Vol.2012-MPS-89 No.4
2012/7/16

Performance Evaluation of Some Inverse Iteration Algorithms on PowerXCell^{TM} 8i Processor

Masami Takata[1,a)] Hiroyuki Ishigami[2,b)] Kinji Kimura[2,c)] Yoshimasa Nakamura[2,d)]

Abstract: In this paper, we compare with the inverse iteration algorithms on PowerXCell^{TM} 8i processor, which has been known as a heterogeneous environment. When some of all the eigenvalues are close together or there are clusters of eigenvalues, reorthogonalization must be adopted to all the eigenvectors associated with such eigenvalues. Reorthogonalization algorithms need a lot of computational cost. The Classical Gram-Schmidt (CGS) algorithm, the modified Gram-Schmidt (MGS) algorithm, and the Householder orthogonalization algorithm in terms of the compact WY representation have been known as reorthogonalization algorithms. These algorithms can be computed using BLAS level-1 and level-2. Since synergistic processor elements in PowerXCell^{TM} 8i processor archive the high performance of BLAS level-2 and level-3, the orthogonalization algorithms except the MGS algorithm can be computed high-speed on parallel computers.

1. Introduction

The eigenvalue decomposition of a symmetric matrix is one of the most important operations in linear algebra. It is used in molecular orbital of chemical, vibrational analysis, image processing, data searches, etc..

Owing to recent improvements in the performance of computers equipped with multicore processors, we have had more opportunities to perform calculations on parallel computers. As a result, there has been an increase in the demand for an eigenvalue decomposition algorithm that can be effectively parallelized.

Any $n \times n$ symmetric matrix is transformed into a symmetric tridiagonal matrix by using a sequence of Householder transformations [4], [9]. This preconditioning process helps to shorten computational time drastically. Hence, eigenvalue decomposition algorithms of symmetric tridiagonal matrices are important. Several eigenvalue decomposition algorithms of a symmetric tridiagonal matrix have been proposed [3], [7], [10], [12], [13], [17]. They are classified into two types. The first type of algorithm computes simultaneously all the eigenvalues and the eigenvectors. Algorithms of this type include the QR algorithm [10] and the divide-and-conquer algorithm [3], [13]. The second type of algorithm computes all or some eigenvalues and all or some eigenvectors. Algorithms for computing eigenvalues include the root-free QR algorithm [12] and the bisection algorithm [10]. Algorithms for computing eigenvectors include the MR^3 algorithm [7] and the inverse iteration algorithm with the modified Gram-Schmidt (MGS) algorithm [10], [17]. LAPACK (Linear Algebra PACKage) [16], which is a software library for numerical linear algebra, has codes that integrate all the above-mentioned algorithms. These algorithms can be parallelized, except the root-free QR algorithm.

The inverse iteration algorithm is an algorithm for computing eigenvectors independently associated with mutually distinct eigenvalues. However, when some eigenvalues are very close to each other, the eigenvectors, which are computed using the inverse iteration algorithm, must be reorthogonalized. As reorthogonalization algorithms, the Classical Gram-Schmidt (CGS) algorithm [10], the MGS algorithm, the Householder orthogonalization algorithm [15] are known. Reorthogonalization algorithms need a lot of computational cost. The CGS algorithm is suitable algorithm for parallel computing. The orthogonality of eigenvectors computed by the CGS algorithm depends on the square of the condition number of the eigenvectors, which are generated using the inverse iteration, in the same cluster of the eigenvalues [20]. The MGS algorithm is sequential and inefficient for parallel computing. The orthogonality of eigenvectors computed by the MGS algorithm depends on the condition number. The Householder orthogonalization algorithm can orthogonalize eigenvectors by using the Householder transformation [19]. The orthogonality in the Householder orthogonalization algorithm does not depend on the condition number. The Householder algorithm is sequential and inefficient for parallel computing. Ishigami et. al. have developed parallel algorithms for the Householder orthogonalization algorithm in terms of the compact WY representation [15], which is named as the cWY algorithm in this paper.

In ExaFLOP computing, since it is critical issue to minimize electricity, heterogeneous environments are suitable. Consequently, it is important to validate the inverse iteration algorithms with the CGS algorithm, the MGS algorithm, and the cWY in heterogeneous environments. As a heterogeneous environment,

1 Nara Women's University, Nara, Nara 630–8506, JAPAN
2 Kyoto University, Kyoto, Kyoto 606–8501, JAPAN
a) takata@ics.nara-wu.ac.jp
b) hishigami@amp.i.kyoto-u.ac.jp
c) kkimur@amp.i.kyoto-u.ac.jp
d) ynaka@i.kyoto-u.ac.jp

cell processor has PowerPC Processor Element (PPE) and eight cores of Synergistic Processor Elements (SPEs). PPE and SPEs can share the same memory. Since SPEs are consisted as multicore, SPEs archive the high performance of BLAS level-2 and level-3 [1]. Basic Linear Algebra Subprograms (BLAS) is an application programming interface standard for publishing libraries to perform basic linear algebra operations such as vector and matrix multiplications. BLAS level-1 can compute vector operations such as inner products, dot products and vector norms. BLAS level-2 and level-3 contain matrix-vector and matrix-matrix operations, respectively. The CGS algorithm and the MGS algorithm can be computed using BLAS level-2 and level-1, respectively. The cWY needs BLAS level-1 and level-2. Note that, the Householder orthogonalization algorithm is almost computed using BLAS level-2. Therefore, these orthogonalization algorithms should be performed in SPEs. By using PPE, an implementation of an inverse iteration is easy. In this paper, we compare with the CGS algorithm, the MGS algorithm, and the cWY on PowerXCell^{TM} 8i processor.

In Section 2, we give a brief review on eigenvalue decomposition. In Section 3, we explain an inverse iteration algorithm and describe its orthogonalization algorithms. In Section 4, we confirm each performance in the inverse iteration algorithms with orthogonalization algorithms on PowerXCell^{TM} 8i processor.

2. Eigenvalue decomposition

Let A be $n \times n$ matrix such that

$$A\boldsymbol{v}_j = \lambda_j \boldsymbol{v}_j \quad (j = 1, 2, ..., n) \tag{1}$$

where λ_j ($\lambda_j : \lambda_j \in \mathbb{C}$) and $\boldsymbol{v}_j$ ($\boldsymbol{v}_j : \boldsymbol{v}_j(\neq 0) \in \mathbb{C}^n$) are an eigenvalue and an eigenvector of A, respectively. If eigenvectors $\boldsymbol{v}_j$ of A are linear independent, then

$$AV = VD, \tag{2}$$

$$D = diag\left[\lambda_1 \quad \lambda_2 \quad \cdots \quad \lambda_n\right], \tag{3}$$

$$V = \left[\boldsymbol{v}_1 \quad \boldsymbol{v}_2 \quad \cdots \quad \boldsymbol{v}_n\right]. \tag{4}$$

Since V is nonsingular, the inverse matrix V^{-1} exists and $V^{-1}V$ is equal to an identity matrix I. Hence, A is decomposed as

$$A = VDV^{-1} \tag{5}$$

Eq.(5) is called eigenvalue decomposition of A.

Let A be real symmetric, then $\lambda_j \in \mathbb{R}$ and $\boldsymbol{v}_j \in \mathbb{R}$. Moreover, eigenvectors $\boldsymbol{v}_j$ are orthogonal to each other, if $\lambda_1 \neq \lambda_2 \neq \cdots \neq \lambda_n$. Note here that V becomes orthogonal matrix by the normalization $\boldsymbol{v}_j \to \boldsymbol{v}_j/\|\boldsymbol{v}_j\|$. Then A is decomposed as

$$A = VDV^\top \tag{6}$$

where $V^\top$ denotes the transposed matrix of V.

In a famous algorithm, a real symmetric matrix A is similarly transformed into a symmetric tridiagonal matrix T by using the Householder transformations. Namely,

$$Q_A^\top A Q_A = T, \tag{7}$$

with suitable orthogonal matrix Q_A. After the tridiagonalization, T is decomposed as

$$T = Q_T D Q_T^\top \tag{8}$$

by some orthogonal matrix Q_T. Consequently, by combining Eq.(7) with Eq.(8), the eigenvalue decomposition of A is given as

$$A = (Q_A Q_T) D (Q_A Q_T)^\top. \tag{9}$$

3. Inverse iteration algorithm

In this section, we introduce the inverse iteration algorithm. When some of all the eigenvalues are close together or there are clusters of eigenvalues, reorthogonalization must be needed to all the eigenvectors associated with such eigenvalues, since the eigenvectors needs to be orthogonal to each other. Therefore, reorthogonalization algorithms should be adopted.

In Section 3.1, we explain a concept of the inverse iteration algorithm. In Section 3.2, 3.3, and 3.4, the CGS algorithm, the MGS algorithm and the cWY are described, respectively. In Section 3.5, these orthogonalization algorithm are compared. In Section 3.6, we descrive a relationship between BLAS and the orthogonalization algorithms.

3.1 Concept

When $\tilde{\lambda}_j$ is an approximate value of λ_j and a starting vector $\boldsymbol{v}_j^{(0)}$ are given, the inverse iteration algorithm can compute an eigenvector of T. To this end, the following equation is solved iteratively:

$$\left(T - \tilde{\lambda}_j I\right) \boldsymbol{v}_j^{(k)} = \boldsymbol{v}_j^{(k-1)} \tag{10}$$

If the eigenvalues of T are mutually well-separated, the solution of $\boldsymbol{v}_j^{(k)}$ in Eq.(10) generically converges to the eigenvector associated with λ_j as k goes to ∞ The above iteration algorithm is the inverse iteration algorithm. When m eigenvectors are computed, the computational cost of this algorithm is of order mn. The computational cost is less than that of other algorithms.In the implementation, the vector $\boldsymbol{v}_j^{(k)}$ must be normalized to avoid overflow.

3.2 Classical Gram-Schmidt algorithm

The CGS algorithm has been proposed as the first reorthogonalization algorithm. In the CGS algorithm, a basis vector $\boldsymbol{x}_j$, which is an orthogonal vector in $\boldsymbol{v}_j$, is computed as follows:

$$\boldsymbol{x}_j' = \boldsymbol{v}_j - \sum_{i=1}^{j-1} \langle \boldsymbol{v}_j, \boldsymbol{x}_i \rangle \boldsymbol{x}_i, \tag{11}$$

$$\boldsymbol{x}_j = \frac{\boldsymbol{x}_j'}{\|\boldsymbol{x}_j'\|} \tag{12}$$

In Eq.(11), $\langle \boldsymbol{v}_j, \boldsymbol{x}_i \rangle \boldsymbol{x}_i$ means an orthographic projection on the direction to $\boldsymbol{x}_i$ of $\boldsymbol{v}_j$. Through $\boldsymbol{v}_j$ is subtracted the orthographic projection, $\boldsymbol{v}_j$ can be picked out of elements $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{j-1}$. Thus, $\boldsymbol{x}_j$ is orthogonalized.

Fig. 1 shows the orthogonalization algorithm using the CGS algorithm. Since Eq.(11) and Eq.(12) are computed using an inner product, BLAS level-1 has to be adopted. Therefore, to adopt

$x_1 = v_1$.
for $j = 2$ to m **do**
  Generate v_j in an algorithm.
  Eq.(11) and Eq.(12) : Orthogonalize v_j to x_j by using $x_1, \cdots, x_{j-1}$.
end for

Fig. 1 Classical Gram-Schmidt algorithm.

1: **for** $j = 1$ to n **do**
2:   Generate $v_j^{(0)}$ from random numbers.
3:   $k = 0$
4:   **repeat**
5:     $k \leftarrow k + 1$.
6:     Normalize $v_j^{(k-1)}$.
7:     Eq.(10) : Compute $v_j^{(k)}$ by using $v_j^{(k-1)}$.
8:     **if** $|\tilde{\lambda}_j - \tilde{\lambda}_{j-1}| \leq 10^{-3}\|T\|$, **then**
9:       **for** $i = j_1$ to $j - 1$ **do**
10:         $v_j^{(k)} \leftarrow v_j^{(k)} - [x_1, x_2, \cdots, x_{j-1}] \begin{bmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_{j-1}^\top \end{bmatrix} v_j^{(k)}$
11:       **end for**
12:     **else**
13:       $j_1 = j$
14:     **end if**
15:   **until** some condition is met.
16:   Normalize $v_j^{(k)}$ to x_j.
17: **end for**

Fig. 2 Inverse iteration algorithm with the CGS algorithm. j_1 means the index j of the first eigenvalue of a cluster.

BLAS level-2, Eq.(11) and Eq.(12) should be transformed into the following vector product.

$$x'_j = v_j - [x_1, x_2, \cdots, x_{j-1}] \begin{bmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_{j-1}^\top \end{bmatrix} v_j. \tag{13}$$

Fig. 2 is a code, which is based on DSTEIN in LAPACK and modified the orthogonalization process from the MGS algorithm to the CGS algorithm. Specifically, line 10 in Fig. 2 is changed to Eq.(13).

3.3 Modified Gram-Schmidt algorithm

If the MGS algorithm is adopted to reorthogonalize eigenvectors, the computational cost is of order $m^2 n$. Therefore, the computational cost, for which eigenvectors of a matrix T are computed, increases significantly. In general, to implement the inverse iteration algorithm on computers, the MGS algorithm with the Peters-Wilkinson method [17] is adopted as the standard orthogonalization process. The MGS algorithm with the Peters-Wilkinson method is also available on DSTEIN, which is implemented in the LAPACK code [16] of the inverse iteration algorithm for computing eigenvectors of a real symmetric tridiagonal matrix. In the Peters-Wilkinson method, when the distance between the close eigenvalues is less than $10^{-3}\|T\|$, these close eigenvalues are regarded as members of the same cluster of eigenvalues, and all of the eigenvectors associated with these eigenvalues are orthogonalized.

Fig. 3 shows the inverse iteration algorithm based on the MGS algorithm with the Peters-Wilkinson method. This loop includes the iteration based on Eq.(10) and the orthogonalization of the eigenvectors. This orthogonalization process becomes a bottleneck of the inverse iteration with respect to the computational time. The MGS algorithm is mainly based on BLAS level-1 such as the inner product operation and the AXPY operation [1].

for $j = 1$ to n **do**
  Generate $v_j^{(0)}$ from random numbers.
  $k = 0$
  repeat
    $k \leftarrow k + 1$.
    Normalize $v_j^{(k-1)}$.
    Eq.(10) : Compute $v_j^{(k)}$ by using $v_j^{(k-1)}$.
    if $|\tilde{\lambda}_j - \tilde{\lambda}_{j-1}| \leq 10^{-3}\|T\|$, **then**
      for $i = j_1$ to $j - 1$ **do**
        $v_j^{(k)} \leftarrow v_j^{(k)} - \langle v_j^{(k)}, x_i \rangle x_i$
      end for
    else
      $j_1 = j$
    end if
  until some condition is met.
  Normalize $v_j^{(k)}$ to x_j.
end for

Fig. 3 Inverse iteration algorithm with the MGS algorithm.

for $j = 1$ to m **do**
  Generate v_j in an algorithm.
  $v'_j = \left(I - s_{j-1} y_{j-1} y_{j-1}^\top\right) \cdots \left(I - s_2 y_2 y_2^\top\right)\left(I - s_1 y_1 y_1^\top\right) v_j$.
  Compute y_j and s_j by using v'_j.
  $x_j = \left(I - s_1 y_1 y_1^\top\right)\left(I - s_2 y_2 y_2^\top\right) \cdots \left(I - s_j y_j y_j^\top\right) e_j$.
end for

Fig. 4 Householder orthogonalization algorithm.

3.4 Householder orthogonalization algorithm

The Householder orthogonalization algorithm is one of the alternative orthogonalization algorithms. When some vectors v_j, $w_j \in \mathbb{R}^n$ satisfy $\|v_j\|_2 = \|w_j\|_2$, there exists the symmetric matrix H_j satisfying $H_j H_j^\top = H_j^\top H_j = I$, $H_j v_j = w_j$ defined by

$$H_j = I - s_j y_j y_j^\top, \tag{14}$$

where $y_j = v_j - w_j$ and $s_j = 2/\|y_j\|_2^2$. The transformation by H_j is called the Householder transformation. **Fig. 4** shows the Householder orthogonalization algorithm. The vector y_j is the vector, in which the elements from 1 to $j - 1$ are the same as the elements of v'_j and the elements from $j + 1$ to n are zero. v'_j and w_j are defined as follows:

$$\begin{aligned} v'_j &= \begin{bmatrix} v'_{j\{1\}} & \cdots & v'_{j\{j-1\}} & v'_{j\{j\}} & \cdots & v'_{j\{n\}} \end{bmatrix}^\top \\ &= H_{j-1} H_{j-2} \cdots H_2 H_1 v_j, \end{aligned} \tag{15}$$

$$w_j = \begin{bmatrix} v'_{j\{1\}} & \cdots & v'_{j\{j-1\}} & c_j & \mathbf{0} \end{bmatrix}^\top, \tag{16}$$

where,

$$c_j = -sgn\left(v'_{j\{j\}}\right) \sqrt{\sum_{i=j}^{n} {v'_{j\{i\}}}^2}. \tag{17}$$

H_j, y_j and s_j are computed using v_j as follows:

```
1: for j = 1 to m do
2:     Generate v_j in an algorithm
3:     v'_j = (I − Y_{j−1} S^T_{j−1} Y^T_{j−1}) v_j.
4:     Compute y_j and s_j by using v'_j.
5:     Eq.(24) and Eq.(25) : Update Y_j and S_j by using s_j, y_j, S_{j−1} and Y_{j−1}.
6:     q_j = (I − Y_j S_j Y^T_j) e_j.
7: end for
```

Fig. 5 Householder orthogonalization algorithm in terms of the compact WY representation.

$$H_j = I - s_j \boldsymbol{y}_j \boldsymbol{y}_j^\top \tag{18}$$

$$\boldsymbol{y}_j = \boldsymbol{v}'_j - \boldsymbol{w}_j \tag{19}$$

$$\|\boldsymbol{y}_j\|_2^2 = (v'_{j\{j\}} - c_j)^2 + \sum_{i=j+1}^{n} {v'_{j\{i\}}}^2 \tag{20}$$

$$= \sum_{i=j}^{n} {v'_{j\{i\}}}^2 - 2v'_{j\{j\}}c_j + c_j^2 \tag{21}$$

$$= 2\left(c_j^2 - v'_{j\{j\}}c_j\right). \tag{22}$$

$$s_j = \frac{2}{\|\boldsymbol{y}_j\|_2^2} = \frac{1}{c_j^2 - v'_{j\{j\}}c_j}. \tag{23}$$

The vector $\boldsymbol{e}_j$ in Fig. 4 is the j-th vector of an n-dimensional identity matrix.

The orthogonality of the vectors $\boldsymbol{x}_j$ generated by the Householder orthogonalization algorithm does not depend on the condition number of a matrix T. Therefore, the Householder orthogonalization algorithm is more stable than the MGS algorithm. On the other hand, being similar to the MGS algorithm, it is sequential algorithm that is mainly based on BLAS level-1. Its computational cost is higher than that of the MGS algorithm. Thus the Householder orthogonalization algorithm is an ineffective algorithm in parallel computing.

By combination with the compact WY representation [18], the Householder orthogonalization algorithm becomes capable of computation with BLAS level-2 [20]. Hence, in this paper, the cWY is adopted to an inverse iteration. Let $Y_1 = \boldsymbol{y}_1 \in \mathbb{R}^{n\times j}$ and $S_1 = s_1 \in \mathbb{R}^{1\times 1}$. Matrices Y_j and upper triangular matrices S_j is defined recursively as follows:

$$Y_j = \begin{bmatrix} Y_{j-1} & \boldsymbol{y}_j \end{bmatrix}, \tag{24}$$

$$S_j = \begin{bmatrix} S_{j-1} & -s_j S_{j-1} Y_{j-1}^\top \boldsymbol{y}_j \\ \boldsymbol{0} & s_j \end{bmatrix}. \tag{25}$$

In this case, the following equation holds

$$H_1 H_2 \cdots H_j = I - Y_j S_j Y_j^\top. \tag{26}$$

As shown by Eq.(26), the product of the Householder matrices $H_1 H_2 \cdots H_j$ can be rewriten in a simple block matrix form. Here $I - Y_j S_j Y_j^\top$ is called the compact WY representation of the product of the Householder matrices. **Fig. 5** shows the orthogonalization algorithm.

Fig. 6 is a code, which is based on DSTEIN in LAPACK and changed the orthogonalization process from the MGS algorithm to the cWY algorithm. In other words, the MGS algorithm (from line 4 to 15 in Fig. 3) is rewriten the cWY algorithm.In Fig. 6, the index j_c denotes the j_c-th eigenvalue of the cluster in computing the j_c-th eigenvector. This index j_c needs to compute and update S_j and Y_j. Therefore, a variable j_c should be confirmed on line 9 in Fig. 6.

```
 1: for j = 1 to n do
 2:     Generate v_j^(0) from random numbers.
 3:     k = 0
 4:     repeat
 5:         k ← k + 1.
 6:         Normalize v_j^(k−1).
 7:         Solve linear equations : (T − λ̃_j I) v_j^(k) = v_j^(k−1).
 8:         if |λ̃_j − λ̃_{j−1}| ≤ 10^{−3}||T||, then
 9:             j_c ← j − j_1.
10:             if j_c = 1 and k = 1,   then
11:                 Compute Y_1 = y_1 and S_1 = s_1 by using v_{j_1}.
12:             end if
13:             v'_{j_c+1} = (I − Y_{j_c} S^T_{j_c} Y^T_{j_c}) v_j^(k).
14:             Compute y_{j_c+1} and s_{j_c+1} by using v'_{j_c+1}.
15:             Eq.(24) and Eq.(25) : Update Y_{j_c+1} and S_{j_c+1} by using s_{j_c+1},
                y_{j_c+1}, S_{j_c} and Y_{j_c}.
16:             v_j^(k) ← (I − Y_{j_c+1} S_{j_c+1} Y^T_{j_c+1}) e_{j_c+1}.
17:         else
18:             j_1 ← j.
19:         end if
20:     until some condition is met.
21:     Normalize v_j^(k) to v_j.
22: end for
```

Fig. 6 Inverse iteration algorithm with the cWY algorithm.

Table 1 Comparison of the orthogonalization algorithms [5] [20].

algorithms	*Computation*	*Synchronization*	*Orthogonality*
CGS	almost $2m^2n$	$O(m)$	$O(\epsilon\kappa(A)^2)$
MGS	almost $2m^2n$	$O(m^2)$	$O(\epsilon\kappa(A))$
House	almost $4m^2n$	$O(m^2)$	$O(\epsilon)$
cWY	almost $4m^2n$	$O(m)$	$O(\epsilon)$

The cWY algorithm has a stable orthogonality arising from the Householder transformations, and its mathematical calculation is mainly performed by BLAS level-2 such as the product of a matrix and a vector and a rank-1 update operation.

3.5 Comparison of the orthogonalization algorithms

The cWY algorithm has a stable orthogonality arising from the Householder transformations, and its mathematical calculation is mainly performed by BLAS level-2 such as the product of a matrix and a vector and a rank-1 update operation. As a result, this orthogonalization has more stable and sophisticated orthogonality, and it is more effective for parallel computing than the MGS algorithm. Table 1 displays the differences in performance of the four orthogonalization methods, considered in the above sections. In this table, “House” denotes the Householder orthogonalization algorithm. *Computation* denotes the order of the computational cost. *Synchronization* denotes the order of the number of synchronizations. *Orthogonality* denotes the norm $\|V^\top V - I\|$, where $V = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n]$. ϵ denotes the machine epsilon and κ denotes the condition number of a matrix. These are the results obtained from [5] and [20].

On the other hand, the computational cost in the CGS algorithm is twice less than that in the cWY algorithm. Therefore, when high orthogonality is not needed, the CGS algorithm is also the suitable selection for the orthogonalization.

3.6 Adoption of BLAS

The line from 1 to 7 on each algorithm is the code in the inverse iteration algorithm without an orthogonalization algorithm. This computational cost mn is relatively smaller than that in the inverse iteration algorithm with an orthogonalization algorithm shown in Table 1. Therefore, we adopt SPEs to orthogonalization algorithms.

In the CGS algorithm, the line 10 on Fig. 2 can be computed using BLAS level-2. In the MGS algorithm, BLAS level-1 is adopted in the line 10 on Fig. 3. In the cWY algorithm, the line 13 and 16 on Fig. 6 can be performed with BLAS level-2, and the line 11 and 14 can be performed with BLAS level-1.

4. Experiments

In this section, we describe some numerical experiments performed using the CGS algorithm, the MGS algorithm, and the cWY algorithm on PowerXCell^{TM} 8i processor.

In the experiments, we use GigaAccel 180, which is a PCI Express board with PowerXCell^{TM} 8i processor. PowerXCell^{TM} 8i processor is one of Cell Broadband Engine^{TM}. The theoretical performances of a single and double precision floating-point arithmetic operation on an SPE in PowerXCell^{TM} 8i processor are 180GFLOPS and 90GFLOPS in 2.8GHz, respectively. We implement those algorithms by using Cell SDK 3.1 [2], which is developed by the IBM corporate [14]. Cell SDK 3.1 includes the parallelized BLAS for Cell Broadband Engine^{TM}. The MGS algorithm is implemented in Cell SDK 3.1.

As experimental matrices, we use three types. Type 1 is a random matrix, of which elements are set to the random number on the interval from 0 to 1. Type 2 is shown as follows:

$$\begin{bmatrix} 1 & 1 & & & \\ 1 & 1 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 1 & 1 \\ & & & 1 & 1 \end{bmatrix}. \tag{27}$$

Type 3 is the glued-Wilkinson matrix $W_g^\dagger$, which is real symmetric and has dimensions on the order of thousands. The glued-Wilkinson matrix has been used to evaluate the performance of the inverse iteration algorithms as the benchmark problems of eigenvalue decomposition [6], [8]. $W_g^\dagger$ consists of the block matrix $W_{21}^\dagger \in \mathbb{R}^{21\times 21}$ and the scalar parameter $\delta \in \mathbb{R}^{1\times 1}$ and is defined as follow:

$$W_g^\dagger = \left[\begin{array}{c|c|cc|c} W_{21}^\dagger & \delta & & & \\ \hline \delta & W_{21}^\dagger & \delta & & \\ \hline & \delta & \ddots & \ddots & \\ & & \ddots & \ddots & \delta \\ \hline & & & \delta & W_{21}^\dagger \end{array}\right], \tag{28}$$

where $W_{21}^\dagger$ is defined by

$$W_{21}^\dagger = \begin{bmatrix} 10 & 1 & & & & \\ 1 & 9 & 1 & & & \\ & 1 & \ddots & \ddots & & \\ & & \ddots & 0 & \ddots & \\ & & & \ddots & \ddots & 1 \\ & & & & 1 & 10 \end{bmatrix}, \tag{29}$$

Table 2 Experimental results

	algorithm	time[sec.]	$\|\|AV - VD\|\|_F$	$\|\|V^\top V - I\|\|_F$
type1	(dimension size is 2100.)			
	CGS	10.35	9.15×10^{-15}	2.50×10^{-14}
	MGS	7.32	9.15×10^{-15}	2.50×10^{-14}
	cWY	13.51	0.70×10^{-15}	2.61×10^{-14}
	(dimension size is 4200.)			
	CGS	60.54	1.25×10^{-14}	3.31×10^{-14}
	MGS	64.51	1.25×10^{-14}	3.32×10^{-14}
	cWY	94.27	0.067×10^{-14}	3.36×10^{-14}
	(dimension size is 6300.)			
	CGS	188.52	1.52×10^{-14}	3.49×10^{-14}
	MGS	478.53	1.53×10^{-14}	3.49×10^{-14}
	cWY	318.04	0.30×10^{-14}	4.52×10^{-14}
	(dimension size is 8400.)			
	CGS	768.40	1.82×10^{-14}	3.47×10^{-14}
	MGS	5887.12	1.81×10^{-14}	3.47×10^{-14}
	cWY	1408.27	1.01×10^{-14}	21.48×10^{-14}
type2	(dimension size is 2100.)			
	CGS	43.15	8.72×10^{-14}	1.06×10^{-13}
	MGS	263.82	8.64×10^{-14}	1.11×10^{-13}
	cWY	78.75	0.37×10^{-14}	2.56×10^{-13}
	(dimension size is 4200.)			
	CGS	247.92	1.79×10^{-13}	1.84×10^{-13}
	MGS	2392.14	1.77×10^{-13}	1.97×10^{-13}
	cWY	456.93	0.052×10^{-13}	4.96×10^{-13}
	(dimension size is 6300.)			
	CGS	754.69	2.64×10^{-13}	2.83×10^{-13}
	MGS	7864.63	2.63×10^{-13}	3.04×10^{-13}
	cWY	1394.79	0.061×10^{-13}	7.45×10^{-13}
	(dimension size is 8400.)			
	CGS	1718.53	3.51×10^{-13}	4.33×10^{-13}
	MGS	16770.71	3.48×10^{-13}	4.53×10^{-13}
	cWY	3186.58	0.078×10^{-13}	10.18×10^{-13}
type3	(dimension size is 2100.)			
	CGS	20.13	1.11×10^{-12}	1.00×10^{-13}
	MGS	28.15	1.11×10^{-12}	1.07×10^{-13}
	cWY	35.64	0.18×10^{-13}	1.06×10^{-13}
	(dimension size is 4200.)			
	CGS	89.47	1.75×10^{-12}	7.72×10^{-12}
	MGS	202.16	1.75×10^{-12}	1.53×10^{-12}
	cWY	158.18	0.25×10^{-12}	0.95×10^{-12}
	(dimension size is 6300.)			
	CGS	210.98	2.37×10^{-11}	77.29×10^{-10}
	MGS	678.24	2.51×10^{-11}	2.00×10^{-10}
	cWY	371.78	26.89×10^{-11}	2.17×10^{-10}
	(dimension size is 8400.)			
	CGS	391.50	7.93×10^{-12}	757.15×10^{-11}
	MGS	1422.99	7.94×10^{-12}	3.13×10^{-11}
	cWY	678.31	3.13×10^{-12}	3.13×10^{-11}

and δ satisfies $0 < \delta < 1$ and is also the semi-diagonal element of $W_g^\dagger$. Since $W_g^\dagger$ is real symmetric tridiagonal and its semi-diagonal elements are nonzero, all the eigenvalues of $W_g^\dagger$ are distinct and real, and they are divided into 21 clusters of close eigenvalues. When δ is small, the distance between the minimum and maximum eigenvalues in any cluster is small. In our experiments, we set $\delta = 10^{-4}$.

Table 2 shows the experimental results of the orthogonalization algorithms. Time in Table 2 is the computational time. $||AV - VD||_F$ and $||V^\top V - I||_F$ mean the frobenius norm of synchronization and orthogonalization, respectively.

In type 1, each eigenvalue is usually separated. On the other hand, eigenvalues in type 2 and 3 become cluster. Therefore, $||AV - VD||_F$ and $||V^\top V - I||_F$ are smaller than that in type 2 and 3.

In 2100 dimension size of type 1 in Table 2, the computational time in the MGS algorithm, which is implemented by the IBM corporate, is smaller than that in the other orthogonalization algorithms. However, the increasing rate of the computational time in the MGS algorithm is higher as shown in Table 2. The MGS algorithm is computed using BLAS level-1. On the other hand, the CGS algorithm and the cWY algorithm are almost computed using BLAS level-2. Therefore, in the computational time, the CGS algorithm and the cWY algorithm are better.

In Table 2, $\|AV - VD\|_F$ in the CGS algorithm is nearly equal to that in the MGS algorithm. $\|AV - VD\|_F$ of the cWY algorithm is the smallest, except the case of 6300 dimension size in type 3. The exception is likely to be caused by the order of $\boldsymbol{v}_j$. In the experiments, $\boldsymbol{v}_j$ is listed in descending order of eigenvalues, which are related to eigenvectors. Therefore, by using the cWY algorithm with suitable order of $\boldsymbol{v}_j$, accuracy of eigenvector computation can become more properly.

In type 1 and 2 of Table 2, $\|V^\top V - I\|_F$ in the CGS algorithm is nearly equal to that in the MGS algorithm. The CGS algorithm and the MGS algorithm are focused on the orthogonality of eigenvectors. On the other hand, in the cWY algorithm, accuracy of eigenvalue decomposition is given importance. Therefore, the orthogonality of eigenvectors is something lower than that in the CGS algorithm and the MGS algorithm.

In type 3 of Table 2, $\|V^\top V - I\|_F$ in the CGS algorithm is worse than that in the other orthogonalization algorithm. In $\delta = 10^{-4}$, eigenvalues in type 3 are extremely close together. Therefore, the CGS algorithm is aborted that $\boldsymbol{v}_j$ is picked out.

In summarization, the computational time in the cWY algorithm is adequate speedy. Furthermore, $\|AV - VD\|_F$ and $\|V^\top V - I\|_F$ in the cWY algorithm is sufficient accuracy. Hence, the cWY algorithm is suitable, except case that the high-orthogonality of eigenvectors is given importance.

5. Conclusions

In this paper, we validated the parallel performance of the inverse iteration algorithms with the CGS algorithm, the MGS algorithm, and the cWY algorithm on PowerXCell^{TM} 8i processor. PowerXCell^{TM} 8i processor is one of heterogeneous environments. In ExaFLOP computing, since it is critical issue to minimize electricity, heterogeneous environments are suitable. SPEs in PowerXCell^{TM} 8i processor archive the high performance of BLAS level-2 and level-3. The inverse iteration algorithms are algorithms for computing eigenvectors and need a lot of computational cost. Therefore, the algorithms should be computed with SPEs. The experimental results show that the computational time of the CGS algorithm and the cWY algorithm are shorter and $\|AV - VD\|_F$ and $\|V^\top V - I\|_F$ of the cWY algorithm are sufficiently small.

In a future work, the inverse iteration algorithms should be compared on General-purpose computing on graphics processing units (GPGPU).

References

[1] Basic Linear Algebra Subprograms, Netlib (Online), available from ⟨http://netlib.org/blas/index.html⟩ (accessed 2012-06-11).
[2] Cell SDK Installation Guide, Gitbrew (Online), available from ⟨http://git.gitbrew.org/openclit/documentation/SDK-Installation_Guide_v3.1.pdf⟩ (accessed 2012-03-31).
[3] Cuppen, J. J. M.: A divide and conquer method for the symmetric tridiagonal eigenproblem, *Numerische Mathematik*, Vol. 36, pp. 177–195 (1981).
[4] Demmel, J.: *Applied Numerical Linear Algebra*, SIAM, Philadelphia America (1997).
[5] Demmel, J., Grigori, L., Hoemmen, M., and Langou, J.: Communication-optimal parallel and sequential QR and LU factorizations, *LAPACK Working Notes*, No.204 (2008).
[6] Demmel, J., Marques, O., Parlett, B., and Vömel, C.: Performance and accuracy of LAPACK's symmetric tridiagonal eigensolvers, *SIAM J. Sci. Comput.*, Vol. 30, No. 3, pp. 1508–1526 (2008).
[7] Dhillon, I.: A new O(n^2) algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem, Ph.D. thesis, Computer Science Division, University of California, Berkeley, California, available as UC Berkeley Technical Report UCB//CSD-97-971 (1997).
[8] Dhillon, I., Parlett, B., and Vömel, C.: Glued matrices and the MRRR algorithm, *SIAM J. Sci. Comput.*, Vol. 27, No. 2, pp. 496–510 (2005).
[9] Golub, G. and Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix, *SIAM J. Numeri. Anal.*, Vol. 2, pp. 205–224 (1965).
[10] Golub, G., and van Loan, V.: *Matrix Computations*, Johns Hopkins University Press, Maryland, America (1996).
[11] Greenbaum, A., Rozloznik, M., and Strakos, Z.: Numerical Behaviour of The Modified Gram-Schmidt GMRES Implementation, *BIT*, No. 69, pp.303–318 (1996).
[12] Gu, M., and Eisenstat, S. C.: *A stable algorithm for the rank-1 modification of the symmetric eigenproblem*, Computer Science Department Report YALEU/DCS/RR-916, Yale University, New Haven, CT (1992).
[13] Gu, M., and Eisenstat, S. C.: A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem, *SIAM Journal on Matrix Analysis and Applications*, Vol. 16, No. 1, pp.172–191 (1995).
[14] IBM United States (Online), available from ⟨http://www.ibm.com/⟩ (accessed 2012-06-11).
[15] Ishigami, H., Kimura, K., and Nakamura, Y.: Implementation and performance evaluation of new inverse iteration algorithm with Householder transformation in terms of the compact WY representation *Proc. PDPTA2011*, Vol. II, pp. 775-781 (2011).
[16] LAPACK-Linear Algebra PACKage, Netlib (Online), available from ⟨http://www.netlib.org/lapack/⟩ (accessed 2012-06-11).
[17] Peters, G., and Wilkinson, J.: *The calculation of specified eigenvectors by inverse iteration*, contribution II/18, in Linear Algebra, Handbook for Automatic Computation, Vol. II, Springer-Verlag, Berlin, pp. 418-439 (1971).
[18] Schreiber, R., and van Loan, C.: A storage-efficient WY representation for products of Householder transformations, *SIAM J. Sci. Stat. Comput.*, Vol. 10, No. 1, pp. 53-57 (1988).
[19] Walker, H.: Implementation of the GMRES method using Householder transformations, *SIAM J. Sci. Stat. Comput.*, Vol. 9, No. 1, pp. 152–163 (1988).
[20] Yamamoto, Y., and Hirota, Y.: A parallel algorithm for incremental orthogonalization based on the compact WY representation, *JSIAM Letters*, Vol. 3, pp. 89–92 (2011).

