An Error Resilience Strategy of a Complex Moment-Based Eigensolver

Akira Imakura, Yasunori Futamura, and Tetsuya Sakurai

Abstract Recently, complex moment-based eigensolvers have been actively developed in highly parallel environments to solve large and sparse eigenvalue problems. In this paper, we provide an error resilience strategy of a Rayleigh–Ritz type complex moment-based parallel eigensolver for solving generalized eigenvalue problems. Our strategy is based on an error bound of the eigensolver in the case that soft-errors like bit-flip occur. Using the error bound, we achieve an inherent error resilience of the eigensolver that does not require standard checkpointing and replication techniques in the most time-consuming part.

1 Introduction

In this paper, we consider complex moment-based eigensolvers for computing all eigenvalues located in a certain region and their corresponding eigenvectors for a generalized eigenvalue problem of the following form

$$A\boldsymbol{x}_i = \lambda_i B\boldsymbol{x}_i, \quad \boldsymbol{x}_i \in \mathbb{C}^n \setminus \{\boldsymbol{0}\}, \quad \lambda_i \in \Omega \subset \mathbb{C}, \tag{1}$$

where $A, B \in \mathbb{C}^{n\times n}$ and the matrix pencil $zB - A$ are assumed to be diagonalizable and nonsingular for any z on the boundary of Ω. Let m be the number of target eigenpairs and X_Ω be an $n \times m$ matrix, whose columns are the target eigenvectors, i.e., $X_\Omega := [\boldsymbol{x}_i | \lambda_i \in \Omega]$.

For solving the generalized eigenvalue problem (1), Sakurai and Sugiura have proposed a projection type method that uses certain complex moment matrices constructed by a contour integral in 2003 [13]. Thereafter, several researchers have actively studied improvements and related eigensolvers based on the complex moment-based eigensolver [5–8, 12, 14, 17]. The concepts of Sakurai and Sugiura have also been extended to solve nonlinear eigenvalue problems [1–3, 18].

A. Imakura (✉) • Y. Futamura • T. Sakurai
University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan
e-mail: imakura@cs.tsukuba.ac.jp; futamura@cs.tsukuba.ac.jp; sakurai@cs.tsukuba.ac.jp

Recently, we analyzed error bounds of the Rayleigh–Ritz type complex moment-based eigensolver called the block SS–RR method [9]. In this paper, we apply the results of the analyses to the case that soft-errors like bit-flip occur. Using the error bound, we provide an error resilience strategy which does not require standard checkpointing and replication techniques in the most time-consuming part of the eigensolver.

The remainder of this paper is organized as follows. In Sect. 2, we briefly describe the basic concepts of the complex moment-based eigensolvers. In Sect. 3, we introduce the algorithm of the block SS–RR method and the results of its error bounds. We also introduce the parallel implementation of the block SS–RR method in Sect. 3. In Sect. 4, we propose an error resilience strategy for the block SS–RR method. In Sect. 5, we show some numerical results and we present conclusions in Sect. 6.

Throughout, the following notations are used. Let $V = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_L] \in \mathbb{C}^{n\times L}$, then $\mathscr{R}(V)$ is the range space of the matrix V, and is defined by $\mathscr{R}(V) := \mathrm{span}\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_L\}$. In addition, for $A \in \mathbb{C}^{n\times n}$, $\mathscr{K}_k^{\square}(A, V)$ are the block Krylov subspaces, $\mathscr{K}_k^{\square}(A, V) = \mathscr{R}([V, AV, \ldots, A^{k-1}V])$.

2 Complex Moment-Based Eigensolvers

As a powerful algorithm for solving the generalized eigenvalue problem (1), Sakurai and Sugiura have proposed the complex moment-based eigensolver in 2003 [13]. This is called the SS–Hankel method. To solve (1), they introduced the rational function

$$r(z) := \widetilde{\boldsymbol{v}}^{\mathrm{H}}(zB - A)^{-1}B\boldsymbol{v}, \quad \boldsymbol{v}, \widetilde{\boldsymbol{v}} \in \mathbb{C}^n \setminus \{\boldsymbol{0}\}, \tag{2}$$

whose poles are the eigenvalues λ of the matrix pencil $zB - A$. They then considered computing all poles located in Ω.

All poles located in a certain region of a meromorphic function can be computed by the algorithm in [11], which is based on Cauchy's integral formula,

$$r(a) = \frac{1}{2\pi \mathrm{i}} \oint_{\Gamma} \frac{r(z)}{z - a} \mathrm{d}z,$$

where Γ is the positively oriented Jordan curve (i.e., the boundary of Ω). By applying the algorithm in [11] to the rational function (2), the target eigenpairs $(\lambda_i, \boldsymbol{x}_i), \lambda_i \in \Omega$ of the generalized eigenvalue problem (1) are obtained by solving the generalized eigenvalue problem:

$$H_M^{<}\boldsymbol{u}_i = \theta_i H_M \boldsymbol{u}_i.$$

Here, H_M and $H_M^<$ are small $M \times M$ Hankel matrices of the form

$$H_M := \begin{pmatrix} \mu_0 & \mu_1 & \cdots & \mu_{M-1} \\ \mu_1 & \mu_2 & \cdots & \mu_M \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{M-1} & \mu_M & \cdots & \mu_{2M-2} \end{pmatrix}, \quad H_M^< := \begin{pmatrix} \mu_1 & \mu_2 & \cdots & \mu_M \\ \mu_2 & \mu_3 & \cdots & \mu_{M+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_M & \mu_{M+1} & \cdots & \mu_{2M-1} \end{pmatrix},$$

whose entries consist of the following complex moments

$$\mu_k := \frac{1}{2\pi \mathrm{i}} \oint_\Gamma z^k r(z) \mathrm{d}z.$$

For details, refer to [13].

For more accurate eigenpairs, improvement of the SS–Hankel method has been proposed [14]. This improvement is based on the Rayleigh–Ritz procedure and is called the SS–RR method. Block variants of the SS–Hankel method and the SS–RR method have also been proposed [5, 6] for higher stability of the algorithms, specifically when multiple eigenvalues exist in Ω. These are called the block SS–Hankel method and the block SS–RR method, respectively. An Arnoldi-based interpretation of the complex moment-based eigensolvers and the resulting algorithm have also been proposed [8]. The algorithm is named the block SS–Arnoldi method.

As another approach of the complex moment-based eigensolver, Polizzi has proposed the FEAST eigensolver for Hermitian generalized eigenvalue problems in 2009 [12] and then developed it further [17]. The FEAST eigensolver is an accelerated subspace iteration-type method, and a single iteration is closely connected to a special case of the block SS–RR method with $M = 1$.

The relationship among these complex moment-based eigensolvers was analyzed in [10].

3 The Block SS–RR Method

In this section, we introduce the algorithm of the block SS–RR method and the results of its error bounds.

3.1 Algorithm of the Block SS–RR Method

Let $L, M \in \mathbb{N}$ be input parameters. Also let $V \in \mathbb{C}^{n \times L}$ be an input matrix, e.g., a random matrix. Then, we define an $n \times LM$ matrix

$$S := [S_0, S_1, \dots, S_{M-1}],$$

where

$$S_k := \frac{1}{2\pi \mathrm{i}} \oint_{\Gamma} z^k (zB - A)^{-1} BV \mathrm{d}z. \tag{3}$$

Then, we have the following theorem; see e.g., [9].

Theorem 1 *Let m be the number of eigenvalues of* (1) *and* $\mathrm{rank}(S) = m$. *Then, we have*

$$\mathscr{R}(S) = \mathscr{R}(X_{\Omega}) = \mathrm{span}\{\boldsymbol{x}_i | \lambda_i \in \Omega\}.$$

Theorem 1 indicates that the target eigenpairs $(\lambda_i, \boldsymbol{x}_i), \lambda_i \in \Omega$ can be obtained by the Rayleigh–Ritz procedure with $\mathscr{R}(S)$. The above forms the basis of the block SS–RR method [5]. Continuous integration (3) is approximated by some numerical integration rule such as the N-point trapezoidal rule with $N \geq M - 1$. The approximated matrix $\widehat{S}_k$ is expressed as

$$S \approx \widehat{S}_k := \sum_{j=1}^{N} \omega_j z_j^k (z_j B - A)^{-1} BV, \tag{4}$$

where z_j are the quadrature points, and ω_j are the corresponding weights. We also set

$$S \approx \widehat{S} := [\widehat{S}_0, \widehat{S}_1, \dots, \widehat{S}_{M-1}]. \tag{5}$$

Here, (z_j, ω_j) are required to satisfy

$$\sum_{j=1}^{N} \omega_j z_j^k \begin{cases} \neq 0, & (k = -1) \\ = 0, & (k = 0, 1, \dots, N-2) \end{cases}. \tag{6}$$

The algorithm of the block SS–RR method with numerical integration is consist of the following three steps:

Step 1. Solve N linear systems with L right-hand sides of the form:

$$(z_j B - A) W_j = BV, \quad j = 1, 2, \dots, N. \tag{7}$$

Step 2. Construct the matrix $\widehat{S}$ by (4) and (5), where $\widehat{S}_k$ can be rewritten by using W_j as follows:

$$\widehat{S}_k = \sum_{j=1}^{N} \omega_j z_j^k W_j, \quad k = 1, 2, \dots, M-1. \tag{8}$$

Algorithm 1 The block SS–RR method

```
Input: $L, M, N \in \mathbb{N}, V \in \mathbb{C}^{n \times L}, (z_j, \omega_j), j = 1, 2, \dots, N$
Output: Approximate eigenpairs $(\widehat{\lambda}_i, \widehat{\boldsymbol{x}}_i)$ for $i = 1, 2, \dots, LM$
1: Solve $W_j = (z_j B - A)^{-1} BV$ for $j = 1, 2, \dots, N$
2: Compute $\widehat{S}_k = \sum_{j=1}^{N} \omega_j z_j^k W_j$ for $k = 0, 1, \dots, M-1$ and set $\widehat{S} = [\widehat{S}_0, \widehat{S}_1, \dots, \widehat{S}_{M-1}]$
3: Compute the orthogonalization of $\widehat{S}$ : $Q = \mathrm{orth}(\widehat{S})$
4: Compute eigenpairs $(\theta_i, \boldsymbol{u}_i)$ of the generalized eigenvalue problem $Q^{\mathrm{H}} A Q \boldsymbol{u}_i = \theta_i Q^{\mathrm{H}} B Q \boldsymbol{u}_i$, and
   $(\widehat{\lambda}_i, \widehat{\boldsymbol{x}}_i) = (\theta_i, Q\boldsymbol{u}_i)$ for $i = 1, 2, \dots, LM$
```

Step 3. Compute approximate eigenpairs by the Rayleigh–Ritz procedure as follows. Solve

$$Q^{\mathrm{H}} A Q \boldsymbol{u}_i = \theta_i Q^{\mathrm{H}} B Q \boldsymbol{u}_i,$$

and $(\widehat{\lambda}_i, \widehat{\boldsymbol{x}}_i) = (\theta_i, Q\boldsymbol{u}_i)$, where $Q = \mathrm{orth}(\widehat{S})$.

The algorithm of the block SS–RR method is summarized as Algorithm 1.

In practice, in order to reduce the computational costs and to improve accuracy, the matrix $\widehat{S}$ is replaced with a low-rank approximation obtained from the singular value decomposition. Moreover, z^k is scaled for improving numerical stability. For details, refer to [5, 15].

The block SS–RR method has some parameters such as L, M, N, and these parameters strongly affect the performance of the method. In the current version of the software of the block SS–RR method, z-pares ver.0.9.6a [19], $N = 32, M = 16$ are used as the default parameters. The parameter L is usually set such that $LM = 2m$, where m is the number of the target eigenvalues in Ω. The optimal parameters depend on the eigenvalue distribution, the required accuracy, computational environments and so on. For the details of how to set the parameters achieving good performance, refer to [15].

3.2 *Error Bounds of the Block SS–RR Method*

In [9], the error bounds of the block SS–RR method are analyzed. Here, we briefly introduce the results.

Let the matrix pencil $zB - A$ be diagonalizable, i.e.,

$$Y^{-1}(zB - A)X = z \begin{bmatrix} I_r & \\ & O_{n-r} \end{bmatrix} - \begin{bmatrix} \Lambda_r & \\ & I_{n-r} \end{bmatrix},$$

where $\Lambda_r := \mathrm{diag}(\lambda_1, \lambda_2, \dots, \lambda_r)$ is a diagonal matrix, and $Y^{-1} := [\widetilde{\boldsymbol{y}}_1, \widetilde{\boldsymbol{y}}_2, \dots, \widetilde{\boldsymbol{y}}_n]^{\mathrm{H}}$ and $X := [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n]$ are nonsingular matrices. The generalized eigenvalue problem $A\boldsymbol{x}_i = \lambda_i B\boldsymbol{x}_i$ has $r := \mathrm{rank}(B)$ finite eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_r$ and

$n-r$ infinite eigenvalues. The vectors $\widetilde{\boldsymbol{y}}_i$ and $\boldsymbol{x}_i$ are the corresponding left and right eigenvectors, respectively. The filter function

$$f(\lambda_i) := \sum_{j=1}^{N} \frac{\omega_j}{z_j - \lambda_i}, \tag{9}$$

is commonly used for analysis of the complex moment-based eigensolvers [6, 16, 17]. Using this filter function, the matrix $\widehat{S}$ can be written as

$$\widehat{S} = \left(X_r f(\Lambda_r)\widetilde{X}_r^{\mathrm{H}}\right)[V, CV, \dots, C^{M-1}V], \quad C := X_r \Lambda_r \widetilde{X}_r^{\mathrm{H}},$$

where $\Lambda_r := \mathrm{diag}(\lambda_1, \lambda_2, \dots, \lambda_r), X_r := [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_r], \widetilde{X}_r := [\widetilde{\boldsymbol{x}}_1, \widetilde{\boldsymbol{x}}_2, \dots, \widetilde{\boldsymbol{x}}_r]$ and $X^{-1} = \widetilde{X}^{\mathrm{H}} = [\widetilde{\boldsymbol{x}}_1, \widetilde{\boldsymbol{x}}_2, \dots, \widetilde{\boldsymbol{x}}_n]^{\mathrm{H}}$. The error bound of the block SS–RR method in [9] can be simplified under some assumption on V as follows.

Theorem 2 *Let $(\lambda_i, \boldsymbol{x}_i)$ be the exact eigenpairs of the matrix pencil $zB - A$. Assume that $f(\lambda_i)$ are ordered in decreasing order of magnitude $|f(\lambda_i)| \geq |f(\lambda_{i+1})|$. Define $\mathscr{P}$ as the orthogonal projector onto the subspace $\mathscr{R}(\widehat{S})$. Then, we have*

$$\|(I - \mathscr{P})\boldsymbol{x}_i\|_2 \leq \alpha\beta_i \left|\frac{f(\lambda_{LM+1})}{f(\lambda_i)}\right|,$$

where $\alpha = \|X\|_2\|X^{-1}\|_2$, β_i depends on the angle between the subspace $\mathscr{K}_M^{\square}(C, V)$ and each eigenvector $\boldsymbol{x}_i$.

Moreover, in [9], the error bound has been proposed for the case in which the solution of the linear system for the j'-th quadrature point is contaminated as follows:

$$(z_{j'}B - A)^{-1}BV + E, \tag{10}$$

where $E \in \mathbb{C}^{n \times L}$ is an error matrix of $\mathrm{rank}(E) = L' \leq L$. Because of the contaminated solution (10), the matrix $\widehat{S}$ is also contaminated. We define the contaminated matrix as $\widehat{S}'$. The error bound of the block SS–RR method with the contaminated matrix in [9] can also be simplified under some assumption on V as follows.

Theorem 3 *Let $(\lambda_i, \boldsymbol{x}_i)$ be the exact eigenpairs of the matrix pencil (A, B). Assume that $f(\lambda_i)$ are ordered in decreasing order of magnitude $|f(\lambda_i)| \geq |f(\lambda_{i+1})|$. Define $\mathscr{P}'$ as the orthogonal projector onto the subspace $\mathscr{R}(\widehat{S}')$. Then, we have*

$$\|(I - \mathscr{P}')\boldsymbol{x}_i\|_2 \leq \alpha\beta_i' \left|\frac{f(\lambda_{LM-L'+1})}{f(\lambda_i)}\right|,$$

where $\alpha = \|X\|_2\|X^{-1}\|_2$, β_i' depends on the error matrix E and the angle between the subspace $\mathscr{K}_M^{\square}(C, V)$ and each eigenvector $\boldsymbol{x}_i$.

Here, we note that the values β_i' is not equivalent to β_i, since β_i' depends on error matrix E and the contaminated quadrature point j'. β_i' may become larger for λ_i near $z_{j'}$ than others, specifically for the case where $L' = L$. For more details of these theorems, refer to [9].

3.3 Parallel Implementation of the Block SS–RR Method

The most time-consuming part of the block SS–RR method is to solve N linear systems with L right-hand sides (7) in Step 1. For solving the linear systems, the block SS–RR method has hierarchical parallelism; see Fig. 1.

Layer 1. Contour paths can be performed independently.
Layer 2. The linear systems can be solved independently.
Layer 3. Each linear system can be solved in parallel.

By making the hierarchical structure of the algorithm responsive to the hierarchical structure of the architecture, the block SS–RR method is expected to achieve high scalability.

Because Layer 1 can be implemented completely without communications, here we describe a basic parallel implementation of the block SS–RR method for one contour path. Let P be the number of MPI processes used for one contour path. Here

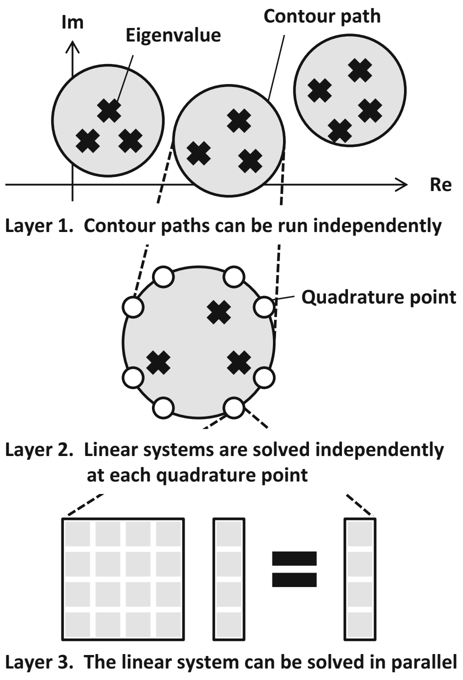
Im
Eigenvalue
Contour path
Re
Layer 1. Contour paths can be run independently
Quadrature point
Layer 2. Linear systems are solved independently at each quadrature point
Layer 3. The linear system can be solved in parallel

Fig. 1 Hierarchical structure of the block SS–RR method

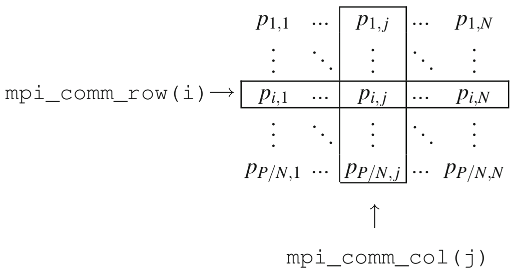

Fig. 2 Processes grid and MPI sub-communicators

we assume $\mathrm{mod}(P, N) = 0$ for simplicity and consider two dimensional processes grid, i.e., $p_{i,j}, i = 1, 2, \ldots, P/N, j = 1, 2, \ldots, N$. Then, we also define MPI sub-communicators for N MPI processes $p_{i,j}, j = 1, 2, \ldots, N$ as `mpi_comm_row(i)` ($i = 1, 2, \ldots, P/N$) and for P/N MPI processes $p_{i,j}, i = 1, 2, \ldots, P/N$ as `mpi_comm_col(j)` ($j = 1, 2, \ldots, N$); see Fig. 2.

3.3.1 Parallel Implementation for Step 1

In Step 1, we need to solve N linear systems (7). Because these linear systems are independent of j (index of quadrature point), we can independently solve these N linear systems in N parallel. Each linear system $(z_j B - A)W_j = V$ is solved by some parallel linear solver on the MPI sub-communicator `mpi_comm_col(j)` in parallel.

In this implementation, the coefficient matrices A, B and the input matrix V require to be distributed to P/N MPI processes in each MPI sub-communicator `mpi_comm_col(j)` with N redundant. As a result, each solution W_j of the linear system is also distributed to P/N MPI processes in the MPI sub-communicator `mpi_comm_col(j)`.

3.3.2 Parallel Implementation for Step 2

Let $W_j^{(i)}, (i = 1, 2, \ldots, P/N, j = 1, 2, \ldots, N)$ be the distributed sub-matrix of W_j, which are stored by the MPI process $p_{i,j}$. Then, for constructing the matrix $\widehat{S}_k$ (8), we independently compute

$$W_{j,k}^{(i)} = \omega_j z_j^k W_j^{(i)}$$

in each MPI process without communication. Then, we perform `mpi_allreduce` on the MPI sub-communicator `mpi_comm_row(i)` with N MPI processes in P/N

parallel as follows:

$$\widehat{S}_k^{(i)} = \sum_{j=1}^{N} W_{j,k}^{(i)}, \quad k = 0, 1, \ldots, M-1.$$

We also set

$$\widehat{S}^{(i)} = [\widehat{S}_0^{(i)}, \widehat{S}_1^{(i)}, \ldots, \widehat{S}_{M-1}^{(i)}],$$

where $\widehat{S}^{(i)}$ are the sub-matrix of $\widehat{S}$, which is redundantly stored by the MPI processes $p_{i,j}, j = 1, 2, \ldots, N$. In this implementation, the matrix $\widehat{S}$ are distributed in P/N MPI processes in each MPI sub-communicator `mpi_comm_row(i)` with N redundant.

3.3.3 Parallel Implementation for Step 3

We have two choices for parallel implementation for Step 3. The first choice is that all P MPI processes perform the orthogonalization of $\widehat{S}$ and the Rayleigh–Ritz procedure. This choice makes it possible to work all MPI processes we can use; however, it needs to redistribution of the matrices A, B and $\widehat{S}$.

The second choice is that only P/N MPI processes in the MPI sub-communicator `mpi_comm_col(j)` perform this calculation. In this case, only P/N MPI processes work and the others are just redundant; however, redistribution of the matrices A, B and $\widehat{S}$ does not be required.

4 An Error Resilience Strategy of the Block SS–RR Method

With the recent development of high-performance computer, systems scale is drastically increasing. In such situation, fault management is considered to play an important role in large scale application. The fault can be classified to hardware fault and software fault. Here, we focus on software fault like bit-flip.

The most standard software fault tolerance techniques are checkpointing techniques. The checkpointing techniques save all correct data at some interval, and if some fault is detected then it restarts with the last correct data. These are efficient for the case that data size required to save is small and that interval between each checkpoint is small. On the other hand, large data size causes large I/O costs and large interval causes large recalculation costs when fault occurs.

The replication techniques are also very basic software fault tolerance techniques. Its basic idea is shown below. Let P be the number of MPI processes we can use and K be the number of redundancies. Firstly, we split MPI communicator into each P/K MPI processes. The replication techniques restrict the parallelism to P/K, i.e., calculation is independently performed by P/K MPI processes in each

MPI sub-communicator. Then, the correct solution is selected from K solutions by e.g. a majority vote. These are efficient when the number of MPI processes is large such that target calculation does not show good scalability. However, if the target calculation shows good scalability, the replication techniques largely increase the execution time even if fault does not occur.

In this section, we consider an error resilience strategy of the block SS–RR method that can use all the MPI processes for the most time-consuming part, i.e., to solve the N linear systems (7) in Step 1 and avoid resolving them even if fault occurs. Here, we assume the following software fault:

- Let $a \in \mathbb{F}$ be the correct value, where $\mathbb{F}$ is the set of floating point numbers. The fault occurs as the numerical error as follows:

$$a' \leftarrow a + e, \quad e \in \mathbb{F}, \tag{11}$$

  where $a' \in \mathbb{F}$ is the contaminated value. Here, a, a', e are not "`Inf`" or "`Nan`".
- Unlike hardware faults, remaining calculation are correctly performed with the contaminated values.

4.1 *Error Resilience Strategy*

As shown in Sect. 3, the algorithm of the block SS–RR method and its parallel implementation can be divided into three steps: solving the linear systems, the numerical integration and the Rayleigh–Ritz procedure. Here, we consider error resilience of each step.

4.1.1 Error Resilience Strategy for Step 1

Step 1 is the most time-consuming part and also the most scalable part of the block SS–RR method. Therefore, standard checkpointing and replication techniques may not be efficient for computational costs. Hence, we introduce an alternative strategy to standard checkpointing and replication techniques for computational costs.

When fault occurs in Step 1, some kind of value(s) in calculation are replaced as (11) due to the fault. Then, the contamination is propagated to all MPI processes in the same MPI sub-communicator `mpi_comm_col(j)` via communication. As a result, the solution of the linear system is replaced as

$$W'_{j'} \leftarrow W_{j'} + E, \quad E \in \mathbb{F}^{n \times L}, \quad \mathrm{rank}(E) = L, \tag{12}$$

when fault occurs in the MPI process $p_{i,j'}$ associated with the j'-th linear system.

Here, we reconsider Theorems 2 and 3. Theorem 2 implies that the error bound of the block SS–RR method is evaluated by the ratio of the magnitude of the filter

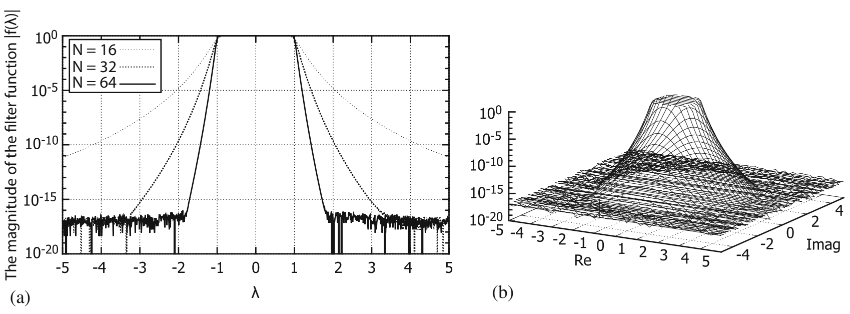

Fig. 3 Magnitude of filter function $|f(\lambda)|$ of the N-point trapezoidal rule with $N = 16, 32, 64$ for the unit circle region Ω. (**a**) On the real axis for $N = 16, 32, 64$. (**b**) On the complex plane for $N = 32$

function $|f(\lambda_i)|$ to the $(LM + 1)$-th largest $|f(\lambda_{LM+1})|$. The magnitude of the filter function $|f(\lambda_i)|$ of the N-point trapezoidal rule with $N = 16, 32, 64$ for the unit circle region Ω is shown in Fig. 3. The filter function has $|f(\lambda)| \approx 1$ inside the region Ω, $|f(\lambda))| \approx 0$ far from the region and $0 < |f(\lambda)| < 1$ outside but near the region. Because of Theorem 2 and the filter function, we usually set subspace size LM such that $|f(\lambda_{LM+1})| \approx 0$ to compute the target eigenpairs $(\lambda_i, \boldsymbol{x}_i)$, $\lambda_i \in \Omega$ with high accuracy.

Regarding the filter function, Theorem 3 implies that the accuracy of the block SS–RR method with the contaminated solution is evaluated by the ratio of the magnitude of the filter function $|f(\lambda_i)|$ to the $(LM - L + 1)$-th largest $|f(\lambda_{LM-L+1})|$. Of course, Theorem 3 support the case when fault occurs in Step 1 like (12). Therefore, if we consider the case that fault occurs in Step 1, we just set subspace size LM such that $|f(\lambda_{LM-L+1})| \approx 0$ in order to obtain the eigenpairs to high accuracy.

Here, we note that, when multiple faults occur in different quadrature points, i.e.,

$$W'_{j'_1} \leftarrow W_{j'_1} + E_1, \quad W'_{j'_2} \leftarrow W_{j'_2} + E_2, \quad E_1, E_2 \in \mathbb{F}^{n\times L}, \quad \mathrm{rank}(E_1) = \mathrm{rank}(E_2) = L,$$

then we can handle the fault in Step 1 by setting larger subspace LM such that $|f(\lambda_{LM-2L+1})| \approx 0$.

This is an error resilience strategy for Step 1, which makes it possible to use all MPI processes for computing the N linear systems (7) and to avoid resolving them even if fault occurs.

4.1.2 Error Resilience Strategy for Step 2

The computational cost for Step 2 is very small, and the data size is not exorbitant large. Therefore, we can apply checkpointing technique with small additional costs for Step 2.

4.1.3 Error Resilience Strategy for Step 3

As noted in Sect. 3.3, we have two choices for implementation of Step 3: to use all processes with redistribution and to replicate without redistribution. If the number of processes P is not so large such that this part shows good scalability, the first choice is better in terms of computational costs. If not, the second choice is better due to the costs of redistribution. In practice, we want to increase the number of processes P, if possible, during N linear systems, which is the most time-consuming part, shows good scalability. And computation of N independent linear systems is expected to have better scalability than one of the orthogonalization and the Rayleigh–Ritz procedure. Hence, we usually employ the second choice.

Therefore, we can apply replication technique without no additional costs for Step 3.

4.2 A Possibility of Development to Other Complex Moment-Based Eigensolvers

In Sect. 4, we proposed the error resilience strategy of the block SS–RR method which is based on the error analysis in [9]. Here, we consider a possibility of development of our strategy to other complex moment-based eigensolvers.

The proposed error resilient strategy is mainly based on Theorem 3 for the block SS–RR method. Similar theorems as Theorem 3 could be derived for other complex moment-based eigensolvers. One of the most important respects of Theorem 3 is that the subspace size LM should be larger than the rank of error matrix L', i.e., $LM > L'$. In the case of one linear solution is contaminated in the block SS–RR method with $M \geq 2$, the condition $LM > L \geq L'$ is always satisfied and this makes it possible to derive the proposed error resilient strategy.

For development of our strategy to other complex moment-based eigensolvers, we can expect that the proposed strategy is also utilized to other complex moment-based eigensolvers with high order complex moments such as the (block) SS–Hankel method and the block SS–Arnoldi method, although more detailed analyses and numerical experiments are required. Because these methods with $M > 2$ always satisfy the condition $LM > L \geq L'$ as well as the block SS–RR method.

On the other hand, the current proposed strategy may be difficult to recover the error of the complex moment-based eigensolvers only with low order complex moments such as the FEAST eigensolver [12, 17] and the Beyn method [3]. The subspace size of these methods is L which is the same as the number of right-hand side of the linear systems. This indicates that the rank of the error matrix reaches the subspace size in the worst case. In this case, our strategy can not recover the error.

5 Numerical Experiments

In this section, we experimentally evaluate the results of the error resilience strategy specifically for Step 1.

5.1 Example I

For the first example, we apply the block SS–RR method with and without soft-error in Step 1 to the following model problem

$$\begin{gathered} A\boldsymbol{x}_i = \lambda \boldsymbol{x}_i, \\ A = \mathrm{diag}(0.01, 0.11, 0.21, \ldots, 9.91) \in \mathbb{R}^{100\times 100}, \\ \lambda_i \in \Omega = [-1, 1], \end{gathered}$$

and evaluate its accuracy.

We evaluate the relation between accuracy with the number of subspace size LM. To evaluate this relation, we fixed the parameters as $L = 10$ and $N = 32$, and tested four cases $M = 1, 2, 3, 4$ ($LM = 10, 20, 30, 40$). For this example, we set Γ as the unit circle and the quadrature points as

$$z_j = \cos(\theta_j) + \mathrm{i}\sin(\theta_j), \quad \theta_j = \frac{2\pi}{N}\left(j - \frac{1}{2}\right)$$

for $j = 1, 2, \ldots, N$. We let fault occur at one of the following quadrature points,

$$z_{j'} = \begin{cases} z_1 = \cos\left(\frac{\pi}{32}\right) + \mathrm{i}\sin\left(\frac{\pi}{32}\right) \\ z_8 = \cos\left(\frac{15\pi}{32}\right) + \mathrm{i}\sin\left(\frac{15\pi}{32}\right) \\ z_{16} = \cos\left(\frac{31\pi}{32}\right) + \mathrm{i}\sin\left(\frac{31\pi}{32}\right) \end{cases}$$

The algorithm was implemented in MATLAB R2014a. The input matrix V and the error matrix E were set as different random matrices generated by the Mersenne Twister in MATLAB, and each linear system was solved by the MATLAB command "\".

We show in Table 1 the relation of the minimum and the maximum values of $\|\boldsymbol{r}_i\|_2$ in $\lambda_i \in \Omega$ with LM. Table 1(a) is for the case without fault and Table 1(b)–(d) are for the case when fault occurs in Step 1. We also show in Fig. 4 the residual 2-norm $\|\boldsymbol{r}_i\|_2 := \|A\boldsymbol{x}_i - \lambda_i B\boldsymbol{x}_i\|_2 / \|\boldsymbol{x}_i\|_2$ for the block SS–RR method with and without fault.

Table 1 shows that $\min_{\lambda_i\in\Omega} \|\boldsymbol{r}_i\|_2$ have approximately the same order as $|f(\lambda_{LM+1})|$ for the case without fault and as $|f(\lambda_{LM-L+1})|$ when fault occurs

Table 1 Relation of accuracy of the block SS–RR method with *LM* when fault occurs in Step 1

(a) Without fault			
M (LM)	$\lvert f(\lambda_{LM+1})\rvert$	$min_{\lambda_i\in\Omega}\lVert\boldsymbol{r}_i\rVert_2$	$max_{\lambda_i\in\Omega}\lVert\boldsymbol{r}_i\rVert_2$
1 (10)	4.21×10^{-1}	1.76×10^{-1}	1.34×10^{-1}
2 (20)	1.98×10^{-10}	2.29×10^{-10}	2.11×10^{-9}
3 (30)	5.06×10^{-16}	1.44×10^{-15}	1.20×10^{-14}
4 (40)	2.25×10^{-17}	2.03×10^{-15}	3.46×10^{-15}
(b) Fault occurs at z_1			
M (LM)	$\lvert f(\lambda_{LM-L+1})\rvert$	$min_{\lambda_i\in\Omega}\lVert\boldsymbol{r}_i\rVert_2$	$max_{\lambda_i\in\Omega}\lVert\boldsymbol{r}_i\rVert_2$
1 (10)	1.00×10^{0}	5.23×10^{-1}	8.23×10^{-1}
2 (20)	4.21×10^{-1}	1.84×10^{-1}	2.63×10^{-1}
3 (30)	1.98×10^{-10}	2.43×10^{-10}	1.63×10^{-8}
4 (40)	5.06×10^{-16}	6.57×10^{-15}	1.91×10^{-13}
(c) Fault occurs at z_8			
M (LM)	$\lvert f(\lambda_{LM-L+1})\rvert$	$min_{\lambda_i\in\Omega}\lVert\boldsymbol{r}_i\rVert_2$	$max_{\lambda_i\in\Omega}\lVert\boldsymbol{r}_i\rVert_2$
1 (10)	1.00×10^{0}	5.42×10^{-1}	7.99×10^{-1}
2 (20)	4.21×10^{-1}	1.04×10^{-1}	7.74×10^{-1}
3 (30)	1.98×10^{-10}	5.11×10^{-10}	4.57×10^{-9}
4 (40)	5.06×10^{-16}	5.16×10^{-15}	2.87×10^{-14}
(d) Fault occurs at z_{16}			
M (LM)	$\lvert f(\lambda_{LM-L+1})\rvert$	$min_{\lambda_i\in\Omega}\lVert\boldsymbol{r}_i\rVert_2$	$max_{\lambda_i\in\Omega}\lVert\boldsymbol{r}_i\rVert_2$
1 (10)	1.00×10^{0}	5.54×10^{-1}	7.85×10^{-1}
2 (20)	4.21×10^{-1}	4.11×10^{-1}	4.84×10^{-1}
3 (30)	1.98×10^{-10}	7.05×10^{-10}	4.96×10^{-9}
4 (40)	5.06×10^{-16}	3.71×10^{-15}	2.51×10^{-14}

in Step 1, respectively. Moreover, Fig. 4 shows that enough large subspace size ($LM = 40$ in this example) provides equally high accuracy independent of fault in Step 1.

5.2 *Example II*

For the second example, we apply the block SS–RR method with and without soft-error in Step 1 to the generalized eigenvalue problem AUNW9180 from ELSES matrix library [4]. The coefficient matrices A, B are 9180 dimensional real sparse symmetric matrices and B is also positive definite. We consider finding all eigenpairs $(\lambda_i, \boldsymbol{x}_i), \lambda_i \in \Omega = [0.119, 0.153]$. In this region, there exist 99 eigenvalues.

We set Γ as the ellipse (center: 0.131, semi-major axis: 0.012 and semi-minor axis: 0.0012), and the quadrature points as

$$z_j = 0.131 + 0.012\left(\cos(\theta_j) + 0.1\mathrm{i}\sin(\theta_j)\right),$$
$$\theta_j = \frac{2\pi}{N}\left(j - \frac{1}{2}\right)$$

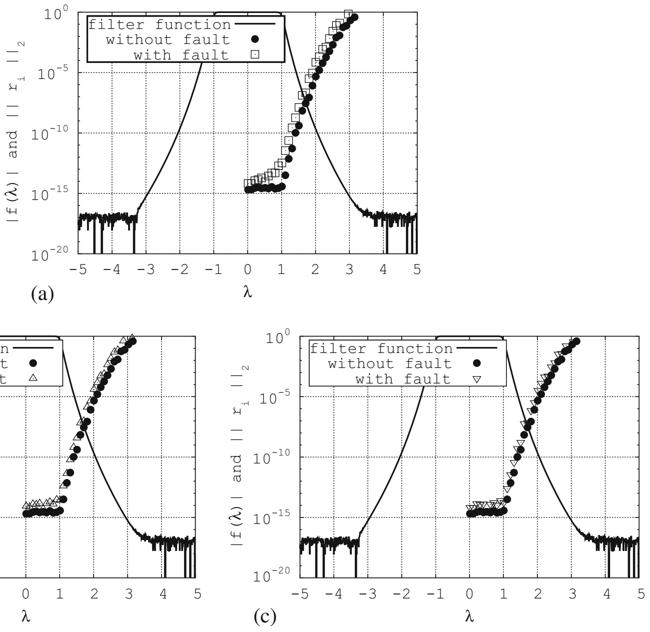

Fig. 4 Accuracy of the block SS–RR method with $L = 10, M = 4, N = 32$ when fault occurs in Step 1. (**a**) Fault occurs at z_1. (**b**) Fault occurs at z_8. (**c**) Fault occurs at z_{16}

for $j = 1, 2, \ldots, N$. We also set parameters as $L = 25, M = 8, N = 32$ for the case without fault and as $L = 25, M = 10, N = 32$ when fault occurs in Step 1.

The input matrix V and the error matrix E were set as different random matrices generated by the Mersenne Twister, and each linear system was solved by "cluster_sparse_solver" in Intel MKL. Here, we note that, in this numerical experiment, we solved only $N/2$ linear systems with multiple right-hand sides for $j = 1, 2, \ldots, N/2$, because the linear solution W_{N-j} can be constructed from W_j using a symmetric property of the problem.

The numerical experiments were carried out in double precision arithmetic on 8 nodes of COMA at University of Tsukuba. COMA has two Intel Xeon E5-2670v2 (2.5 GHz) and two Intel Xeon Phi 7110P (61 cores) per node. In this numerical experiment, we use only CPU part. The algorithm was implemented in Fortran 90 and MPI, and was executed with 8 [node] × 2 [process/node] × 8 [thread/process].

We show in Fig. 5 the residual 2-norm $\|\boldsymbol{r}_i\|_2 := \|A\boldsymbol{x}_i - \lambda_i B\boldsymbol{x}_i\|_2 / \|\boldsymbol{x}_i\|_2$ for the block SS–RR method with and without fault. This shows that, by increasing subspace size LM, the block SS–RR method with fault can achieve approximately the same accuracy as the case without fault.

Table 2 shows that the computation time of the block SS–RR method without fault using 1–16 processes and the computation time of the block SS–RR method

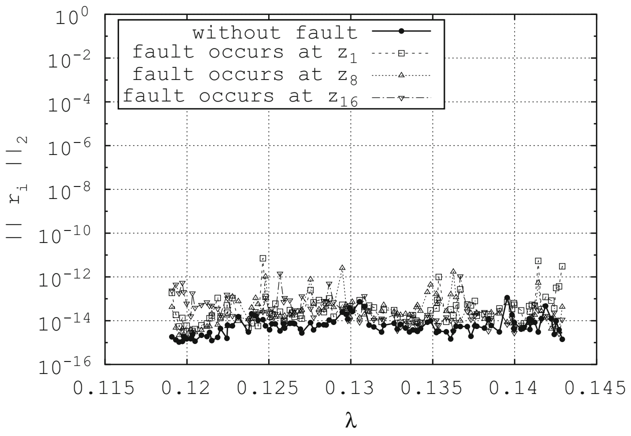

Fig. 5 Accuracy of the block SS–RR method with and without fault in Step 1 for AUNW9180

Table 2 Computation time of the block SS–RR method with and without fault

(a) without fault ($L = 25, M = 8, N = 32$)					
	Time [s]				
#process	*Step 1*	*Step 2*	*Step 3*	*MISC*	*Total*
1	2.14E+02	4.20E−05	4.92E−01	1.18E−01	2.15E+02
2	1.06E+02	1.05E−02	4.76E−01	7.87E−02	1.06E+02
4	5.30E+01	1.49E−02	4.77E−01	6.34E−02	5.36E+01
8	2.66E+01	2.05E−02	4.79E−01	5.73E−02	2.72E+01
16	1.34E+01	1.56E−02	4.78E−01	5.33E−02	1.39E+01
(b) with fault ($L = 25, M = 10, N = 32$)					
	Time [s]				
#process	*Step 1*	*Step 2*	*Step 3*	*MISC*	*Total*
16	1.37E+01	2.09E−02	6.44E−01	1.10E−02	1.44E+01

with fault using 16 processes. This result indicates that Step 1 of the SS–RR method is the most time-consuming. We can also observe from this result that the proposed strategy recovers software faults with very small additional computational costs.

6 Conclusion

In this paper, we investigated the error resilience strategy of the Rayleigh–Ritz type complex moment-based parallel eigensolver (the block SS–RR method) for solving generalized eigenvalue problems. Based on the analyses of the error bound of the method, we provided the error resilience strategy which does not require standard checkpointing and replication techniques in the most time-consuming and the most scalable part. From our numerical experiment, our strategy recovers software faults like bit-flip with small additional costs.

Acknowledgements The authors would like to thank the anonymous referees for their useful comments. This research was supported partly by University of Tsukuba Basic Research Support Program Type A, JST/CREST, JST/ACT-I (Grant No. JPMJPR16U6) and KAKENHI (Grant Nos. 25286097, 25870099, 17K12690). This research in part used computational resources of COMA provided by Interdisciplinary Computational Science Program in Center for Computational Sciences, University of Tsukuba.

References

1. Asakura, J., Sakurai, T., Tadano, H., Ikegami, T., Kimura, K.: A numerical method for nonlinear eigenvalue problems using contour integrals. JSIAM Lett. **1**, 52–55(2009)
2. Asakura, J., Sakurai, T., Tadano, H., Ikegami, T., Kimura, K.: A numerical method for polynomial eigenvalue problems using contour integral. Jpn. J. Ind. Appl. Math. **27**, 73–90 (2010)
3. Beyn, W.-J.: An integral method for solving nonlinear eigenvalue problems. Lin. Algor. Appl. **436**, 3839–3863 (2012)
4. ELSES matrix library, http://www.elses.jp/matrix/
5. Ikegami, T., Sakurai, T.: Contour integral eigensolver for non-Hermitian systems: a Rayleigh-Ritz-type approach. Taiwan. J. Math. **14**, 825–837 (2010)
6. Ikegami, T., Sakurai, T., Nagashima, U.: A filter diagonalization for generalized eigenvalue problems based on the Sakurai-Sugiura projection method. J. Comput. Appl. Math. **233**, 1927–1936 (2010)
7. Imakura, A., Sakurai, T.: Block Krylov-type complex moment-based eigensolvers for solving generalized eigenvalue problems. Numer. Algor. **75**, 413–433 (2017)
8. Imakura, A., Du, L., Sakurai, T.: A block Arnoldi-type contour integral spectral projection method for solving generalized eigenvalue problems. Appl. Math. Lett. **32**, 22–27 (2014)
9. Imakura, A., Du, L., Sakurai, T.: Error bounds of Rayleigh–Ritz type contour integral-based eigensolver for solving generalized eigenvalue problems. Numer. Algor. **71**, 103–120 (2016)
10. Imakura, A., Du, L., Sakurai, T.: Relationships among contour integral-based methods for solving generalized eigenvalue problems. Jpn. J. Ind. Appl. Math. **33**, 721–750 (2016)
11. Kravanja, P., Sakurai, T., van Barel, M.: On locating clusters of zeros of analytic functions. BIT **39**, 646–682 (1999)
12. Polizzi, E.: A density matrix-based algorithm for solving eigenvalue problems. Phys. Rev. B **79**, 115112 (2009)
13. Sakurai, T., Sugiura, H.: A projection method for generalized eigenvalue problems using numerical integration. J. Comput. Appl. Math. **159**, 119–128 (2003)
14. Sakurai, T. Tadano, H.: CIRR: a Rayleigh-Ritz type method with counter integral for generalized eigenvalue problems. Hokkaido Math. J. **36**, 745–757 (2007)
15. Sakurai, T., Futamura, Y., Tadano, H.: Efficient parameter estimation and implementation of a contour integral-based eigensolver. J. Algor. Comput. Technol. **7**, 249–269 (2014)
16. Schofield, G., Chelikowsky, J.R., Saad, Y.: A spectrum slicing method for the Kohn-Sham problem. Comput. Phys. Commun. **183**, 497–505 (2012)

17. Tang, P.T.P., Polizzi, E.: FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection. SIAM J. Matrix Anal. Appl. **35**, 354–390 (2014)
18. Yokota, S., Sakurai, T.: A projection method for nonlinear eigenvalue problems using contour integrals. JSIAM Lett. **5**, 41–44 (2013)
19. z-Pares, http://zpares.cs.tsukuba.ac.jp/