# A Parallel Bisection and Inverse Iteration Solver for a Subset of Eigenpairs of Symmetric Band Matrices

Hiroyuki Ishigami, Hidehiko Hasegawa, Kinji Kimura, and Yoshimasa Nakamura

**Abstract** The tridiagonalization and its back-transformation for computing eigenpairs of real symmetric dense matrices are known to be the bottleneck of the execution time in parallel processing owing to the communication cost and the number of floating-point operations. To overcome this problem, we focus on real symmetric band eigensolvers proposed by Gupta and Murata since their eigensolvers are composed of the bisection and inverse iteration algorithms and do not include neither the tridiagonalization of real symmetric band matrices nor its back-transformation. In this paper, the following parallel solver for computing a subset of eigenpairs of real symmetric band matrices is proposed on the basis of Murata's eigensolver: the desired eigenvalues of the target band matrices are computed directly by using parallel Murata's bisection algorithm. The corresponding eigenvectors are computed by using block inverse iteration algorithm with reorthogonalization, which can be parallelized with lower communication cost than the inverse iteration algorithm. Numerical experiments on shared-memory multi-core processors show that the proposed eigensolver is faster than the conventional solvers.

Hiroyuki Ishigami
Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto-shi, Kyoto, Japan, e-mail: hishigami@amp.i.kyoto-u.ac.jp
Present affiliation: Yahoo Japan Corporation, Akasaka 9-7-1, Minato-ku, Tokyo, Japan, e-mail: hishigam@yahoo-corp.jp

Hidehiko Hasegawa
Faculty of Library, Information and Media Science, University of Tsukuba, Kasuga 1-2, Tsukuba, e-mail: hasegawa@slis.tsukuba.ac.jp

Kinji Kimura
Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto-shi, Kyoto, Japan, e-mail: kkimur@amp.i.kyoto-u.ac.jp

Yoshimasa Nakamura
Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto-shi, Kyoto, Japan, e-mail: ynaka@i.kyoto-u.ac.jp

## 1 Introduction

A subset of eigenpairs of a real symmetric matrix, i.e. eigenvalues and the corresponding eigenvectors, is required in several applications, such as the vibration analysis, the molecular orbital computation, and the kernel principal component analysis. As the scale of the problems appearing in such applications becomes significantly larger, the parallel processing is crucial for reducing the overall execution time. As a result, there has been an increase in the demand for highly scalable algorithms for computing a subset of the eigenpairs in parallel processing.

Let us consider computing a subset of eigenpairs of a real symmetric dense matrix on the basis of the orthogonal similarity transformations. In general cases, the eigenpairs of the target symmetric matrix are computed through the following three phases: The first phase is *tridiagonalization*, which is to reduce the target matrix to symmetric tridiagonal form by orthogonal similarity transformations. The second phase is to compute eigenpairs of the symmetric tridiagonal matrix. The last phase is *back-transformation*, which is to compute the eigenvectors of the original matrix from the computed eigenvectors of the symmetric tridiagonal matrix by using the orthogonal transformations generated for the first phase.

The following two-step framework [7, 8] is widely used for the tridiagonalization: The first step is to reduce the symmetric dense matrix to symmetric band form and the second step is to reduce the symmetric band matrix to symmetric tridiagonal form. Several efficient parallel algorithms based on this framework has been proposed in [2, 3, 4, 7, 17], etc. However, it is pointed out in [4] that the band reduction in the second step and its corresponding back-transformation remains to be the bottleneck of the overall execution time in massively parallel processing if the eigenvectors are required.

To overcome the problem in the tridiagonalization and the corresponding back-transformation, the authors consider not reducing a real symmetric band matrix to the tridiagonal form, but directly computing the eigenpairs of the real symmetric band matrix. The bisection and inverse iteration algorithms [6] for real symmetric band matrices can be used for this purpose. It is to be noted that the bisection algorithm can compute the desired eigenvalues of the target matrices and the inverse iteration algorithm gives the corresponding eigenvectors. Their implementation for real symmetric band matrices is proposed in [12, 20]. As shown in numerical results on Section 4.1, the bisection algorithm proposed in [20] is faster than that proposed in [12]. Let us name the algorithm in [12] *Gupta's bisection algorithm* and that in [20] *Murata's bisection algorithm*, respectively. The inverse iteration algorithm with reorthogonalization is used for computing the corresponding eigenvectors both in [12, 20].

In this paper, the authors propose the following parallel algorithm for computing desired eigenpairs of real symmetric band matrices: the desired eigenvalues are computed by using the parallel implementation of Murata's bisection algorithm and the corresponding eigenvectors are computed by using *block inverse iteration algorithm with reorthogonalization (BIR algorithm)* [14], which is a variant of the inverse iteration algorithm with reorthogonalization. Then, the performance of the

proposed methods is evaluated through numerical experiments on shared-memory multi-core processors.

The rest of this paper is organized as follows. Section 2 gives a review of the bisection algorithms for computing eigenvalues of real symmetric band matrices proposed in [12, 20], and then shows their parallel implementation for shared-memory multi-core processors. Section 3 shows the inverse iteration algorithm with reorthogonalization and the BIR algorithm for computing eigenvectors of real symmetric band matrices, and then shows their parallel implementation for shared-memory multi-core processors. Section 4 shows results of numerical experiments on shared-memory multi-core processors to evaluate the performance of the proposed parallel algorithm, which computes eigenvalues of target matrices by using parallel Murata's bisection algorithm and computes the corresponding eigenvectors by using the BIR algorithm. We end with conclusions and future work in Section 5.

## 2 Eigenvalue Computation of Symmetric Band Matrices

The bisection algorithm [6] computes the desired eigenvalues of a real symmetric matrix by updating repeatedly the half-open intervals $(\mu^L, \mu^R]$. The computation of $\nu(\mu)$ is required for updating the intervals, where $\mu$ is a value in the intervals $(\mu^L, \mu^R]$ and $\nu(\mu)$ is the number of eigenvalues of the target matrix that are less than $\mu$, and is the most time-consuming part of the bisection algorithm.

In this section, we introduce Gupta's bisection algorithm [12] and Murata's bisection algorithm [20] for computing the desired eigenvalues of a real $n \times n$ symmetric band matrix $B$ with half-bandwidth $w$ and then show a parallel implementation of them. These two algorithms differ in the computation of $\nu(\mu)$.

### *2.1 Gupta's Bisection Algorithm*

Gupta's bisection algorithm employs *Martin-Wilkinson's Gaussian elimination* [19] for computing $\nu(\mu)$.

The computation of $\nu(\mu)$ by employing Martin-Wilkinson's Gaussian elimination is based on Sturm's theorem [21]. In this case, all the principal minor determinant of $B - \mu I$ is required. Martin-Wilkinson's Gaussian elimination is adequate for this purpose since economical partial pivoting strategy is implemented to it from the viewpoint of both the numerical stability and the number of floating-point operations. Note that the number of floating-point operations in Martin-Wilkinson's Gaussian elimination is $O(w^2 n)$.

Martin-Wilkinson's Gaussian elimination is mainly composed of the BLAS (Basic Linear Algebra Subprograms [9]) 1 routines such as vector operations. Whenever the partial pivoting occurs in Martin-Wilkinson's Gaussian elimination, the number of floating-point operations increases and, moreover, a pattern of the data access

changes. As a result, Gupta's bisection algorithm is difficult to achieve a high performance from the viewpoint of data reusability.

## 2.2 Murata's Bisection Algorithm

Murata's bisection algorithm [20] employs not only Martin-Wilkinson's Gaussian elimination but also the *LU* factorization without pivoting for computing $\nu(\mu)$.

The computation of $\nu(\mu)$ by employing the *LU* factorization without pivoting is based on Sylvester's law of inertia [21]. In this case, we consider a *LU* factorization without pivoting $B-\mu I = LU$, where $L$ is an $n \times n$ lower triangular matrix with lower bandwidth $w$ and $U$ is an $n \times n$ unit upper triangular matrix with upper bandwidth $w$. On the basis of Sylvester's law of inertia, $\nu(\mu)$ is equivalent to the number of negative values in diagonal elements of $L$.

The number of floating-point operations in Martin-Wilkinson's Gaussian elimination is about 3 times higher than that in the *LU* factorization without pivoting. In addition, the cache hit ratio of the *LU* factorization without pivoting is higher than that of Martin-Wilkinson's Gaussian elimination owing to absence of any pivoting. However, owing to both the rounding errors and the absence of any pivoting, the *LU* factorization without pivoting sometimes fails or the resulting elements of this factorization may be not correct even if accomplished.

As a result, Murata's bisection algorithm computes $\nu(\mu)$ in the following way: In the early stage of computing eigenvalues of $B$, Murata's bisection algorithm employs the *LU* factorization without pivoting for computing rapidly $\nu(\mu)$. In addition, if $\mu$ is significantly close to a certain eigenvalue, Murata's bisection algorithm employs Martin-Wilkinson's Gaussian elimination for computing accurately $\nu(\mu)$. Consequently, Murata's bisection algorithm is expected to be faster than Gupta's algorithm for computing the desired eigenvalues of $B$.

The several acceleration techniques for the *LU* factorization algorithm has been proposed. As shown in [13], the optimal implementation for vector processors is implemented to the *LU* factorization without pivoting and the overall execution time for Murata's bisection algorithm becomes shorter. On recent scalar processors with the large cache, the *LU* factorization had better to be implemented using the block algorithm for enforcing higher cache hit ratio. Hence, in this paper, the block *LU* factorization of real symmetric band matrices is introduced into Murata's bisection algorithm for the purpose of improving further its performance on shared-memory multi-core processors.

## 2.3 Parallel Implementation of Bisection Algorithm

In this paper, Murata's and Gupta's bisection algorithms are implemented on the basis of the `dstebz` routine [15], which is provided in LAPACK (Linear Algebra

**Algorithm 1** Parallel Bisection Algorithm for Symmetric Band Matrices

```
 1: function ParallelBandBisection(B, ℓ)
 2:     Set μ_1^L, μ_1^R                                  ▷ Use Gerschgorin theorem, etc.
 3:     k_b := 1, k_e := 1
 4:     repeat
 5:         !$omp parallel do private(μ_k)
 6:         do k := k_b to k_e                                              ▷ k_e ≤ ℓ
 7:             μ_k := (μ_k^L + μ_k^R)/2
 8:             Compute ν_B(μ_k)
 9:         end do
10:         Update the intervals μ_k^L, μ_k^R for k = k_b, ..., k_e and the indices k_b, k_e
11:     until All of the desired eigenvalues meets the stopping criteria
12:     return λ̃_k := (μ_k^L + μ_k^R)/2 for k = 1, ..., ℓ
13: end function
```

PACKage [1]) and computes the desired eigenvalues of real symmetric tridiagonal matrices. A pseudocode of their parallel implementation is shown in Algorithm 1, where $\ell$ is the number of the desired eigenvalues.

The computation of $\nu(\mu_k)$ on line 8 is applied different algorithms for Murata's and Gupta's bisection algorithm as mentioned in Section 2.1 and 2.2, respectively. In addition, the computation of $\nu(\mu_k)$ is performed in parallel with respect to each search point $\mu_k$ by employing the OpenMP directive shown in line 5.

Note that an initial interval $\mu_1^L$ and $\mu_1^R$ are set on line 2. $\mu_1^L$ and $\mu_1^R$ are, at first, set as the lower and upper bounds derived from Gerschgorin theorem [11] and then are refined by the iterative computation of $\nu(\mu_1^L)$ and $\nu(\mu_1^R)$ in the same way as shown on lines 7 and 8. Moreover, several criteria are designed for stopping the binary search (a **repeat**-**until** iteration on lines 4 to 11) in the `dstebz` routine and its subroutine `dlaebz` and we apply the modified criteria on line 11 for computing eigenvalues of $B$. For more details, see the `dstebz` and `dlaebz` routines [15].

Note that the above-mentioned parallel bisection algorithm requires the working memory for computing $\nu(\mu)$ independently on each computation thread. The amount of the working memory for Martin-Wilkinson's Gaussian elimination is $(3w+1)n$ per a computation thread and is larger than that for the block *LU* factorization. Thus, we have to spend about $t(3w+1)n$ working memory for performing parallel Murata's and Gupta's bisection algorithm, where $t$ is the number of the computation threads.

## 3 Eigenvector Computation of Symmetric Band Matrices

The inverse iteration algorithm with reorthogonalization is used in [12, 20] for computing the eigenvectors of a real symmetric band matrix $B$. In this section, we consider applying *block inverse iteration algorithm with reorthogonalization (BIR algorithm)* [14] for computing the eigenvectors of $B$, which is a variant of the inverse iteration algorithm with reorthogonalization.

### *3.1 Inverse Iteration Algorithm with Reorthogonalization*

We first introduce the inverse iteration with reorthogonalization for computing the eigenvectors of $B$. In the followings, let $\lambda_k$ be an eigenvalue of the target matrix such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_\ell$ ($\ell \leq n$) and $\boldsymbol{q}_k$ be the corresponding eigenvector to $\lambda_k$, respectively. Moreover, let $\tilde{\lambda}_k$ be an approximate value of $\lambda_k$, obtained by some eigenvalue computation algorithm such as the bisection algorithm, and $\boldsymbol{v}_k^{(0)}$ be a starting vector for $k = 1, \ldots, \ell$. Then the inverse iteration is to generate a sequence of vectors $\boldsymbol{v}_k^{(i)}$ by solving iteratively the following linear equation:

$$\left(B - \tilde{\lambda}_k I\right) \boldsymbol{v}_k^{(i)} = \boldsymbol{v}_k^{(i-1)}, \quad i = 1, 2, \ldots, \tag{1}$$

where $I$ is the $n \times n$ identity matrix. If $|\tilde{\lambda}_k - \lambda_k| \ll |\tilde{\lambda}_j - \lambda_k|$ for $j \neq k$ is satisfied, $\boldsymbol{v}_k^{(i)}$ converges to $\boldsymbol{q}_k$ as $i \to \infty$.

If some of the eigenvalues are very close to one another, we must reorthogonalize all the corresponding eigenvectors to these eigenvalues. Hereafter, such eigenvalues are referred to as *clustered eigenvalues*. *Peters-Wilkinson's criterion* [22] is applied in `dstein` [15], which is a LAPACK routine for computing eigenvectors of a real symmetric tridiagonal matrix $T$, as dividing eigenvalues of $T$ into clusters. In Peters-Wilkinson's criterion, $\lambda_{k-1}$ and $\lambda_k$ are regarded as belonging to the same cluster if $|\tilde{\lambda}_{k-1} - \tilde{\lambda}_k| \leq 10^{-3}\|T\|_1$ is satisfied ($2 \leq k \leq \ell$). In the followings, we also use Peters-Wilkinson's criterion for dividing eigenvalues of a real symmetric band matrix $B$ into clusters. However, $\|B\|_1 (= \|B\|_\infty)$ is not adequate to use in this criterion since $\|B\|_1$ becomes significantly large according to $w$, the half-bandwidth of $B$. To the contrary, since $\|B\|_2$ satisfies

$$\|B\|_2 = \sup_{\boldsymbol{x} \in \mathbb{R}^n} \frac{\|B\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \geq \max_i |\lambda_i|, \tag{2}$$

a good lower bound of $\|B\|_2$ is obtained by $\max(\tilde{\lambda}_1, \tilde{\lambda}_n)$, where both $\tilde{\lambda}_1$ and $\tilde{\lambda}_n$ do not depend on $w$. Thus, in this paper, Peters-Wilkinson's criterion for computing the eigenvectors of $B$ is designed by using $\tilde{\lambda}_1$ and $\tilde{\lambda}_n$.

Algorithm 2 shows a pseudocode of the inverse iteration algorithm with reorthogonalization for computing the $\ell$ eigenvectors of $B$ and is designed on the basis of the `dstein` routine. As well as the `dstein`, the modified Gram-Schmidt (MGS) algorithm [11] is applied to the reorthogonalization part on line 11. On line 10, Peters-Wilkinson's criterion with the above-mentioned modification is applied for dividing eigenvalues of $B$ into clusters. For solving the linear equation (1), we once perform the *LU factorization with the partial pivoting (PLU factorization)* of $B - \tilde{\lambda}_k I$ by employing the `dgbtrf` routine (line 5), and then iteratively obtain $\boldsymbol{v}_k^{(i)}$ by employing the `dgbtrs` routine (line 9). Note that both `dgbtrf` and `dgbtrs` routines are provided in LAPACK. The `dgbtrf` routine is implemented on the basis of the block algorithm of the *PLU* factorization and is composed of the BLAS 2 and 3 routines. In addition, the `dgbtrs` routine is mainly composed of the BLAS 2 routines and

**Algorithm 2** Inverse Iteration Algorithm with Reorthogonalization for Symmetric Band Matrices

**function** BANDINV($B$, $\ell$, $\tilde{\lambda}_1$, …, $\tilde{\lambda}_\ell$)
  **do** $k := 1$ to $\ell$
    $i := 0$
    Generate an initial random vector: $\boldsymbol{v}_k^{(0)}$
    $LU$ factorization with partial pivoting: $B - \tilde{\lambda}_k = P_k L_k U_k$ ▷ Call `dgbtrf`
    **repeat**
      $i := i + 1$
      Normalize $\boldsymbol{v}_k^{(i-1)}$ to $\boldsymbol{q}_k^{(i-1)}$
      Solve $P_k L_k U_k \boldsymbol{v}_k^{(i)} = \boldsymbol{q}_k^{(i-1)}$ ▷ Call `dgbtrs`
      **if** $k > 1$ and $|\tilde{\lambda}_{k-1} - \tilde{\lambda}_k| \leq 10^{-3} \times \max(|\tilde{\lambda}_1|, |\tilde{\lambda}_n|)$, **then**
        Reorthogonalize $\boldsymbol{v}_k^{(i)}$ to $\boldsymbol{q}_k^{(i)}$ by employing MGS algorithm
      **else**
        $k_1 := k$
      **end if**
    **until** some condition is met.
    Normalize $\boldsymbol{v}_k^{(i)}$ to $\boldsymbol{q}_k^{(i)}$
    $Q_k := \left[ Q_{k-1} \ \boldsymbol{q}_k^{(i)} \right]$
  **end do**
  **return** $Q_\ell = \left[ \boldsymbol{q}_1 \ \cdots \ \boldsymbol{q}_\ell \right]$
**end function**

requires $P_k$, $L_k$, and $U_k$, which are the resulting elements of the *PLU* factorization by the `dgbtrf` routine. For this purpose, we have to store $P_k$, $L_k$, and $U_k$ in the working memory and their amount is about $(3w+1)n$.

In this paper, let us consider that the inverse iteration algorithm with reorthogonalization is parallelized by employing the parallel BLAS routines.

### *3.2 Block Inverse Iteration Algorithm with Reorthogonalization*

A pseudocode of the block inverse iteration algorithm with reorthogonalization (BIR algorithm) for computing the corresponding eigenvectors to the clustered eigenvalues of $B$ is shown in Algorithm 3, where $\hat{\ell}$ is the number of eigenvalues belonging to a certain cluster and $r$ is a block parameter determined arbitrarily by users ($r \leq \hat{\ell}$). For convenience, we assume the $r$ is a divisor of $\hat{\ell}$. Note that the BIR algorithm corresponds to the inverse iteration algorithm with reorthogonalization in Algorithm 2 if $r = 1$.

The BIR algorithm is composed of two parts as well as the inverse iteration algorithm with reorthogonalization. The one is to solve $r$ linear equations simultaneously. For this part, the `dgbtrf` and `dgbtrs` routines are employed as well as the inverse iteration algorithm with reorthogonalization. Different from the inverse iteration algorithm with reorthogonalization, the computation of solving simultaneously $r$ linear equations can be parallelized in terms of $k$ since the linear equations

**Algorithm 3** Block Inverse Iteration Algorithm with Reorthogonalization for Computing the Corresponding Eigenvectors to Clustered Eigenvalues of Symmetric Band Matrices

1: **function** BANDBIR($B$, $r$, $\hat{\ell}$, $\tilde{\lambda}_1$, …, $\tilde{\lambda}_{\hat{\ell}}$)
2: Set an $n \times r$ matrix $Q_0$ be $Q_0 := \boldsymbol{O}$
3: **do** $j := 1$ to $\hat{\ell}/r$
4: $i := 0$
5: Generate $Q_{j,r}^{(0)} := \left[ \boldsymbol{q}_{(j-1)r+1}^{(0)} \ \cdots \ \boldsymbol{q}_{jr}^{(0)} \right]$
6: `!$omp parallel do`
7: **do** $k =: (j-1)r+1$ to $jr$
8: $LU$ factorization with partial pivoting: $B - \tilde{\lambda}_k = P_k L_k U_k$ ▷ Call `dgbtrf`
9: **end do**
10: **repeat**
11: $i := i+1$
12: `!$omp parallel do`
13: **do** $k =: (j-1)r+1, \ldots, jr$
14: Solve $P_k L_k U_k \boldsymbol{v}_k^{(i)} = \boldsymbol{q}_k^{(i-1)}$ ▷ Call `dgbtrs`
15: **end do**
16: $V_{j,r}^{(i)} := V_{j,r}^{(i)} - Q_{(j-1)r} Q_{(j-1)r}^{\top} V_{j,r}$ ▷ Call `dgemm` ×2
17: QR factorization: $V_{j,r}^{(i)} = \overline{Q}_{j,r}^{(i)} R_{j,r}^{(i)}$
18: $\overline{Q}_{j,r}^{(i)} := \overline{Q}_{j,r}^{(i)} - Q_{(j-1)r} Q_{(j-1)r}^{\top} \overline{Q}_{j,r}^{(i)}$ ▷ Call `dgemm` ×2
19: QR factorization: $\overline{Q}_{j,r}^{(i)} = Q_{j,r}^{(i)} R_{j,r}^{(i)}$
20: **until** converge
21: $Q_{jr} := \left[ Q_{(j-1)r} \ Q_{j,r}^{(i)} \right] \left( Q_r := \left[ Q_{1,r}^{(i)} \right] \right)$
22: **end do**
23: **return** $Q_{\hat{\ell}} = \left[ \boldsymbol{q}_1 \ \cdots \ \boldsymbol{q}_{\hat{\ell}} \right]$
24: **end function**

are independent of each other. Thus, the computation of this part is parallelized by the OpenMP directives shown on lines 6 to 9 and lines 12 to 15. Note that we have to spend $r(3w+1)n$ working memory to store $P_k$, $L_k$, and $U_k$ corresponded to the $r$ linear equations for the above purpose.

The other is the block reorthogonalization part as shown on lines 16 to 19. In Algorithm 3, the block Gram-Schmidt algorithm with reorthogonalization (BCGS2 algorithm) [5] is used for this part. The BCGS2 algorithm is mainly composed of the matrix multiplication, which is one of the BLAS 3 routines. Thus, the `dgemm` routines are employed to the computation on lines 16 and 18 in Algorithm 3. As well as the BIR algorithm proposed in [14], we consider that the recursive implementation of the classical Gram-Schmidt algorithm [24] is applied to the computation of the QR factorization on lines 17 and 19, which is also mainly composed of the matrix multiplications. In this paper, the block reorthogonalization part is parallelized by employing the parallel BLAS routines.

The BIR algorithm corresponds to the simultaneous inverse iteration algorithm [10] if $r = \hat{\ell}$. Thus, the simultaneous inverse iteration algorithm always spends the larger amount of memory than the BIR algorithm does. Note that the memory use for the

BIR algorithm is almost equal to that for the parallel bisection algorithms mentioned in 2.3 if $r$ is set to the number of the computation threads.

### *3.3 Remark on Inverse Iteration Algorithms*

The relationship between the inverse iteration algorithm with reorthogonalization and the BIR algorithm is analogous to that between the *LU* factorization and the block *LU* factorization. Thus, the number of floating-point operations in the BIR algorithm is almost equal to that in the inverse iteration algorithm with reorthogonalization.

As mentioned before, both the inverse iteration algorithm with reorthogonalization and the BIR algorithm are composed two parts: solving linear equation and the (block) reorthogonalization part. Assuming $\ell$ is the number of the desired eigenvectors of $B$, the number of floating-point operations is $O(\ell w^2 n)$ in solving linear equations and is $O(\hat{\ell}_{\max}^2 n)$ in the reorthogonalization part, where $\hat{\ell}_{\max}$ is the number of eigenvalues belonging to the largest eigenvalue cluster ($\hat{\ell}_{\max} \leq \ell$). As a result, solving linear equations is occupied with much of the execution time for computing eigenvectors of $B$ by the inverse iteration algorithm with reorthogonalization as long as it is not a case that $\ell$ is much larger than $w$. The same is true of the BIR algorithm.

## 4 Performance Evaluation

This section gives experimental results on shared-memory multi-core processors to evaluate the performance of the proposed eigensolver, which computes eigenvalues of real symmetric band matrices by employing parallel Murata's bisection algorithm mentioned in Section 2 and computes the corresponding eigenvectors by the BIR algorithm mentioned in Section 3.2.

Table 1 shows our experimental environment, which is one node of Appro Green Blade 8000 at ACCMS, Kyoto University. All the experiments were performed by `numactl --interleave all` to control NUMA policy. In addition, each code was run with 16 threads on the condition that the `KMP_AFFINITY` was set to "none" in all the numerical experiments except for the performance evaluation of the parallel efficiency of the eigensolvers in Section 4.3. Note that the `KMP_AFFINITY` is an environment variable for controlling the OpenMP thread affinity. The Intel Math Kernel Library (MKL) was used for the parallel execution of the BLAS and LAPACK routines and the OpenMP directives are also used for the thread parallelization as mentioned in Section 2 and 3. The block size $r$ of the BIR algorithm in the proposed eigensolver was set to $r = 16$ in all the experiments, which is equal to the number of cores in the experimental environment shown in Table 1. Since the working memory for the BIR algorithm is almost equal to that for the parallel bisection algorithm, the total memory use can be easily estimated on this condition.

Table 1: Specifications of the experimental environment.

| One node of Appro Green Blade 8000 at ACCMS | |
|---|---|
| CPU | Intel Xeon E5-2670@2.6GHz, 16 cores (8 cores × 2 sockets) |
| | L3 cache: 20MB × 2 |
| RAM | DDR3-1600 64GB, 136.4GB/sec |
| Compiler | Intel C++/Fortran Compiler 15.0.2 |
| Options | `-O3 -xHOST -ipo -no-prec-div` |
| | `-openmp -mcmodel=medium -shared-intel` |
| Run command | `numactl --interleave=all` |
| Software | Intel Math Kernel Library 11.2.2 |

Note that the maximum number of iterations in both the BIR algorithm and the inverse iteration algorithm with reorthogonalization is set to 5, as well as the `dstein` routine provided in LAPACK. In fact, the number of iterations in both of them is 3 in all the experiments.

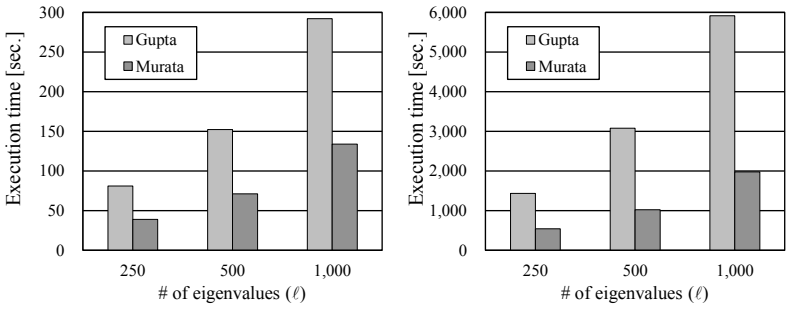
The following $n \times n$ symmetric band matrices with half-bandwidth $w$ were used for test matrices in the performance evaluation, whose elements are set random numbers in the range $[0,1)$: $B_1$ is set to $n = 20,000$ and $w = 64$; $B_2$ is set to $n = 40,000$ and $w = 256$. In the experiments, the largest $\ell$ eigenvalues of them and the corresponding eigenvectors are computed, where $\ell$ is set to $\ell = 250,\ 500,\ 1,000$.

### *4.1 Performance Evaluation of Murata's Bisection Algorithm*

To evaluate the performance of parallel Murata's bisection algorithm in Section 2, we compared the execution time for computing the desired eigenvalues of real symmetric band matrices by using parallel Murata's bisection algorithm with that by using Gupta's bisection algorithm. Their codes are parallelized by employing the OpenMP directives as shown in Section 2.3.

Figs. 1a and 1b show the execution times for computing the $\ell$ largest eigenvalues of $B_1$ and $B_2$, respectively. According to the expectation in Section 2.2, we observe that Murata's bisection algorithm is faster than Gupta's bisection algorithm in all the cases.

Tables 2a and 2b show the number of computing $\nu(\mu)$ on the basis of the block *LU* factorization algorithm and Martin-Wilkinson's Gaussian elimination. These Tables indicate that most of the computation of $\nu(\mu)$ is performed by the block *LU* factorization-based algorithm in Murata's bisection. In addition, the total number of computing $\nu(\mu)$ in Murata's bisection is almost the same as that in Gupta's bisection. We also observe that the total number of computing $\nu(\mu)$ in both bisection algorithms depends on $\ell$, the number of the desired eigenvalues.




(a) Cases of $B_1$ (b) Cases of $B_2$

Fig. 1: Execution times for computing the largest $\ell$ eigenvalues of real symmetric band matrices by using parallel Murata's bisection algorithm and parallel Gupta's bisection algorithm.

Table 2: The number of computing $\nu(\mu)$ on the basis of the block *LU* factorization algorithm and Martin-Wilkinson's Gaussian elimination when we compute the largest $\ell$ eigenvalues by employing parallel Murata's bisection algorithm and parallel Gupta's bisection algorithm

(a) Cases of $B_1$

| # of eigenvalues ($\ell$) | | 250 | 500 | 1,000 |
|---|---|---|---|---|
| Murata | Block *LU* | 9,538 | 18,802 | 36,852 |
| | M-W | 108 | 209 | 692 |
| Gupta | M-W | 9,896 | 19,511 | 38,544 |

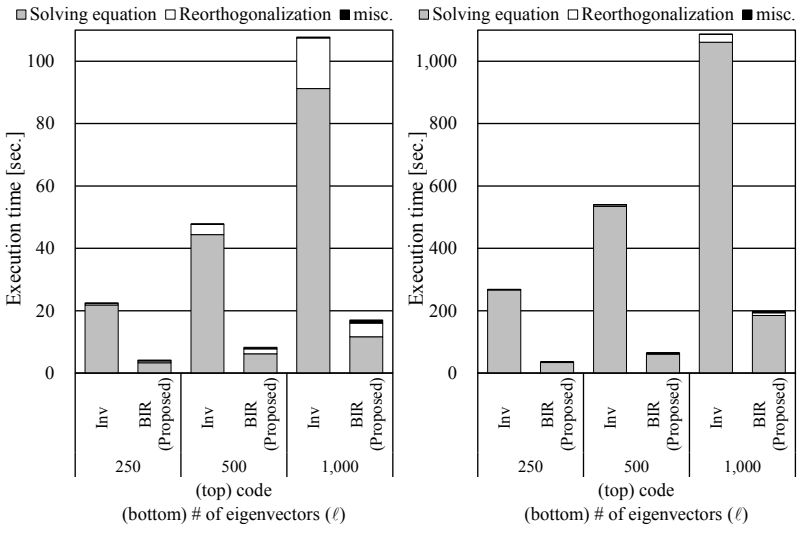
(b) Cases of $B_2$

| # of eigenvalues ($\ell$) | | 250 | 500 | 1,000 |
|---|---|---|---|---|
| Murata | Block *LU* | 9,030 | 17,728 | 34,719 |
| | M-W | 186 | 506 | 1,449 |
| Gupta | M-W | 9,216 | 18,234 | 36,168 |

## *4.2 Performance Evaluation of BIR Algorithm*

In order to evaluate the performance of the proposed eigenvector computation algorithm in Section 3, we compared the execution time for computing of the eigenvectors corresponding to the $\ell$ largest eigenvalues of real symmetric band matrices by using the proposed algorithm ("BIR") with that by using the inverse iteration algorithm with reorthogonalization ("Inv"). Their codes are parallelized by employing the Intel MKL and the OpenMP directives as shown in Section 3. In addition, the $\ell$ largest eigenvalues of the target matrices are obtained by using parallel Murata's bisection algorithm.

Fig. 2 shows the execution times for computing the corresponding eigenvectors to the $\ell$ largest eigenvalues of the target matrices and their details, where "Solving equation" denotes the execution time for solving linear equations (1) and "Reorthogonalization" denotes the execution time for the reorthogonalization part performed by the MGS algorithm in "Inv" or the BCGS2 algorithm in "BIR". Figs. 2a and 2b correspond to the case of $B_1$ and $B_2$, respectively. These figures show that "BIR" is faster than "Inv" in all the cases. According to the discussion about the number of




(a) Cases of $B_1$ (b) Cases of $B_2$

Fig. 2: Execution times for computing the corresponding eigenvectors to the largest $\ell$ eigenvalues of symmetric band matrices by using the block inverse iteration algorithm with reorthogonalization ("BIR") and the inverse iteration algorithm with reorthogonalization ("Inv") and their details.

floating-point operations in each part mentioned in Section 3.3, the "Solving equation" part occupies most parts of the execution time of the eigenvector computation in all the cases.

We also observe that the execution time for the "Solving equation" part of "BIR" is significantly shorter than that of "Inv" in all the cases. As mentioned in Section 3, the parallelization of the "Solving equation" part in "BIR" differs from that in "Inv". In the BIR algorithm, each linear equation is solved on each computation thread, and thus any barrier synchronization between the computation threads does not occur until all the computation assigned to each computation thread is finished. On the other hand, since the inverse iteration algorithm with reorthogonalization is parallelized by employing the parallel BLAS routines, the barrier synchronization between the computation threads occurs each time the BLAS routine is called. Moreover, the BLAS-based computations in the `dgbtrf` and `dgbtrs` routines are difficult to achieve good performance in parallel processing since the size of vectors and matrices appearing in these computations is too small. From the above reasons, the "Solving equation" part of "BIR" achieves the higher performance in parallel processing than that of "Inv".

Finally, we examined the effect of the block size $r$ on the performance of the BIR algorithm. As mentioned in Section 3.2, the block reorthogonalization part of the BIR algorithm includes many matrix multiplications, and thus, the performance

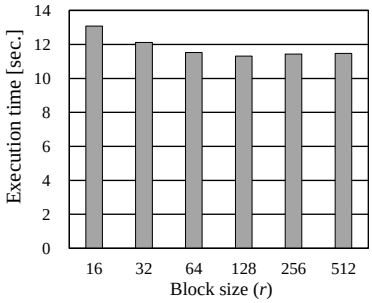
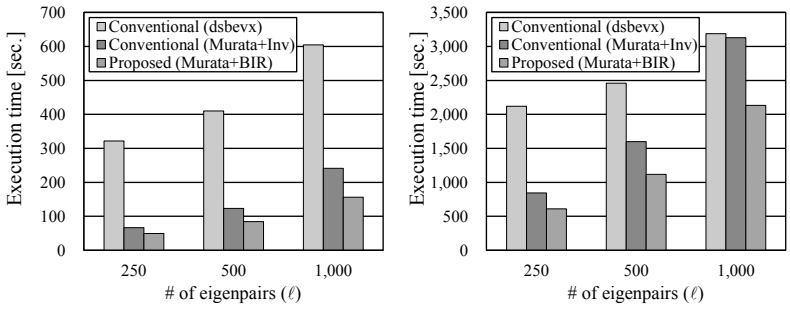



Fig. 3: Execution times for computing the corresponding eigenvectors to the largest $1,000$ eigenvalues of $B_1$ by using the block inverse iteration algorithm with reorthogonalization.

of the BIR algorithm depends on that of the routine for the matrix multiplications `dgemm`. In addition, the `dgemm` routine is difficult to achieve the better performance if the size of the matrices appearing in the computation is sufficiently large. Fig. 3 shows the execution times for computing the corresponding eigenvectors to the largest $1,000$ eigenvalues of $B_1$ by using the BIR algorithm with different block size $r$. From this figure, we observe that the BIR algorithm with $r = 128$ or 256 is somewhat faster than that with $r = 16$ for computing the $1,000$ eigenvectors of $B_1$. In spite of this tendency, we set $r$ as the number of cores in the proposed eigensolver. This is because $r$ must be smaller than the number of the desired eigenvectors as mentioned in Section 3.2.

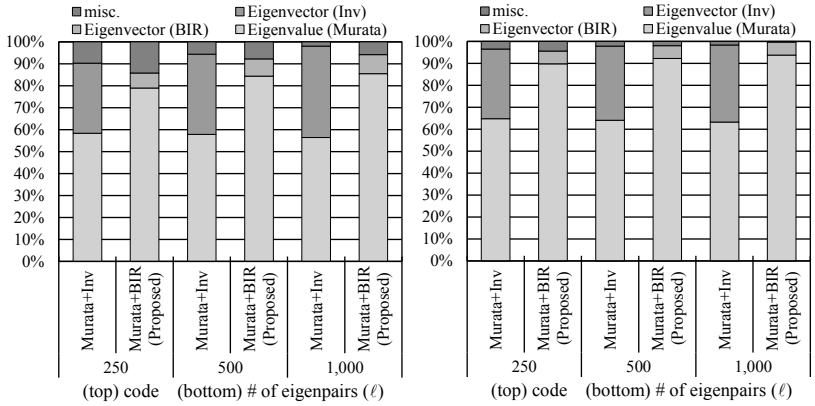
### *4.3 Performance Evaluation of Proposed Band Eigensolver*

In order to evaluate the performance of the proposed symmetric band eigensolver, we compared the proposed solver ("Murata+BIR") with the two conventional eigensolvers. One of the conventional solvers is also to compute eigenvalues by using Murata's bisection algorithm and to compute the corresponding eigenvectors by using the inverse iteration algorithm with reorthogonalization shown in Algorithm 2 and is referred to as "Murata+Inv". The codes of "Murata+BIR" and "Murata+Inv" are also parallelized by employing the Intel MKL and the OpenMP directives as shown in Section 2.3 and Section 3. The other conventional solver is "dsbevx" provided in Intel MKL, which is a LAPACK routine for computing a subset of eigenpairs of real symmetric band matrices through the tridiagonalization. Note that "dsbevx" employs the `dsbtrd` routine to tridiagonalize the target band matrix in the way proposed in [17], the `dstebz` routine to compute the desired eigenvalues of the real symmetric tridiagonal matrix, and the `dstein` routine to compute the corresponding eigenvectors.




(a) Cases of $B_1$ (b) Cases of $B_2$

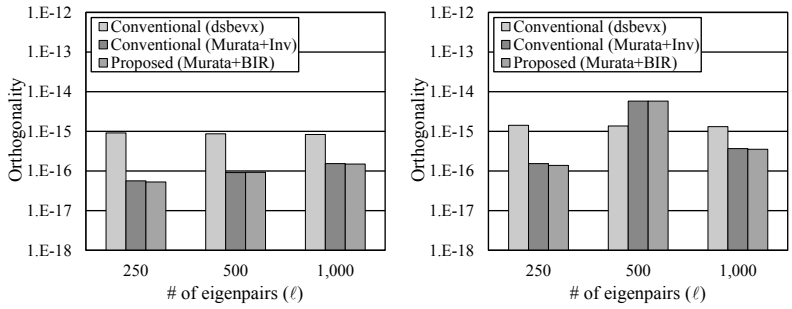
Fig. 4: Execution times for computing the eigenpairs corresponding to the largest $\ell$ eigenvalues of real symmetric band matrices by using the proposed solver ("Murata+BIR") and the conventional solvers ("Murata+Inv" and "dsbevx").




(a) Cases of $B_1$ (b) Cases of $B_2$

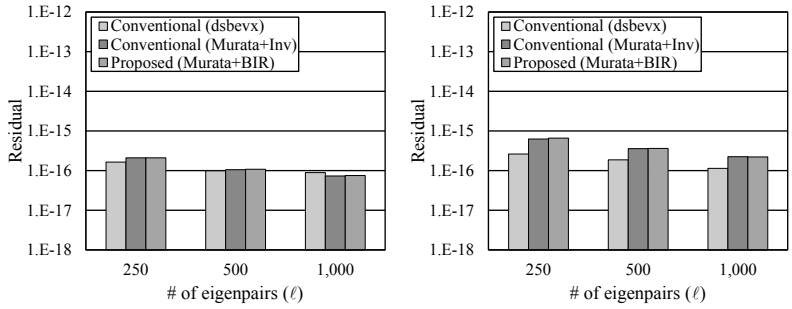
Fig. 5: Details of the execution times for computing the eigenpairs corresponding to the largest $\ell$ eigenvalues of real symmetric band matrices by using "Murata+BIR" and "Murata+Inv".

Figs. 4a and 4b show the overall execution time for computing the eigenpairs corresponding to the $\ell$ largest eigenvalues of $B_1$ and $B_2$, respectively. We observe that the proposed eigensolver, "Murata+BIR", is faster than the conventional solvers in all the cases. Figs. 5a and 5b show the details of the overall execution time for "Murata+BIR" and "Murata+Inv". We observe that most of the execution time in "Murata+BIR" remains to be occupied by that of the eigenvalue computation using parallel Murata's bisection algorithm in all the cases. One reason of this result is that the number of floating-point operations in "Murata", Murata's bisection algorithm, is much higher than that in "BIR". The other reason is that the execution time




(a) Cases of $B_1$ (b) Cases of $B_2$

Fig. 6: Orthogonality $\|Q_\ell^\top Q_\ell - I\|_\infty/\ell$ of the corresponding eigenvectors to the largest $\ell$ eigenvalues of real symmetric band matrices by using the proposed solver ("Murata+BIR") and the conventional solvers ("Murata+Inv" and "dsbevx").



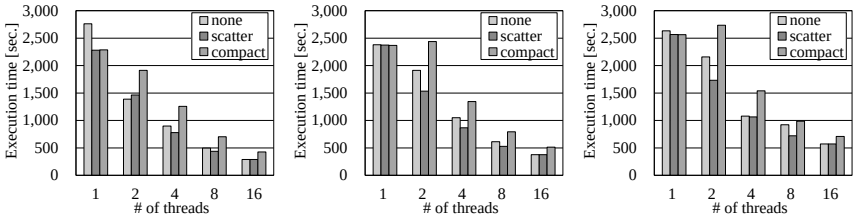

(a) Cases of $B_1$ (b) Cases of $B_2$

Fig. 7: Residual $\|B_i Q_\ell - Q_\ell D_\ell\|_\infty/\ell$ of the eigenpairs corresponding to the largest $\ell$ eigenvalues of real symmetric band matrices by using the proposed solver ("Murata+BIR") and the conventional solvers ("Murata+Inv" and "dsbevx").

for eigenvector computation in "Murata+BIR" is significantly reduced from that in "Murata+Inv" as mentioned in Section 4.2.

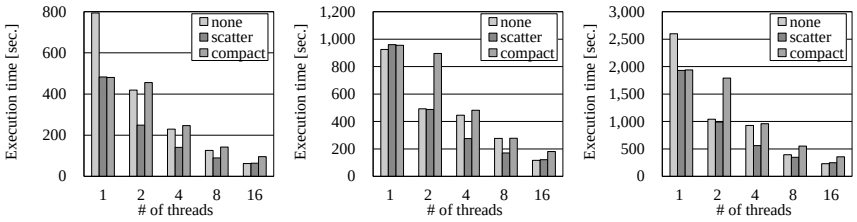
The accuracy of the eigenpairs computed by using the proposed solver and the conventional solvers is shown as follows. Figs. 6a and 6b show the orthogonality $\|Q_\ell^\top Q_\ell - I\|_\infty/\ell$ of $B_1$ and $B_2$, respectively. Similarly, Figs. 7a and 7b show the residual $\|B_i Q_\ell - Q_\ell D_\ell\|_\infty/\ell$ of $B_1$ and $B_2$, respectively. Note that $D_\ell = \mathrm{diag}(\tilde{\lambda}_1, \cdots, \tilde{\lambda}_\ell)$ and $Q_\ell = \begin{bmatrix} \boldsymbol{q}_1 & \cdots & \boldsymbol{q}_\ell \end{bmatrix}$. These figures show that the proposed eigensolver computes the desired eigenpairs as accurately as the conventional solvers.

To evaluate the parallel efficiency, we compared the overall execution times with 1, 2, 4, 8, and 16 threads for computing the eigenpairs corresponding to the $\ell$ largest eigenvalues of the test matrices. Figs. 8, 9, and 10 show the cases of $B_1$. Figs. 11, 12, and 13 show the cases of $B_2$. In these figures, we also compared the execution times of each code run using the different `KMP_AFFINITY` environment variables:



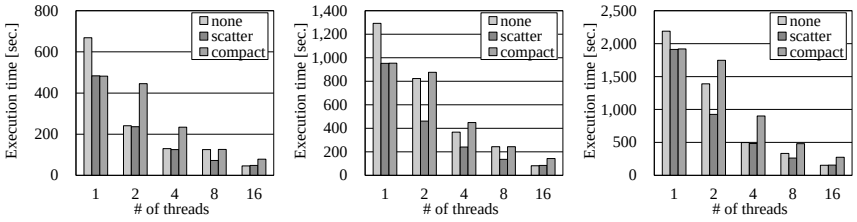

(a) Cases of $\ell = 250$ (b) Cases of $\ell = 500$ (c) Cases of $\ell = 1,000$

Fig. 8: Execution times for computing the eigenpairs corresponding to the largest $\ell$ eigenvalues of $B_1$ by using "dsbevx" in different `KMP_AFFINITY`.




(a) Cases of $\ell = 250$ (b) Cases of $\ell = 500$ (c) Cases of $\ell = 1,000$

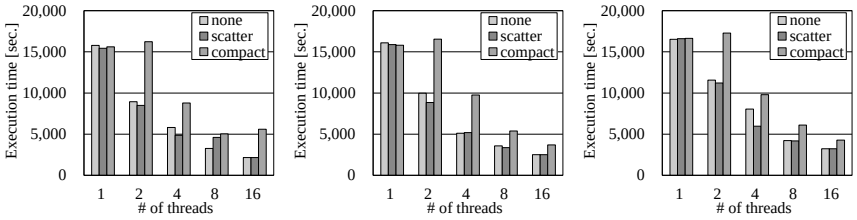
Fig. 9: Execution times for computing the eigenpairs corresponding to the largest $\ell$ eigenvalues of $B_1$ by using "Murata+Inv" in different `KMP_AFFINITY`.




(a) Cases of $\ell = 250$ (b) Cases of $\ell = 500$ (c) Cases of $\ell = 1,000$

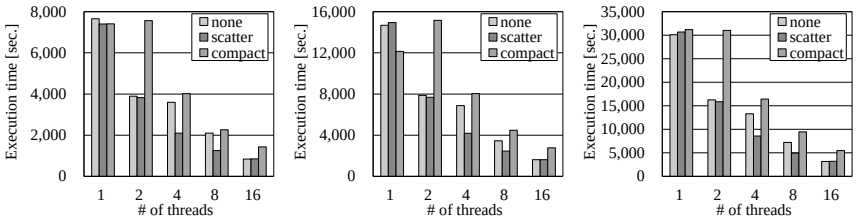
Fig. 10: Execution times for computing the eigenpairs corresponding to the largest $\ell$ eigenvalues of $B_1$ by using the proposed eigensolver "Murata+BIR" in different `KMP_AFFINITY`.

"none", "scatter", and "compact". From these figures, we observe that the proposed eigensolver "Murata+BIR" achieve the higher parallel efficiency than the conventional eigensolvers "dsbevx" and "Murata+Inv". We also observe that, if the number of threads is 16, each of the eigensolvers run with "none" achieves a competitive performance as that run with "scatter" does and the eigensolvers run with "compact" achieves the worst performance. From Figs. 10 and 13, the parallel efficiency of the proposed eigensolver run with "scatter" for $B_2$ is higher than that for $B_1$, which is



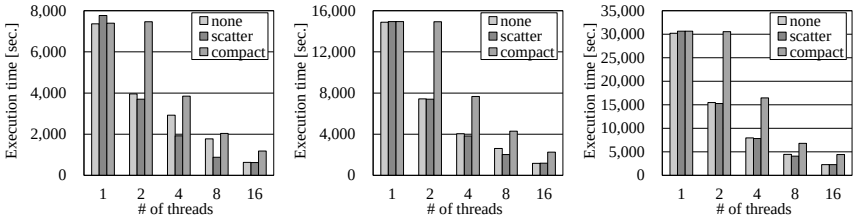

(a) Cases of $\ell = 250$ (b) Cases of $\ell = 500$ (c) Cases of $\ell = 1,000$

Fig. 11: Execution times for computing the eigenpairs corresponding to the largest $\ell$ eigenvalues of $B_2$ by using "dsbevx" in different `KMP_AFFINITY`.




(a) Cases of $\ell = 250$ (b) Cases of $\ell = 500$ (c) Cases of $\ell = 1,000$

Fig. 12: Execution times for computing the eigenpairs corresponding to the largest $\ell$ eigenvalues of $B_2$ by using "Murata+Inv" in different `KMP_AFFINITY`.




(a) Cases of $\ell = 250$ (b) Cases of $\ell = 500$ (c) Cases of $\ell = 1,000$

Fig. 13: Execution times for computing the eigenpairs corresponding to the largest $\ell$ eigenvalues of $B_2$ by using the proposed eigensolver "Murata+BIR" in different `KMP_AFFINITY`.

smaller than $B_2$ in terms of both the matrix size and the bandwidth size. Moreover, the parallel efficiency of it for both $B_1$ and $B_2$ becomes higher than as the number of the desired eigenpairs $\ell$ increases.

## 5 Conclusions and Future Work

In order to accelerate a subset computation of eigenpairs for real symmetric band matrices, the parallel symmetric band eigensolver is proposed, which computes directly the desired eigenvalues by using parallel Murata's bisection algorithm in Section 2 and the corresponding eigenvectors by using the BIR algorithm in Section 3.2. Employing not only Martin-Wilkinson's Gaussian elimination but also the block *LU* factorization, parallel Murata's bisection algorithm is faster than parallel Gupta's bisection algorithm. Numerical experiments on shared-memory multi-core processors show that the BIR algorithm is much faster than the inverse iteration algorithm with reorthogonalization since the BIR algorithm is parallelized with lower communication cost than the other. As the result, the numerical experiments also show that the proposed eigensolver is faster than the conventional solvers. In conclusion, we show that the parallel efficiency of the proposed eigensolver run with "scatter" becomes much higher as the problem size increases.

One of future work is to apply the proposed symmetric band eigensolver for computing a subset of eigenpairs of real symmetric dense matrices appearing in actual applications, such as the kernel principal component analysis. The number of the desired eigenpairs in such problems may be fewer than the number of the processing elements. In this case, the multi-section methods proposed in [18, 23] or the multi-section with multiple eigenvalues method [16] is expected to achieve the higher performance than the parallel bisection algorithm in Section 2.3. Thus, the development of the multi-section algorithm based on Murata's bisection algorithm is considered as the other future work.

**Acknowledgements** The authors would like to express their gratitude to reviewers for their helpful comments. In this work, we used the supercomputer of ACCMS, Kyoto University.

## References

1. Anderson, E., Bai, Z., Bischof, C., Blackford, L., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, third edn. SIAM, Philadelphia, PA, USA (1999)
2. Auckenthaler, T., Blum, V., Bungartz, H.J., Huckle, T., Johanni, R., Krämer, L., Lang, B., Lederer, H., Willems, P.: Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations. Parallel Computing **37**(12), 783–794 (2011)
3. Auckenthaler, T., Bungartz, H.J., Huckle, T., Krämer, L., Lang, B., Willems, P.: Developing algorithms and software for the parallel solution of the symmetric eigenvalue problem. Journal of Computational Science **2**(3), 272–278 (2011)
4. Ballard, G., Demmel, J., Knight, N.: Avoiding communication in successive band reduction. ACM Trans. Parallel Comput. **1**(2), 11:1–11:37 (2015)
5. Barlow, J.L., Smoktunowicz, A.: Reorthogonalized block classical Gram-Schmidt. Numer. Math. **123**(3), 1–29 (2012)
6. Barth, W., Martin, R., Wilkinson, J.: Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection. Numer. Math. **9**(5), 386–393 (1967)

7. Bischof, C., Sun, X., Lang, B.: Parallel tridiagonalization through two-step band reduction. In: Proceedings of the Scalable High-Performance Computing Conference, pp. 23–27. IEEE (1994)
8. Bischof, C.H., Lang, B., Sun, X.: A framework for symmetric band reduction. ACM Trans. Math. Softw. **26**(4), 581–601 (2000)
9. Blackford, L.S., Demmel, J., Dongarra, J., Duff, I., Hammarling, S., Henry, G., Heroux, M., Kaufman, L., Lumsdaine, A., Petitet, A., Pozo, R., Remington, K., Whaley, R.C.: An updated set of basic linear algebra subprograms (BLAS). ACM Trans. Math. Softw. **28**(2), 135–151 (2002)
10. Chatelin, F.: Eigenvalues of Matrices. SIAM, Philadelphia, PA, USA (2012)
11. Golub, G.H., van Loan, C.F.: Matrix Computations, third edn. Johns Hopkins University Press, Baltimore, MD, USA (1996)
12. Gupta, K.K.: Eigenproblem solution by a combined Sturm sequence and inverse iteration technique. Int. J. num. Meth. Engng **7**(1), 17–42 (1973)
13. Hasegawa, H.: Symmetric band eigenvalue solvers for vector computers and conventional computers (in Japanese). J. Inf. Process. **30**(3), 261–268 (1989)
14. Ishigami, H., Kimura, K., Nakamura, Y.: A new parallel symmetric tridiagonal eigensolver based on bisection and inverse iteration algorithms for shared-memory multi-core processors. In: P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2015 Tenth International Conference on, pp. 216–223. IEEE (2015)
15. Kahan, W.: Accurate eigenvalues of a symmetric tridiagonal matrix. Technical Report, Computer Science Dept. Stanford University (CS41) (1966)
16. Katagiri, T., Vömel, C., Demmel, J.W.: Automatic performance tuning for the multi-section with multiple eigenvalues method for symmetric tridiagonal eigenproblems. In: Applied Parallel Computing. State of the Art in Scientific Computing, *Lecture Notes in Computer Science*, vol. 4699, pp. 938–948. Springer Berlin Heidelberg (2007)
17. Kaufman, L.: Band reduction algorithms revisited. ACM Trans. Math. Softw. **26**(4), 551–567 (2000)
18. Lo, S., Philippe, B., Sameh, A.: A multiprocessor algorithm for the symmetric tridiagonal eigenvalue problem. SIAM J. Sci. Stat. Comput. **8**(2), s155–s165 (1987)
19. Martin, R., Wilkinson, J.: Solution of symmetric and unsymmetric band equations and the calculation of eigenvectors of band matrices. Numer. Math. **9**(4), 279–301 (1967)
20. Murata, K.: Reexamination of the standard eigenvalue problem of the symmetric matrix. II The direct sturm inverse-iteration for the banded matrix (in Japanese). Research report of University of Library and Information Science **5**(1), 25–45 (1986)
21. Parlett, B.N.: The Symmetric Eigenvalue Problem. SIAM, Philadelphia, PA, USA (1998)
22. Peters, G., Wilkinson, J.: The calculation of specified eigenvectors by inverse iteration. In: F. Bauer (ed.) Linear Algebra, *Handbook for Automatic Computation*, vol. 2, pp. 418–439. Springer Berlin Heidelberg (1971)
23. Simon, H.: Bisection is not optimal on vector processors. SIAM J. Sci. Stat. Comput. **10**(1), 205–209 (1989)
24. Yokozawa, T., Takahashi, D., Boku, T., Sato, M.: Parallel implementation of a recursive blocked algorithm for classical Gram-Schmidt orthogonalization. Proc. 9th International Workshop on State-of-the-Art in Scientific and Parallel Computing (PARA 2008) (2008)