\nu_p = \sum_{j=1}^{N} w_j \zeta_j^p$.

Proof. Since $|\eta| = |(\lambda-\gamma)/\rho| > |\zeta_j|$ for $1 \le j \le N$, we have

$$\begin{aligned}
\sum_{j=1}^{N} \frac{\rho w_j \zeta_j^k}{z_j - \lambda} &= \sum_{j=1}^{N} \frac{w_j \zeta_j^k}{(z_j-\gamma)/\rho - (\lambda-\gamma)/\rho} = \sum_{j=1}^{N} \frac{w_j \zeta_j^k}{\zeta_j - \eta} \\
&= \sum_{j=1}^{N} \left(\frac{-1}{\eta}\right) \frac{w_j \zeta_j^k}{1 - \zeta_j/\eta} \\
&= -\sum_{p=0}^{\infty} \left(\eta^{-p-1} \sum_{j=1}^{N} w_j \zeta_j^{p+k}\right).
\end{aligned}$$

Since the quadrature weights $w_1, \ldots, w_N$ satisfy

$$\sum_{j=1}^{N} w_j \zeta_j^k = 0, \quad k = 0, \ldots, N-2,$$

we have

$$\begin{aligned}
\hat{\mathcal{F}}_k(\lambda) = \sum_{j=1}^{N} \frac{\rho w_j \zeta_j^k}{z_j - \lambda} &= -\sum_{p=N-1-k}^{\infty} \left(\eta^{-p-1} \sum_{j=1}^{N} w_j \zeta_j^{p+k}\right) \\
&= -\left(\nu_{N-1}\eta^{-N+k} + \sum_{p=0}^{\infty} \nu_{N+p}\eta^{-N+k-1-p}\right).
\end{aligned}$$

Thus we have (3.18). $\square$

If $|(\lambda-\gamma)/\rho|$ is sufficiently large then the filter $\hat{\mathcal{F}}_k(\lambda)$ is approximated by

$$\hat{\mathcal{F}}_k(\lambda) = \sum_{j=1}^{N} \frac{\rho w_j \zeta_j^k}{z_j - \lambda} \approx -\nu_{N-1} \left(\frac{\lambda - \gamma}{\rho} \right)^{-N+k}. \tag{3.19}$$

This implies that the eigencomponents corresponding to the eigenvalues located outside Γ in each column vector of $\hat{S}_k = \hat{F}_k V$ are reduced in proportion to the $(-N+k)$-th power of magnitude of the scaled distance $|(\lambda-\gamma)/\rho|$.

Suppose that the integer m' is taken as

$$\left| \nu_{N-1} \left(\frac{\lambda_i - \gamma}{\rho} \right)^{-N+M-1} \right| \leq \delta, \quad m' < i \leq n \tag{3.20}$$

with small $\delta > 0$. Then, from (3.18), we have

$$\hat{S}_k = \hat{F}_k V = \sum_{i=1}^{n} \hat{\mathcal{F}}_k(\lambda_i) P_i V = \sum_{i=1}^{m'} \hat{\mathcal{F}}_k(\lambda_i) P_i V + O(\delta).$$

3.4 Efficient parameter estimation and implementation

3.4.1 Selection of subspace size

The SS method has some parameters, and the choice of these parameters affects the accuracy and performance of the method. The number of quadrature points N determines the number of systems of linear equations to solve, and consequently N specifies the number of computing nodes to use in parallel computing. Therefore we assume that N is fixed in advance. In practice, N is chosen as $N = 16$ or 32 depending on the number of computing nodes or memory requirements, and it is not necessary to take a large N to reduce the quadrature error as was observed in the previous section.

The parameter M specifies the upper bound of the degree of moments. Increasing M gives a larger subspace size LM. However, the decay factor of the filter depends on $-N+k$ with $0 \leq k \leq M-1$, and a large M diminishes the performance of the filter. Considering a performance of the filer and computational costs, we set $M = N/4$.

The number of column vectors LM of $\hat{S}$ should be taken such that the minimum singular value of $\hat{S}$ becomes sufficiently small. Since M depends on N, we shall extend the number of column vectors of $\hat{S}$ by increasing the number of source vectors L. Since m' is larger than or equal to m, an approximation for m can be used as a lower bound of m'. To predict m, we can use the stochastic estimation method described in Chapter 4.

Using a stochastic estimation $\tilde{m}$ of m, we set the approximation of m' as $\kappa\tilde{m}$ with a parameter $\kappa \geq 1$, and consequently we set $L = \lceil m'/M \rceil \approx \lceil \kappa\tilde{m}/M \rceil$, where $\lceil x \rceil$ returns the smallest integer not less than x. When the subspace size LM is not sufficiently large, the minimum singular value $\sigma_{\min}$ of $\hat{S}$ is not small. In this case, we increment L until $\sigma_{\min}$ satisfies the condition $\sigma_{\min} \leq \delta \times \sigma_1$ with small $\delta > 0$. The computation of the singular values of $\hat{S}$ is rather expensive, so we may use the Hankel matrix $\hat{H}$ instead of $\hat{S}$.

3.4.2 Iterative refinement of a subspace

After setting appropriate L, we apply the Rayleigh-Ritz procedure with $\hat{S}$. The increase of L causes an increase in the size of the projected subspace. It causes an increase in the cost for computing the singular value decomposition of $\hat{S}$ and the solution of the projected eigenvalue problem with matrices $\hat{A}$ and $\hat{B}$. To avoid increasing the size of the projected space, we restrict the size of L, and apply the recurrence refinement described below.

Setting $\hat{S}_0^{(0)} = \hat{S}_0$, and recurrently applying $\hat{F}_0$, we have

$$\hat{S}_0^{(r-1)} = \hat{F}_0 \hat{S}_0^{(r-2)} = \cdots = (\hat{F}_0)^{r-1} \hat{S}_0^{(0)}. \tag{3.21}$$

Using $\hat{S}_0^{(r-1)}$, the output matrix with r refinements is given by

$$\hat{S}_k^{(r)} = \hat{F}_k \hat{S}_0^{(r-1)}, \quad k = 0, \ldots, M-1, \tag{3.22}$$

and $\hat{S}^{(r)} = [\hat{S}_0^{(r)}, \ldots, \hat{S}_{M-1}^{(r)}]$. The corresponding filter is given by $(\mathcal{F}_k(\lambda))^r$ and is approximated by

$$(\hat{\mathcal{F}}_k(\lambda))^r \approx (-\nu_{N-1})^r \left(\frac{\lambda - \gamma}{\rho} \right)^{-r(N-k)}.$$

Therefore the recurrence application of the filter process makes the decay factor of the filter smaller. The refinement is terminated if the smallest singular value of $\hat{S}^{(r)}$ becomes sufficiently small with a threshold $\delta > 0$.

In the case that some residuals of the obtained approximate eigenpairs are not small enough for a given tolerance, we can brush up the resulting approximate eigenpairs by setting the source matrix of the SS method as

$$V = [\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_{\hat{m}}]C,$$

where $C \in \mathbb{R}^{\hat{m} \times L}$ for which the elements are given by random numbers, and $\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_{\hat{m}}$ are the selected eigenvectors that are regarded as the approximate eigenvectors with respect to the eigenvalues inside Γ. This refinement technique using approximate eigenvectors for the source matrix V is used in [35].

3.4.3 Linear solvers for a complex shift

When A and B are real symmetric, the shifted matrix $C = zB - A$ with a complex shift z is complex symmetric. Therefore, a linear solver for complex symmetric systems is used to solve the system

$$(zB - A)Y = BV. \tag{3.23}$$

For a direct solver, the modified Cholesky factorization saves computational costs for factorization. For an iterative solver, Krylov subspace methods for complex symmetric systems, such as the COCG method, can be used.

When Γ is symmetric with respect to the real axis, the quadrature points are set as $z_{N-j+1} = \overline{z}_j$, $j = 1, \ldots, N/2$. Then, for real matrices A and B, the solutions at z_{N-j+1} are obtained by

$$Y_{N-j+1} = (z_{N-j+1}B - A)^{-1}BV = \overline{Y}_j$$

without any computations on z_{N-j+1}.

When A and B are Hermitian, we use the property

$$(z_j B - A)^{\mathrm{H}} = \overline{z}_j B^{\mathrm{H}} - A^{\mathrm{H}} = \overline{z}_j B - A.$$

If the LU factorization at z_j is calculated as $z_j B - A = LU$ then we have

$$Y_{N-j+1} = (\overline{z}_j B - A)^{-1}V = (U^{\mathrm{H}} L^{\mathrm{H}})^{-1}V.$$

Therefore the LU factorization at z_j can be used for the calculation at z_{N-j+1}.

Note that if the Hankel type method shown in Section 3.2.1 is used and only eigenvalues are required (eigenvectors are not required), the solution of the linear system (3.23) is demanded in the form of $V^{\mathrm{H}}Y$ rather than Y. The efficient method for directly computing $V^{\mathrm{H}}Y$ is described in Chapter 5.

3.5 Numerical experiments

In this section we show some numerical examples. The computations are performed in MATLAB 8.0.0. in double precision arithmetic. Random numbers are generated by the function `rand`, and the projected small eigenvalue problems are solved by `eig`. The systems of linear equations are solved by `lu`. The factorized matrices are held during the computation, and only triangular solves are applied in the recurrence refinements.

In the following examples, the quadrature points are set by (3.11) and the corresponding weights are set by (3.12) with $\alpha = 0.1$. The relative residual for the eigenpair $(\hat{\lambda}_i, \hat{\boldsymbol{x}}_i)$ is calculated by

$$\mathrm{res}_i = \frac{\|A\hat{\boldsymbol{x}}_i - \hat{\lambda} B\hat{\boldsymbol{x}}_i\|_2}{\|A\hat{\boldsymbol{x}}_i\|_2 + |\hat{\lambda}_i|\|B\hat{\boldsymbol{x}}_i\|_2}.$$

We removed the eigenvalues with $\mathrm{res}_i \geq 10^{-2}$ inside Γ as spurious eigenvalues.

Example 1. The matrices A and B are taken from BCSSTK11 and BCSSTM11 of the BCS Structural Engineering Matrices in Matrix Market [28]. A and B are real symmetric and B is positive definite. The matrix dimension is $n = 1,473$ with $34,241$ nonzero entries. The parameters are set as $N = 16$ and $L = 16$. The domain is set as $\gamma = 10^3$ and $\rho = 5 \times 10^2$. In this example, L is fixed, and the iterative refinement is not applied.

The results are shown in Table 5.3. The number of singular values that are greater than $\delta = 10^{-12}$ is $K = 18$. Therefore 18 eigenvalues are obtained from the projected problem, of which 7 eigenvalues are located inside Γ. The residuals of the eigenvalues located inside Γ are small, however the residuals of the eigenvalues located outside Γ are related to the scaled distance $|\eta_i| = |(\lambda_i - \gamma)/\rho|$.

Example 2. In this example, we apply the iterative refinement defined by (3.21) and (3.22). The matrices A and B are the same as in Example 1. The parameters are set as $N = 16$, $L = 16$, and the domain is set as $\gamma = 2 \times 10^5$ and $\rho = 2 \times 10^4$.

In Figure 1, The singular values of $\hat{S}^{(r)}$ at r-th refinement are shown. We can see that the ratio of the minimum singular value and the maximum singular value increases by the iterative refinement. After two refinements, the minimum singular value becomes small enough. Table 2 shows the residuals of the calculated eigenvalues located inside Γ. In the table, the notation $\mathrm{mean}(\mathrm{res}_i)$ is given by the geometric mean of the residuals defined by

$$\mathrm{mean}(\mathrm{res}_i) = \left(\prod_{i=1}^{\hat{m}} \mathrm{res}_i \right)^{1/\hat{m}},$$

where $\hat{m}$ is the number of calculated eigenvalues located inside Γ.

Example 3. In this example, we use the stochastic estimation of the number of eigenvalues in Γ to set the initial L, and the iterative refinement

Table 3.1: Results of Example 1.

i	$\hat{\lambda}_i$	res_i	$\lvert\eta_i\rvert$
1	2345.08723030540	3.0×10^{-01}	3.3
2	2398.81729572773	3.3×10^{-01}	3.2
3	2628.94468521146	4.9×10^{-02}	2.7
4	2723.54384863656	1.4×10^{-02}	2.6
5	3383.97540832681	3.8×10^{-08}	1.2
6	3501.25383608303	9.0×10^{-11}	–
7	3561.62085364923	2.7×10^{-11}	–
8	3629.33212408543	4.0×10^{-11}	–
9	3796.50112783802	4.8×10^{-11}	–
10	4022.39762561787	3.1×10^{-11}	–
11	4100.71462746484	1.5×10^{-11}	–
12	4175.86741050601	3.4×10^{-11}	–
13	4770.43635520514	5.1×10^{-06}	1.5
14	5071.04303115872	1.6×10^{-04}	2.1
15	5185.64239506030	3.5×10^{-03}	2.4
16	5325.06302301902	1.6×10^{-02}	2.7
17	5608.24863853754	1.0×10^{-01}	3.2
18	5874.78406307974	6.6×10^{-01}	3.8

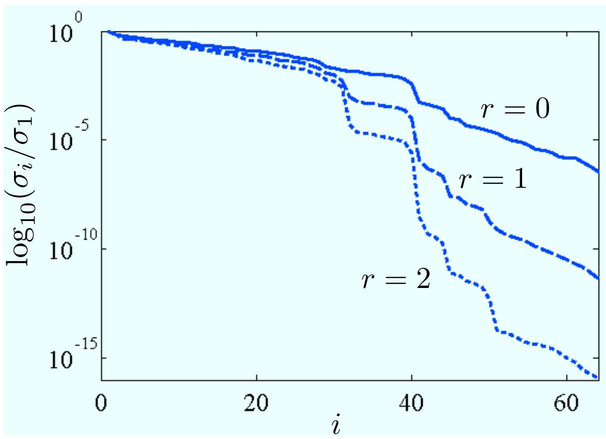

Figure 3.1: Singular values in r-th iterative refinement.

Table 3.2: Results in Example 2.

$\sharp$refinement	$\min(\mathrm{res}_i)$	$\mathrm{mean}(\mathrm{res}_i)$	$\max(\mathrm{res}_i)$
0	1.8×10^{-07}	9.1×10^{-06}	1.7×10^{-04}
1	3.8×10^{-12}	1.1×10^{-10}	1.1×10^{-09}
2	1.9×10^{-14}	7.2×10^{-13}	1.2×10^{-11}

of $\hat{S}$ is also used. The matrices A and B are the same as in Example 1. The parameters are set as $N = 16$ and $\delta = 10^{-12}$. The domain is set as $\gamma = 2\times 10^5$ and $\rho = 2 \times 10^4$. The number of sample vectors for the stochastic estimation of the number of eigenvalues in Γ is set as $L_0 = 16$. The initial guess of the number of column vectors of V is given by $L = \lceil 2\tilde{m}/M \rceil$, i.e. $\kappa = 2$.

In Table 3.3, we show the residuals of the eigenvalues located inside Γ. The number of eigenvalues in Γ is $m = 30$ and the estimated number of eigenvalues is $\tilde{m} = 32.7$. The number of iterative refinement is 2.

Example 4. The matrices A and B are taken from BCSSTK13 and BCSSTM13. A and B are real symmetric and B is positive semi-definite. The matrix dimension is $n = 2,003$ with $83,883$ nonzero entries. The parameters are the same as in Example 3. The domain is set as $\gamma = 10^6$ and $\rho = 4 \times 10^5$.

In Table 3.4, we show the residuals of the eigenvalues located inside Γ. The number of eigenvalues in Γ is $m = 73$ and the estimated number of eigenvalues is $\tilde{m} = 77.7$. The number of column vectors of V is $L = 55$ and the number of iterative refinement is 2. The maximum, mean and minimum residuals are 2.1×10^{-10}, 8.6×10^{-12} and 2.7×10^{-13}, respectively. We can obtain the eigenpairs in the given domain with the same initial parameters.

Example 5. The matrices A and B are derived from molecular orbital calculations for a model DNA [51]. A and B are real symmetric and B is positive definite. The matrix dimension is $n = 1,980$ with $728,080$ nonzero entries. The parameters are the same as in Example 3 and 4. The domains are given by the intervals $[-0.20, -0.15]$, $[-0.25, -0.15]$, $[-0.30, -0.15]$, $[-0.35, -0.15]$, $[-0.40, -0.15]$, $[-0.45, -0.15]$ and $[-0.50, -0.15]$.

In Table 3.5, we show the number of eigenvalues in the given interval ($\sharp$ev), the estimated number of eigenvalues (Est. $\sharp$ev), the number of column vectors of V (L), the number of iterative refinement ($\sharp$refinement) and the maximum residuals of eigenvalues in the interval ($\max(\mathrm{res}_i)$). In the results, the maximum residuals are sufficiently small by estimating appropriate L and the number of iterative refinement for each domain.

Table 3.3: Results in Example 3.

i	$\hat{\lambda}_i$	res_i	i	$\hat{\lambda}_i$	res_i
1	181301.355856	3.0×10^{-12}	16	206423.180896	2.2×10^{-12}
2	181353.297523	8.2×10^{-13}	17	207887.176182	4.5×10^{-12}
3	185810.063953	3.1×10^{-12}	18	209720.799807	1.2×10^{-12}
4	185856.309721	2.2×10^{-12}	19	211359.608331	1.6×10^{-12}
5	189076.069885	1.3×10^{-12}	20	211525.005509	1.2×10^{-12}
6	190580.274469	1.7×10^{-12}	21	211778.728062	1.0×10^{-12}
7	191916.768828	4.6×10^{-12}	22	211798.736010	1.4×10^{-12}
8	192249.997887	6.2×10^{-12}	23	214623.208612	1.7×10^{-12}
9	192450.352262	8.8×10^{-12}	24	215071.649241	1.2×10^{-12}
10	195110.875562	8.9×10^{-13}	25	216638.323804	1.1×10^{-12}
11	195362.147280	1.6×10^{-12}	26	216782.856683	5.0×10^{-13}
12	195522.864186	2.1×10^{-12}	27	216875.914785	4.9×10^{-13}
13	196453.465229	9.5×10^{-13}	28	217120.082795	1.4×10^{-12}
14	196779.318796	1.1×10^{-12}	29	217475.120411	1.3×10^{-13}
15	203358.448118	5.6×10^{-13}	30	217803.381541	5.8×10^{-13}

Table 3.4: Results in Example 4.

i	$\hat{\lambda}_i$	res_i	i	$\hat{\lambda}_i$	res_i
1	602514.527692	1.2×10^{-12}	38	964884.799128	2.7×10^{-11}
2	605178.148251	2.1×10^{-11}	39	971058.404128	3.0×10^{-11}
3	616657.672408	5.2×10^{-12}	40	973436.179279	9.5×10^{-12}
4	623758.141144	2.2×10^{-11}	41	981630.285398	3.0×10^{-11}
5	641859.031825	1.3×10^{-12}	42	985027.771304	6.4×10^{-11}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
33	924036.280859	7.0×10^{-11}	70	1332026.80482	7.5×10^{-12}
34	927854.750782	4.3×10^{-12}	71	1348423.99041	4.7×10^{-13}
35	941218.254886	9.8×10^{-12}	72	1372139.51897	8.4×10^{-12}
36	942132.221466	1.5×10^{-12}	73	1379152.51378	1.1×10^{-12}
37	960716.560772	1.0×10^{-11}			

Table 3.5: Results in Example 5.

Interval	$\sharp$ev	Est. $\sharp$ev	L	$\sharp$refinement	$\max(\mathrm{res}_i)$
$[-0.20, -0.15]$	22	23.9	16	1	2.8×10^{-13}
$[-0.25, -0.15]$	78	80.0	40	2	2.1×10^{-12}
$[-0.35, -0.15]$	198	196.3	99	2	8.5×10^{-12}
$[-0.40, -0.15]$	262	270.1	136	2	1.7×10^{-12}
$[-0.45, -0.15]$	333	327.9	164	2	9.0×10^{-12}
$[-0.50, -0.15]$	406	410.5	206	2	9.4×10^{-12}

3.6 Concluding remark

In this chapter, we have considered an eigensolver for computing the eigenvalues in a given domain and the corresponding eigenvectors of large-scale matrix pencils. The Sakurai-Sugiura (SS) method is an eigensolver based on complex moments given by the contour integrals of the matrix inverses with several shift points.

Some numerical properties of the method have been presented from the view-point of a filter for a subspace. According to the results, efficient parameter estimation techniques have been shown. The contour integral for a matrix inverse is regarded as a filter for an eigensubspace. When the contour integral is approximated by a numerical quadrature, the quadrature error causes contamination of the eigencomponents corresponding to the eigenvalues located outside of the contour path. We have demonstrated the efficiency of our method with numerical experiments.

In the numerical experiments, we have used a sparse direct solver. The use of iterative linear solvers for multiple right-hand sides such as block Krylov subspace solvers are useful because our eigensolver requires very small number of iterative refinement.

We acknowledge here that a part of the study in this chapter is published as [60] in the list of publications.

Chapter 4

Parallel stochastic estimation method of eigenvalue distribution

4.1 Introduction

As described in Section 3.4.1 in Chapter 3, an estimation of eigenvalue count is needed for selection of subspace for the SS method. This information is also valuable for other eigensolvers such as the Arnoldi method with the shift-and-invert spectral transformation (SI-Arnoldi) and the Jacobi-Davidson method (JD). In addition, if one employs multiple contour paths for the SS method or multiple shifts for SI-Arnoldi/JD, a (rough) distribution of a eigenvalues is demanded for efficient setting of contour paths or shifts.

To compute eigenvalue distribution, some methods have been proposed, including the method using Sylvester's law of inertia and the algebraic substructure method [41]. Both methods require a matrix factorization, such as the $\mathrm{LDL}^{\mathrm{T}}$ factorization. However, it is not feasible to apply these method to large sparse matrices or matrices that are only referenced in the form of matrix-vector multiplications. In this chapter, we propose a stochastic estimation method of the eigenvalue distribution that is based on a stochastic estimator of the matrix trace. We evaluate the performance of the proposed method by applying it to matrices from practical applications.

This chapter is organized as follows. In Section 4.2, a stochastic estimator of an eigenvalue distribution and its parallelization are described. We show a simple implementation of our method in Section 4.3. In Section 4.4, we investigate the performance of our method through numerical experiments with four matrices from Matrix Market [28] and a matrix derived from a real-space density functional calculation. This is followed by the concluding remarks in Section 4.5.

4.2 A stochastic estimator of eigenvalue distribution

4.2.1 A stochastic estimator of eigenvalue count

Let $A, B \in \mathbb{C}^{n\times n}$, $z \in \mathbb{C}$ be such that $(zB - A)$ is a regular matrix pencil. It is known that matrices A, B can be decomposed $A = URV^{\mathrm{H}}, B = UTV^{\mathrm{H}}$, where R, T are upper triangular matrices whose diagonal elements are r_{jj}, t_{jj}, respectively, and U, V are unitary matrices. Since

$$(zB - A)^{-1}B = V(zT - R)^{-1}TV^{\mathrm{H}},$$

and the matrix trace is similarity-invariant,

$$\begin{aligned}\mathrm{tr}((zB - A)^{-1}B) &= \mathrm{tr}((zT - R)^{-1}T)\\ &= \sum_{j=1}^{n} \frac{t_{jj}}{zt_{jj} - r_{jj}}\\ &= \sum_{j=1}^{n'} \frac{1}{z - \lambda_j},\end{aligned} \tag{4.1}$$

where

$$t_{jj} \begin{cases} \neq 0 & (1 \leq j \leq n') \\ = 0 & (n' + 1 \leq j \leq n) \end{cases},$$

and $\lambda_j = r_{jj}/t_{jj}$ $(j = 1, 2, \ldots, n')$ are finite eigenvalues of the matrix pencil (A, B).

When the contour integration

$$\begin{aligned}\mu &= \frac{1}{2\pi\mathrm{i}} \oint_\Gamma \mathrm{tr}((zB - A)^{-1}B)\mathrm{d}z\\ &= \frac{1}{2\pi\mathrm{i}} \oint_\Gamma \sum_{j=1}^{n'} \frac{1}{z - \lambda_j}\mathrm{d}z\end{aligned} \tag{4.2}$$

is performed, the eigenvalue count μ in a positively oriented Jordan curve Γ is derived by the residue theorem. To discretize (4.2), an N-point quadrature rule is applied and we approximate μ by

$$\mu \approx \hat{\mu} = \sum_{k=0}^{N-1} w_k \operatorname{tr}((z_k B - A)^{-1} B), \tag{4.3}$$

where z_j and w_j are a quadrature point and a weight, respectively. In the case of the trapezoidal rule on a circle with a center γ and a radius ρ, quadrature points and weights are defined by

$$z_k = \gamma + \rho e^{\frac{2\pi \mathrm{i}}{N}(k+1/2)} \quad k = 0, 1, \ldots, N-1,$$

and

$$w_k = \frac{z_k - \gamma}{N} \quad k = 0, 1, \ldots, N-1,$$

respectively, where i is the imaginary unit. According to [38], when the contour path is a circle, (4.3) is written as

$$\hat{\mu} = \sum_{j=1}^{n'} \frac{1}{1 + \left(\frac{\gamma - \lambda_j}{\rho}\right)^N}, \tag{4.4}$$

where $|\frac{\gamma-\lambda_1}{\rho}| \leq |\frac{\gamma-\lambda_2}{\rho}| \leq \cdots \leq |\frac{\gamma-\lambda_{n'}}{\rho}|$. Let m' be an integer such that $\rho/(1 + (\frac{\gamma-\lambda_j}{\rho})^N) = O(\varepsilon)$ for any j with $m' < j \leq n'$ for sufficiently small $\varepsilon > 0$. Then (4.4) can be expressed as

$$\hat{\mu} = \sum_{j=1}^{m'} \frac{1}{1 + \left(\frac{\gamma - \lambda_j}{\rho}\right)^N} + O(\varepsilon). \tag{4.5}$$

Thus, the eigenvalues that exist nearby and outside of Γ are attributed to quadrature error.

According to [4, 22], an unbiased estimation of the matrix trace is given by

$$\operatorname{tr}((z_k B - A)^{-1} B) \approx \frac{1}{s} \sum_{j=1}^{s} \boldsymbol{v}_j{}^{\mathrm{T}} (z_k B - A)^{-1} B \boldsymbol{v}_j, \tag{4.6}$$

where s is the number of sample vectors and $\boldsymbol{v}_j$ are vectors whose entries take 1 or -1 with equal probability. Using (4.6), one can estimate $\hat{\mu}$ as

$$\begin{aligned} \hat{\mu} &\approx \tilde{\mu} \\ &= \frac{1}{s} \sum_{k=0}^{N-1} w_k \sum_{j=1}^{s} (\boldsymbol{v}_j{}^{\mathrm{T}} (z_k B - A)^{-1} B \boldsymbol{v}_j). \end{aligned} \tag{4.7}$$

4.2.2 Solution for linear systems

The most time consuming part of the estimation of the trace of $(z_k B - A)^{-1}B$ is the solution of s independent linear systems

$$(z_k B - A)\boldsymbol{x}_j^k = B\boldsymbol{v}_j \begin{cases} j = 1, 2, \ldots, s \\ k = 0, 1, \ldots, N-1 \end{cases} . \tag{4.8}$$

The subscript of $\boldsymbol{x}_j^k$ refers the sample vector $\boldsymbol{v}_j$ and the superscript refers the quadrature point z_k. If the matrices A and B are large sparse matrices or they are only referenced in the form of matrix-vector multiplications, an iterative method is a reasonable choice to solve these linear systems. Additionally, if B is the identity matrix I, the linear systems (4.8) are written as $(z_k I - A)\boldsymbol{x}_j^k = \boldsymbol{v}_j$. In this case, the shifted Krylov subspace method [26, 13] can be applied to solve simultaneously the linear systems $(z_k I - A)\boldsymbol{x}_j^k = \boldsymbol{v}_j$ for the scalar parameters z_k. By using the shifted Krylov subspace method, the total number of matrix-vector multiplications in each iteration is reduced to $1/N$ that of solving N systems separately by the normal Krylov subspace method. When A is a real symmetric matrix, $(z_k I - A)$ is a complex symmetric (but not Hermitian) matrix. The shifted conjugate orthogonal conjugate gradient (COCG) method [50, 52] is a reasonable choice to solve linear systems of complex symmetric matrices.

Note that in our method, the solution of the linear system (3.23) is needed in the form of $\boldsymbol{v}_j^{\mathrm{H}}\boldsymbol{x}_j^k$ rather than $\boldsymbol{x}_j^k$ $(j = 1, 2, \ldots, s)$. The efficient method for directly computing $\boldsymbol{v}_j^{\mathrm{H}}\boldsymbol{x}_j^k$ is described in Chapter 5.

4.2.3 Method for estimating eigenvalue distribution

A stochastic estimation method of the eigenvalue distribution is defined by the estimator of the eigenvalue count straightforwardly. Let Γ be a given Jordan curve, D the domain closed by Γ, and $\Gamma_\ell(\ell = 1, 2, \ldots, n_{\mathrm{c}})$ a Jordan curve which closes sub-domain D_ℓ such that $D = D_1 + D_2 + \cdots + D_{n_{\mathrm{c}}}$. It is easy to see that the estimations of the eigenvalue count in Γ_ℓ can be executed independently. Below this independence, there is another independence: that of the solutions of the linear systems (4.8). Furthermore, the linear solver can be parallelized, if it is possible. Thus, our method is efficient on modern massively parallel computing environments.

4.3 Implementation

In this section, we describe a simple implementation of our method in which A is a Hermitian matrix and B is a non-singular Hermitian matrix. The algorithm of the implementation is shown in Algorithm 4.1. For simplicity, we assume the Jordan curves are circles. This algorithm estimates the eigenvalue distribution in the interval $[\alpha, \beta]$ on the real axis. n_c circles are placed so that each circle occupies an equally separated sub-interval. ρ is the radius of all circles and γ_ℓ is the center of the ℓth circle. $\tilde{\mu}_\ell$ is the estimated eigenvalue count in the ℓth circle. The same number of quadrature points N is set for each circle.

Algorithm 4.1 Stochastic estimation method for eigenvalue distribution.

Input : $A, B, \alpha, \beta, n_\mathrm{c}, N, s$
Output : $\tilde{\mu}_1, \tilde{\mu}_2, \ldots, \tilde{\mu}_{n_\mathrm{c}}$
Set $\boldsymbol{v}_j$ whose elements take 1 or -1 with equal probability, for $j = 1, 2, \ldots, s$
$\rho = (\beta - \alpha)/2n_\mathrm{c}$
for $\ell = 1, 2, \ldots, n_\mathrm{c}$ **do**
    $\gamma_\ell = \alpha + (2\ell - 1)\rho$
    $z_{\ell k} = \gamma_\ell + \rho e^{\frac{2\pi \mathrm{i}}{N}(k+1/2)}$
    Solve $(z_{\ell k} B - A)\boldsymbol{x}_j^{\ell k} = B\boldsymbol{v}_j$, for $j = 1, 2, \ldots, s, \quad k = 0, 1, \ldots, N-1$
    $\tilde{\mu}_\ell = \dfrac{\rho}{sN} \sum_{k=0}^{N-1} e^{\frac{2\pi \mathrm{i}}{N}(k+1/2)} \sum_{j=1}^{s} \boldsymbol{v}_j{}^\mathrm{T} \boldsymbol{x}_j^{\ell k}$
end for

Table 4.1: Matrix properties.

Matrix pencil	*Size*	*nnz(A)*	*nnz(B)*	*Type(A)*	*Type(B)*	*Center*	*Radius*	*#eig in* Γ
LUND	147	1298	1294	Indefinite	Indefinite	1.0×10^4	1.0×10^4	40
BCSST07	420	4140	3836	Positive definite	Positive semi-definite	0.23	0.17	398
PLAT1919	1919	17159	—	Indefinite	—	2.0×10^7	2.5×10^7	40
BCSST13	2003	42943	11973	Positive definite	Positive semi-definite	3.0×10^3	2.0×10^3	11

4.4 Numerical experiments

In this section, we perform numerical experiments to evaluate the efficiency of our method by using the algorithm shown in Algorithm 4.1. Examples 1 and 2 are carried out using Matlab 7.4, and Example 3 is carried out using PGI Fortran 90. All operations are done in double precision arithmetic.

4.4.1 Example 1

In Example 1, we investigate how the eigenvalue count changes for an increase in the number of quadrature points N. We evaluate the effect of numerical integrations (4.3) on the eigenvalue count without trace estimations. The exact value of the matrix trace is calculated using the relation described in (4.1). The eigenvalues λ_j are obtained by Matlab function `eig`. The test problems were taken from Matrix Market; their properties are shown in Table 4.1. All eigenvalue problems are that of real symmetric matrices. We set $n_c = 1$ for the algorithm. Columns *nnz(A)* and *nnz(B)* show the number of non-zero entries of matrices A and B, respectively. Columns *Type(A)* and *Type(B)* show the properties of A and B. Columns *Center* and *Radius* show the center and radius of the circles, respectively. The column *#eig in* Γ shows the number of eigenvalues in Γ. The number of eigenvalues is calculated by using the results of `eig`. The number of quadrature points N is set to be 4, 8, 16, 32, and 64. The results of this example are shown in Table 4.2. All results converge to the exact values.

4.4.2 Example 2

In Example 2, we investigate how the eigenvalue count changes for an increase in the number of sample vectors s. The test matrices used are the same as those in Example 1, s is set to from 10 to 1000, n_c is set to 1, and the linear systems are solved using the Matlab function `mldivide`. The number of quadrature points is set to $N = 16$. The elements of the sample vectors are given by the Matlab function `rand`, and their random seed is set by `rand('twister', 5489)`. The results of this example are shown in Table 3.We consider the exact eigenvalue count $\hat{\mu}$ to be that shown for the $N = 16$ case in Table 4.2. Increasing s does not much effect the efficiency or accuracy of the eigenvalue count, even though it increases the computational cost. The trace estimation is slow in converging to the exact value because the convergence rate is $O(\sqrt{s})$. Similar results on trace estimations are shown in [4].

Table 4.2: Results for Example 1.

N	*eigenvalue count*			
	LUND	BCSST07	PLAT1919	BCSST13
4	38.024	318.03	55.559	10.917
8	38.268	364.80	42.350	10.926
16	38.880	392.98	40.606	10.988
32	39.373	397.89	39.945	11.000
64	39.749	398.00	39.540	11.000
exact	40.000	398.00	40.000	11.000

Table 4.3: Results for Example 2.

#vectors	*eigenvalue count*			
	LUND	BCSST07	PLAT1919	BCSST13
10	44.344	391.08	40.759	12.866
20	43.394	392.58	40.371	11.747
30	43.195	391.92	40.926	10.765
40	39.547	393.83	39.874	10.590
50	40.039	393.09	41.018	10.313
100	37.716	392.27	40.632	11.293
200	39.805	393.45	40.341	11.460
500	41.147	392.76	40.542	11.104
1000	39.874	392.53	40.731	11.229
exact	38.880	392.98	40.606	10.988

4.4.3 Example 3

In Example 3, the test matrix is derived from real-space density functional calculations [25]. It is a standard eigenvalue problem $A\boldsymbol{x} = \lambda\boldsymbol{x}$, where A is a real symmetric matrix and is only referenced in the form of matrix-vector multiplications. Thus, applying conventional approaches mentioned in Section 4.1 is not feasible in this case. In this problem, the M_{B} smallest eigenvalues are desired, where M_{B} is the total number of orbitals. The test matrix is derived from the density functional calculation of a 510-atom system of silicon. The matrix size is $n = 175,616$, and the smallest 1,020 eigenpairs are desired. The linear systems are solved by the shifted COCG method using stopping criterion 10^{-4}. One hundred circles are placed in the interval

$[-0.230, 0.243]$. The number of quadrature points of each circle is $N = 8$, and the number of sample vectors is $s = 20$. The results are shown in Figure 4.1. The horizontal axis indicates the index of the circles, and the vertical axis indicates the eigenvalue count for the circle's sub-domain. The exact values are calculated by the conjugate gradient method for eigenvalue problems [25]. Although s is significantly smaller than the matrix size n, our method roughly estimates the eigenvalue count. We obtained a rough eigenvalue distribution that can be used in setting parameters for an accurate eigensolver using only a few quadrature points and sample vectors.

The computational cost of the conjugate gradient method for eigenvalue problems is $O({M_{\mathrm{B}}}^3)$ (see [25]). We confirmed that the number of iteration of the shifted COCG method is proportional to n in preliminary experiments. The cost of the matrix-vector multiplication is $O(n)$ due to the sparsity of the matrix. Therefore, when s is set much less than n and the scalar recurrences are introduced to the shifted COCG method, the computational cost of our method is $O(n^2)$. Since n, the number of grid points, is set to be proportional to M_{B}, for example $n \approx 200M_{\mathrm{B}}$, the cost of our method is $O({M_{\mathrm{B}}}^2)$ with a large coefficient. When the number of atoms in the target system is large, our method can be employed as a preprocessing of accurate eigensolvers, due to the lower order of computational cost and the high parallel performance.

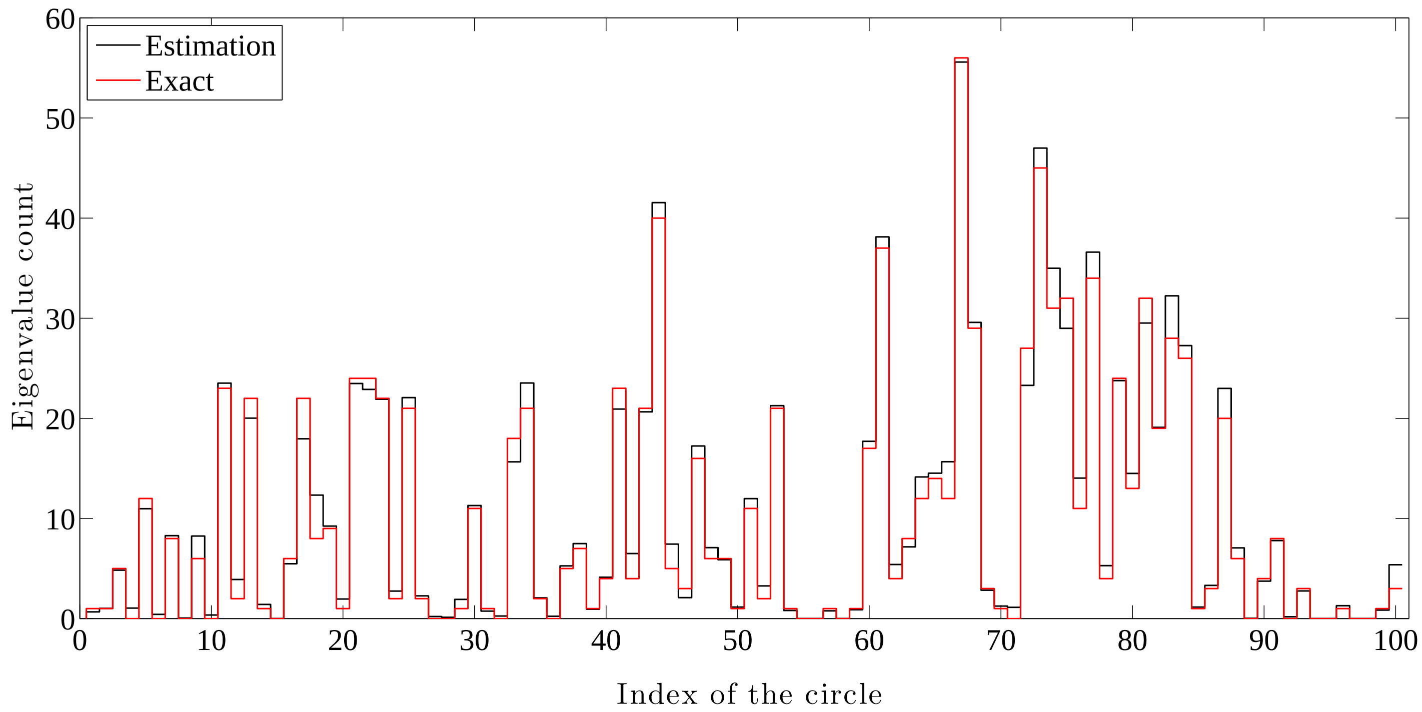

Figure 4.1: Eigenvalue distribution of a 510-atom system of silicon.

4.5 Concluding remarks

In this chapter, we have proposed a stochastic estimation method of eigenvalue counting within a given closed curve. Our method is feasible for large sparse matrices or matrices that are only referenced in the form of matrix-vector multiplication. The stochastic estimation method for the eigenvalue distribution is defined by separating the given domain to several sub-domains and estimating the eigenvalue count in each sub-domain. Furthermore, because the computation of our method has independence, it is easy to execute on massively parallel computing environments. An acceleration technique has been introduced to standard eigenvalue problems by using the shifted Krylov subspace method. We have shown using numerical examples that our method roughly estimates the eigenvalue distribution using only a few quadrature points and sample vectors. The parameters of eigensolvers can be effectively set by using a given knowledge of the eigenvalue distribution, and this distribution need not to be accurate, but does need to be computed at low cost. Our method is effective in such situations.

We acknowledge here that a part of the study in this chapter is published as [57] in the list of publications.

Chapter 5

Block conjugate gradient type methods for the approximation of bilinear form $C^{\mathrm{H}}A^{-1}B$

5.1 Introduction

In the SS method and the stochastic estimation method for eigenvalue count, one need to solve linear systems

$$(zB - A)\boldsymbol{x}_i = \boldsymbol{w}_i \tag{5.1}$$

where $A, B \in \mathbb{C}^{n\times n}$, $z \in \mathbb{C}$ such that $\det(zB - A)) \neq 0$, $\boldsymbol{v}_i, \boldsymbol{x}_i \in \mathbb{C}^n$ and $\boldsymbol{w}_i \equiv B\boldsymbol{v}_i (i = 1, 2, \dots m)$. (5.1) can be represented as

$$(zB - A)X = W$$

with $X \equiv [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_m]$ and $W \equiv [\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_m]$. In some case of the SS method and also in the stochastic estimation method for eigenvalue count, the solutions is demanded in the form of $V^{\mathrm{H}}X$ rather than X, where $V \equiv [\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_m]$ (see Section 3.4.3 and Section 4.2.2). Thus, in such case, what we actually need to do is to compute

$$V^{\mathrm{H}}(zB - A)^{-1}W. \tag{5.2}$$

In this chapter, we consider to compute an approximation of

$$C^{\mathrm{H}}A^{-1}B, \tag{5.3}$$

where $C \in \mathbb{C}^{n\times m}$, here we redefine $A \in \mathbb{C}^{n\times n}$ as the coefficient matrix and $B \in \mathbb{C}^{n\times m}$ as the matrix whose columns are right hand side vectors.

The need to approximate the block bilinear form (5.3) arises not only in the above examples but also in a rich variety of fields in science and engineering such as computational fluid dynamics, inverse problems; see [4, 17] and the references therein. In the specific case that $m = 1$, efficient methods based on the conjugate gradient (CG) method and the BiCG method for approximating the scalar $\boldsymbol{c}^H A^{-1}\boldsymbol{b}$ ($\boldsymbol{b}, \boldsymbol{c} \in \mathbb{C}^n$) have been discussed in [45, 46] and in [47], respectively. Both methods do not need explicitly compute and store the approximate solution $\boldsymbol{x}_k$ of $A\boldsymbol{x} = \boldsymbol{b}$. Despite the fact that the kth step approximation of $\boldsymbol{c}^{\mathrm{H}} A^{-1}\boldsymbol{b}$ using methods in [45, 46, 47] is mathematically identical with the corresponding $\boldsymbol{c}^{\mathrm{H}}\boldsymbol{x}_k$, where $\boldsymbol{x}_k$ is obtained by CG or BiCG, numerical results have illustrated that those methods in [45, 46, 47] can be more stable and accurate. For the estimation of general form $\boldsymbol{u}^{\mathrm{T}} f(A)\boldsymbol{v}$, where $f(\cdot)$ is a smooth function, algorithms based on the look-ahead Lanczos and the Arnoldi process were developed in [18]. Motivated by those methods in [45, 46, 47], we propose block conjugate gradient type methods for (5.3). Since $m > 1$, we know that one class of iterative methods for solving linear systems with multiple right-hand sides is the block Krylov subspace methods, which include block (Bi)CG [34], block GMRES [42], block QMR [14], block BiCGSTAB [11] and block IDR(s) [9], etc. Therefore it is natural to generalize the results in [45, 46, 47] to the block Krylov subspace methods. In this chapter, we develop numerical methods based on block CG and block BiCG for the approximation of (5.3). As the block CG method is usually used for solving linear systems whose coefficient matrix is Hermitian and positive-definite (HPD), when matrix A is HPD, we will limit the considered problem (5.3) to the particular case that matrix C is identical to matrix B. We mention here that both bilinear and block bilinear forms have been discussed in [16] based on the use of quadrature rules.

This chapter is organized as follows. In Section 5.2, we describe the block Krylov subspace methods and recall the block BiCG and block CG methods, then we propose methods based on block BiCG and block CG for (5.3). In Section 5.3, we present a variant of the block BiCG method by orthogonalizing the residual matrices and give alternative ways to approximate (5.3). In Section 5.4, we report some numerical results to compare our proposed methods with block solvers. Finally, some concluding remarks are made in Section 5.5.

5.2 Block Conjugate Gradient type methods

In this section, we firstly review some fundamental knowledge of block Krylov subspace methods. Then we present two methods for the approximation of

$C^{\mathrm{H}}A^{-1}B$. The first method derived from the block BiCG method is suitable for general problem. The second method based on the block CG method is a specific case of the first method and will be applied to $B^{\mathrm{H}}A^{-1}B$ with an Hermitian matrix A.

5.2.1 Block Krylov subspace methods

In this subsection, some fundamental knowledge of block Krylov subspace methods is recalled. For more details, please refer to [19, 20].

Definition 5.1. *Let $U \in \mathbb{C}^{n\times m}$, the subspace $\mathcal{K}_k(A,U)$ generated by A and increasing powers of A applied to U*

$$\mathcal{K}_k(A,U) \equiv \left\{ \sum_{i=0}^{k-1} A^i U \gamma_i ; \gamma_i \in \mathbb{C}^{m\times m} \right\} \tag{5.4}$$

is called the kth-order block Krylov subspace.

When $m = 1$, the matrix U is reduced to a vector, subspace (5.4) becomes a standard Krylov subspace. For solving linear systems with multiple right-hand sides $AX = B$, when initial guess X_0 and the corresponding matrix residual $R_0 \equiv B - AX_0$ are given, all block Krylov subspace methods compute approximate solutions in the framework of $X_k = X_0 + Z_k$ where $Z_k \in \mathcal{K}_k(A, R_0)$. From the definition of (5.4), there are γ_j's $\in \mathbb{C}^{\mathrm{m}\times\mathrm{m}}$ $(j = 0, \ldots, k-1)$ that satisfy

$$Z_k = \sum_{j=0}^{k-1} A^j R_0 \gamma_j. \tag{5.5}$$

If we expand equation (5.5), each column of Z_k can be represented as

$$\boldsymbol{z}_k^{(i)} = \sum_{l=1}^{m} \sum_{j=0}^{k-1} \gamma_j(l,i) A^j \boldsymbol{r}_0^{(l)} \in \mathcal{B}_k(A, R_0), \quad i = 1, \ldots, m,$$

where

$$\mathcal{B}_k(A, R_0) \equiv \mathcal{K}_k(A, \boldsymbol{r}_0^{(1)}) + \cdots + \mathcal{K}_k(A, \boldsymbol{r}_0^{(m)}). \tag{5.6}$$

The corresponding approximate solutions $\boldsymbol{x}_k^{(i)}$ can be written as $\boldsymbol{x}_k^{(i)} = \boldsymbol{x}_0^{(i)} + \boldsymbol{z}_k^{(i)}$. Compared with traditional Krylov subspace methods, where $\boldsymbol{x}_k^{(i)} - \boldsymbol{x}_0^{(i)} \in \mathcal{K}_k(A, \boldsymbol{r}_0^{(i)})$, we see that block Krylov subspace methods could search approximate solutions in a bigger search space than the traditional Krylov subspace methods at the same iteration step. This implies that block Krylov subspace methods may find an approximation within less iterations.

Similar to the Krylov subspace, the grade of the block Krylov subspace has been defined in [19, 40]. We recall it as follows:

Definition 5.2. *[19, 40]The positive integer $v \equiv v(A, U)$ defined by*

$$\begin{aligned} v(A, U) &\equiv \min\{k | \dim\, \mathcal{B}_k(A, U) = \dim\, \mathcal{B}_{k+1}(A, U)\} \\ &= \min\{k | \mathcal{B}_k(A, U) = \mathcal{B}_{k+1}(A, U)\} \end{aligned}$$

is called block grade of U with respect to A.

The following corollary shows the relationship between the block grade of block Krylov subspace and the exact solution of linear systems with multiple right-hand sides.

Corollary 5.1. *[19] Let X_* be the exact solution of $AX = B$, for any initial block guess X_0 and with its corresponding block residual R_0, it always holds*

$$X_* \in X_0 + \mathcal{K}_{v(A,R_0)}(A, R_0).$$

Algorithm 5.1 Block bi-conjugate gradient (Block BiCG) [34]

1: Given X_0, compute $R_0 = B - AX_0$, let $P_0 = R_0$
and $\tilde{P}_0 = \tilde{R}_0$, where $\tilde{R}_0$ can be an arbitrary $n \times m$ matrix;
2: **for** $k = 0, 1, \ldots,$ **do**
3: $\quad \alpha_k = (\tilde{P}_k^{\mathrm{H}} A P_k)^{-1} (\tilde{R}_k^{\mathrm{H}} R_k)$; $\tilde{\alpha}_k = (P_k^{\mathrm{H}} A^{\mathrm{H}} \tilde{P}_k)^{-1} (R_k^{\mathrm{H}} \tilde{R}_k)$;
4: $\quad X_{k+1} = X_k + P_k \alpha_k$;
5: $\quad R_{k+1} = R_k - A P_k \alpha_k$; $\tilde{R}_{k+1} = \tilde{R}_k - A^{\mathrm{H}} \tilde{P}_k \tilde{\alpha}_k$;
6: $\quad \beta_k = (\tilde{R}_k^{\mathrm{H}} R_k)^{-1} (\tilde{R}_{k+1}^{\mathrm{H}} R_{k+1})$; $\tilde{\beta}_k = (R_k^{\mathrm{H}} \tilde{R}_k)^{-1} (R_{k+1}^{\mathrm{H}} \tilde{R}_{k+1})$;
7: $\quad P_{k+1} = R_{k+1} + P_k \beta_k$; $\tilde{P}_{k+1} = \tilde{R}_{k+1} + \tilde{P}_k \tilde{\beta}_k$;
8: **end for**

Here we recall the block BiCG algorithm in Algorithm 5.1, which is an extension of the BiCG [12] method for solving linear systems with multiple right-hand sides. Some orthogonality properties of the block BiCG algorithm hold as follows:

$$R_k^{\mathrm{H}} \tilde{P}_j = 0, \; j = 0, 1, \ldots, k-1, \tag{5.7}$$

$$\tilde{R}_k^{\mathrm{H}} P_j = 0, \; j = 0, 1, \ldots, k-1. \tag{5.8}$$

When matrix A is Hermitian, i.e., $A^{\mathrm{H}} = A$, we can compute the approximate solution of $AX = B$ more effectively using a specific implementation of the block BiCG method, the so-called block CG method, with properly choosing $\tilde{R}_0 = R_0$ in Algorithm 5.1.

5.2.2 Block Conjugate Gradient type methods for $C^{\mathrm{H}}A^{-1}B$

In this subsection, we will describe two methods for approximating $C^{\mathrm{H}}A^{-1}B$ based on the block CG and block BiCG methods, which are generalized the corresponding methods in [45, 46, 47]. Since the method based on block CG could also be regarded as a special implementation of the block BiCG-based method that applied to $B^{\mathrm{H}}A^{-1}B$ with Hermitian matrix A, here we first discuss the derivation process of the block BiCG-based method.

If we set $X_0 = 0$ and $\tilde{R}_0 = C$ in the initial step of Algorithm 5.1, the following relationship always holds

$$C^{\mathrm{H}}A^{-1}B - \tilde{R}_{k+1}^{\mathrm{H}}A^{-1}R_{k+1} = \sum_{i=0}^{k}(\tilde{R}_i^{\mathrm{H}}A^{-1}R_i - \tilde{R}_{i+1}^{\mathrm{H}}A^{-1}R_{i+1}),$$

or equivalently

$$C^{\mathrm{H}}A^{-1}B = \sum_{i=0}^{k}(\tilde{R}_i^{\mathrm{H}}A^{-1}R_i - \tilde{R}_{i+1}^{\mathrm{H}}A^{-1}R_{i+1}) + \tilde{R}_{k+1}^{\mathrm{H}}A^{-1}R_{k+1}. \tag{5.9}$$

Similar to the process in [47], the key step for deriving our method uses the following relation from Algorithm 5.1,

$$\begin{aligned}
&\tilde{R}_k^{\mathrm{H}}A^{-1}R_k - \tilde{R}_{k+1}^{\mathrm{H}}A^{-1}R_{k+1} \\
&= (\tilde{R}_{k+1} + A^{\mathrm{H}}\tilde{P}_k\tilde{\alpha}_k)^{\mathrm{H}}A^{-1}(R_{k+1} + AP_k\alpha_k) - \tilde{R}_{k+1}^{\mathrm{H}}A^{-1}R_{k+1} \\
&= \tilde{R}_{k+1}^{\mathrm{H}}P_k\alpha_k + (\tilde{P}_k\tilde{\alpha}_k)^{\mathrm{H}}R_{k+1} + (\tilde{P}_k\tilde{\alpha}_k)^{\mathrm{H}}AP_k\alpha_k.
\end{aligned} \tag{5.10}$$

From the orthogonality properties (5.7) and (5.8), it is obvious that the first two terms of (5.10) vanish. As

$$\tilde{\alpha}_k = (P_k^{\mathrm{H}}A^{\mathrm{H}}\tilde{P}_k)^{-1}(R_k^{\mathrm{H}}\tilde{R}_k),$$

the third term of (5.10) can be represented as

$$(\tilde{P}_k\tilde{\alpha}_k)^{\mathrm{H}}AP_k\alpha_k = \tilde{\alpha}_k^{\mathrm{H}}(\tilde{P}_k^{\mathrm{H}}AP_k)\alpha_k = \tilde{R}_k^{\mathrm{H}}R_k\alpha_k.$$

Or substituting

$$\alpha_k = (\tilde{P}_k^{\mathrm{H}}AP_k)^{-1}(\tilde{R}_k^{\mathrm{H}}R_k)$$

into the third term of (5.10), we have an equivalent result that

$$(\tilde{P}_k\tilde{\alpha}_k)^{\mathrm{H}}AP_k\alpha_k = \tilde{\alpha}_k^{\mathrm{H}}\tilde{R}_k^{\mathrm{H}}R_k.$$

Finally we obtain that

$$\tilde{R}_k^{\mathrm{H}}A^{-1}R_k - \tilde{R}_{k+1}^{\mathrm{H}}A^{-1}R_{k+1} = \tilde{R}_k^{\mathrm{H}}R_k\alpha_k$$

(or $\tilde{\alpha}_k^{\mathrm{H}} \tilde{R}_k^{\mathrm{H}} R_k$).

Now we can rewrite (5.9) as follows:

$$C^{\mathrm{H}} A^{-1} B = \sum_{i=0}^{k} \tilde{R}_i^{\mathrm{H}} R_i \alpha_i + \tilde{R}_{k+1}^{\mathrm{H}} A^{-1} R_{k+1},$$

which motivates us to approximate $C^{\mathrm{H}} A^{-1} B$ using

$$\begin{aligned} \eta_{k+1} &\equiv \sum_{i=0}^{k} \tilde{R}_i^{\mathrm{H}} R_i \alpha_i \\ &= \eta_k + \tilde{R}_k^{\mathrm{H}} R_k \alpha_k, \end{aligned} \tag{5.11}$$

instead of computing $C^{\mathrm{H}} X_{k+1}$. Note that the computational cost for η_{k+1} is very less because $\tilde{R}_k^{\mathrm{H}} R_k$ has been computed in the block BiCG algorithm. In exact arithmetic, it is easy to prove that η_{k+1} is equal to $C^{\mathrm{H}} X_{k+1}$, but we will see that their computational results could be quite different. Here we give a summary and describe this block BiCG-based method in Algorithm 5.2.

Algorithm 5.2 Block BiCG-based

1: Let $R_0 = B$, $P_0 = R_0$, $\tilde{R}_0 = C$ and $\tilde{P}_0 = \tilde{R}_0$; $\eta_0 = \mathbf{0}$;
2: **for** $k = 0, 1, \ldots,$ **do**
3: $\quad \alpha_k = (\tilde{P}_k^{\mathrm{H}} A P_k)^{-1} (\tilde{R}_k^{\mathrm{H}} R_k)$; $\tilde{\alpha}_k = (P_k^{\mathrm{H}} A^{\mathrm{H}} \tilde{P}_k)^{-1} (R_k^{\mathrm{H}} \tilde{R}_k)$;
4: $\quad \eta_{k+1} = \eta_k + \tilde{R}_k^{\mathrm{H}} R_k \alpha_k$;
5: $\quad R_{k+1} = R_k - A P_k \alpha_k$; $\tilde{R}_{k+1} = \tilde{R}_k - A^{\mathrm{H}} \tilde{P}_k \tilde{\alpha}_k$;
6: $\quad \beta_k = (\tilde{R}_k^{\mathrm{H}} R_k)^{-1} (\tilde{R}_{k+1}^{\mathrm{H}} R_{k+1})$; $\tilde{\beta}_k = (R_k^{\mathrm{H}} \tilde{R}_k)^{-1} (R_{k+1}^{\mathrm{H}} \tilde{R}_{k+1})$;
7: $\quad P_{k+1} = R_{k+1} + P_k \beta_k$; $\tilde{P}_{k+1} = \tilde{R}_{k+1} + \tilde{P}_k \tilde{\beta}_k$;
8: **end for**

For the case that matrix A is Hermitian and matrix C is identical to B, $B^{\mathrm{H}} A^{-1} B$ can be approximated more effectively based on the block CG method. Analogous to the derivation process of (5.9), we can obtain

$$B^{\mathrm{H}} A^{-1} B = \sum_{i=0}^{k} R_i^{\mathrm{H}} R_i \alpha_i + R_{k+1}^{\mathrm{H}} A^{-1} R_{k+1},$$

and approximate $B^{\mathrm{H}} A^{-1} B$ by

$$\eta_{k+1} \equiv \sum_{i=0}^{k} R_i^{\mathrm{H}} R_i \alpha_i = \eta_k + R_k^{\mathrm{H}} R_k \alpha_k. \tag{5.12}$$

We present this block CG-based method in Algorithm 5.3.

Algorithm 5.3 Block CG-based

Let $R_0 = B$, $P_0 = R_0$; $\eta_0 = \mathbf{0}$;
for $k = 0, 1, \ldots,$ **do**
    $\alpha_k = (P_k^{\mathrm{H}} A P_k)^{-1}(R_k^{\mathrm{H}} R_k)$;
    $\eta_{k+1} = \eta_k + R_k^{\mathrm{H}} R_k \alpha_k$;
    $R_{k+1} = R_k - A P_k \alpha_k$;
    $\beta_k = (R_k^{\mathrm{H}} R_k)^{-1}(R_{k+1}^{\mathrm{H}} R_{k+1})$;
    $P_{k+1} = R_{k+1} + P_k \beta_k$;
end for

5.3 Block Conjugate Gradient type methods with residual matrix orthogonalization

In this section, we propose new ways of approximating block bilinear form based on other variants of the block conjugate gradient methods. Numerous variants of the block CG method have been implemented and analyzed by Dubrulle in [10], where it was shown that the implementation of the block CG method by orthogonalization of the residual matrix R_k (called block CGrQ method) has shown better computational performance. Motivated by this, we give an implementation of the block BiCG method by orthogonalization of the residual matrix R_k and matrix $\tilde{R}_k$. Then their corresponding methods for approximating the block bilinear form are presented.

We now discuss how to apply the strategy of residual matrix orthogonalization to the block BiCG method. Although this extension is very natural, to the best of our knowledge, there is no literature to discuss the corresponding variant of the block BiCG method. To give this variant, both the residual matrix R_k and matrix $\tilde{R}_k$ are decomposed using the thin QR decomposition, which are denoted by $R_k = Q_k C_k$ and $\tilde{R}_k = \tilde{Q}_k \tilde{C}_k$, respectively. We also define $V_k = P_k C_k^{-1}$, $\tilde{V}_k = \tilde{P}_k \tilde{C}_k^{-1}$, $S_k = C_k C_{k-1}^{-1}$ and $\tilde{S}_k = \tilde{C}_k \tilde{C}_{k-1}^{-1}$. Substituting them into Algorithm 5.1 to replace those related matrices, meanwhile matrix α_k and $\tilde{\alpha}_k$ can be represented as

$$\alpha_k = C_k^{-1}(\tilde{V}_k^{\mathrm{H}} A V_k)^{-1} \tilde{Q}_k^{\mathrm{H}} Q_k C_k$$

and

$$\tilde{\alpha}_k = \tilde{C}_k^{-1}(V_k^{\mathrm{H}} A^{\mathrm{H}} \tilde{V}_k)^{-1} Q_k^{\mathrm{H}} \tilde{Q}_k \tilde{C}_k,$$

respectively. Then reformulating the block BiCG algorithm, we can obtain a new variant. We name this new variant as block BiCGrQ and describe it in Algorithm 5.4.

Algorithm 5.4 Block BiCGrQ

Given X_0, $\tilde{R}_0$, compute $[Q_0, C_0] = \mathrm{qr}(B - AX_0)$, $[\tilde{Q}_0, \tilde{C}_0] = \mathrm{qr}(\tilde{R}_0)$; let $V_0 = Q_0$, $\tilde{V}_0 = \tilde{Q}_0$;
for $k = 0, 1, \ldots,$ **do**
  $T_k = (\tilde{V}_k^{\mathrm{H}} A V_k)^{-1}(\tilde{Q}_k^{\mathrm{H}} Q_k)$; $\tilde{T}_k = (V_k^{\mathrm{H}} A^{\mathrm{H}} \tilde{V}_k)^{-1}(Q_k^{\mathrm{H}} \tilde{Q}_k)$;
  $X_{k+1} = X_k + V_k T_k C_k$;
  $[Q_{k+1}, S_{k+1}] = \mathrm{qr}(Q_k - AV_k T_k)$; $[\tilde{Q}_{k+1}, \tilde{S}_{k+1}] = \mathrm{qr}(\tilde{Q}_k - A^H \tilde{V}_k \tilde{T}_k)$;
  $W_k = (\tilde{Q}_k^{\mathrm{H}} Q_k)^{-1} \tilde{S}_{k+1}^{\mathrm{H}} (\tilde{Q}_{k+1}^{\mathrm{H}} Q_{k+1})$; $\tilde{W}_k = (Q_k^{\mathrm{H}} \tilde{Q}_k)^{-1} S_{k+1}^{\mathrm{H}} (Q_{k+1}^{\mathrm{H}} \tilde{Q}_{k+1})$;
  $V_{k+1} = Q_{k+1} + V_k W_k$; $\tilde{V}_{k+1} = \tilde{Q}_{k+1} + \tilde{V}_k \tilde{W}_k$;
  $C_{k+1} = S_{k+1} C_k$; $\tilde{C}_{k+1} = \tilde{S}_{k+1} \tilde{C}_k$;
end for

An equivalent expression of equation (5.11) to approximate $C^{\mathrm{H}} A^{-1} B$ can be rewritten from Algorithm 5.4 as

$$\eta_{k+1} = \sum_{i=0}^{k} \tilde{C}_i^{\mathrm{H}} \tilde{Q}_i^{\mathrm{H}} Q_i T_i C_i = \eta_k + \tilde{C}_k^{\mathrm{H}} \tilde{Q}_k^{\mathrm{H}} Q_k T_k C_k.$$

Now, we can give an implementation for approximating block bilinear form $C^{\mathrm{H}} A^{-1} B$ based on the block BiCGrQ method in Algorithm 5.5.

Algorithm 5.5 Block BiCGrQ-based

Compute $[Q_0, C_0] = \mathrm{qr}(B)$, $[\tilde{Q}_0, \tilde{C}_0] = \mathrm{qr}(C)$; let $V_0 = Q_0$, $\tilde{V}_0 = \tilde{Q}_0$, $\eta_0 = \mathbf{0}$;
for $k = 0, 1, \ldots,$ **do**
  $T_k = (\tilde{V}_k^{\mathrm{H}} A V_k)^{-1}(\tilde{Q}_k^{\mathrm{H}} Q_k)$; $\tilde{T}_k = (V_k^{\mathrm{H}} A^{\mathrm{H}} \tilde{V}_k)^{-1}(Q_k^{\mathrm{H}} \tilde{Q}_k)$;
  $\eta_{k+1} = \eta_k + \tilde{C}_k^{\mathrm{H}} \tilde{Q}_k^{\mathrm{H}} Q_k T_k C_k$;
  $[Q_{k+1}, S_{k+1}] = \mathrm{qr}(Q_k - AV_k T_k)$; $[\tilde{Q}_{k+1}, \tilde{S}_{k+1}] = \mathrm{qr}(\tilde{Q}_k - A^{\mathrm{H}} \tilde{V}_k \tilde{T}_k)$;
  $W_k = (\tilde{Q}_k^{\mathrm{H}} Q_k)^{-1} \tilde{S}_{k+1}^{\mathrm{H}} (\tilde{Q}_{k+1}^{\mathrm{H}} Q_{k+1})$; $\tilde{W}_k = (Q_k^{\mathrm{H}} \tilde{Q}_k)^{-1} S_{k+1}^{\mathrm{H}} (Q_{k+1}^{\mathrm{H}} \tilde{Q}_{k+1})$;
  $V_{k+1} = Q_{k+1} + V_k W_k$; $\tilde{V}_{k+1} = \tilde{Q}_{k+1} + \tilde{V}_k \tilde{W}_k$;
  $C_{k+1} = S_{k+1} C_k$; $\tilde{C}_{k+1} = \tilde{S}_{k+1} \tilde{C}_k$;
end for

Similarly, if the orthogonalization strategy is also applied to the block CG-based method, we can obtain a block CGrQ-based method. In more detail, if we compute the QR decomposition of residual matrix $R_k = Q_k C_k$ and define $V_k = P_k C_k^{-1}$ and $S_k = C_k C_{k-1}^{-1}$, the equation (5.12) for approximating $B^{\mathrm{H}} A^{-1} B$ can be rewritten as

$$\eta_{k+1} = \sum_{i=0}^{k} C_i^{\mathrm{H}} (V_k^{\mathrm{H}} A V_k)^{-1} C_i = \eta_k + C_k^{\mathrm{H}} (V_k^{\mathrm{H}} A V_k)^{-1} C_k.$$

Meanwhile, a variant of the block CG-based method corresponding to Algorithm 5.3 can be proposed. We present it in Algorithm 5.6.

Algorithm 5.6 Block CGrQ-based

```
1: Let [Q_0, C_0] = qr(B), V_0 = Q_0, η_0 = 0;
2: for k = 0, 1, ..., do
3:     T_k = V_k^H A V_k;
4:     η_{k+1} = η_k + V_k T_k^{-1} C_k;
5:     [Q_{k+1}, S_{k+1}] = qr(Q_k − A V_k T_k^{-1});
6:     V_{k+1} = Q_{k+1} + V_k S_{k+1}^H;
7:     C_{k+1} = S_{k+1} C_k;
8: end for
```

Throughout the above discussion, we know our proposed block conjugate gradient type methods are easily implemented and only slight modifications of the corresponding block algorithms for linear systems with multiple right-hand sides are needed. Take the block BiCGrQ-based algorithm (Algorithm 5.5) for example, there is a main difference between Algorithms 5.4 and 5.5 in the step four, where the update of approximate solution of linear systems is replaced by that of the block bilinear form. Furthermore, from the computational point of view, our methods take less computational cost and memory usage than the corresponding block methods for solving linear systems since the approximate solution X_k does not need to be computed. More precisely, the memory usage for X_k is mn, and computational complexity for the update of X_k is $\mathcal{O}(m^2 n)$ per iteration; while the memory usage and computational complexity for η_k are m^2 and $\mathcal{O}(m^3)$, respectively. Meanwhile, we see that the QR decomposition should be calculated during each iteration when using block methods with residual matrix orthogonalization. Even though it indicates more computational operations, numerical results in the next section will show that all these efforts have not been wasted.

5.4 Numerical experiments

In this section, we give some numerical examples to compare the performance of our proposed methods with their corresponding block methods (block CG, block CGrQ, block BiCG and block BiCGrQ). The experiments have been performed with MATLAB R2012b on a Mac OS X Lion 10.7.5 with an Intel Core i5 processor and 4GB memory. Seven test matrices obtained from the University of Florida Sparse Matrix Collection [7] are used. A detailed description of all test matrices is provided in Table 5.1.

Table 5.1: Test matrices (matrix size: n; number of nonzero matrix elements: nnz)

Matrix A	n	nnz	type	structure	application field
1138_bus	1138	4054	real	symmetric	power system networks
Si10H16	17077	875923	real	symmetric	density functional theory calculation
Si34H36	97569	5156379	real	symmetric	density functional theory calculation
Crashbasis	160000	1750416	real	unsymmetric	mixed complementarity optimization
Pde2961	2961	14585	real	unsymmetric	model PDE problem
Tols1090	1090	3546	real	unsymmetric	computational fluid dynamics
Young1c	841	4089	complex	non-Hermitian	aero research

As our objective is to compare the performance of different methods, we assume that the exact solution of $C^{\mathrm{H}}A^{-1}B$ for each test problem, denoted by η_*, is known in advance. Two rectangular matrices C and X_* are initialized by calling the MATLAB's build-in function **rand**, and we let B be the product of AX_*. Thus $\eta_* = C^{\mathrm{H}}X_*$, note we compute $\eta_* = B^{\mathrm{H}}X_*$ when A is symmetric. The column number m of rectangular matrices is 6 and initial guess $X_0 = \mathbf{0}$ (if necessary). We stop all algorithms after a certain number of iterations. In practice, a feasible stopping criterion of relative residual can be used.

We present the corresponding results of each test matrix in Figures 5.1–5.7, respectively. In these figures the horizontal axis is labelled the iteration number, the vertical axis is labelled the relative error norm that is represented by

$$\log_{10}\frac{\|\eta_* - \eta_k\|_{\mathrm{F}}}{\|\eta_*\|_{\mathrm{F}}}$$

for block (Bi)CG-based and block (Bi)CGrQ-based, and

$$\log_{10}\frac{\|\eta_* - C^{\mathrm{H}}X_k\|_{\mathrm{F}}}{\|\eta_*\|_{\mathrm{F}}}$$

for block (Bi)CG and block (Bi)CGrQ.

We first discuss the results of three symmetric problems in Figures 5.1–5.3. Some observations of the four block CG type methods are made as follows:

- At the first few iterations, all methods show almost same behavior. Approximations computed by block methods with residual matrix orthogonalization perform significantly better. For all test problems, both block CGrQ and block CGrQ-based achieve a high accuracy of approximations as the number of iterations increases. While both block CG and block CG-based provide worse approximations except for '1138_bus'.

- Block CG-based and block CGrQ-based perform better than block CG and block CGrQ, respectively. Before stagnant approximations appear, the accuracy of approximations obtained by block CG-based and block CGrQ-based is higher than that of block CG and block CGrQ at the same iteration step, respectively. When the same accuracy is required, it means block CG-based and block CGrQ-based would take less iterations than block CG and block CGrQ. Take '1138_bus' for example, the relative errors of block CG-based, block CGrQ-based, block CG and block CGrQ after 600 iterations are about $10^{-9}, 10^{-12}, 10^{-6}$ and

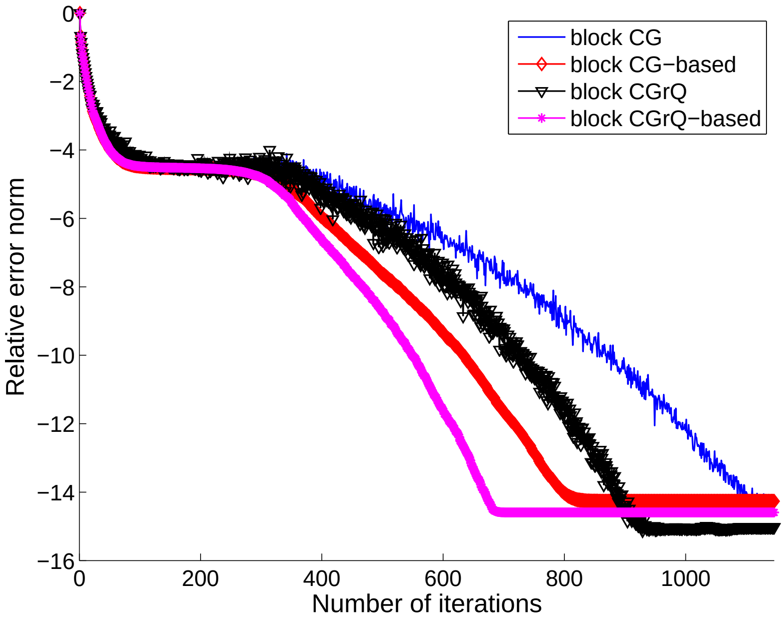

Figure 5.1: 1138_bus.

10^{-7}, respectively. For a required accuracy of 10^{-10}, block CG needs the most iterations (about 900), and block CGrQ-based needs the least iterations (about 560).

In Figures 5.4–5.7, we present the numerical results of block BiCG, block BiCGrQ, block BiCG-based and block BiCGrQ-based for the four remaining unsymmetric test problems. These block BiCG type methods show similar behavior like the block CG type methods, but also show some differences. We summarize the observations as follows.

- Approximations computed by block methods with residual matrix orthogonalization perform significantly better. Both block BiCGrQ and block BiCGrQ-based can improve the accuracy of approximations as the number of iterations increases.

- Both block BiCG and block BiCG-based cannot improve the accuracy of approximations for 'Pde2961' and 'Tols1090'. Figure 5.7 shows that the computed approximation via block BiCG yields almost the same accuracy, less than 10^{-8}, as BiCGrQ-based. But in terms of the number of iterations block BiCG requires about 180 iterations, twice

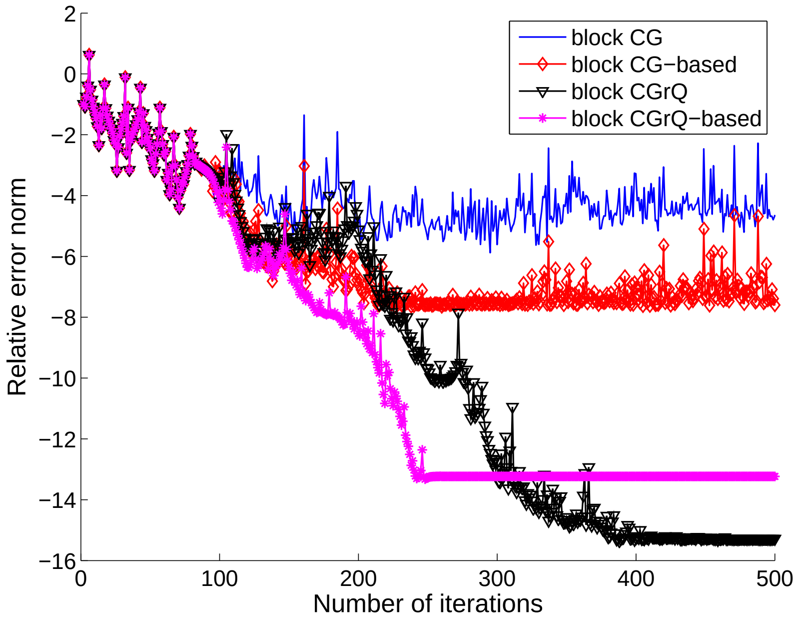

Figure 5.2: Si10H16.

more than block BiCGrQ-based (about 80 iterations). Block BiCGrQ gets the highest accuracy of approximation. All figures show block BiCGrQ-based converges toward η_* fastest before stagnation occurs. Take the matrix 'Crashbasis' for example, for a required accuracy of 10^{-8}, block BiCGrQ-based takes about 210 iterations but block BiCGrQ needs more than 500 iterations.

From all seven figures, it seems that the accuracy of approximations obtained by block CGrQ-based is a little worse than that of block CGrQ. Block BiCGrQ-based and block BiCGrQ show similar feature. It will be our future work to investigate the reasons of stagnation of block (Bi)CGrQ-based and to improve their accuracy. In order to give a more comprehensive evaluation of each method, we also present the computational time per iteration in Tables 5.2 and 5.3. We can see the difference of computational time of each method for solving the test problems.

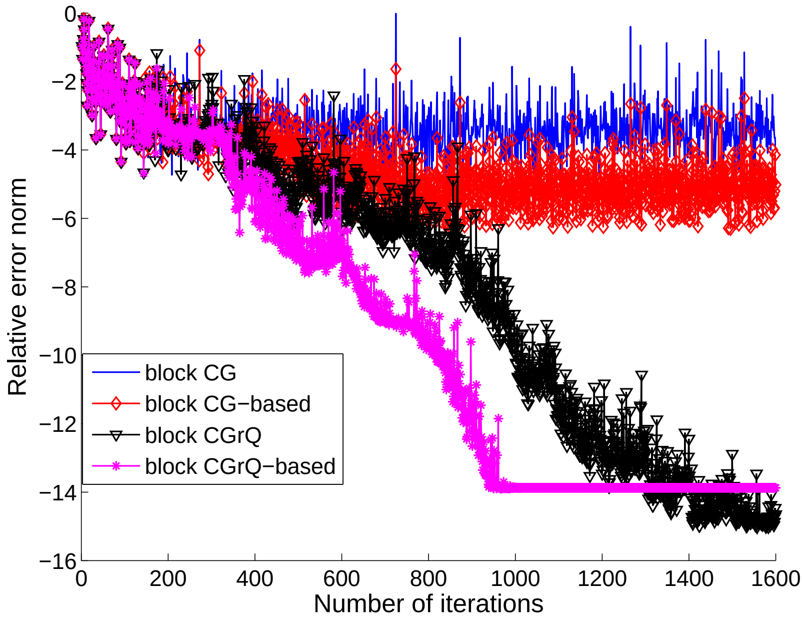

Figure 5.3: Si34H36.

5.5 Concluding remarks

In this chapter, we have discussed computing the approximation of the block bilinear form $C^{\mathrm{H}}A^{-1}B$. We first have reviewed some fundamental knowledge of block Krylov subspace methods for solving linear systems with multiple right-hand sides, and then we have proposed the block BiCG-based and block CG-based methods for the approximation of $C^{\mathrm{H}}A^{-1}B$. Taking numerical stability into account and motivated by the block CGrQ method, we also have developed a variant of the block BiCG method, named block BiCGrQ,

Table 5.2: Computational time [sec.] of block CG, block CG-based, block CGrQ and block CGrQ-based per iteration

Matrix A	block CG	block CG-based	block CGrQ	block CGrQ-based
1138_bus	3.263e-4	3.113e-4	4.938e-4	4.076e-4
Si10H16	9.238e-3	9.052e-3	1.255e-2	1.133e-2
Si34H36	7.391e-2	6.035e-2	8.455e-2	7.808e-2

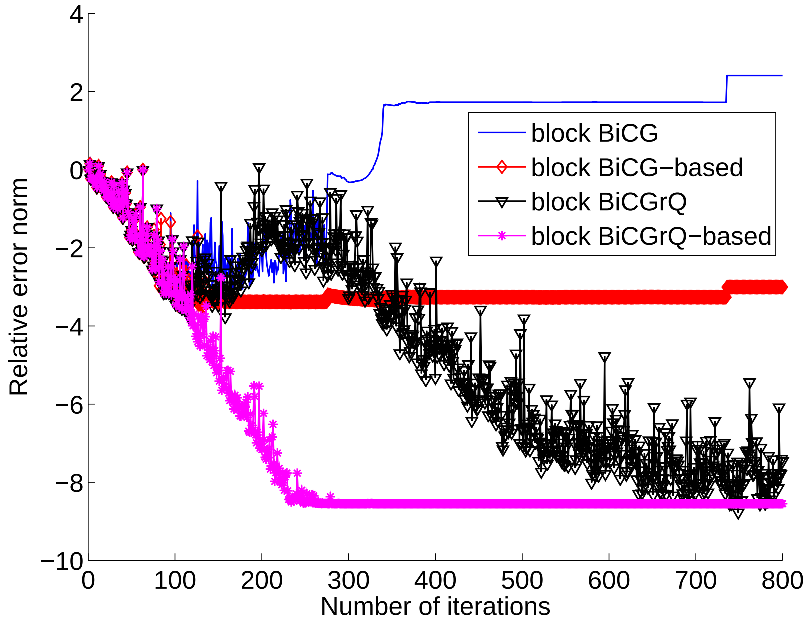

Figure 5.4: Crashbasis.

for solving linear systems with multiple right-hand sides. Then, the block CGrQ-based and block BiCGrQ-based methods have been presented. Several examples have been given to compare our proposed methods with other existing block methods. Although all the methods for computing the block bilinear form are mathematically equivalent, our methods take less computational cost and memory usage. Numerical results have shown our methods, especially the block CGrQ-based and block BiCGrQ-based methods, can effectively compute the approximation of the block bilinear form. It is known that preconditioning is key to the efficiency of iterative methods. Although we do not discuss preconditioning here, all methods discussed previously can be easily combined with efficient preconditioners.

We acknowledge here that a part of the study in this chapter is published as [55] in the list of publications.

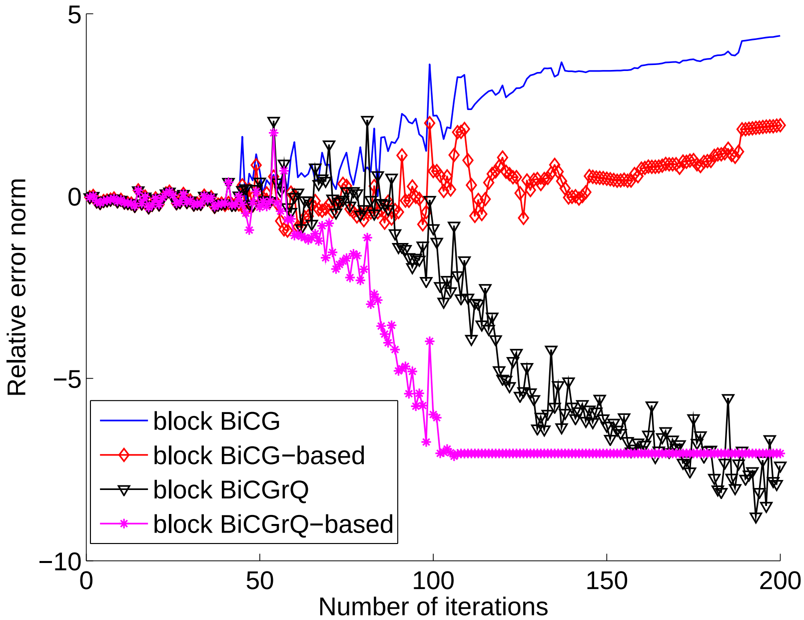

Figure 5.5: Pde2961.

Table 5.3: Computational time [sec.] of block BiCG, block BiCG-based, block BiCGrQ and block BiCGrQ-based per iteration

Matrix A	block BiCG	block BiCG-based	block BiCGrQ	block BiCGrQ-based
Crashbasis	9.325e-2	8.895e-2	1.628e-1	1.478e-1
Pde2961	1.092e-3	1.089e-3	2.074e-3	1.916e-3
Tols1090	5.946e-4	5.802e-4	9.542e-4	9.522e-4
Young1c	1.105e-3	9.869e-4	1.755e-3	1.679e-3

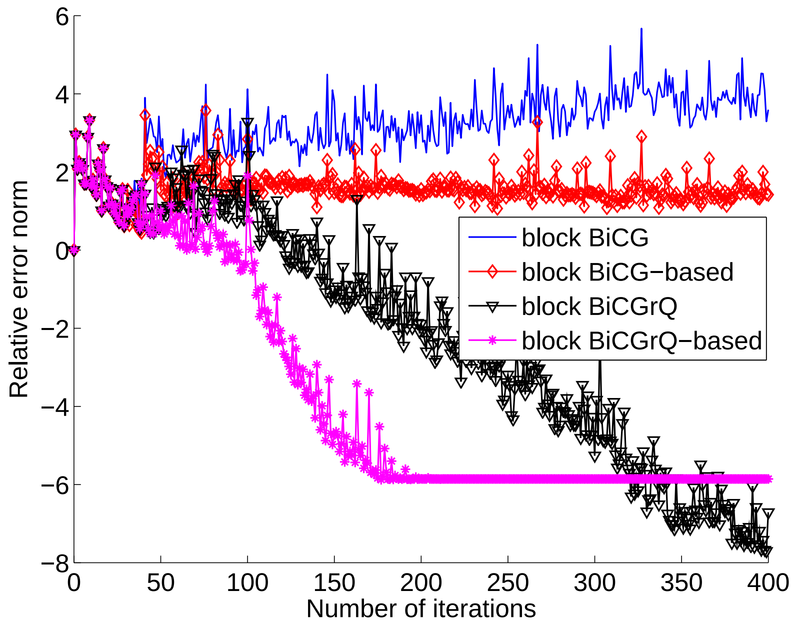

Figure 5.6: Tols1090.

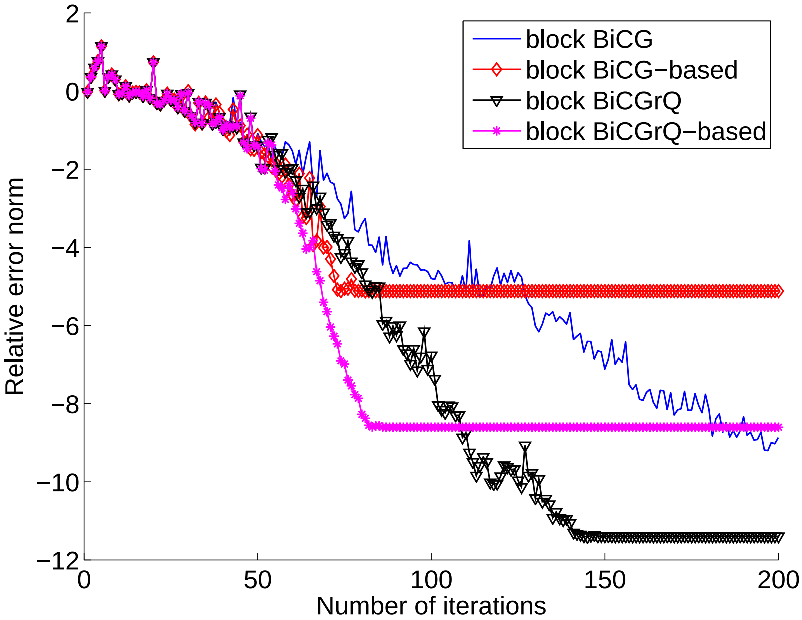

Figure 5.7: Young1c.

Chapter 6

A conjugate gradient type method for linear system with multiple shifts and multiple right hand sides

6.1 Introduction

The standard eigenvalue problem

$$A\boldsymbol{u} = \lambda \boldsymbol{u}$$

is a specific case of the generalized eigenvalue problem (1.1) and arises variety of numerical computations in science and engineering.

In the SS method, if it is applied to a standard eigenvalue problem, solutions of linear systems with multiple shifts and multiple right hand sides (RHSs)

$$(A + \sigma_j I)X_j = B, \quad j = 0, 1, \ldots, N-1. \tag{6.1}$$

are required, where $\sigma_j \in \mathbb{C}$, $A \in \mathbb{C}^{n \times n}$ and $X_j, B \in \mathbb{C}^{n \times L}$. We refer to $AX = B$ as the seed system. In the study of this chapter, we consider the case that A is Hermitian i.e. $A = A^{\mathrm{H}}$. [33] and [29] solve them by conjugate gradient (CG) type methods in case of $L = 1$. They compare the SS method with a widely used method, the Lanczos method, and found that the methods are comparable. When seed system is Hermitian, the linear systems with multiple shifts can be solved by the shifted CG method [26] even if σ_j are complex numbers [33]. Using the shift invariance of the Krylov subspace, the update of solution vectors for shifted systems can be performed without time-consuming matrix-vector products, i.e. matrix-vector products are only required for the seed system. In the study of this chapter, we deal with multiple RHSs in addition to multiple shifts to reduce the iteration count by exploiting this additional degree of freedom. A GMRES algorithm for both multiple shifts and multiple RHSs was proposed by [6]. Since we consider the case that the seed system is Hermitian, we choose the CG method as the base method. Thus, we propose the CG method for multiple shifts and multiple RHSs. We refer to the approach shown in [33] as the conventional approach.

This chapter is organized as follows. Section 6.2 describes derivation of an algorithm of a CG type method for multiple shifts and multiple RHSs. We describe implementation techniques to reduce time-consuming data copies for the algorithm in Section 6.3. We show the performance evaluation of our algorithm on the K computer in Section 6.4. Conclusion are presented in Section 6.5.

6.2 Derivation of the shifted block CG-rQ method

We derive the CG method for multiple shifts and multiple RHSs by extending the block CG method [34] for shifted systems. The block CG method solves systems with multiple RHSs by using the block Krylov subspace [20]. In the block CG method, the search space is extended by L basis per iteration. The block CG method often requires fewer iteration count than the CG method. Several techniques and variants to stabilize the block CG method are presented in [34, 32, 10]. [10] showed that a variant BCGrQ (we refer this as the block CG-rQ method) is the best variant in terms of execution time by numerical experiments. Therefore we choose the block CG-rQ method as the base method of extension for shifted systems. Algorithm 6.1 shows the block CG-rQ method.

By using subspace $\mathcal{B}_k$ of (5.6) in Chapter 5, the residual vector at the k-th iteration of the block CG-rQ which applied for the seed system $AX = B$ corresponds the i-th right hand side can be represented as

$$\boldsymbol{r}_k^{(i)} \in \mathcal{B}_{k+1}(A, R_0) \cap \mathcal{B}_k^{\perp}(A, R_0) \equiv \mathcal{M},$$

where $R_0 \equiv B - AX_0$. Similarly, the residual vector at the k-th iteration of the block BiCG of Algorithm 5.1 which applied for the $(A + \sigma I)X^\sigma = B$

$$\begin{aligned} \boldsymbol{r}_k^\sigma &\in \mathcal{B}_{k+1}(A + \sigma I, R_0^\sigma) \cap \mathcal{B}_k^{\perp}((A + \sigma I)^{\mathrm{H}}, R_0^\sigma) \\ &= \mathcal{B}_{k+1}(A + \sigma I, R_0^\sigma) \cap \mathcal{B}_k^{\perp}(A + \overline{\sigma} I, R_0^\sigma), \end{aligned}$$

where $R_0^\sigma \equiv B - (A + \sigma I)X_0^\sigma$. If

$$R_0^\sigma = R_0 \xi_0^\sigma \tag{6.2}$$

with $\xi_0^\sigma \in \mathbb{C}^{n\times n}$, $\boldsymbol{r}_k^\sigma \in \mathcal{M}$ holds due to the shift invariance of $\mathcal{B}_k$. It is known that if is vector subspaces $\mathcal{L}_1$ $\mathcal{L}_2$ are given,

$$\dim(\mathcal{L}_1) + \dim(\mathcal{L}_2) = \dim(\mathcal{L}_1 + \mathcal{L}_2) + \dim(\mathcal{L}_1 \cap \mathcal{L}_2)$$

holds. Thus

$$\begin{aligned}\dim(\mathcal{M}) &= \dim(\mathcal{B}_{k+1} \cap \mathcal{B}_k^\perp)\\ &= \dim(\mathcal{B}_{k+1}) + \dim(\mathcal{B}_k^\perp) + \dim(\mathcal{B}_{k+1} + \mathcal{B}_k^\perp)\\ &= L \end{aligned} \tag{6.3}$$

since $\dim(\mathcal{B}_{k+1}) = (k+1)L$, $\dim(\mathcal{B}_k^\perp) = n - kL$ and $\dim(\mathcal{B}_{k+1} + \mathcal{B}_k^\perp) = n$. Here $\mathcal{B}_k(A, R_0)$ is denoted as $\mathcal{B}_k$ for simplicity. Consequently, by (6.3),

$$R_k^\sigma = R_k \tilde{\xi}_k^\sigma \tag{6.4}$$

holds, where $\tilde{\xi}_k^\sigma \in \mathbb{C}^{n\times n}$. Once we obtain R_k^σ by (6.4), we can update the solution X_k^σ without a matrix vector multiplication of $A + \sigma I$. Note that since $R_k = Q_k \varDelta_k$,

$$R_k^\sigma = Q_k \xi_k^\sigma, \tag{6.5}$$

where $\xi_k^\sigma \equiv \varDelta_k \tilde{\xi}_k^\sigma$.

In following discussion, we show how to compute ξ_k^σ cheaply. By using the 7-th line and the 9-th line of Algorithm 6.1, we have

$$Q_{k+1} = Q_k \rho_{k+1}^{-1} - AQ_k \alpha_k \rho_{k+1}^{-1} - AP_{k-1} \rho_{k-1}^{\mathrm{H}} \alpha_k \rho_{k+1}^{-1} \tag{6.6}$$

and

$$AP_{k-1} = (Q_{k-1} - Q_k \rho_k) \alpha_{k-1}^{-1}. \tag{6.7}$$

Then by (6.6) and (6.7), we have

$$\begin{aligned} Q_{k+1} &= Q_k \rho_{k+1}^{-1} - AR_k \alpha_k \rho_{k+1}^{-1} - (R_{k-1} - R_k) \alpha_{k-1}^{-1} \rho_k^{\mathrm{H}} \alpha_k \rho_{k+1}^{-1}\\ &= -AQ_k \alpha_k \rho_{k+1}^{-1} + Q_k \big(I_L + \alpha_{k-1}^{-1} \rho_k^{\mathrm{H}} \alpha_k\big) \rho_{k+1}^{-1}\\ &\quad -Q_{k-1} \alpha_{k-1}^{-1} \rho_k^{\mathrm{H}} \alpha_k \rho_{k+1}^{-1}. \end{aligned} \tag{6.8}$$

On the other hand, similarly,

$$\begin{aligned} R_k^\sigma &= -AR_k^\sigma \alpha_k^\sigma + R_k^\sigma \big\{I_L - \sigma \alpha_k^\sigma + \big(\alpha_{k-1}^\sigma\big)^{-1} \beta_{k-1}^\sigma \alpha_k^\sigma\big\}\\ &\quad -R_{k-1}^\sigma \big(\alpha_{k-1}^\sigma\big)^{-1} \beta_{k-1}^\sigma \alpha_k^\sigma \end{aligned} \tag{6.9}$$

holds, where R_k^σ, α_k^σ and β_k^σ are R_k, α_k and β_k of block BiCG (see Algorithm 5.1) which applied for $(A + \sigma I)$, respectively. Since (6.5), we have

$$\begin{aligned} &-AQ_k \alpha_k \rho_{k+1}^{-1} \xi_{k+1}^\sigma + Q_k \big(I_L + \rho_k \alpha_{k-1}^{-1} \rho_k^{\mathrm{H}} \alpha_k\big) \rho_{k+1}^{-1} \xi_{k+1}^\sigma - Q_{k-1} \alpha_{k-1}^{-1} \rho_k^{\mathrm{H}} \alpha_k \rho_{k+1}^{-1} \xi_{k+1}^\sigma\\ &= -AQ_k \xi_k^\sigma \alpha_k^\sigma + Q_k \xi_k^\sigma \big\{I_L - \sigma \alpha_k^\sigma + \big(\alpha_{k-1}^\sigma\big)^{-1} \beta_{k-1}^\sigma \alpha_k^\sigma\big\} - Q_{k-1} \xi_{k-1}^\sigma \big(\alpha_{k-1}^\sigma\big)^{-1} \beta_{k-1}^\sigma \alpha_k^\sigma. \end{aligned}$$

Thus

$$\xi_k^\sigma \alpha_k^\sigma = \alpha_k \rho_{k+1}^{-1} \xi_{k+1}^\sigma,$$

$$\xi_k^\sigma \left\{ I_L - \sigma \alpha_k^\sigma + \left(\alpha_{k-1}^\sigma \right)^{-1} \beta_{k-1}^\sigma \alpha_k^\sigma \right\} = \left(I_L + \rho_k \alpha_{k-1}^{-1} \rho_k^{\mathrm{H}} \alpha_k \right) \rho_{k+1}^{-1} \xi_{k+1}^\sigma$$

and

$$\xi_{k-1}^\sigma \left(\alpha_{k-1}^\sigma \right)^{-1} \beta_{k-1}^\sigma \alpha_k^\sigma = \alpha_{k-1}^{-1} \rho_k^{\mathrm{H}} \alpha_k \rho_{k+1}^{-1} \xi_{k+1}^\sigma$$

hold. By using these relations, we have

$$\alpha_k^\sigma = \left(\xi_k^\sigma \right)^{-1} \alpha_k \rho_{k+1}^{-1} \xi_{k+1}^\sigma,$$

$$\beta_k^\sigma = \left(\xi_k^\sigma \right)^{-1} \alpha_k \rho_{k+1}^{-1} \xi_{k+1}^\sigma \left(\xi_k^\sigma \right)^{-1} \alpha_k^{-1} \rho_{k+1}^{\mathrm{H}} \xi_{k+1}^\sigma$$

and

$$\xi_{k+1}^\sigma = \rho_{k+1} \left[I_L + \sigma \alpha_k + \left\{ \rho_k - \xi_k^\sigma \left(\xi_{k-1}^\sigma \right)^{-1} \right\} \alpha_{k-1}^{-1} \rho_k^{\mathrm{H}} \alpha_k \right]^{-1} \xi_k^\sigma.$$

Therefore one can compute ξ_{k+1} by using small $L \times L$ matrices ξ_k, ξ_{k-1}, α_k, α_{k-1}, ρ_{k+1} and ρ_k. And note that this is done without time-consuming matrix-vector multiplication of $(A + \sigma I)$. Let X_k^σ and $\hat{P}_k^\sigma$ be X_k and P_k of block BiCG which applied for $(A + \sigma I)$. Here we have

$$X_{k+1}^\sigma = X_k^\sigma + \hat{P}_k^\sigma \left(\xi_k^\sigma \right)^{-1} \alpha_k \rho_{k+1}^{-1} \xi_{k+1}^\sigma$$

and

$$\hat{P}_{k+1}^\sigma = Q_{k+1} \xi_{k+1}^\sigma + \hat{P}_k^\sigma \left(\xi_k^\sigma \right)^{-1} \alpha_k \rho_{k+1}^{-1} \xi_{k+1}^\sigma \left(\xi_k^\sigma \right)^{-1} \alpha_k^{-1} \rho_{k+1}^{\mathrm{H}} \xi_{k+1}^\sigma.$$

To reduce the computational cost, we introduce

$$P_k^\sigma \equiv \hat{P}_k^\sigma \left(\xi_k^\sigma \right)^{-1}.$$

By using this, we have

$$X_{k+1}^\sigma = X_k^\sigma + P_k^\sigma \alpha_k \rho_{k+1}^{-1} \xi_{k+1}^\sigma$$

and

$$P_{k+1}^\sigma = Q_{k+1} + P_k^\sigma \alpha_k \rho_{k+1}^{-1} \xi_{k+1}^\sigma \left(\xi_k^\sigma \right)^{-1} \alpha_k^{-1} \rho_{k+1}^{\mathrm{H}}.$$

Consequently, we have obtained an algorithm for computing the solutions of $(A + \sigma_j I) X_j = B (j = 0, 1, \ldots, N-1)$ along the way of solving $AX = B$, based on block CG-rQ. We refer to this algorithm as the shifted block CG-rQ (SBCGrQ) method. The pseudo code of SBCGrQ is shown in Algorithm 6.2. Note that we need to introduce zero initial solutions so that (6.2) is automatically satisfied.

In some case of the SS method and also in the stochastic estimation method for eigenvalue count (see Section 3.4.3 and Section 4.2.2), the solutions are required in the bilinear form $B^{\mathrm{H}}X_j^{\sigma}$ rather than X_j^{σ}. In such cases, one can use recurrences

$$\eta_{k+1}^{\sigma_j} = \eta_k^{\sigma_j} + \tau_k^{\sigma_j}\alpha_k^{\sigma_j} \tag{6.10}$$

and

$$\tau_{k+1}^{\sigma_j} = \tau_k^{\sigma_j}\beta_k^{\sigma_j} \tag{6.11}$$

instead of lines 14-15. Here $\eta_k = B^{\mathrm{H}}X_k^{\sigma_j}$ and $\tau_k = B^{\mathrm{H}}P_k^{\sigma_j}$. To derive (6.11), we have used the orthogonality of residual matrices i.e. $B^{\mathrm{H}}Q_k = R_0^{\mathrm{H}}Q_k = O^{L\times L}$ (k=0,1,...), where $O^{L\times L}$ is $L \times L$ zero matrix. By using (6.10) and (6.11), the computational cost is drastically reduced when the number of shifts N is large. We refer to this variant of SBCGrQ for bilinear form as SBCGrQ-based.

If a preconditioner is applied, preconditioned coefficient matrices of shifted linear systems are no longer shifted matrices in general. Thus applicable preconditioners are limited (e.g. the incomplete LU preconditioner can not be applied) for block Krylov subspace methods that use the shift invariance. For this reason we omit considering preconditioners in this study.

To implement the SBCGrQ method for distributed parallel computers, we introduce the row-wise distribution. We implement our distributed parallel code with Message Passing Interface (MPI). In row-wise distribution, matrix-matrix product with a Hermitian transpose matrix in the third line and the QR decomposition in the 7th line are performed with MPI_Allreduce to sum local results. The parallel implementation for the matrix-vector products AP_k depends on the application. The calculations in lines 8,11-13 are replicated. Other lines can be executed without MPI communications.

6.3 Efficient implementation with recurrence unrolling

In the SS method, a number of shifted systems should be solved. In such a case, computational cost for lines 11-15 becomes dominant. Especially lines 14,15 are the most time-consuming part of the algorithm. In addition, the computational cost of lines 14,15 increases $O(L^2)$ with increasing L. We reduce execution time for this computation by following techniques. Algorithm 6.3 shows an naive implementation of the 9th line. Note that we reuse the memory area of the variables with subscript k for corresponding variables with subscript $k+1$. We use simplified notations of the two BLAS subroutines ZGEMM and ZCOPY. Here, ZGEMM(A,B,C) operates $C \leftarrow AB + C$

Algorithm 6.1 Pseudo code of the block CG-rQ method. $O_{n\times L}$ is the $n\times L$ dimensional zero matrix. I_L is the L dimensional unit matrix. $\mathrm{qr}(C)$ indicates the QR decomposition of matrix C.

$R_0 = B - AX_0$
$Q_0\Delta_0 = \mathrm{qr}(R_0)$
$P_0 = Q_0$
for $k = 0, 1, \dots$ until solutions converge **do**
  $\alpha_k = \left(P_k{}^{\mathrm{H}}AP_k\right)^{-1}$
  $X_{k+1} = X_k + P_k\alpha_k\Delta_k$
  $Q_{k+1}\rho_{k+1} = \mathrm{qr}(Q_k - AP_k\alpha_k)$
  $\Delta_{k+1} = \rho_{k+1}\Delta_k$
  $P_{k+1} = Q_{k+1} + P_k\rho_{k+1}^{\mathrm{H}}$
end for

Algorithm 6.2 Pseudo code of the SBCGrQ method. $O_{n\times L}$ is the $n\times L$ dimensional zero matrix. I_L is the L dimensional unit matrix. $\mathrm{qr}(C)$ indicates the QR decomposition of matrix C.

$X_0^{\sigma_j} = O_{n\times L}$, $\xi_{-1}^{\sigma_j} = \alpha_{-1} = I_L$,
$Q_0\rho_0 = \mathrm{qr}(B)$
$\xi_0^{\sigma_j} = \Delta_0 = \rho_0$, $P_0^{\sigma_j} = P_0 = Q_0$
for $k = 0, 1, \dots$ until solutions converge **do**
  $\alpha_k = \left(P_k{}^{\mathrm{H}}AP_k\right)^{-1}$
  $X_{k+1} = X_k + P_k\alpha_k\Delta_k$
  $Q_{k+1}\rho_{k+1} = \mathrm{qr}(Q_k - AP_k\alpha_k)$
  $\Delta_{k+1} = \rho_{k+1}\Delta_k$
  $P_{k+1} = Q_{k+1} + P_k\rho_{k+1}^{\mathrm{H}}$
  for $j = 0, 1, \dots, N-1$ **do**
    $\xi_{k+1}^{\sigma_j} = \rho_{k+1}\left[I_L + \sigma_j\alpha_k + \left\{\rho_k - \xi_k^{\sigma_j}\left(\xi_{k-1}^{\sigma_j}\right)^{-1}\right\}\left(\alpha_{k-1}\right)^{-1}\rho_k^{\mathrm{H}}\alpha_k\right]^{-1}\xi_k^{\sigma_j}$
    $\alpha_k^{\sigma_j} = \alpha_k\left(\rho_{k+1}\right)^{-1}\xi_{k+1}^{\sigma_j}$
    $\beta_k^{\sigma_j} = \alpha_k\left(\rho_{k+1}\right)^{-1}\xi_{k+1}^{\sigma_j}\left(\xi_k^{\sigma_j}\right)^{-1}\left(\alpha_k\right)^{-1}\rho_{k+1}^{\mathrm{H}}$
    $X_{k+1}^{\sigma_j} = X_k^{\sigma_j} + P_k^{\sigma_j}\alpha_k^{\sigma_j}$
    $P_{k+1}^{\sigma_j} = Q_{k+1} + P_k^{\sigma_j}\beta_k^{\sigma_j}$
  end for
end for

and ZCOPY(A,B) operates $B \leftarrow A$. To exploit the efficiency of the cache

Algorithm 6.3 Naive implementation. $T \in \mathbb{C}^{n \times L}$ is a temporary variable.

ZGEMM($P_k^{\sigma_j}$, $\alpha_k^{\sigma_j}$, $X_{k+1}^{\sigma_j}$)
ZCOPY(Q_{k+1}, T)
ZGEMM($P_k^{\sigma_j}$, $\beta_k^{\sigma_j}$, T)
ZCOPY(T, $P_{k+1}^{\sigma_j}$)

blocking of ZGEMM, we operate the products $P_k^{\sigma_j}\alpha_k^{\sigma_j}$ and $P_k^{\sigma_j}\beta_k^{\sigma_j}$ in block as $P_k^{\sigma_j}[\alpha_k^{\sigma_j}, \beta_k^{\sigma_j}]$. The drawback of this approach is that additional 2 ZCOPY calls for X^{σ_j} are required. We reduce the total number of ZCOPY calls by unrolling the recurrences for X_{k+1} and P_{k+1}. The recurrences can be unrolled as

$$X_{k+1}^{\sigma_j} = X_{k-u}^{\sigma_j} + \sum_{h=0}^{u-1} Q_{k-h}\gamma_h^{\sigma_j} + P_{k-u}\gamma_u^{\sigma_j}$$

and

$$P_{k+1}^{\sigma_j} = Q_{k+1} + \sum_{h=0}^{u-1} Q_{k-h}\delta_h^{\sigma_j} + P_{k-u}^{\sigma_j}\delta_u^{\sigma_j}.$$

Here,

$$\begin{cases} \gamma_0^{\sigma_j} = \alpha_k^{\sigma_j} \\ \gamma_h^{\sigma_j} = \alpha_{k-h}^{\sigma_j} + \beta_{k-h}^{\sigma_j}\gamma_{h-1}^{\sigma_j} \end{cases},$$

$$\begin{cases} \delta_0^{\sigma_j} = \beta_k^{\sigma_j} \\ \delta_h^{\sigma_j} = \beta_{k-h}^{\sigma_j}\delta_{h-1}^{\sigma_j} \end{cases}$$

and

$$\theta_h^{\sigma_j} = [\gamma_h^{\sigma_j}, \delta_h^{\sigma_j}].$$

Algorithm 6.4 shows the implementation which uses these relations. By this implementation, the total number of ZCOPY calls is reduced from $2K$ to $4K/u$ when $u > 2$ since ZCOPY is only called every u iterations. Here K is the number of iterations which is required to satisfy the stopping criterion. Simular to the implementation in Algorithm 6.3, we reuse the memory area of the variables with subscript $k - u$ for corresponding variables with subscript $k + 1$. The problem is that the implementation shown in Algorithm 6.4 requires an additional memory requirement, mainly that of Q_{k-h} ($h = 0, 1, \ldots, u-1$). Note that this memory requirement is comparable with that of $X_k^{\sigma_j}$ and $P_k^{\sigma_j}$ ($j = 0, 1, \ldots, N-1$) when $u \approx N$.

Algorithm 6.4 Implementation with recurrence unrolling. $T_2 \in \mathbb{C}^{n\times 2L}$ is a temporary variable.

```
if mod(k + 1,u + 1)= 0 then
   ZCOPY(X_{k-u}^{σ_j}, T_2(:,1:L))
   ZCOPY(Q_{k+1}, T_2(:,L + 1:2L))
   for h = 0, 1, ..., u − 1 do
      ZGEMM(Q_{k-h}, θ_h, T_2)
   end for
   ZGEMM(P_{k-u}^{σ_j}, θ_u, T_2)
   ZCOPY(T_2(:,1:L), X_{k+1}^{σ_j})
   ZCOPY(T_2(:,L + 1:2L), P_{k+1}^{σ_j})
end if
```

6.4 Numerical experiments

In this section we show the performance of the SBCGrQ method and the SBCG-based method in two examples. In the experiments, all examples are performed on the K computer. The K computer is a distributed memory supercomputer system which has more than 80,000 compute nodes. It is installed in the RIKEN Advanced Institute for Computational Science as a Japanese national project. A SPARC64TM VIIIfx CPU which has eight cores is equipped for a compute node. The clock frequency and the peak performance of the CPU are 2 GHz and 128 giga-flops, respectively. Our code is compiled with Fujitsu Fortran Compiler.

6.4.1 Example 1

In this example, we perform numerical experiments to evaluate the efficiency of the SBCGrQ method and the recurrence unrolling technique described in the previous section. We utilize the SBCGrQ method in the eigenvalue solution of the SS method. The test matrix is a matrix derived from a real-space density functional theory calculation of a silicon nanowire which consists of 9924 Si atoms [21]. The dimension of the matrix is $n = 8,719,488$. We describe common parameter setting for all experiments as follows. The contour pass for the SS method is a circle with a center of 0.05 and a radius of 0.01. The number of quadrature points is $N = 32$. The RHS vectors are generated by random numbers. We executed the experiments with 768 MPI processes and each MPI process had 8 OpenMP threads. Note that the results of the numerical experiments are obtained by early access to the K computer.

First, we evaluate the execution time of the SS method, the number of eigenvalues that can be obtained by the SS method, and the iteration count and the execution time for the SBCGrQ method. The results of experiments are shown in Table 6.1. The parameter u is set to $u = 32$. Table 6.1 shows the elapsed time for the SS method is mostly occupied by the solutions of the linear systems with the SBCGrQ method in all cases. Large *#eig* is obtained by large L. This result is predictable since large subspace is given by large number of RHSs. The remarkable thing is that although the number of linear systems to be solved increase L-fold, *linsol_time* does not. This trend is mainly supported by the behavior that *#iter* decreases with increasing L as is the case in the block CG method [32]. We have succeeded to extend this feature for multiple shifts by developing the SBCGrQ method. Note that the case $L = 1$ and the conventional approach described in [33] are equivalent except that scaling of the vectors are different and the conventional approach was not implemented with recurrence unrolling. Thus, we can find in the column *Speed-up* for the case $L = 32$ that the SBCGrQ method is more than five times faster than the case that the shifted CG method is sequentially applied to each RHS if these is no significant difference in the iteration count for different RHSs.

Table 6.1: *#iter* and *linsol_time* are iteration count and elapsed time for SBCGrQ method, respectively. *#eig* is the number of eigenvalues derived in contour pass with relative residuals less than 1e−2. *SS_time* is elapsed time for the SS method. *Speed_up* is the speed-up ratio of average elapsed time for one RHS comparing to $L = 1$, i.e. $(128.2 \times L)$ / *linsol_time*.

L	1	2	4	8	16	32	64
#iter	10626	10560	9999	8382	6501	4455	4026
#eig	10	21	43	82	159	212	271
SS_time [sec]	131.8	197.7	247.0	395.2	442.5	721.1	1714.6
linsol_time [sec]	128.2	195.3	246.3	349.3	432.8	698.1	1600.5
Speed-up	1	1.31	2.08	2.93	4.74	5.87	5.12

Next we see the detailed data that support the remarkable results described above. Figure 6.1 shows the results of experiments to see the behaviors of the dominant parts of the SBCGrQ method with increasing L. *Matvec* is the elapsed time of the matrix-vector products with A in the 5th line of Algorithm 6.2. *QR* is the elapsed time of the QR decomposition for the 7th line of Algorithm 6.2. *Shift* is the elapsed time of the calculations for lines 11-15 of Algorithm 6.2. Note that the time data are average data for one RHS of

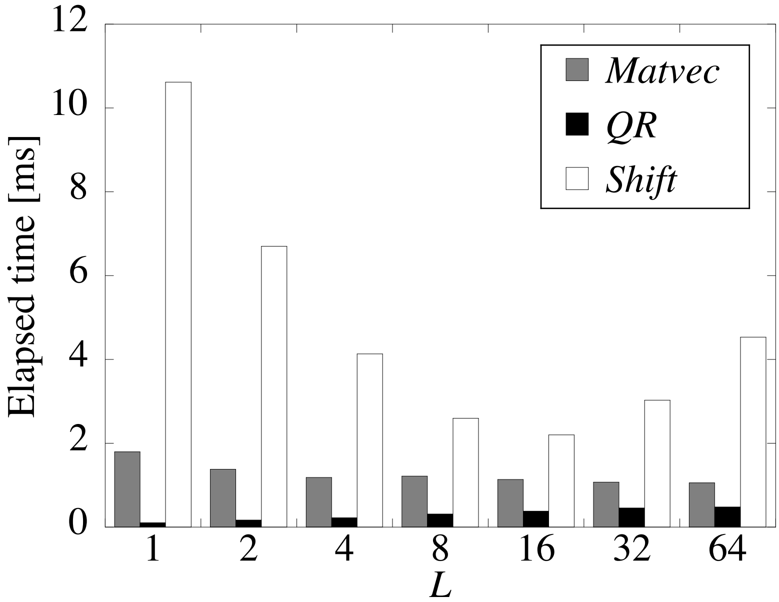

Figure 6.1: Details of elapsed time for *linsol_time*.

one iteration. *Matvec* slightly decreases with increasing L since latency for communication was reduced by sending or receiving L-fold data at once. QR increases with increasing L since the computational cost increases $O(L^2)$. The most time-consuming item *Shift* decreases until $L = 16$. This result indicates that the efficiency of cache blocking of ZGEMM hides the growth of the computational cost. However, *Shift* increases when $L = 32, 64$ due to the high complexity. Figure 6.2 shows the results of experiments to see the behaviors of the dominant parts of *Shift* with increasing u of the recurrence unrolling technique. The number of RHSs is fixed to $L = 32$. *Square* is the elapsed time for calculations that involve L dimensional square matrices in lines 11-13 of Algorithm 6.2. *ZCOPY* and *ZGEMM* are the elapsed time for ZCOPY and ZGEMM in Algorithm 6.3 or Algorithm 6.4 that implement the calculations for lines 14-15 of Algorithm 6.2. Note that the time data indicates average data for one shift of one iteration. The computational cost for *Square* other than *naive* is larger than that of *naive* due to calculations for θ_h. Practically, the elapsed time of all cases rarely different since this additional computational cost is negligible. We can find that elapsed time for

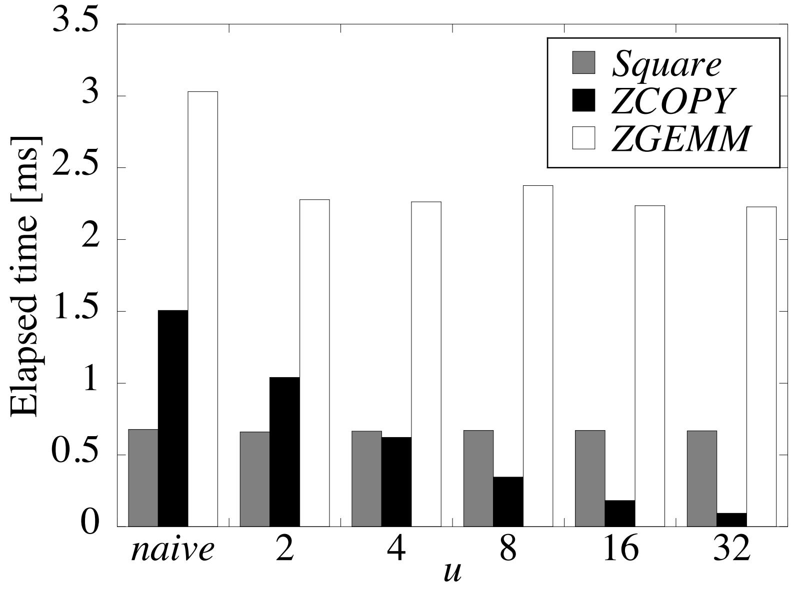

Figure 6.2: Details of elapsed time for *Shift*.

ZGEMM is reduced by the recurrence unrolling technique. This is because the cache hit ratio is improved by merging two calls of ZGEMM into once. Moreover the elapsed time for ZCOPY decreases linearly with increasing u, since ZCOPY is only called every u iterations. We can find in these details that the efficient use of ZGEMM and the reduction of total call for time-consuming ZCOPY contribute to the remarkable efficiency of the SBCGrQ method.

6.4.2 Example 2

In Example 2, we show the performance of SBCGrQ-based by using it for the stochastic estimation method proposed in Chapter 4. The test matrix A is derived from the density functional calculation of a 2,744-atom system of silicon. The matrix size is $n = 592,704$, and the smallest 5,488 eigenpairs are desired. The stopping criterion for the SBCGrQ method with respect to the relative residual norm of linear system is 1e-4. Five hundred circles are placed in the interval $[-0.25, 0.16]$ which includes desired eigenvalues.

The number of quadrature points of each circle is $N = 8$, and the number of sample vectors is $s = 20$. We executed the experiments with 64 MPI processes and each MPI process had 8 OpenMP threads. The compile option was `-Kfast,parallel,openmp`. The results are shown in Figure 6.3, Figure 6.4, Figure 6.5, and Figure 6.6. The horizontal axis indicates the index of the circles, and the vertical axis indicates the eigenvalue count for the circle's subdomain. The exact values are calculated by the conjugate gradient method for eigenvalue problems [25]. In these figures, we can see that the eigenvalue counts are roughly estimated. The computation time for the combination of the stochastic estimation method and SBCGrQ-based is 472 seconds, whereas it takes 13,200 seconds for the conjugate gradient eigensolver. Thus, in this case, the combination of the stochastic estimation method and the SBCGrQ method can be cheaply used as a preprocess for the SS method.

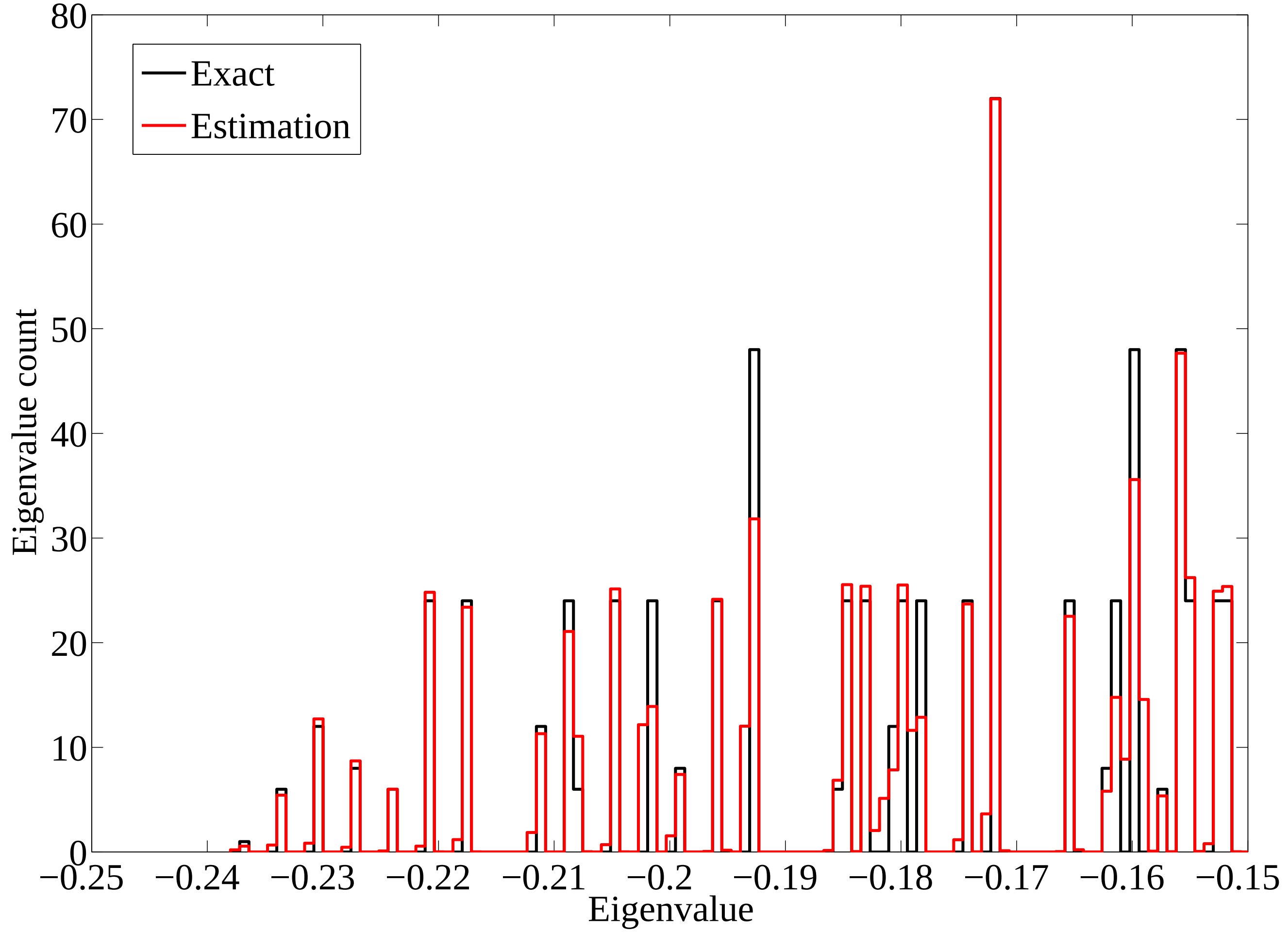

Figure 6.3: Estimated and exact eigenvalue distribution of a matrix of 2744-atom system of silicon. (1)

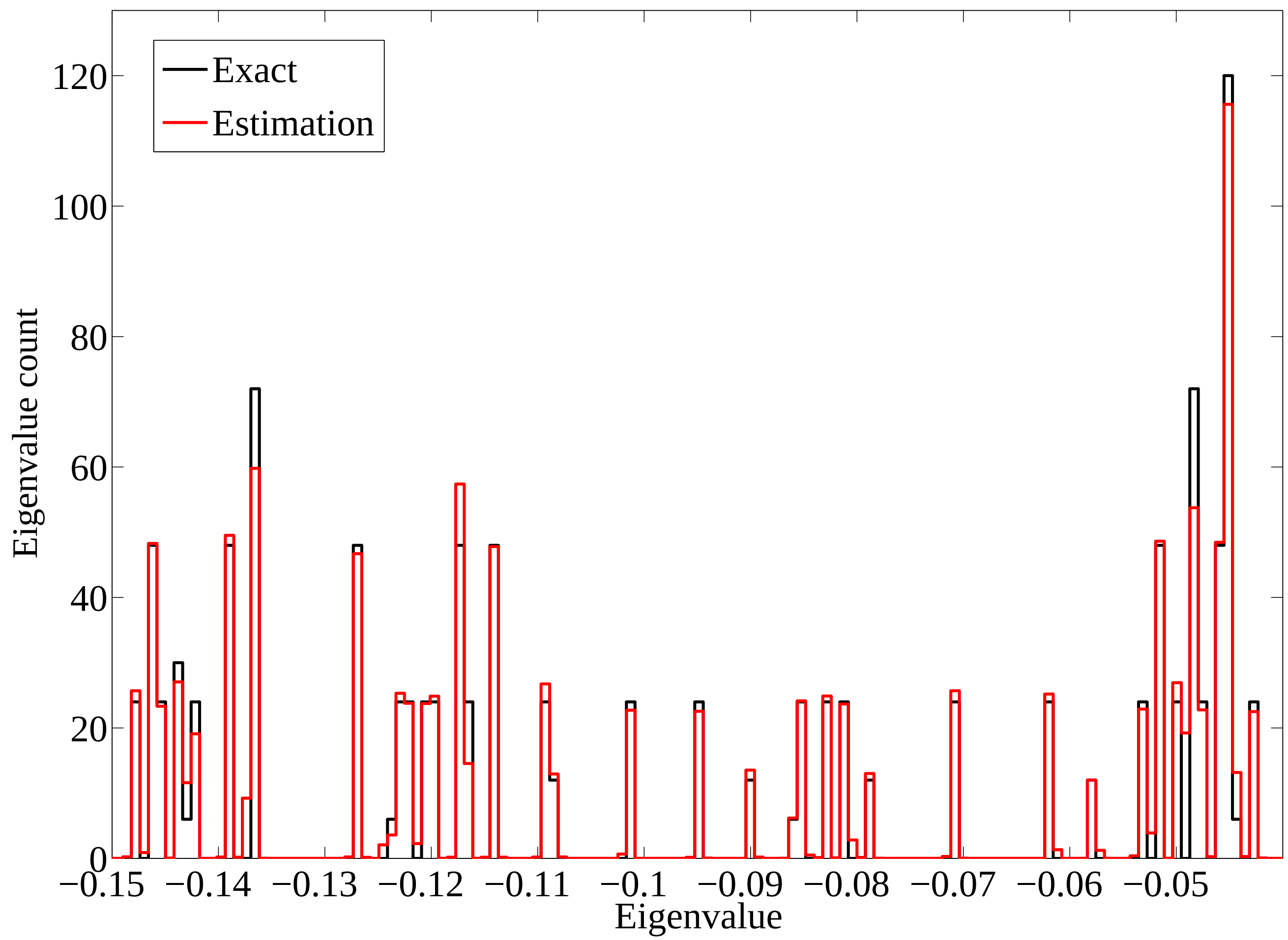

Figure 6.4: Estimated and exact eigenvalue distribution of a matrix of 2744-atom system of silicon. (2)

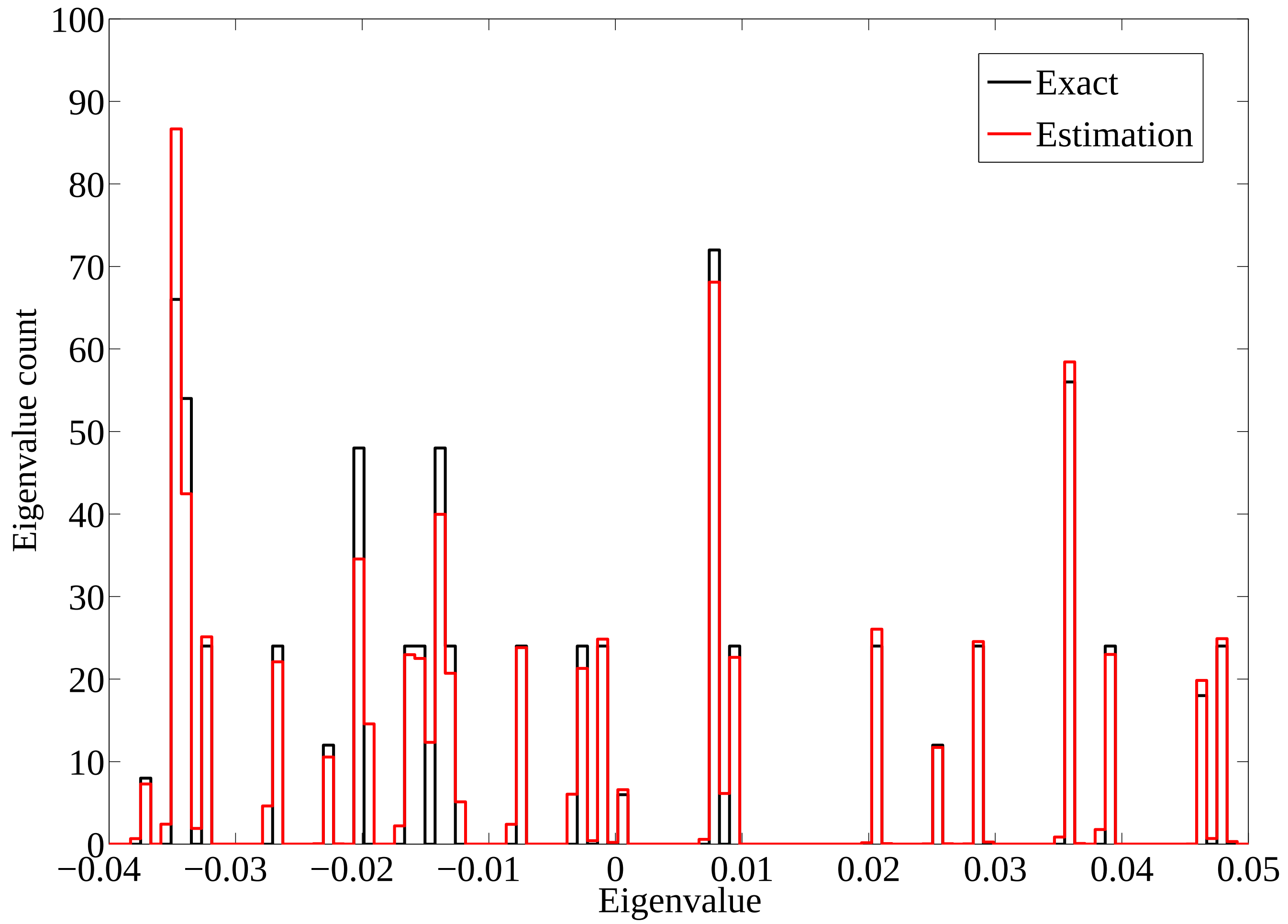

Figure 6.5: Estimated and exact eigenvalue distribution of a matrix of 2744-atom system of silicon. (3)

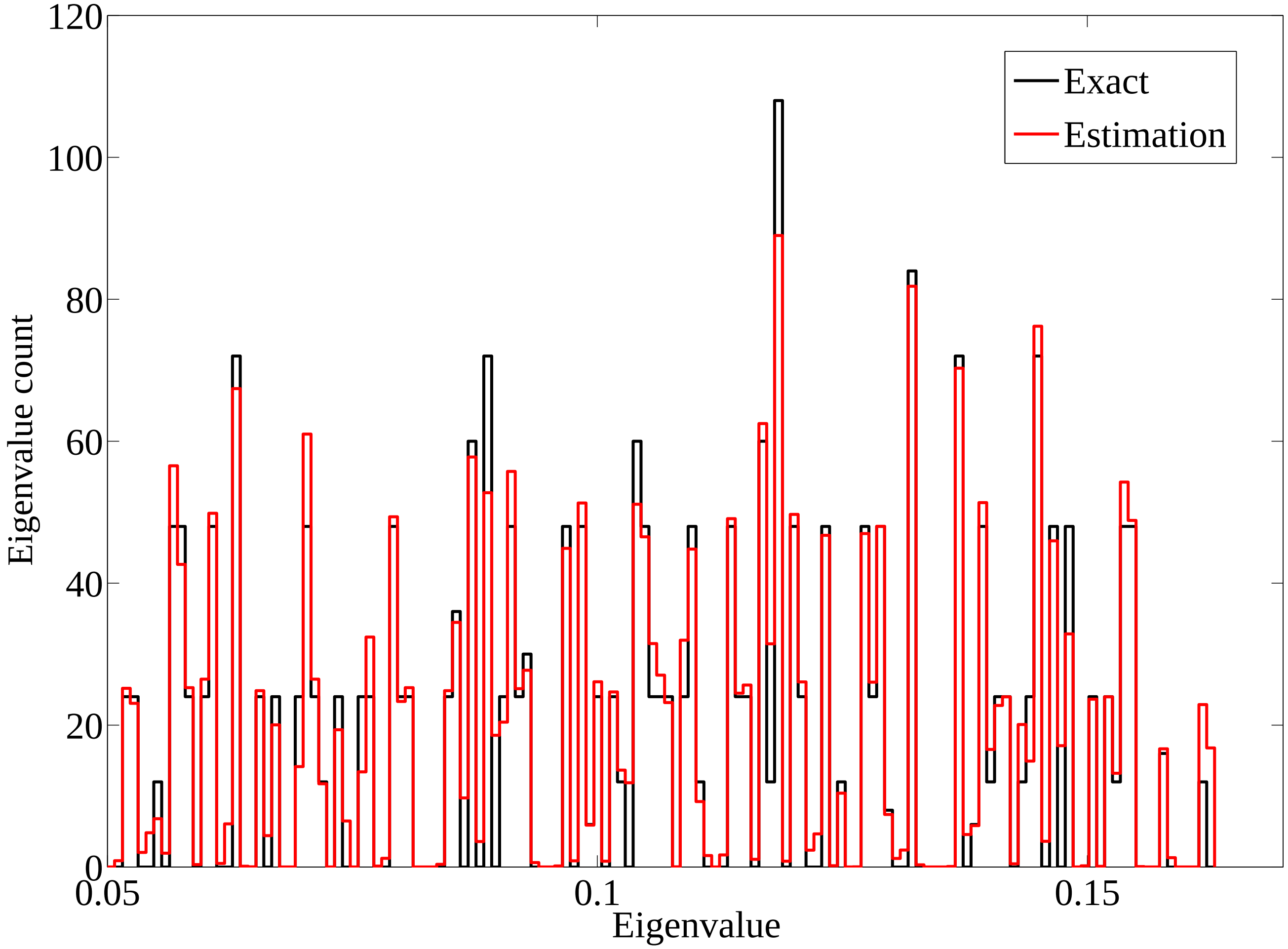

Figure 6.6: Estimated and exact eigenvalue distribution of a matrix of 2744-atom system of silicon. (4)

6.5 Concluding remarks

We have proposed a CG type method for linear systems with multiple shifts and multiple RHSs and efficient implementation techniques that reduce time-consuming data copies in the method. The proposed method can be used for linear systems that arise in the SS method. We have utilized the proposed method for the electronic-structure calculation of a large system which consists of about 10,000 Si atoms. We have found that the proposed method solves the linear systems more than five times faster than the conventional approach and have shown how much our implementation techniques contribute to efficiency of the proposed method. We have also shown that the combination of the stochastic estimation method for eigenvalue distribution and the proposed method is much faster than an accurate solution by a eigensolver in a numerical experiment with matrix of 2,744-atom system of silicon.

We acknowledge here that a part of the study in this chapter is published as [56] (in the list of publications) by Springer.

Chapter 7

Conclusion

Throughout this thesis, we have described methods and techniques for efficiently computing eigenvalues and eigenvectors of standard and generalized eigenvalue problems with a contour integral method.

In Chapter 3, some numerical properties of a contour integral method, namely the Sakurai-Sugiura (SS) method were presented from the view-point of a filter for a subspace. According to the results, efficient parameter estimation techniques were shown. The contour integral for a matrix inverse is regarded as a filter for an eigensubspace. When the contour integral is approximated by a numerical quadrature, the quadrature error causes contamination of the eigencomponents corresponding to the eigenvalues located outside of the contour path. We showed the efficiency of the SS method with numerical experiments.

In Chapter 4, we proposed a stochastic estimation method of eigenvalue counting within a given closed curve. The method is feasible for large sparse matrices or matrices that are only referenced in the form of matrix-vector multiplication. The stochastic estimation method for the eigenvalue distribution is defined by separating the given domain to several sub-domains and estimating the eigenvalue count in each sub-domain. Furthermore, since the computation of the method has independence, it is easy to execute on massively parallel computing environments. The proposed method can be used for a preprocess of the SS method to set efficient parameters. In the numerical examples, we found that the stochastic estimation method roughly estimates the eigenvalue distribution using only a few quadrature points and sample vectors.

In Chapter 5, we proposed the block BiCG-based and block CG-based methods for the approximation of a block bilinear form which need to be computed in a special case of the SS method and the method proposed in Chapter 4. Taking numerical stability into account and motivated by the block CGrQ method, we also developed a variant of the block BiCG method, named block BiCGrQ, for solving linear systems with multiple right-hand sides. Then, the block CGrQ-based and block BiCGrQ-based methods were presented. Several numerical examples were shown to compare the proposed methods with other existing block methods. Although all the methods for computing the block bilinear form are mathematically equivalent, our methods take less computational cost and memory usage. The numerical results showed the proposed methods, especially the block CGrQ-based and block BiCGrQ-based methods, can effectively compute the approximation of the block bilinear form.

In Chapter 6, we proposed a CG type method for linear systems with multiple shifts and multiple RHSs and efficient implementation techniques that reduce time-consuming data copies in the method. We call the proposed method as the shifted block CG-rQ (SBCGrQ) method. The SBCGrQ method can be used for linear systems that arise in the algorithm of the SS method if it is used for Hermitian standard eigenvalue problems. We utilized the SBCGrQ method for the electronic-structure calculation of a large system which consists of about 10,000 Si atoms. We found that the proposed method solves the linear systems more than five times faster than the conventional approach and have shown how much our implementation techniques contribute to efficiency of the SBCGrQ method. We also proposed a variant of SBCGrQ method for computing (shifted) block bilinear form which referred to as SBCGrQ-based. We applied the combination of SBCGrQ-based and the stochastic estimation method for eigenvalue distribution to a large matrix derived from electronic structure calculation of a 2,744-atom system of silicon. We observed that the estimation of the eigenvalue distribution is much faster than an accurate solution by a eigensolver. The combination of SBCGrQ-based and the stochastic estimation method for eigenvalue distribution can be efficiently used for a preprocess of the SS method.

For a future work, we will extend the study in Chapter 5 for linear systems arising in the solutions of the generalized eigenproblem and the polynomial eigenproblem. The strategy of parameter setting for the SS method using the stochastic estimation method is not studied in this thesis, the study for the strategy of parameter setting is also stated as a future work.

Bibliography

[1] J. Asakura, T. Sakurai, H. Tadano, T. Ikegami, and K. Kimura. A numerical method for nonlinear eigenvalue problems using contour integrals. *JSIAM Letters*, 1:52–55, 2009.

[2] J. Asakura, T. Sakurai, H. Tadano, T. Ikegami, and K. Kimura. A numerical method for polynomial eigenvalue problems using contour integral. *Japan Journal of Industrial and Applied Mathematics*, 27(1):73–90, 2010.

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

[4] Z. Bai, M. Fahey, and G. Golub. Some Large Scale Matrix Computation Problems. *J. Comput. Appl. Math*, 74:71–89, 1996.

[5] W.-J. Beyn. An integral method for solving nonlinear eigenvalue problems. *Linear Algebra and its Applications*, 436(10):3839–3863, 2012. Special Issue dedicated to Heinrich Voss's 65th birthday.

[6] D. Darnell, R. B. Morgan, and W. Wilcox. Deflated GMRES for Systems with Multiple Shifts and Multiple Right-Hand Sides. *Linear Algebra Appl.*, 429(10):19, 2007.

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

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

[9] L. Du, T. Sogabe, B. Yu, Y. Yamamoto, and S. L. Zhang. A Block IDR(s) Method for Nonsymmetric Linear Systems with Multiple Right-hand Sides. *J. Comput. Appl. Math.*, 235(14):4095–4106, 2011.

[10] A. Dubrulle, A. Retooling the method of block conjugate gradients. *Electronic Trans. Numer. Anal.*, 12:216–233, 2001.

[11] K. S. H. el Guennouni A. Jbilou. A block version of BiCGSTAB for linear systems with multiple right-hand sides. *ETNA. Electronic Transactions on Numerical Analysis [electronic only]*, 16, 2003.

[12] R. Fletcher. Conjugate gradient methods for indefinite systems. In G. Watson, editor, *Numerical Analysis*, volume 506 of *Lecture Notes in Mathematics*, pages 73–89. Springer Berlin Heidelberg, 1976.

[13] R. Freund. Solution of shifted linear systems by quasi-minimal residual iterations. In L. Reichel, A. Ruttan, R. Varga, and W. de Gruyter, editors, *Numerical Linear Algebra*, pages 101–121. Berlin, 1993.

[14] R. W. Freund and M. Malhotra. A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides. *Linear Algebra and its Applications*, 254(1-3):119–157, 1997.

[15] G. Golub and C. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[16] G. H. Golub and G. Meurant. Matrices, moments and quadrature, in Numerical Analysis 1993. In *D. F. Griffiths and G. A. Watson*, volume 303, pages 105–156. Pitman Research Notes in Mathematics, 1994.

[17] G. H. Golub and G. Meurant. *Matrices, Moments and Quadrature with applications*. Princeton University Press, Princeton, NJ, 2010.

[18] H. Guo and R. A. Renaut. Estimation of $\boldsymbol{u}^{\mathrm{T}} f(A)\boldsymbol{v}$ for large-scale unsymmetric matrices. *Numerical Linear Algebra with Applications*, 11(1):75–89, 2004.

[19] M. H. Gutknecht. Block Krylov space methods for linear systems with multiple right-hand sides: An introduction. In *Modern Mathematical Models, Methods and Algorithms for Real World Systems*. Anamaya Publishers, 2006.

[20] M. H. Gutknecht and T. Schmelzer. The block grade of a block Krylov space. *Linear Algebra Appl.*, 430(1):174–185, 1995.

[21] Y. Hasegawa, J.-I. Iwata, M. Tsuji, D. Takahashi, A. Oshiyama, K. Minami, T. Boku, F. Shoji, A. Uno, M. Kurokawa, H. Inoue, I. Miyoshi,

and M. Yokokawa. First-principles calculations of electron states of a silicon nanowire with 100,000 atoms on the K computer. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 1:1–1:11, New York, NY, USA, 2011. ACM.

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

[23] T. Ikegami and T. Sakurai. Contour integral eigensolver for non-Hermitian systems: a Rayleigh-Ritz-type approach. *Taiwanese J. Math.*, 14:825–837, 2010.

[24] T. Ikegami, T. Sakurai, and U. Nagashima. A Filter Diagonalization for Generalized Eigenvalue Problems Based on the Sakurai-Sugiura Projection Method. *J. Comput. Appl. Math.*, 233(8):1927–1936, 2010.

[25] J.-I. Iwata, D. Takahashi, A. Oshiyama, T. Boku, K. Shiraishi, S. Okada, and K. Yabana. A massively-parallel electronic-structure calculations based on real-space density functional theory. *J. Comput. Phys.*, 229(6):2339–2363, 2010.

[26] B. Jegerlehner. Krylov space solvers for shifted linear systems. *arXiv:hep-lat/9612014v1*, 1996.

[27] Y. Maeda, Y. Futamura, and T. Sakurai. Stochastic estimation method of eigenvalue density for nonlinear eigenvalue problem on the complex plane. *JSIAM Letters*, 3:61–64, 2011.

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

[29] T. Mizusaki, K. Kaneko, M. Honma, and T. Sakurai. Filter diagonalization of shell-model calculations. *Phys. Rev. C*, 82(2):10, 2010.

[30] C. B. Moler and G. W. Stewart. An Algorithm for Generalized Matrix Eigenvalue Problems. *SIAM J. Num. Anal.*, 10(2):241–256, 1973.

[31] M. Naito, H. Tadano, and T. Sakurai. A modified Block IDR(s) method for computing high accuracy solutions. *JSIAM Letters*, 4:25–28, 2012.

[32] A. A. Nikishin and A. Y. Yeremin. Variable Block CG Algorithms for Solving Large Sparse Symmetric Positive Definite Linear Systems on Parallel Computers, I: General Iterative Scheme. *SIAM J. Matrix Anal. Appl.*, 16:1135–1153, 1995.

[33] H. Ohno, Y. Kuramashi, T. Sakurai, and H. Tadano. A quadrature-based eigensolver with a Krylov subspace method for shifted linear systems for Hermitian eigenproblems in lattice QCD. *JSIAM Letters*, 2:115–118, 2010.

[34] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra Appl.*, 29:293–322, 1980. Special Volume Dedicated to Alson S. Householder.

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

[36] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. SIAM, Philadelphia, PA, 2nd edition, 2011.

[37] T. Sakurai and H. Tadano. CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems. *Hokkaido Math. J.*, 36:745–757, 2007.

[38] T. Sakurai, J. Asakura, H. Tadano, and T. Ikegami. Error analysis for a matrix pencil of Hankel matrices with perturbed complex moments. *JSIAM Letters*, 1:76–79, 2009.

[39] T. Sakurai and H. Sugiura. A projection method for generalized eigenvalue problems using numerical integration. *J. Comput. Appl. Math.*, 159(1):119–128, 2003.

[40] T. Schmelzer. Block Krylov methods for Hermitian linear systems, 2004. Diploma thesis, Department of Mathematics, University of Kaiserslautern, Germany.

[41] K. Senzaki, H. Tadano, T. Sakurai, and Z. Bai. A Method for Profiling the Distribution of Eigenvalues Using the AS Method. *Taiwanese Journal of Mathematics*, 14(3A):839–853, 2010.

[42] V. Simoncini, V. Simoncini, E. Gallopoulos, and E. Gallopoulos. An Iterative Method for Nonsymmetric Systems with Multiple Right-Hand Sides. *SIAM J. Sci. Comput*, 16:917–933, 1995.

[43] D. C. Sorensen. Implicit Application of Polynomial Filters in a K-step Arnoldi Method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[44] G. W. Stewart. A Krylov–Schur Algorithm for Large Eigenproblems. *SIAM J. Matrix Anal. Appl.*, 23(3):601–614, 2001.

[45] Z. Strakoš and P. Tichý. On error estimation in the conjugate gradient method and why it works in finite precision computations. *ETNA. Electronic Transactions on Numerical Analysis [electronic only]*, 13:56–80, 2002.

[46] Z. Strakoš and P. Tichý. Error Estimation in Preconditioned Conjugate Gradients. *BIT Numerical Mathematics*, 45(4):789–817, 2005.

[47] Z. Strakoš and P. Tichý. On efficient numerical approximation of the bilinear form $\boldsymbol{c}^* A^{-1} \boldsymbol{b}$. *SIAM J. Sci. Comput.*, 33:565–587, 2011.

[48] M. Sugihara and K. Murota. *Theoretical Numerical Linear Algebra.* Iwatani Shoten, Tokyo, Japan, 2009. (in Japanese).

[49] H. Tadano, T. Sakurai, and Y. Kuramashi. Block BiCGGR: a new Block Krylov subspace method for computing high accuracy solutions. *JSIAM Letters*, 1:44–47, 2009.

[50] R. Takayama, T. Hoshi, T. Sogabe, S.-L. Zhang, and T. Fujiwara. Linear algebraic calculation of the Green's function for large-scale electronic structure theory. *Phys. Rev. B*, 73(16):165108, 2006.

[51] T. Watanabe, Y. Inadomi, H. Umeda, K. Fukuzawa, S. Tanaka, T. Nakano, and U. Nagashima. Fragment Molecular Orbital (FMO) and FMO-MO Calculations of DNA: Accuracy Validation of Energy and Interfragment Interaction Energy. *Journal of Computational and Theoretical Nanoscience*, 6(6):1328–1337, 2009.

[52] S. Yamamoto, T. Sogabe, T. Hoshi, S.-L. Zhang, and T. Fujiwara. Shifted Conjugate-Orthogonal-Conjugate-Gradient Method and Its Application to Double Orbital Extended Hubbard Model(Condensed matter: electronic structure and electrical, magnetic, and optical properties). *Journal of the Physical Society of Japan*, 77(11):114713–1–114713–8, 2008-11-15.

[53] I. Yamazaki, H. Tadano, T. Sakurai, and T. Ikegami. Performance comparison of parallel eigensolvers based on a contour integral method and a Lanczos method. *Parallel Computing*, 39(6-7):280–290, 2013.

[54] S. Yokota and T. Sakurai. A projection method for nonlinear eigenvalue problems using contour integrals. *JSIAM Letters*, 5:41–44, 2013.

List of Publications

[55] L. Du, Y. Futamura, and T. Sakurai. Block conjugate gradient type methods for the approximation of bilinear form $C^H A^{-1} B$. *Computers & Mathematics with Applications*, 66(12):2446–2455, 2014.

[56] Y. Futamura, T. Sakurai, S. Furuya, and J.-I. Iwata. Efficient Algorithm for Linear Systems Arising in Solutions of Eigenproblems and Its Application to Electronic-Structure Calculations. In M. Daydé, O. Marques, and K. Nakajima, editors, *High Performance Computing for Computational Science - VECPAR 2012*, volume 7851 of *Lecture Notes in Computer Science*, pages 226–235. Springer Berlin Heidelberg, 2013.

[57] Y. Futamura, H. Tadano, and T. Sakurai. Parallel stochastic estimation method of eigenvalue distribution. *JSIAM Letters*, 2:127–130, 2010.

[58] Y. Maeda, Y. Futamura, and T. Sakurai. Stochastic estimation method of eigenvalue density for nonlinear eigenvalue problem on the complex plane. *JSIAM Letters*, 3:61–64, 2011.

[59] Y. Nagai, Y. Shinohara, Y. Futamura, Y. Ota, and T. Sakurai. Numerical Construction of a Low-Energy Effective Hamiltonian in a Self-Consistent Bogoliubov–de Gennes Approach of Superconductivity. *Journal of the Physical Society of Japan*, 82(9):094701, 2013.

[60] T. Sakurai, Y. Futamura, and H. Tadano. Efficient Parameter Estimation and Implementations of a Contour Integral-Based Eigensolver. *J. Algo. Comput. Tech.*, 7(3):249–269, 2013.

[61] K. Yamamoto, Y. Maeda, Y. Futamura, and T. Sakurai. Adaptive Parallel Algorithm for Stochastic Estimation of Nonlinear Eigenvalue Density. *IPSJ Transactions on Advanced Computing Systems*, 5(3):22–29, 2012. (in Japanese).

[62] T. Yano, Y. Futamura, and T. Sakurai. Multi-GPU Scalable Implementation of a Contour-Integral-Based Eigensolver for Real Symmetric

Dense Generalized Eigenvalue Problems. In *Proceedings of 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 121–127, 2013.